SCons

API Documentation

March 2, 2014

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 ............................................ 11
      2.6.2 Properties .......................................... 12
   2.7 Class FunctionAction ..................................... 13
      2.7.1 Methods ............................................ 13
      2.7.2 Properties .......................................... 14
   2.8 Class ListAction .......................................... 14
      2.8.1 Methods ............................................ 14
      2.8.2 Properties .......................................... 15
   2.9 Class ActionCaller ........................................ 15
      2.9.1 Methods ............................................ 16
      2.9.2 Properties .......................................... 16
   2.10 Class ActionFactory ..................................... 17
      2.10.1 Methods ........................................... 17
      2.10.2 Properties ......................................... 17

3 Module SCons.Builder ........................................ 18
   3.1 Functions .............................................. 19
   3.2 Variables ............................................... 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Subsections</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 Class DictCmdGenerator</td>
<td>3.3.1 Methods, 3.3.2 Class Variables</td>
<td>20</td>
</tr>
<tr>
<td>3.4 Class CallableSelector</td>
<td>3.4.1 Methods, 3.4.2 Class Variables</td>
<td>21</td>
</tr>
<tr>
<td>3.5 Class DictEmitter</td>
<td>3.5.1 Methods, 3.5.2 Class Variables</td>
<td>22</td>
</tr>
<tr>
<td>3.6 Class ListEmitter</td>
<td>3.6.1 Methods, 3.6.2 Properties, 3.6.3 Class Variables</td>
<td>23</td>
</tr>
<tr>
<td>3.7 Class OverrideWarner</td>
<td>3.7.1 Methods, 3.7.2 Class Variables</td>
<td>24</td>
</tr>
<tr>
<td>3.8 Class EmitterProxy</td>
<td>3.8.1 Methods, 3.8.2 Properties</td>
<td>25</td>
</tr>
<tr>
<td>3.9 Class BuilderBase</td>
<td>3.9.1 Methods, 3.9.2 Properties, 3.9.3 Class Variables</td>
<td>26</td>
</tr>
<tr>
<td>3.10 Class CompositeBuilder</td>
<td>3.10.1 Methods, 3.10.2 Properties</td>
<td>29</td>
</tr>
<tr>
<td>4 Module SCons.CacheDir</td>
<td>4.1 Functions, 4.2 Variables, 4.3 Class CacheDir</td>
<td>30</td>
</tr>
<tr>
<td>5 Module SCons.Conftest</td>
<td>5.1 Functions, 5.2 Variables</td>
<td>33</td>
</tr>
<tr>
<td>6 Module SCons.Debug</td>
<td>6.1 Functions, 6.2 Variables</td>
<td>38</td>
</tr>
<tr>
<td>7 Module SCons.Defaults</td>
<td>7.1 Functions, 7.2 Variables, 7.3 Class NullCmdGenerator, 7.3.1 Methods, 7.3.2 Properties, 7.4 Class Variable_Method_Caller, 7.4.1 Methods, 7.4.2 Properties</td>
<td>40</td>
</tr>
</tbody>
</table>

2
## Contents

### Module SCons.Environment

8.1 Functions ............................................. 45
8.2 Variables ............................................. 45
8.3 Class MethodWrapper ................................. 46
  8.3.1 Methods ........................................... 46
  8.3.2 Properties ........................................ 47
8.4 Class BuilderWrapper ................................. 47
  8.4.1 Methods ........................................... 47
  8.4.2 Properties ........................................ 48
8.5 Class BuilderDict ..................................... 48
  8.5.1 Methods ........................................... 48
  8.5.2 Class Variables ................................... 49
8.6 Class SubstitutionEnvironment ....................... 49
  8.6.1 Methods ........................................... 50
  8.6.2 Properties ........................................ 53
  8.6.3 Class Variables ................................... 53
8.7 Class Base ............................................ 53
  8.7.1 Methods ........................................... 53
  8.7.2 Properties ........................................ 61
  8.7.3 Class Variables ................................... 61
8.8 Class OverrideEnvironment ........................... 61
  8.8.1 Methods ........................................... 62
  8.8.2 Properties ........................................ 64
  8.8.3 Class Variables ................................... 64
8.9 Class Base ............................................ 65
  8.9.1 Methods ........................................... 65
  8.9.2 Properties ........................................ 73
  8.9.3 Class Variables ................................... 73

### Module SCons.Errors

9.1 Functions ............................................. 74
9.2 Variables ............................................. 74
9.3 Class BuildError ...................................... 74
  9.3.1 Methods ........................................... 76
  9.3.2 Properties ........................................ 76
9.4 Class InternalError ................................. 76
  9.4.1 Methods ........................................... 77
  9.4.2 Properties ........................................ 77
9.5 Class UserError ...................................... 77
  9.5.1 Methods ........................................... 77
  9.5.2 Properties ........................................ 78
9.6 Class StopError ...................................... 78
  9.6.1 Methods ........................................... 78
  9.6.2 Properties ........................................ 78
9.7 Class EnvironmentError .............................. 79
  9.7.1 Methods ........................................... 79
  9.7.2 Properties ........................................ 79
9.8 Class MSVCError ...................................... 80
  9.8.1 Methods ........................................... 80
  9.8.2 Properties ........................................ 80
9.9 Class ExplicitExit ................................... 81
  9.9.1 Methods ........................................... 81
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Module SCons.Executor</td>
<td>82</td>
</tr>
<tr>
<td>10.1 Functions</td>
<td>82</td>
</tr>
<tr>
<td>10.2 Variables</td>
<td>82</td>
</tr>
<tr>
<td>10.3 Class Batch</td>
<td>83</td>
</tr>
<tr>
<td>10.3.1 Methods</td>
<td>83</td>
</tr>
<tr>
<td>10.3.2 Properties</td>
<td>83</td>
</tr>
<tr>
<td>10.4 Class TSLlist</td>
<td>83</td>
</tr>
<tr>
<td>10.4.1 Methods</td>
<td>84</td>
</tr>
<tr>
<td>10.4.2 Properties</td>
<td>85</td>
</tr>
<tr>
<td>10.4.3 Class Variables</td>
<td>85</td>
</tr>
<tr>
<td>10.5 Class TSOBJECT</td>
<td>85</td>
</tr>
<tr>
<td>10.5.1 Methods</td>
<td>85</td>
</tr>
<tr>
<td>10.5.2 Properties</td>
<td>86</td>
</tr>
<tr>
<td>10.6 Class Executor</td>
<td>86</td>
</tr>
<tr>
<td>10.6.1 Methods</td>
<td>86</td>
</tr>
<tr>
<td>10.6.2 Properties</td>
<td>90</td>
</tr>
<tr>
<td>10.6.3 Class Variables</td>
<td>90</td>
</tr>
<tr>
<td>10.6.4 Instance Variables</td>
<td>90</td>
</tr>
<tr>
<td>10.7 Class Null</td>
<td>90</td>
</tr>
<tr>
<td>10.7.1 Methods</td>
<td>90</td>
</tr>
<tr>
<td>10.7.2 Properties</td>
<td>91</td>
</tr>
<tr>
<td>11 Module SCons.Job</td>
<td>93</td>
</tr>
<tr>
<td>11.1 Variables</td>
<td>93</td>
</tr>
<tr>
<td>11.2 Class InterruptState</td>
<td>93</td>
</tr>
<tr>
<td>11.2.1 Methods</td>
<td>93</td>
</tr>
<tr>
<td>11.2.2 Properties</td>
<td>94</td>
</tr>
<tr>
<td>11.3 Class Jobs</td>
<td>94</td>
</tr>
<tr>
<td>11.3.1 Methods</td>
<td>94</td>
</tr>
<tr>
<td>11.3.2 Properties</td>
<td>95</td>
</tr>
<tr>
<td>11.4 Class Serial</td>
<td>95</td>
</tr>
<tr>
<td>11.4.1 Methods</td>
<td>95</td>
</tr>
<tr>
<td>11.4.2 Properties</td>
<td>96</td>
</tr>
<tr>
<td>11.5 Class Worker</td>
<td>96</td>
</tr>
<tr>
<td>11.5.1 Methods</td>
<td>96</td>
</tr>
<tr>
<td>11.5.2 Properties</td>
<td>97</td>
</tr>
<tr>
<td>11.6 Class ThreadPool</td>
<td>97</td>
</tr>
<tr>
<td>11.6.1 Methods</td>
<td>97</td>
</tr>
<tr>
<td>11.6.2 Properties</td>
<td>98</td>
</tr>
<tr>
<td>11.7 Class Parallel</td>
<td>98</td>
</tr>
<tr>
<td>11.7.1 Methods</td>
<td>99</td>
</tr>
<tr>
<td>11.7.2 Properties</td>
<td>99</td>
</tr>
<tr>
<td>12 Module SCons.Memoize</td>
<td>100</td>
</tr>
<tr>
<td>12.1 Functions</td>
<td>102</td>
</tr>
<tr>
<td>12.2 Variables</td>
<td>102</td>
</tr>
<tr>
<td>12.3 Class Counter</td>
<td>102</td>
</tr>
<tr>
<td>12.3.1 Methods</td>
<td>103</td>
</tr>
<tr>
<td>12.3.2 Properties</td>
<td>103</td>
</tr>
<tr>
<td>12.4 Class CountValue</td>
<td>103</td>
</tr>
</tbody>
</table>
12.4.1 Methods .......................................................... 104
12.4.2 Properties ....................................................... 104
12.5 Class CountDict .................................................. 104
12.5.1 Methods ....................................................... 105
12.5.2 Properties ..................................................... 105
12.6 Class Memoizer .................................................... 105
12.6.1 Methods ....................................................... 105
12.6.2 Properties ..................................................... 106
12.7 Class Memoized_Metaclass .................................... 106
12.7.1 Methods ....................................................... 106
12.7.2 Properties ..................................................... 106

