SCons

API Documentation

September 18, 2017

Contents

1 Package SCons
  1.1 Modules ........................................... 2
  1.2 Variables ........................................ 4

2 Module SCons.Action
  2.1 Functions ........................................ 6
  2.2 Variables ........................................ 6
  2.3 Class ActionBase .................................. 6
    2.3.1 Methods ...................................... 7
    2.3.2 Properties .................................... 7
  2.4 Class CommandAction ............................... 8
    2.4.1 Methods ...................................... 8
    2.4.2 Properties .................................... 9
  2.5 Class CommandGeneratorAction ..................... 9
    2.5.1 Methods ...................................... 10
    2.5.2 Properties .................................... 11
  2.6 Class LazyAction .................................. 11
    2.6.1 Methods ...................................... 12
    2.6.2 Properties .................................... 13
  2.7 Class FunctionAction .............................. 13
    2.7.1 Methods ...................................... 13
    2.7.2 Properties .................................... 14
  2.8 Class ListAction .................................. 14
    2.8.1 Methods ...................................... 15
    2.8.2 Properties .................................... 16
  2.9 Class ActionCaller ................................ 16
    2.9.1 Methods ...................................... 16
    2.9.2 Properties .................................... 17
  2.10 Class ActionFactory .............................. 17
    2.10.1 Methods ...................................... 17
    2.10.2 Properties .................................... 18

3 Module SCons.Builder .................................. 19
  3.1 Functions ........................................ 20
  3.2 Variables ........................................ 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Class DictCmdGenerator</td>
<td>21</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Methods</td>
<td>21</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Class Variables</td>
<td>21</td>
</tr>
<tr>
<td>3.4</td>
<td>Class CallableSelector</td>
<td>22</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Methods</td>
<td>22</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Class Variables</td>
<td>22</td>
</tr>
<tr>
<td>3.5</td>
<td>Class DictEmitter</td>
<td>23</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Methods</td>
<td>23</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Class Variables</td>
<td>23</td>
</tr>
<tr>
<td>3.6</td>
<td>Class ListEmitter</td>
<td>24</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Methods</td>
<td>24</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Properties</td>
<td>25</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Class Variables</td>
<td>25</td>
</tr>
<tr>
<td>3.7</td>
<td>Class OverrideWarner</td>
<td>25</td>
</tr>
<tr>
<td>3.7.1</td>
<td>Methods</td>
<td>25</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Class Variables</td>
<td>25</td>
</tr>
<tr>
<td>3.8</td>
<td>Class EmitterProxy</td>
<td>26</td>
</tr>
<tr>
<td>3.8.1</td>
<td>Methods</td>
<td>26</td>
</tr>
<tr>
<td>3.8.2</td>
<td>Properties</td>
<td>26</td>
</tr>
<tr>
<td>3.9</td>
<td>Class BuilderBase</td>
<td>27</td>
</tr>
<tr>
<td>3.9.1</td>
<td>Methods</td>
<td>27</td>
</tr>
<tr>
<td>3.9.2</td>
<td>Properties</td>
<td>29</td>
</tr>
<tr>
<td>3.10</td>
<td>Class CompositeBuilder</td>
<td>29</td>
</tr>
<tr>
<td>3.10.1</td>
<td>Methods</td>
<td>30</td>
</tr>
<tr>
<td>3.10.2</td>
<td>Properties</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>Module SCons.CacheDir</td>
<td>31</td>
</tr>
<tr>
<td>4.1</td>
<td>Functions</td>
<td>31</td>
</tr>
<tr>
<td>4.2</td>
<td>Variables</td>
<td>31</td>
</tr>
<tr>
<td>4.3</td>
<td>Class CacheDir</td>
<td>31</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Methods</td>
<td>32</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Properties</td>
<td>33</td>
</tr>
<tr>
<td>5</td>
<td>Module SCons.Conftest</td>
<td>34</td>
</tr>
<tr>
<td>5.1</td>
<td>Functions</td>
<td>34</td>
</tr>
<tr>
<td>5.2</td>
<td>Variables</td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>Module SCons.Debug</td>
<td>39</td>
</tr>
<tr>
<td>6.1</td>
<td>Functions</td>
<td>39</td>
</tr>
<tr>
<td>6.2</td>
<td>Variables</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>Module SCons.Defaults</td>
<td>41</td>
</tr>
<tr>
<td>7.1</td>
<td>Functions</td>
<td>41</td>
</tr>
<tr>
<td>7.2</td>
<td>Variables</td>
<td>42</td>
</tr>
<tr>
<td>7.3</td>
<td>Class NullCmdGenerator</td>
<td>43</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Methods</td>
<td>44</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Properties</td>
<td>44</td>
</tr>
<tr>
<td>7.4</td>
<td>Class Variable_Method_Caller</td>
<td>44</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Methods</td>
<td>44</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Properties</td>
<td>45</td>
</tr>
<tr>
<td>8</td>
<td>Module SCons.Environment</td>
<td>46</td>
</tr>
</tbody>
</table>
8.1 Functions ......................................................... 46
8.2 Variables ......................................................... 47
8.3 Class MethodWrapper ........................................... 47
  8.3.1 Methods .................................................... 48
  8.3.2 Properties .................................................. 48
8.4 Class BuilderWrapper ........................................... 48
  8.4.1 Methods .................................................... 49
  8.4.2 Properties .................................................. 50
8.5 Class BuilderDict .............................................. 50
  8.5.1 Methods .................................................... 50
  8.5.2 Class Variables ............................................ 50
8.6 Class SubstitutionEnvironment ................................. 51
  8.6.1 Methods .................................................... 51
  8.6.2 Properties .................................................. 54
8.7 Class Base ...................................................... 55
  8.7.1 Methods .................................................... 55
  8.7.2 Properties .................................................. 63
8.8 Class OverrideEnvironment .................................... 63
  8.8.1 Methods .................................................... 64
  8.8.2 Properties .................................................. 66
8.9 Class Base ...................................................... 66
  8.9.1 Methods .................................................... 67
  8.9.2 Properties .................................................. 75
9 Module SCons.Errors ........................................... 76
  9.1 Functions .................................................... 76
  9.2 Variables .................................................... 76
  9.3 Class BuildError ............................................. 76
    9.3.1 Methods .................................................. 77
    9.3.2 Properties ............................................... 78
  9.4 Class InternalError .......................................... 78
    9.4.1 Methods .................................................. 78
    9.4.2 Properties ............................................... 78
  9.5 Class UserError .............................................. 79
    9.5.1 Methods .................................................. 79
    9.5.2 Properties ............................................... 79
  9.6 Class StopError .............................................. 80
    9.6.1 Methods .................................................. 80
    9.6.2 Properties ............................................... 80
  9.7 Class EnvironmentError ..................................... 80
    9.7.1 Methods .................................................. 81
    9.7.2 Properties ............................................... 81
  9.8 Class MSVCError ............................................. 81
    9.8.1 Methods .................................................. 81
    9.8.2 Properties ............................................... 82
  9.9 Class ExplicitExit .......................................... 82
    9.9.1 Methods .................................................. 82
    9.9.2 Properties ............................................... 83
10 Module SCons.Executor ......................................... 84
  10.1 Functions .................................................. 84
  10.2 Variables .................................................. 84
### Module SCons.Job

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1 Variables</td>
<td>96</td>
</tr>
<tr>
<td>11.2 Class InterruptState</td>
<td>96</td>
</tr>
<tr>
<td>11.2.1 Methods</td>
<td>96</td>
</tr>
<tr>
<td>11.2.2 Properties</td>
<td>97</td>
</tr>
<tr>
<td>11.3 Class Jobs</td>
<td>97</td>
</tr>
<tr>
<td>11.3.1 Methods</td>
<td>97</td>
</tr>
<tr>
<td>11.3.2 Properties</td>
<td>98</td>
</tr>
<tr>
<td>11.4 Class Serial</td>
<td>98</td>
</tr>
<tr>
<td>11.4.1 Methods</td>
<td>98</td>
</tr>
<tr>
<td>11.4.2 Properties</td>
<td>99</td>
</tr>
<tr>
<td>11.5 Class Worker</td>
<td>99</td>
</tr>
<tr>
<td>11.5.1 Methods</td>
<td>100</td>
</tr>
<tr>
<td>11.5.2 Properties</td>
<td>101</td>
</tr>
<tr>
<td>11.6 Class ThreadPool</td>
<td>101</td>
</tr>
<tr>
<td>11.6.1 Methods</td>
<td>101</td>
</tr>
<tr>
<td>11.6.2 Properties</td>
<td>102</td>
</tr>
<tr>
<td>11.7 Class Parallel</td>
<td>102</td>
</tr>
<tr>
<td>11.7.1 Methods</td>
<td>103</td>
</tr>
<tr>
<td>11.7.2 Properties</td>
<td>103</td>
</tr>
</tbody>
</table>

### Module SCons.Memoize

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1 Functions</td>
<td>105</td>
</tr>
<tr>
<td>12.2 Variables</td>
<td>106</td>
</tr>
<tr>
<td>12.3 Class Counter</td>
<td>106</td>
</tr>
<tr>
<td>12.3.1 Methods</td>
<td>106</td>
</tr>
<tr>
<td>12.3.2 Properties</td>
<td>107</td>
</tr>
<tr>
<td>12.4 Class CountValue</td>
<td>107</td>
</tr>
<tr>
<td>12.4.1 Methods</td>
<td>107</td>
</tr>
<tr>
<td>12.4.2 Properties</td>
<td>108</td>
</tr>
<tr>
<td>12.5 Class CountDict</td>
<td>108</td>
</tr>
<tr>
<td>12.5.1 Methods</td>
<td>108</td>
</tr>
</tbody>
</table>
12.5.2 Properties ......................................................... 109

13 Package SCons.Node .............................................. 110
  13.1 Modules .......................................................... 110
  13.2 Functions ........................................................ 110
  13.3 Variables ........................................................ 112
  13.4 Class NodeInfoBase ............................................ 113
    13.4.1 Methods ................................................... 114
    13.4.2 Properties ................................................ 114
    13.4.3 Class Variables .......................................... 114
  13.5 Class BuildInfoBase ......................................... 115
    13.5.1 Methods ................................................... 115
    13.5.2 Properties ................................................ 116
    13.5.3 Class Variables .......................................... 116
  13.6 Class Node ....................................................... 116
    13.6.1 Methods ................................................... 117
    13.6.2 Properties ................................................ 129
  13.7 Class NodeList ................................................ 130
    13.7.1 Methods ................................................... 130
    13.7.2 Properties ................................................ 131
    13.7.3 Class Variables .......................................... 131
  13.8 Class Walker ................................................... 131
    13.8.1 Methods ................................................... 132
    13.8.2 Properties ................................................ 132

14 Module SCons.Node.Alias .......................................... 133
  14.1 Variables ........................................................ 133
  14.2 Class AliasNameSpace ........................................ 133
    14.2.1 Methods ................................................... 133
    14.2.2 Class Variables .......................................... 133
  14.3 Class AliasNodeInfo .......................................... 134
    14.3.1 Methods ................................................... 134
    14.3.2 Properties ................................................ 134
    14.3.3 Class Variables .......................................... 135
  14.4 Class AliasBuildInfo ......................................... 135
    14.4.1 Methods ................................................... 135
    14.4.2 Properties ................................................ 135
    14.4.3 Class Variables .......................................... 136
  14.5 Class Alias ...................................................... 136
    14.5.1 Methods ................................................... 136
    14.5.2 Properties ................................................ 138

15 Module SCons.Node.FS ............................................. 139
  15.1 Functions ........................................................ 139
  15.2 Variables ........................................................ 140
  15.3 Class EntryProxyAttributeError ............................... 142
    15.3.1 Methods ................................................... 142
    15.3.2 Properties ................................................ 142
  15.4 Class DiskChecker ............................................. 143
    15.4.1 Methods ................................................... 143
    15.4.2 Properties ................................................ 143
  15.5 Class EntryProxy ................................................ 144
16 Module SCons.Node.Python 186
16.1 Variables ........................................... 186
## 16.2 Class ValueNodeInfo
- **Methods**
- **Properties**
- **Class Variables**

## 16.3 Class ValueBuildInfo
- **Methods**
- **Properties**
- **Class Variables**

## 16.4 Class Value
- **Methods**
- **Properties**

## 17 Module SCons.PathList
- **Functions**
- **Variables**

## 18 Module SCons.SConf
- **Functions**
- **Variables**
- **Class SConfWarning**
  - **Methods**
  - **Properties**
- **Class SConfError**
  - **Methods**
  - **Properties**
- **Class ConfigureDryRunError**
  - **Methods**
  - **Properties**
- **Class ConfigureCacheError**
  - **Methods**
  - **Properties**
- **Class SConfBuildInfo**
  - **Methods**
  - **Properties**
  - **Class Variables**
- **Class Streamer**
  - **Methods**
  - **Properties**
- **Class SConfBuildTask**
  - **Methods**
  - **Properties**
- **Class SConfBase**
  - **Methods**
  - **Properties**
- **Class CheckContext**
  - **Methods**
  - **Properties**

## 19 Module SCons.SConsign
- **Functions**
- **Variables**
- **Class SConsignEntry**
## CONTENTS

19.3.1 Methods .................................................. 210
19.3.2 Properties .................................................. 210
19.3.3 Class Variables ............................................. 210
19.4 Class Base .................................................... 211
19.4.1 Methods .................................................... 211
19.4.2 Properties .................................................. 212
19.5 Class DB ...................................................... 212
19.5.1 Methods .................................................... 212
19.5.2 Properties .................................................. 212
19.6 Class Dir ...................................................... 213
19.6.1 Methods .................................................... 213
19.6.2 Properties .................................................. 213
19.7 Class DirFile .................................................. 214
19.7.1 Methods .................................................... 214
19.7.2 Properties .................................................. 214
19.8 Class DB ...................................................... 215
19.8.1 Methods .................................................... 215
19.8.2 Properties .................................................. 215

20 Package SCons.Scanner ........................................ 216
20.1 Modules ....................................................... 216
20.2 Functions ..................................................... 216
20.3 Variables ...................................................... 216
20.4 Class FindPathDirs ........................................... 217
   20.4.1 Methods ................................................ 217
   20.4.2 Properties .............................................. 217
20.5 Class Base .................................................... 218
   20.5.1 Methods ................................................ 218
   20.5.2 Properties .............................................. 220
20.6 Class Selector ................................................ 220
   20.6.1 Methods ................................................ 223
   20.6.2 Properties .............................................. 224
20.7 Class Current ................................................ 224
   20.7.1 Methods ................................................ 224
   20.7.2 Properties .............................................. 227
20.8 Class Classic ................................................ 227
   20.8.1 Methods ................................................ 227
   20.8.2 Properties .............................................. 230
20.9 Class ClassicCPP ............................................ 230
   20.9.1 Methods ................................................ 231
   20.9.2 Properties .............................................. 231

21 Module SCons.Scanner.C ......................................... 232
21.1 Functions ..................................................... 232
21.2 Variables ...................................................... 232
21.3 Class SConsCPPScanner ...................................... 232
   21.3.1 Methods ................................................ 233
   21.3.2 Properties .............................................. 233
21.4 Class SConsCPPScannerWrapper .............................. 234
   21.4.1 Methods ................................................ 234
   21.4.2 Properties .............................................. 234
<table>
<thead>
<tr>
<th>Module</th>
<th>Functions</th>
<th>Variables</th>
<th>Class</th>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module SCons.Scanner.D</td>
<td>235</td>
<td>235</td>
<td>235</td>
<td>237</td>
<td>238</td>
</tr>
<tr>
<td>22.1 Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.2 Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.3 Class D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.3.1 Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.3.2 Properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module SCons.Scanner.Dir</td>
<td>239</td>
<td>239</td>
<td>239</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23.1 Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23.2 Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module SCons.Scanner.Fortran</td>
<td>241</td>
<td>241</td>
<td>241</td>
<td>244</td>
<td>245</td>
</tr>
<tr>
<td>24.1 Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.2 Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.3 Class F90Scanner</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.3.1 Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.3.2 Properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module SCons.Scanner.IDL</td>
<td>246</td>
<td>246</td>
<td>246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.1 Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.2 Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module SCons.Scanner.LaTeX</td>
<td>247</td>
<td>247</td>
<td>247</td>
<td>248</td>
<td>248</td>
</tr>
<tr>
<td>26.1 Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.2 Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.3 Class FindENVPathDirs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.3.1 Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.3.2 Properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.4 Class LaTeX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.4.1 Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.4.2 Properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.4.3 Class Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module SCons.Scanner.Prog</td>
<td>254</td>
<td>254</td>
<td>254</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27.1 Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27.2 Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module SCons.Scanner.RC</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28.1 Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28.2 Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module SCons.Scanner.SWIG</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29.1 Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29.2 Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package SCons.Script</td>
<td>257</td>
<td>257</td>
<td>257</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>30.1 Modules</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.2 Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.3 Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.4 Class TargetList</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.4.1 Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.4.2 Properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.4.3 Class Variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 31 Module SCons.Script.Interactive

- **31.1 Functions** ............................................. 266
- **31.2 Variables** ............................................ 266
- **31.3 Class SConsInteractiveCmd** .......................... 266
  - **31.3.1 Methods** ............................................ 267
  - **31.3.2 Class Variables** ................................. 268

### 32 Module SCons.Script.Main

- **32.1 Functions** ............................................. 269
- **32.2 Variables** ............................................ 270
- **32.3 Class SConsPrintHelpException** .................... 271
  - **32.3.1 Methods** ............................................ 271
  - **32.3.2 Properties** ....................................... 271
- **32.4 Class Progressor** ..................................... 272
  - **32.4.1 Methods** ............................................ 272
  - **32.4.2 Properties** ....................................... 272
  - **32.4.3 Class Variables** ................................. 272
- **32.5 Class BuildTask** ..................................... 273
  - **32.5.1 Methods** ............................................ 273
  - **32.5.2 Properties** ....................................... 275
  - **32.5.3 Class Variables** ................................. 275
- **32.6 Class CleanTask** .................................... 275
  - **32.6.1 Methods** ............................................ 276
  - **32.6.2 Properties** ....................................... 277
- **32.7 Class QuestionTask** ................................ 277
  - **32.7.1 Methods** ............................................ 278
  - **32.7.2 Properties** ....................................... 279
- **32.8 Class TreePrinter** .................................. 279
  - **32.8.1 Methods** ............................................ 279
  - **32.8.2 Properties** ....................................... 279
- **32.9 Class FakeOptionParser** ............................ 280
  - **32.9.1 Methods** ............................................ 280
  - **32.9.2 Properties** ....................................... 280
  - **32.9.3 Class Variables** ................................. 280
- **32.10 Class Stats** ....................................... 280
  - **32.10.1 Methods** ............................................ 281
  - **32.10.2 Properties** ....................................... 281
- **32.11 Class CountStats** ................................ 281
  - **32.11.1 Methods** ............................................ 281
  - **32.11.2 Properties** ....................................... 282
- **32.12 Class MemStats** .................................. 282
  - **32.12.1 Methods** ............................................ 282
  - **32.12.2 Properties** ....................................... 282

### 33 Module SCons.Script.SConscript'

- **33.1 Functions** ............................................. 284
- **33.2 Variables** ............................................ 285
- **33.3 Class SConscriptReturn** ............................ 285
  - **33.3.1 Methods** ............................................ 285
  - **33.3.2 Properties** ....................................... 286
- **33.4 Class Frame** ........................................ 286
  - **33.4.1 Methods** ............................................ 286
34 Module SCons.Subst ................................................. 290
34.1 Functions ...................................................... 290
34.2 Variables ........................................................ 291
34.3 Class Literal .................................................... 292
  34.3.1 Methods ................................................... 292
  34.3.2 Properties .................................................. 293
34.4 Class SpecialAttrWrapper ................................. 293
  34.4.1 Methods ................................................... 293
  34.4.2 Properties .................................................. 294
34.5 Class CmdStringHolder ......................................... 294
  34.5.1 Methods ................................................... 294
  34.5.2 Properties .................................................. 295
  34.5.3 Class Variables ........................................... 295
34.6 Class NLWrapper ................................................ 296
  34.6.1 Methods ................................................... 296
  34.6.2 Properties .................................................. 296
34.7 Class Targets_or_Sources ....................................... 297
  34.7.1 Methods ................................................... 297
  34.7.2 Properties .................................................. 298
  34.7.3 Class Variables ........................................... 298
34.8 Class Target_or_Source ........................................ 299
  34.8.1 Methods ................................................... 299
  34.8.2 Properties .................................................. 299
34.9 Class NullNodeList ............................................. 300
  34.9.1 Methods ................................................... 300
  34.9.2 Properties .................................................. 300

35 Module SCons.Taskmaster ...................................... 301
35.1 Functions ...................................................... 301
35.2 Variables ........................................................ 301
35.3 Class Stats ..................................................... 302
  35.3.1 Methods ................................................... 302
  35.3.2 Properties .................................................. 302
35.4 Class Task ....................................................... 303
  35.4.1 Methods ................................................... 303
  35.4.2 Properties .................................................. 307
35.5 Class AlwaysTask ............................................... 307
  35.5.1 Methods ................................................... 308
  35.5.2 Properties .................................................. 308
35.6 Class OutOfDateTask ........................................... 308
  35.6.1 Methods ................................................... 309
  35.6.2 Properties .................................................. 309
35.7 Class Taskmaster ................................................ 309
  35.7.1 Methods ................................................... 309
36 Module SCons.Util ........................................ 312
  36.1 Functions ........................................ 312
  36.2 Variables ........................................ 319
  36.3 Class NodeList .................................... 320
      36.3.1 Methods .................................... 320
      36.3.2 Properties .................................. 321
      36.3.3 Class Variables .............................. 322
  36.4 Class DisplayEngine ................................. 322
      36.4.1 Methods .................................... 322
      36.4.2 Properties .................................. 322
      36.4.3 Class Variables .............................. 322
  36.5 Class Proxy ......................................... 323
      36.5.1 Methods .................................... 323
      36.5.2 Properties .................................. 324
  36.6 Class Delegate ...................................... 324
      36.6.1 Methods .................................... 324
      36.6.2 Properties .................................. 325
  36.7 Class _NoError ..................................... 325
      36.7.1 Methods .................................... 325
      36.7.2 Properties .................................. 325
  36.8 Class PlainWindowsError ............................. 326
      36.8.1 Methods .................................... 326
      36.8.2 Properties .................................. 326
  36.9 Class PlainWindowsError ............................. 327
      36.9.1 Methods .................................... 327
      36.9.2 Properties .................................. 327
  36.10 Class CLVar ........................................ 328
      36.10.1 Methods .................................... 328
      36.10.2 Properties .................................. 329
      36.10.3 Class Variables .............................. 329
  36.11 Class OrderedDict ................................. 329
      36.11.1 Methods .................................... 330
      36.11.2 Class Variables .............................. 331
  36.12 Class Selector ..................................... 331
      36.12.1 Methods .................................... 331
      36.12.2 Class Variables .............................. 331
  36.13 Class LogicalLines ................................. 332
      36.13.1 Methods .................................... 332
      36.13.2 Properties .................................. 332
  36.14 Class UniqueList ................................. 333
      36.14.1 Methods .................................... 333
      36.14.2 Properties .................................. 336
      36.14.3 Class Variables .............................. 336
  36.15 Class Unbuffered .................................. 336
      36.15.1 Methods .................................... 337
      36.15.2 Properties .................................. 337
  36.16 Class Null ......................................... 337
      36.16.1 Methods .................................... 337
      36.16.2 Properties .................................. 338
  36.17 Class NullSeq ...................................... 339
43.11 Class DuplicateEnvironmentWarning .............................................. 359
  43.11.1 Methods ................................................................. 359
  43.11.2 Properties ............................................................. 359
43.12 Class FutureReservedVariableWarning .......................................... 360
  43.12.1 Methods ................................................................. 360
  43.12.2 Properties ............................................................. 360
43.13 Class LinkWarning ............................................................. 361
  43.13.1 Methods ................................................................. 361
  43.13.2 Properties ............................................................. 361
43.14 Class MisleadingKeywordsWarning .............................................. 362
  43.14.1 Methods ................................................................. 362
  43.14.2 Properties ............................................................. 362
43.15 Class MissingSConscriptWarning ............................................... 363
  43.15.1 Methods ................................................................. 363
  43.15.2 Properties ............................................................. 363
43.16 Class NoMD5ModuleWarning ..................................................... 364
  43.16.1 Methods ................................................................. 364
  43.16.2 Properties ............................................................. 364
43.17 Class NoMetaclassSupportWarning ............................................. 365
  43.17.1 Methods ................................................................. 365
  43.17.2 Properties ............................................................. 365
43.18 Class NoObjectCountWarning .................................................... 366
  43.18.1 Methods ................................................................. 366
  43.18.2 Properties ............................................................. 366
43.19 Class NoParallelSupportWarning ................................................ 367
  43.19.1 Methods ................................................................. 367
  43.19.2 Properties ............................................................. 367
43.20 Class ReservedVariableWarning .................................................. 368
  43.20.1 Methods ................................................................. 368
  43.20.2 Properties ............................................................. 368
43.21 Class StackSizeWarning .......................................................... 369
  43.21.1 Methods ................................................................. 369
  43.21.2 Properties ............................................................. 369
43.22 Class VisualCMissingWarning ...................................................... 370
  43.22.1 Methods ................................................................. 370
  43.22.2 Properties ............................................................. 370
43.23 Class VirtualVersionMismatch ................................................... 371
  43.23.1 Methods ................................................................. 371
  43.23.2 Properties ............................................................. 371
43.24 Class VisualStudioMissingWarning .......................................... 372
  43.24.1 Methods ................................................................. 372
  43.24.2 Properties ............................................................. 372
43.25 Class FortranCxxMixWarning ..................................................... 373
  43.25.1 Methods ................................................................. 373
  43.25.2 Properties ............................................................. 373
43.26 Class FutureDeprecatedWarning ................................................. 374
  43.26.1 Methods ................................................................. 374
  43.26.2 Properties ............................................................. 374
43.27 Class DeprecatedWarning .......................................................... 375
  43.27.1 Methods ................................................................. 375
  43.27.2 Properties ............................................................. 375
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>43.28</td>
<td>Class MandatoryDeprecatedWarning</td>
<td>376</td>
</tr>
<tr>
<td>43.28.1</td>
<td>Methods</td>
<td>376</td>
</tr>
<tr>
<td>43.28.2</td>
<td>Properties</td>
<td>376</td>
</tr>
<tr>
<td>43.29</td>
<td>Class PythonVersionWarning</td>
<td>377</td>
</tr>
<tr>
<td>43.29.1</td>
<td>Methods</td>
<td>377</td>
</tr>
<tr>
<td>43.29.2</td>
<td>Properties</td>
<td>377</td>
</tr>
<tr>
<td>43.30</td>
<td>Class DeprecatedSourceCodeWarning</td>
<td>378</td>
</tr>
<tr>
<td>43.30.1</td>
<td>Methods</td>
<td>378</td>
</tr>
<tr>
<td>43.30.2</td>
<td>Properties</td>
<td>378</td>
</tr>
<tr>
<td>43.31</td>
<td>Class DeprecatedBuildDirWarning</td>
<td>379</td>
</tr>
<tr>
<td>43.31.1</td>
<td>Methods</td>
<td>379</td>
</tr>
<tr>
<td>43.31.2</td>
<td>Properties</td>
<td>379</td>
</tr>
<tr>
<td>43.32</td>
<td>Class TaskmasterNeedsExecuteWarning</td>
<td>380</td>
</tr>
<tr>
<td>43.32.1</td>
<td>Methods</td>
<td>380</td>
</tr>
<tr>
<td>43.32.2</td>
<td>Properties</td>
<td>380</td>
</tr>
<tr>
<td>43.33</td>
<td>Class DeprecatedCopyWarning</td>
<td>381</td>
</tr>
<tr>
<td>43.33.1</td>
<td>Methods</td>
<td>381</td>
</tr>
<tr>
<td>43.33.2</td>
<td>Properties</td>
<td>381</td>
</tr>
<tr>
<td>43.34</td>
<td>Class DeprecatedOptionsWarning</td>
<td>382</td>
</tr>
<tr>
<td>43.34.1</td>
<td>Methods</td>
<td>382</td>
</tr>
<tr>
<td>43.34.2</td>
<td>Properties</td>
<td>382</td>
</tr>
<tr>
<td>43.35</td>
<td>Class DeprecatedSourceSignaturesWarning</td>
<td>383</td>
</tr>
<tr>
<td>43.35.1</td>
<td>Methods</td>
<td>383</td>
</tr>
<tr>
<td>43.35.2</td>
<td>Properties</td>
<td>383</td>
</tr>
<tr>
<td>43.36</td>
<td>Class DeprecatedTargetSignaturesWarning</td>
<td>384</td>
</tr>
<tr>
<td>43.36.1</td>
<td>Methods</td>
<td>384</td>
</tr>
<tr>
<td>43.36.2</td>
<td>Properties</td>
<td>384</td>
</tr>
<tr>
<td>43.37</td>
<td>Class DeprecatedDebugOptionsWarning</td>
<td>385</td>
</tr>
<tr>
<td>43.37.1</td>
<td>Methods</td>
<td>385</td>
</tr>
<tr>
<td>43.37.2</td>
<td>Properties</td>
<td>385</td>
</tr>
<tr>
<td>43.38</td>
<td>Class DeprecatedSigModuleWarning</td>
<td>386</td>
</tr>
<tr>
<td>43.38.1</td>
<td>Methods</td>
<td>386</td>
</tr>
<tr>
<td>43.38.2</td>
<td>Properties</td>
<td>386</td>
</tr>
<tr>
<td>43.39</td>
<td>Class DeprecatedBuilderKeywordsWarning</td>
<td>387</td>
</tr>
<tr>
<td>43.39.1</td>
<td>Methods</td>
<td>387</td>
</tr>
<tr>
<td>43.39.2</td>
<td>Properties</td>
<td>387</td>
</tr>
</tbody>
</table>

### 44 Module SCons.cpp

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.1</td>
<td>Functions</td>
<td>388</td>
</tr>
<tr>
<td>44.2</td>
<td>Variables</td>
<td>388</td>
</tr>
<tr>
<td>44.3</td>
<td>Class FunctionEvaluator</td>
<td>389</td>
</tr>
<tr>
<td>44.3.1</td>
<td>Methods</td>
<td>389</td>
</tr>
<tr>
<td>44.3.2</td>
<td>Properties</td>
<td>389</td>
</tr>
<tr>
<td>44.4</td>
<td>Class PreProcessor</td>
<td>389</td>
</tr>
<tr>
<td>44.4.1</td>
<td>Methods</td>
<td>390</td>
</tr>
<tr>
<td>44.4.2</td>
<td>Properties</td>
<td>394</td>
</tr>
<tr>
<td>44.5</td>
<td>Class DumbPreProcessor</td>
<td>394</td>
</tr>
<tr>
<td>44.5.1</td>
<td>Methods</td>
<td>394</td>
</tr>
<tr>
<td>44.5.2</td>
<td>Properties</td>
<td>394</td>
</tr>
</tbody>
</table>

### 45 Module SCons.dblite

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>45.1</td>
<td>Functions</td>
<td>396</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>45.2 Variables</td>
<td>396</td>
<td></td>
</tr>
<tr>
<td>45.3 Class dblite</td>
<td>396</td>
<td></td>
</tr>
<tr>
<td>45.3.1 Methods</td>
<td>397</td>
<td></td>
</tr>
<tr>
<td>45.3.2 Properties</td>
<td>397</td>
<td></td>
</tr>
<tr>
<td>46 Module SCons.exitfuncs</td>
<td>398</td>
<td></td>
</tr>
<tr>
<td>46.1 Functions</td>
<td>398</td>
<td></td>
</tr>
<tr>
<td>46.2 Variables</td>
<td>398</td>
<td></td>
</tr>
</tbody>
</table>
1 Package SCons

SCons

The main package for the SCons software construction utility. **Version:** 3.0.0

**Date:** 2017/09/18 12:59:24

1.1 Modules

- **Action:** SCons.Action
  
  *(Section 2, p. 5)*

- **Builder:** SCons.Builder
  
  *(Section 3, p. 19)*

- **CacheDir:** CacheDir support
  
  *(Section 4, p. 31)*

- **Conftest:** SCons.Conftest
  
  *(Section 5, p. 34)*

- **Debug:** SCons.Debug
  
  *(Section 6, p. 39)*

- **Defaults:** SCons.Defaults
  
  *(Section 7, p. 41)*

- **Environment:** SCons.Environment
  
  *(Section 8, p. 46)*

- **Errors:** SCons.Errors
  
  *(Section 9, p. 76)*

- **Executor:** SCons.Executor
  
  *(Section 10, p. 84)*

- **Job:** SCons.Job
  
  *(Section 11, p. 96)*

- **Memoize:** Memoizer
  
  *(Section 12, p. 104)*

- **Node:** SCons.Node
  
  *(Section 13, p. 110)*
  
  - **Alias:** scons.Node.Alias
    
    *(Section 14, p. 133)*
  
  - **FS:** scons.Node.FS
    
    *(Section 15, p. 139)*
  
  - **Python:** scons.Node.Python
    
    *(Section 16, p. 186)*

- **PathList:** SCons.PathList
  
  *(Section 17, p. 192)*

- **SConf:** SCons.SConf
  
  *(Section 18, p. 193)*

- **SConsign:** SCons.SConsign
  
  *(Section 19, p. 209)*

- **Scanner:** SCons.Scanner
  
  *(Section 20, p. 216)*
  
  - **C:** SCons.Scanner.C
    
    *(Section 21, p. 232)*
  
  - **D:** SCons.Scanner.D
Modules

- **Dir** *(Section 23, p. 239)*
- **Fortran**: SCons.Scanner.Fortran *(Section 24, p. 241)*
- **IDL**: SCons.Scanner.IDL *(Section 25, p. 246)*
- **LaTeX**: SCons.Scanner.LaTeX *(Section 26, p. 247)*
- **Prog** *(Section 27, p. 254)*
- **RC**: SCons.Scanner.RC *(Section 28, p. 255)*
- **SWIG**: SCons.Scanner.SWIG *(Section 29, p. 256)*

- **Script**: SCons.Script *(Section 30, p. 257)*
  - **Interactive**: SCons interactive mode *(Section 31, p. 266)*
  - **Main**: SCons.Script *(Section 32, p. 269)*
  - **SConscript**: SCons.Script.SConscript *(Section 33, p. 284)*

- **Subst**: SCons.Subst *(Section 34, p. 290)*

- **Taskmaster**: This module contains the primary interface(s) between a wrapping user interface and the SCons build engine. There are two key classes here:
  *(Section 35, p. 301)*

- **Util**: SCons.Util *(Section 36, p. 312)*

- **Variables**: engine.SCons.Variables *(Section 37, p. 340)*
  - **BoolVariable** *(Section ??, p. ??)*
  - **BoolVariable’**: engine.SCons.Variables.BoolVariable *(Section 38, p. 344)*
  - **EnumVariable** *(Section ??, p. ??)*
  - **EnumVariable’**: engine.SCons.Variables.EnumVariable *(Section 39, p. 345)*
  - **ListVariable** *(Section ??, p. ??)*
  - **ListVariable’**: engine.SCons.Variables.ListVariable *(Section 40, p. 346)*
  - **PackageVariable** *(Section ??, p. ??)*
  - **PackageVariable’**: engine.SCons.Variables.PackageVariable *(Section 41, p. 347)*
  - **PathVariable** *(Section ??, p. ??)*
  - **PathVariable’**: SCons.Variables.PathVariable *(Section 42, p. 348)*

- **Warnings**: SCons.Warnings *(Section 43, p. 350)*

- **cpp**: SCons C Pre-Processor module *(Section 44, p. 388)*

- **dblite** *(Section 45, p. 396)*

- **exitfuncs**: SCons.exitfuncs
(Section 46, p. 398)

1.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>build</strong></td>
<td>Value: 'rel_3.0.0:4395:8972f6a2f699'</td>
</tr>
<tr>
<td><strong>buildsys</strong></td>
<td>Value: 'ubuntu-16'</td>
</tr>
<tr>
<td><strong>developer</strong></td>
<td>Value: 'bdbaddog'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>
| __revision__    | Value: 'src/engine/SCons/__init__.py' 
rel_3.0.0:4395:8972f6a2f699...
2 Module SCons.Action

SCons.Action

This encapsulates information about executing any sort of action that can build one or more target Nodes (typically files) from one or more source Nodes (also typically files) given a specific Environment.

The base class here is ActionBase. The base class supplies just a few OO utility methods and some generic methods for displaying information about an Action in response to the various commands that control printing.

A second-level base class is _ActionAction. This extends ActionBase by providing the methods that can be used to show and perform an action. True Action objects will subclass _ActionAction; Action factory class objects will subclass ActionBase.

The heavy lifting is handled by subclasses for the different types of actions we might execute:

   CommandAction CommandGeneratorAction FunctionAction ListAction

The subclasses supply the following public interface methods used by other modules:

   __call__() THE public interface, "calling" an Action object executes the command or Python function. This also takes care of printing a pre-substitution command for debugging purposes.

   get_contents() Fetches the 'contents' of an Action for signature calculation plus the varlist. This is what gets MD5 checksummed to decide if a target needs to be rebuilt because its action changed.

   genstring() Returns a string representation of the Action without command substitution, but allows a CommandGeneratorAction to generate the right action based on the specified target, source and env. This is used by the Signature subsystem (through the Executor) to obtain an (imprecise) representation of the Action operation for informative purposes.

Subclasses also supply the following methods for internal use within this module:

   __str__() Returns a string approximation of the Action; no variable substitution is performed.

   execute() The internal method that really, truly, actually handles the execution of a command or Python function. This is used so that the __call__() methods can take care of displaying any pre-substitution representations, and then execute an action without worrying about the specific Actions involved.

   get_presig() Fetches the 'contents' of a subclass for signature calculation. The varlist is added to this to produce the Action's contents. TODO(?): Change this to always return ascii/bytes and not unicode (or py3 strings)

   strfunction() Returns a substituted string representation of the Action. This is used by the _ActionAction.show() command to display the command/function that will be executed to generate the target(s).

There is a related independent ActionCaller class that looks like a regular Action, and which serves as a wrapper for arbitrary functions that we want to let the user specify the arguments to now, but actually execute later (when an out-of-date check determines that it's needed to be executed, for example). Objects
of this class are returned by an ActionFactory class that provides a \_\_call\_\_(\) method as a convenient way for wrapping up the functions.

### 2.1 Functions

**rfile(n)**

**default\_exitstatfunc(s)**

**Action(act, *args, **kw)**

A factory for action objects.

**get\_default\_ENV(env)**

A fiddlin’ little function that has an ‘import SCons.Environment’ which can’t be moved to the top level without creating an import loop. Since this import creates a local variable named ‘SCons’, it blocks access to the global variable, so we move it here to prevent complaints about local variables being used uninitialized.

### 2.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__revision_</td>
<td>Value: ‘src/engine/SCons/Action.py rel_3.0.0:4395:8972f6a2f699 2...</td>
</tr>
<tr>
<td>print_actions</td>
<td>Value: 1</td>
</tr>
<tr>
<td>execute_actions</td>
<td>Value: 1</td>
</tr>
<tr>
<td>print_actions_presub</td>
<td>Value: 0</td>
</tr>
<tr>
<td>ACTION_SIGNATURE_PICKLE_PROTOCOL</td>
<td>Value: 1</td>
</tr>
<tr>
<td>strip_quotes</td>
<td>Value: re.compile(r’”([^\’”]*\’”)[\’”]’$)</td>
</tr>
<tr>
<td>default_ENV</td>
<td>Value: None</td>
</tr>
<tr>
<td>__package_</td>
<td>Value: ‘SCons’</td>
</tr>
</tbody>
</table>

### 2.3 Class ActionBase

object \_\_SCons.Action.ActionBase\_


Base class for all types of action objects that can be held by other objects (Builders, Executors, etc.) This provides the common methods for manipulating and combining those actions.
### 2.3.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__eq__(self, other)</code></td>
<td></td>
</tr>
<tr>
<td><code>no_batch_key(self, env, target, source)</code></td>
<td></td>
</tr>
<tr>
<td><code>batch_key(self, env, target, source)</code></td>
<td></td>
</tr>
<tr>
<td><code>genstring(self, target, source, env)</code></td>
<td></td>
</tr>
<tr>
<td><code>get__contents(self, target, source, env)</code></td>
<td></td>
</tr>
<tr>
<td><code>__add__(self, other)</code></td>
<td></td>
</tr>
<tr>
<td><code>__radd__(self, other)</code></td>
<td></td>
</tr>
<tr>
<td><code>presub__lines(self, env)</code></td>
<td></td>
</tr>
</tbody>
</table>
| `get__varlist(self, target, source, env, executor=None)` | |}
| `get__targets(self, env, executor)` | Returns the type of targets ($TARGETS, $CHANGED_TARGETS) used by this action. |

Inherited from `object`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>delattr()</code>, <code>format()</code>, <code>getattr()</code>, <code>hash()</code>, <code>init()</code>, <code>new()</code>, <code>reduce()</code>, <code>reduce_ex()</code>, <code>repr()</code>, <code>setattr()</code>, <code>sizeof()</code>, <code>str()</code>, <code>subclasshook()</code></td>
<td></td>
</tr>
</tbody>
</table>

### 2.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Inherited from object</code></td>
<td></td>
</tr>
<tr>
<td><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>
2.4 Class CommandAction

object

SCons.Action.ActionBase

SCons.Action._ActionAction

SCons.Action.CommandAction

Known Subclasses: SCons.Action.LazyAction

Class for command-execution actions.

2.4.1 Methods

```python
__init__(self, cmd, **kw)
```

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides:

object.__init__ extit(inherited documentation)

```python
__str__(self)
```

str(x) Overrides: object.__str__ extit(inherited documentation)

```python
process(self, target, source, env, executor=None)
```

```python
strfunction(self, target, source, env, executor=None)
```

```python
execute(self, target, source, env, executor=None)
```

Execute a command action.

This will handle lists of commands as well as individual commands, because construction variable substitution may turn a single "command" into a list. This means that this class can actually handle lists of commands, even though that’s not how we use it externally.
**get_presig**(*self, target, source, env, executor=None*)

Return the signature contents of this action’s command line.

This strips $(-$) and everything in between the string, since those parts don’t affect signatures.

**get_implicit_deps**(*self, target, source, env, executor=None*)

*Inherited from SCons.Action._ActionAction*

__call__(), print_cmd_line()

*Inherited from SCons.Action.ActionBase (Section 2.3)*

__add__(), __eq__(), __radd__(), batch_key(), genstring(), get_contents(),
get_targets(), get_varlist(), no_batch_key(), presub_lines()

*Inherited from object*

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()

### 2.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 2.5 Class CommandGeneratorAction

object

SCons.Action.ActionBase

SCons.Action.CommandGeneratorAction

**Known Subclasses:** SCons.Action.LazyAction

Class for command-generator actions.
2.5.1 Methods

```python
__init__(self, generator, kw)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__

__str__(self)

str(x) Overrides: object.__str__

batch_key(self, env, target, source)

Overrides: SCons.Action.ActionBase.batch_key

genstring(self, target, source, env, executor=None)


get_presig(self, target, source, env, executor=None)

Return the signature contents of this action’s command line.
This strips $(-$) and everything in between the string, since those parts don’t affect signatures.

get_implicit_deps(self, target, source, env, executor=None)

get_varlist(self, target, source, env, executor=None)


get_targets(self, env, executor)

Returns the type of targets ($TARGETS, $CHANGED_TARGETS) used by this action. Overrides: SCons.Action.ActionBase.get_targets
```
Inherited from `SCons.Action.ActionBase` (Section 2.3)

```
__add__(), __eq__(), __radd__(), get_contents(), no_batch_key(), presub_lines()
```

Inherited from `object`

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()  
```

### 2.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>

#### 2.6 Class LazyAction

A LazyAction is a kind of hybrid generator and command action for strings of the form "$VAR". These strings normally expand to other strings (think "$CCCOM" to "$CC -c -o $TARGET $SOURCE"), but we also want to be able to replace them with functions in the construction environment. Consequently, we want lazy evaluation and creation of an Action in the case of the function, but that’s overkill in the more normal case of expansion to other strings.

So we do this with a subclass that’s both a generator and a command action. The overridden methods all do a quick check of the construction variable, and if it’s a string we just call the corresponding CommandAction method to do the heavy lifting. If not, then we call the same-
named CommandGeneratorAction method. The CommandGeneratorAction methods work by using the overridden _generate() method, that is, our own way of handling "generation" of an action based on what’s in the construction variable.

2.6.1 Methods

```python
__init__(self, var, kw)
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ extit(inherited documentation)
```

```python
get__parent__class(self, env)
```

```python
__call__(self, target, source, env, *args, **kw)
Overrides: SCons.Action._ActionAction.__call__
```

```python
get__presig(self, target, source, env)
Return the signature contents of this action’s command line.
This strips $(-$) and everything in between the string, since those parts don’t affect signatures.  Overrides: SCons.Action.CommandAction.get__presig extit(inherited documentation)
```

```python
get__varlist(self, target, source, env, executor=None)
```

Inherited from SCons.Action.CommandGeneratorAction(Section 2.5)

```python
__str__(), batch_key(), genstring(), get_implicit_deps(), get_targets()
```

Inherited from SCons.Action.CommandAction(Section 2.4)

```python
execute(), process(), strfunction()
```

Inherited from SCons.Action._ActionAction

```python
print_cmd_line()
```

Inherited from SCons.Action.ActionBase(Section 2.3)

```python
__add__(), __eq__(), __radd__(), get_contents(), no_batch_key(), presub_lines()
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
```
Class FunctionAction

Module SCons.Action

```python
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
```

2.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

2.7 Class FunctionAction

object

SCons.Action.ActionBase

SCons.Action._ActionAction

SCons.Action.FunctionAction

Class for Python function actions.

2.7.1 Methods

```python
__init__(self, execfunction, kw)
```

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)

```python
function_name(self)
```

```python
strfunction(self, target, source, env, executor=None)
```

```python
__str__(self)
```

str(x)  Overrides: object.__str__ extit(inherited documentation)

```python
execute(self, target, source, env, executor=None)
```
get_presig(self, target, source, env)

Return the signature contents of this callable action.

get_implicit_deps(self, target, source, env)

Inherited from SCons.Action._ActionAction

__call__(), print_cmd_line()

Inherited from SCons.Action.ActionBase(Section 2.3)

__add__(), __eq__(), __radd__(), batch_key(), genstring(), get_contents(), get_targets(), get_varlist(), no_batch_key(), presub_lines()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

2.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

2.8 Class ListAction

object

SCons.Action.ActionBase

SCons.Action.ListAction

Class for lists of other actions.
2.8.1 Methods

```python
__init__(self, actionlist)

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)
```

```python
genstring(self, target, source, env)

```

```python
__str__(self)

str(x) Overrides: object.__str__ extit(inherited documentation)
```

```python
presub_lines(self, env)

Overrides: SCons.Action.ActionBase.presub_lines
```

```python
get_presig(self, target, source, env)

Return the signature contents of this action list.
Simple concatenation of the signatures of the elements.
```

```python
__call__(self, target, source, env, exitstatfunc=<class
'SCons.Action._null'>, presub=<class 'SCons.Action._null'>,
show=<class 'SCons.Action._null'>, execute=<class
'SCons.Action._null'>, chdir=<class 'SCons.Action._null'>,
executor=None)
```

```python
get_implicit_deps(self, target, source, env)
```

```python
get_varlist(self, target, source, env, executor=None)

```

*Inherited from SCons.Action.ActionBase (Section 2.3)*

```python
__add__(), __eq__(), __radd__(), batch_key(), get_contents(), get_targets(),
no_batch_key()
```

*Inherited from object*

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
```
Class ActionCaller

Module SCons.Action

```python
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
```

### 2.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 2.9 Class ActionCaller

object

SCons.Action.ActionCaller

A class for delaying calling an Action function with specific (positional and keyword) arguments until the Action is actually executed.