13 Package SCons.Node .................................................. 108
13.1 Modules ............................................................ 108
13.2 Functions .......................................................... 108
13.3 Variables .......................................................... 108
13.4 Class NodeInfoBase .............................................. 109
13.4.1 Methods ....................................................... 109
13.4.2 Properties ..................................................... 110
13.4.3 Class Variables .............................................. 110
13.5 Class BuildInfoBase ............................................. 110
13.5.1 Methods ....................................................... 111
13.5.2 Properties ..................................................... 111
13.5.3 Class Variables .............................................. 111
13.6 Class Node .......................................................... 111
13.6.1 Methods ....................................................... 111
13.6.2 Properties ..................................................... 125
13.6.3 Class Variables .............................................. 125
13.7 Class NodeList ..................................................... 125
13.7.1 Methods ....................................................... 125
13.7.2 Properties ..................................................... 126
13.7.3 Class Variables .............................................. 126
13.8 Class Walker ...................................................... 126
13.8.1 Methods ....................................................... 127
13.8.2 Properties ..................................................... 127

14 Module SCons.Node.Alias ............................................ 128
14.1 Variables .......................................................... 128
14.2 Class AliasNameSpace .......................................... 128
14.2.1 Methods ....................................................... 128
14.2.2 Class Variables .............................................. 128
14.3 Class AliasNodeInfo ............................................ 129
14.3.1 Methods ....................................................... 129
14.3.2 Properties ..................................................... 129
14.3.3 Class Variables .............................................. 129
14.4 Class AliasBuildInfo ........................................... 130
14.4.1 Methods ....................................................... 130
14.4.2 Properties ..................................................... 130
14.4.3 Class Variables .............................................. 130
14.5 Class Alias ........................................................ 131
14.5.1 Methods ....................................................... 131
14.5.2 Properties ..................................................... 133
15 Module SCons.Node.FS

15.1 Functions ................................................. 135
15.2 Variables ................................................. 137
15.3 Class EntryProxyAttributeError ......................... 138
  15.3.1 Methods .............................................. 138
  15.3.2 Properties ........................................... 139
15.4 Class DiskChecker ......................................... 139
  15.4.1 Methods .............................................. 139
  15.4.2 Properties ........................................... 140
15.5 Class EntryProxy .......................................... 140
  15.5.1 Methods .............................................. 140
  15.5.2 Properties ........................................... 140
  15.5.3 Class Variables ..................................... 141
15.6 Class Base ................................................ 141
  15.6.1 Methods .............................................. 141
  15.6.2 Properties ........................................... 145
  15.6.3 Class Variables ..................................... 145
  15.6.4 Instance Variables ................................. 146
15.7 Class Entry ............................................... 146
  15.7.1 Methods .............................................. 146
  15.7.2 Properties ........................................... 149
  15.7.3 Class Variables ..................................... 149
  15.7.4 Instance Variables ................................. 149
15.8 Class LocalFS ............................................. 149
  15.8.1 Methods .............................................. 150
  15.8.2 Properties ........................................... 151
  15.8.3 Class Variables ..................................... 151
15.9 Class FS .................................................. 151
  15.9.1 Methods .............................................. 151
  15.9.2 Properties ........................................... 154
  15.9.3 Class Variables ..................................... 154
15.10 Class DirNodeInfo ........................................ 154
  15.10.1 Methods ............................................. 154
  15.10.2 Properties ........................................... 155
  15.10.3 Class Variables ..................................... 155
15.11 Class DirBuildInfo ....................................... 155
  15.11.1 Methods .............................................. 155
  15.11.2 Properties ........................................... 156
  15.11.3 Class Variables ..................................... 156
15.12 Class Dir ................................................ 156
  15.12.1 Methods .............................................. 156
  15.12.2 Properties ........................................... 163
  15.12.3 Class Variables ..................................... 163
  15.12.4 Instance Variables ................................. 163
15.13 Class RootDir ............................................ 163
  15.13.1 Methods .............................................. 164
  15.13.2 Properties ........................................... 166
  15.13.3 Class Variables ..................................... 166
  15.13.4 Instance Variables ................................. 166
15.14 Class FileNodeInfo ....................................... 166
## 15.14 Class FileBuildInfo

- **Methods**: 167
- **Properties**: 168
- **Class Variables**: 169

## 15.15 Class File

- **Methods**: 168
- **Properties**: 169
- **Class Variables**: 169

## 15.16 Class FileFinder

- **Methods**: 169
- **Properties**: 177
- **Class Variables**: 169
- **Instance Variables**: 177

## 15.17 Class FileFinder

- **Methods**: 169
- **Properties**: 169
- **Class Variables**: 169

## 16 Module SCons.Node.Python

- **Variables**: 180

## 17 Module SCons.PathList

- **Functions**: 186
- **Variables**: 186

## 18 Module SCons.SConf

- **Functions**: 187
- **Variables**: 188

### 18.2 Class SConfWarning

- **Methods**: 189
- **Properties**: 190

### 18.3 Class SConfError

- **Methods**: 190
- **Properties**: 191

### 18.4 Class ConfigureDryRunError

- **Methods**: 191
- **Properties**: 192

### 18.5 Class ConfigureCacheError

- **Methods**: 192
- **Properties**: 193

### 18.6 Class SConfBuildInfo

- **Methods**: 193
- **Properties**: 194
# CONTENTS

## 18.7 Methods
- 18.7.1 Class Variables ........................................ 193
- 18.7.2 Properties .................................................. 194
- 18.7.3 Class Base .................................................. 194

## 18.8 Class SCons.Scanner
- 18.8.1 Class Methods ............................................ 194
- 18.8.2 Properties .................................................. 195

## 18.9 Class SConsignEntry
- 18.9.1 Class Methods ............................................ 195
- 18.9.2 Properties .................................................. 196

## 18.10 Class SConfBase
- 18.10.1 Class Methods ........................................... 197
- 18.10.2 Properties .................................................. 199

## 18.11 Class CheckContext
- 18.11.1 Class Methods ........................................... 200
- 18.11.2 Properties .................................................. 201

## 19 Module SCons.SConsign
- 19.1 Functions ..................................................... 202
- 19.2 Variables ...................................................... 202
- 19.3 Class SConsignEntry
  - 19.3.1 Class Methods ........................................... 203
  - 19.3.2 Properties .................................................. 203
  - 19.3.3 Class Variables .......................................... 203
- 19.4 Class Base
  - 19.4.1 Class Methods ........................................... 204
  - 19.4.2 Properties .................................................. 205
- 19.5 Class DB
  - 19.5.1 Class Methods ........................................... 205
  - 19.5.2 Properties .................................................. 205
- 19.6 Class Dir
  - 19.6.1 Class Methods ........................................... 206
  - 19.6.2 Properties .................................................. 206
- 19.7 Class DirFile
  - 19.7.1 Class Methods ........................................... 207
  - 19.7.2 Properties .................................................. 207
- 19.8 Class DB
  - 19.8.1 Class Methods ........................................... 208
  - 19.8.2 Properties .................................................. 208