This class looks to the rest of the world like a normal Action object, but what it’s really doing is hanging on to the arguments until we have a target, source and env to use for the expansion.

### 2.9.1 Methods

```python
__init__(self, parent, args, kw)
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)

get_contents(self, target, source, env)

subst(self, s, target, source, env)

subst_args(self, target, source, env)

subst_kw(self, target, source, env)

___call__(self, target, source, env, executor=None)

strfunction(self, target, source, env)
```
2.10 Class ActionFactory

object ▼

SCons.Action.ActionFactory

A factory class that will wrap up an arbitrary function as an SCons-executable Action object. The real heavy lifting here is done by the ActionCaller class. We just collect the (positional and keyword) arguments that we’re called with and give them to the ActionCaller object we create, so it can hang onto them until it needs them.

2.10.1 Methods

__init__(self, actfunc, strfunc, convert=<_builtin__>.function object)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides:
object.__init__ (inherited documentation)

__call__(self, *args, **kw)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
### 2.10.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
3 Module SCons.Builder

SCons.Builder

Builder object subsystem.

A Builder object is a callable that encapsulates information about how to execute actions to create a target Node (file) from source Nodes (files), and how to create those dependencies for tracking.

The main entry point here is the Builder() factory method. This provides a procedural interface that creates the right underlying Builder object based on the keyword arguments supplied and the types of the arguments.

The goal is for this external interface to be simple enough that the vast majority of users can create new Builders as necessary to support building new types of files in their configurations, without having to dive any deeper into this subsystem.

The base class here is BuilderBase. This is a concrete base class which does, in fact, represent the Builder objects that we (or users) create.

There is also a proxy that looks like a Builder:

CompositeBuilder

This proxies for a Builder with an action that is actually a dictionary that knows how to map file suffixes to a specific action. This is so that we can invoke different actions (compilers, compile options) for different flavors of source files.

Builders and their proxies have the following public interface methods used by other modules:

- **___call____()** THE public interface. Calling a Builder object (with the use of internal helper methods) sets up the target and source dependencies, appropriate mapping to a specific action, and the environment manipulation necessary for overridden construction variable. This also takes care of warning about possible mistakes in keyword arguments.

- **add_emitter()** Adds an emitter for a specific file suffix, used by some Tool modules to specify that (for example) a yacc invocation on a .y can create a .h and a .c file.

- **add_action()** Adds an action for a specific file suffix, heavily used by Tool modules to add their specific action(s) for turning a source file into an object file to the global static and shared object file Builders.

There are the following methods for internal use within this module:

- **__execute____()** The internal method that handles the heavily lifting when a
Builder is called. This is used so that the `__call__()` methods can set up warning about possible mistakes in keyword-argument overrides, and then execute all of the steps necessary so that the warnings only occur once.

- **get_name()** Returns the Builder’s name within a specific Environment, primarily used to try to return helpful information in error messages.
- **adjust_suffix()**
- **get_prefix()**
- **get_suffix()**
- **get_src_suffix()**
- **set_src_suffix()** Miscellaneous stuff for handling the prefix and suffix manipulation we use in turning source file names into target file names.

### 3.1 Functions

```python
match_splitext(path, suffixes=[]) Builder(**kw)
```

`match_splitext` is a function that takes a `path` and `suffixes` as arguments. It returns the name of the Builder object.

`Builder(**kw)`

A factory for builder objects.

```python
is_a_Builder(obj)
```

`is_a_Builder` is a function that returns True if the specified `obj` is one of our Builder classes.

The test is complicated a bit by the fact that CompositeBuilder is a proxy, not a subclass of BuilderBase.

### 3.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: <code>src/engine/SCons/Builder.py rel_3.0.0:4395:8972f6a2f699 ...</code></td>
</tr>
<tr>
<td>misleading_keywords</td>
<td>Value: <code>{’sources’: ’source’, ’targets’: ’target’}</code></td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: <code>’SCons’</code></td>
</tr>
</tbody>
</table>
### 3.3 Class DictCmdGenerator

UserDict.UserDict →
SCons.Util.OrderedDict →
SCons.Util.Selector →
SCons.Builder.DictCmdGenerator

This is a callable class that can be used as a command generator function. It holds on to a dictionary mapping file suffixes to Actions. It uses that dictionary to return the proper action based on the file suffix of the source file.

#### 3.3.1 Methods

```python
__init__(self, dict=None, source_ext_match=1)
```
Overriding: UserDict.UserDict.__init__

```python
src_suffixes(self)
```

```python
add_action(self, suffix, action)
```
Add a suffix-action pair to the mapping.

```python
__call__(self, target, source, env, for_signature)
```
Overriding: SCons.Util.Selector.__call__

**Inherited from SCons.Util.OrderedDict (Section 36.11)**

```python
__delitem__(), __setitem__(), clear(), copy(), items(), keys(), popitem(), setdefault(), update(), values()
```

**Inherited from UserDict.UserDict**

```python
__cmp__(), __contains__(), __getitem__(), __len__(), __repr__(), fromkeys(), get(), has_key(), iteritems(), iterkeys(), itervalues(), pop()
```

#### 3.3.2 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from UserDict.UserDict</td>
<td></td>
</tr>
</tbody>
</table>

*continued on next page*
3.4 Class CallableSelector

A callable dictionary that will, in turn, call the value it finds if it can.

3.4.1 Methods

``` python
__call__(self, env, source)
```

Overrides: SCons.Util.Selector.__call__

*Inherited from SCons.Util.OrderedDict (Section 36.11)*

- __delitem__(), __init__(), __setitem__(), clear(), copy(), items(), keys(), popitem(), setdefault(), update(), values()

*Inherited from UserDict.UserDict*

- __cmp__(), __contains__(), __getitem__(), __len__(), __repr__(), fromkeys(), get(), has_key(), iteritems(), iterkeys(), itervalues(), pop()

3.4.2 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hash</strong></td>
<td></td>
</tr>
</tbody>
</table>

*Inherited from UserDict.UserDict*
3.5 Class DictEmitter

A callable dictionary that maps file suffixes to emitters. When called, it finds the right emitter in its dictionary for the suffix of the first source file, and calls that emitter to get the right lists of targets and sources to return. If there’s no emitter for the suffix in its dictionary, the original target and source are returned.

3.5.1 Methods

```python
__call__(self, target, source, env)
```

Overrides: SCons.Util.Selector.__call__

Inherited from SCons.Util.OrderedDict (Section 36.11)

- __delitem__()
- __init__()
- __setitem__()
- clear()
- copy()
- items()
- keys()
- popitem()
- setdefault()
- update()
- values()

Inherited from UserDict.UserDict

- __cmp__()
- __contains__()
- __getitem__()
- __len__()
- __repr__()
- fromkeys()
- get()
- has_key()
- iteritems()
- iterkeys()
- itervalues()
- pop()

3.5.2 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hash</strong></td>
<td>Inherited from UserDict.UserDict</td>
</tr>
</tbody>
</table>
3.6 Class ListEmitter

A callable list of emitters that calls each in sequence, returning the result.

3.6.1 Methods

\_\_call\_\_(self, target, source, env)

Inherited from UserList.UserList

\_\_add\_\_(\_), \_\_cmp\_\_(\_), \_\_contains\_\_(\_), \_\_delitem\_\_(\_), \_\_delslice\_\_(\_),
\_\_eq\_\_(\_), \_\_ge\_\_(\_), \_\_getitem\_\_(\_), \_\_getslice\_\_(\_), \_\_gt\_\_(\_), \_\_iadd\_\_(\_),
\_\_imul\_\_(\_), \_\_init\_\_(\_), \_\_le\_\_(\_), \_\_len\_\_(\_), \_\_lt\_\_(\_), \_\_mul\_\_(\_), \_\_ne\_\_(\_),
\_\_radd\_\_(\_), \_\_repr\_\_(\_), \_\_rmul\_\_(\_), \_\_setitem\_\_(\_), \_\_setslice\_\_(\_), append\_(\_), count\_(\_),
extend\_(\_), index\_(\_), insert\_(\_), pop\_(\_), remove\_(\_), reverse\_(\_), sort\_(\_)

Inherited from \_\_abcoll.Sequence

\_\_iter\_\_(\_), \_\_reversed\_\_(\_)

Inherited from \_\_abcoll.Sized

\_\_subclasshook\_\_(\_)

Inherited from object

\_\_delattr\_\_(\_), \_\_format\_\_(\_), \_\_getattribute\_\_(\_), \_\_new\_\_(\_), \_\_reduce\_\_(\_),

39
3.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

3.6.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from UserList.UserList</td>
<td></td>
</tr>
<tr>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
<td></td>
</tr>
</tbody>
</table>

3.7 Class OverrideWarner

A class for warning about keyword arguments that we use as overrides in a Builder call.

This class exists to handle the fact that a single Builder call can actually invoke multiple builders. This class only emits the warnings once, no matter how many Builders are invoked.

3.7.1 Methods

```python
__init__(self, dict)
```
 Overrides: UserDict.UserDict.__init__

```python
warn(self)
```

Inherited from UserDict.UserDict

```python
__cmp__(), __contains__(), __delitem__(), __getitem__(), __len__(), __repr__(), __setitem__(), clear(), copy(), fromkeys(), get(), has_key(), items(), iteritems(), iterkeys(), itervalues(), keys(), pop(), popitem(), setdefault(), update(), values()
```

3.7.2 Class Variables
3.8 Class EmitterProxy

object — SCons.Builder.EmitterProxy

This is a callable class that can act as a Builder emitter. It holds on to a string that is a key into an Environment dictionary, and will look there at actual build time to see if it holds a callable. If so, we will call that as the actual emitter.

3.8.1 Methods

__init__(self, var)

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ extit(inherited documentation)

__call__(self, target, source, env)

__eq__(self, other)

__lt__(self, other)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

3.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
3.9 Class BuilderBase

object — SCons.Builder.BuilderBase

Base class for Builders, objects that create output nodes (files) from input nodes (files).

3.9.1 Methods

```python
__init__(self, action=None, prefix='', suffix='', src_suffix='',
         target_factory=None, source_factory=None, target_scanner=None,
         source_scanner=None, emitter=None, multi=0, env=None, single_source=0,
         name=None, chdir=<class 'SCons.Builder._Null'>, is_explicit=1,
         src_builder=None, ensure_suffix=False, **overrides)
```

__nonzero__(self)

__bool__(self)

get_name(self, env)

Attempts to get the name of the Builder.

Look at the BUILDERS variable of env, expecting it to be a dictionary containing this Builder, and return the key of the dictionary. If there’s no key, then return a directly-configured name (if there is one) or the name of the class (by default).

__eq__(self, other)

splitext(self, path, env=None)

__call__(self, env, target=None, source=None, chdir=<class 'SCons.Builder._Null'>, **kw)

adjust_suffix(self, suff)
```
get_prefix(self, env, sources=[])  

set_suffix(self, suffix)  

get_suffix(self, env, sources=[])  

set_src_suffix(self, src_suffix)  

get_src_suffix(self, env)  

Get the first src_suffix in the list of src_suffixes.

add_emitter(self, suffix, emitter)  

Add a suffix-emitter mapping to this Builder.

This assumes that emitter has been initialized with an appropriate dictionary type, and will throw a TypeError if not, so the caller is responsible for knowing that this is an appropriate method to call for the Builder in question.

add_src_builder(self, builder)  

Add a new Builder to the list of src_builders.

This requires wiping out cached values so that the computed lists of source suffixes get re-calculated.

src_builder_sources(self, env, source, overwarn={})  

get_src_builders(self, env)  

Returns the list of source Builders for this Builder.

This exists mainly to look up Builders referenced as strings in the 'BUILDER' variable of the construction environment and cache the result.
The suffix list may contain construction variable expansions, so we have to evaluate the individual strings. To avoid doing this over and over, we memoize the results for each construction environment.

Returns the list of source suffixes for all src_builders of this Builder.

This is essentially a recursive descent of the src_builder 'tree.' (This value isn’t cached because there may be changes in a src_builder many levels deep that we can’t see.)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

3.9.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

3.10 Class CompositeBuilder

A Builder Proxy whose main purpose is to always have a DictCmdGenerator as its action, and to provide access to the DictCmdGenerator’s add_action() method.
3.10.1 Methods

```python
__init__(self, builder, cmdgen)

Wrap an object as a Proxy object. Overrides: object.__init__ (inherited documentation)
```

```python
__call__(...)

A Python Descriptor class that delegates attribute fetches to an underlying wrapped subject of a Proxy. Typical use:

class Foo(Proxy):
    __str__ = Delegate('__str__')
```

```python
add_action(self, suffix, action)
```

Inherited from `SCons.Util.Proxy` (Section 36.5)

```python
__eq__(), __getattr__(), get()
```

Inherited from `object`

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

3.10.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from <code>object</code></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
4 Module SCons.CacheDir

CacheDir support

4.1 Functions

**CacheRetrieveFunc**(target, source, env)

**CacheRetrieveString**(target, source, env)

**CachePushFunc**(target, source, env)

4.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: <code>src/engine/SCons/CacheDir.py rel_3.0.0:4395:8972f6a2f699...</code></td>
</tr>
<tr>
<td><strong>doc</strong></td>
<td>Value: ...</td>
</tr>
<tr>
<td>cache_enabled</td>
<td>Value: True</td>
</tr>
<tr>
<td>cache_debug</td>
<td>Value: False</td>
</tr>
<tr>
<td>cache_force</td>
<td>Value: False</td>
</tr>
<tr>
<td>cache_show</td>
<td>Value: False</td>
</tr>
<tr>
<td>cache_readonly</td>
<td>Value: False</td>
</tr>
<tr>
<td>CacheRetrieve</td>
<td>Value: <code>&lt;SCons.Action.FunctionAction object&gt;</code></td>
</tr>
<tr>
<td>CacheRetrieveSilent</td>
<td>Value: <code>&lt;SCons.Action.FunctionAction object&gt;</code></td>
</tr>
<tr>
<td>warned</td>
<td>Value: <code>{}</code></td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: <code>'SCons'</code></td>
</tr>
</tbody>
</table>

4.3 Class CacheDir

```
object SCons.CacheDir.CacheDir
```
4.3.1 Methods

```python
__init__(self, path)
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ extit(inherited documentation)
```

```python
CacheDebug(self, fmt, target, cachefile)
```

```python
is_enabled(self)
```

```python
is_readonly(self)
```

```python
cachepath(self, node)
```

```python
retrieve(self, node)
```

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in built().

Note that there’s a special trick here with the execute flag (one that’s not normally done for other actions). Basically if the user requested a no_exec (-n) build, then SCons.Action.execute_actions is set to 0 and when any action is called, it does its showing but then just returns zero instead of actually calling the action execution operation. The problem for caching is that if the file does NOT exist in cache then the CacheRetrieveString won’t return anything to show for the task, but the Action.__call__ won’t call CacheRetrieveFunc; instead it just returns zero, which makes the code below think that the file was successfully retrieved from the cache, therefore it doesn’t do any subsequent building. However, the CacheRetrieveString didn’t print anything because it didn’t actually exist in the cache, and no more build actions will be performed, so the user just sees nothing. The fix is to tell Action.__call__ to always execute the CacheRetrieveFunc and then have the latter explicitly check SCons.Action.execute_actions itself.

```python
push(self, node)
```

```python
push_if_forced(self, node)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
```
```python
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 4.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

48
5 Module SCons.Conftest

SCons.Conftest

Autoconf-like configuration support; low level implementation of tests.

5.1 Functions

**CheckBuilder** *(context, text=None, language=None)*

Configure check to see if the compiler works. Note that this uses the current value of compiler and linker flags, make sure $CFLAGS, $CPPFLAGS and $LIBS are set correctly. "language" should be "C" or "C++" and is used to select the compiler. Default is "C". "text" may be used to specify the code to be build. Returns an empty string for success, an error message for failure.

**CheckCC**(context)

Configure check for a working C compiler.

This checks whether the C compiler, as defined in the $CC construction variable, can compile a C source file. It uses the current $CCCOM value too, so that it can test against non working flags.

**CheckSHCC**(context)

Configure check for a working shared C compiler.

This checks whether the C compiler, as defined in the $SHCC construction variable, can compile a C source file. It uses the current $SHCCCOM value too, so that it can test against non working flags.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CheckCXX(context)</strong></td>
<td>Configure check for a working CXX compiler. This checks whether the CXX compiler, as defined in the $CXX construction variable, can compile a CXX source file. It uses the current $CXXCOM value too, so that it can test against non working flags.</td>
</tr>
<tr>
<td><strong>CheckSHCXX(context)</strong></td>
<td>Configure check for a working shared CXX compiler. This checks whether the CXX compiler, as defined in the $SHCXX construction variable, can compile a CXX source file. It uses the current $SHCXXCOM value too, so that it can test against non working flags.</td>
</tr>
<tr>
<td><strong>CheckFunc(context, function_name, header=None, language=None)</strong></td>
<td>Configure check for a function &quot;function_name&quot;. &quot;language&quot; should be 'C' or 'C++' and is used to select the compiler. Default is 'C'. Optional 'header' can be defined to define a function prototype, include a header file or anything else that comes before main(). Sets HAVE_function_name in context.havedict according to the result. Note that this uses the current value of compiler and linker flags, make sure $CFLAGS, $CPPFLAGS and $LIBS are set correctly. Returns an empty string for success, an error message for failure.</td>
</tr>
<tr>
<td><strong>CheckHeader(context, header_name, header=None, language=None, include_quotes=None)</strong></td>
<td>Configure check for a C or C++ header file 'header_name'. Optional 'header' can be defined to do something before including the header file (unusual, supported for consistency). &quot;language&quot; should be 'C' or 'C++' and is used to select the compiler. Default is 'C'. Sets HAVE_header_name in context.havedict according to the result. Note that this uses the current value of compiler and linker flags, make sure $CFLAGS and $CPPFLAGS are set correctly. Returns an empty string for success, an error message for failure.</td>
</tr>
</tbody>
</table>
**CheckType**(*context, type_name, fallback=None, header=None, language=None*)

Configure check for a C or C++ type 'type_name'. Optional 'header' can be defined to include a header file. 'language' should be 'C' or 'C++' and is used to select the compiler. Default is 'C'. Sets HAVE_type_name in context.havedict according to the result. Note that this uses the current value of compiler and linker flags, make sure $CFLAGS, $CPPFLAGS and $LIBS are set correctly. Returns an empty string for success, an error message for failure.

**CheckTypeSize**(*context, type_name, header=None, language=None, expect=None*)

This check can be used to get the size of a given type, or to check whether the type is of expected size.

**Arguments:**

- **type** (*str*)
  the type to check
- **includes** (*sequence*)
  list of headers to include in the test code before testing the type
- **language** (*str*)
  'C' or 'C++'
- **expect** (*int*)
  if given, will test wether the type has the given number of bytes. If not given, will automatically find the size.

**Returns:**

- **status** (*int*)
  0 if the check failed, or the found size of the type if the check succeeded.
CheckDeclaration\((context, symbol, includes=\texttt{None}, language=\texttt{None})\)

Checks whether symbol is declared.

Use the same test as autoconf, that is test whether the symbol is defined as a macro or can be used as an r-value.

Arguments:

- `symbol` (str)
  the symbol to check
- `includes` (str)
  Optional "header" can be defined to include a header file.
- `language` (str)
  only C and C++ supported.

Returns:

- `status` (bool)
  True if the check failed, False if succeeded.

CheckLib\((context, libs, func\_name=\texttt{None}, header=\texttt{None}, extra\_libs=\texttt{None}, call=\texttt{None}, language=\texttt{None}, autoadd=1, append=\texttt{True})\)

Configure check for a C or C++ libraries 'libs'. Searches through the list of libraries, until one is found where the test succeeds. Tests if 'func\_name' or 'call' exists in the library. Note: if it exists in another library the test succeeds anyway! Optional "header" can be defined to include a header file. If not given a default prototype for 'func\_name' is added. Optional 'extra\_libs' is a list of library names to be added after 'lib\_name' in the build command. To be used for libraries that 'lib\_name' depends on. Optional 'call' replaces the call to 'func\_name' in the test code. It must consist of complete C statements, including a trailing ';'. Both 'func\_name' and 'call' arguments are optional, and in that case, just linking against the libs is tested. 'language' should be 'C' or 'C++' and is used to select the compiler. Default is 'C'. Note that this uses the current value of compiler and linker flags, make sure $CFLAGS, $CPPFLAGS and $LFLAGS are set correctly. Returns an empty string for success, an error message for failure.
CheckProg(context, prog_name)

Configure check for a specific program.

Check whether program prog_name exists in path. If it is found, returns the path for it, otherwise returns None.

5.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogInputFiles</td>
<td>Value: 1</td>
</tr>
<tr>
<td>LogErrorMessages</td>
<td>Value: 1</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>


6 Module SCons.Debug

SCons.Debug

Code for debugging SCons internal things. Shouldn’t be needed by most users. Quick shortcuts:

from SCons.Debug import caller_trace caller_trace()

6.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>logInstanceCreation(instance, name=None)</td>
</tr>
<tr>
<td>string_to_classes(s)</td>
</tr>
<tr>
<td>fetchLoggedInstances(classes='*')</td>
</tr>
<tr>
<td>countLoggedInstances(classes, file=&lt;epydoc.docintrospector._DevNull object&gt;)</td>
</tr>
<tr>
<td>listLoggedInstances(classes, file=&lt;epydoc.docintrospector._DevNull object&gt;)</td>
</tr>
<tr>
<td>dumpLoggedInstances(classes, file=&lt;epydoc.docintrospector._DevNull object&gt;)</td>
</tr>
<tr>
<td>memory()</td>
</tr>
<tr>
<td>caller__stack()</td>
</tr>
<tr>
<td>caller_trace(back=0)</td>
</tr>
</tbody>
</table>

Trace caller stack and save info into global dicts, which are printed automatically at the end of SCons execution.

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump_caller_counts(file=&lt;epydoc.docintrospector._DevNull object&gt;)</td>
</tr>
<tr>
<td>func_shorten(func_tuple)</td>
</tr>
</tbody>
</table>
**Trace**(*msg*, *file=None*, *mode='w'*, *tstamp=None*)

Write a trace message to a file. Whenever a file is specified, it becomes the default for the next call to `Trace()`.

### 6.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__revision__</code></td>
<td>Value: 'src/engine/SCons/Debug.py rel_3.0.0:4395:8972f6a2f699 20...'</td>
</tr>
<tr>
<td><code>track_instances</code></td>
<td>Value: False</td>
</tr>
<tr>
<td><code>tracked_classes</code></td>
<td>Value: {}</td>
</tr>
<tr>
<td><code>caller_bases</code></td>
<td>Value: {}</td>
</tr>
<tr>
<td><code>caller_dicts</code></td>
<td>Value: {}</td>
</tr>
<tr>
<td><code>shorten_list</code></td>
<td>Value: [('/scons/SCons/', 1), ('/src/engine/SCons/', 1), ('/usr/...</td>
</tr>
<tr>
<td><code>TraceFP</code></td>
<td>Value: {}</td>
</tr>
<tr>
<td><code>TraceDefault</code></td>
<td>Value: '/dev/tty'</td>
</tr>
<tr>
<td><code>TimeStampDefault</code></td>
<td>Value: None</td>
</tr>
<tr>
<td><code>StartTime</code></td>
<td>Value: 1505764882.16</td>
</tr>
<tr>
<td><code>PreviousTime</code></td>
<td>Value: 1505764882.16</td>
</tr>
<tr>
<td><code>__package__</code></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>
7 Module SCons.Defaults

SCons.Defaults

Builders and other things for the local site. Here’s where we’ll duplicate the functionality of autoconf until we move it into the installation procedure or use something like qmconf.

The code that reads the registry to find MSVC components was borrowed from distutils.msvccompiler.

7.1 Functions

**DefaultEnvironment(**\*args, **kw\*)**

Initial public entry point for creating the default construction Environment.

After creating the environment, we overwrite our name (DefaultEnvironment) with the _fetch_DefaultEnvironment() function, which more efficiently returns the initialized default construction environment without checking for its existence.

(This function still exists with its _default_check because someone else (cough Script/__init__.py cough) may keep a reference to this function. So we can’t use the fully functional idiom of having the name originally be a something that only creates the construction environment and then overwrites the name.)

**StaticObjectEmitter**(target, source, env)

**SharedObjectEmitter**(target, source, env)

**SharedFlagChecker**(source, target, env)

**get_paths_str**(dest)

**chmod_func**(dest, mode)

**chmod_strfunc**(dest, mode)
Variables Module SCons.Defaults

**copy_func**(dest, src, symlinks=True)

If symlinks (is true), then a symbolic link will be shallow copied and recreated as a symbolic link; otherwise, copying a symbolic link will be equivalent to copying the symbolic link’s final target regardless of symbolic link depth.

**delete_func**(dest, must_exist=0)

**delete_strfunc**(dest, must_exist=0)

**mkdir_func**(dest)

**move_func**(dest, src)

**touch_func**(dest)

**processDefines**(defs)

process defines, resolving strings, lists, dictionaries, into a list of strings

### 7.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Defaults.py rel_3.0.0:4395:8972f6a2f699...</td>
</tr>
<tr>
<td>SharedCheck</td>
<td>Value: &lt;SCons.Action.FunctionAction object&gt;</td>
</tr>
<tr>
<td>CScan</td>
<td>Value: &lt;SCons.Scanner.ClassicCPP object&gt;</td>
</tr>
<tr>
<td>DScan</td>
<td>Value: &lt;SCons.Scanner.D.D object&gt;</td>
</tr>
<tr>
<td>LaTeXScan</td>
<td>Value: &lt;SCons.Scanner.LaTeX.LaTeX object&gt;</td>
</tr>
<tr>
<td>ObjSourceScan</td>
<td>Value: &lt;SCons.Scanner.Base object&gt;</td>
</tr>
<tr>
<td>ProgScan</td>
<td>Value: &lt;SCons.Scanner.Base object&gt;</td>
</tr>
<tr>
<td>DirScanner</td>
<td>Value: &lt;SCons.Scanner.Base object&gt;</td>
</tr>
<tr>
<td>DirEntryScanner</td>
<td>Value: &lt;SCons.Scanner.Base object&gt;</td>
</tr>
<tr>
<td>CAAction</td>
<td>Value: &lt;SCons.Action.LazyAction object&gt;</td>
</tr>
<tr>
<td>ShCAction</td>
<td>Value: &lt;SCons.Action.LazyAction object&gt;</td>
</tr>
<tr>
<td>CXXAction</td>
<td>Value: &lt;SCons.Action.LazyAction object&gt;</td>
</tr>
<tr>
<td>ShCXXAction</td>
<td>Value: &lt;SCons.Action.LazyAction object&gt;</td>
</tr>
</tbody>
</table>

*continued on next page*
### 7.3 Class NullCmdGenerator

object

```python
SCons.Defaults.NullCmdGenerator
```

This is a callable class that can be used in place of other command generators if you don’t want them to do anything.

The `__call__` method for this class simply returns the thing you instantiated it with.

Example usage:  
```
env['DO NOTHING'] = NullCmdGenerator  
env['LINKCOM'] = '$DO NOTHING('$L $SOURCES $TARGET')'
```
7.3.1 Methods

```python
__init__(self, cmd)
```

Overrides:
object.__init__ extit(inherited documentation)

```python
__call__(self, target, source, env, for_signature=None)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

7.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

7.4 Class Variable_Method_Caller

object

```
SCons.Defaults.Variable_Method_Caller
```

A class for finding a construction variable on the stack and calling one of its methods.

We use this to support "construction variables" in our string eval()s that actually stand in for methods--specifically, use of "RDirs" in call to _concat that should actually execute the 'TARGET.RDirs' method. (We used to support this by creating a little "build dictionary" that mapped RDirs to the method, but this got in the way of Memoizing construction environments, because we had to create new environment objects to hold the variables.)

7.4.1 Methods

```python
__init__(self, variable, method)
```

Overrides:
object.__init__ extit(inherited documentation)
```python
__call__(self, *args, **kw)
```

**Inherited from object**

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 7.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inherited from object</strong></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
8 Module SCons.Environment

SCons.Environment

Base class for construction Environments. These are the primary objects used to communicate dependency and construction information to the build engine.

Keyword arguments supplied when the construction Environment is created are construction variables used to initialize the Environment.

8.1 Functions

```python
alias_builder(env, target, source)
```

```python
apply_tools(env, tools, toolpath)
```

```python
copy_non_reserved_keywords(dict)
```

```python
is_valid_construction_var(varstr)
```

Return if the specified string is a legitimate construction variable.

```python
default_decide_source(dependency, target, prev_ni)
```

```python
default_decide_target(dependency, target, prev_ni)
```

```python
default_copy_from_cache(src, dst)
```
NoSubstitutionProxy(subject)

An entry point for returning a proxy subclass instance that overrides the subst*() methods so they don’t actually perform construction variable substitution. This is specifically intended to be the shim layer in between global function calls (which don’t want construction variable substitution) and the DefaultEnvironment() (which would substitute variables if left to its own devices).

We have to wrap this in a function that allows us to delay definition of the class until it’s necessary, so that when it subclasses Environment it will pick up whatever Environment subclass the wrapper interface might have assigned to SCons.Environment.Environment.

### 8.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Environment.py rel_3.0.0:4395:8972f6a2f...</td>
</tr>
<tr>
<td>CleanTargets</td>
<td>Value: {}</td>
</tr>
<tr>
<td>CalculatorArgs</td>
<td>Value: {}</td>
</tr>
<tr>
<td>reserved_construction_var_names</td>
<td>Value: ['CHANGED_SOURCES', 'CHANGED_TARGETS', 'SOURCE', 'SOURCES']</td>
</tr>
<tr>
<td>future_reserved_construction_var_names</td>
<td>Value: []</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

### 8.3 Class MethodWrapper

object

SCons.Environment.MethodWrapper

**Known Subclasses:** SCons.Environment.BuilderWrapper

A generic Wrapper class that associates a method (which can actually be any callable) with an object. As part of creating this MethodWrapper object an attribute with the specified (by default, the name of the supplied method) is added to the underlying object. When
that new 'method' is called, our \texttt{\_\_call\_\_()} method adds the object as the first argument, simulating the Python behavior of supplying 'self' on method calls.

We hang on to the name by which the method was added to the underlying base class so that we can provide a method to "clone" ourselves onto a new underlying object being copied (without which we wouldn’t need to save that info).

\subsection{8.3.1 Methods}

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
\texttt{\_\_init\_\_}(\texttt{self}, \texttt{object}, \texttt{method}, \texttt{name=\texttt{None}})  \\
\hline
\texttt{x.\_\_init\_\_(...)} initializes \texttt{x}; see \texttt{help(type(x))} for signature  \\
Overrides: \texttt{object.\_\_init\_\_} \texttt{exttit(inherited\ documentation)}  \\
\hline
\texttt{\_\_call\_\_}(\texttt{self}, \*\texttt{args}, **\texttt{kwargs})  \\
\hline
\texttt{clone}(\texttt{self}, \texttt{new\_\_object})  \\
\hline
\end{tabular}
\caption{Class BuilderWrapper Methods}
\end{table}

\textit{Inherited from object}

\begin{verbatim}
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
\end{verbatim}

\subsection{8.3.2 Properties}

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
\textit{Inherited from object}  \\
\hline
\texttt{\_\_class\_\_}  \\
\hline
\end{tabular}
\caption{Class BuilderWrapper Properties}
\end{table}

\subsection{8.4 Class BuilderWrapper}

\begin{verbatim}
object
SCons.Environment.MethodWrapper
\end{verbatim}
A MethodWrapper subclass that associates an environment with a Builder.

This mainly exists to wrap the \_\_call\_\_() function so that all calls to Builders can have their argument lists massaged in the same way (treat a lone argument as the source, treat two arguments as target then source, make sure both target and source are lists) without having to have cut-and-paste code to do it.

As a bit of obsessive backwards compatibility, we also intercept attempts to get or set the 'env' or 'builder' attributes, which were the names we used before we put the common functionality into the MethodWrapper base class. We’ll keep this around for a while in case people shipped Tool modules that reached into the wrapper (like the Tool/qt.py module does, or did). There shouldn’t be a lot attribute fetching or setting on these, so a little extra work shouldn’t hurt.

### 8.4.1 Methods

- **\_\_call\_\_\_
  \(self, target=None, source=<class \`SCons.Environment._Null\'>, *args, **kw\)**


- **\_\_repr\_\_
  \(self\)

  \(\text{repr}(x)\)\n
  Overrides: object.\_\_repr\_\_

- **\_\_str\_\_
  \(self\)

  \(\text{str}(x)\)\n
  Overrides: object.\_\_str\_\_

- **\_\_getattr\_\_
  \(self, name\)**

- **\_\_setattr\_\_
  \(self, name, value\)**

  \(x.\_\_setattr\_\_\_\_\_\_('name', value) \Rightarrow x.name = value\)

  Overrides: object.\_\_setattr__

_Inherited from SCons.Environment.MethodWrapper (Section 8.3)_

- **\_\_init\_\_()**, **clone()**

_Inherited from object_

- **\_\_delattr\_\_()**, **\_\_format\_\_()**, **\_\_getattribute\_\_\_()**, **\_\_hash\_\_()**, **\_\_new\_\_()**, **\_\_reduce\_\_()**, **\_\_reduce_ex\_\_()**, **\_\_sizeof\_\_()**, **\_\_subclasshook\_\_()**

64
8.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

8.5 Class BuilderDict


This is a dictionary-like class used by an Environment to hold the Builders. We need to do this because every time someone changes the Builders in the Environment’s BUILDERS dictionary, we must update the Environment’s attributes.

8.5.1 Methods

```python
__init__(self, dict, env)
Overrides: UserDict.UserDict.__init__

__semi_deepcopy__(self)

__setitem__(self, item, val)
Overrides: UserDict.UserDict.__setitem__

__delitem__(self, item)
Overrides: UserDict.UserDict.__delitem__

update(self, dict)
Overrides: UserDict.UserDict.update
```

Inherited from UserDict.UserDict

```python
__cmp__(), __contains__(), __getitem__(), __len__(), __repr__(), clear(), copy(), fromkeys(), get(), has_key(), items(), iteritems(), iterkeys(), itervalues(), keys(), pop(), popitem(), setdefault(), values()
```

8.5.2 Class Variables
8.6 Class SubstitutionEnvironment

object ->
SCons.Environment.SubstitutionEnvironment

Known Subclasses: SCons.Environment.Base

Base class for different flavors of construction environments.

This class contains a minimal set of methods that handle construction variable expansion and conversion of strings to Nodes, which may or may not be actually useful as a stand-alone class. Which methods ended up in this class is pretty arbitrary right now. They’re basically the ones which we’ve empirically determined are common to the different construction environment subclasses, and most of the others that use or touch the underlying dictionary of construction variables.

Eventually, this class should contain all the methods that we determine are necessary for a "minimal" interface to the build engine. A full "native Python" SCons environment has gotten pretty heavyweight with all of the methods and Tools and construction variables we’ve jammed in there, so it would be nice to have a lighter weight alternative for interfaces that don’t need all of the bells and whistles. (At some point, we’ll also probably rename this class 'Base,' since that more reflects what we want this class to become, but because we’ve released comments that tell people to subclass Environment.Base to create their own flavors of construction environment, we’ll save that for a future refactoring when this class actually becomes useful.)

8.6.1 Methods

```python
__init__(self, **kw)
```

Initialization of an underlying SubstitutionEnvironment class. Overrides: object.__init__

```python
__eq__(self, other)
```

```python
__delitem__(self, key)
```

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from UserDict.UserDict</td>
<td></td>
</tr>
<tr>
<td><strong>hash</strong></td>
<td></td>
</tr>
</tbody>
</table>
___getitem__(self, key)

___setitem__(self, key, value)

get(self, key, default=None)

Emulates the get() method of dictionaries.

has_key(self, key)

___contains__(self, key)

items(self)

arg2nodes(self, args, node_factory=<class 'SCons.Environment._Null'>, lookup_list=<class 'SCons.Environment._Null'>, **kw)

gvars(self)

lvars(self)

subst(self, string, raw=0, target=None, source=None, conv=None, executor=None)

Recursively interpolates construction variables from the Environment into the specified string, returning the expanded result. Construction variables are specified by a $ prefix in the string and begin with an initial underscore or alphabetic character followed by any number of underscores or alphanumeric characters. The construction variable names may be surrounded by curly braces to separate the name from trailing characters.

subst_kw(self, kw, raw=0, target=None, source=None)

subst_list(self, string, raw=0, target=None, source=None, conv=None, executor=None)

Calls through to SCons.Subst.scons_subst_list(). See the documentation for that function.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>subst_path(self, path, target=None, source=None)</code></td>
<td>Substitute a path list, turning EntryProxies into Nodes and leaving Nodes (and other objects) as-is.</td>
</tr>
<tr>
<td><code>subst_target_source(self, string, raw=0, target=None, source=None, conv=None, executor=None)</code></td>
<td>Recursively interpolates construction variables from the Environment into the specified string, returning the expanded result. Construction variables are specified by a $ prefix in the string and begin with an initial underscore or alphabetic character followed by any number of underscores or alphanumeric characters. The construction variable names may be surrounded by curly braces to separate the name from trailing characters.</td>
</tr>
<tr>
<td><code>backtick(self, command)</code></td>
<td></td>
</tr>
<tr>
<td><code>AddMethod(self, function, name=None)</code></td>
<td>Adds the specified function as a method of this construction environment with the specified name. If the name is omitted, the default name is the name of the function itself.</td>
</tr>
<tr>
<td><code>RemoveMethod(self, function)</code></td>
<td>Removes the specified function’s MethodWrapper from the added_methods list, so we don’t re-bind it when making a clone.</td>
</tr>
</tbody>
</table>
**Override**(self, overrides)

Produce a modified environment whose variables are overridden by the overrides dictionaries. "overrides" is a dictionary that will override the variables of this environment.

This function is much more efficient than Clone() or creating a new Environment because it doesn’t copy the construction environment dictionary, it just wraps the underlying construction environment, and doesn’t even create a wrapper object if there are no overrides.

**ParseFlags**(self, *flags*)

Parse the set of flags and return a dict with the flags placed in the appropriate entry. The flags are treated as a typical set of command-line flags for a GNU-like toolchain and used to populate the entries in the dict immediately below. If one of the flag strings begins with a bang (exclamation mark), it is assumed to be a command and the rest of the string is executed; the result of that evaluation is then added to the dict.

**MergeFlags**(self, args, unique=1, dict=None)

Merge the dict in args into the construction variables of this env, or the passed-in dict. If args is not a dict, it is converted into a dict using ParseFlags. If unique is not set, the flags are appended rather than merged.

*Inherited from object*

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

**8.6.2 Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
8.7 Class Base

![Class Base Diagram]

**Known Subclasses:** SCons.Environment.OverrideEnvironment, SCons.Script.SConscript.SConsEnvironment

Base class for "real" construction Environments. These are the primary objects used to communicate dependency and construction information to the build engine.

Keyword arguments supplied when the construction Environment is created are construction variables used to initialize the Environment.

### 8.7.1 Methods

- **`Action(self, *args, **kw)`**
- **`AddPostAction(self, files, action)`**
- **`AddPreAction(self, files, action)`**
- **`Alias(self, target, source=[], action=None, **kw)`**
- **`AlwaysBuild(self, *targets)`**
- **`Append(self, **kw)`**

Append values to existing construction variables in an Environment.
Append path elements to the path `name` in the `ENV` dictionary for this environment. Will only add any particular path once, and will normpath and normcase all paths to help assure this. This can also handle the case where the env variable is a list instead of a string.

If `delete_exists` is 0, a newpath which is already in the path will not be moved to the end (it will be left where it is).

Append values to existing construction variables in an Environment, if they're not already there. If `delete_exists` is 1, removes existing values first, so values move to end.

Return a copy of a construction Environment. The copy is like a Python 'deep copy'--that is, independent copies are made recursively of each objects--except that a reference is copied when an object is not deep-copyable (like a function). There are no references to any mutable objects in the original Environment.

Builds the supplied target files from the supplied source files using the supplied action. Action may be any type that the Builder constructor will accept for an action.
Configure\((self, *args, **kw)\)

Copy\((self, *args, **kw)\)

Decider\((self, function)\)

Depends\((self, target, dependency)\)

Explicitly specify that ‘target’s depend on ‘dependency’.

Detect\((self, progs)\)

Return the first available program in progs.

Dictionary\((self, *args)\)

Dir\((self, name, *args, **kw)\)

Dump\((self, key=None)\)

Using the standard Python pretty printer, return the contents of the scons build environment as a string.

If the key passed in is anything other than None, then that will be used as an index into the build environment dictionary and whatever is found there will be fed into the pretty printer. Note that this key is case sensitive.

Entry\((self, name, *args, **kw)\)

Environment\((self, **kw)\)

Execute\((self, action, *args, **kw)\)

Directly execute an action through an Environment

File\((self, name, *args, **kw)\)
**Class Base Module SCons.Environment**

**FindFile** *(self, file, dirs)*

**FindInstalledFiles** *(self)*

returns the list of all targets of the Install and InstallAs Builder.

**FindIxes** *(self, paths, prefix, suffix)*

Search a list of paths for something that matches the prefix and suffix.

paths - the list of paths or nodes. prefix - construction variable for the prefix.
suffix - construction variable for the suffix.

**FindSourceFiles** *(self, node=’.’)*

returns a list of all source files.

**Flatten** *(self, sequence)*

**GetBuildPath** *(self, files)*

**Glob** *(self, pattern, ondisk=True, source=False, strings=False, exclude=None)*

**Ignore** *(self, target, dependency)*

Ignore a dependency.

**Literal** *(self, string)*

**Local** *(self, *targets)*

**NoCache** *(self, *targets)*

Tags a target so that it will not be cached
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NoClean</strong></td>
<td>Tags a target so that it will not be cleaned by -c</td>
</tr>
<tr>
<td><strong>ParseConfig</strong></td>
<td>Use the specified function to parse the output of the command in order to modify the current environment. The 'command' can be a string or a list of strings representing a command and its arguments. 'Function' is an optional argument that takes the environment, the output of the command, and the unique flag. If no function is specified, MergeFlags, which treats the output as the result of a typical 'X-config' command (i.e. gtk-config), will merge the output into the appropriate variables.</td>
</tr>
<tr>
<td><strong>ParseDepends</strong></td>
<td>Parse a mkdep-style file for explicit dependencies. This is completely abusable, and should be unnecessary in the <em>normal</em> case of proper SCons configuration, but it may help make the transition from a Make hierarchy easier for some people to swallow. It can also be genuinely useful when using a tool that can write a .d file, but for which writing a scanner would be too complicated.</td>
</tr>
<tr>
<td><strong>Platform</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Precious</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Prepend</strong></td>
<td>Prepend values to existing construction variables in an Environment.</td>
</tr>
</tbody>
</table>
### PrependENVPath

```python
PrependENVPath(self, name, newpath, envname='ENV', sep=':', delete_existing=1)
```

Prepend path elements to the path `name` in the `ENV` dictionary for this environment. Will only add any particular path once, and will normpath and normcase all paths to help assure this. This can also handle the case where the env variable is a list instead of a string.

If `delete_existing` is 0, a newpath which is already in the path will not be moved to the front (it will be left where it is).

### PrependUnique

```python
PrependUnique(self, delete_existing=0, **kw)
```

Prepend values to existing construction variables in an Environment, if they’re not already there. If `delete_existing` is 1, removes existing values first, so values move to front.

### Pseudo

```python
Pseudo(self, *targets)
```

### PyPackageDir

```python
PyPackageDir(self, modulename)
```

### Replace

```python
Replace(self, **kw)
```

Replace existing construction variables in an Environment with new construction variables and/or values.

### ReplaceIxes

```python
ReplaceIxes(self, path, old_prefix, old_suffix, new_prefix, new_suffix)
```

Replace `old_prefix` with `new_prefix` and `old_suffix` with `new_suffix`.

- `env` - Environment used to interpolate variables
- `path` - the path that will be modified
- `old_prefix` - construction variable for the old prefix
- `old_suffix` - construction variable for the old suffix
- `new_prefix` - construction variable for the new prefix
- `new_suffix` - construction variable for the new suffix

### Repository

```python
Repository(self, *dirs, **kw)
```

```
**Requires**(*self, target, prerequisite*)

Specify that ‘prerequisite’ must be built before ‘target’, (but ‘target’ does not actually depend on ‘prerequisite’ and need not be rebuilt if it changes).

**SConsignFile**(*self, name=’.sconsign’, dbm_module=None*)

**Scanner**(*self, *args, **kw*)

**SetDefault**(*self, **kw*)

**SideEffect**(*self, side_effect, target*)

Tell scons that side_effects are built as side effects of building targets.

**SourceCode**(*self, entry, builder*)

Arrange for a source code builder for (part of) a tree.

**SourceSignatures**(*self, type*)

**Split**(*self, arg*)

This function converts a string or list into a list of strings or Nodes. This makes things easier for users by allowing files to be specified as a white-space separated list to be split.

**The input rules are:**

- A single string containing names separated by spaces. These will be split apart at the spaces.
- A single Node instance
- A list containing either strings or Node instances. Any strings in the list are not split at spaces.

In all cases, the function returns a list of Nodes and strings.
**Class Base Module SCons.Environment**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TargetSignatures</strong>(<em>self, type</em>)</td>
<td></td>
</tr>
<tr>
<td><strong>Tool</strong>(*self, tool, toolpath=None, *<em>kw</em>)</td>
<td></td>
</tr>
<tr>
<td><strong>Value</strong>(<em>self, value, built_value=None</em>)</td>
<td></td>
</tr>
<tr>
<td><strong>VariantDir</strong>(<em>self, variant_dir, src_dir, duplicate=1</em>)</td>
<td></td>
</tr>
<tr>
<td><strong>WhereIs</strong>(*self, prog, path=None, pathext=None, reject=[])</td>
<td>Find prog in the path.</td>
</tr>
</tbody>
</table>

**__init__**(*self, platform=None, tools=None, toolpath=None, variables=None, parse_flags=None, **kw*)

Initialization of a basic SCons construction environment, including setting up special construction variables like BUILDER, PLATFORM, etc., and searching for and applying available Tools.

Note that we do not call the underlying base class (SubstitutionEnvironment) initialization, because we need to initialize things in a very specific order that doesn’t work with the much simpler base class initialization. Overrides: object.__init__

**get_CacheDir**(*self*)

**get_builder**(*self, name*)

Fetch the builder with the specified name from the environment.

**get_factory**(*self, factory, default='File*)

Return a factory function for creating Nodes for this construction environment.
get_scanner(self, skey)

Find the appropriate scanner given a key (usually a file suffix).

get_src_sig_type(self)

get_tgt_sig_type(self)

scanner_map_delete(self, kw=None)

Delete the cached scanner map (if we need to).

Inherited from SCons.Environment.SubstitutionEnvironment (Section 8.6)

AddMethod(), MergeFlagns(), Override(), ParseFlags(), RemoveMethod(), __contains__(), __delitem__(), __eq__(), __getitem__(), __iter__(), __len__(), __setitem__(), arg2nodes(), backtick(), get(), gvars(), has_key(), items(), 1vars(), subst(), subst_kw(), subst_list(), subst_path(), subst_target_source()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

8.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

8.8 Class OverrideEnvironment

object

SCons.Environment.SubstitutionEnvironment

SCons.Environment.Base

SCons.Environment.OverrideEnvironment
A proxy that overrides variables in a wrapped construction environment by returning values from an overrides dictionary in preference to values from the underlying subject environment.