## 20 Package SCons.Scanner
- 20.1 Modules ....................................................... 209
- 20.2 Functions ..................................................... 209
- 20.3 Variables ...................................................... 209
- 20.4 Class FindPathDirs
  - 20.4.1 Class Methods ........................................... 210
  - 20.4.2 Properties .................................................. 210
- 20.5 Class Base
  - 20.5.1 Class Methods ........................................... 210
  - 20.5.2 Properties .................................................. 213
- 20.6 Class Selector
  - 20.6.1 Class Methods ........................................... 216
  - 20.6.2 Properties .................................................. 217
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.7</td>
<td>Class Current</td>
<td>217</td>
</tr>
<tr>
<td>20.7.1</td>
<td>Methods</td>
<td>219</td>
</tr>
<tr>
<td>20.7.2</td>
<td>Properties</td>
<td>220</td>
</tr>
<tr>
<td>20.8</td>
<td>Class Classic</td>
<td>220</td>
</tr>
<tr>
<td>20.8.1</td>
<td>Methods</td>
<td>222</td>
</tr>
<tr>
<td>20.8.2</td>
<td>Properties</td>
<td>223</td>
</tr>
<tr>
<td>20.9</td>
<td>Class ClassicCPP</td>
<td>223</td>
</tr>
<tr>
<td>20.9.1</td>
<td>Methods</td>
<td>224</td>
</tr>
<tr>
<td>20.9.2</td>
<td>Properties</td>
<td>225</td>
</tr>
<tr>
<td>21</td>
<td>Module SCons.Scanner.C</td>
<td>225</td>
</tr>
<tr>
<td>21.1</td>
<td>Functions</td>
<td>225</td>
</tr>
<tr>
<td>21.2</td>
<td>Variables</td>
<td>225</td>
</tr>
<tr>
<td>21.3</td>
<td>Class SConsCPPScanner</td>
<td>225</td>
</tr>
<tr>
<td>21.3.1</td>
<td>Methods</td>
<td>226</td>
</tr>
<tr>
<td>21.3.2</td>
<td>Properties</td>
<td>226</td>
</tr>
<tr>
<td>21.4</td>
<td>Class SConsCPPScannerWrapper</td>
<td>227</td>
</tr>
<tr>
<td>21.4.1</td>
<td>Methods</td>
<td>227</td>
</tr>
<tr>
<td>21.4.2</td>
<td>Properties</td>
<td>227</td>
</tr>
<tr>
<td>22</td>
<td>Module SCons.Scanner.D</td>
<td>228</td>
</tr>
<tr>
<td>22.1</td>
<td>Functions</td>
<td>228</td>
</tr>
<tr>
<td>22.2</td>
<td>Variables</td>
<td>228</td>
</tr>
<tr>
<td>22.3</td>
<td>Class D</td>
<td>228</td>
</tr>
<tr>
<td>22.3.1</td>
<td>Methods</td>
<td>230</td>
</tr>
<tr>
<td>22.3.2</td>
<td>Properties</td>
<td>231</td>
</tr>
<tr>
<td>23</td>
<td>Module SCons.Scanner.Dir</td>
<td>232</td>
</tr>
<tr>
<td>23.1</td>
<td>Functions</td>
<td>232</td>
</tr>
<tr>
<td>23.2</td>
<td>Variables</td>
<td>232</td>
</tr>
<tr>
<td>24</td>
<td>Module SCons.Scanner.Fortran</td>
<td>234</td>
</tr>
<tr>
<td>24.1</td>
<td>Functions</td>
<td>234</td>
</tr>
<tr>
<td>24.2</td>
<td>Variables</td>
<td>234</td>
</tr>
<tr>
<td>24.3</td>
<td>Class F90Scanner</td>
<td>234</td>
</tr>
<tr>
<td>24.3.1</td>
<td>Methods</td>
<td>237</td>
</tr>
<tr>
<td>24.3.2</td>
<td>Properties</td>
<td>238</td>
</tr>
<tr>
<td>25</td>
<td>Module SCons.Scanner.IDL</td>
<td>239</td>
</tr>
<tr>
<td>25.1</td>
<td>Functions</td>
<td>239</td>
</tr>
<tr>
<td>25.2</td>
<td>Variables</td>
<td>239</td>
</tr>
<tr>
<td>26</td>
<td>Module SCons.Scanner.LaTeX</td>
<td>240</td>
</tr>
<tr>
<td>26.1</td>
<td>Functions</td>
<td>240</td>
</tr>
<tr>
<td>26.2</td>
<td>Variables</td>
<td>240</td>
</tr>
<tr>
<td>26.3</td>
<td>Class FindENVPathDirs</td>
<td>240</td>
</tr>
<tr>
<td>26.3.1</td>
<td>Methods</td>
<td>241</td>
</tr>
<tr>
<td>26.3.2</td>
<td>Properties</td>
<td>241</td>
</tr>
<tr>
<td>26.4</td>
<td>Class LaTeX</td>
<td>241</td>
</tr>
<tr>
<td>26.4.1</td>
<td>Methods</td>
<td>244</td>
</tr>
<tr>
<td>26.4.2</td>
<td>Properties</td>
<td>245</td>
</tr>
<tr>
<td>26.4.3</td>
<td>Class Variables</td>
<td>245</td>
</tr>
</tbody>
</table>
## Modules

### Module SCons.Scanner.Prog

- 27.1 Functions
- 27.2 Variables

### Module SCons.Scanner.RC

- 28.1 Functions
- 28.2 Variables

### Package SCons.Script

- 29.1 Modules
- 29.2 Functions
- 29.3 Variables
- 29.4 Class TargetList
  - 29.4.1 Methods
  - 29.4.2 Properties
  - 29.4.3 Class Variables

### Module SCons.Script.Interactive

- 30.1 Functions
- 30.2 Variables
- 30.3 Class SConsInteractiveCmd
  - 30.3.1 Methods
  - 30.3.2 Class Variables

### Module SCons.Script.Main

- 31.1 Functions
- 31.2 Variables
- 31.3 Class SConsPrintHelpException
  - 31.3.1 Methods
  - 31.3.2 Properties
- 31.4 Class Progressor
  - 31.4.1 Methods
  - 31.4.2 Properties
  - 31.4.3 Class Variables
- 31.5 Class BuildTask
  - 31.5.1 Methods
  - 31.5.2 Properties
  - 31.5.3 Class Variables
- 31.6 Class CleanTask
  - 31.6.1 Methods
  - 31.6.2 Properties
- 31.7 Class QuestionTask
  - 31.7.1 Methods
  - 31.7.2 Properties
- 31.8 Class TreePrinter
  - 31.8.1 Methods
  - 31.8.2 Properties
- 31.9 Class FakeOptionParser
  - 31.9.1 Methods
  - 31.9.2 Properties
  - 31.9.3 Class Variables
- 31.10 Class Stats
## Module SCons.Util

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.7.1</td>
<td>Methods</td>
<td>291</td>
</tr>
<tr>
<td>34.7.2</td>
<td>Properties</td>
<td>292</td>
</tr>
<tr>
<td>34.7.3</td>
<td>Class Variables</td>
<td>292</td>
</tr>
<tr>
<td>34.8</td>
<td>Class Target_or_Source</td>
<td>293</td>
</tr>
<tr>
<td>34.8.1</td>
<td>Methods</td>
<td>293</td>
</tr>
<tr>
<td>34.8.2</td>
<td>Properties</td>
<td>293</td>
</tr>
<tr>
<td>34.9</td>
<td>Class NodeList</td>
<td>294</td>
</tr>
<tr>
<td>34.9.1</td>
<td>Methods</td>
<td>294</td>
</tr>
<tr>
<td>34.9.2</td>
<td>Properties</td>
<td>294</td>
</tr>
</tbody>
</table>

## Module SCons.Taskmaster

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>35.1</td>
<td>Functions</td>
<td>295</td>
</tr>
<tr>
<td>35.2</td>
<td>Variables</td>
<td>295</td>
</tr>
<tr>
<td>35.3</td>
<td>Class Stats</td>
<td>296</td>
</tr>
<tr>
<td>35.3.1</td>
<td>Methods</td>
<td>296</td>
</tr>
<tr>
<td>35.3.2</td>
<td>Properties</td>
<td>296</td>
</tr>
<tr>
<td>35.4</td>
<td>Class Task</td>
<td>297</td>
</tr>
<tr>
<td>35.4.1</td>
<td>Methods</td>
<td>297</td>
</tr>
<tr>
<td>35.4.2</td>
<td>Properties</td>
<td>301</td>
</tr>
<tr>
<td>35.5</td>
<td>Class AlwaysTask</td>
<td>301</td>
</tr>
<tr>
<td>35.5.1</td>
<td>Methods</td>
<td>302</td>
</tr>
<tr>
<td>35.5.2</td>
<td>Properties</td>
<td>302</td>
</tr>
<tr>
<td>35.6</td>
<td>Class OutOfDateTask</td>
<td>302</td>
</tr>
<tr>
<td>35.6.1</td>
<td>Methods</td>
<td>303</td>
</tr>
<tr>
<td>35.6.2</td>
<td>Properties</td>
<td>303</td>
</tr>
<tr>
<td>35.7</td>
<td>Class Taskmaster</td>
<td>303</td>
</tr>
<tr>
<td>35.7.1</td>
<td>Methods</td>
<td>305</td>
</tr>
<tr>
<td>35.7.2</td>
<td>Properties</td>
<td>305</td>
</tr>
</tbody>
</table>

## Module SCons.Util

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.1</td>
<td>Functions</td>
<td>306</td>
</tr>
<tr>
<td>36.2</td>
<td>Variables</td>
<td>312</td>
</tr>
<tr>
<td>36.3</td>
<td>Class NodeList</td>
<td>314</td>
</tr>
<tr>
<td>36.3.1</td>
<td>Methods</td>
<td>314</td>
</tr>
<tr>
<td>36.3.2</td>
<td>Properties</td>
<td>315</td>
</tr>
<tr>
<td>36.3.3</td>
<td>Class Variables</td>
<td>315</td>
</tr>
<tr>
<td>36.4</td>
<td>Class DisplayEngine</td>
<td>315</td>
</tr>
<tr>
<td>36.4.1</td>
<td>Methods</td>
<td>316</td>
</tr>
<tr>
<td>36.4.2</td>
<td>Properties</td>
<td>316</td>
</tr>
<tr>
<td>36.4.3</td>
<td>Class Variables</td>
<td>316</td>
</tr>
<tr>
<td>36.5</td>
<td>Class Proxy</td>
<td>316</td>
</tr>
<tr>
<td>36.5.1</td>
<td>Methods</td>
<td>317</td>
</tr>
<tr>
<td>36.5.2</td>
<td>Properties</td>
<td>317</td>
</tr>
<tr>
<td>36.6</td>
<td>Class Delegate</td>
<td>318</td>
</tr>
<tr>
<td>36.6.1</td>
<td>Methods</td>
<td>318</td>
</tr>
<tr>
<td>36.6.2</td>
<td>Properties</td>
<td>318</td>
</tr>
<tr>
<td>36.7</td>
<td>Class _NoError</td>
<td>318</td>
</tr>
<tr>
<td>36.7.1</td>
<td>Methods</td>
<td>319</td>
</tr>
<tr>
<td>36.7.2</td>
<td>Properties</td>
<td>319</td>
</tr>
<tr>
<td>36.8</td>
<td>Class WindowsError</td>
<td>319</td>
</tr>
<tr>
<td>36.8.1</td>
<td>Methods</td>
<td>319</td>
</tr>
<tr>
<td>36.8.2</td>
<td>Properties</td>
<td>320</td>
</tr>
<tr>
<td>Section</td>
<td>Content</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>36.9</td>
<td>Class CLVar 321</td>
<td></td>
</tr>
<tr>
<td>36.9.1</td>
<td>Methods 321</td>
<td></td>
</tr>
<tr>
<td>36.9.2</td>
<td>Properties 322</td>
<td></td>
</tr>
<tr>
<td>36.9.3</td>
<td>Class Variables 322</td>
<td></td>
</tr>
<tr>
<td>36.10</td>
<td>Class OrderedDict 322</td>
<td></td>
</tr>
<tr>
<td>36.10.1</td>
<td>Methods 323</td>
<td></td>
</tr>
<tr>
<td>36.10.2</td>
<td>Class Variables 324</td>
<td></td>
</tr>
<tr>
<td>36.11</td>
<td>Class Selector 324</td>
<td></td>
</tr>
<tr>
<td>36.11.1</td>
<td>Methods 324</td>
<td></td>
</tr>
<tr>
<td>36.11.2</td>
<td>Class Variables 324</td>
<td></td>
</tr>
<tr>
<td>36.12</td>
<td>Class LogicalLines 324</td>
<td></td>
</tr>
<tr>
<td>36.12.1</td>
<td>Methods 325</td>
<td></td>
</tr>
<tr>
<td>36.12.2</td>
<td>Properties 325</td>
<td></td>
</tr>
<tr>
<td>36.13</td>
<td>Class UniqueList 326</td>
<td></td>
</tr>
<tr>
<td>36.13.1</td>
<td>Methods 326</td>
<td></td>
</tr>
<tr>
<td>36.13.2</td>
<td>Properties 329</td>
<td></td>
</tr>
<tr>
<td>36.13.3</td>
<td>Class Variables 329</td>
<td></td>
</tr>
<tr>
<td>36.14</td>
<td>Class Unbuffered 329</td>
<td></td>
</tr>
<tr>
<td>36.14.1</td>
<td>Methods 329</td>
<td></td>
</tr>
<tr>
<td>36.14.2</td>
<td>Properties 329</td>
<td></td>
</tr>
<tr>
<td>36.15</td>
<td>Class Null 330</td>
<td></td>
</tr>
<tr>
<td>36.15.1</td>
<td>Methods 330</td>
<td></td>
</tr>
<tr>
<td>36.15.2</td>
<td>Properties 331</td>
<td></td>
</tr>
<tr>
<td>36.16</td>
<td>Class NullSeq 331</td>
<td></td>
</tr>
<tr>
<td>36.16.1</td>
<td>Methods 331</td>
<td></td>
</tr>
<tr>
<td>36.16.2</td>
<td>Properties 332</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Package SCons.Variables 333</td>
<td></td>
</tr>
<tr>
<td>37.1</td>
<td>Modules 333</td>
<td></td>
</tr>
<tr>
<td>37.2</td>
<td>Variables 333</td>
<td></td>
</tr>
<tr>
<td>37.3</td>
<td>Class Variables 333</td>
<td></td>
</tr>
<tr>
<td>37.3.1</td>
<td>Methods 334</td>
<td></td>
</tr>
<tr>
<td>37.3.2</td>
<td>Properties 336</td>
<td></td>
</tr>
<tr>
<td>37.3.3</td>
<td>Class Variables 336</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Module SCons.Variables.BoolVariable’ 337</td>
<td></td>
</tr>
<tr>
<td>38.1</td>
<td>Functions 337</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Module SCons.Variables.EnumVariable’ 338</td>
<td></td>
</tr>
<tr>
<td>39.1</td>
<td>Functions 339</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Module SCons.Variables.ListVariable’ 340</td>
<td></td>
</tr>
<tr>
<td>40.1</td>
<td>Functions 340</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Module SCons.Variables.PackageVariable’ 341</td>
<td></td>
</tr>
<tr>
<td>41.1</td>
<td>Functions 341</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Module SCons.Variables.PathVariable’ 342</td>
<td></td>
</tr>
<tr>
<td>42.1</td>
<td>Variables 343</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Module SCons.Warnings 344</td>
<td></td>
</tr>
<tr>
<td>43.1</td>
<td>Functions 344</td>
<td></td>
</tr>
<tr>
<td>43.2</td>
<td>Variables 345</td>
<td></td>
</tr>
<tr>
<td>43.3 Class Warning</td>
<td>.................................</td>
<td>345</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------</td>
<td>----</td>
</tr>
<tr>
<td>43.3.1 Methods</td>
<td>.................................</td>
<td>346</td>
</tr>
<tr>
<td>43.3.2 Properties</td>
<td>.................................</td>
<td>346</td>
</tr>
<tr>
<td>43.4 Class WarningOnByDefault</td>
<td>.....................</td>
<td>346</td>
</tr>
<tr>
<td>43.4.1 Methods</td>
<td>.................................</td>
<td>347</td>
</tr>
<tr>
<td>43.4.2 Properties</td>
<td>.................................</td>
<td>347</td>
</tr>
<tr>
<td>43.5 Class TargetNotBuiltWarning</td>
<td>..................</td>
<td>347</td>
</tr>
<tr>
<td>43.5.1 Methods</td>
<td>.................................</td>
<td>347</td>
</tr>
<tr>
<td>43.5.2 Properties</td>
<td>.................................</td>
<td>348</td>
</tr>
<tr>
<td>43.6 Class CacheWriteErrorWarning</td>
<td>..................</td>
<td>348</td>
</tr>
<tr>
<td>43.6.1 Methods</td>
<td>.................................</td>
<td>348</td>
</tr>
<tr>
<td>43.6.2 Properties</td>
<td>.................................</td>
<td>349</td>
</tr>
<tr>
<td>43.7 Class CorruptSConsignWarning</td>
<td>..................</td>
<td>349</td>
</tr>
<tr>
<td>43.7.1 Methods</td>
<td>.................................</td>
<td>349</td>
</tr>
<tr>
<td>43.7.2 Properties</td>
<td>.................................</td>
<td>350</td>
</tr>
<tr>
<td>43.8 Class DependencyWarning</td>
<td>..................</td>
<td>350</td>
</tr>
<tr>
<td>43.8.1 Methods</td>
<td>.................................</td>
<td>350</td>
</tr>
<tr>
<td>43.8.2 Properties</td>
<td>.................................</td>
<td>350</td>
</tr>
<tr>
<td>43.9 Class DuplicateEnvironmentWarning</td>
<td>..............</td>
<td>351</td>
</tr>
<tr>
<td>43.9.1 Methods</td>
<td>.................................</td>
<td>351</td>
</tr>
<tr>
<td>43.9.2 Properties</td>
<td>.................................</td>
<td>351</td>
</tr>
<tr>
<td>43.10 Class FutureReservedVariableWarning</td>