This is a lightweight (I hope) proxy that passes through most use of attributes to the underlying Environment.Base class, but has just enough additional methods defined to act like a real construction environment with overridden values. It can wrap either a Base construction environment, or another OverrideEnvironment, which can in turn nest arbitrary OverrideEnvironments...

Note that we do not call the underlying base class (SubsitutionEnvironment) initialization, because we get most of those from proxying the attributes of the subject construction environment. But because we subclass SubstitutionEnvironment, this class also has inherited arg2nodes() and subst*() methods; those methods can’t be proxied because they need this object’s methods to fetch the values from the overrides dictionary.

8.8.1 Methods

```python
__init__(self, subject, overrides={})
```

Initialization of a basic SCons construction environment, including setting up special construction variables like BUILDER, PLATFORM, etc., and searching for and applying available Tools.

Note that we do not call the underlying base class (SubsititutionEnvironment) initialization, because we need to initialize things in a very specific order that doesn’t work with the much simpler base class initialization. Overrides: object.__init__ extit(inherited documentation)

```python
__getattr__(self, name)
```

```python
__setattr__(self, name, value)
x.__setattr__('name', value) <===> x.name = value  Overrides: object.__setattr__ extit(inherited documentation)
```

```python
__getitem__(self, key)
```

Overrides: SCons.Environment.SubstitutionEnvironment.__getitem__

```python
__setitem__(self, key, value)
```

Overrides: SCons.Environment.SubstitutionEnvironment.__setitem__
Class OverrideEnvironment

Module SCons.Environment

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>delitem</strong>(self, key)</td>
<td>Overrides: SCons.Environment.SubstitutionEnvironment.<strong>delitem</strong></td>
</tr>
<tr>
<td>has_key(self, key)</td>
<td>Overrides: SCons.Environment.SubstitutionEnvironment.has_key</td>
</tr>
<tr>
<td><strong>contains</strong>(self, key)</td>
<td>Overrides: SCons.Environment.SubstitutionEnvironment.<strong>contains</strong></td>
</tr>
<tr>
<td>Replace(self, **kw)</td>
<td>Replace existing construction variables in an Environment with new construction variables and/or values. Overrides: SCons.Environment.Base.Replace (inherited documentation)</td>
</tr>
</tbody>
</table>

Inherited from SCons.Environment.Base (Section 8.9)
Action(), AddPostAction(), AddPreAction(), Alias(), AlwaysBuild(), Append(),
AppendENVPath(), AppendUnique(), BuildDir(), Builder(), CacheDir(), Clean(),
Clone(), Command(), Configure(), Copy(), Decider(), Depends(), Detect(), Dir(),
Dump(), Entry(), Environment(), Execute(), File(), FindFile(), FindInstalledFiles(),
FindIxes(), FindSourceFiles(), Flatten(), GetBuildPath(), Glob(), Ignore(),
Literal(), Local(), NoCache(), NoClean(), ParseConfig(), ParseDepends(), Platform(),
Precious(), Prepend(), PrependENVPath(), PrependUnique(), Pseudo(),
PyPackageDir(), ReplaceIxes(), Repository(), Requires(), SConsignFile(), Scanner(),
SetDefault(), SideEffect(), SourceCode(), SourceSignatures(), Split(), TargetSignatures(),
Tool(), Value(), VariantDir(), WhereIs(), get_CacheDir(), get_builder(),
get_factory(), get_scanner(), get_src_sig_type(), get_tgt_sig_type(), scanner_map_delete()

Inherited from SCons.Environment.SubstitutionEnvironment (Section 8.6)

AddMethod(), MergeFlags(), Override(), ParseFlags(), RemoveMethod(), __eq__(),
arg2nodes(), backtick(), subst(), subst_kw(), subst_list(), subst_path(), subst_target_source()

Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __sizeof__(), __str__(), subclasshook()

8.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

8.9 Class Base

object

SCons.Environment.SubstitutionEnvironment

SCons.Environment.Base

SConsEnvironment

Base class for "real" construction Environments. These are the primary objects used to
communicate dependency and construction information to the build engine.

Keyword arguments supplied when the construction Environment is created are construction
variables used to initialize the Environment.
8.9.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action(self, *args, **kw)</td>
<td></td>
</tr>
<tr>
<td>AddPostAction(self, files, action)</td>
<td></td>
</tr>
<tr>
<td>AddPreAction(self, files, action)</td>
<td></td>
</tr>
<tr>
<td>Alias(self, target, source=[], action=None, **kw)</td>
<td></td>
</tr>
<tr>
<td>AlwaysBuild(self, *targets)</td>
<td></td>
</tr>
<tr>
<td>Append(self, **kw)</td>
<td>Append values to existing construction variables in an Environment.</td>
</tr>
</tbody>
</table>
| AppendENVPath(self, name, newpath, envname='ENV', sep=':', delete_existing=1) | Append path elements to the path 'name' in the 'ENV' dictionary for this environment. Will only add any particular path once, and will normpath and normcase all paths to help assure this. This can also handle the case where the env variable is a list instead of a string.
If delete existing is 0, a newpath which is already in the path will not be moved to the end (it will be left where it is). |
<p>| AppendUnique(self, delete_existing=0, **kw) | Append values to existing construction variables in an Environment, if they're not already there. If delete existing is 1, removes existing values first, so values move to end. |
| BuildDir(self, *args, **kw) |                                                                            |
| Builder(self, **kw)        |                                                                            |
| CacheDir(self, path)       |                                                                            |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clean</strong></td>
<td><code>self, targets, files</code></td>
<td>Return a copy of a construction Environment. The copy is like a Python 'deep copy'--that is, independent copies are made recursively of each objects--except that a reference is copied when an object is not deep-copyable (like a function). There are no references to any mutable objects in the original Environment.</td>
</tr>
<tr>
<td><strong>Clone</strong></td>
<td><code>self, tools=[], toolpath=None, parse_flags=None, **kw</code></td>
<td></td>
</tr>
<tr>
<td><strong>Command</strong></td>
<td><code>self, target, source, action, **kw</code></td>
<td>Builds the supplied target files from the supplied source files using the supplied action. Action may be any type that the Builder constructor will accept for an action.</td>
</tr>
<tr>
<td><strong>Configure</strong></td>
<td><code>self, *args, **kw</code></td>
<td></td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td><code>self, *args, **kw</code></td>
<td></td>
</tr>
<tr>
<td><strong>Decider</strong></td>
<td><code>self, function</code></td>
<td></td>
</tr>
<tr>
<td><strong>Depends</strong></td>
<td><code>self, target, dependency</code></td>
<td>Explicity specify that 'target's depend on 'dependency'.</td>
</tr>
<tr>
<td><strong>Detect</strong></td>
<td><code>self, progs</code></td>
<td>Return the first available program in progs.</td>
</tr>
<tr>
<td><strong>Dictionary</strong></td>
<td><code>self, *args</code></td>
<td></td>
</tr>
<tr>
<td><strong>Dir</strong></td>
<td><code>self, name, *args, **kw</code></td>
<td></td>
</tr>
</tbody>
</table>
**Dump**(*self, key=None*)

Using the standard Python pretty printer, return the contents of the scons build environment as a string.

If the key passed in is anything other than None, then that will be used as an index into the build environment dictionary and whatever is found there will be fed into the pretty printer. Note that this key is case sensitive.

**Entry**(*self, name, *args, **kw*)

**Environment**(*self, **kw*)

**Execute**(*self, action, *args, **kw*)

Directly execute an action through an Environment

**File**(*self, name, *args, **kw*)

**FindFile**(*self, file, dirs*)

**FindInstalledFiles**(*self*)

returns the list of all targets of the Install and InstallAs Builder.

**FindIxes**(*self, paths, prefix, suffix*)

Search a list of paths for something that matches the prefix and suffix.

paths - the list of paths or nodes. prefix - construction variable for the prefix. suffix - construction variable for the suffix.

**FindSourceFiles**(*self, node='.'*)

returns a list of all source files.

**Flatten**(*self, sequence*)
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBuildPath</td>
<td><code>self, files</code></td>
<td></td>
</tr>
<tr>
<td>Glob</td>
<td><code>self, pattern, ondisk=True, source=False, strings=False, exclude=None</code></td>
<td></td>
</tr>
<tr>
<td>Ignore</td>
<td><code>self, target, dependency</code></td>
<td>Ignore a dependency.</td>
</tr>
<tr>
<td>Literal</td>
<td><code>self, string</code></td>
<td></td>
</tr>
<tr>
<td>Local</td>
<td><code>self, *targets</code></td>
<td></td>
</tr>
<tr>
<td>NoCache</td>
<td><code>self, *targets</code></td>
<td>Tags a target so that it will not be cached</td>
</tr>
<tr>
<td>NoClean</td>
<td><code>self, *targets</code></td>
<td>Tags a target so that it will not be cleaned by -c</td>
</tr>
<tr>
<td>ParseConfig</td>
<td><code>self, command, function=None, unique=1</code></td>
<td>Use the specified function to parse the output of the command in order to modify the current environment. The 'command' can be a string or a list of strings representing a command and its arguments. 'Function' is an optional argument that takes the environment, the output of the command, and the unique flag. If no function is specified, MergeFlags, which treats the output as the result of a typical 'X-config' command (i.e. gtk-config), will merge the output into the appropriate variables.</td>
</tr>
</tbody>
</table>
Class Base Module SCons.Environment

### ParseDepends(self, filename, must_exist=None, only_one=0)

Parse a mkdep-style file for explicit dependencies. This is completely abusable, and should be unnecessary in the 'normal' case of proper SCons configuration, but it may help make the transition from a Make hierarchy easier for some people to swallow. It can also be genuinely useful when using a tool that can write a .d file, but for which writing a scanner would be too complicated.

### Platform(self, platform)

### Precious(self, *targets)

### Prepend(self, **kw)

Prepend values to existing construction variables in an Environment.

### PrependENVPath(self, name, newpath, envname='ENV', sep=':', delete_existing=1)

Prepend path elements to the path 'name' in the 'ENV' dictionary for this environment. Will only add any particular path once, and will normpath and normcase all paths to help assure this. This can also handle the case where the env variable is a list instead of a string.

If delete_existing is 0, a newpath which is already in the path will not be moved to the front (it will be left where it is).

### PrependUnique(self, delete_existing=0, **kw)

Prepend values to existing construction variables in an Environment, if they’re not already there. If delete_existing is 1, removes existing values first, so values move to front.

### Pseudo(self, *targets)

### PyPackageDir(self, modulename)
Replace(self, **kw)

Replace existing construction variables in an Environment with new construction variables and/or values.

ReplaceIxes(self, path, old_prefix, old_suffix, new_prefix, new_suffix)

Replace old_prefix with new_prefix and old_suffix with new_suffix.

env - Environment used to interpolate variables. path - the path that will be modified. old_prefix - construction variable for the old prefix. old_suffix - construction variable for the old suffix. new_prefix - construction variable for the new prefix. new_suffix - construction variable for the new suffix.

Repository(self, *dirs, **kw)

Requires(self, target, prerequisite)

Specify that 'prerequisite' must be built before 'target', (but 'target' does not actually depend on 'prerequisite' and need not be rebuilt if it changes).

SConsignFile(self, name='\'.sconsign\', dbm_module=None)

Scanner(self, *args, **kw)

SetDefault(self, **kw)

SideEffect(self, side_effect, target)

Tell scons that side_effects are built as side effects of building targets.

SourceCode(self, entry, builder)

Arrange for a source code builder for (part of) a tree.

SourceSignatures(self, type)
Split\((self, \text{arg})\)

This function converts a string or list into a list of strings or Nodes. This makes things easier for users by allowing files to be specified as a white-space separated list to be split.

The input rules are:

- A single string containing names separated by spaces. These will be split apart at the spaces.
- A single Node instance
- A list containing either strings or Node instances. Any strings in the list are not split at spaces.

In all cases, the function returns a list of Nodes and strings.

TargetSignatures\((self, \text{type})\)

Tool\((self, \text{tool}, toolpath=\text{None}, **kw)\)

Value\((self, \text{value}, built\_value=\text{None})\)

VariantDir\((self, \text{variant}\_\text{dir}, src\_\text{dir}, duplicate=1)\)

WhereIs\((self, \text{prog}, path=\text{None}, pathext=\text{None}, reject=\text{[]})\)

Find prog in the path.
Initialization of a basic SCons construction environment, including setting up special construction variables like BUILDER, PLATFORM, etc., and searching for and applying available Tools.

Note that we do not call the underlying base class (SubstitutionEnvironment) initialization, because we need to initialize things in a very specific order that doesn’t work with the much simpler base class initialization. Overrides: object.__init__

get_CacheDir(self)

get_builder(self, name)

Fetch the builder with the specified name from the environment.

get_factory(self, factory, default='File')

Return a factory function for creating Nodes for this construction environment.

get_scanner(self, skey)

Find the appropriate scanner given a key (usually a file suffix).

get_src_sig_type(self)

get_tgt_sig_type(self)

scanner_map_delete(self, kw=None)

Delete the cached scanner map (if we need to).

Inherited from SCons.Environment.SubstitutionEnvironment(Section 8.6)
AddMethod(), MergeFlags(), Override(), ParseFlags(), RemoveMethod(), __contains__(), __delitem__(), __eq__(), __getitem__(), __getitem__(), __setitem__(), arg2nodes(), backtick(), get(), gvars(), has_key(), items(), lvars(), subst(), subst_kw(), subst_list(), subst_path(), subst_target_source()

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasscheck__()

### 8.9.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
9 Module SCons.Errors

SCons.Errors

This file contains the exception classes used to handle internal and user errors in SCons.

9.1 Functions

```
convert_to_BuildError(status, exc_info=None)
```

Convert any return code a BuildError Exception.

The buildError.status we set here will normally be used as the exit status of
the 'scons' process.  **Parameters**

status: : can either be a return code or an Exception.

9.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Errors.py rel_3.0.0:4395:8972f6a2f699 2...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

9.3 Class BuildError

```
object exceptions.BaseException
      exceptions.Exception
      SCons.Errors.BuildError
```

Errors occurring while building.

BuildError have the following attributes:

**Information about the cause of the build error:** errstr : a description of the error message
status : the return code of the action that caused the build error. Must be set to a non-zero value even if the build error is not due to an action returning a non-zero returned code.

exitstatus: SCons exit status due to this build error. Must be nonzero unless due to an explicit Exit() call. Not always the same as status, since actions return a status code that should be respected, but SCons typically exits with 2 irrespective of the return value of the failed action.

filename: The name of the file or directory that caused the build error. Set to None if no files are associated with this error. This might be different from the target being built. For example, failure to create the directory in which the target file will appear. It can be None if the error is not due to a particular filename.

exc_info: Info about exception that caused the build error. Set to (None, None, None) if this build error is not due to an exception.

Information about the cause of the location of the error:

node: the error occurred while building this target node(s)

executor (the executor that caused the build to fail (might)

be None if the build failures is not due to the executor failing)

action (the action that caused the build to fail (might be)

None if the build failures is not due to the an action failure)

command (the command line for the action that caused the)

build to fail (might be None if the build failures is not due to the an action failure)

9.3.1 Methods

```
__init__(self, node=None, errstr='Unknown error', status=2,
         exitstatus=2, filename=None, executor=None, action=None, command=None,
         exc_info=(None, None, None))
```

Overrides:
object.__init__ (inherited documentation)

```
__str__(self)
```

Overrides: object.__str__ (inherited documentation)

Inherited from exceptions.Exception

```
__new__()
**Class InternalError**

Inherited from `exceptions.BaseException`

- `__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__unicode__()`

Inherited from `object`

- `__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 9.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Inherited from exceptions.BaseException</code></td>
<td><code>args</code>, <code>message</code></td>
</tr>
<tr>
<td><code>Inherited from object</code></td>
<td><code>__class__</code></td>
</tr>
</tbody>
</table>

**9.4 Class InternalError**

```
object
```

```
exceptions.BaseException
    exceptions.Exception  SCons.Errors.InternalError
```

**9.4.1 Methods**

Inherited from `exceptions.Exception`

- `__init__()`, `__new__()`

Inherited from `exceptions.BaseException`

- `__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

Inherited from `object`

- `__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**9.4.2 Properties**

93
9.5 Class UserError

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

Known Subclasses: SCons.SConf.SConfError, SCons.Warnings.Warning

9.5.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

9.5.2 Properties
9.6 Class StopError

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.StopError

9.6.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

9.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

9.7 Class EnvironmentError

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.EnvironmentError
9.7.1 Methods

*Inherited from exceptions.Exception*

\_\_init\_\_(), \_\_new\_\_()

*Inherited from exceptions.BaseException*

\_\_delattr\_\_(), \_\_getattribute\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

*Inherited from object*

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

9.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from exceptions.BaseException</em></td>
<td>args, message</td>
</tr>
<tr>
<td><em>Inherited from object</em></td>
<td>__class_</td>
</tr>
</tbody>
</table>

9.8 Class MSVCError

object

exceptions.BaseException

exceptions.Exception

exceptions.StandardError

exceptions.EnvironmentError

exceptions.IOError

SCons.Errors.MSVCError

9.8.1 Methods

*Inherited from exceptions.IOError*
Class ExplicitExit

Module SCons.Errors

__init__(), __new__()

Inherited from exceptions.EnvironmentError

__reduce__(), __str__()

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

9.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.EnvironmentError</td>
<td>errno, filename, strerror</td>
</tr>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

9.9 Class ExplicitExit

object ──>

exceptions.BaseException ──>

exceptions.Exception ──>

SCons.Errors.ExplicitExit

9.9.1 Methods

___init___(self, node=None, status=None, *args)

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception
__new__()  

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

9.9.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
10 Module SCons.Executor

SCons.Executor

A module for executing actions with specific lists of target and source Nodes.

10.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rfile(node)</code></td>
<td>A function to return the results of a Node’s <code>rfile()</code> method, if it exists, and the Node itself otherwise (if it’s a Value Node, e.g.).</td>
</tr>
<tr>
<td><code>execute_nothing(obj, target, kw)</code></td>
<td></td>
</tr>
<tr>
<td><code>execute_action_list(obj, target, kw)</code></td>
<td>Actually execute the action list.</td>
</tr>
<tr>
<td><code>execute_actions_str(obj)</code></td>
<td></td>
</tr>
<tr>
<td><code>execute_null_str(obj)</code></td>
<td></td>
</tr>
<tr>
<td><code>GetBatchExecutor(key)</code></td>
<td></td>
</tr>
<tr>
<td><code>AddBatchExecutor(key, executor)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_NullEnvironment()</code></td>
<td>Use singleton pattern for Null Environments.</td>
</tr>
</tbody>
</table>

10.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__revision__</code></td>
<td>Value: <code>src/engine/SCons/Executor.py rel_3.0.0:4395:8972f6a2f699...</code></td>
</tr>
</tbody>
</table>
Class Batch Module SCons.Executor

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nullenv</td>
<td>Value: None</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

10.3 Class Batch

object

SCons.Executor.Batch

Remembers exact association between targets and sources of executor.

10.3.1 Methods

```python
__init__(self, targets=[], sources=[])  # x.__init__(...) initializes x; see help(type(x)) for signature
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

10.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sources</td>
<td></td>
</tr>
<tr>
<td>targets</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object

```python
__class__
```
10.4 Class TSList

A class that implements $TARGETS or $SOURCES expansions by wrapping an executor Method. This class is used in the Executor.lvars() to delay creation of NodeList objects until they’re needed.

Note that we subclass collections.UserList purely so that the is_Sequence() function will identify an object of this class as a list during variable expansion. We’re not really using any collections.UserList methods in practice.

10.4.1 Methods

```python
__init__(self, func)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ (inherited documentation)
```

```python
__getattr__(self, attr)
```

```python
__getitem__(self, i)
Overrides: _abcoll.Sequence.__getitem__
```
__getslice__(self, i, j)

Overrides: UserList.UserList.__getslice__

__str__(self)

str(x)  Overrides: object.__str__

extit(inherited documentation)

__repr__(self)

repr(x)  Overrides: object.__repr__

extit(inherited documentation)

Inherited from UserList.UserList

__add__(), __cmp__(), __contains__(), __delitem__(), __delslice__(),
__eq__(), __ge__(), __gt__(), __iadd__(), __imul__(), __le__(), __len__(),
__lt__(), __mul__(), __ne__(), __radd__(), __rmul__(), __setitem__(),
__setslice__(), append(), count(), extend(), index(), insert(), pop(), remove(),
reverse(), sort()

Inherited from _abcoll.Sequence

__iter__(), __reversed__()

Inherited from _abcoll.Sized

__subclasshook__()  

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __setattr__(), __sizeof__()

10.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

10.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from UserList.UserList</td>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
</tr>
</tbody>
</table>
10.5 Class TSObject

A class that implements $TARGET or $SOURCE expansions by wrapping an Executor method.

10.5.1 Methods

```python
__init__(self, func)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```python
__getattr__(self, attr)
```

```python
__str__(self)
str(x) Overrides: object.__str__ extit(inherited documentation)
```

```python
__repr__(self)
repr(x) Overrides: object.__repr__ extit(inherited documentation)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()
```

10.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object

10.6 Class Executor

A class that implements $TARGET or $SOURCE expansions by wrapping an Executor method.

```
```
A class for controlling instances of executing an action.

This largely exists to hold a single association of an action, environment, list of environment override dictionaries, targets and sources for later processing as needed.

10.6.1 Methods

___init___(self, action, env=None, overridelist=\[
\], targets=\[
], sources=\[
], builder_kw=\[
])

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides:
object.__init__ extit(inherited documentation)

get_lvars(self)

get_action_targets(self)

set_action_list(self, action)

get_action_list(self)

get_all_targets(self)

Returns all targets for all batches of this Executor.

get_all_sources(self)

Returns all sources for all batches of this Executor.

get_all_children(self)

Returns all unique children (dependencies) for all batches of this Executor.

The Taskmaster can recognize when it’s already evaluated a Node, so we don’t have to make this list unique for its intended canonical use case, but we expect there to be a lot of redundancy (long lists of batched .cc files #including the same .h files over and over), so removing the duplicates once up front should save the Taskmaster a lot of work.
### get_all_prerequisites(self)

Returns all unique (order-only) prerequisites for all batches of this Executor.

### get_action_side_effects(self)

Returns all side effects for all batches of this Executor used by the underlying Action.

### get_build_env(self)

Fetch or create the appropriate build Environment for this Executor.

### get_build_scanner_path(self, scanner)

Fetch the scanner path for this executor’s targets and sources.

### get_kw(self, kw={})

### __call__(self, target, **kw)

### cleanup(self)

### add_sources(self, sources)

Add source files to this Executor’s list. This is necessary for "multi" Builders that can be called repeatedly to build up a source file list for a given target.

### get_sources(self)
**add_batch**(self, targets, sources)

Add pair of associated target and source to this Executor’s list. This is necessary for "batch" Builders that can be called repeatedly to build up a list of matching target and source files that will be used in order to update multiple target files at once from multiple corresponding source files, for tools like MSVC that support it.

**prepare**(self)

Preparatory checks for whether this Executor can go ahead and (try to) build its targets.

**add_pre__action**(self, action)

**add_post__action**(self, action)

**__str__**(self)

str(x)  Overrides: object.__str__ extit(inherited documentation)

**nullify**(self)

**get__contents**(self)

Fetch the signature contents. This is the main reason this class exists, so we can compute this once and cache it regardless of how many target or source Nodes there are.

**get__timestamp**(self)

Get a time stamp for this Executor. We don’t have one, of course (only files do), but this is the interface used by the timestamp module.

**scan__targets**(self, scanner)

**scan__sources**(self, scanner)
**scan**(self, scanner, node_list)

Scan a list of this Executor’s files (targets or sources) for implicit dependencies and update all of the targets with them. This essentially short-circuits an N*M scan of the sources for each individual target, which is a hell of a lot more efficient.

**get_unignored_sources**(self, node, ignore=())

**get_implicit_deps**(self)

Return the executor’s implicit dependencies, i.e. the nodes of the commands to be executed.

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()
10.7 Class NullEnvironment

**object**

SCons.Util.Null →

SCons.Executor.NullEnvironment

### 10.7.1 Methods

**get_CacheDir**(self)

*Inherited from SCons.Util.Null (Section 36.16)*

__bool__(), __call__(), __delattr__(), __getattr__(), __init__(), __new__(), __nonzero__(), __repr__(), __setattr__()

*Inherited from object*

__format__(), __getattribute__(), __hash__(), __reduce__(), __reduce_ex__(), __sizeof__(), __str__(), __subclasshook__()

### 10.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td>Inherited from object</td>
</tr>
</tbody>
</table>

10.8 Class Null

**object**

SCons.Executor.Null

A null Executor, with a null build Environment, that does nothing when the rest of the methods call it.

This might be able to disappear when we refactor things to disassociate Builders from Nodes entirely, so we’re not going to worry about unit tests for this—at least for now.
10.8.1 Methods

```python
__init__(self, *args, **kw)
```
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ extit(inherited documentation)

```python
get_build_env(self)
```

```python
get_build_scanner_path(self)
```

```python
cleanup(self)
```

```python
prepare(self)
```

```python
get_unignored_sources(self, *args, **kw)
```

```python
get_action_targets(self)
```

```python
get_action_list(self)
```

```python
get_all_targets(self)
```

```python
get_all_sources(self)
```

```python
get_all_children(self)
```

```python
get_all_prerequisites(self)
```

```python
get_action_side_effects(self)
```

```python
__call__(self, *args, **kw)
```

```python
get_contents(self)
```

```python
add_pre_action(self, action)
```

```python
add_post_action(self, action)
```

**set_action_list** *(self, action)*

**Inherited from object**

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 10.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action_list</td>
<td></td>
</tr>
<tr>
<td>batches</td>
<td></td>
</tr>
<tr>
<td>builder_kw</td>
<td></td>
</tr>
<tr>
<td>env</td>
<td></td>
</tr>
<tr>
<td>lvars</td>
<td></td>
</tr>
<tr>
<td>overridelist</td>
<td></td>
</tr>
<tr>
<td>post_actions</td>
<td></td>
</tr>
<tr>
<td>pre_actions</td>
<td></td>
</tr>
</tbody>
</table>

### Inherited from object

```python
___class___
```
11 Module SCons.Job

SCons.Job

This module defines the Serial and Parallel classes that execute tasks to complete a build. The Jobs class provides a higher level interface to start, stop, and wait on jobs.

11.1 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: src/engine/SCons/Job.py rel_3.0.0:4395:8972f6a2f699 2017...</td>
</tr>
<tr>
<td>explicit_stack_size</td>
<td>Value: None</td>
</tr>
<tr>
<td>default_stack_size</td>
<td>Value: 256</td>
</tr>
<tr>
<td>interrupt_msg</td>
<td>Value: 'Build interrupted.'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

11.2 Class InterruptState

```
object └── SCons.Job.InterruptState
```

11.2.1 Methods

```
__init__(self)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```
set(self)
```

```
__call__(self)

Inherited from object
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>delattr</strong></td>
<td></td>
</tr>
<tr>
<td><strong>format</strong></td>
<td></td>
</tr>
<tr>
<td><strong>getattr</strong></td>
<td></td>
</tr>
<tr>
<td><strong>hash</strong></td>
<td></td>
</tr>
<tr>
<td><strong>new</strong></td>
<td></td>
</tr>
<tr>
<td><strong>reduce</strong></td>
<td></td>
</tr>
<tr>
<td><strong>reduce_ex</strong></td>
<td></td>
</tr>
<tr>
<td><strong>repr</strong></td>
<td></td>
</tr>
<tr>
<td><strong>setattr</strong></td>
<td></td>
</tr>
<tr>
<td><strong>sizeof</strong></td>
<td></td>
</tr>
<tr>
<td><strong>str</strong></td>
<td></td>
</tr>
<tr>
<td><strong>subclasshook</strong></td>
<td></td>
</tr>
</tbody>
</table>
11.2.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

11.3 Class Jobs

```
object    
SCons.Job.Jobs
```

An instance of this class initializes N jobs, and provides methods for starting, stopping, and waiting on all N jobs.

11.3.1 Methods

```
__init__(self, num, taskmaster)
```

Create 'num' jobs using the given taskmaster.

If 'num' is 1 or less, then a serial job will be used, otherwise a parallel job with 'num' worker threads will be used.

The 'num_jobs' attribute will be set to the actual number of jobs allocated. If more than one job is requested but the Parallel class can’t do it, it gets reset to 1. Wrapping interfaces that care should check the value of 'num_jobs' after initialization. Overrides: object.__init__

```
run(self, postfunc=__builtins__.function object>)
```

Run the jobs.

postfunc() will be invoked after the jobs has run. It will be invoked even if the jobs are interrupted by a keyboard interrupt (well, in fact by a signal such as either SIGINT, SIGTERM or SIGHUP). The execution of postfunc() is protected against keyboard interrupts and is guaranteed to run to completion.
were_interrupted(self)

Returns whether the jobs were interrupted by a signal.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

11.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

11.4 Class Serial

object

SCons.Job.Serial

This class is used to execute tasks in series, and is more efficient than Parallel, but is only appropriate for non-parallel builds. Only one instance of this class should be in existence at a time.

This class is not thread safe.

11.4.1 Methods

__init__(self, taskmaster)

Create a new serial job given a taskmaster.

The taskmaster’s next_task() method should return the next task that needs to be executed, or None if there are no more tasks. The taskmaster’s executed() method will be called for each task when it is successfully executed, or failed() will be called if it failed to execute (e.g. execute() raised an exception). Overrides: object.__init__

113
**start**(self)

Start the job. This will begin pulling tasks from the taskmaster and executing them, and return when there are no more tasks. If a task fails to execute (i.e. execute() raises an exception), then the job will stop.

**Inherited from object**

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 11.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 11.5 Class Worker

```
object
```

```
threading._Verbose
```

```
threading.Thread
```

**SCons.Job.Worker**

A worker thread waits on a task to be posted to its request queue, dequeues the task, executes it, and posts a tuple including the task and a boolean indicating whether the task executed successfully.
11.5.1 Methods

```python
__init__(self, requestQueue, resultsQueue, interrupted)
```

This constructor should always be called with keyword arguments. Arguments are:

- `group` should be None; reserved for future extension when a ThreadGroup class is implemented.
- `target` is the callable object to be invoked by the run() method. Defaults to None, meaning nothing is called.
- `name` is the thread name. By default, a unique name is constructed of the form "Thread-N" where N is a small decimal number.
- `args` is the argument tuple for the target invocation. Defaults to (()).
- `kwargs` is a dictionary of keyword arguments for the target invocation. Defaults to {}.

If a subclass overrides the constructor, it must make sure to invoke the base class constructor (Thread.__init__() ) before doing anything else to the thread. Overrides: object.__init__ extit(inherited documentation)

```python
run(self)
```

Method representing the thread’s activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object’s constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively. Overrides: threading.Thread.run extit(inherited documentation)

Inherited from threading.Thread

- __repr__(), getName(), isAlive(), isDaemon(), is_alive(), join(), setDaemon(), setName(), start()

Inherited from object

- __delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
### 11.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from threading.Thread</td>
<td>daemon, ident, name</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

### 11.6 Class ThreadPool

```
object └── SCons.Job.ThreadPool
```

This class is responsible for spawning and managing worker threads.

#### 11.6.1 Methods

**__init__(self, num, stack_size, interrupted)**

Create the request and reply queues, and 'num' worker threads.

One must specify the stack size of the worker threads. The stack size is specified in kilobytes. Overrides: object.__init__

**put(self, task)**

Put task into request queue.

**get(self)**

Remove and return a result tuple from the results queue.

**preparation_failed(self, task)**
**cleanup**(*self*)

Shuts down the thread pool, giving each worker thread a chance to shut down gracefully.

*Inherited from object*

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

### 11.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 11.7 Class Parallel

```
object ->
SCons.Job.Parallel
```

This class is used to execute tasks in parallel, and is somewhat less efficient than Serial, but is appropriate for parallel builds.

This class is thread safe.
11.7.1 Methods

__init__(self, taskmaster, num, stack_size)

Create a new parallel job given a taskmaster.

The taskmaster’s next_task() method should return the next task that needs to be executed, or None if there are no more tasks. The taskmaster’s executed() method will be called for each task when it is successfully executed, or failed() will be called if the task failed to execute (i.e. execute() raised an exception).

Note: calls to taskmaster are serialized, but calls to execute() on distinct tasks are not serialized, because that is the whole point of parallel jobs: they can execute multiple tasks simultaneously. Overrides: object.__init__

start(self)

Start the job. This will begin pulling tasks from the taskmaster and executing them, and return when there are no more tasks. If a task fails to execute (i.e. execute() raises an exception), then the job will stop.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

11.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
12 Module SCons.Memoize

Memoizer

A decorator-based implementation to count hits and misses of the computed values that various methods cache in memory.

Use of this module assumes that wrapped methods be coded to cache their values in a consistent way. In particular, it requires that the class uses a dictionary named "_memo" to store the cached values.

Here is an example of wrapping a method that returns a computed value, with no input parameters:

```python
@SCons.Memoize.CountMethodCall
def foo(self):
    try: # Memoization
        return self._memo['foo'] # Memoization
    except KeyError: # Memoization
        pass # Memoization

    result = self.compute_foo_value()
    self._memo['foo'] = result # Memoization
    return result
```

Here is an example of wrapping a method that will return different values based on one or more input arguments:

```python
def _bar_key(self, argument): # Memoization
    return argument # Memoization

@SCons.Memoize.CountDictCall(_bar_key)
def bar(self, argument):
    memo_key = argument # Memoization
    try:
        memo_dict = self._memo['bar'] # Memoization
    except KeyError:
        memo_dict = {} # Memoization
    else:
        try:
            self._memo['dict'] = memo_dict # Memoization

    return result # Memoization
```

119
return memo_dict[memo_key]  # Memoization
except KeyError:
    pass  # Memoization

result = self.compute_bar_value(argument)  # Memoization

memo_dict[memo_key] = result  # Memoization

return result

Deciding what to cache is tricky, because different configurations can have radically different performance tradeoffs, and because the tradeoffs involved are often so non-obvious. Consequently, deciding whether or not to cache a given method will likely be more of an art than a science, but should still be based on available data from this module. Here are some VERY GENERAL guidelines about deciding whether or not to cache return values from a method that’s being called a lot:

-- The first question to ask is, "Can we change the calling code so this method isn’t called so often?" Sometimes this can be done by changing the algorithm. Sometimes the caller should be memoized, not the method you’re looking at.

-- The memoized function should be timed with multiple configurations to make sure it doesn’t inadvertently slow down some other configuration.

-- When memoizing values based on a dictionary key composed of input arguments, you don’t need to use all of the arguments if some of them don’t affect the return values.

12.1 Functions

Dump(title=None)

Dump the hit/miss count for all the counters collected so far.

EnableMemoization()

CountMethodCall(fn)

Decorator for counting memoizer hits/misses while retrieving a simple value in a class method. It wraps the given method fn and uses a CountValue object to keep track of the caching statistics. Wrapping gets enabled by calling EnableMemoization().
# CountDictCall(keyfunc)

Decorator for counting memoizer hits/misses while accessing dictionary values with a key-generating function. Like CountMethodCall above, it wraps the given method fn and uses a CountDict object to keep track of the caching statistics. The dict-key function keyfunc has to get passed in the decorator call and gets stored in the CountDict instance. Wrapping gets enabled by calling EnableMemoization().

## 12.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Memoize.py rel_3.0.0:4395:8972f6a2f699 ...</td>
</tr>
<tr>
<td><strong>doc</strong></td>
<td>Value: &quot;&quot;&quot;Memo...</td>
</tr>
<tr>
<td>use_memoizer</td>
<td>Value: None</td>
</tr>
<tr>
<td>CounterList</td>
<td>Value: {}</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

## 12.3 Class Counter

```
object —

SCons.Memoize.Counter
```

**Known Subclasses:** SCons.Memoize.CountDict, SCons.Memoize.CountValue

Base class for counting memoization hits and misses.

We expect that the initialization in a matching decorator will fill in the correct class name and method name that represents the name of the function being counted.

### 12.3.1 Methods

```
__init__(self, cls_name, method_name)
```

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__

```
key(self)
```
display(self)

__eq__(self, other)

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

### 12.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 12.4 Class CountValue

object  

SCons.Memoize.Counter  

SCons.Memoize.CountValue

A counter class for simple, atomic memoized values.

A CountValue object should be instantiated in a decorator for each of the class’s methods that memoizes its return value by simply storing the return value in its _memo dictionary.

#### 12.4.1 Methods

**count**(self, *args, **kw)

Counts whether the memoized value has already been set (a hit) or not (a miss).

**Inherited from SCons.Memoize.Counter** *(Section 12.3)*

__eq__(), __init__(), display(), key()

**Inherited from object**
Class CountDict

Module SCons.Memoize

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

12.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

12.5 Class CountDict

object

SCons.Memoize.Counter

SCons.Memoize.CountDict

A counter class for memoized values stored in a dictionary, with keys based on the method’s input arguments.

A CountDict object is instantiated in a decorator for each of the class’s methods that memoizes its return value in a dictionary, indexed by some key that can be computed from one or more of its input arguments.

12.5.1 Methods

__init__(self, cls_name, method_name, keymaker)

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__

count(self, *args, **kw)

Counts whether the computed key value is already present in the memoization dictionary (a hit) or not (a miss).

Inherited from SCons.Memoize.Counter(Section 12.3)

__eq__(), display(), key()

Inherited from object
Class CountDict

Module SCons.Memoize

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
13 Package SCons.Node

SCons.Node

The Node package for the SCons software construction utility.

This is, in many ways, the heart of SCons.

A Node is where we encapsulate all of the dependency information about any thing that SCons can build, or about any thing which SCons can use to build some other thing. The canonical "thing," of course, is a file, but a Node can also represent something remote (like a web page) or something completely abstract (like an Alias).

Each specific type of "thing" is specifically represented by a subclass of the Node base class: Node.FS.File for files, Node.Alias for aliases, etc. Dependency information is kept here in the base class, and information specific to files/aliases/etc. is in the subclass. The goal, if we’ve done this correctly, is that any type of 'thing' should be able to depend on any other type of 'thing.'

13.1 Modules

  (Section 14, p. 133)
- FS: scons.Node.FS  
  (Section 15, p. 139)
- Python: scons.Node.Python  
  (Section 16, p. 186)

13.2 Functions

<table>
<thead>
<tr>
<th><strong>classname</strong>(obj)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotate(node)</td>
</tr>
<tr>
<td><strong>is_derived_none</strong>(node)</td>
</tr>
<tr>
<td><strong>is_derived_node</strong>(node)</td>
</tr>
</tbody>
</table>

Returns true if this node is derived (i.e. built).

| **exists_none**(node) |
exists_always(node)

exists_base(node)

exists_entry(node)

Return if the Entry exists. Check the file system to see what we should turn into first. Assume a file if there’s no directory.

exists_file(node)

rexists_none(node)

rexists_node(node)

rexists_base(node)

get_contents_none(node)

get_contents_entry(node)

Fetch the contents of the entry. Returns the exact binary contents of the file.

get_contents_dir(node)

Return content signatures and names of all our children separated by new-lines. Ensure that the nodes are sorted.

get_contents_file(node)

target_from_source_none(node, prefix, suffix, splitext)

target_from_source_base(node, prefix, suffix, splitext)
changed_since_last_build_node(node, target, prev_ni)

Must be overridden in a specific subclass to return True if this Node (a dependency) has changed since the last time it was used to build the specified target. prev_ni is this Node’s state (for example, its file timestamp, length, maybe content signature) as of the last time the target was built.

Note that this method is called through the dependency, not the target, because a dependency Node must be able to use its own logic to decide if it changed. For example, File Nodes need to obey if we’re configured to use timestamps, but Python Value Nodes never use timestamps and always use the content. If this method were called through the target, then each Node’s implementation of this method would have to have more complicated logic to handle all the different Node types on which it might depend.

changed_since_last_build_alias(node, target, prev_ni)

changed_since_last_build_entry(node, target, prev_ni)

changed_since_last_build_state_changed(node, target, prev_ni)

decide_source(node, target, prev_ni)

decide_target(node, target, prev_ni)

changed_since_last_build_python(node, target, prev_ni)

store_info_pass(node)

store_info_file(node)

get_children(node, parent)

ignore_cycle(node, stack)

do_nothing(node, parent)

13.3 Variables
13.4 Class NodeInfoBase

The generic base class for signature information for a Node.

Node subclasses should subclass NodeInfoBase to provide their own logic for dealing with their own Node-specific signature information.
13.4.1 Methods

___getstate___(self)

Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a '__dict__' slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class.

___setstate___(self, state)

Restore the attributes from a pickled state. The version is discarded.

convert(self, node, val)

format(self, field_list=None, names=0)

merge(self, other)

Merge the fields of another object into this object. Already existing information is overwritten by the other instance's data. WARNING: If a '__dict__' slot is added, it should be updated instead of replaced.

update(self, node)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___init___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

13.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><em><strong>class</strong></em></td>
<td></td>
</tr>
</tbody>
</table>

13.4.3 Class Variables
### Class BuildInfoBase

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
</tbody>
</table>

#### 13.5 Class BuildInfoBase

The generic base class for build information for a Node.

This is what gets stored in a .sconsign file for each target file. It contains a NodeInfo instance for this node (signature information that’s specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.

#### 13.5.1 Methods

**__getstate__(self)**

Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a '__dict__' slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class.

**__init__(self)**

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ (inherited documentation)

**__setstate__(self, state)**

Restore the attributes from a pickled state.
merge(self, other)

Merge the fields of another object into this object. Already existing information is overwritten by the other instance’s data. WARNING: If a ‘__dict__’ slot is added, it should be updated instead of replaced.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

13.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bact</td>
<td></td>
</tr>
<tr>
<td>bactsig</td>
<td></td>
</tr>
<tr>
<td>bdepends</td>
<td></td>
</tr>
<tr>
<td>bdependsigs</td>
<td></td>
</tr>
<tr>
<td>bimplicit</td>
<td></td>
</tr>
<tr>
<td>bimplicitsigs</td>
<td></td>
</tr>
<tr>
<td>bsources</td>
<td></td>
</tr>
<tr>
<td>bsourcesigs</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object

__class__

13.5.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
</tbody>
</table>

13.6 Class Node

object

SCons.Node.Node


The base Node class, for entities that we know how to build, or use to build other Nodes.
### 13.6.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Decider(self, function)</code></td>
<td>Return a user-defined tag.</td>
</tr>
<tr>
<td><code>GetTag(self, key)</code></td>
<td>Return a user-defined tag.</td>
</tr>
<tr>
<td><code>Tag(self, key, value)</code></td>
<td>Add a user-defined tag.</td>
</tr>
<tr>
<td><code>__init__(self)</code></td>
<td>Adds dependencies.</td>
</tr>
<tr>
<td><code>add_dependency(self, depend)</code></td>
<td>Adds dependencies.</td>
</tr>
<tr>
<td><code>add_ignore(self, depend)</code></td>
<td>Adds dependencies to ignore.</td>
</tr>
<tr>
<td><code>add_prerequisite(self, prerequisite)</code></td>
<td>Adds prerequisites</td>
</tr>
<tr>
<td><code>add_source(self, source)</code></td>
<td>Adds sources.</td>
</tr>
<tr>
<td><code>add_to_implicit(self, deps)</code></td>
<td>Adds sources.</td>
</tr>
</tbody>
</table>
### Class Node Package SCons.Node

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>add_to_waiting_parents</strong>(<em>self</em>, <em>node</em>)</td>
<td>Returns the number of nodes added to our waiting parents list: 1 if we add a unique waiting parent, 0 if not. (Note that the returned values are intended to be used to increment a reference count, so don’t think you can 'clean up' this function by using True and False instead…)</td>
</tr>
<tr>
<td><strong>add_to_waiting_s_e</strong>(<em>self</em>, <em>node</em>)</td>
<td></td>
</tr>
<tr>
<td><strong>add_wkid</strong>(<em>self</em>, <em>wkid</em>)</td>
<td>Add a node to the list of kids waiting to be evaluated</td>
</tr>
<tr>
<td><strong>all_children</strong>(<em>self</em>, <em>scan=1</em>)</td>
<td>Return a list of all the node’s direct children.</td>
</tr>
<tr>
<td><strong>alter_targets</strong>(<em>self</em>)</td>
<td>Return a list of alternate targets for this Node.</td>
</tr>
<tr>
<td><strong>build</strong>(<em>self</em>, *<em>kw</em>)</td>
<td>Actually build the node. This is called by the Taskmaster after it’s decided that the Node is out-of-date and must be rebuilt, and after the prepare() method has gotten everything, uh, prepared. This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in built().</td>
</tr>
<tr>
<td><strong>builder_set</strong>(<em>self</em>, <em>builder</em>)</td>
<td></td>
</tr>
</tbody>
</table>
**built**(self)

Called just after this node is successfully built.

**changed**(self, node=None, allowcache=False)

Returns if the node is up-to-date with respect to the BuildInfo stored last time it was built. The default behavior is to compare it against our own previously stored BuildInfo, but the stored BuildInfo from another Node (typically one in a Repository) can be used instead.

Note that we now *always* check every dependency. We used to short-circuit the check by returning as soon as we detected any difference, but we now rely on checking every dependency to make sure that any necessary Node information (for example, the content signature of an #included .h file) is updated.

The allowcache option was added for supporting the early release of the executor/builder structures, right after a File target was built. When set to true, the return value of this changed method gets cached for File nodes. Like this, the executor isn’t needed any longer for subsequent calls to changed().

@see: FS.File.changed(), FS.File.release_target_info()

**children**(self, scan=1)

Return a list of the node’s direct children, minus those that are ignored by this node.

**children_are_up_to_date**(self)

Alternate check for whether the Node is current: If all of our children were up-to-date, then this Node was up-to-date, too.

clear(self)

Completely clear a Node of all its cached state (so that it can be re-evaluated by interfaces that do continuous integration builds).

clear_memoized_values(self)

del_binfo(self)

Delete the build info from this node.

disambiguate(self, must_exist=None)

env_set(self, env, safe=0)

executor_cleanup(self)

Let the executor clean up any cached information.

exists(self)

Does this node exists?

explain(self)

for_signature(self)

Return a string representation of the Node that will always be the same for this particular Node, no matter what. This is by contrast to the __str__() method, which might, for instance, return a relative path for a file Node. The purpose of this method is to generate a value to be used in signature calculation for the command line used to build a target, and we use this method instead of str() to avoid unnecessary rebuilds. This method does not need to return something that would actually work in a command line; it can return any kind of nonsense, so long as it does not change.
get_abspath(self)

Return an absolute path to the Node. This will return simply str(Node) by
default, but for Node types that have a concept of relative path, this might
return something different.

get_binfo(self)

Fetch a node’s build information.

node - the node whose sources will be collected
    cache - alternate node to use
    for the signature cache
    returns - the build signature

This no longer handles the recursive descent of the node’s children’s
signatures. We expect that they’re already built and updated by someone else,
if that’s what’s wanted.

get_build_env(self)

Fetch the appropriate Environment to build this node.

get_build_scanner_path(self, scanner)

Fetch the appropriate scanner path for this node.

g_get_builder(self, default_builder=None)

Return the set builder, or a specified default value.

g_get_cachedir_csig(self)

g_get_contents(self)

Fetch the contents of the entry.

g_get_csig(self)
get_env(self)

get_env_scanner(self, env, kw={})

get_executor(self, create=1)

Fetch the action executor for this node. Create one if there isn’t already one, and requested to do so.

get_found_includes(self, env, scanner, path)

Return the scanned include lines (implicit dependencies) found in this node.

The default is no implicit dependencies. We expect this method to be overridden by any subclass that can be scanned for implicit dependencies.

getImplicit_deps(self, env, initial_scanner, path_func, kw={})

Return a list of implicit dependencies for this node.

This method exists to handle recursive invocation of the scanner on the implicit dependencies returned by the scanner, if the scanner’s recursive flag says that we should.

get_ninfo(self)

getSource_scanner(self, node)

Fetch the source scanner for the specified node

NOTE: "self" is the target being built, "node" is the source file for which we want to fetch the scanner.

Implies self.has_builder() is true; again, expect to only be called from locations where this is already verified.

This function may be called very often; it attempts to cache the scanner found to improve performance.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get_state(self)</code></td>
<td>Fetch the stored implicit dependencies</td>
</tr>
<tr>
<td><code>get_stored_implicit(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_stored_info(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_string(self, for_signature)</code></td>
<td>This is a convenience function designed primarily to be used in command generators (i.e., CommandGeneratorActions or Environment variables that are callable), which are called with a for_signature argument that is nonzero if the command generator is being called to generate a signature for the command line, which determines if we should rebuild or not. Such command generators should use this method in preference to str(Node) when converting a Node to a string, passing in the for_signature parameter, such that we will call Node.for_signature() or str(Node) properly, depending on whether we are calculating a signature or actually constructing a command line.</td>
</tr>
<tr>
<td><code>get_subst_proxy(self)</code></td>
<td>This method is expected to return an object that will function exactly like this Node, except that it implements any additional special features that we would like to be in effect for Environment variable substitution. The principle use is that some Nodes would like to implement a <strong>getattr</strong>() method, but putting that in the Node type itself has a tendency to kill performance. We instead put it in a proxy and return it from this method. It is legal for this method to return self if no new functionality is needed for Environment substitution.</td>
</tr>
<tr>
<td><code>get_suffix(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_target_scanner(self)</code></td>
<td></td>
</tr>
</tbody>
</table>
### `has_builder(self)`

Return whether this Node has a builder or not.

In Boolean tests, this turns out to be a *lot* more efficient than simply examining the builder attribute directly (`if node.builder: ...`). When the builder attribute is examined directly, it ends up calling `__getattr__` for both the `__len__` and `__nonzero__` attributes on instances of our `Builder Proxy` class(es), generating a bazillion extra calls and slowing things down immensely.

### `has_explicit_builder(self)`

Return whether this Node has an explicit builder.

This allows an internal Builder created by SCons to be marked non-explicit, so that it can be overridden by an explicit builder that the user supplies (the canonical example being directories).

### `is_derived(self)`

Returns true if this node is derived (i.e. built).

This should return true only for nodes whose path should be in the variant directory when `duplicate=0` and should contribute their build signatures when they are used as source files to other derived files. For example: source with source builders are not derived in this sense, and hence should not return true.

### `is_literal(self)`

Always pass the string representation of a Node to the command interpreter literally.

### `is_up_to_date(self)`

Default check for whether the Node is current: unknown Node subtypes are always out of date, so they will always get built.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>make_ready(self)</code></td>
<td>Get a Node ready for evaluation.</td>
</tr>
<tr>
<td></td>
<td>This is called before the Taskmaster decides if the Node is up-to-date or not.</td>
</tr>
<tr>
<td></td>
<td>Overriding this method allows for a Node subclass to be disambiguated if necessary, or for an implicit source builder to be attached.</td>
</tr>
<tr>
<td><code>missing(self)</code></td>
<td>Return whether this Node has a builder or not.</td>
</tr>
<tr>
<td></td>
<td>In Boolean tests, this turns out to be a lot more efficient than simply examining the builder attribute directly ('if node.builder: ...'). When the builder attribute is examined directly, it ends up calling <code>__getattr__</code> for both the <code>__len__</code> and <code>__nonzero__</code> attributes on instances of our Builder Proxy class(es), generating a bazillion extra calls and slowing things down immensely.</td>
</tr>
<tr>
<td><code>new_binfo(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>new_ninfo(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>postprocess(self)</code></td>
<td>Clean up anything we don’t need to hang onto after we’ve been built.</td>
</tr>
</tbody>
</table>
**prepare***(self)***

Prepare for this Node to be built.

This is called after the Taskmaster has decided that the Node is out-of-date and must be rebuilt, but before actually calling the method to build the Node.

This default implementation checks that explicit or implicit dependencies either exist or are derived, and initializes the BuildInfo structure that will hold the information about how this node is, uh, built.

(The existence of source files is checked separately by the Executor, which aggregates checks for all of the targets built by a specific action.)

Overriding this method allows for a Node subclass to remove the underlying file from the file system. Note that subclass methods should call this base class method to get the child check and the BuildInfo structure.

**push_to_cache**(self)

Try to push a node into a cache

**release_target_info**(self)

Called just after this node has been marked up-to-date or was built completely.

This is where we try to release as many target node infos as possible for clean builds and update runs, in order to minimize the overall memory consumption.

By purging attributes that aren’t needed any longer after a Node (=File) got built, we don’t have to care that much how many KBytes a Node actually requires...as long as we free the memory shortly afterwards.

@see: built() and File.release_target_info()

**remove**(self)

Remove this Node: no-op by default.
**render_include_tree(self)**

Return a text representation, suitable for displaying to the user, of the include tree for the sources of this node.

**reset_executor(self)**

Remove cached executor; forces recompute when needed.

**retrieve_from_cache(self)**

Try to retrieve the node’s content from a cache

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in built().

Returns true if the node was successfully retrieved.

**rexists(self)**

Does this node exist locally or in a repository?

**scan(self)**

Scan this node’s dependents for implicit dependencies.

**scanner_key(self)**

**select.Scanner(self, scanner)**

Selects a scanner for this Node.

This is a separate method so it can be overridden by Node subclasses (specifically, Node.FS.Dir) that must use their own Scanner and don’t select one the Scanner.Selector that’s configured for the target.
**Class Node**

**Package SCons.Node**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>set_always_build(self, always_build=1)</code></td>
<td>Set the Node’s always_build value.</td>
</tr>
<tr>
<td><code>set_executor(self, executor)</code></td>
<td>Set the action executor for this node.</td>
</tr>
<tr>
<td><code>set_explicit(self, is_explicit)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_nocache(self, nocache=1)</code></td>
<td>Set the Node’s nocache value.</td>
</tr>
<tr>
<td><code>set_noclean(self, noclean=1)</code></td>
<td>Set the Node’s noclean value.</td>
</tr>
<tr>
<td><code>set_precious(self, precious=1)</code></td>
<td>Set the Node’s precious value.</td>
</tr>
<tr>
<td><code>set_pseudo(self, pseudo=True)</code></td>
<td>Set the Node’s precious value.</td>
</tr>
<tr>
<td><code>set_specific_source(self, source)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_state(self, state)</code></td>
<td></td>
</tr>
<tr>
<td><code>visited(self)</code></td>
<td>Called just after this node has been visited (with or without a build).</td>
</tr>
</tbody>
</table>
**Inherited from object**

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(),
___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(),
___str___(), ___subclasshook___() 

13.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>always_build</td>
<td></td>
</tr>
<tr>
<td>attributes</td>
<td></td>
</tr>
<tr>
<td>binfo</td>
<td></td>
</tr>
<tr>
<td>builder</td>
<td></td>
</tr>
<tr>
<td>cached</td>
<td></td>
</tr>
<tr>
<td>changed_since_last_buil-d</td>
<td></td>
</tr>
<tr>
<td>depends</td>
<td></td>
</tr>
<tr>
<td>depends_set</td>
<td></td>
</tr>
<tr>
<td>env</td>
<td></td>
</tr>
<tr>
<td>executor</td>
<td></td>
</tr>
<tr>
<td>ignore</td>
<td></td>
</tr>
<tr>
<td>ignore_set</td>
<td></td>
</tr>
<tr>
<td>implicit</td>
<td></td>
</tr>
<tr>
<td>implicit_set</td>
<td></td>
</tr>
<tr>
<td>includes</td>
<td></td>
</tr>
<tr>
<td>is_explicit</td>
<td></td>
</tr>
<tr>
<td>linked</td>
<td></td>
</tr>
<tr>
<td>ninfo</td>
<td></td>
</tr>
<tr>
<td>nocache</td>
<td></td>
</tr>
<tr>
<td>noclean</td>
<td></td>
</tr>
<tr>
<td>precious</td>
<td></td>
</tr>
<tr>
<td>prerequisites</td>
<td></td>
</tr>
<tr>
<td>pseudo</td>
<td></td>
</tr>
<tr>
<td>ref_count</td>
<td></td>
</tr>
<tr>
<td>side_effect</td>
<td></td>
</tr>
<tr>
<td>side_effects</td>
<td></td>
</tr>
<tr>
<td>sources</td>
<td></td>
</tr>
<tr>
<td>sources_set</td>
<td></td>
</tr>
<tr>
<td>state</td>
<td></td>
</tr>
<tr>
<td>store_info</td>
<td></td>
</tr>
<tr>
<td>waiting_parents</td>
<td></td>
</tr>
<tr>
<td>waiting_s_e</td>
<td></td>
</tr>
<tr>
<td>wkids</td>
<td></td>
</tr>
</tbody>
</table>

*Inherited from object*
13.7 Class NodeList

object

  __abcoll.Sized

object

  __abcoll.Iterable

object

  __abcoll.Container

     __abcoll.Sequence

     __abcoll.MutableSequence


13.7.1 Methods

__str__(self)

str(x) Overrides: object.__str__ (inherited documentation)

Inherited from UserList.UserList

  __add__(), __cmp__(), __contains__(), __delitem__(), __delslice__(),
  __eq__(), __ge__(), __getitem__(), __getslice__(), __gt__(), __iadd__(),
  __imul__(), __init__(), __le__(), __len__(), __lt__(), __mul__(), __ne__(),
  __radd__(), __repr__(), __rmul__(), __setitem__(), __setslice__(), append(),
  count(), extend(), index(), insert(), pop(), remove(), reverse(), sort()

Inherited from __abcoll.Sequence

  __iter__(), __reversed__()

Inherited from __abcoll.Sized
__subclasshook__()  

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __setattr__(), __sizeof__()

13.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td>Inherited from object</td>
</tr>
</tbody>
</table>

13.7.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
<td>Inherited from UserList.UserList</td>
</tr>
</tbody>
</table>

13.8 Class Walker

object | SCons.Node.Walker

An iterator for walking a Node tree.

This is depth-first, children are visited before the parent. The Walker object can be initialized with any node, and returns the next node on the descent with each get_next() call. 'kids_func' is an optional function that will be called to get the children of a node instead of calling 'children'. 'cycle_func' is an optional function that will be called when a cycle is detected.

This class does not get caught in node cycles caused, for example, by C header file include loops.
13.8.1 Methods