<td>...........</td>
<td>352</td>
</tr>
<tr>
<td>43.10.1 Methods</td>
<td>.................................</td>
<td>352</td>
</tr>
<tr>
<td>43.10.2 Properties</td>
<td>.................................</td>
<td>352</td>
</tr>
<tr>
<td>43.11 Class LinkWarning</td>
<td>..........................</td>
<td>353</td>
</tr>
<tr>
<td>43.11.1 Methods</td>
<td>.................................</td>
<td>353</td>
</tr>
<tr>
<td>43.11.2 Properties</td>
<td>.................................</td>
<td>353</td>
</tr>
<tr>
<td>43.12 Class MisleadingKeywordsWarning</td>
<td>.............</td>
<td>354</td>
</tr>
<tr>
<td>43.12.1 Methods</td>
<td>.................................</td>
<td>354</td>
</tr>
<tr>
<td>43.12.2 Properties</td>
<td>.................................</td>
<td>354</td>
</tr>
<tr>
<td>43.13 Class MissingSConscriptWarning</td>
<td>............</td>
<td>355</td>
</tr>
<tr>
<td>43.13.1 Methods</td>
<td>.................................</td>
<td>355</td>
</tr>
<tr>
<td>43.13.2 Properties</td>
<td>.................................</td>
<td>355</td>
</tr>
<tr>
<td>43.14 Class NoMD5ModuleWarning</td>
<td>...............</td>
<td>356</td>
</tr>
<tr>
<td>43.14.1 Methods</td>
<td>.................................</td>
<td>356</td>
</tr>
<tr>
<td>43.14.2 Properties</td>
<td>.................................</td>
<td>356</td>
</tr>
<tr>
<td>43.15 Class NoMetaClassSupportWarning</td>
<td>..........</td>
<td>357</td>
</tr>
<tr>
<td>43.15.1 Methods</td>
<td>.................................</td>
<td>357</td>
</tr>
<tr>
<td>43.15.2 Properties</td>
<td>.................................</td>
<td>357</td>
</tr>
<tr>
<td>43.16 Class NoObjectCountWarning</td>
<td>..............</td>
<td>358</td>
</tr>
<tr>
<td>43.16.1 Methods</td>
<td>.................................</td>
<td>358</td>
</tr>
<tr>
<td>43.16.2 Properties</td>
<td>.................................</td>
<td>358</td>
</tr>
<tr>
<td>43.17 Class NoParallelSupportWarning</td>
<td>............</td>
<td>359</td>
</tr>
<tr>
<td>43.17.1 Methods</td>
<td>.................................</td>
<td>359</td>
</tr>
<tr>
<td>43.17.2 Properties</td>
<td>.................................</td>
<td>359</td>
</tr>
<tr>
<td>43.18 Class ReservedVariableWarning</td>
<td>............</td>
<td>360</td>
</tr>
<tr>
<td>43.18.1 Methods</td>
<td>.................................</td>
<td>360</td>
</tr>
<tr>
<td>43.18.2 Properties</td>
<td>.................................</td>
<td>360</td>
</tr>
<tr>
<td>43.19 Class StackSizeWarning</td>
<td>..................</td>
<td>361</td>
</tr>
<tr>
<td>43.19.1 Methods</td>
<td>.................................</td>
<td>361</td>
</tr>
<tr>
<td>43.19.2 Properties</td>
<td>.................................</td>
<td>361</td>
</tr>
<tr>
<td>Class</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>43.20 Class VisualCMissingWarning</td>
<td>362</td>
<td></td>
</tr>
<tr>
<td>43.20.1 Methods</td>
<td>362</td>
<td></td>
</tr>
<tr>
<td>43.20.2 Properties</td>
<td>362</td>
<td></td>
</tr>
<tr>
<td>43.21 Class VisualVersionMismatch</td>
<td>363</td>
<td></td>
</tr>
<tr>
<td>43.21.1 Methods</td>
<td>363</td>
<td></td>
</tr>
<tr>
<td>43.21.2 Properties</td>
<td>363</td>
<td></td>
</tr>
<tr>
<td>43.22 Class VisualStudioMissingWarning</td>
<td>364</td>
<td></td>
</tr>
<tr>
<td>43.22.1 Methods</td>
<td>364</td>
<td></td>
</tr>
<tr>
<td>43.22.2 Properties</td>
<td>364</td>
<td></td>
</tr>
<tr>
<td>43.23 Class FortranCxxMixWarning</td>
<td>365</td>
<td></td>
</tr>
<tr>
<td>43.23.1 Methods</td>
<td>365</td>
<td></td>
</tr>
<tr>
<td>43.23.2 Properties</td>
<td>365</td>
<td></td>
</tr>
<tr>
<td>43.24 Class FutureDeprecatedWarning</td>
<td>366</td>
<td></td>
</tr>
<tr>
<td>43.24.1 Methods</td>
<td>366</td>
<td></td>
</tr>
<tr>
<td>43.24.2 Properties</td>
<td>366</td>
<td></td>
</tr>
<tr>
<td>43.25 Class DeprecatedWarning</td>
<td>367</td>
<td></td>
</tr>
<tr>
<td>43.25.1 Methods</td>
<td>367</td>
<td></td>
</tr>
<tr>
<td>43.25.2 Properties</td>
<td>367</td>
<td></td>
</tr>
<tr>
<td>43.26 Class MandatoryDeprecatedWarning</td>
<td>368</td>
<td></td>
</tr>
<tr>
<td>43.26.1 Methods</td>
<td>368</td>
<td></td>
</tr>
<tr>
<td>43.26.2 Properties</td>
<td>368</td>
<td></td>
</tr>
<tr>
<td>43.27 Class PythonVersionWarning</td>
<td>369</td>
<td></td>
</tr>
<tr>
<td>43.27.1 Methods</td>
<td>369</td>
<td></td>
</tr>
<tr>
<td>43.27.2 Properties</td>
<td>369</td>
<td></td>
</tr>
<tr>
<td>43.28 Class DeprecatedSourceCodeWarning</td>
<td>370</td>
<td></td>
</tr>
<tr>
<td>43.28.1 Methods</td>
<td>370</td>
<td></td>
</tr>
<tr>
<td>43.28.2 Properties</td>
<td>370</td>
<td></td>
</tr>
<tr>
<td>43.29 Class DeprecatedBuildDirWarning</td>
<td>371</td>
<td></td>
</tr>
<tr>
<td>43.29.1 Methods</td>
<td>371</td>
<td></td>
</tr>
<tr>
<td>43.29.2 Properties</td>
<td>371</td>
<td></td>
</tr>
<tr>
<td>43.30 Class TaskmasterNeedsExecuteWarning</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>43.30.1 Methods</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>43.30.2 Properties</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>43.31 Class DeprecatedCopyWarning</td>
<td>373</td>
<td></td>
</tr>
<tr>
<td>43.31.1 Methods</td>
<td>373</td>
<td></td>
</tr>
<tr>
<td>43.31.2 Properties</td>
<td>373</td>
<td></td>
</tr>
<tr>
<td>43.32 Class DeprecatedOptionsWarning</td>
<td>374</td>
<td></td>
</tr>
<tr>
<td>43.32.1 Methods</td>
<td>374</td>
<td></td>
</tr>
<tr>
<td>43.32.2 Properties</td>
<td>374</td>
<td></td>
</tr>
<tr>
<td>43.33 Class DeprecatedSourceSignaturesWarning</td>
<td>375</td>
<td></td>
</tr>
<tr>
<td>43.33.1 Methods</td>
<td>375</td>
<td></td>
</tr>
<tr>
<td>43.33.2 Properties</td>
<td>375</td>
<td></td>
</tr>
<tr>
<td>43.34 Class DeprecatedTargetSignaturesWarning</td>
<td>376</td>
<td></td>
</tr>
<tr>
<td>43.34.1 Methods</td>
<td>376</td>
<td></td>
</tr>
<tr>
<td>43.34.2 Properties</td>
<td>376</td>
<td></td>
</tr>
<tr>
<td>43.35 Class DeprecatedDebugOptionsWarning</td>
<td>377</td>
<td></td>
</tr>
<tr>
<td>43.35.1 Methods</td>
<td>377</td>
<td></td>
</tr>
<tr>
<td>43.35.2 Properties</td>
<td>377</td>
<td></td>
</tr>
<tr>
<td>43.36 Class DeprecatedSigModuleWarning</td>
<td>378</td>
<td></td>
</tr>
<tr>
<td>43.36.1 Methods</td>
<td>378</td>
<td></td>
</tr>
<tr>
<td>43.36.2 Properties</td>
<td>378</td>
<td></td>
</tr>
</tbody>
</table>
43.37 Class DeprecatedBuilderKeywordsWarning ........................................... 379
  43.37.1 Methods .................................................................................. 379
  43.37.2 Properties .............................................................................. 379

44 Module SCons.cpp ................................................................................. 380
  44.1 Functions ...................................................................................... 380
  44.2 Variables ...................................................................................... 380
  44.3 Class FunctionEvaluator ................................................................. 381
    44.3.1 Methods ............................................................................... 381
    44.3.2 Properties ........................................................................... 381
  44.4 Class PreProcessor ......................................................................... 381
    44.4.1 Methods ............................................................................... 382
    44.4.2 Properties ........................................................................... 386
  44.5 Class DumbPreProcessor ................................................................. 386
    44.5.1 Methods ............................................................................... 386
    44.5.2 Properties ........................................................................... 386

45 Module SCons.dblite ............................................................................. 388
  45.1 Functions ...................................................................................... 388
  45.2 Variables ...................................................................................... 388
  45.3 Class dblite .................................................................................. 388
    45.3.1 Methods ............................................................................... 388
    45.3.2 Properties ........................................................................... 389

46 Module SCons.exitfuncs ...................................................................... 390
  46.1 Functions ...................................................................................... 390
  46.2 Variables ...................................................................................... 390
1 Package SCons

SCons

The main package for the SCons software construction utility. **Version:** 2.3.1

**Date:** 2014/03/02 14:18:15

1.1 Modules

- **Action:** SCons.Action
  *(Section 2, p. 5)*
- **Builder:** SCons.Builder
  *(Section 3, p. 18)*
- **CacheDir:** CacheDir support
  *(Section 4, p. 30)*
- **Conftest:** SCons.Conftest
  *(Section 5, p. 33)*
- **Debug:** SCons.Debug
  *(Section 6, p. 38)*
- **Defaults:** SConsDefaults
  *(Section 7, p. 40)*
- **Environment:** SCons.Environment
  *(Section 8, p. 45)*
- **Errors:** SCons.Errors
  *(Section 9, p. 74)*
- **Executor:** SCons.Executor
  *(Section 10, p. 82)*
- **Job:** SCons.Job
  *(Section 11, p. 93)*
- **Memoize:** Memoizer
  *(Section 12, p. 100)*
- **Node:** SCons.Node
  *(Section 13, p. 108)*
  - **Alias:** scons.Node.Alias
    *(Section 14, p. 128)*
  - **FS:** scons.Node.FS
    *(Section 15, p. 135)*
  - **Python:** scons.Node.Python
    *(Section 16, p. 180)*
- **PathList:** SCons.PathList
  *(Section 17, p. 186)*
- **SConf:** SCons.SConf
  *(Section 18, p. 187)*
- **SConsign:** SCons.SConsign
  *(Section 19, p. 202)*
- **Scanner:** SCons.Scanner
  *(Section 20, p. 209)*
  - **C:** SCons.Scanner.C
    *(Section 21, p. 225)*
  - **D:** SCons.Scanner.D
• **Script**: SCons.Script
  (Section 29, p. 249)
  – **Interactive**: SCons interactive mode
    (Section 30, p. 258)
  – **Main**: SCons.Script
    (Section 31, p. 261)
  – **SConscript**: SCons.Script.SConscript
    (Section 32, p. 275)

• **Sig**: Place-holder for the old SCons.Sig module hierarchy
  (Section 33, p. 281)

• **Subst**: SCons.Subst
  (Section 34, p. 284)

• **Taskmaster**: Generic Taskmaster module for the SCons build engine.
  (Section 35, p. 295)

• **Util**: SCons.Util
  (Section 36, p. 306)

• **Variables**: engine.SCons.Variables
  (Section 37, p. 333)
  – **BoolVariable**: engine.SCons.Variables.BoolVariable
    (Section 38, p. 337)
  – **EnumVariable**: engine.SCons.Variables.EnumVariable
    (Section 39, p. 338)
  – **ListVariable**: engine.SCons.Variables.ListVariable
    (Section 40, p. 340)
  – **PackageVariable**: engine.SCons.Variables.PackageVariable
    (Section 41, p. 341)
  – **PathVariable**: engine.SCons.Variables.PathVariable
    (Section 42, p. 342)

• **Warnings**: SCons.Warnings
  (Section 43, p. 344)

• **cpp**: SCons C Pre-Processor module
  (Section 44, p. 380)

• **dblite**: (Section 45, p. 388)

• **exitfuncs**: SCons.exitfuncs
  (Section 46, p. 390)
## 1.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>build</strong></td>
<td>Value: <code>'</code></td>
</tr>
<tr>
<td><strong>buildsys</strong></td>
<td>Value: <code>'lubuntu'</code></td>
</tr>
<tr>
<td><strong>developer</strong></td>
<td>Value: <code>'garyo'</code></td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: <code>'SCons'</code></td>
</tr>
<tr>
<td><strong>revision</strong></td>
<td>Value: <code>'src/engine/SCons/__init__.py 2014/03/02 14:18:15 garyo'</code></td>
</tr>
</tbody>
</table>
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.

- **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)**
- **remove_set_lineno_codes(x)**
- **Action(act,*args,**kw)**

A factory for action objects.

- **get_default_ENV(env)**

### 2.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Action.py 2014/03/02 14:18:15 garyo'</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>SET_LINENO</td>
<td>Value: dis.SET_LINENO</td>
</tr>
<tr>
<td>HAVE_ARGUMENT</td>
<td>Value: dis.HAVE_ARGUMENT</td>
</tr>
<tr>
<td>strip_quotes</td>
<td>Value: re.compile(r&quot;^[&quot;&quot;(.*)[&quot;&quot;)]$&quot;)</td>
</tr>
<tr>
<td>default_ENV</td>
<td>Value: None</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

### 2.3 Class ActionBase

**Object**  
*SCons.Action.ActionBase*

**Known Subclasses:**  

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

```python
__cmp__(self, other)
```

```
no_batch_key(self, env, target, source)
```

```
batch_key(self, env, target, source)
```

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

```
get_contents(self, target, source, env)
```

```
__add__(self, other)
```

```
__radd__(self, other)
```

```
presub_lines(self, env)
```

```
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

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

2.3.2 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>
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__(), __cmp__(), __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

**`__init__`**

```
x.__init__(...)
```
Overrides: object.__init__ extit(inherited documentation)

**`__str__`**

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

**`batch_key`**

```
batch_key(self, env, target, source)
```
Overrides: SCons.Action.ActionBase.batch_key

**`genstring`**

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

**`__call__`**

```
```

**`get_presig`**

```
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`**

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

**`get_varlist`**

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

**`get_targets`**

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

```
__add__(), __cmp__(), __radd__(), get_contents(), no_batch_key(), pre
sub_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>
</table>

Inherited from object

```
__class__
```

2.6 Class `LazyAction`

```
object

SCons.Action.ActionBase

SCons.Action.CommandGeneratorAction

object

SCons.Action.ActionBase

SCons.Action._ActionAction

SCons.Action.CommandAction

SCons.Action.LazyAction
```

2.6.1 Methods

```
__init__(self, var, kw)
```

x.__init__(...) initializes x; see help(type(x)) for signature  
Overrides: object.__init__ extit(inherited documentation)
**get_parent_class**(*self, env*)

**___call___**(*self, target, source, env, *args, **kw*)

Overrides: SCons.Action.__ActionAction.__call__

**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 (inherited documentation)

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


---

**Inherited from SCons.Action.CommandGeneratorAction** *(Section 2.5)*

___str___(), batch_key(), genstring(), get_implicit_deps(), get_targets()

**Inherited from SCons.Action.CommandAction** *(Section 2.4)*

execute(), process(), strfunction()

**Inherited from SCons.Action.__ActionAction**

print_cmd_line()

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

___add___(), ___cmp___(), ___radd___(), get_contents(), no_batch_key(), pre-sub_lines()

**Inherited from object**

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___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><strong>class</strong></td>
</tr>
</tbody>
</table>

27
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__ (inherited documentation)

```python
function_name(self)
```

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

```python
__str__(self)
```

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

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

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

Return the signature contents of this callable action.

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

Inherited from SCons.Action._ActionAction

```python
__call__(), print_cmd_line()
```

Inherited from SCons.Action.ActionBase (Section 2.3)

```python
__add__(), __cmp__(), __radd__(), batch_key(), genstring(), get_contents(),
geet_targets(), get_varlist(), no_batch_key(), presub_lines()
```
Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__subclasshook__()
get__presig__(self, target, source, env)

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


get__implicit__deps(self, target, source, env)

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

Inherited from SCons.Action.ActionBase(Section 2.3)

__add__(), __cmp__(), __radd__(), batch_key(), get__contents(), get__targets(), no__batch_key()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__init__</code></td>
<td><code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature. Overrides: object.<strong>init</strong> (inherited documentation)</td>
</tr>
<tr>
<td><code>get_contents</code></td>
<td></td>
</tr>
<tr>
<td><code>subst</code></td>
<td></td>
</tr>
<tr>
<td><code>subst_args</code></td>
<td></td>
</tr>
<tr>
<td><code>subst_kw</code></td>
<td></td>
</tr>
<tr>
<td><code>__call__</code></td>
<td><code>__call__(self, target, source, env)</code></td>
</tr>
<tr>
<td><code>strfunction</code></td>
<td></td>
</tr>
<tr>
<td><code>__str__</code></td>
<td><code>str(x)</code> Overrides: object.<strong>str</strong> (inherited documentation)</td>
</tr>
</tbody>
</table>

**Inherited from object**

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

### 2.9.2 Properties

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

```python
__init__(self, actfunc, strfunc, convert=<function <lambda> at 0x95fec6c>)
```

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__()
```

2.10.2 Properties

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

Inherited from object

```python
__class__
```
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:

_execlute()
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

match_splitext(path, suffixes=[]) A factory for builder objects.
Class DictCmdGenerator

Module SCons.Builder

`is_a_Builder(obj)`

"Returns True iff 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>
</table>
| __revision__       | Value: 'src/engine/SCons/Builder.py
2014/03/02 14:18:15 garyo'             |
| misleading_keywords| Value: {'sources': 'source', 'targets': 'target'} |
| __package__        | Value: 'SCons'                           |