```python
__init__(self, node, kids_func=<__builtin__.function object>,
cycle_func=<__builtin__.function object>,
eval_func=<__builtin__.function object>)
```

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)

```python
get_next(self)
```

Return the next node for this walk of the tree.

This function is intentionally iterative, not recursive, to sidestep any issues of stack size limitations.

```python
is_done(self)
```

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

13.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inherited from object</strong></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
14 Module SCons.Node.Alias

scons.Node.Alias

Alias nodes.

This creates a hash of global Aliases (dummy targets).

14.1 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Node/Alias.py rel_3.0.0:4395:8972f6a2f6...</td>
</tr>
<tr>
<td>default_ans</td>
<td>Value: {}</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Node'</td>
</tr>
</tbody>
</table>

14.2 Class AliasNameSpace

UserDict.UserDict

\[ \text{SCons.Node.Alias.AliasNameSpace} \]

14.2.1 Methods

- \texttt{Alias}(self, name, **kw)
- \texttt{lookup}(self, name, **kw)

\textit{Inherited from UserDict.UserDict}

\[ \text{\_cmp\_\_}(), \text{\_contains\_\_}(), \text{\_delitem\_\_}(), \text{\_getitem\_\_}(), \text{\_init\_\_}(), \text{\_len\_\_}(), \text{\_repr\_\_}(), \text{\_setitem\_\_}(), \text{clear}(), \text{copy}(), \text{fromkeys}(), \text{get}(), \text{has\_key}(), \text{items}(), \text{iteritems}(), \text{iterkeys}(), \text{itervalues}(), \text{keys}(), \text{pop}(), \text{popitem}(), \text{setdefault}(), \text{update}(), \text{values}() \]

14.2.2 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hash</strong></td>
<td>Inherited from UserDict.UserDict</td>
</tr>
</tbody>
</table>

148
14.3 Class AliasNodeInfo

object

SCons.Node.NodeInfoBase


The generic base class for signature information for a Node.

Node subclasses should subclass NodeInfoBase to provide their own logic for dealing with their own Node-specific signature information.

14.3.1 Methods

```python
str_to_node(self, s)
```

```python
getstate(self)
```

Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a ‘__dict__’ slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class. Overrides: SCons.Node.NodeInfoBase.__getstate__

```python
setstate(self, state)
```

Restore the attributes from a pickled state. Overrides: SCons.Node.NodeInfoBase.__setstate__

Inherited from SCons.Node.NodeInfoBase(Section 13.4)

convert(), format(), merge(), update()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(),
__new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(),
__sizeof__(), __str__(), __subclasshook__()

14.3.2 Properties
### 14.3.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
<tr>
<td>field_list</td>
<td>Value: ['csig']</td>
</tr>
</tbody>
</table>

### 14.4 Class AliasBuildInfo

The generic base class for build information for a Node.

This is what gets stored in a .sconsign file for each target file. It contains a NodeInfo instance for this node (signature information that’s specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.

#### 14.4.1 Methods

**Inherited from SCons.Node.BuildInfoBase (Section 13.5)**

- `__getstate__()`, `__init__()`, `__setstate__()`, `merge()`

**Inherited from object**

- `__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 14.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

*continued on next page*
### 14.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
</tbody>
</table>

### 14.5 Class Alias

```plaintext
object

SCons.Node.Node

```

#### 14.5.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__init__(self, name)</code></td>
<td>x.<strong>init</strong>(...) initializes x; see help(type(x)) for signature. Overrides: object.<strong>init</strong> (inherited documentation)</td>
</tr>
<tr>
<td><code>str_for_display(self)</code></td>
<td>str(x) Overrides: object.<strong>str</strong> (inherited documentation)</td>
</tr>
<tr>
<td><code>__str__(self)</code></td>
<td>str(x) Overrides: object.<strong>str</strong> (inherited documentation)</td>
</tr>
<tr>
<td><code>make_ready(self)</code></td>
<td>Get a Node ready for evaluation. This is called before the Taskmaster decides if the Node is up-to-date or not. Overriding this method allows for a Node subclass to be disambiguated if necessary, or for an implicit source builder to be attached. Overrides: SCons.Node.Node.make_ready (inherited documentation)</td>
</tr>
</tbody>
</table>
**really_build** *(self, **kw)*

Actually build the node.

This is called by the Taskmaster after it's decided that the Node is out-of-date and must be rebuilt, and after the prepare() method has gotten everything, uh, prepared.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in built().

**is_up_to_date** *(self)*

Alternate check for whether the Node is current: If all of our children were up-to-date, then this Node was up-to-date, too.

SCons.Node.Node.is_up_to_date

**is_under** *(self, dir)*

**get_contents** *(self)*

The contents of an alias is the concatenation of the content signatures of all its sources. Overrides: SCons.Node.Node.get_contents

**sconsign** *(self)*

An Alias is not recorded in .sconsign files

**build** *(self)*


**convert** *(self)*
get_csig(self)

Generate a node’s content signature, the digested signature of its content.
node - the node cache - alternate node to use for the signature cache returns - the content signature  Overrides: SCons.Node.Node.get_csig

Inherited from SCons.Node.Node (Section 13.6)

Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(),
add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_se(),
add_wkid(), all_children(), alter_targets(), builder_set(), built(), changed(), children(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(),
disambiguate(), env_set(), executor_cleanup(), exists(), explain(), for_signature(),
get_abspath(), get_binfo(), get_build_env(), get_build_scanner_path(), get_builder(),
get_cachedir_csig(), get_env(), get_env_scanner(), get_executor(), get_found_includes(),
get_implicit_deps(), get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(),
get_stored_info(), get_string(), get_subst_proxy(), get_suffix(), get_target_scanner(),
has_builder(), has_explicit_builder(), is_derived(), isLiteral(), missing(), multiple_side_effect_has_builder(), new_binfo(), new_ninfo(), postprocess(), prepare(),
push_to_cache(), release_target_info(), remove(), render_include_tree(),
reset_executor(), retrieve_from_cache(), reexists(), scan(), scanner_key(), select_scanner(),
set_always_build(), set_executor(), set_explicit(), set_nocache(), set_noclean(),
set_precious(), set_pseudo(), set_specific_source(), set_state(), visited()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()

14.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.Node (Section 13.6)</td>
<td>always_build, attributes, binfo, builder, cached, changed_since_last_build, depends, depends_set, env, executor, ignore, ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect, side_effects, sources, sources_set, state, store_info, waiting_parents, waiting_se, wkids</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong>_</td>
</tr>
</tbody>
</table>
Module SCons.Node.FS

scons.Node.FS

File system nodes.

These Nodes represent the canonical external objects that people think of when they think of building software: files and directories.

This holds a 'default_fs' variable that should be initialized with an FS that can be used by scripts or modules looking for the canonical default.

15.1 Functions

- **sconsign_none**(node)
- **sconsign_dir**(node)
  
  Return the .sconsign file info for this directory, creating it first if necessary.
- **save_strings**(val)
- **initialize_do_splitdrive()**
- **needs_normpath_match**(...

  match(string[, pos[, endpos]]) --> match object or None. Matches zero or more characters at the beginning of the string
- **set_duplicate**(duplicate)
- **LinkFunc**(target, source, env)
- **LocalString**(target, source, env)
- **UnlinkFunc**(target, source, env)
- **MkdirFunc**(target, source, env)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get_MkdirBuilder()</code></td>
<td></td>
</tr>
<tr>
<td><code>do_diskcheck_match(node, predicate, errorfmt)</code></td>
<td></td>
</tr>
<tr>
<td><code>ignore_diskcheck_match(node, predicate, errorfmt)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_diskcheck(list)</code></td>
<td></td>
</tr>
<tr>
<td><code>diskcheck_types()</code></td>
<td></td>
</tr>
<tr>
<td><code>has_glob_magic(s)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_default_fs()</code></td>
<td></td>
</tr>
<tr>
<td><code>find_file(filename, paths, verbose=None)</code></td>
<td>Find a node corresponding to either a derived file or a file that exists already. Only the first file found is returned, and none is returned if no file is found. filename: A filename to find paths: A list of directory path nodes to search in. Can be represented as a list, a tuple, or a callable that is called with no arguments and returns the list or tuple. returns The node created from the found file.</td>
</tr>
<tr>
<td><code>invalidate_node_memos(targets)</code></td>
<td>Invalidate the memoized values of all Nodes (files or directories) that are associated with the given entries. Has been added to clear the cache of nodes affected by a direct execution of an action (e.g. Delete/Copy/Chmod). Existing Node caches become inconsistent if the action is run through Execute(). The argument targets can be a single Node object or filename, or a sequence of Nodes/filenames.</td>
</tr>
</tbody>
</table>

15.2 Variables

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Node/FS.py rel_3.0.0:4395:8972f6a2f699 ...</td>
</tr>
<tr>
<td>print_duplicate</td>
<td>Value: 0</td>
</tr>
<tr>
<td>default_max_drift</td>
<td>Value: 172800</td>
</tr>
<tr>
<td>Save_Strings</td>
<td>Value: None</td>
</tr>
<tr>
<td>do_splitdrive</td>
<td>Value: False</td>
</tr>
<tr>
<td>needs_normpath_check</td>
<td>Value: re.compile(r'(?x).*//</td>
</tr>
<tr>
<td>Valid_Duplicates</td>
<td>Value: ['hard-soft-copy', 'soft-hard-copy', 'hard-copy', 'soft-c...</td>
</tr>
<tr>
<td>Link_Funcs</td>
<td>Value: []</td>
</tr>
<tr>
<td>Link</td>
<td>Value: &lt;SCons.Action.FunctionAction object&gt;</td>
</tr>
<tr>
<td>LocalCopy</td>
<td>Value: &lt;SCons.Action.FunctionAction object&gt;</td>
</tr>
<tr>
<td>Mkdir</td>
<td>Value: &lt;SCons.Action.FunctionAction object&gt;</td>
</tr>
<tr>
<td>MkdirBuilder</td>
<td>Value: None</td>
</tr>
<tr>
<td>diskcheck_match</td>
<td>Value: &lt;SCons.Node.FS.DiskChecker object&gt;</td>
</tr>
<tr>
<td>diskcheckers</td>
<td>Value: [&lt;SCons.Node.FS.DiskChecker object&gt;]</td>
</tr>
<tr>
<td>node_bwcomp</td>
<td>Value: {'abspath': &lt;unbound method Base.get_abspath&gt;, 'labspath'...</td>
</tr>
<tr>
<td>glob_magic_check</td>
<td>Value: re.compile(r'[^*?[]]')</td>
</tr>
<tr>
<td>default_fs</td>
<td>Value: None</td>
</tr>
<tr>
<td>OS_SEP</td>
<td>Value: '/'</td>
</tr>
<tr>
<td>UNC_PREFIX</td>
<td>Value: '//'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Node'</td>
</tr>
<tr>
<td>has_unc</td>
<td>Value: False</td>
</tr>
<tr>
<td>os_sep_is_slash</td>
<td>Value: True</td>
</tr>
</tbody>
</table>
15.3 Class EntryProxyAttributeError


An AttributeError subclass for recording and displaying the name of the underlying Entry involved in an AttributeError exception.

15.3.1 Methods

```python
__init__(self, entry_proxy, attribute)
```
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ (inherited documentation)

```python
__str__(self)
```
str(x) Overrides: object.__str__ (inherited documentation)

Inherited from exceptions.AttributeError

```python
__new__()
```

Inherited from exceptions.BaseException

```python
__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()
```

Inherited from object

```python
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

15.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td></td>
</tr>
</tbody>
</table>
15.4 Class DiskChecker

```
object —
SCons.Node.FS.DiskChecker
```

15.4.1 Methods

```
__init__(self, type, do, ignore)
```

\[x.__init__(...)\] initializes \(x\); see help(type(x)) for signature  Overrides:  \nobject.__init__ extit(inherited documentation)

```
__call__(self, *args, **kw)  
```

```
set(self, list)
```

Inherited from object

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

15.4.2 Properties

```
Name | Description
--- | ---

Inherited from object

___class___
15.5 Class EntryProxy

object

SCons.Util.Proxy

SCons.Node.FS.EntryProxy

15.5.1 Methods

__str__(...)

A Python Descriptor class that delegates attribute fetches to an underlying wrapped subject of a Proxy. Typical use:

```python
class Foo(Proxy): __str__ = Delegate('__str__')
```

Overrides: object.__str__

__hash__(...)

A Python Descriptor class that delegates attribute fetches to an underlying wrapped subject of a Proxy. Typical use:

```python
class Foo(Proxy): __str__ = Delegate('__str__')
```

Overrides: object.__hash__

getattr(self, name)

Retrieve an attribute from the wrapped object. If the named attribute doesn’t exist, AttributeError is raised. Overrides: SCons.Util.Proxy.__getattr__

Inherited from SCons.Util.Proxy(Section 36.5)

__eq__(), __init__() get()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()
15.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

15.5.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dictSpecialAttrs</td>
<td>Value: {'abspath': &lt;<strong>builtin</strong>.function object&gt;, 'base': &lt;__builtin...</td>
</tr>
</tbody>
</table>

15.6 Class Base

```
object                                   
\|-- SCons.Node.Node                      
   \|___ SCons.Node.FS.Base               

```

A generic class for file system entries. This class is for when we don’t know yet whether the entry being looked up is a file or a directory. Instances of this class can morph into either Dir or File objects by a later, more precise lookup.

Note: this class does not define __cmp__ and __hash__ for efficiency reasons. SCons does a lot of comparing of Node.FS.{Base,Entry,File,Dir} objects, so those operations must be as fast as possible, which means we want to use Python’s built-in object identity comparisons.

15.6.1 Methods

```
__init__(self, name, directory, fs)
```

Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.__init__
**str_for_display**(self)

**must_be_same**(self, klass)

This node, which already existed, is being looked up as the specified klass. Raise an exception if it isn’t.

**get_dir**(self)

**get_suffix**(self)


**rfile**(self)

**__getattr__**(self, attr)

Together with the node_bwcomp dict defined below, this method provides a simple backward compatibility layer for the Node attributes ‘abspath’, ‘labspath’, ‘path’, ‘tpath’, ‘suffix’ and ‘path_elements’. These Node attributes used to be directly available in v2.3 and earlier, but have been replaced by getter methods that initialize the single variables lazily when required, in order to save memory. The redirection to the getters lets older Tools and SConstruct continue to work without any additional changes, fully transparent to the user. Note, that **__getattr__** is only called as fallback when the requested attribute can’t be found, so there should be no speed performance penalty involved for standard builds.

**__str__**(self)

A Node.FS.Base object’s string representation is its path name. Overrides: object.__str__

**__lt__**(self, other)

less than operator used by sorting on py3
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rstr(self)</td>
<td>A Node.FS.Base object’s string representation is its path name.</td>
</tr>
<tr>
<td>stat(self)</td>
<td></td>
</tr>
<tr>
<td>rexists(self)</td>
<td>Does this node exist locally or in a repository? Overrides: SCons.Node.Node.rexists extit(inherited documentation)</td>
</tr>
<tr>
<td>getmtime(self)</td>
<td></td>
</tr>
<tr>
<td>getsize(self)</td>
<td></td>
</tr>
<tr>
<td>isdir(self)</td>
<td></td>
</tr>
<tr>
<td>isfile(self)</td>
<td></td>
</tr>
<tr>
<td>islink(self)</td>
<td></td>
</tr>
<tr>
<td>is__under(self, dir)</td>
<td></td>
</tr>
<tr>
<td>set__local(self)</td>
<td></td>
</tr>
<tr>
<td>srcnode(self)</td>
<td>If this node is in a build path, return the node corresponding to its source file. Otherwise, return ourself.</td>
</tr>
</tbody>
</table>
get_path(self, dir=None)

Return path relative to the current working directory of the Node.FS.Base object that owns us.

set_src_builder(self, builder)

Set the source code builder for this node.

dc_src_builder(self)

Fetch the source code builder for this node.
If there isn’t one, we cache the source code builder specified for the directory (which in turn will cache the value from its parent directory, and so on up to the file system root).

get_abspath(self)


get_labspath(self)

Get the absolute path of the file.

get_internal_path(self)

get_tpath(self)

get_path_elements(self)
**for_signature(self)**

Return a string representation of the Node that will always be the same for this particular Node, no matter what. This is by contrast to the `__str__()` method, which might, for instance, return a relative path for a file Node. The purpose of this method is to generate a value to be used in signature calculation for the command line used to build a target, and we use this method instead of `str()` to avoid unnecessary rebuilds. This method does not need to return something that would actually work in a command line; it can return any kind of nonsense, so long as it does not change. Overrides: `SCons.Node.Node.for_signature` (inherited documentation)

**get_subst_proxy(self)**

This method is expected to return an object that will function exactly like this Node, except that it implements any additional special features that we would like to be in effect for Environment variable substitution. The principle use is that some Nodes would like to implement a `__getattr__()` method, but putting that in the Node type itself has a tendency to kill performance. We instead put it in a proxy and return it from this method. It is legal for this method to return `self` if no new functionality is needed for Environment substitution. Overrides: `SCons.Node.Node.get_subst_proxy` (inherited documentation)

**target_from_source(self, prefix, suffix, splitext=<_builtin__.function object>)**

Generates a target entry that corresponds to this entry (usually a source file) with the specified prefix and suffix.

Note that this method can be overridden dynamically for generated files that need different behavior. See Tool/swig.py for an example.

**findalldirs(self, pathlist)**

Return all of the directories for a given path list, including corresponding 'backing' directories in any repositories.

The Node lookups are relative to this Node (typically a directory), so memoizing result saves cycles from looking up the same path for each target in a given directory.
**RDirs**

```python
self, pathlist)
```

Search for a list of directories in the Repository list.

**rentry**

```python
self)
```

### Inherited from SCons.Node.Node (Section 13.6)

Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(), add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_s_e(), add_wkid(), all_children(), alter_targets(), build(), builder_set(), built(), changed(), children(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(), disambiguate(), env_set(), executor_cleanup(), explain(), get_binfo(), get_build_env(), get_build_scanner_path(), get_builder(), get_cachefile(), get_contents(), get_csig(), get_env(), get_env_scanner(), get_executor(), get_found_includes(), get Implicit_deps(), get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(), get_stored_info(), get_string(), get_target_scanner(), has_builder(), has_explicit_builder(), is_derived(), is literal(), is_up_to_date(), make_ready(), missing(), multiple_side_effect_has Builder(), new_binfo(), new_ninfo(), postprocess(), prepare(), push_to_cache(), release_target_info(), remove(), render_include_tree(), reset_executor(), retrieve_from_cache(), scan(), scanner_key(), select_scanner(), set_always_build(), set_implicit(), set_nocache(), set_noclean(), set_precious(), set_pseudo(), set_specific_source(), set_state(), visited()

### Inherited from object

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()
```

### 15.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwd</td>
<td></td>
</tr>
<tr>
<td>dir</td>
<td></td>
</tr>
<tr>
<td>duplicate</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>sbuilder</td>
<td></td>
</tr>
</tbody>
</table>

*Inherited from SCons.Node.Node (Section 13.6)*

*continued on next page*
### Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs</td>
<td></td>
</tr>
</tbody>
</table>

### Class Entry

**object** 🔄

**SCons.Node.Node** 🔄

**SCons.Node.FS.Base** 🔄 **SCons.Node.FS.Entry**

This is the class for generic Node.FS entries—that is, things that could be a File or a Dir, but we're just not sure yet. Consequently, the methods in this class really exist just to transform their associated object into the right class when the time comes, and then call the same-named method in the transformed class.

#### Methods

```python
__init__(self, name, directory, fs)
```

Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.__init__ (inherited documentation)
**diskcheck_match**(<em>self</em>)

**disambiguate**(<em>self</em>, <em>must_exist=None</em>)

**rfile**(<em>self</em>)
We’re a generic Entry, but the caller is actually looking for a File at this point, so morph into one. Override: SCons.Node.FS.Base.rfile

**scanner_key**(<em>self</em>)

**get_contents**(<em>self</em>)

**get_text_contents**(<em>self</em>)
Fetch the decoded text contents of a Unicode encoded Entry. Since this should return the text contents from the file system, we check to see into what sort of subclass we should morph this Entry.

**must_be_same**(<em>self</em>, <em>klass</em>)
Called to make sure a Node is a Dir. Since we’re an Entry, we can morph into one. Override: SCons.Node.FS.Base.must_be_same

**exists**(<em>self</em>)
Does this node exist? Override: SCons.Node.Node.exists (inherited documentation)

**rel_path**(<em>self</em>, <em>other</em>)
**new_ninfo** *(self)*


**get_subst_proxy** *(self)*

This method is expected to return an object that will function exactly like this Node, except that it implements any additional special features that we would like to be in effect for Environment variable substitution. The principle use is that some Nodes would like to implement a **getattr** () method, but putting that in the Node type itself has a tendency to kill performance. We instead put it in a proxy and return it from this method. It is legal for this method to return self if no new functionality is needed for Environment substitution. Overrides: SCons.Node.Node.get_subst_proxy

---

**Inherited from SCons.Node.FS.Base**(Section 15.6)

- RDirs(), Rfindalldirs(), **__getattr__()**, **__lt__()**, **__str__()**, for_signature(), get_abspath(), get_dir(), get_internal_path(), get_labspath(), get_path(), get_path_elements(), get_suffix(), get_tpath(), getmtime(), getsize(), is_under(), isdir(), isfile(), islink(), entry(), exists(), rstr(), set_local(), set_src_builder(), src_builder(), srcnode(), stat(), str_for_display(), target_from_source()

**Inherited from SCons.Node.Node**(Section 13.6)

- Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(), add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_se(), add_wkid(), all_children(), alter_targets(), build(), builder_set(), built(), changed(), children(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(), env_set(), executor_cleanup(), explain(), get_binfo(), get_build_env(), get_build_scanner_path(), get_builder(), get_cachedir_csig(), get_csig(), get_env(), get_env_scanner(), get_executor(), get_found_includes(), get_implicit_deps(), get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(), get_stored_info(), get_string(), get_target_scanner(), has_builder(), has_explicit_builder(), isDerived(), is_literal(), is_up_to_date(), make_ready(), missing(), multiple_side_effect_has_builder(), new_binfo(), postprocess(), prepare(), push_to_cache(), release_target_info(), remove(), render_include_tree(), reset_executor(), retrieve_from_cache(), scan(), select_scanner(), set_always_build(), set_executor(), set_explicit(), set_nocache(), set_nocheck(), set_pseudo(), set_specific_source(), set_state(), visited()

**Inherited from object**

- **__delattr__()**, **__format__()**, **__getattribute__()**, **__hash__()**, **__new__()**, **__reduce__()**, **__reduce_ex__()**, **__repr__()**, **__setattr__()**, **__sizeof__()**, **__subclasshook__()**
### 15.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachedir_csig</td>
<td></td>
</tr>
<tr>
<td>cachesig</td>
<td></td>
</tr>
<tr>
<td>contentsig</td>
<td></td>
</tr>
<tr>
<td>dirname</td>
<td></td>
</tr>
<tr>
<td>entries</td>
<td></td>
</tr>
<tr>
<td>on_disk_entries</td>
<td></td>
</tr>
<tr>
<td>released_target_info</td>
<td></td>
</tr>
<tr>
<td>repositories</td>
<td></td>
</tr>
<tr>
<td>root</td>
<td></td>
</tr>
<tr>
<td>scanner_paths</td>
<td></td>
</tr>
<tr>
<td>searched</td>
<td></td>
</tr>
<tr>
<td>srccdir</td>
<td></td>
</tr>
<tr>
<td>variant_dirs</td>
<td></td>
</tr>
</tbody>
</table>

*Inherited from SCons.Node.FS.Base (Section 15.6)*

cwd, dir, duplicate, name, sbuilder

*Inherited from SCons.Node.Node (Section 13.6)*

always_build, attributes, binfo, builder, cached,
changed_since_last_build, depends, depends_set, env, executor, ignore,
ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo,
ocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect,
side_effects, sources, sources_set, state, store_info, waiting_parents,
waiting_s_e, wkids

*Inherited from object*

__class__

### 15.7.3 Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Inherited from SCons.Node.FS.Base (Section 15.6)*

fs

### 15.8 Class LocalFS

```
object

SCons.Node.FS.LocalFS
```

**Known Subclasses:** SCons.Node.FS.FS
15.8.1 Methods

```python
chmod(self, path, mode)

 copy(self, src, dst)

 copy2(self, src, dst)

 exists(self, path)

 getmtime(self, path)

 getsize(self, path)

 isdir(self, path)

 isfile(self, path)

 link(self, src, dst)

 lstat(self, path)

 listdir(self, path)

 makedirs(self, path)

 mkdir(self, path)

 rename(self, old, new)

 stat(self, path)

 symlink(self, src, dst)

 open(self, path)

 unlink(self, path)

 islink(self, path)
```
Class FS

Module SCons.Node.FS

```python
readlink(self, file)
```

**Inherited from object**

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(),
__new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(),
sizeof(), __str__(), __subclasshook__()
```

15.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

15.9 Class FS

```python
object
```

```python
SCons.Node.FS.LocalFS
```

```python
SCons.Node.FS.FS
```

15.9.1 Methods

```python
__init__(self, path=None)
```

Initialize the Node.FS subsystem.

The supplied path is the top of the source tree, where we expect to find the top-level build file. If no path is supplied, the current directory is the default.

The path argument must be a valid absolute path. Overrides:
```python
object.__init__
```

```python
set_SConstruct_dir(self, dir)
```

```python
get_max_drift(self)
```

```python
set_max_drift(self, max_drift)
```
getcwd(self)

chdir(self, dir, change_os_dir=0)

Change the current working directory for lookups. If change_os_dir is true, we will also change the *real* cwd to match.

get_root(self, drive)

Returns the root directory for the specified drive, creating it if necessary.

Entry(self, name, directory=None, create=1)

Look up or create a generic Entry node with the specified name. If the name is a relative path (begins with ./, ../, or a file name), then it is looked up relative to the supplied directory node, or to the top level directory of the FS (supplied at construction time) if no directory is supplied.

File(self, name, directory=None, create=1)

Look up or create a File node with the specified name. If the name is a relative path (begins with ./, ../, or a file name), then it is looked up relative to the supplied directory node, or to the top level directory of the FS (supplied at construction time) if no directory is supplied.

This method will raise TypeError if a directory is found at the specified path.

Dir(self, name, directory=None, create=True)

Look up or create a Dir node with the specified name. If the name is a relative path (begins with ./, ../, or a file name), then it is looked up relative to the supplied directory node, or to the top level directory of the FS (supplied at construction time) if no directory is supplied.

This method will raise TypeError if a normal file is found at the specified path.
**Class FS**  
*Module SCons.Node.FS*

### VariantDir

```python
VariantDir(self, variant_dir, src_dir, duplicate=1)
```

Link the supplied variant directory to the source directory for purposes of building files.

### Repository

```python
Repository(self, *dirs)
```

Specify Repository directories to search.

### PyPackageDir

```python
PyPackageDir(self, modulename)
```

Locate the directory of a given python module name

For example scons might resolve to Windows:  
C:Python27Libsite-package-scons-2.5.1 Linux: /usr/lib/scons

This can be useful when we want to determine a toolpath based on a python module name

### variant_dir_target_climb

```python
variant_dir_target_climb(self, orig, dir, tail)
```

Create targets in corresponding variant directories

Climb the directory tree, and look up path names relative to any linked variant directories we find.

Even though this loops and walks up the tree, we don’t memoize the return value because this is really only used to process the command-line targets.

### Glob

```python
Glob(self, pathname, ondisk=True, source=True, strings=False, exclude=None, cwd=None)
```

Globs

This is mainly a shim layer

**Inherited from SCons.Node.FS.LocalFS(Section 15.8)**

- `chmod()`, `copy()`, `copy2()`, `exists()`, `getmtime()`, `getsize()`, `isdir()`, `isfile()`, `islink()`,
link(), listdir(), lstat(), mkdir(), makedirs(), open(), readlink(), rename(), stat(), symlink(), unlink()

**Inherited from object**

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

### 15.9.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 15.10 Class DirNodeInfo

```
object

SCons.Node.NodeInfoBase

SCons.Node.FS.DirNodeInfo
```

The generic base class for signature information for a Node.

Node subclasses should subclass NodeInfoBase to provide their own logic for dealing with their own Node-specific signature information.

#### 15.10.1 Methods

```python
str_to_node(self, s)
```

**Inherited from SCons.Node.NodeInfoBase** (*Section 13.4*)

```python
__getstate__(), __setstate__(), convert(), format(), merge(), update()
```

**Inherited from object**

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(),
__new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(),
__sizeof__(), __str__(), __subclasshook__()
```

### 15.10.2 Properties
15.10.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
<tr>
<td>fs</td>
<td>Value: None</td>
</tr>
</tbody>
</table>

15.11 Class DirBuildInfo

```
object ▼
SCons.Node.BuildInfoBase ▼
SCons.Node.FS.DirBuildInfo
```

The generic base class for build information for a Node.

This is what gets stored in a .sconsign file for each target file. It contains a NodeInfo instance for this node (signature information that’s specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.

15.11.1 Methods

*Inherited from SCons.Node.BuildInfoBase (Section 13.5)*

```python
__getstate__(), __init__(), __setstate__(), merge()
```

*Inherited from object*

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

15.11.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>bact, bactsig, bdepends, bdependsigs, bimplicit, bimplicitssigs, bsources, bsourcesigs</td>
<td>Inherited from object</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

15.11.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
</tbody>
</table>

15.12 Class Dir

```
object
   SCons.Node.Node
     SCons.Node.FS.Base
       SCons.Node.FS.Dir
```

**Known Subclasses:** SCons.Node.FS.RootDir

A class for directories in a file system.

15.12.1 Methods

```python
___init__(self, name, directory, fs)
```

Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.__init__(inherited documentation)

```python
diskcheck__match(self)
```
**Entry**(self, name)

Looks up or creates an entry node named ‘name’ relative to this directory.

**Dir**(self, name, create=True)

Looks up or creates a directory node named ‘name’ relative to this directory.

**File**(self, name)

Looks up or creates a file node named ‘name’ relative to this directory.

**link**(self, srcdir, duplicate)

Set this directory as the variant directory for the supplied source directory.

**getRepositories**(self)

Returns a list of repositories for this directory.

**get_all_rdirs**(self)

**addRepository**(self, dir)

**up**(self)

**rel_path**(self, other)

Return a path to 'other' relative to this directory.

**get_env_scanner**(self, env, kw={})

**get**\_**target**\_**scanner**(*self*)


**get**\_**found**\_**includes**(*self, env, scanner, path*)

Return this directory’s implicit dependencies.

We don’t bother caching the results because the scan typically shouldn’t be requested more than once (as opposed to scanning .h file contents, which can be requested as many times as the files is #included by other files).

Overrides: SCons.Node.Node.get\_found\_includes

**prepare**(*self*)

Prepare for this Node to be built.

This is called after the Taskmaster has decided that the Node is out-of-date and must be rebuilt, but before actually calling the method to build the Node.

This default implementation checks that explicit or implicit dependencies either exist or are derived, and initializes the BuildInfo structure that will hold the information about how this node is, uh, built.

(The existence of source files is checked separately by the Executor, which aggregates checks for all of the targets built by a specific action.)

Overriding this method allows for a Node subclass to remove the underlying file from the file system. Note that subclass methods should call this base class method to get the child check and the BuildInfo structure.

Overrides: SCons.Node.Node.prepare extit(inherited documentation)

**build**(*self, **kw*)

**multiple_side_effect_has_builder(self)**

Return whether this Node has a builder or not.

In Boolean tests, this turns out to be a *lot* more efficient than simply examining the builder attribute directly ('if node.builder: ...'). When the builder attribute is examined directly, it ends up calling `__getattr__` for both the `__len__` and `__nonzero__` attributes on instances of our Builder Proxy class(es), generating a bazillion extra calls and slowing things down immensely. Overrides: `SCons.Node.Node.multiple_side_effect_has_builder` extit(inherited documentation)

**alter_targets(self)**


**scanner_key(self)**


**get_text_contents(self)**

We already emit things in text, so just return the binary version.

**get_contents(self)**

Return content signatures and names of all our children separated by new-lines. Ensure that the nodes are sorted. Overrides: `SCons.Node.Node.get_contents`
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Overrides</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>get_csig(self)</strong></td>
<td>Compute the content signature for Directory nodes. In general, this is not needed and the content signature is not stored in the DirNodeInfo. However, if get_contents on a Dir node is called which has a child directory, the child directory should return the hash of its contents. Overrides: SCons.Node.Node.get_csig</td>
<td></td>
</tr>
<tr>
<td><strong>do_duplicate(self, src)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>is_up_to_date(self)</strong></td>
<td>If any child is not up-to-date, then this directory isn’t, either. Overrides: SCons.Node.Node.is_up_to_date</td>
<td></td>
</tr>
<tr>
<td><strong>rdir(self)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>sconsign(self)</strong></td>
<td>Return the .sconsign file info for this directory.</td>
<td></td>
</tr>
<tr>
<td><strong>srcnode(self)</strong></td>
<td>Dir has a special need for srcnode()...if we have a srcdir attribute set, then that is our srcnode. Overrides: SCons.Node.FS.Base.srcnode</td>
<td></td>
</tr>
<tr>
<td><strong>get_timestamp(self)</strong></td>
<td>Return the latest timestamp from among our children</td>
<td></td>
</tr>
</tbody>
</table>
### get\_labspath(self)

Get the absolute path of the file. Overrides: SCons.Node.FS.Base.get\_labspath

### get\_internal\_path(self)

Overrides: SCons.Node.FS.Base.get\_internal\_path

### get\_tpath(self)

Overrides: SCons.Node.FS.Base.get\_tpath

### get\_path\_elements(self)

Overrides: SCons.Node.FS.Base.get\_path\_elements

### entry\_abspath(self, name)

### entry\_labspath(self, name)

### entry\_path(self, name)

### entry\_tpath(self, name)

### entry\_exists\_on\_disk(self, name)

Searches through the file/dir entries of the current directory, and returns True if a physical entry with the given name could be found.

@see entry\_exists\_on\_disk

### rentry\_exists\_on\_disk(self, name)

Searches through the file/dir entries of the current and all its remote directories (repos), and returns True if a physical entry with the given name could be found. The local directory (self) gets searched first, so repositories take a lower precedence regarding the searching order.

@see entry\_exists\_on\_disk
**srcdir_list**(self)

**srcdir_duplicate**(self, name)

**srcdir_find_file**(self, filename)

**dir_on_disk**(self, name)

**file_on_disk**(self, name)

**walk**(self, func, arg)

Walk this directory tree by calling the specified function for each directory in the tree.

This behaves like the os.path.walk() function, but for in-memory Node.FS.Dir objects. The function takes the same arguments as the functions passed to os.path.walk():

    func(arg, dirname, fnames)

Except that "dirname" will actually be the directory Node, not the string. The "." and "." entries are excluded from fnames. The fnames list may be modified in-place to filter the subdirectories visited or otherwise impose a specific order. The "arg" argument is always passed to func() and may be used in any way (or ignored, passing None is common).
The \texttt{glob()} function returns a list of Nodes (or strings) matching a specified pathname pattern.

Pathname patterns follow UNIX shell semantics: * matches any-length strings of any characters, ? matches any character, and [] can enclose lists or ranges of characters. Matches do not span directory separators.

The matches take into account Repositories, returning local Nodes if a corresponding entry exists in a Repository (either an in-memory Node or something on disk).

By default, the \texttt{glob()} function matches entries that exist on-disk, in addition to in-memory Nodes. Setting the 'ondisk' argument to False (or some other non-true value) causes the \texttt{glob()} function to only match in-memory Nodes. The default behavior is to return both the on-disk and in-memory Nodes.

The 'source' argument, when true, specifies that corresponding source Nodes must be returned if you're globbing in a build directory (initialized with \texttt{VariantDir()}). The default behavior is to return Nodes local to the \texttt{VariantDir()}.

The 'strings' argument, when true, returns the matches as strings, not Nodes. The strings are path names relative to this directory.

The 'exclude' argument, if not None, must be a pattern or a list of patterns following the same UNIX shell semantics. Elements matching at least one pattern of this list will be excluded from the result.