3.3 Class 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)
```

Overrides: UserDict.UserDict.__init__

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

Add a suffix-action pair to the mapping.

__call__(self, target, source, env, for_signature)

Overrides: SCons.Util.Selector.__call__

Inherited from SCons.Util.OrderedDict(Section 36.10)

__delitem__(), __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()

3.3.2 Class Variables

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

3.4 Class CallableSelector

UserDict.UserDict

SCons.Util.OrderedDict

SCons.Util.Selector

SCons.Builder.CallableSelector

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

3.4.1 Methods

__call__(self, env, source)

 Overrides: SCons.Util.Selector.__call__

Inherited from SCons.Util.OrderedDict(Section 36.10)
Inherited from UserDict.UserDict

Inherited from SCons.Util.OrderedDict (Section 36.10)

Inherited from SCons.Util.Selector

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

__call__(self, target, source, env)

Overrides: SCons.Util.Selector.__call__

Inherited from SCons.Util.OrderedDict (Section 36.10)

Inherited from SCons.Builder.DictEmitter

Inherited from UserDict.UserDict

3.5 Class DictEmitter

UserDict.UserDict

SCons.Util.OrderedDict

SCons.Util.Selector

SCons.Builder.DictEmitter

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

3.4.2 Class Variables
3.5.2 Class Variables

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

```
_incall_(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
3.6.2 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>

3.6.3 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>

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
3.8 Class 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

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

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

```python
__cmp__(self, other)
```

Inherited from object

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

3.8.2 Properties
### 3.9 Class BuilderBase

![Object Hierarchy](image.png)

**SCons.Builder.BuilderBase**

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

#### 3.9.1 Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>init</strong></td>
<td>(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=&lt;class 'SCons.Builder._Null'&gt;, is_explicit=1, src_builder=None, ensure_suffix=False, **overrides)</td>
</tr>
<tr>
<td><strong>nonzero</strong></td>
<td>(self)</td>
</tr>
<tr>
<td>get_name</td>
<td>(self, env)</td>
</tr>
</tbody>
</table>

__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).

| __cmp__ | (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.

subst_src_suffixes(self, env)

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.

src_suffixes(self, env)

Returns the list of source suffixes for all src_builders of this Builder.
This is essentially a recursive descent of the src_builders “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>

3.9.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>metaclass</strong></td>
<td>Value: SCons.Memoize.Memoized_Metaclass</td>
</tr>
<tr>
<td>memoizer_counters</td>
<td>Value: []</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
__cmp__(), __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><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

*Inherited from object*
4 Module SCons.CacheDir

CacheDir support

4.1 Functions

\textbf{CacheRetrieveFunc}(target, source, env)

\textbf{CacheRetrieveString}(target, source, env)

\textbf{CachePushFunc}(target, source, env)

4.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em><strong>revision</strong></em></td>
<td>Value: 'src/engine/SCons/CacheDir.py 2014/03/02 14:18:15 garyo'</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>CacheRetrieveSilent</td>
<td>Value: SCons.Action.Action(CacheRetrieveFunc, None)</td>
</tr>
<tr>
<td><em><strong>package</strong></em></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

4.3 Class CacheDir

\texttt{object \_\_\_package\_\_\_}

\texttt{SCons.CacheDir.CacheDir}
4.3.1 Methods

```python
__init__(self, path)
```

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

Overrides:
`object.__init__` (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__(),
```
Class CacheDir

__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>
5 Module SCons.Conftest

SCons.Conftest

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

5.1 Functions

<table>
<thead>
<tr>
<th><strong>CheckBuilder</strong>(context, text=None, language=None)</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CheckCC</strong>(context)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure check for a working C compiler.</td>
</tr>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>CheckSHCC</strong>(context)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure check for a working shared C compiler.</td>
</tr>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>
Functions

Module SCons.Conftest

**CheckCXX**(context)

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.

**CheckSHCXX**(context)

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.

**CheckFunc**(context, function_name, header=None, language=None)

Configure check for a function “function_name”. “language” 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.

**CheckHeader**(context, header_name, header=None, language=None, include_quotes=None)

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). “language” 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.
**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

`CheckDeclaration(context, symbol, includes=None, language=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

`CheckLib(context, libs, func_name=None, header=None, extra_libs=None, call=None, language=None, autoadd=1, append=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 $LIBS are set correctly. Returns an empty string for success, an error message for failure.
### 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.

6.1 Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logInstanceCreation</td>
<td>instance creation logging function</td>
</tr>
<tr>
<td>string_to_classes</td>
<td>Convert string to classes</td>
</tr>
<tr>
<td>fetchLoggedInstances</td>
<td>Fetch logged instances</td>
</tr>
<tr>
<td>countLoggedInstances</td>
<td>Count logged instances</td>
</tr>
<tr>
<td>listLoggedInstances</td>
<td>List logged instances</td>
</tr>
<tr>
<td>dumpLoggedInstances</td>
<td>Dump logged instances</td>
</tr>
<tr>
<td>memory()</td>
<td>Access memory</td>
</tr>
<tr>
<td>caller_stack()</td>
<td>Get caller stack</td>
</tr>
<tr>
<td>caller_trace(back=0)</td>
<td>Call stack tracing</td>
</tr>
<tr>
<td>dump_caller_counts(file=sys.stderr)</td>
<td>Dump caller counts file</td>
</tr>
<tr>
<td>func_shorten(func_tuple)</td>
<td>Function shorten</td>
</tr>
<tr>
<td>Trace(msg, file=None, mode='w', tstamp=None)</td>
<td>Trace message to file. Whenever a file is specified, it becomes the default for the next call to Trace().</td>
</tr>
</tbody>
</table>

6.2 Variables
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Debug.py 2014/03/02 14:18:15 garyo'</td>
</tr>
<tr>
<td>track_instances</td>
<td>Value: False</td>
</tr>
<tr>
<td>tracked_classes</td>
<td>Value: {}</td>
</tr>
<tr>
<td>caller_bases</td>
<td>Value: {}</td>
</tr>
<tr>
<td>caller_dicts</td>
<td>Value: {}</td>
</tr>
<tr>
<td>shorten_list</td>
<td>Value: [('scons/SCons/', 1), ('/usr/...</td>
</tr>
<tr>
<td>TraceFP</td>
<td>Value: {}</td>
</tr>
<tr>
<td>TraceDefault</td>
<td>Value: '/dev/tty'</td>
</tr>
<tr>
<td>TimeStampDefault</td>
<td>Value: None</td>
</tr>
<tr>
<td>StartTime</td>
<td>Value: 1393788010.84</td>
</tr>
<tr>
<td>PreviousTime</td>
<td>Value: 1393788010.84</td>
</tr>
<tr>
<td><strong>package</strong></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)
```

```
copy_func(dest, src)
```
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 2014/03/02 14:18:15 garyo'</td>
</tr>
<tr>
<td>SharedCheck</td>
<td>Value: SCons.Action.Action(SharedFlagChecker, None)</td>
</tr>
<tr>
<td>CScan</td>
<td>Value: SCons.Defaults.CScan</td>
</tr>
<tr>
<td>DScan</td>
<td>Value: SCons.Tool.DScanner</td>
</tr>
<tr>
<td>LaTeXScan</td>
<td>Value: SCons.Tool.LaTeXScanner</td>
</tr>
<tr>
<td>ObjSourceScan</td>
<td>Value: SCons.Tool.SourceFileScanner</td>
</tr>
<tr>
<td>ProgScan</td>
<td>Value: SCons.Tool.ProgramScanner</td>
</tr>
<tr>
<td>DirScanner</td>
<td>Value: SCons.Defaults.DirScanner</td>
</tr>
<tr>
<td>DirEntryScanner</td>
<td>Value: SCons.Scanner.Dir.DirEntryScanner()</td>
</tr>
<tr>
<td>CAction</td>
<td>Value: SCons.Action.Action(&quot;$CCCOM&quot;, &quot;$CCCOMSTR&quot;)</td>
</tr>
<tr>
<td>ShCAction</td>
<td>Value: SCons.Action.Action(&quot;$SHCCCOM&quot;, &quot;$SHCCCOMSTR&quot;)</td>
</tr>
<tr>
<td>CXXAction</td>
<td>Value: SCons.Action.Action(&quot;$CXXCOM&quot;, &quot;$CXXCOMSTR&quot;)</td>
</tr>
<tr>
<td>ShCXXAction</td>
<td>Value: SCons.Action.Action(&quot;$SHCXXCOM&quot;, &quot;$SHCXXCOMSTR&quot;)</td>
</tr>
<tr>
<td>ASAction</td>
<td>Value: SCons.Action.Action(&quot;$ASCOM&quot;, &quot;$ASCOMSTR&quot;)</td>
</tr>
</tbody>
</table>

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

`nullcmdgenerator` 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:

```python
env["DO NOTHING"] = NullCmdGenerator
env["LINKCOM"] = "${DO NOTHING($SOURCES $TARGET)}"
```