The underlying algorithm is adapted from the \texttt{glob.glob()} function in the Python library (but heavily modified), and uses \texttt{fnmatch()} under the covers.

\begin{verbatim}
glob(self, pathname, ondisk=True, source=False, strings=False, exclude=None)
\end{verbatim}

\textit{Inherited from SCons.Node.FS.Base(Section 15.6)}

\texttt{RDirs()}, \texttt{Rfindalldirs()}, \_\_getattr\_\_(), \_\_lt\_\_(), \_\_str\_\_(), \texttt{exists()}, \texttt{for_signature()}, \texttt{get_dir()}, \texttt{get_path()}, \texttt{get_subst_proxy()}, \texttt{get_suffix()}, \texttt{getmtime()}, \texttt{getsize()}, \texttt{is_under()}, \texttt{isdir()}, \texttt{isfile()}, \texttt{islink()}, \texttt{must_be_same()}, \texttt{entry()}, \texttt{reexists()}, \texttt{rfile()}, \texttt{rstr()}, \texttt{set_local()}, \texttt{set_src_builder()}, \texttt{src_builder()}, \texttt{stat()}, \texttt{str_for_display()}, \texttt{target_from_source()}

\textit{Inherited from SCons.Node.Node(Section 13.6)}

\texttt{Decider()}, \texttt{GetTag()}, \texttt{Tag()}, \texttt{add_dependency()}, \texttt{add_ignore()}, \texttt{add_prerequisite()}, \texttt{add_source()}, \texttt{add_to_implicit()}, \texttt{add_to_waiting_parents()}, \texttt{add_to_waiting_s_e()}, \texttt{add_wkid()}, \texttt{all_children()}, \texttt{builder_set()}, \texttt{built()}, \texttt{changed()}, \texttt{children()}, \texttt{children_are_up_to_date()}, \texttt{clear()}, \texttt{clear.memoized_values()}, \texttt{del_binfo()}, \texttt{disambiguate()}, \texttt{env_set()}, \texttt{execu-}

183
tor_cleanup(), explain(), get_binfo(), get_build_env(), get_build_scanner_path(),
get_builder(), get_cachedir_csig(), get_env(), get_executor(), get_implicit_deps(),
get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(), get_stored_info(),
get_string(), has_builder(), has_explicit_builder(), is_derived(), is_literal(), make_ready(),
missing(), new_binfo(), new_ninfo(), postprocess(), push_to_cache(), release_target_info(),
remove(), render_include_tree(), reset_executor(), retrieve_from_cache(), scan(),
select_scanner(), set_always_build(), set_executor(), set_explicit(), set_nocache(),
set_noclean(), set_precious(), set_pseudo(), set_specific_source(), set_state(),
visited()
15.12.3 Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs</td>
<td>Inherited from SCons.Node.FS.Base (Section 15.6)</td>
</tr>
</tbody>
</table>

15.13 Class RootDir

A class for the root directory of a file system.

This is the same as a Dir class, except that the path separator ('/' or '”) is actually part of the name, so we don’t need to add a separator when creating the path names of entries within this directory.

15.13.1 Methods

```python
__init__(self, drive, fs)
```

Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.__init__ (inherited documentation)

```python
must_be_same(self, klass)
```

This node, which already existed, is being looked up as the specified klass. Raise an exception if it isn’t. Overrides: SCons.Node.FS.Base.must_be_same (inherited documentation)
Class RootDir

Module SCons.Node.FS

```python
__str__(self)
A Node.FS.Base object’s string representation is its path name. Overrides: object.__str__ (inherited documentation)

entry__abspath(self, name)
Overrides: SCons.Node.FS.Dir.entry__abspath

dir(entry__abs
path(self, name)
Overrides: SCons.Node.FS.Dir.entry__abspath

entry__l pathname(self, name)
Overrides: SCons.Node.FS.Dir.entry__pathname

entry__tpath(self, name)
Overrides: SCons.Node.FS.Dir.entry__tpath

is__under(self, dir)
Overrides: SCons.Node.FS.Base.is__under

up(self)
Overrides: SCons.Node.FS.Dir.up

get__dir(self)
Overrides: SCons.Node.FS.Base.get__dir

src_builder(self)
Fetch the source code builder for this node.
If there isn’t one, we cache the source code builder specified for the directory
(which in turn will cache the value from its parent directory, and so on up to
the file system root). Overrides: SCons.Node.FS.Base.src_builder

Inherited from SCons.Node.FS.Dir(Section 15.12)

Dir(), Entry(), File(), addRepository(), alter_targets(), build(), dir_on_disk(),
diskcheck_match(), do_duplicate(), entry_exists_on_disk(), file_on_disk(), getRepositories(), get__abs path(), get_all_rdirs(), get_contents(), get__csig(), get_env__scanner(),
get_found_includes(), get__internal_path(), get__labspath(), get__path_elements(),
```
get_target_scanner(), get_text_contents(), get_timestamp(), get_tpath(), glob(),
is_up_to_date(), link(), multiple_side_effect_has_builder(), prepare(), rdir(),
rel_path(), rentry_exists_on_disk(), scanner_key(), sconsign(), srcdir_duplicate(),
srcdir_find_file(), srcdir_list(), srcnode(), walk()

**Inherited from SCons.Node.FS.Base (Section 15.6)**

RDirs(), Rfindalldirs(), __getattr__(), __lt__(), exists(), for_signature(), get_path(),
get_subst_proxy(), get_suffix(), getmtime(), getsize(), isdir(), isfile(), islink(),
rentry(), rexists(), rfile(), rstr(), set_local(), set_src_builder(), stat(), str_for_display(),
target_from_source()

**Inherited from SCons.Node.Node (Section 13.6)**

Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(),
add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_s_e(),
add_wkid(), all_children(), builder_set(), built(), changed(), children(), children_are_up_to_date(),
clear(), clear_memoized_values(), del_binfo(), disambiguate(), env_set(), executor_cleanup(), explain(),
get_binfo(), get_build_env(), get_build_scanner_path(),
get_builder(), get_cachefile(), get_env(), get_executor(), get_implicit_deps(),
get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(), get_stored_info(),
get_string(), has_builder(), has_explicit_builder(), is_derived(), is_literal(), make_ready(),
missing(), new_binfo(), new_ninfo(), postprocess(), push_to_cache(), release_target_info(),
remove(), renderIncludeTree(), reset_executor(), retrieve_from_cache(), scan(),
select_scanner(), set_always_build(), set_executor(), set_explicit(), set_nocache(),
set_noclean(), set_noclean(), set_pseudo(), set_specific_source(), set_state(),
visited()

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()

### 15.13.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.FS.Dir (Section 15.12)</td>
<td></td>
</tr>
<tr>
<td>cachedir_csig, cachesig, contentsig, dirname, entries, on_disk_entries, released_target_info, repositories, root, scanner_paths, searched, srcdir, variant_dirs</td>
<td></td>
</tr>
<tr>
<td>Inherited from SCons.Node.FS.Base (Section 15.6)</td>
<td></td>
</tr>
<tr>
<td>cwd, dir, duplicate, name, sbuilder</td>
<td></td>
</tr>
<tr>
<td>Inherited from SCons.Node.Node (Section 13.6)</td>
<td></td>
</tr>
</tbody>
</table>

*continued on next page*
15.13.3 Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs</td>
<td>Reference to parent Node.FS object</td>
</tr>
</tbody>
</table>

15.14 Class FileNodeInfo

The generic base class for signature information for a Node.

Node subclasses should subclass NodeInfoBase to provide their own logic for dealing with their own Node-specific signature information.

15.14.1 Methods

```python
str_to_node(self, s)
```

```python
__getstate__(self)
```

Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a '__dict__' slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class. Overrides: SCons.Node.NodeInfoBase.__getstate__
```
__setstate__(self, state)
```

Restore the attributes from a pickled state. Overrides: SCons.Node.NodeInfoBase.__setstate__

Inherited from SCons.Node.NodeInfoBase (Section 13.4)

convert(), format(), merge(), update()

Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

15.14.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csig</td>
<td></td>
</tr>
<tr>
<td>size</td>
<td></td>
</tr>
<tr>
<td>timestamp</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object

__class__

15.14.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
<tr>
<td>field_list</td>
<td>Value: ['csig', 'timestamp', 'size']</td>
</tr>
<tr>
<td>fs</td>
<td>Value: None</td>
</tr>
</tbody>
</table>

15.15 Class FileBuildInfo

object

SCons.Node.BuildInfoBase

SCons.Node.FS.FileBuildInfo

Known Subclasses: SCons.SConf.SConfBuildInfo

The generic base class for build information for a Node.
This is what gets stored in a .sconsign file for each target file. It contains a NodeInfo instance for this node (signature information that’s specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.

## 15.15.1 Methods

### `convert_from_sconsign(self, dir, name)`

Converts a newly-read FileBuildInfo object for in-SCons use

For normal up-to-date checking, we don’t have any conversion to perform--but we’re leaving this method here to make that clear.

### `convert_to_sconsign(self)`

Converts this FileBuildInfo object for writing to a .sconsign file

This replaces each Node in our various dependency lists with its usual string representation: relative to the top-level SConstruct directory, or an absolute path if it’s outside.

### `format(self, names=0)`

### `prepare_dependencies(self)`

Prepares a FileBuildInfo object for explaining what changed

The bsources, bdepends and bimplicit lists have all been stored on disk as paths relative to the top-level SConstruct directory. Convert the strings to actual Nodes (for use by the --debug=explain code and --implicit-cache).

*Inherited from SCons.Node.BuildInfoBase (Section 13.5)*

    __getstate__(), __init__(), __setstate__(), merge()

*Inherited from object*

    __delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
    __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
    __str__(), __subclasshook__()
## 15.15.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</td>
<td>bact, bactsig, bdepends, bdependsigs, bimplicit, bimplicitistsigs, bsources, bsourcesigs</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td>Inherited from object</td>
</tr>
</tbody>
</table>

## 15.15.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
</tbody>
</table>

## 15.16 Class File

object → 

SCons.Node.Node → 


A class for files in a file system.

### 15.16.1 Methods

**diskcheck_match(self)**

**__init__(self, name, directory, fs)**

Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.__init__ (inherited documentation)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entry</strong> <em>(self, name)</em></td>
<td>Create an entry node named 'name' relative to the directory of this file.</td>
</tr>
<tr>
<td><strong>Dir</strong> <em>(self, name, create=True)</em></td>
<td>Create a directory node named 'name' relative to the directory of this file.</td>
</tr>
<tr>
<td><strong>Dirs</strong> <em>(self, pathlist)</em></td>
<td>Create a list of directories relative to the SConscript directory of this file.</td>
</tr>
<tr>
<td><strong>File</strong> <em>(self, name)</em></td>
<td>Create a file node named 'name' relative to the directory of this file.</td>
</tr>
<tr>
<td><strong>scanner_key</strong> <em>(self)</em></td>
<td>Overrides: SCons.Node.Node.scanner_key</td>
</tr>
<tr>
<td><strong>get_contents</strong> <em>(self)</em></td>
<td>Fetch the contents of the entry. Overrides: SCons.Node.Node.get_contents</td>
</tr>
<tr>
<td><strong>get_text_contents</strong> <em>(self)</em></td>
<td>This attempts to figure out what the encoding of the text is based upon the BOM bytes, and then decodes the contents so that it’s a valid Python string.</td>
</tr>
<tr>
<td><strong>get_content_hash</strong> <em>(self)</em></td>
<td>Compute and return the MD5 hash for this file.</td>
</tr>
<tr>
<td><strong>get_size</strong> <em>(self)</em></td>
<td>192</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>get_timestamp(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>convert_old_entry(self, old_entry)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_stored_implicit(self)</code></td>
<td>Fetch the stored implicit dependencies  Overrides: SCons.Node.Node.get_stored_implicit (inherited documentation)</td>
</tr>
<tr>
<td><code>rel_path(self, other)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_found_includes(self, env, scanner, path)</code></td>
<td>Return the included implicit dependencies in this file. Cache results so we only scan the file once per path regardless of how many times this information is requested.  Overrides: SCons.Node.Node.get_found_includes</td>
</tr>
<tr>
<td><code>push_to_cache(self)</code></td>
<td>Try to push the node into a cache  Overrides: SCons.Node.Node.push_to_cache</td>
</tr>
<tr>
<td><code>retrieve_from_cache(self)</code></td>
<td>Try to retrieve the node’s content from a cache</td>
</tr>
<tr>
<td></td>
<td>This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in built().</td>
</tr>
<tr>
<td></td>
<td>Returns true if the node was successfully retrieved.  Overrides: SCons.Node.Node.retrieve_from_cache</td>
</tr>
<tr>
<td><code>visited(self)</code></td>
<td>Called just after this node has been visited (with or without a build).</td>
</tr>
</tbody>
</table>
release_target_info(self)

Called just after this node has been marked up-to-date or was built completely.

This is where we try to release as many target node infos as possible for clean builds and update runs, in order to minimize the overall memory consumption.

We’d like to remove a lot more attributes like self.sources and self.sources_set, but they might get used in a next build step. For example, during configuration the source files for a built E{*.}o file are used to figure out which linker to use for the resulting Program (gcc vs. g++)! That’s why we check for the ’keep_targetinfo’ attribute, config Nodes and the Interactive mode just don’t allow an early release of most variables.

In the same manner, we can’t simply remove the self.attributes here. The smart linking relies on the shared flag, and some parts of the java Tool use it to transport information about nodes...

@see: built() and Node.release_target_info()  Overrides:
SCons.Node.Node.release_target_info

dfind_src_builder(self)

has_src_builder(self)

Return whether this Node has a source builder or not.

If this Node doesn’t have an explicit source code builder, this is where we figure out, on the fly, if there’s a transparent source code builder for it.

Note that if we found a source builder, we also set the self.builder attribute, so that all of the methods that actually build this file don’t have to do anything different.

alter_targets(self)

Return any corresponding targets in a variant directory.  Overrides:
SCons.Node.Node.alter_targets
make_ready(self)
Get a Node ready for evaluation.

This is called before the Taskmaster decides if the Node is up-to-date or not. Overriding this method allows for a Node subclass to be disambiguated if necessary, or for an implicit source builder to be attached. Overrides: SCons.Node.Node.make_ready (inherited documentation)

prepare(self)

Prepare for this file to be created. Overrides: SCons.Node.Node.prepare

remove(self)


do_duplicate(self, src)

exists(self)

Does this node exists? Overrides: SCons.Node.Node.exists (inherited documentation)

get_max_drift_csig(self)

Returns the content signature currently stored for this node if it’s been unmodified longer than the max_drift value, or the max_drift value is 0. Returns None otherwise.

get_csig(self)

Generate a node’s content signature, the digested signature of its content.

node - the node cache - alternate node to use for the signature cache returns - the content signature Overrides: SCons.Node.Node.get_csig
**builder_set**(*self, builder*)


**built**(*self*)

Called just after this File node is successfully built.

Just like for `release_target_info` we try to release some more target node attributes in order to minimize the overall memory consumption.

@see: release_target_info  Overrides: SCons.Node.Node.built

**changed**(*self, node=None, allowcache=False*)

Returns if the node is up-to-date with respect to the BuildInfo stored last time it was built.

For File nodes this is basically a wrapper around Node.changed(), but we allow the return value to get cached after the reference to the Executor got released in release_target_info().

@see: Node.changed()  Overrides: SCons.Node.Node.changed

**changed_content**(*self, target, prev_ni*)

**changed_state**(*self, target, prev_ni*)

**changed_timestamp_then_content**(*self, target, prev_ni*)

**changed_timestamp_newer**(*self, target, prev_ni*)

**changed_timestamp_match**(*self, target, prev_ni*)

**is_up_to_date**(*self*)

Default check for whether the Node is current: unknown Node subtypes are always out of date, so they will always get built.  Overrides: SCons.Node.Node.is_up_to_date  extit(inherited documentation)
**rfile**(*self*)

Overrides: SCons.Node.FS.Base.rfile

**rstr**(*self*)

A Node.FS.Base object’s string representation is its path name. Overrides: SCons.Node.FS.Base.rstr (inherited documentation)

**get__cachedir__csig**(*self*)

Fetch a Node’s content signature for purposes of computing another Node’s cachesig.

This is a wrapper around the normal get_csig() method that handles the somewhat obscure case of using CacheDir with the -n option. Any files that don’t exist would normally be 'built' by fetching them from the cache, but the normal get_csig() method will try to open up the local file, which doesn’t exist because the -n option meant we didn’t actually pull the file from cachedir. But since the file does actually exist in the cachedir, we can use its contents for the csig. Overrides: SCons.Node.Node.get__cachedir__csig

**get__contents__sig**(*self*)

A helper method for get__cachedir__bsig.

It computes and returns the signature for this node’s contents.

**get__cachedir__bsig**(*self*)

Return the signature for a cached file, including its children.

It adds the path of the cached file to the cache signature, because multiple targets built by the same action will all have the same build signature, and we have to differentiate them somehow.

**Inherited from SCons.Node.FS.Base** *(Section 15.6)*

RDirs(), Rfindalldirs(), __getattr__(), __lt__(), __str__(), for_signature(), get_abspath(), get_dir(), get_internal_path(), get_labspath(), get_path(), get_path_elements(), get_subst_proxy(), get_suffix(), get_tpath(), getmtime(), getsize(), is_under(), isdir(), isfile(), islink(), must_be_same(), rentry(), rexists(), set_local(), set_src_builder(),
src_builder(), srcnode(), stat(), str_for_display(), target_from_source()

**Inherited from SCons.Node.Node (Section 13.6)**

Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(),
add_source(), add_toImplicit(), add_to_waiting_parents(), add_to_waiting_se(),
add_wkid(), all_children(), build(), children(), children_are_up_to_date(), clear(),
clear_memoized_values(), del_binfo(), disambiguate(), env_set(), executor_cleanup(),
explain(), get_binfo(), get_build_env(), get_build_scanner_path(), get_builder(),
get_env(), get_env_scanner(), get_executor(), get_implicit_deps(), get_ninfo(),
get_source_scanner(), get_state(), get_string(), get_target_scanner(), has_builder(),
has_explicit_builder(), is_derived(), is_literal(), missing(), multiple_side_effect_has_builder(),
new_binfo(), new_ninfo(), postprocess(), render_include_tree(), reset_executor(),
scan(), select_scanner(), set_always_build(), set_executor(), set_explicit(), set_nocache(),
set_noclean(), set_precious(), set_pseudo(), set_specific_source(), set_state()

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()

### 15.16.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachedir_csig</td>
<td></td>
</tr>
<tr>
<td>cachesig</td>
<td></td>
</tr>
<tr>
<td>contentsig</td>
<td></td>
</tr>
<tr>
<td>dirname</td>
<td></td>
</tr>
<tr>
<td>entries</td>
<td></td>
</tr>
<tr>
<td>on_disk_entries</td>
<td></td>
</tr>
<tr>
<td>released_target_info</td>
<td></td>
</tr>
<tr>
<td>repositories</td>
<td></td>
</tr>
<tr>
<td>root</td>
<td></td>
</tr>
<tr>
<td>scanner_paths</td>
<td></td>
</tr>
<tr>
<td>searched</td>
<td></td>
</tr>
<tr>
<td>srcdir</td>
<td></td>
</tr>
<tr>
<td>variant_dirs</td>
<td></td>
</tr>
</tbody>
</table>

*Inherited from SCons.Node.FS.Base (Section 15.6)*

cwd, dir, duplicate, name, sbuilder

*Inherited from SCons.Node.Node (Section 13.6)*

*continued on next page*
### Name | Description
---|---
always_build, attributes, binfo, builder, cached, changed_since_last_build, depends, depends_set, env, executor, ignore, ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect, side_effects, sources, sources_set, state, store_info, waiting_parents, waiting_s_e, wkids

*Inherited from object__class__*

15.16.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>md5_chunksize</td>
<td>Value: 64</td>
</tr>
<tr>
<td>convert_copyattrs</td>
<td>Value: ['bsources', 'bimplicit', 'bdepends', 'bact', 'bactsig', ...]</td>
</tr>
<tr>
<td>convert_sigattrs</td>
<td>Value: ['bsourcesigs', 'bimplicitssigs', 'bdependssigs']</td>
</tr>
</tbody>
</table>

15.16.4 Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.FS.Base (Section 15.6)</td>
<td></td>
</tr>
</tbody>
</table>

15.17 Class FileFinder

object —

SCons.Node.FS.FileFinder

15.17.1 Methods

```python
__init__(self)
```

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ extit(inherited documentation)
filedir_lookup(self, p, fd=None)

A helper method for find_file() that looks up a directory for a file we’re trying to find. This only creates the Dir Node if it exists on-disk, since if the directory doesn’t exist we know we won’t find any files in it... :-)

It would be more compact to just use this as a nested function with a default keyword argument (see the commented-out version below), but that doesn’t work unless you have nested scopes, so we define it here just so this work under Python 1.5.2.

find_file(self, filename, paths, verbose=None)

Find a node corresponding to either a derived file or a file that exists already. Only the first file found is returned, and none is returned if no file is found.

filename: A filename to find
paths: A list of directory path nodes to search in. Can be represented as a list, a tuple, or a callable that is called with no arguments and returns the list or tuple.

returns The node created from the found file.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

15.17.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
16 Module SCons.Node.Python

scons.Node.Python

Python nodes.

16.1 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Node/Python.py rel_3.0.0:4395:8972f6a2f...'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Node'</td>
</tr>
</tbody>
</table>

16.2 Class ValueNodeInfo

object

SCons.Node.NodeInfoBase


The generic base class for signature information for a Node.

Node subclasses should subclass NodeInfoBase to provide their own logic for dealing with their own Node-specific signature information.

16.2.1 Methods

```python
str_to_node(self, s)
```

```python
__getstate__(self)
```

Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a '__dict__' slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class. Overrides: SCons.Node.NodeInfoBase.__getstate__
Class ValueBuildInfo

Module SCons.Node.Python

```python
__setstate__(self, state)
```

Restore the attributes from a pickled state. Overrides:
SCons.Node.NodeInfoBase.__setstate__

Inherited from SCons.Node.NodeInfoBase (Section 13.4)

convert(), format(), merge(), update()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(),
__new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(),
__sizeof__(), __str__(), __subclasshook__()

16.2.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csig</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object

__class__

16.2.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
<tr>
<td>field_list</td>
<td>Value: ['csig']</td>
</tr>
</tbody>
</table>

16.3 Class ValueBuildInfo

object

SCons.Node.BuildInfoBase


The generic base class for build information for a Node.

This is what gets stored in a .sconsign file for each target file. It contains a NodeInfo instance for this node (signature information that’s specific to the type of Node) and direct attributes for the generic build stuff we have to track: sources, explicit dependencies, implicit dependencies, and action information.
16.3.1 Methods

*Inherited from SCons.Node.BuildInfoBase (Section 13.5)*

___getstate___(), ___init___(), ___setstate___(), merge()

*Inherited from object*

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(),
___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(),
___str___(), ___subclasshook___()

16.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</em></td>
<td></td>
</tr>
<tr>
<td>bact, bactsig, bdepends, bdependsgigs, bimplicit, bimplicitgsigs, bsources, bsourcesigs</td>
<td></td>
</tr>
<tr>
<td><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><em><strong>class</strong></em></td>
<td></td>
</tr>
</tbody>
</table>

16.3.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
</tbody>
</table>

16.4 Class Value

```
object

SCons.Node.Node

```

A class for Python variables, typically passed on the command line or generated by a script, but not from a file or some other source.
16.4.1 Methods

```
__init__(self, value, built_value=None)
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)
```

```
str_for_display(self)
```

```
__str__(self)
str(x)  Overrides: object.__str__ extit(inherited documentation)
```

```
make_ready(self)
Get a Node ready for evaluation.
This is called before the Taskmaster decides if the Node is up-to-date or not.
Overriding this method allows for a Node subclass to be disambiguated if necessary, or for an implicit source builder to be attached.  Overrides:
SCons.Node.Node.make_ready extit(inherited documentation)
```

```
build(self, **kw)
Actually build the node.
This is called by the Taskmaster after it’s decided that the Node is out-of-date
and must be rebuilt, and after the prepare() method has gotten everything,
uh, prepared.
This method is called from multiple threads in a parallel build, so only do
thread safe stuff here. Do thread unsafe stuff in built().  Overrides:
SCons.Node.Node.build extit(inherited documentation)
```

```
is__up__to__date(self)
Alternate check for whether the Node is current: If all of our children were
up-to-date, then this Node was up-to-date, too.
current() method to this method.  Overrides:
SCons.Node.Node.is_up_to_date
```
is_under(self, dir)

write(self, built_value)

Set the value of the node.

read(self)

Return the value. If necessary, the value is built.

get_text_contents(self)

By the assumption that the node.built_value is a deterministic product of the sources, the contents of a Value are the concatenation of all the contents of its sources. As the value need not be built when get_contents() is called, we cannot use the actual node.built_value.

get_contents(self)


changed_since_last_build(self, target, prev_ni)

Overrides: SCons.Node.Node.changed_since_last_build

get_csig(self, calc=None)

Because we’re a Python value node and don’t have a real timestamp, we get to ignore the calculator and just use the value contents. Overrides: SCons.Node.Node.get_csig

Inherited from SCons.Node.Node(Section 13.6)

Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(),
add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_s_e(),
add_wkid(), all_children(), alter_targets(), builder_set(), built(), changed(), chil-
dren(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(),
disambiguate(), env_set(), executor_cleanup(), exists(), explain(), for_signature(),
get_abspath(), get_binfo(), get_build_env(), get_build_scanner_path(), get_builder(),
get_cachedir_csig(), get_env(), get_env_scanner(), get_executor(), get_found_includes(),
get_implicit_deps(), get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(),
get_stored_info(), get_string(), get_subst_proxy(), get_suffix(), get_target_scanner(),
has_builder(), has_explicit_builder(), is_derived(), is_literal(), missing(), multiple_side_effect_has_builder(), new_binfo(), new_ninfo(), postprocess(), prepare(), push_to_cache(), release_target_info(), remove(), render_include_tree(),
reset_executor(), retrieve_from_cache(), rexists(), scan(), scanner_key(), select_scanner(),
set_always_build(), set_executor(), set_explicit(), set_nocache(), set_noclean(),
set_precious(), set_pseudo(), set_specific_source(), set_state(), visited()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
17 Module SCons.PathList

SCons.PathList

A module for handling lists of directory paths (the sort of things that get set as CPPPATH, LIBPATH, etc.) with as much caching of data and efficiency as we can, while still keeping the evaluation delayed so that we Do the Right Thing (almost) regardless of how the variable is specified.

17.1 Functions

node_conv(obj)

This is the 'string conversion' routine that we have our substitutions use to return Nodes, not strings. This relies on the fact that an EntryProxy object has a get() method that returns the underlying Node that it wraps, which is a bit of architectural dependence that we might need to break or modify in the future in response to additional requirements.

PathList(pathlist)

Returns the cached _PathList object for the specified pathlist, creating and caching a new object as necessary.

17.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/PathList.py rel_3.0.0:4395:8972f6a2f699...'</td>
</tr>
<tr>
<td><strong>doc</strong></td>
<td>Value: &quot;&quot;&quot;SCons.PathL...</td>
</tr>
<tr>
<td>TYPE_STRING_NO_SUBST</td>
<td>Value: 0</td>
</tr>
<tr>
<td>TYPE_STRING_SUBST</td>
<td>Value: 1</td>
</tr>
<tr>
<td>TYPE_OBJECT</td>
<td>Value: 2</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>
18 Module SCons.SConf

SCons.SConf

Autoconf-like configuration support.

In other words, SConf allows to run tests on the build machine to detect capabilities of system and do some things based on result: generate config files, header files for C/C++, update variables in environment.

Tests on the build system can detect if compiler sees header files, if libraries are installed, if some command line options are supported etc.

18.1 Functions

SetBuildType(type)

Set the Configure cache mode. mode must be one of "auto", "force", or 'cache'.

SetCacheMode(mode)

Set the progress display to use (called from SCons.Script)

SetProgressDisplay(display)

NeedConfigHBuilder()

Called if necessary just before the building targets phase begins.

SConf(*args, **kw)

CheckFunc(context, function_name, header=None, language=None)

CheckType(context, type_name, includes='', language=None)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CheckTypeSize(context, type_name, includes='', language=None, expect=None)</code></td>
<td>Check function size at runtime.</td>
</tr>
<tr>
<td><code>CheckDeclaration(context, declaration, includes='', language=None)</code></td>
<td>Check that declaration compiles as expected.</td>
</tr>
<tr>
<td><code>createIncludesFromHeaders(headers, leaveLast, include_quotes='')</code></td>
<td>Create includes for the given header files.</td>
</tr>
<tr>
<td><code>CheckHeader(context, header, include_quotes='&lt;&gt;', language=None)</code></td>
<td>A test for a C or C++ header file.</td>
</tr>
<tr>
<td><code>CheckCC(context)</code></td>
<td>Test C compilation.</td>
</tr>
<tr>
<td><code>CheckCXX(context)</code></td>
<td>Test C++ compilation.</td>
</tr>
<tr>
<td><code>CheckSHCC(context)</code></td>
<td>Test shell C compilation.</td>
</tr>
<tr>
<td><code>CheckSHCXX(context)</code></td>
<td>Test shell C++ compilation.</td>
</tr>
<tr>
<td><code>CheckCHeader(context, header, include_quotes='')</code></td>
<td>A test for a C header file.</td>
</tr>
<tr>
<td><code>CheckCXXHeader(context, header, include_quotes='')</code></td>
<td>A test for a C++ header file.</td>
</tr>
<tr>
<td><code>CheckLib(context, library=None, symbol='main', header=None, language=None, autoadd=1)</code></td>
<td>A test for a library. See also CheckLibWithHeader. Note that library may also be None to test whether the given symbol compiles without flags.</td>
</tr>
</tbody>
</table>
**Variables**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/SConf.py rel_3.0.0:4395:8972f6a2f699 20...</td>
</tr>
<tr>
<td>build_type</td>
<td>Value: None</td>
</tr>
<tr>
<td>build_types</td>
<td>Value: ['clean', 'help']</td>
</tr>
<tr>
<td>dryrun</td>
<td>Value: 0</td>
</tr>
<tr>
<td>AUTO</td>
<td>Value: 0</td>
</tr>
<tr>
<td>FORCE</td>
<td>Value: 1</td>
</tr>
<tr>
<td>CACHE</td>
<td>Value: 2</td>
</tr>
<tr>
<td>cache_mode</td>
<td>Value: 0</td>
</tr>
<tr>
<td>progress_display</td>
<td>Value: &lt;SCons.Util.DisplayEngine object&gt;</td>
</tr>
<tr>
<td>SConfFS</td>
<td>Value: None</td>
</tr>
<tr>
<td>sconf_global</td>
<td>Value: None</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>
18.3 Class SConfWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.SConf.SConfWarning

18.3.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()  

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()  

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasscheck__()  

18.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
18.4 Class SConfError

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.SConf.SConfError

Known Subclasses: SCons.SConf.ConfigureCacheError, SCons.SConf.ConfigureDryRunError

18.4.1 Methods

__init__(self, msg)

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

_new__()

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

18.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
18.5 Class ConfigureDryRunError

Raised when a file or directory needs to be updated during a Configure process, but the user requested a dry-run

18.5.1 Methods

```python
__init__(self, target)
```

Inherited from exceptions.Exception

```
__new__()
```

Inherited from exceptions.BaseException

```
__delattr__, __getattr__, __getattribute__, __getitem__, __getslice__, __reduce__, __repr__, __setattr__, __setstate__, __str__, __unicode__
```

Inherited from object

```
__format__, __hash__, __reduce_ex__, __sizeof__, __subclasshook__
```

18.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>args, message</td>
<td>Inherited from exceptions.BaseException</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td>Inherited from object</td>
</tr>
</tbody>
</table>

18.6 Class ConfigureCacheError

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.SConf.SConfError

SCons.SConf.ConfigureCacheError

Raised when a use explicitly requested the cache feature, but the test is run the first time.

18.6.1 Methods

```python
__init__(self, target)
```

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

```python
__new__()```

Inherited from exceptions.BaseException

```python
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()
```

Inherited from object

```python
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

18.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
Class SConfBuildInfo

object

SCons.Node.BuildInfoBase

SCons.Node.FS.FileBuildInfo

SCons.SConf.SConfBuildInfo

Special build info for targets of configure tests. Additional members are result (did the builder succeed last time?) and string, which contains messages of the original build phase.

18.7.1 Methods

```python
__init__(self)
```

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides:

```python
object.__init__
```

```python
set__build__result(self, result, string)
```

Inherited from SCons.Node.FS.FileBuildInfo (Section 15.15)

convert_from_sconsign(), convert_to_sconsign(), format(), prepare_dependencies()

Inherited from SCons.Node.BuildInfoBase (Section 13.5)

```python
__getstate__(), __setstate__(), merge()
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

18.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td></td>
</tr>
<tr>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from SCons.Node.BuildInfoBase (Section 13.5)

bact, bactsig, bdepends, bdependsigs, bimplicit, bimplicitargs, bsources, bsourcesigs

Inherited from object

continued on next page
18.7.3  Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Inherited from SCons.Node.FS.FileBuildInfo (Section 15.15)</td>
</tr>
</tbody>
</table>

18.8  Class Streamer

`object` ↳

**SCons.SConf.Streamer**

'Sniffer' for a file-like writable object. Similar to the unix tool tee.

18.8.1  Methods

```python
__init__(self, orig)
```

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides:

object.__init__ extit(inherited documentation)

```python
write(self, str)
```

```python
writelines(self, lines)
```

```python
getvalue(self)
```

Return everything written to orig since the Streamer was created.

```python
flush(self)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```
18.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

18.9 Class SConfBuildTask

object

SCons.Taskmaster.Task

SCons.Taskmaster.AlwaysTask

SCons.SConf.SConfBuildTask

This is almost the same as SCons.Script.BuildTask. Handles SConfErrors correctly and knows about the current cache_mode.

18.9.1 Methods

**display**(self, message)

Hook to allow the calling interface to display a message.

This hook gets called as part of preparing a task for execution (that is, a Node to be built). As part of figuring out what Node should be built next, the actual target list may be altered, along with a message describing the alteration. The calling interface can subclass Task and provide a concrete implementation of this method to see those messages. Overrides: SCons.Taskmaster.Task.display extit(inherited documentation)

**display_cached_string**(self, bi)

Logs the original builder messages, given the SConfBuildInfo instance bi.
failed(self)

Default action when a task fails: stop the build.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure(). Overrides: SCons.Taskmaster.Task.failed (inherited documentation)

collect_node_states(self)

execute(self)

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in prepare(), executed() or failed(). Overrides: SCons.Taskmaster.Task.execute (inherited documentation)

Inherited from SCons.Taskmaster.AlwaysTask (Section 35.5)

needs_execute()

Inherited from SCons.Taskmaster.Task (Section 35.4)

__init__(), exc_clear(), exc_info(), exception_set(), executed(), executed_with_callbacks(),
executed_without_callbacks(), fail_continue(), fail_stop(), get_target(), make_ready(),
make_ready_all(), make_ready_current(), postprocess(), prepare(), trace_message()
18.10 Class SConfBase

object  
| SConfBase |

This is simply a class to represent a configure context. After creating a SConf object, you can call any tests. After finished with your tests, be sure to call the Finish() method, which returns the modified environment. Some words about caching: In most cases, it is not necessary to cache Test results explicitly. Instead, we use the scons dependency checking mechanism. For example, if one wants to compile a test program (SConf.TryLink), the compiler is only called, if the program dependencies have changed. However, if the program could not be compiled in a former SConf run, we need to explicitly cache this error.

18.10.1 Methods

__init__(self, env, custom_tests={}, conf_dir='CONFIGUREDIR', log_file='CONFIGURELOG', config_h=None, _depth=0)

Constructor. Pass additional tests in the custom_tests-dictionary, e.g. custom_tests={'CheckPrivate':MyPrivateTest}, where MyPrivateTest defines a custom test. Note also the conf_dir and log_file arguments (you may want to build tests in the VariantDir, not in the SourceDir)  Overrides: object.__init__

Finish(self)

Call this method after finished with your tests: env = sconf.Finish()

Define(self, name, value=None, comment=None)

Define a pre processor symbol name, with the optional given value in the current config header.

If value is None (default), then #define name is written. If value is not none, then #define name value is written.

comment is a string which will be put as a C comment in the header, to explain the meaning of the value (appropriate C comments will be added automatically).
**BuildNodes**(*self, nodes*)

Tries to build the given nodes immediately. Returns 1 on success, 0 on error.

**pspawn__wrapper**(*self, sh, escape, cmd, args, env*)

Wrapper function for handling piped spawns.

This looks to the calling interface (in Action.py) like a 'normal' spawn, but associates the call with the PSPAWN variable from the construction environment and with the streams to which we want the output logged. This gets slid into the construction environment as the SPAWN variable so Action.py doesn’t have to know or care whether it’s spawning a piped command or not.

**TryBuild**(*self, builder, text=None, extension='')

Low level TryBuild implementation. Normally you don’t need to call that - you can use TryCompile / TryLink / TryRun instead.

**TryAction**(*self, action, text=None, extension='')

Tries to execute the given action with optional source file contents <text> and optional source file extension <extension>, Returns the status (0 : failed, 1 : ok) and the contents of the output file.

**TryCompile**(*self, text, extension*)

Compiles the program given in text to an env.Object, using extension as file extension (e.g. '.c'). Returns 1, if compilation was successful, 0 otherwise. The target is saved in self.lastTarget (for further processing).
TryLink(self, text, extension)

Compiles the program given in text to an executable env.Program, using extension as file extension (e.g. `.c`). Returns 1, if compilation was successful, 0 otherwise. The target is saved in self.lastTarget (for further processing).

TryRun(self, text, extension)

Compiles and runs the program given in text, using extension as file extension (e.g. `.c`). Returns (1, outputStr) on success, (0, ”) otherwise. The target (a file containing the program’s stdout) is saved in self.lastTarget (for further processing).

AddTest(self, test_name, test_instance)

Adds test_class to this SConf instance. It can be called with self.test_name(...)

AddTests(self, tests)

Adds all the tests given in the tests dictionary to this SConf instance

Inherited from object

_delattr_(), _format_(), _getattr_(), _hash_(), _new_(),
_reduce_(), _reduce_ex_(), _repr_(), _setattr_(), _sizeof_(),
__str__(), __subclasshook__()

18.10.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Inherited from object
| __class__           |             |
18.11 Class CheckContext

```
object
    SCons.SConf.CheckContext
```

Provides a context for configure tests. Defines how a test writes to the screen and log file.

A typical test is just a callable with an instance of CheckContext as first argument:

```
def CheckCustom(context, ...):
    context.Message('Checking my weird test ...
    ') ret = myWeirdTestFunction(...) context.Result(ret)
```

Often, myWeirdTestFunction will be one of context.TryCompile/context.TryLink/context.TryRun. The results of those are cached, for they are only rebuild, if the dependencies have changed.

18.11.1 Methods

```
__init__(self, sconf)
```

Constructor. Pass the corresponding SConf instance. Overrides:

```
object.__init__
```

```
Message(self, text)
```

Inform about what we are doing right now, e.g. 'Checking for SOMETHING ...

```
Result(self, res)
```

Inform about the result of the test. If res is not a string, displays 'yes' or 'no' depending on whether res is evaluated as true or false. The result is only displayed when self.did_show_result is not set.

```
TryBuild(self, *args, **kw)
```

```
TryAction(self, *args, **kw)
```

```
TryCompile(self, *args, **kw)
```

222
Class CheckContext

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
19 Module SCons.SConsign

SCons.SConsign
Writing and reading information to the .sconsign file or files.

19.1 Functions

- **corrupt_dblite_warning(filename)**
- **Get__DataBase(dir)**
- **Reset()**
  
  Reset global state. Used by unit tests that end up using SConsign multiple times to get a clean slate for each test.
- **write()**
- **File(name, dbm_module=None)**
  
  Arrange for all signatures to be stored in a global .sconsign.db* file.

19.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_<em><strong>revision</strong></em></td>
<td>Value: <code>src/engine/SCons/SConsign.py</code> rel_3.0.0:4395:8972f6a2f699...</td>
</tr>
<tr>
<td>sig_files</td>
<td>Value: []</td>
</tr>
<tr>
<td>Database</td>
<td>Value: {}</td>
</tr>
<tr>
<td>DB_Name</td>
<td>Value: <code>.sconsign</code></td>
</tr>
<tr>
<td>DB_sync_list</td>
<td>Value: []</td>
</tr>
<tr>
<td>_<em><strong>package</strong></em></td>
<td>Value: <code>SCons</code></td>
</tr>
</tbody>
</table>
19.3 Class SConsignEntry

object → SCons.SConsign.SConsignEntry

Wrapper class for the generic entry in a .sconsign file. The Node subclass populates it with attributes as it pleases.

XXX As coded below, we do expect a `.binfo` attribute to be added, but we'll probably generalize this in the next refactorings.

19.3.1 Methods

```python
__init__(self)

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ (inherited documentation)
```

```python
convert_to_sconsign(self)
```

```python
convert_from_sconsign(self, dir, name)
```

```python
__getstate__(self)
```

```python
__setstate__(self, state)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

19.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binfo</td>
<td></td>
</tr>
<tr>
<td>ninfo</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object

```python
__class__
```

19.3.3 Class Variables
Class Base Module SCons.SConsign

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
</tbody>
</table>

19.4 Class Base

object

SCons.SConsign.Base

Known Subclasses: SCons.SConsign.DB, SCons.SConsign.Dir

This is the controlling class for the signatures for the collection of entries associated with a specific directory. The actual directory association will be maintained by a subclass that is specific to the underlying storage method. This class provides a common set of methods for fetching and storing the individual bits of information that make up signature entry.

19.4.1 Methods

```python
__init__(self)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides:
object.__init__ extit(inherited documentation)
```

```python
get_entry(self, filename)

Fetch the specified entry attribute.
```

```python
set_entry(self, filename, obj)

Set the entry.
```

```python
do_not_set_entry(self, filename, obj)
```

```python
store_info(self, filename, node)
```

```python
do_not_store_info(self, filename, node)
```

```python
merge(self)
```

226
Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
19.6 Class Dir

object

SCons.SConsign.Base

SCons.SConsign.Dir

Known Subclasses: SCons.SConsign.DirFile

19.6.1 Methods

```python
__init__(self, fp=None, dir=None)
```

fp - file pointer to read entries from  

**Overrides:** object.__init__

*Inherited from SCons.SConsign.Base (Section 19.4)*

do_not_set_entry(), do_not_store_info(), get_entry(), merge(), set_entry(), store_info()

*Inherited from object*

_delattr_(), _format_(), _getattribute_(), _hash_(), _new_(), _reduce_(), _reduce_ex_(), _repr_(), _setattr_(), _sizeof_(), _str_(), _subclasshook_()

19.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
19.7 Class DirFile

object

SCons.SConsign.Base

SCons.SConsign.Dir

SCons.SConsign.DirFile

Encapsulates reading and writing a per-directory .sconsign file.

19.7.1 Methods

___init__(self, dir)

dir - the directory for the file  Overrides: object.__init___

write(self, sync=1)

Write the .sconsign file to disk.

Try to write to a temporary file first, and rename it if we succeed. If we can’t write to the temporary file, it’s probably because the directory isn’t writable (and if so, how did we build anything in this directory, anyway?), so try to write directly to the .sconsign file as a backup. If we can’t rename, try to copy the temporary contents back to the .sconsign file. Either way, always try to remove the temporary file at the end.

Inherited from SCons.SConsign.Base(Section 19.4)

 do_not_set_entry(), do_not_store_info(), get_entry(), merge(), set_entry(), store_info()

Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
## 19.8 Class DB

A Base subclass that reads and writes signature information from a global .sconsign.db* file—the actual file suffix is determined by the database module.

### 19.8.1 Methods

- **__init__(self, dir)**
  
  `x.__init__(...)` initializes `x`; see help(type(x)) for signature. Overrides: object.__init__ (inherited documentation)

- **write(self, sync=1)**

  Inherited from SCons.SConsign.Base (Section 19.4)

  do_not_set_entry(), do_not_store_info(), get_entry(), merge(), set_entry(), store_info()

  Inherited from object

  __delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
  __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
  __str__(), __subclasshook__()

### 19.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

| Inherited from object
| __class__     |                             |
20 Package SCons.Scanner

SCons.Scanner

The Scanner package for the SCons software construction utility.

20.1 Modules

- **C**: SCons.Scanner.C
  *(Section 21, p. 232)*
- **D**: SCons.Scanner.D
  *(Section 22, p. 235)*
- **Dir**: SCons.Scanner.Dir
  *(Section 23, p. 239)*
- **Fortran**: SCons.Scanner.Fortran
  *(Section 24, p. 241)*
- **IDL**: SCons.Scanner.IDL
  *(Section 25, p. 246)*
- **LaTeX**: SCons.Scanner.LaTeX
  *(Section 26, p. 247)*
- **Prog**: SCons.Scanner.Prog
  *(Section 27, p. 254)*
- **RC**: SCons.Scanner.RC
  *(Section 28, p. 255)*
- **SWIG**: SCons.Scanner.SWIG
  *(Section 29, p. 256)*

20.2 Functions

```
Scanner(function, *args, **kw)
```

Public interface factory function for creating different types of Scanners based on the different types of 'functions' that may be supplied.

TODO: Deprecate this some day. We’ve moved the functionality inside the Base class and really don’t need this factory function any more. It was, however, used by some of our Tool modules, so the call probably ended up in various people’s custom modules patterned on SCons code.

20.3 Variables
20.4 Class FindPathDirs

object SCons.Scanner.FindPathDirs

A class to bind a specific `E{PATH}` variable name to a function that will return all of the `E{PATH}` path directories.

20.4.1 Methods

```python
__init__(self, variable)
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ (inherited documentation)
```

```python
__call__(self, env, dir=None, target=None, source=None, argument=None)

Inherited from object
```

- _delattr_, _format_, _getattribute_, _hash_, _new_, _reduce_, _reduce_ex_, _repr_, _setattr_, _sizeof_, _str_, _subclasshook_

20.4.2 Properties

```text
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/<strong>init</strong>.py rel_3.0.0:4395:8972...'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td>232</td>
</tr>
</tbody>
</table>
```

Inherited from object
20.5 Class Base

object  SCons.Scanner.Base


The base class for dependency scanners. This implements straightforward, single-pass scanning of a single file.

20.5.1 Methods

```
__call__(self, node, env, path=())
```

This method scans a single object. 'node' is the node that will be passed to the scanner function, and 'env' is the environment that will be passed to the scanner function. A list of direct dependency nodes for the specified node will be returned.

```
__eq__(self, other)
```

```
__hash__(self)
```

hash(x) Overrides: object.__hash__ extit(inherited documentation)
Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path_function.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SCons script file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node_class' - the class of Nodes which this scan will return. If node_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node_class objects.

'scan_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function’s first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path_function, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

s = Scanner(my_scanner_function)

s = Scanner(function = my_scanner_function)
__str__(self)

str(x) Overrides: object.__str__ (inherited documentation)

add_scanner(self, skey, scanner)

add_skey(self, skey)

Add a skey to the list of skeys

get_skeys(self, env=None)

path(self, env, dir=None, target=None, source=None)

select(self, node)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

20.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

20.6 Class Selector

A class for selecting a more specific scanner based on the scanner_key() (suffix) for a specific Node.

TODO: This functionality has been moved into the inner workings of the Base class, and this class will be deprecated at some point. (It was never exposed directly as part of the public interface, although it is used by the Scanner() factory function that was used by various Tool
modules and therefore was likely a template for custom modules that may be out there.)
20.6.1 Methods

```python
__init__(self, dict, *args, **kw)
```

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path_function.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node_class' - the class of Nodes which this scan will return. If node_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node_class objects.

'scan_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function’s first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path_function, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```python
s = Scanner(my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function, argument = 'foo')
```

Overrides:

```python
object.__init__
```
```
__call__(self, node, env, path=())
```

This method scans a single object. 'node' is the node that will be passed to
the scanner function, and 'env' is the environment that will be passed to the
scanner function. A list of direct dependency nodes for the specified node will
be returned. Overrides: SCons.Scanner.Base.__call__ extit(inherited
documentation)

```
select(self, node)
```

Overrides: SCons.Scanner.Base.select

```
add_scanner(self, skey, scanner)
```

Overrides: SCons.Scanner.Base.add_scanner

**Inherited from SCons.Scanner.Base (Section 20.5)**

```
__eq__(), __hash__(), __str__(), add_skey(), get_skeys(), path()
```

**Inherited from object**

```
__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()
```

### 20.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 20.7 Class Current

```
object
```

```
SCons.Scanner.Base
```

```
SCons.Scanner.Current
```

**Known Subclasses:** SCons.Scanner.Classic

A class for scanning files that are source files (have no builder) or are derived files and are
current (which implies that they exist, either locally or in a repository).
20.7.1 Methods

```python
__init__(self, *args, **kw)
```

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path_function.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node_class' - the class of Nodes which this scan will return. If node_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node_class objects.

'scan_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function’s first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path_function, and the fourth argument will be the value passed into ‘argument’, and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```python
s = Scanner(my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function, argument = 'foo')
```

Overrides:

object.__init__(inherited documentation)
Inherited from SCons.Scanner.Base (Section 20.5)

\_call\_(), \_eq\_(), \_hash\_(), \_str\_(), add\_scanner(), add\_skey(),
get\_skeys(), path(), select()

Inherited from object

\_delattr\_(), \_format\_(), \_getattribute\_(), \_new\_(), \_reduce\_(),
\_reduce\_ex\_(), \_repr\_(), \_setattr\_(), \_sizeof\_(), \_subclasshook\_()

20.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td>_class_</td>
<td></td>
</tr>
</tbody>
</table>

20.8 Class Classic

object ─

SCons.Scanner.Base ─

SCons.Scanner.Current ─

SCons.Scanner.Classic


A Scanner subclass to contain the common logic for classic CPP-style include scanning, but which can be customized to use different regular expressions to find the includes.

Note that in order for this to work "out of the box" (without overriding the find\_include() and sort\_key() methods), the regular expression passed to the constructor must return the name of the include file in group 0.
20.8.1 Methods

```python
__init__(self, name, suffixes, path_variable, regex, *args, **kw)
```

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path_function.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node_class' - the class of Nodes which this scan will return. If node_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node_class objects.

'scan_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function’s first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path_function, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```python
s = Scanner(my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function, argument = 'foo')
```
find__include__(self, include, source_dir, path)

find__include__names(self, node)

scan(self, node, path=())

sort__key(self, include)

Inherited from SCons.Scanner.Base(Section 20.5)

__call__(), __eq__(), __hash__(), __str__(), add_scanner(), add_skey(),
get_skeys(), path(), select()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

20.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

20.9 Class ClassicCPP

A Classic Scanner subclass which takes into account the type of bracketing used to include the file, and uses classic CPP rules for searching for the files based on the bracketing.

Note that in order for this to work, the regular expression passed to the constructor must return the leading bracket in group 0, and the contained filename in group 1.
20.9.1 Methods

```
find__include(self, include, source_dir, path)
Overrides: SCons.Scanner.Classic.find__include
```

```
sort__key(self, include)
Overrides: SCons.Scanner.Classic.sort__key
```

*Inherited from SCons.Scanner.Classic (Section 20.8)*

`__init__()`, `find__include_names()`, `scan()`

*Inherited from SCons.Scanner.Base (Section 20.5)*

`__call__()`, `__eq__()`, `__hash__()`, `__str__()`, `add__scanner()`, `add__skey()`, `get__skeys()`, `path()`, `select()`

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattribute__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

20.9.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
21 Module SCons.Scanner.C

SCons.Scanner.C

This module implements the dependency scanner for C/C++ code.

21.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dictify_CPPDEFINES(env)</code></td>
</tr>
</tbody>
</table>

Return a prototype Scanner instance for scanning source files that use the C pre-processor.

21.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `__revision__` | Value: `src/engine/SCons/Scanner/C.py
rel_3.0.0:4395:8972f6a2f69...` |
| `__package__` | Value: `'SCons.Scanner'` |

21.3 Class SConsCPPScanner

SCons-specific subclass of the cpp.py module's processing.

We subclass this so that: 1) we can deal with files represented by Nodes, not strings; 2) we can keep track of the files that are missing.
21.3.1 Methods

```python
__init__(self, *args, **kw)
```
x.__init__(...) initializes x; see help(type(x)) for signature  
Overrides:
object.__init__ extit(inherited documentation)

```python
initialize_result(self, fname)
```
Overrides: SCons.cpp.PreProcessor.initialize_result

```python
finalize_result(self, fname)
```
Overrides: SCons.cpp.PreProcessor.finalize_result

```python
find__include__file(self, t)
```
Finds the #include file for a given preprocessor tuple.  
Overrides:
SCons.cpp.PreProcessor.find__include__file extit(inherited documentation)

```python
read__file(self, file)
```
Overrides: SCons.cpp.PreProcessor.read__file

*Inherited from SCons.cpp.PreProcessor*(Section 44.4)*

__call__(), all__include__(), do__define__(), do__elif__(), do__else__(), do__endif__(), do__if__(),
do__ifdef__(), do__ifndef__(), do__import__(), do__include__(), do__include__next__(), do__nothing__(),
do__undef__(), eval__expression__(), process__contents__(), resolve__include__(), restore__()

*Inherited from object*

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

21.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
21.4 Class SConsCPPScannerWrapper

The SCons wrapper around a cpp.py scanner.

This is the actual glue between the calling conventions of generic SCons scanners, and the (subclass of) cpp.py class that knows how to look for #include lines with reasonably real C-preprocessor-like evaluation of #if/#ifdef/#else/#elif lines.

### 21.4.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__init__</code></td>
<td><code>x.__init__(...) initializes x; see help(type(x)) for signature</code></td>
</tr>
<tr>
<td><code>__call__</code></td>
<td><code>recurse_nodes(self, nodes)</code></td>
</tr>
<tr>
<td><code>select</code></td>
<td><code>select(self, node)</code></td>
</tr>
</tbody>
</table>

**Inherited from object**

- `__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`,
- `__reduce_ex__()`
- `__repr__()`,
- `__setattr__()`,
- `__sizeof__()`,
- `__str__()`,
- `__subclasshook__()`

### 21.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>
22 Module SCons.Scanner.D

SCons.Scanner.D

Scanner for the Digital Mars 'D' programming language.

Coded by Andy Friesen 17 Nov 2003

22.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DScanner()</td>
<td>Return a prototype Scanner instance for scanning D source files</td>
</tr>
</tbody>
</table>

22.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/D.py rel_3.0.0:4395:8972f6a2f69...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>

22.3 Class D

```
object
  SCons.Scanner.Base
    SCons.Scanner.Current
      SCons.Scanner.Classic
        SCons.Scanner.D.D
```
22.3.1 Methods

```python
__init__(self)
```

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path_function.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node_class' - the class of Nodes which this scan will return. If node_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node_class objects.

'scan_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function’s first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path_function, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

s = Scanner(my_scanner_function)

s = Scanner(function = my_scanner_function)

s = Scanner(function = my_scanner_function, argument = 'foo')
```

Overriding:

object.__init__ (inherited documentation)
```python
find_include(self, include, source_dir, path)
Overrides: SCons.Scanner.Classic.find_include
```

```python
find_include_names(self, node)
Overrides: SCons.Scanner.Classic.find_include_names
```

**Inherited from SCons.Scanner.Classic (Section 20.8)**

- scan(), sort_key()

**Inherited from SCons.Scanner.Base (Section 20.5)**

- __call__(), __eq__(), __hash__(), __str__(), add_scanner(), add_skey(), get_skeys(), path(), select()

**Inherited from object**

- __delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

### 22.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
Module SCons.Scanner.Dir

23.1 Functions

<table>
<thead>
<tr>
<th>only_dirs(nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirScanner(**kw)</td>
</tr>
</tbody>
</table>

Return a prototype Scanner instance for scanning directories for on-disk files

<table>
<thead>
<tr>
<th>DirEntryScanner(**kw)</th>
</tr>
</thead>
</table>

Return a prototype Scanner instance for "scanning" directory Nodes for their in-memory entries

<table>
<thead>
<tr>
<th>do_not_scan(k)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>scan_on_disk(node, env, path=())</th>
</tr>
</thead>
</table>

Scans a directory for on-disk files and directories therein.

Looking up the entries will add these to the in-memory Node tree representation of the file system, so all we have to do is just that and then call the in-memory scanning function.

<table>
<thead>
<tr>
<th>scan_in_memory(node, env, path=())</th>
</tr>
</thead>
</table>

'Scans" a Node.FS.Dir for its in-memory entries.

23.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/Dir.py rel_3.0.0:4395:8972f6a2f...</td>
</tr>
<tr>
<td>skip_entry</td>
<td>Value: {'.': 1, '..': 1, '.sconsign': 1, '.sconsign.bak': 1, '.s...</td>
</tr>
</tbody>
</table>

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip_entry_list</td>
<td>Value: ['.','..','.sconsign',</td>
</tr>
<tr>
<td></td>
<td>'.sconsign.db','.sconsign.dblite',</td>
</tr>
<tr>
<td></td>
<td>'.sconsign.d...'</td>
</tr>
<tr>
<td><strong>package</strong>_</td>
<td>Value: 'SCons.Scanner'</td>
</tr>
<tr>
<td>skip</td>
<td>Value: '.sconsign.db'</td>
</tr>
</tbody>
</table>
24 Module SCons.Scanner.Fortran

SCons.Scanner.Fortran

This module implements the dependency scanner for Fortran code.

24.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FortranScan</code></td>
<td>Return a prototype Scanner instance for scanning source files for Fortran USE &amp; INCLUDE statements</td>
</tr>
</tbody>
</table>

24.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>'src/engine/SCons/Scanner/Fortran.py rel_3.0.0:4395:8972f...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>'SCons.Scanner'</td>
</tr>
</tbody>
</table>

24.3 Class F90Scanner

A Classic Scanner subclass for Fortran source files which takes into account both USE and INCLUDE statements. This scanner will work for both F77 and F90 (and beyond) compilers.

Currently, this scanner assumes that the include files do not contain USE statements. To enable the ability to deal with USE statements in include files, add logic right after the module names are found to loop over each include file, search for and locate each USE statement, and append each module name to the list of dependencies. Caching the search...
results in a common dictionary somewhere so that the same include file is not searched multiple times would be a smart thing to do.
24.3.1 Methods

__init__(self, name, suffixes, path_variable, use_regex, incl_regex, def_regex, *args, **kw)

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path_function.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node_class' - the class of Nodes which this scan will return. If node_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node_class objects.

'scan_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function’s first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path_function, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

s = Scanner(my_scanner_function)

s = Scanner(function = my_scanner_function)
scan(self, node, env, path=())
Overrides: SCons.Scanner.Classic.scan

Inherited from SCons.Scanner.Classic(Section 20.8)
find_include(), find_include_names(), sort_key()

Inherited from SCons.Scanner.Base(Section 20.5)
__call__(), __eq__(), __hash__(), __str__(), add_scanner(), add_skey(),
get_skeys(), path(), select()

Inherited from object
__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

24.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
25 Module SCons.Scanner.IDL

SCons.Scanner.IDL

This module implements the dependency scanner for IDL (Interface Definition Language) files.

25.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLScan()</td>
<td>Return a prototype Scanner instance for scanning IDL source files</td>
</tr>
</tbody>
</table>

25.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/IDL.py rel_3.0.0:4395:8972f6a2f...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>
26 Module SCons.Scanner.LaTeX

SCons.Scanner.LaTeX

This module implements the dependency scanner for LaTeX code.

26.1 Functions

modify_env_var(env, var, abspath)

LaTeXScanner()

Return a prototype Scanner instance for scanning LaTeX source files when built with latex.

PDFLaTeXScanner()

Return a prototype Scanner instance for scanning LaTeX source files when built with pdflatex.

26.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/LaTeX.py rel_3.0.0:4395:8972f6a...'</td>
</tr>
<tr>
<td>TexGraphics</td>
<td>Value: ['.eps', '.ps']</td>
</tr>
<tr>
<td>LatexGraphics</td>
<td>Value: ['.png', '.jpg', '.gif', '.tif']</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>

26.3 Class FindENVPathDirs

object SCons.Scanner.LaTeX.FindENVPathDirs

A class to bind a specific E{*}PATH variable name to a function that will return all of the E{*}path directories.
26.3.1 Methods

```python
__init__(self, variable)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides:
object.__init__(inherited documentation)
```

```python
__call__(self, env, dir=None, target=None, source=None, argument=None)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

26.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

26.4 Class \LaTeX

```
object

SCons.Scanner.Base

SCons.Scanner.LaTeX.LaTeX
```

Class for scanning \LaTeX files for included files.

Unlike most scanners, which use regular expressions that just return the included file name, this returns a tuple consisting of the keyword for the inclusion ('include', 'includegraphics', 'input', or 'bibliography'), and then the file name itself. Based on a quick look at \LaTeX documentation, it seems that we should append .tex suffix for the 'include' keywords, append .tex if there is no extension for the 'input' keyword, and need to add .bib for the 'bibliography' keyword that does not accept extensions by itself.

Finally, if there is no extension for an 'includegraphics' keyword latex will append .ps or .eps to find the file, while pdftex may use .pdf, .jpg, .tif, .mps, or .png.

The actual subset and search order may be altered by DeclareGraphicsExtensions command. This complication is ignored. The default order corresponds to experimentation with teTeX:
$ latex --version
pdfeTeX 3.141592-1.21a-2.2 (Web2C 7.5.4)
kpathsea version 3.5.4

The order is: ['.eps', '.ps'] for latex ['.png', '.pdf', '.jpg', '.tif'].

Another difference is that the search path is determined by the type of the file being searched:
env['TEXINPUTS'] for "input" and "include" keywords
env['TEXINPUTS'] for "include-graphics" keyword
env['TEXINPUTS'] for "lstinputlisting" keyword
env['BIBINPUTS'] for "bibliography" keyword
env['BSTINPUTS'] for "bibliographystyle" keyword
env['INDEXSTYLE'] for "makeindex" keyword, no scanning support needed just allows user to set it if needed.

FIXME: also look for the class or style in document[class|style]{ }
FIXME: also look for the argument of bibliographystyle{ }
26.4.1 Methods

```python
__init__(self, name, suffixes, graphics_extensions, *args, **kw)
```

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path_function.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node_class' - the class of Nodes which this scan will return. If node_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node_class objects.

'scan_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function’s first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path_function, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```python
s = Scanner(my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function, argument = 'foo')
```

Overrides: object.__init__(inherited documentation)
The `LaTeX` module in `SCons.Scanner` provides a set of methods for processing LaTeX files, including:

- **sort_key**: Determines the sorting key for LaTeX nodes.
- **find_include**: Searches for include files based on the main file's directory.
- **canonical_text**: Standardizes input TeX-file contents.
- **scan** and **scan__recurse**: Scan for included files recursively.

**Standardize an input TeX-file contents.**

**Currently:**
- Removes comments, unwrapping comment-wrapped lines.

Inherited from `SCons.Scanner.Base` (Section 20.5)

- `__call__`, `__eq__`, `__hash__`, `__str__`, `add_scanner`, `add_skey`, `get_skeys`, `path`, `select`.

Inherited from `object`

- `__delattr__`, `__format__`, `__getattribute__`, `__new__`, `__reduce__`, `__reduce_ex__`, `__repr__`, `__setattr__`, `__sizeof__`, `__subclasshook__`.

### 26.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 26.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyword_paths</td>
<td>Value: {'addbibresource': 'BIBINPUTS', 'addglobalbib': 'BIBINPUT'...</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>env_variables</td>
<td><strong>Value:</strong> ['INDEXSTYLE', 'BIBINPUTS', 'TEXINPUTS', 'BSTINPUTS']</td>
</tr>
<tr>
<td>two_arg_commands</td>
<td><strong>Value:</strong> ['import', 'subimport', 'includefrom', 'subincludefrom', ...]</td>
</tr>
</tbody>
</table>
27  Module SCons.Scanner.Prog

27.1  Functions

**ProgramScanner(**\*kw\**)***

Return a prototype Scanner instance for scanning executable files for static-lib dependencies

**scan**\((node, env, libpath=())\)**

This scanner scans program files for static-library dependencies. It will search the LIBPATH environment variable for libraries specified in the LIBS variable, returning any files it finds as dependencies.

27.2  Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/Prog.py rel_3.0.0:4395:8972f6a2...'</td>
</tr>
<tr>
<td>print_find_libs</td>
<td>Value: None</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>
28 Module SCons.Scanner.RC

This module implements the dependency scanner for RC (Interface Definition Language) files.

28.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_tlb(nodes)</td>
<td>Filter out .tlb files as they are binary and shouldn’t be scanned</td>
</tr>
</tbody>
</table>

RCScan()

Return a prototype Scanner instance for scanning RC source files

28.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| __revision__ | Value: 'src/engine/SCons/Scanner/RC.py
rel_3.0.0:4395:8972f6a2f6...

| _package__   | Value: 'SCons.Scanner'                                |
29 Module SCons.Scanner.SWIG

SCons.Scanner.SWIG

This module implements the dependency scanner for SWIG code.

29.1 Functions

```
SWIGScanner()
```

29.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/SWIG.py rel_3.0.0:4395:8972f6a2...</td>
</tr>
<tr>
<td>SWIGSuffixes</td>
<td>Value: ['i']</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>
30 Package SCons.Script

SCons.Script

This file implements the main() function used by the scons script.

Architecturally, this is the scons script, and will likely only be called from the external "scons' wrapper. Consequently, anything here should not be, or be considered, part of the build engine. If it's something that we expect other software to want to use, it should go in some other module. If it’s specific to the 'scons' script invocation, it goes here.

30.1 Modules

- **Interactive**: SCons interactive mode
  (Section 31, p. 266)
- **Main**: SCons.Script
  (Section 32, p. 269)
- **SConscript**: SCons.Script.SConscript
  (Section 33, p. 284)

30.2 Functions

<table>
<thead>
<tr>
<th>HelpFunction(text, append=False)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables(files=[], args={})</td>
</tr>
<tr>
<td>Options(files=[], args={})</td>
</tr>
</tbody>
</table>

30.3 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Script/<strong>init</strong>.py rel_3.0.0:4395:8972f...</td>
</tr>
<tr>
<td>start_time</td>
<td>Value: 1505764883.13</td>
</tr>
<tr>
<td>call_stack</td>
<td>Value: []</td>
</tr>
<tr>
<td>PathVariable</td>
<td>Value: &lt;SCons.Variables.PathVariable._PathVariableClass object&gt;</td>
</tr>
</tbody>
</table>

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathOption</td>
<td>Value: <code>&lt;SCons.Options.PathOption._PathOptionClass object&gt;</code></td>
</tr>
<tr>
<td>CScanner</td>
<td>Value: <code>&lt;SCons.Scanner.ClassicCPP object&gt;</code></td>
</tr>
<tr>
<td>DScanner</td>
<td>Value: <code>&lt;SCons.Scanner.D.D object&gt;</code></td>
</tr>
<tr>
<td>DirScanner</td>
<td>Value: <code>&lt;SCons.Scanner.Base object&gt;</code></td>
</tr>
<tr>
<td>ProgramScanner</td>
<td>Value: <code>&lt;SCons.Scanner.Base object&gt;</code></td>
</tr>
<tr>
<td>SourceFileScanner</td>
<td>Value: <code>&lt;SCons.Scanner.Base object&gt;</code></td>
</tr>
<tr>
<td>CScan</td>
<td>Value: <code>&lt;SCons.Scanner.ClassicCPP object&gt;</code></td>
</tr>
<tr>
<td>ARGUMENTS</td>
<td>Value: <code>{}</code></td>
</tr>
<tr>
<td>ARGLIST</td>
<td>Value: <code>[]</code></td>
</tr>
<tr>
<td>COMMAND_LINE_TARGETS</td>
<td>Value: <code>[]</code></td>
</tr>
<tr>
<td>DEFAULT_TARGETS</td>
<td>Value: <code>[]</code></td>
</tr>
<tr>
<td>help_text</td>
<td>Value: None</td>
</tr>
<tr>
<td>sconscript_reading</td>
<td>Value: 0</td>
</tr>
<tr>
<td>GlobalDefaultBuilders</td>
<td>Value: <code>['CFile', 'CXXFile', 'DVI', 'Jar', 'Java', 'JavaH', 'Libr...</code></td>
</tr>
<tr>
<td>SConscript</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</code></td>
</tr>
<tr>
<td>Command</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</code></td>
</tr>
<tr>
<td>AddPostAction</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</code></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AddPreAction</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Alias</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>AlwaysBuild</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>BuildDir</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>CFile</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>CXXFile</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>CacheDir</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Clean</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>DVI</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Decider</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Default</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Depends</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Dir</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>EnsurePythonVersion</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnsureSConsVersion</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Entry</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Execute</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Exit</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Export</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>File</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>FindFile</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>FindInstalledFiles</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>FindSourceFiles</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Flatten</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>GetBuildPath</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>GetLaunchDir</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Glob</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Help</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Import</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Install</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>InstallAs</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>InstallVersionedLib</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Jar</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Java</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>JavaH</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Library</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Literal</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>LoadableModule</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Local</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>M4</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>MSVSProject</td>
<td><strong>Value:</strong> &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoCache</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>NoClean</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Object</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>PCH</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>PDF</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Package</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>ParseDepends</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>PostScript</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Precious</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Program</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>PyPackageDir</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>RES</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>RMIC</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Repository</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>SConscriptChdir</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>SConsignFile</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>SharedLibrary</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>SharedObject</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>SideEffect</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>SourceCode</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>SourceSignatures</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Split</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>StaticLibrary</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>StaticObject</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Tag</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>Tar</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
<tr>
<td>TargetSignatures</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object&gt;</td>
</tr>
</tbody>
</table>

*continued on next page*
### 30.4 Class TargetList

```
object
    _abcoll.Sized

object
    _abcoll.Iterable

object
    _abcoll.Container

    _abcoll.Sequence

    _abcoll.MutableSequence

UserList.UserList

SCons.Script.TargetList
```

#### 30.4.1 Methods

*Inherited from UserList.UserList*

- `__add__()`, `__cmp__()`, `__contains__()`, `__delitem__()`, `__delslice__()`, `__getitem__()`, `__getattribute__()`, `__getslice__()`, `__hash__()`, `__init__()`, `__iter__()`, `__length__()`, `__len__()`, `__nonzero__()`, `__setitem__()`, `__setslice__()`, `__sizeof__()`, `__subclasshook__()`
Inherited from `__abcoll.Sequence`

```python
__iter__(), __reversed__()
```

Inherited from `__abcoll.Sized`

```python
__subclasshook__()
```

Inherited from `object`

```python
__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __setattr__(), __sizeof__(), __str__()
```

### 30.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Inherited from object</code></td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

### 30.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Inherited from UserList.UserList</code></td>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
</tr>
</tbody>
</table>
31 Module SCons.Script.Interactive

SCons interactive mode

31.1 Functions

```
interact(fs, parser, options, targets, target_top)
```

31.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Script/Interactive.py rel_3.0.0:4395:89...</td>
</tr>
<tr>
<td><strong>doc</strong></td>
<td>Value: ...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Script'</td>
</tr>
</tbody>
</table>

31.3 Class SConsInteractiveCmd

```
cmd.Cmd ⊃
    SCons.Script.Interactive.SConsInteractiveCmd
```

build [TARGETS] Build the specified TARGETS and their dependencies. 'b' is a synonym.
clean [TARGETS] Clean (remove) the specified TARGETS and their dependencies. 'c' is a synonym.
exit Exit SCons interactive mode.
help [COMMAND] Prints help for the specified COMMAND. 'h' and '?' are synonyms.
shell [COMMANDLINE] Execute COMMANDLINE in a subshell. 'sh' and '!' are synonyms.
version Prints SCons version information.
31.3.1 Methods

```python
__init__(self, **kw)
```

Instantiate a line-oriented interpreter framework.

The optional argument 'completekey' is the readline name of a completion key; it defaults to the Tab key. If completekey is not None and the readline module is available, command completion is done automatically. The optional arguments stdin and stdout specify alternate input and output file objects; if not specified, sys.stdin and sys.stdout are used. Overrides: cmd.Cmd.__init__(inherited documentation)

```python
default(self, argv)
```

Called on an input line when the command prefix is not recognized.

If this method is not overridden, it prints an error message and returns. Overrides: cmd.Cmd.default extit(inherited documentation)

```python
onecmd(self, line)
```

Interpret the argument as though it had been typed in response to the prompt.

This may be overridden, but should not normally need to be; see the precmd() and postcmd() methods for useful execution hooks. The return value is a flag indicating whether interpretation of commands by the interpreter should stop. Overrides: cmd.Cmd.onecmd extit(inherited documentation)

```python
do_build(self, argv)
```

build [TARGETS] Build the specified TARGETS and their dependencies. 'b' is a synonym.

```python
do_clean(self, argv)
```

clean [TARGETS] Clean (remove) the specified TARGETS and their dependencies. 'c' is a synonym.

```python
do_EOF(self, argv)
```
do\_exit(self, argv)

exit Exit SCons interactive mode.

do\_help(self, argv)

help [COMMAND] Prints help for the specified COMMAND. 'h' and '?' are synonyms. Overrides: cmd.Cmd.do\_help

do\_shell(self, argv)

shell [COMMANDLINE] Execute COMMANDLINE in a subshell. 'sh' and '!' are synonyms.

do\_version(self, argv)

version Prints SCons version information.

**Inherited from cmd.Cmd**

cmdloop(), columnize(), complete(), complete\_help(), completedefault(), complete\_names(), emptyline(), get\_names(), parseline(), postcmd(), postloop(), precmd(), preloop(), print\_topics()

### 31.3.2 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>synonyms</td>
<td>Value: {'b': 'build', 'c': 'clean', 'h': 'help', 'scons': 'build...'}</td>
</tr>
</tbody>
</table>

**Inherited from cmd.Cmd**
doc\_header, doc\_leader, identchars, intro, lastcmd, misc\_header, nohelp, prompt, ruler, undoc\_header, use\_rawinput
Module SCons.Script.Main

SCons.Script

This file implements the main() function used by the scons script.

Architecturally, this is the scons script, and will likely only be called from the external 'scons' wrapper. Consequently, anything here should not be, or be considered, part of the build engine. If it’s something that we expect other software to want to use, it should go in some other module. If it’s specific to the 'scons' script invocation, it goes here.

32.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch_win32_parallel_msg()</td>
</tr>
<tr>
<td>revert_io()</td>
</tr>
<tr>
<td>Progress(*args, **kw)</td>
</tr>
<tr>
<td>GetBuildFailures()</td>
</tr>
<tr>
<td>python_version_string()</td>
</tr>
<tr>
<td>python_version_unsupported(version=sys.version_info(major=2, minor=7, micro=12, releaselevel...))</td>
</tr>
<tr>
<td>python_version_deprecated(version=sys.version_info(major=2, minor=7, micro=12, releaselevel...))</td>
</tr>
<tr>
<td>AddOption(*args, **kw)</td>
</tr>
<tr>
<td>GetOption(name)</td>
</tr>
<tr>
<td>SetOption(name, value)</td>
</tr>
<tr>
<td>PrintHelp(file=None)</td>
</tr>
</tbody>
</table>
find_deepest_user_frame(tb)

Find the deepest stack frame that is not part of SCons.
Input is a "pre-processed" stack trace in the form returned by
traceback.extract_tb() or traceback.extract_stack()

test_load_all_site_scons_dirs(d)

version_string(label, module)

path_string(label, module)

main()

32.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsupported_python_version</td>
<td>Value: (2, 6, 0)</td>
</tr>
<tr>
<td>deprecated_python_version</td>
<td>Value: (2, 7, 0)</td>
</tr>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Script/Main.py rel_3.0.0:4395:8972f6a2f...</td>
</tr>
<tr>
<td>display</td>
<td>Value: &lt;SCons.Util.DisplayEngine object&gt;</td>
</tr>
<tr>
<td>progress_display</td>
<td>Value: &lt;SCons.Util.DisplayEngine object&gt;</td>
</tr>
<tr>
<td>first_command_start</td>
<td>Value: None</td>
</tr>
<tr>
<td>last_command_end</td>
<td>Value: None</td>
</tr>
<tr>
<td>ProgressObject</td>
<td>Value: Null(0x7FDACAF3C1D0)</td>
</tr>
<tr>
<td>print_objects</td>
<td>Value: 0</td>
</tr>
<tr>
<td>print_memoizer</td>
<td>Value: 0</td>
</tr>
<tr>
<td>print_stacktrace</td>
<td>Value: 0</td>
</tr>
<tr>
<td>print_time</td>
<td>Value: 0</td>
</tr>
<tr>
<td>sconscript_time</td>
<td>Value: 0</td>
</tr>
<tr>
<td>cumulative_command_time</td>
<td>Value: 0</td>
</tr>
<tr>
<td>exit_status</td>
<td>Value: 0</td>
</tr>
<tr>
<td>this_build_status</td>
<td>Value: 0</td>
</tr>
<tr>
<td>num_jobs</td>
<td>Value: None</td>
</tr>
<tr>
<td>delayed_warnings</td>
<td>Value: []</td>
</tr>
</tbody>
</table>

continued on next page
### Class SConsPrintHelpException

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptionsParser</td>
<td>Value: &lt;SCons.Script.Main.FakeOptionParser object&gt;</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Script'</td>
</tr>
</tbody>
</table>

#### 32.3 Class SConsPrintHelpException

```
object

exceptions.BaseException

    exceptions.Exception

SCons.Script.Main.SConsPrintHelpException
```

#### 32.3.1 Methods

**Inherited from exceptions.Exception**

- `__init__()`, `__new__()`

**Inherited from exceptions.BaseException**

- `__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

**Inherited from object**

- `__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 32.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td></td>
</tr>
<tr>
<td>args, message</td>
<td></td>
</tr>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
32.4 Class Progressor

object ↳ SCons.Script.Main.Progressor

32.4.1 Methods

```makefile
__init__(self, obj, interval=1, file=None, overwrite=False)

_x.__init__(...) initializes x; see help(type(x)) for signature
Overrides:
object.__init__ extit(inherited documentation)
```

write(self, s)

erase_previous(self)

spinner(self, node)

string(self, node)

replace_string(self, node)

__call__(self, node)

Inherited from object

```makefile
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

32.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Inherited from object
| __class__           |

32.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prev</td>
<td>Value: '',</td>
</tr>
</tbody>
</table>

continued on next page
## 32.5 Class BuildTask

Object Tree

- `object`  
  - `SCons.Taskmaster.Task`  
    - `SCons.Taskmaster.OutOfDateTask`  
      - `SCons.Script.Main.BuildTask`

An SCons build task.

### 32.5.1 Methods

**display**(*self*, *message*)

Hook to allow the calling interface to display a message.

This hook gets called as part of preparing a task for execution (that is, a Node to be built). As part of figuring out what Node should be built next, the actual target list may be altered, along with a message describing the alteration. The calling interface can subclass Task and provide a concrete implementation of this method to see those messages. Overrides: `SCons.Taskmaster.Task.display` (inherited documentation)

**prepare**(*self*)

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to unlink underlying files and make all necessary directories before the Action is actually called to build the targets. Overrides: `SCons.Taskmaster.Task.prepare` (inherited documentation)

**needs__execute**(*self*)

Returns True (indicating this Task should be executed) if this Task’s target state indicates it needs executing, which has already been determined by an earlier up-to-date check. Overrides: `SCons.Taskmaster.Task.needs__execute`
execute(self)

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in prepare(), executed() or failed(). Overrides: SCons.Taskmaster.Task.execute (inherited documentation)

do_failed(self, status=2)

executed(self)

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node's callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node's state before deciding whether it was 'built', in which case we call the appropriate Node method. In any event, we always call 'visited()', which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node. Overrides: SCons.Taskmaster.Task.executed (inherited documentation)

failed(self)

Default action when a task fails: stop the build.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure(). Overrides: SCons.Taskmaster.Task.failed (inherited documentation)

postprocess(self)

Post-processes a task after it's been executed.

This examines all the targets just built (or not, we don't care if the build was successful, or even if there was no build because everything was up-to-date) to see if they have any waiting parent Nodes, or Nodes waiting on a common side effect, that can be put back on the candidates list. Overrides: SCons.Taskmaster.Task.postprocess (inherited documentation)
make_ready(self)

Make a task ready for execution  Overrides:
SCons.Taskmaster.Task.make_ready

Inherited from SCons.Taskmaster.Task(Section 35.4)

__init__(), exc_clear(), exc_info(), exception_set(), executed_with_callbacks(),
excluded_without_callbacks(), fail_continue(), fail_stop(), get_target(), make_ready_all(),
make_ready_current(), trace_message()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

### 32.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 32.5.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>progress</td>
<td>Value: Null(0x7FDACAF3C1D0)</td>
</tr>
</tbody>
</table>

### 32.6 Class CleanTask

An SCons clean task.
32.6.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fs_delete(self, path, pathstr, remove=True)</code></td>
<td>Called to execute the task. This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in <code>prepare()</code>, <code>executed()</code> or <code>failed()</code>.</td>
</tr>
<tr>
<td><code>show(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>remove(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>execute(self)</code></td>
<td>Called to execute the task. This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in <code>prepare()</code>, <code>executed()</code> or <code>failed()</code>. Overrides: <code>SCons.Taskmaster.Task.execute</code> (inherited documentation)</td>
</tr>
<tr>
<td><code>executed(self)</code></td>
<td>Called when the task has been successfully executed and the Taskmaster instance doesn’t want to call the Node’s callback methods. Overrides: <code>SCons.Taskmaster.Task.executed</code></td>
</tr>
<tr>
<td><code>make_ready(self)</code></td>
<td>Marks all targets in a task ready for execution. This is used when the interface needs every target Node to be visited—the canonical example being the &quot;scons -c&quot; option. Overrides: <code>SCons.Taskmaster.Task.make_ready</code></td>
</tr>
</tbody>
</table>
**prepare**(*self*)

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to unlink underlying files and make all necessary directories before the Action is actually called to build the targets. Overrides: SCons.Taskmaster.Task.prepare exit(inherited documentation)

**Inherited from SCons.Taskmaster.AlwaysTask(Section 35.5)**

needs_execute()

**Inherited from SCons.Taskmaster.Task(Section 35.4)**

__init__(), display(), exc_clear(), exc_info(), exception_set(), executed_with_callbacks(), executed_without_callbacks(), fail_continue(), fail_stop(), failed(), get_target(), make_ready_all(), make_ready_current(), postprocess(), trace_message()

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

### 32.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 32.7 Class QuestionTask

object  
SCons.Taskmaster.Task  
SCons.Taskmaster.AlwaysTask  
SCons.Script.Main.QuestionTask

An SCons task for the -q (question) option.
32.7.1 Methods

**prepare**(self)

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to unlink underlying files and make all necessary directories before the Action is actually called to build the targets. Overrides: SCons.Taskmaster.Task.prepare extit(inherited documentation)

**execute**(self)

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in prepare(), executed() or failed(). Overrides: SCons.Taskmaster.Task.execute extit(inherited documentation)

**executed**(self)

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node’s callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node's state before deciding whether it was 'built', in which case we call the appropriate Node method. In any event, we always call 'visited()', which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node. Overrides: SCons.Taskmaster.Task.executed extit(inherited documentation)

Inherited from **SCons.Taskmaster.AlwaysTask** *(Section 35.5)*

needs_execute()

Inherited from **SCons.Taskmaster.Task** *(Section 35.4)*

__init__(), display(), exc_clear(), exc_info(), exception_set(), executed_with_callbacks(), executed_without_callbacks(), fail_continue(), fail_stop(), failed(), get_target(), make_ready(), make_ready_all(), make_ready_current(), postprocess(), trace_message()

Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
32.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

32.8 Class TreePrinter

object └── SCons.Script.Main.TreePrinter

32.8.1 Methods

```python
__init__(self, derived=False, prune=False, status=False)
```

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides:
object.__init__ extit(inherited documentation)

```python
get_all_children(self, node)
```

```python
get_derived_children(self, node)
```

```python
display(self, t)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

32.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
32.9 Class FakeOptionParser

object

\[ SCons.Script.Main.FakeOptionParser \]

A do-nothing option parser, used for the initial OptionsParser variable.

During normal SCons operation, the OptionsParser is created right away by the main() function. Certain tests scripts however, can introspect on different Tool modules, the initialization of which can try to add a new, local option to an otherwise uninitialized OptionsParser object. This allows that introspection to happen without blowing up.

32.9.1 Methods

add_local_option(self, *args, **kw)

Inherited from object

\[ \_delattr\_(), \_format\_(), \_getattribute\_(), \_hash\_(), \_init\_(), \_new\_(), \_reduce\_(), \_reduce\_ex\_(), \_repr\_(), \_setattr\_(), \_sizeof\_(), \_str\_(), \_subclasshook\_() \]

32.9.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><em><strong>class</strong></em></td>
<td></td>
</tr>
</tbody>
</table>

32.9.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>values</td>
<td>Value:</td>
</tr>
<tr>
<td></td>
<td>&lt;SCons.Script.Main.FakeOptionValues object&gt;</td>
</tr>
</tbody>
</table>

32.10 Class Stats

object

\[ SCons.Script.Main.Stats \]

295

32.10.1 Methods

```python
__init__(self)
```
x.__init__(...) initializes x; see help(type(x)) for signature Overridest:
object.__init__ extit(inherited documentation)

```python
enable(self, outfp)
```

```python
do_nothing(self, *args, **kw)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

32.10.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td>Inherited from object</td>
</tr>
</tbody>
</table>

32.11 Class CountStats

```
object
```
```
SCons.Script.Main.Stats
```
```
SCons.Script.Main.CountStats
```

32.11.1 Methods

```python
do__append(self, label)
```

```python
do__print(self)
```

Inherited from SCons.Script.Main.Stats(Section 32.10)
32.11.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

32.12 Class MemStats

object

SCons.Script.Main.Stats

SCons.Script.Main.MemStats

32.12.1 Methods

```python
__init__(self, label)
```

```python
do_print(self)
```

Inherited from SCons.Script.Main.Stats (Section 32.10)

__init__(), do_nothing(), enable()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), 
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), 
__str__(), __subclasshook__()
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
33 Module SCons.Script.SConscript'

SCons.Script.SConscript

This module defines the Python API provided to SConscript and SConstruct files.

33.1 Functions

```python
get_calling_namespaces()
```

Return the locals and globals for the function that called into this module in the current call stack.

```python
compute_exports(exports)
```

Compute a dictionary of exports given one of the parameters to the Export() function or the exports argument to SConscript().

```python
Return(*vars, **kw)
```

```python
SConscript_exception(file=<epydoc.docintrospector._DevNull object>)
```

Print an exception stack trace just for the SConscript file(s). This will show users who have Python errors where the problem is, without cluttering the output with all of the internal calls leading up to where we exec the SConscript.

```python
annotate(node)
```

Annotate a node with the stack frame describing the SConscript file and line number that created it.

```python
Configure(*args, **kw)
```

```python
get__DefaultEnvironmentProxy()
```

BuildDefaultGlobals()

Create a dictionary containing all the default globals for SConstruct and SConscript files.

33.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Script/SConscript.py rel_3.0.0:4395:897...</td>
</tr>
<tr>
<td>launch_dir</td>
<td>Value: '/home/bdbaddog/devel/scons/as_scons'</td>
</tr>
<tr>
<td>GlobalDict</td>
<td>Value: None</td>
</tr>
<tr>
<td>global_exports</td>
<td>Value: {}</td>
</tr>
<tr>
<td>sconscript_chdir</td>
<td>Value: 1</td>
</tr>
<tr>
<td>call_stack</td>
<td>Value: []</td>
</tr>
<tr>
<td>stack_bottom</td>
<td>Value: '% Stack boTTom %'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Script'</td>
</tr>
</tbody>
</table>

33.3 Class SConscriptReturn

```
object

exceptions.BaseException

    exceptions.Exception
        SCons.Script.SConscript'.SConscriptReturn
```

33.3.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()  

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getstate__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()
Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

33.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

33.4 Class Frame

object

SCons.Script.SConscript’.Frame

A frame on the SConstruct/SConscript call stack

33.4.1 Methods

```python
__init__(self, fs, exports, sconscript)
```

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides:
object.__init__ extit(inherited documentation)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

33.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

301
### 33.5 Class SConsEnvironment

An Environment subclass that contains all of the methods that are particular to the wrapper SCons interface and which aren’t (or shouldn’t be) part of the build engine itself.

Note that not all of the methods of this class have corresponding global functions, there are some private methods.

#### 33.5.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong> *(self, <em>targets)</em></td>
<td></td>
</tr>
<tr>
<td><strong>EnsureSConsVersion</strong> <em>(self, major, minor, revision=0)</em></td>
<td>Exit abnormally if the SCons version is not late enough.</td>
</tr>
<tr>
<td><strong>EnsurePythonVersion</strong> <em>(self, major, minor)</em></td>
<td>Exit abnormally if the Python version is not late enough.</td>
</tr>
<tr>
<td><strong>Exit</strong> <em>(self, value=0)</em></td>
<td></td>
</tr>
<tr>
<td><strong>Export</strong> *(self, *vars, *<em>kw)</em></td>
<td></td>
</tr>
<tr>
<td><strong>GetLaunchDir</strong> <em>(self)</em></td>
<td></td>
</tr>
<tr>
<td><strong>GetOption</strong> <em>(self, name)</em></td>
<td></td>
</tr>
</tbody>
</table>
Help(self, text, append=False)

Import(self, *vars)

SConscript(self, *ls, **kw)

SConscriptChdir(self, flag)

SetOption(self, name, value)

Inherited from SCons.Environment.Base(Section 8.9)

Action(), AddPostAction(), AddPreAction(), Alias(), AlwaysBuild(), Append(), AppendENVPath(), AppendUnique(), BuildDir(), Builder(), CacheDir(), Clean(), Clone(), Command(), Copy(), Decider(), Depends(), Detect(), Dictionary(), Dir(), Dump(), Entry(), Environment(), Execute(), File(), FindFile(), FindInstalledFiles(), FindIxes(), FindSourceFiles(), Flatten(), GetBuildPath(), Glob(), Ignore(), Literal(), Local(), NoCache(), NoClean(), ParseConfig(), ParseDepends(), Platform(), Precious(), Prepend(), PrependENVPath(), PrependUnique(), Pseudo(), PyPackageDir(), Replace(), ReplaceIxes(), Repository(), Requires(), SConsignFile(), Scanner(), SetDefault(), SideEffect(), SourceCode(), SourceSignatures(), Split(), TargetSignatures(), Tool(), Value(), VariantDir(),.WhereIs(), __init__(), get_CacheDir(), get_builder(), get_factory(), get_scanner(), get_source_signatures(),
get_tgt_sig_type(), scanner_map_delete()

Inherited from SCons.Environment.SubstitutionEnvironment(Section 8.6)

AddMethod(), MergeFlags(), Override(), ParseFlags(), RemoveMethod(), __contains__(), __delitem__(), __eq__(), __getitem__(), __getitem__(), __init__(), __init__, arg2nodes(), backtick(), get(), gvars(), has_key(), items(), lvars(), subst(), subst_kw(), subst_path(), subst_target_source(),

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

33.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
33.6 Class DefaultEnvironmentCall

A class that implements "global function" calls of Environment methods by fetching the specified method from the DefaultEnvironment’s class. Note that this uses an intermediate proxy class instead of calling the DefaultEnvironment method directly so that the proxy can override the subst() method and thereby prevent expansion of construction variables (since from the user’s point of view this was called as a global function, with no associated construction environment).

33.6.1 Methods

```python
__init__(self, method_name, subst=0)
```

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)

```python
__call__(self, *args, **kw)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

33.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
34 Module SCons.Subst

SCons.Subst
SCons string substitution.

34.1 Functions

**SetAllowableExceptions(** *excepts*)

**raise_exception**(exception, target, s)

**quote_spaces**(arg)

Generic function for putting double quotes around any string that has white space in it.

**escape_list**(mylist, escape_func)

Escape a list of arguments by running the specified escape_func on every object in the list that has an escape() method.

**subst_dict**(target, source)

Create a dictionary for substitution of special construction variables.

This translates the following special arguments:

**target** - the target (object or array of objects), used to generate the TARGET and TARGETS construction variables

**source** - the source (object or array of objects), used to generate the SOURCES and SOURCE construction variables
scons_subst\(\text{strSubst, env, mode}=1, \text{target}=\text{None, source}=\text{None, gvars}=\emptyset, lvars}=\emptyset, conv=\text{None}\)

Expand a string or list containing construction variable substitutions.

This is the work-horse function for substitutions in file names and the like. The companion `scons_subst_list()` function (below) handles separating command lines into lists of arguments, so see that function if that’s what you’re looking for.

scons_subst_list\(\text{strSubst, env, mode}=1, \text{target}=\text{None, source}=\text{None, gvars}=\emptyset, lvars}=\emptyset, conv=\text{None}\)

Substitute construction variables in a string (or list or other object) and separate the arguments into a command list.

The companion `scons_subst()` function (above) handles basic substitutions within strings, so see that function instead if that’s what you’re looking for.

scons__subst__once\(\text{strSubst, env, key}\)

Perform single (non-recursive) substitution of a single construction variable keyword.

This is used when setting a variable when copying or overriding values in an Environment. We want to capture (expand) the old value before we override it, so people can do things like:

\[
\text{env2 = env.Clone(CCFLAGS = '}\$\text{CCFLAGS -g}')
\]

We do this with some straightforward, brute-force code here...

### 34.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: ’src/engine/SCons/Subst.py rel_3.0.0:4395:8972f6a2f699 20...</td>
</tr>
<tr>
<td>AllowableExceptions</td>
<td>Value: (\text{&lt;type 'exceptions.IndexError'&gt;, &lt;type 'exceptions.NameEr... }</td>
</tr>
<tr>
<td>NullNodesList</td>
<td>Value: Null(0x7FDACB854410)</td>
</tr>
</tbody>
</table>
### 34.3 Class Literal

object →

SCons.Subst.Literal

A wrapper for a string. If you use this object wrapped around a string, then it will be interpreted as literal. When passed to the command interpreter, all special characters will be escaped.

#### 34.3.1 Methods

```python
__init__(self, lstr)
```

x.__init__(...) initializes x; see help(type(x)) for signature. Overrides: object.__init__ (inherited documentation)

```python
__str__(self)
```

str(x) Overrides: object.__str__ (inherited documentation)

```python
escape(self, escape_func)
```

```python
for_signature(self)
```

```python
is_literal(self)
```

```python
__eq__(self, other)
```

```python
__neq__(self, other)
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
```
### 34.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><strong><strong>class</strong></strong></td>
<td></td>
</tr>
</tbody>
</table>

### 34.4 Class SpecialAttrWrapper

```python
object └── SCons.Subst.SpecialAttrWrapper
```

This is a wrapper for what we call a 'Node special attribute.' This is any of the attributes of a Node that we can reference from Environment variable substitution, such as `$TARGET.abspath` or `$SOURCES[1].filebase`. We implement the same methods as Literal so we can handle special characters, plus a `for_signature` method, such that we can return some canonical string during signature calculation to avoid unnecessary rebuilds.

#### 34.4.1 Methods

```python
__init__(self, lstr, for_signature=None)
```

The `for_signature` parameter, if supplied, will be the canonical string we return from `for_signature()`. Else we will simply return `lstr`. Overrides: `object.__init__`

```python
__str__(self)
```

`str(x)` Overrides: `object.__str__` extit(inherited documentation)

```python
escape(self, escape_func)
```

```python
for_signature(self)
```

```python
is_literal(self)
```

*Inherited from object*

```python
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()
```
34.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td>__<strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

34.5 Class CmdStringHolder

object

          __abcoll.Sized

object

          __abcoll.Iterable

object

          __abcoll.Container

          __abcoll.Sequence

UserString.UserString

SCons.Subst.CmdStringHolder

This is a special class used to hold strings generated by scons_subst() and scons_subst_list(). It defines a special method escape(). When passed a function with an escape algorithm for a particular platform, it will return the contained string with the proper escape sequences inserted.

34.5.1 Methods

```python
__init__(self, cmd, literal=None)
```

x.__init__(...) initializes x; see help(type(x)) for signature Overrides: object.__init__ extit(inherited documentation)

```python
is__literal(self)
```


**escape**(*self*, *escape_func*, *quote_func=<__builtin___.function object>*)

Escape the string with the supplied function. The function is expected to take an arbitrary string, then return it with all special characters escaped and ready for passing to the command interpreter.

After calling this function, the next call to **str()** will return the escaped string.

**Inherited from UserString.UserString**

__add__(), __cmp__(), __complex__(), __contains__(), __float__(), __getattr__(),
__getitem__(), __getslice__(), __hash__(), __int__(), __len__(), __long__(), __mod__(),
__mul__(), __radd__(), __repr__(), __rmul__(), __str__(), capitalize(),
center(), count(), decode(), encode(), endswith(), expandtabs(), find(), index(),
isalnum(), isalpha(), isdecimal(), isdigit(), islower(), isnumeric(), isspace(), istitle(),
isupper(), join(), ljust(), lower(), lstrip(), partition(), replace(), rfind(),
rimax(), rjust(), rpartition(), rsplit(), rstrip(), split(), splitlines(), startswith(),
strip(), swapcase(), title(), translate(), upper(), zfill()

**Inherited from _abcoll.Sequence**

__iter__(), __reversed__()

**Inherited from _abcoll.Sized**

__subclasshook__()

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __setattr__(), __sizeof__()

34.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

34.5.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from UserString.UserString</td>
<td><strong>abstractmethods</strong></td>
</tr>
</tbody>
</table>
34.6 Class NLWrapper

A wrapper class that delays turning a list of sources or targets into a NodeList until it’s needed. The specified function supplied when the object is initialized is responsible for turning raw nodes into proxies that implement the special attributes like .abspath, .source, etc. This way, we avoid creating those proxies just "in case" someone is going to use $TARGET or the like, and only go through the trouble if we really have to.

In practice, this might be a wash performance-wise, but it’s a little cleaner conceptually...

34.6.1 Methods

```python
__init__(self, list, func)
```

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ extit(inherited documentation)

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

34.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
34.7 Class Targets_or_Sources

A class that implements $TARGETS or $SOURCES expansions by in turn wrapping a NLWrapper. This class handles the different methods used to access the list, calling the NLWrapper to create proxies on demand.

Note that we subclass collections.UserList purely so that the is_Sequence() function will identify an object of this class as a list during variable expansion. We’re not really using any collections.UserList methods in practice.

34.7.1 Methods

```python
__init__(self, nl)
```

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ (inherited documentation)

```python
__getattr__(self, attr)
```

```python
__getitem__(self, i)
```

Overrides: __abcoll.Sequence.__getitem__
__getslice__**(self, i, j)**

Overrides: UserList.UserList.__getslice__

__str__**(self)**

str(x)  Overrides: object.__str__ (inherited documentation)

__repr__**(self)**

repr(x)  Overrides: object.__repr__ (inherited documentation)

*Inherited from UserList.UserList*

__add__(), __cmp__(), __contains__(), __delitem__(), __delslice__(),
__eq__(), __ge__(), __gt__(), __iadd__(), __imul__(), __le__(), __len__(),
__lt__(), __mul__(), __ne__(), __radd__(), __rmul__(), __setitem__(),
__setslice__(), append(), count(), extend(), index(), insert(), pop(), remove(),
reverse(), sort()

*Inherited from _abcoll.Sequence*

__iter__(), __reversed__()

*Inherited from _abcoll.Sized*

__subclasshook__()  *

*Inherited from object*

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __setattr__(), __sizeof__()

34.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inherited from object</strong></td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

34.7.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inherited from UserList.UserList</strong></td>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
</tr>
</tbody>
</table>
34.8 Class Target_or_Source

A class that implements $TARGET or $SOURCE expansions by in turn wrapping a NL-Wrapper. This class handles the different methods used to access an individual proxy Node, calling the NLWrapper to create a proxy on demand.

34.8.1 Methods

```python
__init__(self, nl)
```

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ extit(inherited documentation)

```python
__getattr__(self, attr)
```

```python
__str__(self)
```

str(x)  Overrides: object.__str__ extit(inherited documentation)

```python
__repr__(self)
```

repr(x)  Overrides: object.__repr__ extit(inherited documentation)

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()
```

34.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
34.9 Class NullNodeList

object

SCons.Util.Null

SCons.Util.NullSeq

SCons.Subst_NullNodeList

34.9.1 Methods

```python
__call__(self, *args, **kwargs)
```
Overrides: SCons.Util.Null.__call__

```python
str(self)
```

\[\text{str}(x)\] Overrides: object.__str__ (inherited documentation)

Inherited from SCons.Util.NullSeq (Section 36.17)

```python
__delitem__(), __getitem__(), __iter__(), __len__(), __setitem__()
```

Inherited from SCons.Util.Null (Section 36.16)

```python
__bool__(), __delattr__(), __getattr__(), __init__(), __new__(), __nonzero__(),
__repr__(), __setattr__()
```

Inherited from object

```python
__format__(), __getattribute__(), __hash__(), __reduce__(), __reduce_ex__(),
__sizeof__(), __subclasshook__()
```

34.9.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
Module SCons.Taskmaster

This module contains the primary interface(s) between a wrapping user interface and the SCons build engine. There are two key classes here:

Taskmaster

This is the main engine for walking the dependency graph and calling things to decide what does or doesn’t need to be built.

Task

This is the base class for allowing a wrapping interface to decide what does or doesn’t actually need to be done. The intention is for a wrapping interface to subclass this as appropriate for different types of behavior it may need.

The canonical example is the SCons native Python interface, which has Task subclasses that handle its specific behavior, like printing "foo is up to date" when a top-level target doesn’t need to be built, and handling the -c option by removing targets as its "build" action. There is also a separate subclass for suppressing this output when the -q option is used.

The Taskmaster instantiates a Task object for each (set of) target(s) that it decides need to be evaluated and/or built.

35.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump_stats()</td>
<td></td>
</tr>
<tr>
<td>find_cycle(stack, visited)</td>
<td></td>
</tr>
</tbody>
</table>

35.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>doc</strong></td>
<td>Value: ...</td>
</tr>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Taskmaster.py rel_3.0.0:4395:8972f6a2f6...'</td>
</tr>
<tr>
<td>StateString</td>
<td>Value: {0: 'no_state', 1: 'pending', 2: 'executing', 3: 'up_to_d...</td>
</tr>
<tr>
<td>NODE_NO_STATE</td>
<td>Value: 0</td>
</tr>
<tr>
<td>NODE_PENDING</td>
<td>Value: 1</td>
</tr>
</tbody>
</table>

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODE_EXECUTING</td>
<td>Value: 2</td>
</tr>
<tr>
<td>NODE_UP_TO_DATE</td>
<td>Value: 3</td>
</tr>
<tr>
<td>NODE_EXECUTED</td>
<td>Value: 4</td>
</tr>
<tr>
<td>NODE_FAILED</td>
<td>Value: 5</td>
</tr>
<tr>
<td>print_prepare</td>
<td>Value: 0</td>
</tr>
<tr>
<td>CollectStats</td>
<td>Value: None</td>
</tr>
<tr>
<td>StatsNodes</td>
<td>Value: []</td>
</tr>
</tbody>
</table>
| fmt                   | Value: 
%(considered)3d
%(already_handled)3d
%(problem)3d
%(chil... |
| __package__           | Value: 'SCons'                       |

### 35.3 Class Stats

A simple class for holding statistics about the disposition of a Node by the Taskmaster. If we’re collecting statistics, each Node processed by the Taskmaster gets one of these attached, in which case the Taskmaster records its decision each time it processes the Node. (Ideally, that’s just once per Node.)

#### 35.3.1 Methods

```python
__init__(self)
```

Instantiates a Taskmaster.Stats object, initializing all appropriate counters to zero. Overrides: object.__init__

**Inherited from object**

- __delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
- __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
- __str__(), __subclasshook__()

#### 35.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
</tbody>
</table>

*continued on next page*
35.4 Class Task

object

SCons.Taskmaster.Task

**Known Subclasses:** SCons.Taskmaster.AlwaysTask, SCons.Taskmaster.OutOfDateTask

Default SCons build engine task.

This controls the interaction of the actual building of node and the rest of the engine.

This is expected to handle all of the normally-customizable aspects of controlling a build, so any given application *should* be able to do what it wants by sub-classing this class and overriding methods as appropriate. If an application needs to customize something by sub-classing Taskmaster (or some other build engine class), we should first try to migrate that functionality into this class.

Note that it’s generally a good idea for sub-classes to call these methods explicitly to update state, etc., rather than roll their own interaction with Taskmaster from scratch.

35.4.1 Methods

`__init__(self, tm, targets, top, node)`

overrides:
object.__init__ (inherited documentation)

`trace_message(self, method, node, description=None)`

`display(self, message)`

Hook to allow the calling interface to display a message.

This hook gets called as part of preparing a task for execution (that is, a Node to be built). As part of figuring out what Node should be built next, the actual target list may be altered, along with a message describing the alteration. The calling interface can subclass Task and provide a concrete implementation of this method to see those messages.
### prepare(self)

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to unlink underlying files and make all necessary directories before the Action is actually called to build the targets.

### get_target(self)

Fetch the target being built or updated by this task.

### needs_execute(self)

### execute(self)

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in prepare(), executed() or failed().

### executed_without_callbacks(self)

Called when the task has been successfully executed and the Taskmaster instance doesn’t want to call the Node’s callback methods.

### executed_with_callbacks(self)

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node’s callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node’s state before deciding whether it was "built", in which case we call the appropriate Node method. In any event, we always call ‘visited()’, which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node.
executed(self)

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node's callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node's state before deciding whether it was "built", in which case we call the appropriate Node method. In any event, we always call 'visited()', which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node.

failed(self)

Default action when a task fails: stop the build.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().

fail_stop(self)

Explicit stop-the-build failure.

This sets failure status on the target nodes and all of their dependent parent nodes.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().

fail_continue(self)

Explicit continue-the-build failure.

This sets failure status on the target nodes and all of their dependent parent nodes.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().
**make_ready_all**(self)

Marks all targets in a task ready for execution.

This is used when the interface needs every target Node to be visited—the canonical example being the "scons -c" option.

**make_ready_current**(self)

Marks all targets in a task ready for execution if any target is not current.

This is the default behavior for building only what’s necessary.

**make_ready**(self)

Marks all targets in a task ready for execution if any target is not current.

This is the default behavior for building only what’s necessary.

**postprocess**(self)

Post-processes a task after it’s been executed.

This examines all the targets just built (or not, we don’t care if the build was successful, or even if there was no build because everything was up-to-date) to see if they have any waiting parent Nodes, or Nodes waiting on a common side effect, that can be put back on the candidates list.

**exc_info**(self)

Returns info about a recorded exception.
Class AlwaysTask

Module SCons.Taskmaster

```
exc_clear(self)
```

Clears any recorded exception.
This also changes the 'exception_raise' attribute to point to the appropriate
do-nothing method.

```
exception_set(self, exception=None)
```

Records an exception to be raised at the appropriate time.
This also changes the 'exception_raise' attribute to point to the method that
will, in fact

*Inherited from object*

```__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()```

35.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

35.5 Class AlwaysTask

object ⊃

SCons.Taskmaster.Task ⊃

SCons.Taskmaster.AlwaysTask

35.5.1 Methods

```python
needs_execute(self)
```

Always returns True (indicating this Task should always be executed).

Subclasses that need this behavior (as opposed to the default of only executing Nodes that are out of date w.r.t. their dependencies) can use this as follows:

```python
class MyTaskSubclass(SCons.Taskmaster.Task):
    needs_execute = SCons.Taskmaster.Task.execute_always

Overrides: SCons.Taskmaster.Task.needs_execute
```

*Inherited from SCons.Taskmaster.Task (Section 35.4)*

```python
__init__(), display(), exc_clear(), exc_info(), exception_set(), execute(), executed(), executed_with_callbacks(), executed_without_callbacks(), fail_continue(), fail_stop(), failed(), get_target(), make_ready(), make_ready_all(), make_ready_current(), postprocess(), prepare(), trace_message()
```

*Inherited from object*

```python
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

35.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

35.6 Class OutOfDateTask

```python
object
```

```python
SCons.Taskmaster.Task
```

```python
SCons.Taskmaster.OutOfDateTask
```

Known Subclasses: SCons.Script.Main.BuildTask
35.6.1 Methods

\begin{verbatim}
needs__execute(self)
\end{verbatim}

Returns True (indicating this Task should be executed) if this Task’s target state indicates it needs executing, which has already been determined by an earlier up-to-date check. Overrides: SCons.Taskmaster.Task.needs__execute

\textit{Inherited from SCons.Taskmaster.Task (Section 35.4)}

\begin{verbatim}
__init__(), display(), exc_clear(), exc_info(), exception_set(), execute(), executed(), executed_with_callbacks(), executed_without_callbacks(), fail_continue(), fail_stop(), failed(), get_target(), make_ready(), make_ready_all(), make_ready_current(), postprocess(), prepare(), trace_message()
\end{verbatim}

\textit{Inherited from object}

\begin{verbatim}
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),__str__(), __subclasshook__()
\end{verbatim}

35.6.2 Properties

\begin{table}[h]
\begin{tabular}{|c|c|}
\hline
\textbf{Name} & \textbf{Description} \\
\hline
\textit{Inherited from object} & \\
\hline
\textit{__class__} & \\
\hline
\end{tabular}
\end{table}

35.7 Class Taskmaster

\begin{verbatim}
object
\end{verbatim}

\begin{verbatim}
SCons.Taskmaster.Taskmaster
\end{verbatim}

The Taskmaster for walking the dependency DAG.

35.7.1 Methods

\begin{verbatim}
__init__ (self, targets=[], tasker=None, order=None, trace=None)
\end{verbatim}

\begin{verbatim}
x.__init__(...) initializes x; see help(type(x)) for signature Overrides: object.__init__ extit(inherited documentation)
\end{verbatim}
**Class Taskmaster**

**Module SCons.Taskmaster**

```python
find_next_candidate(self)
```

Returns the next candidate Node for (potential) evaluation.

The candidate list (really a stack) initially consists of all of the top-level (command line) targets provided when the Taskmaster was initialized. While we walk the DAG, visiting Nodes, all the children that haven’t finished processing get pushed on to the candidate list. Each child can then be popped and examined in turn for whether their children are all up-to-date, in which case a Task will be created for their actual evaluation and potential building.

Here is where we also allow candidate Nodes to alter the list of Nodes that should be examined. This is used, for example, when invoking SCons in a source directory. A source directory Node can return its corresponding build directory Node, essentially saying, 'Hey, you really need to build this thing over here instead.'

```python
no_next_candidate(self)
```

Stops Taskmaster processing by not returning a next candidate.

Note that we have to clean-up the Taskmaster candidate list because the cycle detection depends on the fact all nodes have been processed somehow.

```python
trace_message(self, message)
```

```python
trace_node(self, node)
```

```python
next_task(self)
```

Returns the next task to be executed.

This simply asks for the next Node to be evaluated, and then wraps it in the specific Task subclass with which we were initialized.

```python
will_not_build(self, nodes, node_func=<_builtins<function object>>)
```

Perform clean-up about nodes that will never be built. Invokes a user defined function on all of these nodes (including all of their parents).
**stop**(self)

Stops the current build completely.

**cleanup**(self)

Check for dependency cycles.

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

---

### 35.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
Module SCons.Util

Various utility functions go here.

### 36.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dictify(keys, values, result={})</code></td>
<td></td>
</tr>
<tr>
<td><code>rightmost_separator(path, sep)</code></td>
<td></td>
</tr>
<tr>
<td><code>containsAny(str, set)</code></td>
<td>Check whether sequence str contains ANY of the items in set.</td>
</tr>
<tr>
<td><code>containsAll(str, set)</code></td>
<td>Check whether sequence str contains ALL of the items in set.</td>
</tr>
<tr>
<td><code>containsOnly(str, set)</code></td>
<td>Check whether sequence str contains ONLY items in set.</td>
</tr>
<tr>
<td><code>splitext(path)</code></td>
<td>Same as os.path.splitext() but faster.</td>
</tr>
<tr>
<td><code>updrive(path)</code></td>
<td>Make the drive letter (if any) upper case. This is useful because Windows is</td>
</tr>
<tr>
<td></td>
<td>inconsistent on the case of the drive letter, which can cause inconsistencies</td>
</tr>
<tr>
<td></td>
<td>when calculating command signatures.</td>
</tr>
</tbody>
</table>
**get_environment_var(varstr)**

Given a string, first determine if it looks like a reference to a single environment variable, like "$FOO" or "${FOO}". If so, return that variable with no decorations ("FOO"). If not, return None.

**render_tree(root, child_func, prune=0, margin=[0], visited=None)**

Render a tree of nodes into an ASCII tree view. **Parameters**

- root: the root node of the tree
- child_func: the function called to get the children of a node
- prune: don’t visit the same node twice
- margin: the format of the left margin to use for children of root. 1 results in a pipe, and 0 results in no pipe.
- visited: a dictionary of visited nodes in the current branch if not prune, or in the whole tree if prune.

**print_tree(root, child_func, prune=0, showtags=0, margin=[0], visited=None)**

Print a tree of nodes. This is like render_tree, except it prints lines directly instead of creating a string representation in memory, so that huge trees can be printed. **Parameters**

- root: the root node of the tree
- child_func: the function called to get the children of a node
- prune: don’t visit the same node twice
- showtags: print status information to the left of each node line
- margin: the format of the left margin to use for children of root. 1 results in a pipe, and 0 results in no pipe.
- visited: a dictionary of visited nodes in the current branch if not prune, or in the whole tree if prune.

**is_Dict(obj, isinstance=<built-in function is_instance>, DictTypes=dict, UserDict)***
is_List(obj, isinstance=<built-in function isinstance>, ListTypes=(<type 'list'>, <class 'UserList.UserList'>))

is_Sequence(obj, isinstance=<built-in function isinstance>, SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>))

is_Tuple(obj, isinstance=<built-in function isinstance>, tuple=<type 'tuple'>)

is_String(obj, isinstance=<built-in function isinstance>, StringType=(<type 'str'>, <type 'unicode'>, <class 'UserString.UserString'>, <class 'UserList.UserList'>))

is_Scalar(obj, isinstance=<built-in function isinstance>, StringType=(<type 'str'>, <type 'unicode'>, <class 'UserString.UserString'>, <class 'UserList.UserList'>), SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>))

do_flatten(sequence, result, isinstance=<built-in function isinstance>, StringTypes=(<type 'str'>, <type 'unicode'>, <class 'UserString.UserString'>, <class 'UserList.UserList'>), SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>))

flatten(obj, isinstance=<built-in function isinstance>, StringType=(<type 'str'>, <type 'unicode'>, <class 'UserString.UserString'>, <class 'UserList.UserList'>), SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>), do_flatten=<_builtins__function object>)

Flatten a sequence to a non-nested list.

Flatten() converts either a single scalar or a nested sequence to a non-nested list. Note that flatten() considers strings to be scalars instead of sequences like Python would.
**flatten_sequence**

```python
flatten_sequence(sequence, isinstance=<built-in function isinstance>, StringType=(<type 'str'>, <type 'unicode'>), <class 'UserString.UserString'>, SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>), do_flatten=<__builtin__.function object>)
```

Flatten a sequence to a non-nested list.

Same as flatten(), but it does not handle the single scalar case. This is slightly more efficient when one knows that the sequence to flatten can not be a scalar.

**to_String**

```python
to_String(s, isinstance=<built-in function isinstance>, str=<type 'str'>, <class 'UserString.UserString'>, BaseStringTypes=(<type 'str'>, <type 'unicode'>))
```

**to_String_for_substr**

```python
to_String_for_substr(s, isinstance=<built-in function isinstance>, str=<type 'str'>, to_String=<__builtin__.function object>, BaseStringTypes=(<type 'str'>, <type 'unicode'>), SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>), UserString=<class 'UserString.UserString'>)
```

**to_String_for_signature**

```python
to_String_for_signature(obj, to_String_for_substr=<__builtin__.function object>, AttributeError=<type 'exceptions.AttributeError'>)
```

**semi_deepcopy_dict**

```python
semi_deepcopy_dict(x, exclude=[])  
semi_deepcopy(x)
```

**RegGetValue**

```python
RegGetValue(root, key)
```

**RegOpenKeyEx**

```python
RegOpenKeyEx(root, key)
```

**WhereIs**

```python
WhereIs(file, path=None, pathext=None, reject=[])  
330```
### PrependPath

```python
PrependPath(olddir, newdir, sep=':', delete_existing=1,
            canonicalize=None)
```

This prepends newpath elements to the given oldpath. Will only add any particular path once (leaving the first one it encounters and ignoring the rest, to preserve path order), and will os.path.normpath and os.path.normcase all paths to help assure this. This can also handle the case where the given old path variable is a list instead of a string, in which case a list will be returned instead of a string.

**Example:** Old Path: "/foo/bar:/foo" New Path: "/biz/boom:/foo" Result: "/biz/boom:/foo:/foo/bar"

If `delete_existing` is 0, then adding a path that exists will not move it to the beginning; it will stay where it is in the list.

If `canonicalize` is not `None`, it is applied to each element of `newpath` before use.

### AppendPath

```python
AppendPath(olddir, newdir, sep=':', delete_existing=1,
           canonicalize=None)
```

This appends new path elements to the given old path. Will only add any particular path once (leaving the last one it encounters and ignoring the rest, to preserve path order), and will os.path.normpath and os.path.normcase all paths to help assure this. This can also handle the case where the given old path variable is a list instead of a string, in which case a list will be returned instead of a string.

**Example:** Old Path: "/foo/bar:/foo" New Path: "/biz/boom:/foo" Result: "/foo/bar:/biz/boom:/foo"

If `delete_existing` is 0, then adding a path that exists will not move it to the end; it will stay where it is in the list.

If `canonicalize` is not `None`, it is applied to each element of `newpath` before use.
### AddPathIfNotExists(env_dict, key, path, sep=':')

This function will take 'key' out of the dictionary 'env_dict', then add the path 'path' to that key if it is not already there. This treats the value of env_dict[key] as if it has a similar format to the PATH variable...a list of paths separated by tokens. The 'path' will get added to the list if it is not already there.

### get_native_path(path)

Transforms an absolute path into a native path for the system. Non-Cygwin version, just leave the path alone.

### Split(arg)

### case_sensitive_suffixes(s1, s2)

### adjustixes(fname, pre, suf, ensure_suffix=False)

### unique(s)

Return a list of the elements in s, but without duplicates.

For example, unique([1,2,3,1,2,3]) is some permutation of [1,2,3], unique('abcabc') some permutation of ['a', 'b', 'c'], and unique(([1, 2], [2, 3], [1, 2])) some permutation of [[2, 3], [1, 2]].

For best speed, all sequence elements should be hashable. Then unique() will usually work in linear time.

If not possible, the sequence elements should enjoy a total ordering, and if list(s).sort() doesn’t raise TypeError it’s assumed that they do enjoy a total ordering. Then unique() will usually work in O(N*log2(N)) time.

If that’s not possible either, the sequence elements must support equality-testing. Then unique() will usually work in quadratic time.

### uniquer(seq, idfun=None)
###uner_hashables(seq)

The `uner_hashables` function takes a sequence `seq` and returns a unique list of hashable elements from `seq`.

###logical_lines(physical_lines, joiner=<built-in method join of str object at 0x7fdacdd95508>)

The `logical_lines` function takes a list of physical lines `physical_lines` and a joiner `joiner` and returns a list of logical lines.

###make_path_relative(path)

The `make_path_relative` function takes a path `path` and returns a relative path.

This function makes an absolute path name to a relative pathname.

###AddMethod(obj, function, name=None)

The `AddMethod` function adds either a bound method to an instance or the function itself (or an unbound method in Python 2) to a class. If `name` is omitted, the name of the specified function is used by default.

**Example:**

```python
a = A()
def f(self, x, y):
    self.z = x + y
AddMethod(f, A, "add")
a.add(2, 4)
print(a.z)
```

###RenameFunction(function, name)

The `RenameFunction` function returns a function identical to the specified function, but with the specified name.

###MD5signature(s)

###MD5filesignature(fname, chunksize=65536)

These are MD5 signature functions.
**MD5collect**(*signatures*)

Collects a list of signatures into an aggregate signature.

*signatures* - a list of signatures  
*returns* - the aggregate signature

**silent__intern**(*x*)

Perform `sys.intern()` on the passed argument and return the result. If the input is ineligible (e.g. a unicode string) the original argument is returned and no exception is thrown.

**to__bytes**(*s*)

**to__str**(*s*)

### 36.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PY3</td>
<td>Value: False</td>
</tr>
<tr>
<td>DictTypes</td>
<td>Value: dict, UserDict</td>
</tr>
<tr>
<td>ListTypes</td>
<td>Value: (&lt;type 'list'&gt;, &lt;class 'UserList.UserList'&gt;)</td>
</tr>
<tr>
<td>SequenceTypes</td>
<td>Value: (&lt;type 'list'&gt;, &lt;type 'tuple'&gt;, &lt;class 'UserList.UserList'&gt;)</td>
</tr>
<tr>
<td>StringTypes</td>
<td>Value: (&lt;type 'str'&gt;, &lt;type 'unicode'&gt;, &lt;class 'UserString.UserString'&gt;)</td>
</tr>
<tr>
<td>BaseStringTypes</td>
<td>Value: (&lt;type 'str'&gt;, &lt;type 'unicode'&gt;)</td>
</tr>
<tr>
<td>d</td>
<td>Value: {&lt;type 'tuple'&gt;: &lt;<strong>builtin</strong>.__function object&gt;, &lt;type 'di...</td>
</tr>
<tr>
<td>can_read_reg</td>
<td>Value: 0</td>
</tr>
<tr>
<td>hkey_mod</td>
<td>Value: win32con</td>
</tr>
<tr>
<td>RegEnumKey</td>
<td>Value: win32api.RegEnumKey</td>
</tr>
<tr>
<td>RegEnumValue</td>
<td>Value: win32api.RegEnumValue</td>
</tr>
<tr>
<td>RegQueryValueEx</td>
<td>Value: win32api.RegQueryValueEx</td>
</tr>
<tr>
<td>HKEY_CLASSES_ROOT</td>
<td>Value: None</td>
</tr>
<tr>
<td>HKEY_LOCAL_MACHINE</td>
<td>Value: None</td>
</tr>
</tbody>
</table>

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKEY_CURRENT_USER</td>
<td>Value: None</td>
</tr>
<tr>
<td>HKEY_USERS</td>
<td>Value: None</td>
</tr>
<tr>
<td>display</td>
<td>Value: &lt;SCons.Util.DisplayEngine object&gt;</td>
</tr>
<tr>
<td>md5</td>
<td>Value: True</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

### 36.3 Class NodeList

```plaintext
object
  _abcoll.Sized
object
  _abcoll.Iterable
object
  _abcoll.Container
    _abcoll.Sequence
    _abcoll.MutableSequence
UserList.UserList
  SCons.Util.NodeList
```

This class is almost exactly like a regular list of Nodes (actually it can hold any object), with one important difference. If you try to get an attribute from this list, it will return that attribute from every item in the list. For example:

```python
>>> someList = NodeList(['foo', 'bar'])
>>> someList.strip()
['foo', 'bar']
```

### 36.3.1 Methods

```python
___nonzero___(self)
```
__bool__(self)

__str__(self)

str(x) Overrides: object.__str__ (inherited documentation)

__iter__(self)

Overrider: _abcoll.Iterable.__iter__

__call__(self, *args, **kwargs)

__getattr__(self, name)

__getitem__(self, index)

This comes for free on py2, but py3 slices of NodeList are returning a list breaking slicing nodelist and refering to properties and methods on contained object.Overrides: _abcoll.Sequence.__getitem__

Inherited from UserList.UserList

__add__(), __cmp__(), __contains__(), __delitem__(), __delslice__(),
__eq__(), __ge__(), __getslice__(), __gt__(), __iadd__(), __imul__(),
__init__(), __le__(), __len__(), __lt__(), __mul__(), __ne__(), __radd__(),
__repr__(), __rmul__(), __setitem__(), __setslice__(), append(), count(),
extend(), index(), insert(), pop(), remove(), reverse(), sort()

Inherited from _abcoll.Sequence

__reversed__()

Inherited from _abcoll.Sized

__subclasshook__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __setattr__(), __sizeof__()

36.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
</tbody>
</table>

continued on next page
36.3.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>abstractmethods</strong></td>
<td></td>
</tr>
<tr>
<td><strong>hash</strong></td>
<td></td>
</tr>
</tbody>
</table>

36.4 Class DisplayEngine

```
object → SCons.Util.DisplayEngine
```

36.4.1 Methods

```
__call__(self, text, append_newline=1)
```

```
set_mode(self, mode)
```

Inherited from object

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(),
__new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(),
__sizeof__(), __str__(), __subclasshook__()
```

36.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

36.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>print_it</td>
<td>Value: True</td>
</tr>
</tbody>
</table>
### 36.5 Class Proxy

A simple generic Proxy class, forwarding all calls to subject. So, for the benefit of the python newbie, what does this really mean? Well, it means that you can take an object, let’s call it 'objA’, and wrap it in this Proxy class, with a statement like this

```python
proxyObj = Proxy(objA),
```

Then, if in the future, you do something like this

```python
x = proxyObj.var1,
```

since Proxy does not have a 'var1' attribute (but presumably objA does), the request actually is equivalent to saying

```python
x = objA.var1
```

Inherit from this class to create a Proxy.

Note that, with new-style classes, this does not work transparently for Proxy subclasses that use special .__*__() method names, because those names are now bound to the class, not the individual instances. You now need to know in advance which .__*__() method names you want to pass on to the underlying Proxy object, and specifically delegate their calls like this:

```python
class Foo(Proxy):  __str__ = Delegate('__str__')
```

### 36.5.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__init__()</code></td>
<td>Wrap an object as a Proxy object. Overrides: <code>object.__init__</code></td>
</tr>
<tr>
<td><code>__getattr__()</code></td>
<td>Retrieve an attribute from the wrapped object. If the named attribute doesn’t exist, <code>AttributeError</code> is raised</td>
</tr>
</tbody>
</table>
Class Delegate

Module SCons.Util

```
get(self)

Retrieve the entire wrapped object
```

```
__eq__(self, other)
```

**Inherited from object**

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

36.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

36.6 Class Delegate

```
object

SCons.Util.Delegate
```

A Python Descriptor class that delegates attribute fetches to an underlying wrapped subject of a Proxy. Typical use:

```
class Foo(Proxy): __str__ = Delegate('__str__')
```

36.6.1 Methods

```
__init__(self, attribute)

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)
```

```
__get__(self, obj, cls)
```

**Inherited from object**

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
```

339
36.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

36.7 Class _NoError

object

exceptions.BaseException

exceptions.Exception

SCons.Util._NoError

36.7.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()  

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

36.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td></td>
</tr>
<tr>
<td>args, message</td>
<td></td>
</tr>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
36.8 Class PlainWindowsError

Inherited from exceptions.OSError

__init__(), __new__()

Inherited from exceptions.EnvironmentError

__reduce__(), __str__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

36.8.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.EnvironmentError</td>
<td>errno, filename, strerror</td>
</tr>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
Class PlainWindowsError

object

exceptions.BaseException

exceptions.Exception

exceptions.StandardError

exceptions.EnvironmentError

exceptions.OSError

SCons.Util.PlainWindowsError

36.9 Class PlainWindowsError

Inherited from exceptions.OSError

__init__(), __new__()

Inherited from exceptions.EnvironmentError

__reduce__(), __str__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __repr__(),
__setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

36.9.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.EnvironmentError</td>
<td>errno, filename, strerror</td>
</tr>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
36.10 Class CLVar

A class for command-line construction variables.

This is a list that uses Split() to split an initial string along white-space arguments, and similarly to split any strings that get added. This allows us to Do the Right Thing with Append() and Prepend() (as well as straight Python foo = env[‘VAR’] + ‘arg1 arg2’) regardless of whether a user adds a list or a string to a command-line construction variable.

36.10.1 Methods

```python
__init__(self, seq=[])

x.__init__(...) initializes x; see help(type(x)) for signature  
Overrides: object.__init__ extit(inherited documentation)
```

```python
__add__(self, other)

Overrides: UserList.UserList.__add__
```

```python
__radd__(self, other)

Overrides: UserList.UserList.__radd__
```

```python
__coerce__(self, other)
```

Class OrderedDict

Module SCons.Util

```python
__str__(self)
str(x) Overrides: object.__str__ (inherited documentation)
```

Inherited from UserList.UserList

```python
__cmp__(), __contains__(), __delitem__(), __delattr__(), __eq__(), __getattribute__(), __getitem__(), __getslice__(), __gt__(), __iadd__(), __imul__(), __le__(), __len__(), __lt__(), __mul__(), __ne__(), __nonzero__(), __repr__(), __reversed__(), __rmul__(), __setattr__(), __setitem__(), __setslice__(), append(), count(), extend(), index(), insert(), pop(), remove(), reverse(), sort()
```

Inherited from _abcoll.Sequence

```python
__iter__(), __reversed__()
```

Inherited from _abcoll.Sized

```python
__subclasshook__()
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__()
```

36.10.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

36.10.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from UserList.UserList</td>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
</tr>
</tbody>
</table>

36.11 Class OrderedDict

UserDict.UserDict

| SCons.Util.OrderedDict |

Known Subclasses: SCons.Util.Selector

344
36.11.1 Methods

```python
__init__(self, dict=None)
Overides: UserDict.UserDict.__init__
```

```python
__delitem__(self, key)
Overides: UserDict.UserDict.__delitem__
```

```python
__setitem__(self, key, item)
Overides: UserDict.UserDict.__setitem__
```

```python
clear(self)
Overides: UserDict.UserDict.clear
```

```python
copy(self)
Overides: UserDict.UserDict.copy
```

```python
items(self)
Overides: UserDict.UserDict.items
```

```python
keys(self)
Overides: UserDict.UserDict.keys
```

```python
popitem(self)
Overides: UserDict.UserDict.popitem
```

```python
setdefault(self, key, failobj=None)
Overides: UserDict.UserDict.setdefault
```

```python
update(self, dict)
Overides: UserDict.UserDict.update
```

```python
values(self)
Overides: UserDict.UserDict.values
```

*Inherited from UserDict.UserDict*

```python
__cmp__(), __contains__(), __getitem__(), __len__(), __repr__(), fromkeys(),
get(), has_key(), iteritems(), iterkeys(), itervalues(), pop()
```
36.11.2 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from UserDict.UserDict</td>
<td></td>
</tr>
<tr>
<td><strong>hash</strong></td>
<td></td>
</tr>
</tbody>
</table>

36.12 Class Selector

UserDict.UserDict  

SCons.Util.OrderedDict  

SCons.Util.Selector


A callable ordered dictionary that maps file suffixes to dictionary values. We preserve the order in which items are added so that get_suffix() calls always return the first suffix added.

36.12.1 Methods

```python
__call__(self, env, source, ext=None)
```

**Inherited from SCons.Util.OrderedDict (Section 36.11)**

__delitem__(), __init__(), __setitem__(), clear(), copy(), items(), keys(), popitem(), setdefault(), update(), values()

**Inherited from UserDict.UserDict**

__cmp__(), __contains__(), __getitem__(), __len__(), __repr__(), fromkeys(), get(), has_key(), iteritems(), iterkeys(), itervalues(), pop()

36.12.2 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from UserDict.UserDict</td>
<td></td>
</tr>
<tr>
<td><strong>hash</strong></td>
<td></td>
</tr>
</tbody>
</table>
36.13 Class LogicalLines

Wrapper class for the logical_lines method.

Allows us to read all "logical" lines at once from a given file object.

36.13.1 Methods

```python
__init__(self, fileobj)
```

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides:

```
object.__init__ extit(inherited documentation)
```

```python
readlines(self)
```

Inherited from object

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

36.13.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
36.14 Class UniqueList

```
object    ┌─────────┐
          │          │
          │  _abcoll.Sized  │
          │              │
object    ┌─────────┐
          │  _abcoll.Iterable  │
          │              │
object    ┌─────────┐
          │  _abcoll.Container  │
          │              │
          │  _abcoll.Sequence  │
          │              │
          │  _abcoll.MutableSequence  │
          │              │
UserList.UserList    ┌─────────┐
                    │  SCons.Util.UniqueList  │
```

36.14.1 Methods

```python
__init__(self, seq=[])  
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: 
object.__init__ (inherited documentation)
```

```python
__lt__(self, other)  
 Overrides: UserList.UserList.__lt__
```

```python
__le__(self, other)  
 Overrides: UserList.UserList.__le__
```

```python
__eq__(self, other)  
 Overrides: UserList.UserList.__eq__
```

```python
__ne__(self, other)  
 Overrides: UserList.UserList.__ne__
```
Class UniqueList
Module SCons.Util

```python
__gt__(self, other)
 Overrides: UserList.UserList.__gt__

__ge__(self, other)
 Overrides: UserList.UserList.__ge__

__cmp__(self, other)
 Overrides: UserList.UserList.__cmp__

__len__(self)
 Overrides: _abcoll.Sized.__len__

__getitem__(self, i)
 Overrides: _abcoll.Sequence.__getitem__

__setitem__(self, i, item)
 Overrides: _abcoll.MutableSequence.__setitem__

__getslice__(self, i, j)
 Overrides: UserList.UserList.__getslice__

__setslice__(self, i, j, other)
 Overrides: UserList.UserList.__setslice__

__add__(self, other)
 Overrides: UserList.UserList.__add__

__radd__(self, other)
 Overrides: UserList.UserList.__radd__

__iadd__(self, other)
 Overrides: _abcoll.MutableSequence.__iadd__

__mul__(self, other)
 Overrides: UserList.UserList.__mul__
```
### Class UniqueList

**Module**: SCons.Util

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong><strong>rmul</strong>(self, other)</strong></td>
<td>Overrides: UserList.UserList.<strong>rmul</strong></td>
</tr>
<tr>
<td><strong><strong>imul</strong>(self, other)</strong></td>
<td>Overrides: UserList.UserList.<strong>imul</strong></td>
</tr>
<tr>
<td><strong>append(self, item)</strong></td>
<td>Append object to the end of the sequence. Overrides: _abcoll.MutableSequence.append (inherited documentation)</td>
</tr>
<tr>
<td><strong>insert(self, i)</strong></td>
<td>Insert object before index. Overrides: _abcoll.MutableSequence.insert (inherited documentation)</td>
</tr>
<tr>
<td><strong>count(self, item)</strong></td>
<td>Return number of occurrences of value. <strong>Return Value</strong>: integer. Overrides: _abcoll.Sequence.count (inherited documentation)</td>
</tr>
<tr>
<td><strong>index(self, item)</strong></td>
<td>Return first index of value. Raises ValueError if the value is not present. <strong>Return Value</strong>: integer. Overrides: _abcoll.Sequence.index (inherited documentation)</td>
</tr>
<tr>
<td><strong>reverse(self)</strong></td>
<td>Reverse in place. Overrides: _abcoll.MutableSequence.reverse (inherited documentation)</td>
</tr>
<tr>
<td>**sort(self, *args, <strong>kwds)</strong></td>
<td>Overrides: UserList.UserList.sort</td>
</tr>
<tr>
<td><strong>extend(self, other)</strong></td>
<td>Extend sequence by appending elements from the iterable. Overrides: _abcoll.MutableSequence.extend (inherited documentation)</td>
</tr>
</tbody>
</table>
Inherited from `UserList.UserList`

__contains__(), __delitem__(), __delslice__(), __repr__(), pop(), remove()

Inherited from `_abcoll.Sequence`

__iter__(), __reversed__()

Inherited from `_abcoll.Sized`

__subclasshook__()

Inherited from `object`

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__reduce_ex__(), __setattr__(), __sizeof__(), __str__()

### 36.14.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 36.14.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Inherited from UserList.UserList</code></td>
<td></td>
</tr>
<tr>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 36.15 Class Unbuffered

object

```
SCons.Util.Unbuffered
```

A proxy class that wraps a file object, flushing after every write, and delegating everything else to the wrapped object.
36.15.1 Methods

```python
__init__(self, file)
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ (inherited documentation)
```

```python
write(self, arg)
```

```python
__getattr__(self, attr)
```

*Inherited from object*

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

36.15.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

36.16 Class Null

```
object -> SCons.Util.Null
```


Null objects always and reliably "do nothing."

36.16.1 Methods

```python
__new__(cls, *args, **kwargs)
```

*Return Value*

A new object with type S, a subtype of T

*Overrides:* object.__new__ (inherited documentation)
__init__(self, *args, **kwargs)

x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ (inherited documentation)

__call__(self, *args, **kwargs)

__repr__(self)

repr(x)  Overrides: object.__repr__ (inherited documentation)

__nonzero__(self)

__bool__(self)

__getattr__(self, name)

__setattr__(self, name, value)

x.__setattr__('name', value) <==> x.name = value  Overrides:
object.__setattr__ (inherited documentation)

__delattr__(self, name)

x.__delattr__('name') <==> del x.name  Overrides: object.__delattr__
(inherited documentation)

Inherited from object

__format__(), __getattribute__(), __hash__(), __reduce__(), __reduce_ex__(),
__sizeof__(), __str__(), __subclasshook__()

36.16.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
36.17 Class NullSeq

```
object
SCons.Util.Null
SCons.Util.NullSeq
```

**Known Subclasses:** SCons.Subst_NullNodeList

### 36.17.1 Methods

- `__len__(self)`
- `__iter__(self)`
- `__getitem__(self, i)`
- `__delitem__(self, i)`
- `__setitem__(self, i, v)`

*Inherited from SCons.Util.Null (Section 36.16)*

- `__bool__()`, `__call__()`, `__delattr__()`, `__getattr__()`, `__init__()`, `__new__()`, `__nonzero__()`, `__repr__()`, `__setattr__()`

*Inherited from object*

- `__format__()`, `__getattribute__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 36.17.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>

*Inherited from object*
37 Package SCons.Variablies

engine.SCons.Variablies

This file defines the Variables class that is used to add user-friendly customizable variables to an SCons build.

37.1 Modules

- **BoolVariable** (*Section ??, p. ??*)
- **BoolVariable**: engine.SCons.Variablies.BoolVariable (*Section 38, p. 344*)
- **EnumVariable** (*Section ??, p. ??*)
- **EnumVariable**: engine.SCons.Variablies.EnumVariable (*Section 39, p. 345*)
- **ListVariable** (*Section ??, p. ??*)
- **ListVariable**: engine.SCons.Variablies.ListVariable (*Section 40, p. 346*)
- **PackageVariable** (*Section ??, p. ??*)
- **PackageVariable**: engine.SCons.Variablies.PackageVariable (*Section 40, p. 346*)
- **PathVariable** (*Section ??, p. ??*)
- **PathVariable**: SCons.Variablies.PathVariable (*Section 42, p. 348*)

37.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Variablies/<strong>init</strong>.py rel_3.0.0:4395:89...'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Variablies'</td>
</tr>
</tbody>
</table>

37.3 Class Variables

```
object

SCons.Variablies.Variablies
```
37.3.1 Methods

```python
__init__(self, files=None, args=None, is_global=1)
```

files - [optional] List of option configuration files to load

(backward compatibility) If a single string is passed it is automatically placed in a file list

Overrides: object.__init__

```python
keys(self)
```

Returns the keywords for the options

```python
Add(self, key, help='', default=None, validator=None, converter=None, **kw)
```

Add an option.

@param key: the name of the variable, or a list or tuple of arguments @param help: optional help text for the options @param default: optional default value @param validator: optional function that is called to validate the option’s value @type validator: Called with (key, value, environment) @param converter: optional function that is called to convert the option’s value before putting it in the environment.
## AddVariables

**AddVariables**(*self, *optlist*)

Add a list of options.

Each list element is a tuple/list of arguments to be passed on to the underlying method for adding options.

Example:

```python
opt.AddVariables(
    ('debug', '', 0),
    ('CC', 'The C compiler'),
    ('VALIDATE', 'An option for testing validation', 'notset',
     validator, None),
)
```

## Update

**Update**(*self, env, args=None*)

Update an environment with the option variables.

env - the environment to update.

## UnknownVariables

**UnknownVariables**(*self*)

Returns any options in the specified arguments lists that were not known, declared options in this object.

## Save

**Save**(*self, filename, env*)

Saves all the options in the given file. This file can then be used to load the options next run. This can be used to create an option cache file.

filename - Name of the file to save into env - the environment get the option values from
**GenerateHelpText** *(self, env, sort=None)*

Generate the help text for the options.

**env** - an environment that is used to get the current values of the options.

**FormatVariableHelpText** *(self, env, key, help, default, actual, aliases=[])*

**Inherited from object**

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()

37.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

37.3.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance</td>
<td>Holds all the options, updates the environment with the variables, and renders the help text. Value: None</td>
</tr>
<tr>
<td>format</td>
<td>Value: '\n%s: %s\n default: %s\n actual: %s\n'</td>
</tr>
<tr>
<td>format_</td>
<td>Value: '\n%s: %s\n default: %s\n actual: %s aliases: ...'</td>
</tr>
</tbody>
</table>
38 Module SCons.Variables.BoolVariable'

This file defines the option type for SCons implementing true/false values.

Usage example:

```python
opts = Variables()
opts.Add(BoolVariable('embedded', 'build for an embedded system', 0))
...
if env['embedded'] == 1:
...
```

38.1 Functions

<table>
<thead>
<tr>
<th>BoolVariable(key, help, default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The input parameters describe a boolean option, thus they are returned with the correct converter and validator appended. The 'help' text will be appended by '(yes</td>
</tr>
</tbody>
</table>

39 Module SCons.Variables.EnumVariable'

engine.SCons.Variables.EnumVariable

This file defines the option type for SCons allowing only specified input-values.

Usage example:

```python
opts = Variables()
opts.Add(EnumVariable('debug', 'debug output and symbols', 'no',
    allowed_values=('yes', 'no', 'full'),
    map={}, ignorecase=2))
...
if env['debug'] == 'full':
...
```

39.1 Functions

<table>
<thead>
<tr>
<th>EnumVariable(key, help, default, allowed_values, map={}, ignorecase=0)</th>
</tr>
</thead>
</table>

The input parameters describe an option with only certain values allowed. They are returned with an appropriate converter and validator appended. The result is usable for input to Variables.Add().

'key' and 'default' are the values to be passed on to Variables.Add().

'help' will be appended by the allowed values automatically

'allowed_values' is a list of strings, which are allowed as values for this option.

The 'map'-dictionary may be used for converting the input value into canonical values (e.g. for aliases).

'ignorecase' defines the behaviour of the validator:

If ignorecase == 0, the validator/converter are case-sensitive. If ignorecase == 1, the validator/converter are case-insensitive. If ignorecase == 2, the validator/converter is case-insensitive and the converted value will always be lower-case.

The 'validator' tests whether the value is in the list of allowed values. The 'converter' converts input values according to the given 'map'-dictionary (unmapped input values are returned unchanged).
40 Module SCons.Variables.ListVariable

engine.SCons.Variables.ListVariable

This file defines the option type for SCons implementing 'lists'.

A 'list' option may either be 'all', 'none' or a list of names separated by comma. After the option has been processed, the option value holds either the named list elements, all list elements or no list elements at all.

Usage example:

```python
list_of_libs = Split('x11 gl qt ical')

opts = Variables()
opts.Add(ListVariable('shared',
    'libraries to build as shared libraries',
    'all',
    elems = list_of_libs))

...  
for lib in list_of_libs:
    if lib in env['shared']:
        env.SharedObject(...)  
    else:
        env.Object(...)  
```

40.1 Functions

```
ListVariable(key, help, default, names, map={})
```

The input parameters describe a 'package list' option, thus they are returned with the correct converter and validator appended. The result is usable for input to opts.Add().

A 'package list' option may either be 'all', 'none' or a list of package names (separated by space).
Module SCons.Variables.PackageVariable

This file defines the option type for SCons implementing 'package activation'.
To be used whenever a 'package' may be enabled/disabled and the package path may be specified.

Usage example:

**Examples:** x11=no (disables X11 support) x11=yes (will search for the package installation dir) x11=/usr/local/X11 (will check this path for existence)

To replace autoconf's --with-xxx=yyy

```python
opts = Variables()
opts.Add(PackageVariable('x11',
   'use X11 installed here (yes = search some places',
   'yes'))
```

```python
if env['x11'] == True:
   dir = ... search X11 in some standard places ...
   env['x11'] = dir
if env['x11']:
   ... build with x11 ...
```

41.1 Functions

<table>
<thead>
<tr>
<th>PackageVariable(key, help, default, searchfunc=None)</th>
</tr>
</thead>
</table>

The input parameters describe a 'package list' option, thus they are returned with the correct converter and validator appended. The result is usable for input to opts.Add().

A 'package list' option may either be 'all', 'none' or a list of package names (separated by space).
Module SCons.Variables.PathVariable’

SCons.Variables.PathVariable

This file defines an option type for SCons implementing path settings.

To be used whenever a user-specified path override should be allowed.

**Arguments to PathVariable are:** option-name = name of this option on the command line (e.g. 'prefix') option-help = help string for option option-dflt = default value for this option validator = [optional] validator for option value. Predefined validators are:

- PathAccept -- accepts any path setting; no validation
- PathIsDir -- path must be an existing directory
- PathIsDirCreate -- path must be a dir; will create
- PathIsFile -- path must be a file
- PathExists -- path must exist (any type) [default]

The validator is a function that is called and which should return True or False to indicate if the path is valid. The arguments to the validator function are: (key, val, env). The key is the name of the option, the val is the path specified for the option, and the env is the env to which the Options have been added.

Usage example:

```
Examples:
  prefix=/usr/local

opts = Variables()

opts = Variables()
 opts.Add(PathVariable('qtdir','where the root of Qt is installed',
                     qtdir, PathIsDir))
 opts.Add(PathVariable('qt_includes','where the Qt includes are installed',
                     '$qtdir/includes', PathIsDirCreate))
 opts.Add(PathVariable('qt_libraries','where the Qt library is installed',
                     '$qtdir/lib'))
```

42.1 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathVariable</td>
<td>Value:</td>
</tr>
<tr>
<td></td>
<td>&lt;SCons.Variables.PathVariable._PathVariableClass object&gt;</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>

Module SCons.Variables.PathVariable'
43  Module SCons.Warnings

SCons.Warnings

This file implements the warnings framework for SCons.

43.1  Functions

**suppressWarningClass**(clazz)

Suppresses all warnings that are of type clazz or derived from clazz.

**enableWarningClass**(clazz)

Enables all warnings that are of type clazz or derived from clazz.

**warningAsException**(flag=1)

Turn warnings into exceptions. Returns the old value of the flag.

**warn**(clazz, *args)
process_warn_strings(\textit{arguments})

Process string specifications of enabling/disabling warnings, as passed to the --warn option or the SetOption('warn') function.

An argument to this option should be of the form \textless warning-class\textgreater{} or no-\textless warning-class\textgreater{}. The warning class is munged in order to get an actual class name from the classes above, which we need to pass to the \{enable,disable\}WarningClass() functions. The supplied \textless warning-class\textgreater{} is split on hyphens, each element is capitalized, then smushed back together. Then the string 'Warning' is appended to get the class name.

For example, 'deprecated' will enable the DeprecatedWarning class. 'no-dependency' will disable the DependencyWarning class.

As a special case, --warn=all and --warn=no-all will enable or disable (respectively) the base Warning class of all warnings.

### 43.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>___revision__</td>
<td>Value: \texttt{'src/engine/SCons/Warnings.py rel_3_0_0:4395:8972f6a2f699...}</td>
</tr>
<tr>
<td>___package__</td>
<td>Value: \texttt{'SCons'}</td>
</tr>
</tbody>
</table>

### 43.3 Class Warning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

\texttt{SCons.Warnings.Warning}

Class WarningOnByDefault

Module SCons.Warnings

43.3.1 Methods

Inherited from exceptions.Exception

`__init__()`, `__new__()`

Inherited from exceptions.BaseException

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

Inherited from object

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

43.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><code>__class__</code></td>
</tr>
</tbody>
</table>

43.4 Class WarningOnByDefault

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

43.4.1 Methods

*Inherited from exceptions.Exception*

\_\_init\_\_(), \_\_new\_\_()

*Inherited from exceptions.BaseException*

\_\_delattr\_\_(), \_\_getattribute\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

*Inherited from object*

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

43.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td>__class_</td>
</tr>
</tbody>
</table>

43.5 Class TargetNotBuiltWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.TargetNotBuiltWarning

43.5.1 Methods

*Inherited from exceptions.Exception*

\_\_init\_\_(), \_\_new\_\_()
### Inherited from `exceptions.BaseException`

- `__delattr__()`
- `__getattribute__()`
- `__getitem__()`
- `__getslice__()`
- `__reduce__()`
- `__repr__()`
- `__setattr__()`
- `__setstate__()`
- `__str__()`
- `__unicode__()`

### Inherited from `object`

- `__format__()`
- `__hash__()`
- `__reduce_ex__()`
- `__sizeof__()`
- `__subclasshook__()`

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Inherited from exceptions.BaseException</code></td>
<td>args, message</td>
</tr>
<tr>
<td><code>Inherited from object</code></td>
<td><code>__class__</code></td>
</tr>
</tbody>
</table>

### 43.6 Class CacheVersionWarning

```
object
```
```
exceptions.BaseException
```
```
exceptions.Exception
```
```
SCons.Errors.UserError
```
```
SCons.Warnings.Warning
```
```
SCons.Warnings.WarningOnByDefault
```
```
SCons.Warnings.CacheVersionWarning
```

### 43.6.1 Methods

**Inherited from `exceptions.Exception`**

- `__init__()`
- `__new__()`

**Inherited from `exceptions.BaseException`**

- `__delattr__()`
- `__getattribute__()`
- `__getitem__()`
- `__getslice__()`
- `__reduce__()`
Class CacheWriteErrorWarning

Inherited from object

43.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td></td>
</tr>
<tr>
<td>args, message</td>
<td></td>
</tr>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

43.7 Class CacheWriteErrorWarning

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
43.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from exceptions.BaseException</em></td>
<td>args, message</td>
</tr>
<tr>
<td><em>Inherited from object</em></td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

43.8 Class CorruptSConsignWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.CorruptSConsignWarning

43.8.1 Methods

*Inherited from exceptions.Exception*

__init__(), __new__()

*Inherited from exceptions.BaseException*

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

*Inherited from object*

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.8.2 Properties
43.9 Class DependencyWarning

object ──

exceptions.BaseException ──

exceptions.Exception ──

SCons.Errors.UserError ──

SCons.Warnings.Warning ──

SCons.Warnings.DependencyWarning

43.9.1 Methods

Inherited from exceptions.Exception

__init__(), __new__() 

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.9.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.10  Class DevelopmentVersionWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.DevelopmentVersionWarning

43.10.1  Methods

*Inherited from exceptions.Exception*

  __init__(), __new__()  

*Inherited from exceptions.BaseException*

  __delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()  

*Inherited from object*

  __format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()  

43.10.2  Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td></td>
</tr>
<tr>
<td>args, message</td>
<td></td>
</tr>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
Class DuplicateEnvironmentWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.DuplicateEnvironmentWarning

43.11 Methods

_Inherited from exceptions.Exception_

`__init__()`, `__new__()`

_Inherited from exceptions.BaseException_

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

_Inherited from object_

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

43.11.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.12 Class FutureReservedVariableWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.FutureReservedVariableWarning

43.12.1 Methods

_Inherited from exceptions.Exception_

__init__(), __new__()

_Inherited from exceptions.BaseException_

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

_Inherited from object_

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.12.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.13 Class LinkWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.LinkWarning

Known Subclasses: SCons.Warnings.FortranCxxMixWarning

43.13.1 Methods

**Inherited from exceptions.Exception**

__init__(), __new__()  

**Inherited from exceptions.BaseException**

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

**Inherited from object**

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()  

43.13.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.14 Class MisleadingKeywordsWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.MisleadingKeywordsWarning

43.14.1 Methods

*Inherited from exceptions.Exception*

`__init__()`, `__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

43.14.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from exceptions.BaseException</em></td>
<td>args, message</td>
</tr>
<tr>
<td><em>Inherited from object</em></td>
<td><code>__class__</code></td>
</tr>
</tbody>
</table>

377
43.15 Class MissingSConscriptWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.MissingSConscriptWarning

43.15.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.15.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.16 Class NoMD5ModuleWarning

Inherited from exceptions.Exception

__init__(), __new__()  

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.16.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.17 Class NoMetaClassSupportWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.NoMetaClassSupportWarning

43.17.1 Methods

Inherited from exceptions.Exception

    __init__(), __new__()  

Inherited from exceptions.BaseException

    __delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

    __format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.17.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.18  Class NoObjectCountWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.NoObjectCountWarning

43.18.1  Methods

*Inherited from exceptions.Exception*

__init__(), __new__()  

*Inherited from exceptions.BaseException*

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

*Inherited from object*

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.18.2  Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.19 Class NoParallelSupportWarning

```
object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.NoParallelSupportWarning
```

43.19.1 Methods

*Inherited from exceptions.Exception*

```
__init__(), __new__()  
```

*Inherited from exceptions.BaseException*

```
__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()
```

*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

43.19.2 Properties

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
```
43.20  Class ReservedVariableWarning

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.20.2  Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td></td>
</tr>
<tr>
<td>args, message</td>
<td></td>
</tr>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.21 Class StackSizeWarning

object
exceptions.BaseException
exceptions.Exception
SCons.Errors.UserError
SCons.Warnings.Warning
SCons.Warnings.WarningOnByDefault
SCons.Warnings.StackSizeWarning

43.21.1 Methods

*Inherited from exceptions.Exception*

__init__(), __new__() 

*Inherited from exceptions.BaseException*

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

*Inherited from object*

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.21.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from exceptions.BaseException</em></td>
<td>args, message</td>
</tr>
<tr>
<td><em>Inherited from object</em></td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.22 Class VisualCMissingWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.VisualCMissingWarning

43.22.1 Methods

_Inherited from exceptions.Exception_

__init__(), __new__()

_Inherited from exceptions.BaseException_

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

_Inherited from object_

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.22.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

385
43.23 Class VisualVersionMismatch

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.VisualVersionMismatch

43.23.1 Methods

*Inherited from exceptions.Exception*

`__init__()`, `__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

43.23.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

386
43.24 Class VisualStudioMissingWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.VisualStudioMissingWarning

43.24.1 Methods

_Inherited from exceptions.Exception_

__init__(), __new__()  

_Inherited from exceptions.BaseException_

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()  

_Inherited from object_

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()  

43.24.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td></td>
</tr>
<tr>
<td>args, message</td>
<td></td>
</tr>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.25 Class FortranCxxMixWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.LinkWarning

SCons.Warnings.FortranCxxMixWarning

43.25.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()  

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getstate__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.25.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.26 Class FutureDeprecatedWarning

```
object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.FutureDeprecatedWarning
```

**Known Subclasses:** SCons.Warnings.DeprecatedSourceCodeWarning

43.26.1 Methods

**Inherited from exceptions.Exception**

- `__init__()`, `__new__()`

**Inherited from exceptions.BaseException**

- `__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

**Inherited from object**

- `__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

43.26.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Inherited from exceptions.BaseException</code></td>
<td>args, message</td>
</tr>
<tr>
<td><code>Inherited from object</code></td>
<td><code>__class__</code></td>
</tr>
</tbody>
</table>
43.27 Class DeprecatedWarning

```
object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning
```


43.27.1 Methods

*Inherited from exceptions.Exception*

```
__init__() , __new__()
```

*Inherited from exceptions.BaseException*

```
__delattr__() , __getattribute__() , __getitem__() , __getslice__() , __reduce__() , __repr__() , __setattr__() , __setstate__() , __str__() , __unicode__()
```

*Inherited from object*

```
__format__() , __hash__() , __reduce_ex__() , __sizeof__() , __subclasshook__()
```

43.27.2 Properties

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from exceptions.BaseException</em></td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
```
43.28 Class MandatoryDeprecatedWarning

object  

exceptions.BaseException  

exceptions.Exception  

SCons.Errors.UserError  

SCons.Warnings.Warning  

SCons.Warnings.DeprecatedWarning  

SCons.Warnings.MandatoryDeprecatedWarning


43.28.1 Methods

*Inherited from exceptions.Exception*

`__init__()`, `__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

43.28.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><code>__class__</code></td>
</tr>
</tbody>
</table>

391
43.29 Class PythonVersionWarning

```
object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.PythonVersionWarning
```

43.29.1 Methods

*Inherited from exceptions.Exception*

```python
__init__(), __new__()
```

*Inherited from exceptions.BaseException*

```python
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()
```

*Inherited from object*

```python
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

43.29.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from exceptions.BaseException</em></td>
<td></td>
</tr>
<tr>
<td>args, message</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.30 Class DeprecatedSourceCodeWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.FutureDeprecatedWarning

SCons.Warnings.DeprecatedSourceCodeWarning

43.30.1 Methods

Inherited from exceptions.Exception

___init__() , ___new__() 

Inherited from exceptions.BaseException

__delattr__() , __getattribute__() , __getitem__() , __getslice__() , __reduce__() , __repr__() , __setattr__() , __setstate__() , __str__() , __unicode__() 

Inherited from object

__format__() , __hash__() , __reduce_ex__() , __sizeof__() , __subclasshook__() 

43.30.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td>_<strong>class</strong></td>
</tr>
</tbody>
</table>
43.31 Class DeprecatedBuildDirWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.DeprecatedBuildDirWarning

43.31.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.31.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.32 Class TaskmasterNeedsExecuteWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.TaskmasterNeedsExecuteWarning

43.32.1 Methods

_Inherited from exceptions.Exception_

__init__(), __new__()  

_Inherited from exceptions.BaseException_

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), reduce(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

_Inherited from object_

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.32.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.33 Class DeprecatedCopyWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedCopyWarning

43.33.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.33.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.34 Class DeprecatedOptionsWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedOptionsWarning

43.34.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()  

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.34.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.35 Class DeprecatedSourceSignaturesWarning

Inherited from exceptions.BaseException

__init__(), __new__()

Inherited from exceptions.Exception

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
43.36 Class DeprecatedTargetSignaturesWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedTargetSignaturesWarning

43.36.1 Methods

*Inherited from exceptions.Exception*

__init__(), __new__()

*Inherited from exceptions.BaseException*

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

*Inherited from object*

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

43.36.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.37 Class DeprecatedDebugOptionsWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

Inherited from exceptions.Exception

__init__() , __new__() 

Inherited from exceptions.BaseException

__delattr__() , __getattr__() , __getitem__() , __getslice__() , __reduce__() , __repr__() , __setattr__() , __setstate__() , __str__() , __unicode__() 

Inherited from object

__format__() , __hash__() , __reduce_ex__() , __sizeof__() , __subclasshook__() 

43.37.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.38 Class DeprecatedSigModuleWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedSigModuleWarning

43.38.1 Methods

*Inherited from exceptions.Exception*

__init__(),  __new__()  

*Inherited from exceptions.BaseException*

__delattr__(),  __getattribute__(),  __getitem__(),  __getslice__(),  __reduce__(),  __repr__(),  __setattr__(),  __setstate__(),  __str__(),  __unicode__()

*Inherited from object*

__format__(),  __hash__(),  __reduce_ex__(),  __sizeof__(),  __subclasshook__()

43.38.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from exceptions.BaseException</em></td>
<td></td>
</tr>
<tr>
<td>args, message</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.39 Class DeprecatedBuilderKeywordsWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedBuilderKeywordsWarning

43.39.1 Methods

*Inherited from exceptions.Exception*

    __init__(), __new__()

*Inherited from exceptions.BaseException*

    __delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

*Inherited from object*

    __format__(), __hash__(), __reduce_ex__(), __setstate__(), __sizeof__(), __subclasshook__()

43.39.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from exceptions.BaseException</em></td>
<td>args, message</td>
</tr>
<tr>
<td><em>Inherited from object</em></td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

402
44 Module SCons.cpp

SCons C Pre-Processor module

44.1 Functions

<table>
<thead>
<tr>
<th>CPP_to_Python_Ops_Sub(m)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CPP_to_Python(s)</th>
</tr>
</thead>
</table>

Converts a C pre-processor expression into an equivalent Python expression that can be evaluated.

44.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Value: ...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>doc</strong></td>
<td></td>
</tr>
<tr>
<td>cpp_lines_dict</td>
<td>Value: {('define'): \s+([_A-Za-z][_A-Za-z0-9]<em>)((([^]])</em>))?...</td>
</tr>
<tr>
<td>Table</td>
<td>Value: {'define': re.compile(r'\s+([_A-Za-z][_A-Za-z0-9]<em>)(([^)])</em>))(([^...</td>
</tr>
<tr>
<td>e</td>
<td>Value: '^\s*#\s*(elif</td>
</tr>
<tr>
<td>CPP_Expression</td>
<td>Value: re.compile(r'(?m)\s*#\s*(elif</td>
</tr>
<tr>
<td>CPP_to_Python_Ops__Dict</td>
<td>Value: {'\r': '', '(': 'not', '!=': 'and', ',' : 'if'...</td>
</tr>
<tr>
<td>CPP_to_Python_Ops__Expression</td>
<td>Value: re.compile(r'\r</td>
</tr>
<tr>
<td>CPP_to_Python__Eval__List</td>
<td>Value: [[re.compile(r'defined\s+(w+)'), ,1&quot; in <strong>dict</strong>'], [...</td>
</tr>
<tr>
<td>line_continuations</td>
<td>Value: re.compile(r'\r</td>
</tr>
<tr>
<td>function_name</td>
<td>Value: re.compile(r'(S+)(([^)])*))'</td>
</tr>
<tr>
<td>function_arg_separator</td>
<td>Value: re.compile(r',' )</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
<tr>
<td>x</td>
<td>Value: 'if'</td>
</tr>
</tbody>
</table>

403
44.3 Class FunctionEvaluator

```
object ->
SCons.cpp.FunctionEvaluator
```

Handles delayed evaluation of a #define function call.

44.3.1 Methods

```
__init__(self, name, args, expansion)
```

Squirrels away the arguments and expansion value of a #define macro function for later evaluation when we must actually expand a value that uses it.

Overrides: object.__init__

```
__call__(self, *values)
```

Evaluates the expansion of a #define macro function called with the specified values.

*Inherited from object*

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

44.3.2 Properties

```
Name | Description
Inherited from object
___class___
```

44.4 Class PreProcessor

```
object ->
SCons.cpp.PreProcessor
```

*Known Subclasses:* SCons.cpp.DumbPreProcessor, SCons.Scanner.C.SConsCPPScanner
The main workhorse class for handling C pre-processing.

44.4.1  Methods

```python
__call__(self, file)
```
Pre-processes a file.
This is the main public entry point.

```python
__init__(self, current='.', cpppath=(), dict={}, all=0)
```
x.__init__(...) initializes x; see help(type(x)) for signature  Overrides:
object.__init__ extit(inherited documentation)

```python
all__include__(self, t)
```

```python
do__define__(self, t)
```
Default handling of a #define line.

```python
do__elif__(self, t)
```
Default handling of a #elif line.

```python
do__else__(self, t)
```
Default handling of a #else line.

```python
do__endif__(self, t)
```
Default handling of a #endif line.
do_if(self, t)

Default handling of a #if line.

do_ifdef(self, t)

Default handling of a #ifdef line.

do ifndef(self, t)

Default handling of a #ifndef line.

do_import(self, t)

Default handling of a #import line.

do_include(self, t)

Default handling of a #include line.

do_include_next(self, t)

Default handling of a #include line.

do_nothing(self, t)

Null method for when we explicitly want the action for a specific preprocessor directive to do nothing.

do_undef(self, t)

Default handling of a #undef line.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>eval_expression</strong>(self, t)</td>
<td>Evaluates a C preprocessor expression. This is done by converting it to a Python equivalent and eval()ing it in the C preprocessor namespace we use to track #define values.</td>
</tr>
<tr>
<td><strong>finalize_result</strong>(self, fname)</td>
<td></td>
</tr>
<tr>
<td><strong>find_include_file</strong>(self, t)</td>
<td>Finds the #include file for a given preprocessor tuple.</td>
</tr>
<tr>
<td><strong>initialize_result</strong>(self, fname)</td>
<td></td>
</tr>
<tr>
<td><strong>process_contents</strong>(self, contents, fname=None)</td>
<td>Pre-processes a file contents. This is the main internal entry point.</td>
</tr>
<tr>
<td><strong>read_file</strong>(self, file)</td>
<td></td>
</tr>
</tbody>
</table>
| **resolve_include**(self, t) | Resolve a tuple-ized #include line. This handles recursive expansion of values without "" or <> surrounding the name until an initial " or < is found, to handle

    #include FILE

where FILE is a #define somewhere else. |
| **restore**(self) | Pops the previous dispatch table off the stack and makes it the current one. |
**save***(self)**

Pushes the current dispatch table on the stack and re-initializes the current dispatch table to the default.

**scons_current_file**(self, t)

**start_handling_includes**(self, t=None)

Causes the PreProcessor object to start processing #import, #include and #include_next lines.

This method will be called when a #if, #ifdef, #ifndef or #elif evaluates True, or when we reach the #else in a #if, #ifdef, #ifndef or #elif block where a condition already evaluated False.

**stop_handling_includes**(self, t=None)

Causes the PreProcessor object to stop processing #import, #include and #include_next lines.

This method will be called when a #if, #ifdef, #ifndef or #elif evaluates False, or when we reach the #else in a #if, #ifdef, #ifndef or #elif block where a condition already evaluated True.

**tupleize**(self, contents)

Turns the contents of a file into a list of easily-processed tuples describing the CPP lines in the file.

The first element of each tuple is the line’s preprocessor directive (#if, #include, #define, etc., minus the initial ’#’). The remaining elements are specific to the type of directive, as pulled apart by the regular expression.

*Inherited from object*

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
44.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

44.5 Class DumbPreProcessor

object  
SCons.cpp.PreProcessor  
SCons.cpp.DumbPreProcessor

A preprocessor that ignores all #if/#elif/#else/#endif directives and just reports back all of the #include files (like the classic SCons scanner did).

This is functionally equivalent to using a regular expression to find all of the #include lines, only slower. It exists mainly as an example of how the main PreProcessor class can be sub-classed to tailor its behavior.

44.5.1 Methods

```python
__init__(self, *args, **kw)
```

x.__init__(...) initializes x; see help(type(x)) for signature  
Overrides: object.__init__  
Inherited from SCons.cpp.PreProcessor(Section 44.4)

```python
__call__(), all_include(), do_define(), do_elif(), do_else(), do endif(), do_if(), do_ifdef(), do_ifndef(), do_import(), do_include(), do_include_next(), do_nothing(), do_undef(), eval_expression(), finalize_result(), find_include_file(), initialize_result(), process_contents(), read_file(), resolve_include(), restore(), save(), scons_current_file(), start_handling_includes(), stop_handling_includes(), tupleize()
```

Inherited from object

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

44.5.2 Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
45 Module SCons.dblite

45.1 Functions

```python
corruption_warning(filename)
```

```python
is_string(s)
```

```python
is_bytes(s)
```

```python
unicode(s)
```

```python
open(file, flag=None, mode=438)
```

45.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>keep_all_files</td>
<td>Value: 0</td>
</tr>
<tr>
<td>ignore_corrupt_dbfiles</td>
<td>Value: 0</td>
</tr>
<tr>
<td>dblite_suffix</td>
<td>Value: '.dblite'</td>
</tr>
<tr>
<td>tmp_suffix</td>
<td>Value: '.tmp'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

45.3 Class dblite

```python
object
```

```python
SCons.dblite.dblite
```

Squirrel away references to the functions in various modules that we’ll use when our `__del__() ` method calls our `sync()` method during shutdown. We might get destroyed when Python is in the midst of tearing down the different modules we import in an essentially arbitrary order, and some of the various modules’s global attributes may already be wiped out from under us.

See the discussion at: http://mail.python.org/pipermail/python-bugs-list/2003-March/016877.html
45.3.1 Methods

```python
__init__(self, file_base_name, flag, mode)
```

x.__init__(...) initializes x; see help(type(x)) for signature

 Overrides: object.__init__ (inherited documentation)

```python
close(self)
```

```python
__del__(self)
```

```python
sync(self)
```

```python
__getitem__(self, key)
```

```python
__setitem__(self, key, value)
```

```python
keys(self)
```

```python
has_key(self, key)
```

```python
__contains__(self, key)
```

```python
iterkeys(self)
```

```python
__iter__(self)
```

```python
__len__(self)
```

**Inherited from object**

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
```

45.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
46 Module SCons.exitfuncs

SCons.exitfuncs

Register functions which are executed when SCons exits for any reason.

46.1 Functions

```
register(func, *targs, **kargs)
```

register a function to be executed upon normal program termination

func - function to be called at exit
targs - optional arguments to pass to func
kargs - optional keyword arguments to pass to func

46.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/exitfuncs.py' rel_3.0.0:4395:8972f6a2f69...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>
Index

SCons (package), 2–4
  SCons.Action (module), 5–18
    SCons.Action.Action (function), 6
    SCons.Action.ActionBase (class), 6–7
    SCons.Action.ActionCaller (class), 16–17
    SCons.Action.ActionFactory (class), 17–18
    SCons.Action.CommandAction (class), 7–9
    SCons.Action.CommandGeneratorAction (class), 9–11
    SCons.Action.default_exitstatfunc (function), 6
    SCons.Action.FunctionAction (class), 13–14
    SCons.Action.get_default_ENV (function), 6
    SCons.Action.LazyAction (class), 11–13
    SCons.Action.ListAction (class), 14–16
    SCons.Action.rfile (function), 6
  SCons.Builder (module), 19–30
    SCons.Builder.Builder (function), 20
    SCons.Builder.BuilderBase (class), 26–29
    SCons.Builder.CallableObject (class), 22
    SCons.Builder.CompositeBuilder (class), 29–30
    SCons.Builder.DictCmdGenerator (class), 20–22
    SCons.Builder.DictEmitter (class), 22–23
    SCons.Builder.EmitProxy (class), 26
    SCons.Builder.is_a_Builder (function), 20
    SCons.Builder.ListEmitter (class), 23–25
    SCons.Builder.match_splitext (function), 20
    SCons.Builder.OverrideWarner (class), 25–26
  SCons.CacheDir (module), 31–33
  SCons.Conftest (module), 34–38
  SCons.cpp (module), 388–395
  SCons.dblite (module), 396–397
    SCons.dblite.corruption_warning (function), 396
    SCons.dblite.dblite (class), 396–397
    SCons.dblite.is_bytes (function), 396
    SCons.dblite.is_string (function), 396
    SCons.dblite.open (function), 396
    SCons.dblite.unicode (function), 396
  SCons.Debug (module), 39–40
    SCons.Debug.caller_stack (function), 39
    SCons.Debug.caller_trace (function), 39
    SCons.Debug.countLoggedInstances (function), 39
    SCons.Debug.dump_caller_counts (function), 39
    SCons.Debug.dumpLoggedInstances (function), 39
    SCons.Debug.fetchLoggedInstances (function), 39
    SCons.Debug.func_shorten (function), 39
    SCons.Debug.listLoggedInstances (function), 39
    SCons.Debug.logInstanceCreation (function), 39
    SCons.Debug.memory (function), 39
    SCons.Debug.string_to_classes (function), 39
    SCons.Debug.Trace (function), 39
  SCons.Defaults (module), 41–45
  SCons.Environment (module), 46–75
    SCons.Environment.alias_builder (function), 46
    SCons.Environment.apply_tools (function), 46
    SCons.Environment.Base (class), 54–63, 66–75
  SCons.Environment.BuilderDict (class), 50–51

414
<table>
<thead>
<tr>
<th>Function/Method/Class</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCons.Environment.BuilderWrapper (class)</td>
<td>48–50</td>
</tr>
<tr>
<td>SCons.Environment.copy_non_reserved_keywords (function)</td>
<td>46</td>
</tr>
<tr>
<td>SCons.Environment.default_copy_from_cache (function)</td>
<td>46</td>
</tr>
<tr>
<td>SCons.Environment.default_decide_source (function)</td>
<td>46</td>
</tr>
<tr>
<td>SCons.Environment.default_decide_target (function)</td>
<td>46</td>
</tr>
<tr>
<td>SCons.Environment.is_valid_construction_var (function)</td>
<td>46</td>
</tr>
<tr>
<td>SCons.Environment.MethodWrapper (class)</td>
<td>47–48</td>
</tr>
<tr>
<td>SCons.Environment.NoSubstitutionProxy (function)</td>
<td>46</td>
</tr>
<tr>
<td>SCons.Environment.OverrideEnvironment (class)</td>
<td>63–66</td>
</tr>
<tr>
<td>SCons.Environment.SubstitutionEnvironment (class)</td>
<td>51–54</td>
</tr>
<tr>
<td>SCons.Executor (module)</td>
<td>76–83</td>
</tr>
<tr>
<td>SCons.Executor.Batch (class)</td>
<td>85</td>
</tr>
<tr>
<td>SCons.Executor.execute_action_list (function)</td>
<td>84</td>
</tr>
<tr>
<td>SCons.Executor.execute_actions_str (function)</td>
<td>84</td>
</tr>
<tr>
<td>SCons.Executor.execute_nothing (function)</td>
<td>84</td>
</tr>
<tr>
<td>SCons.Executor.execute_null_str (function)</td>
<td>84</td>
</tr>
<tr>
<td>SCons.Executor.Executor (class)</td>
<td>88–92</td>
</tr>
<tr>
<td>SCons.Executor.get_NullEnvironment (function)</td>
<td>84</td>
</tr>
<tr>
<td>SCons.Executor.GetBatchExecutor (function)</td>
<td>84</td>
</tr>
<tr>
<td>SCons.Executor.Null (class)</td>
<td>93–95</td>
</tr>
<tr>
<td>SCons.Executor.NullEnvironment (class)</td>
<td>92–93</td>
</tr>
<tr>
<td>SCons.Executor.rfile (function)</td>
<td>84</td>
</tr>
<tr>
<td>SCons.Executor.TSList (class)</td>
<td>85–87</td>
</tr>
<tr>
<td>SCons.Executor.TSObject (class)</td>
<td>87–</td>
</tr>
<tr>
<td>SCons.Executor.TSObject (class)</td>
<td>87–</td>
</tr>
<tr>
<td>SCons.Error (module)</td>
<td>398</td>
</tr>
<tr>
<td>SCons.exitfuncs (module)</td>
<td>398</td>
</tr>
<tr>
<td>SCons.exitfuncs.register (function)</td>
<td>398</td>
</tr>
<tr>
<td>SCons.Job (module)</td>
<td>96–103</td>
</tr>
<tr>
<td>SCons.Job.InterruptState (class)</td>
<td>96–97</td>
</tr>
<tr>
<td>SCons.Job.Parallel (class)</td>
<td>102–103</td>
</tr>
<tr>
<td>SCons.Job.Serial (class)</td>
<td>98–99</td>
</tr>
<tr>
<td>SCons.Job.ThreadPool (class)</td>
<td>101–102</td>
</tr>
<tr>
<td>SCons.Job.Worker (class)</td>
<td>99–101</td>
</tr>
<tr>
<td>SCons.Memoize (module)</td>
<td>104–109</td>
</tr>
<tr>
<td>SCons.Memoize.CountDictCall (function)</td>
<td>105</td>
</tr>
<tr>
<td>SCons.Memoize.Counter (class)</td>
<td>106–107</td>
</tr>
<tr>
<td>SCons.Memoize.CountMethodCall (function)</td>
<td>105</td>
</tr>
<tr>
<td>SCons.Memoize.CountValue (class)</td>
<td>107–108</td>
</tr>
<tr>
<td>SCons.Memoize.Dump (function)</td>
<td>105</td>
</tr>
<tr>
<td>SCons.Memoize.EnableMemoization (function)</td>
<td>105</td>
</tr>
<tr>
<td>SCons.Node (package)</td>
<td>110–132</td>
</tr>
<tr>
<td>SCons.Node.Alias (module)</td>
<td>133–138</td>
</tr>
<tr>
<td>SCons.Node.FS (module)</td>
<td>139–185</td>
</tr>
<tr>
<td>SCons.Node.Python (module)</td>
<td>186–191</td>
</tr>
<tr>
<td>SCons.PathList (module)</td>
<td>192</td>
</tr>
<tr>
<td>SCons.PathList.node_conv (function)</td>
<td>192</td>
</tr>
<tr>
<td>SCons.Scanner (module)</td>
<td>216–231</td>
</tr>
<tr>
<td>SCons.Scanner.Base (class)</td>
<td>217–220</td>
</tr>
<tr>
<td>SCons.Scanner.Classic (class)</td>
<td>227–230</td>
</tr>
<tr>
<td>SCons.Scanner.ClassicCPP (class)</td>
<td>230–231</td>
</tr>
<tr>
<td>SCons.Scanner.Current (class)</td>
<td>224–227</td>
</tr>
<tr>
<td>SCons.Scanner.FindPathDirs (class)</td>
<td>217</td>
</tr>
<tr>
<td>SCons.Scanner.Scanner (function)</td>
<td>216</td>
</tr>
<tr>
<td>SCons.Scanner.Selector (class)</td>
<td>220–224</td>
</tr>
<tr>
<td>SCons.Scanner.Scanner (package)</td>
<td>216–231</td>
</tr>
<tr>
<td>SCons.Scanner.C (module)</td>
<td>232–234</td>
</tr>
<tr>
<td>SCons.Scanner.D (module)</td>
<td>235–238</td>
</tr>
<tr>
<td>SCons.Scanner.Dir (module)</td>
<td>239–240</td>
</tr>
</tbody>
</table>
SCons.cpp.PreProcessor.do_undef (method), SCons.Defaults.SharedObjectEmitter (function), 391
SCons.cpp.PreProcessor.eval_expression (method), SCons.Defaults.StaticObjectEmitter (function), 391
SCons.cpp.PreProcessor.finalize_result (method), SCons.Defaults.touch_func (function), 42
SCons.cpp.PreProcessor.find__include___file (method), SCons.Defaults.Variable_Method_Caller (class), 392
SCons.cpp.PreProcessor.initialize_result (method), SCons.Defaults.Variable_Method_Caller.__call__, 392
SCons.cpp.PreProcessor.process_contents (method), SCons.Defaults.convert_to_BuildError (function), 76–78
SCons.cpp.PreProcessor.read__file (method), SCons.Defaults.EnvironmentError (class), 80–81
SCons.cpp.PreProcessor.resolve__include (function), SCons.Errors.ExplicitExit (class), 82–83
SCons.cpp.PreProcessor.restore (method), SCons.Errors.MSVCError (class), 81–82
SCons.cpp.PreProcessor.save (method), SCons.Errors.StopError (class), 79–80
SCons.cpp.PreProcessor.stop_handling__includes (method), SCons.Node.BuildInfoBase.__getstate__ (method), 115
SCons.Defaults.mkdir_func (function), SCons.Node.changed_since_last_build_alias (function), 112
SCons.Defaults.mkdir_strfunc (function), SCons.Node.changed_since_last_build_entry (function), 112
SCons.Defaults.copy_func (function), SCons.Node.changed_since_last_build_node (function), 111
SCons.Defaults.DefaultEnvironment (function), 41
SCons.Defaults.delete_func (function), SCons.Node.changed_since_last_build_python (function), 112
SCons.Defaults.delete_strfunc (function), SCons.Node.changed_since_last_build_state_changed (function), 112
SCons.Defaults.get_paths_str (function), SCons.Node.classname (function), 110
SCons.Defaults.mkdir_func (function), SCons.Node.decide_source (function), 112
SCons.Defaults.move_func (function), SCons.Node.decide_target (function), 112
SCons.Defaults.mkdir_strfunc (function), SCons.Node.do_nothing (function), 112
SCons.Defaults.processDefines (function), SCons.Node.exists_always (function), 110
SCons.Defaults.SharedFlagChecker (function), SCons.Node.exists_base (function), 111
SCons.Defaults.V ariable_Method_Caller (class), SCons.Node.exists_entry (function), 111

SCons.Node.exists_file (function), 111
SCons.Node.exists_none (function), 110
SCons.Node.get_children (function), 112
SCons.Node.get_contents_dir (function), 111
SCons.Node.get_contents_entry (function), 111
SCons.Node.get_contents_file (function), 111
SCons.Node.get_contents_none (function), 111
SCons.Node.ignore_cycle (function), 112
SCons.Node.is_derived_node (function), 110
SCons.Node.is_derived_none (function), 110
SCons.Node.Node (class), 116–130
    SCons.Node.Node.add_ignore (method), 117
    SCons.Node.Node.add_prerequisite (method), 117
    SCons.Node.Node.add_toImplicit (method), 117
    SCons.Node.Node.add_to_waiting_parents (method), 117
    SCons.Node.Node.add_to_waiting_s_e (method), 118
    SCons.Node.Node.add_wkid (method), 118
    SCons.Node.Node.all_children (method), 118
    SCons.Node.Node.alter_targets (method), 118
    SCons.Node.Node.build (method), 118
    SCons.Node.Node.builder_set (method), 118
    SCons.Node.Node.built (method), 118
    SCons.Node.Node.children_are_up_to_date (method), 119
    SCons.Node.Node.clear (method), 119
    SCons.Node.Node.clear_memoized_values (method), 120
    SCons.Node.Node.del_binfo (method), 120
    SCons.Node.Node.disambiguate (method), 120
    SCons.Node.Node.env_set (method), 120
    SCons.Node.Node.executor_cleanup (method), 120
    SCons.Node.Node.exists (method), 120
    SCons.Node.Node.for_signature (method), 120
    SCons.Node.Node.get_abspath (method), 120
    SCons.Node.Node.get_binfo (method), 121
    SCons.Node.Node.get_build_env (method), 121
    SCons.Node.Node.get_build_scanner_path (method), 121
    SCons.Node.Node.get_builder (method), 121
    SCons.Node.Node.get_cachedir_csSig (method), 121
    SCons.Node.Node.get_contents (method), 121
    SCons.Node.Node.get_contents_dir (method), 121
    SCons.Node.Node.get_contents_entry (method), 121
    SCons.Node.Node.get_contents_file (method), 121
    SCons.Node.Node.get_contents_none (method), 121
    SCons.Node.Node.get_csSig (method), 121
    SCons.Node.Node.get_env (method), 121
    SCons.Node.Node.get_env_scanner (method), 121
    SCons.Node.Node.get_executor (method), 122
    SCons.Node.Node.get_found_includes (method), 122
    SCons.Node.Node.get_implicit_deps (method), 122
    SCons.Node.Node.get_ninfo (method), 122
    SCons.Node.Node.get_source_scanner (method), 122
    SCons.Node.Node.get_state (method), 122
    SCons.Node.Node.get_stored_implicit (method), 123
    SCons.Node.Node.get_stored_info (method), 123
    SCons.Node.Node.get_subst_proxy (method), 123
    SCons.Node.Node.get_suffix (method), 123
    SCons.Node.Node.get_target_scanner (method), 123

419
SCons.Script.Interactive.SConsInteractiveCmd
(class), 266–268
SCons.Script.Interactive.SConsInteractiveCmd
.do_build (method), 267
SCons.Script.Interactive.SConsInteractiveCmd
.do_clean (method), 267
SCons.Script.Interactive.SConsInteractiveCmd
.do_EOF (method), 267
SCons.Script.Interactive.SConsInteractiveCmd
.do_exit (method), 267
SCons.Script.Interactive.SConsInteractiveCmd
.do_shell (method), 268
SCons.Script.Interactive.SConsInteractiveCmd
.do_version (method), 268
SCons.Subst.CmdStringHolder
(class), 294–295
SCons.Subst.CmdStringHolder.escape (method), 294
SCons.Subst.CmdStringHolder.is_literal (method), 294
SCons.Subst.escape_list (function), 290
SCons.Subst.Literal
(class), 292–293
SCons.Subst.Literal.__eq__ (method), 292
SCons.Subst.Literal.__neq__ (method), 292
SCons.Subst.Literal.escape (method), 292
SCons.Subst.Literal.for_signature (method), 292
SCons.Subst.Literal.is_literal (method), 292
SCons.Subst.NLW rapper
(class), 295–296
SCons.Subst.NullNodeList
(class), 299–300
SCons.Subst.quote_spaces (function), 290
SCons.Subst.raise_exception (function), 290
SCons.Subst.scons_subst (function), 290
SCons.Subst.scons_subst_list (function), 291
SCons.Subst.scons_subst_once (function), 291
SCons.Subst.SetAllowableExceptions (function), 291
SCons.Subst.SpecialAttrW rapper
(class), 293–294
SCons.Subst.SpecialAttrW rapper.escape (method), 293
SCons.Subst.SpecialAttrW rapper.for_signature (method), 293
SCons.Subst.SpecialAttrW rapper.is_literal (method), 293
SCons.Subst.subst_dict (function), 290
SCons.Subst.Target_or_Source
(class), 298–299
SCons.Subst.Target_or_Source.__getattr__ (method), 299
SCons.Subst.Targets_or_Sources (class), 296–297
SCons.Subst.Targets_or_Sources.__getattr__ (method), 297
SCons.Subst.CL Var
(class), 327–329
SCons.Subst.CL Var.__coerce__ (method), 328
SCons.Subst.containsAll (function), 312
SCons.Subst.containsAny (function), 312
SCons.Subst.containsOnly (function), 312
SCons.Subst.Delegate
(class), 324–325
SCons.Subst.Delegate.__get__ (method), 324
SCons.Subst.DisplayEngine
(class), 322
SCons.Subst.DisplayEngine.__call__ (method), 322
SCons.Subst.DisplayEngine.set_mode (method), 322
SCons.Subst.do_flatten (function), 314
SCons.Subst.flatten (function), 314
SCons.Subst.flatten_sequence (function), 314
SCons.Subst.GetAllowableExceptions (function), 312
SCons.Subst.get_environment_var (function), 312
SCons.Subst.get_native_path (function), 317
SCons.Subst.IDX (function), 313
SCons.Subst.is_Dict (function), 313
SCons.Subst.is_List (function), 313
SCons.Subst.is_Scalar (function), 314
SCons.Subst.is_Sequence (function), 314
SCons.Util._NoError
(class), 325
SCons.Util.AddMethod (function), 318
SCons.Util.AppendPath (function), 316
SCons.Util.ClVar
(class), 327–329
SCons.Util.ClVar.__coerce__ (method), 328
SCons.Util.adjustixes (function), 317
SCons.Util.appendPathIfNotExists (function), 316
SCons.Util.containsAll (function), 312
SCons.Util.containsAny (function), 312
SCons.Util.containsOnly (function), 312
SCons.Util.Delegate
(class), 324–325
SCons.Util.Delegate.__get__ (method), 324
SCons.Util.Dictify (function), 312
SCons.Util.DisplayEngine
(class), 322
SCons.Util.DisplayEngine.__call__ (method), 322
SCons.Util.DisplayEngine.set_mode (method), 322
SCons.Util.flatten (function), 314
SCons.Util.flatten_sequence (function), 314
SCons.Util.GetAllowableExceptions (function), 312
SCons.Util.get_environment_var (function), 312
SCons.Util.get_native_path (function), 317
SCons.Util.IDX (function), 313
SCons.Util.is_Dict (function), 313
SCons.Util.is_List (function), 313
SCons.Util.is_Scalar (function), 314
SCons.Util.is_Sequence (function), 314
SCons.Util.is_String (function), 314
SCons.Util.is_Tuple (function), 314
SCons.Util.logical_lines (function), 318
SCons.Util.LogicalLines (class), 331–332
SCons.Util.LogicalLines.readlines (method), 332
SCons.Util.make_path_relative (function), 318
SCons.Util.MD5collect (function), 318
SCons.Util.MD5filesignature (function), 318
SCons.Util.MD5signature (function), 318
SCons.Util.NodeList (class), 320–322
SCons.Util.NodeList.__bool__ (method), 320
SCons.Util.NodeList.__call__ (method), 321
SCons.Util.NodeList.__getattr__ (method), 321
SCons.Util.NodeList.__nonzero__ (method), 320
SCons.Util.Null (class), 337–338
SCons.Util.Null.__bool__ (method), 338
SCons.Util.Null.__call__ (method), 338
SCons.Util.Null.__getattr__ (method), 338
SCons.Util.Null.__nonzero__ (method), 338
SCons.Util.NullSeq (class), 338–339
SCons.Util.NullSeq.__delitem__ (method), 339
SCons.Util.NullSeq.__getitem__ (method), 339
SCons.Util.NullSeq.__iter__ (method), 339
SCons.Util.NullSeq.__len__ (method), 339
SCons.Util.NullSeq.__setitem__ (method), 339
SCons.Util.OrderedDict (class), 329–331
SCons.Util.PlainWindowsError (class), 325–327
SCons.Util.PrependPath (function), 315
SCons.Util.print_tree (function), 313
SCons.Util.Proxy (class), 322–324
SCons.Util.Proxy.__eq__ (method), 324
SCons.Util.Proxy.__getattribute__ (method), 323
SCons.Warnings.DeprecatedDebugOptionsWarning (class), 384–385
SCons.Warnings.DeprecatedOptionsWarning (class), 381–382
SCons.Warnings.DeprecatedSigModuleWarning (class), 385–386
SCons.Warnings.DeprecatedSourceCodeWarning (class), 377–378
SCons.Warnings.DeprecatedSourceSignaturesWarning (class), 382–383
SCons.Warnings.DeprecatedTargetSignaturesWarning (class), 383–384
SCons.Warnings.DeprecatedWarning (class), 374–375
SCons.Warnings.DevelopmentVersionWarning (class), 358
SCons.Warnings.DuplicateEnvironmentWarning (class), 358–359
SCons.Warnings.FortranCxxMixWarning (class), 372–373
SCons.Warnings.FutureDeprecatedWarning (class), 373–374
SCons.Warnings.FutureReservedVariableWarning (class), 359–360
SCons.Warnings.LinkWarning (class), 360–361
SCons.Warnings.MandatoryDeprecatedWarning (class), 375–376
SCons.Warnings.MisleadingKeywordsWarning (class), 361–362
SCons.Warnings.MissingSConscriptWarning (class), 362–363
SCons.Warnings.NoMD5ModuleWarning (class), 363–364
SCons.Warnings.NoMetaclassSupportWarning (class), 364–365
SCons.Warnings.NoObjectCountWarning (class), 365–366
SCons.Warnings.NoParallelSupportWarning (class), 366–367
SCons.Warnings.process_warn_strings (function), 350
SCons.Warnings.PythonVersionWarning (class),
SCons.Warnings.ReservedVariableWarning (class), 367–368
SCons.Warnings.StackTraceWarning (class), 368–369
SCons.Warnings.suppressWarningClass (function), 350
SCons.Warnings.TargetNotBuiltWarning (class),
SCons.Warnings.TaskmasterNeedsExecuteWarning
SCons.Warnings.VisualCMissingWarning (class), 369–370
SCons.Warnings.VisualStudioMissingWarning (class), 371–372
SCons.Warnings.VisualVersionMismatch (class),
SCons.Warnings.warn (function), 350
SCons.Warnings.warningAsException (function), 350
SCons.Warnings.WarningOnByDefault (class), 352–353
423