#### 7.3.1 Methods

- **__init__**

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

- **__call__**

  ```python
  __call__(self, target, source, env, for_signature=None)
  ```
7.4 Class 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)
x.__init__(...) initializes x; see help(type(x)) for signature Overrides:
object.__init__ extit(inherited documentation)
```

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

7.4.2 Properties

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>Inherited from object</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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias_builder(env, target, source)</td>
<td></td>
</tr>
<tr>
<td>apply_tools(env, tools, toolpath)</td>
<td></td>
</tr>
<tr>
<td>copy_non_reserved_keywords(dict)</td>
<td></td>
</tr>
<tr>
<td>is_valid_construction_var(varstr)</td>
<td>Return if the specified string is a legitimate construction variable.</td>
</tr>
<tr>
<td>default_decide_source(dependency, target, prev_ni)</td>
<td></td>
</tr>
<tr>
<td>default_decide_target(dependency, target, prev_ni)</td>
<td></td>
</tr>
<tr>
<td>default_copy_from_cache(src, dst)</td>
<td></td>
</tr>
<tr>
<td>NoSubstitutionProxy(subject)</td>
<td></td>
</tr>
</tbody>
</table>

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 2014/03/02 14:18:15 garyo’</td>
</tr>
<tr>
<td>CleanTargets</td>
<td>Value: {}</td>
</tr>
<tr>
<td>CalculatorArgs</td>
<td>Value: {}</td>
</tr>
<tr>
<td>AliasBuilder</td>
<td>Value: SCons.Builder.Builder(action= alias_builder, target_facto...</td>
</tr>
</tbody>
</table>
### Class MethodWrapper

**object**

```plaintext
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 `__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).

#### 8.3.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong><strong>init</strong></strong></td>
<td><code>__init__(self, object, method, name=None)</code></td>
</tr>
<tr>
<td><strong><strong>call</strong></strong></td>
<td><code>__call__(self, *args, **kwargs)</code></td>
</tr>
<tr>
<td><strong>clone</strong></td>
<td><code>clone(self, new_object)</code></td>
</tr>
</tbody>
</table>

*Returns an object that re-binds the underlying “method” to the specified new object.*
8.3.2 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>

8.4 Class BuilderWrapper

A MethodWrapper subclass that 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

```python
__call__(self, target=None, source=<class 'SCons.Environment._Null'>, *args, **kw)
```

```python
__repr__(self)
repr(x) Overrides: object.__repr__ extit(inherited documentation)
```
8.4.2 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>

8.5 Class BuilderDict

UserDict.UserDict

\[\text{SCons.Environment.BuilderDict}\]

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

\[
\text{__init__}(\text{self}, \text{dict}, \text{env})
\]

Overrides: UserDict.UserDict.__init__

\[
\text{__semi__deepcopy__}(\text{self})
\]
Class SubstitutionEnvironment

__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
__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

<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>

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
def __init__(self, **kw):
    pass

def __cmp__(self, other):
    pass

def __delitem__(self, key):
    pass

def __getitem__(self, key):
    pass

def __setitem__(self, key, value):
    pass

def get(self, key, default=None):
    pass

def has_key(self, key):
    pass

def __contains__(self, key):
    pass

def items(self):
    pass

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

def gvars(self):
    pass

def lvars(self):
    pass
```
Class SubstitutionEnvironment

Module SCons.Environment

```python
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.

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

```python
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.

```python
subst_path(self, path, target=None, source=None)
```

Substitute a path list, turning EntryProxies into Nodes and leaving Nodes (and other objects) as-is.

```python
subst_target_source(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.

```python
backtick(self, command)
```
### AddMethod(self, function, name=None)

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.

### RemoveMethod(self, function)

Removes the specified function’s MethodWrapper from the added_methods list, so we don’t re-bind it when making a clone.

### 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**
Class Base

Module SCons.Environment

__delattr__(), __format__(), __getattribute__(), __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>

8.6.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>metaclass</strong></td>
<td>Value: SCons.Memoize.Memoized_Metaclass</td>
</tr>
</tbody>
</table>

8.7 Class Base

object

SCons.Environment.SubstitutionEnvironment

SCons.Environment.Base


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

**AlwaysBuild**(self, *targets)**

Append values to existing construction variables in an Environment.

### Append

**Append**(self, **kw)**

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).

### AppendENVPath

**AppendENVPath**(self, name, newpath, envname='ENV', sep=':',
delete_existing=1)

### AppendUnique

**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

**BuildDir**(self, *args, **kw)**

### Builder

**Builder**(self, **kw)**

### CacheDir

**CacheDir**(self, path)

### Clean

**Clean**(self, targets, files)

### Clone

**Clone**(self, tools=[], toolpath=None, parse_flags=None, **kw)**

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.
Class Base Module SCons.Environment

**Command**(self, target, source, action, **kw)

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, dump the contents of the scons build environment to stdout.

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\texttt{(self, action, *args, **kw)}

Directly execute an action through an Environment.

File\texttt{(self, name, *args, **kw)}

FindFile\texttt{(self, file, dirs)}

FindInstalledFiles\texttt{(self)}

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

FindIxes\texttt{(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\texttt{(self, node=’.’)}

returns a list of all source files.

Flatten\texttt{(self, sequence)}

GetBuildPath\texttt{(self, files)}

Glob\texttt{(self, pattern, ondisk=True, source=False, strings=False)}

Ignore\texttt{(self, target, dependency)}

Ignore a dependency.

Literal\texttt{(self, string)}

Local\texttt{(self, *targets)}
### NoCache

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

Tags a target so that it will not be cached

### NoClean

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

Tags a target so that it will not be cleaned by -c

### ParseConfig

```python
ParseConfig(self, command, function=None, unique=1)
```

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.

### ParseDepends

```python
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

```python
Platform(self, platform)
```

### Precious

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

### Prepend

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

Prepend values to existing construction variables in an Environment.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrependENVPath</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>PrependUnique</td>
<td>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.</td>
</tr>
<tr>
<td>Replace</td>
<td>Replace existing construction variables in an Environment with new construction variables and/or values.</td>
</tr>
<tr>
<td>ReplaceIxes</td>
<td>Replace old_prefix with new_prefix and old_suffix with new_suffix.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Repository</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

TargetSignatures(self, type)

Tool(self, tool, toolpath=None, **kw)
**Value**(self, value, built_value=None)

**VariantDir**(self, variant_dir, src_dir, duplicate=1)

**WhereIs**(self, prog, path=None, pathext=None, reject=[])  
Find prog in the path.

**__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 (SubsitutionEnvironment) 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(), __cmp__(),
__contains__(), __delitem__(), __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__(), __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>

### 8.7.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memoizer_counters</td>
<td>Value: []</td>
</tr>
<tr>
<td>Inherited from SCons.Environment.SubstitutionEnvironment (Section 8.6)</td>
<td></td>
</tr>
<tr>
<td><strong>metaclass</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 8.8 Class 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 (SubstitutionEnvironment) 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

~~~
__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 (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__(inherited documentation)

__getattr__(self, name)

__setattr__(self, name, value)

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

__getitem__(self, key)

Overrides: SCons.Environment.SubstitutionEnvironment.__getitem__

__setitem__(self, key, value)

Overrides: SCons.Environment.SubstitutionEnvironment.__setitem__
~~~
Class `OverrideEnvironment`  
Module `SCons.Environment`

```python
__delitem__(self, key)
Overrides: SCons.Environment.SubstitutionEnvironment.__delitem__
```

```python
get(self, key, default=None)
```

```python
has_key(self, key)
Overrides: SCons.Environment.SubstitutionEnvironment.has_key
```

```python
__contains__(self, key)
Overrides: SCons.Environment.SubstitutionEnvironment.__contains__
```

```python
Dictionary(self)
```

```python
items(self)
```

```python
gvars(self)
```

```python
lvars(self)
```

```python
Replace(self, **kw)
```
Replace existing construction variables in an Environment with new construction variables and/or values. Overrides: SCons.Environment.Base.Replace extit(inherited documentation)

*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(), FindInstalled-
Files(), FindIxes(), FindSourceFiles(), Flatten(), GetBuildPath(), Glob(), Ignore(),
Literal(), Local(), NoCache(), NoClean(), ParseConfig(), ParseDepends(), Platform(),
Precious(), Prepend(), PrependENVPath(), PrependUnique(), Pseudo(),
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(), __cmp__() ,
arg2nodes(), backtick(), subst(), subst_kw(), subst_list(), subst_path(), subst_target_source()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __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>

8.8.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Environment.Base (Section 8.9)</td>
<td>memoizer_conters</td>
</tr>
<tr>
<td>Inherited from SCons.Environment.SubstitutionEnvironment (Section 8.6)</td>
<td><strong>metaclass</strong></td>
</tr>
</tbody>
</table>
8.9 Class Base

object →

SCons.Environment.SubstitutionEnvironment →

SCons.Environment.Base


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><strong>Action</strong>(self, *args, **kw)</td>
<td>Append values to existing construction variables in an Environment.</td>
</tr>
<tr>
<td><strong>AddPostAction</strong>(self, files, action)</td>
<td></td>
</tr>
<tr>
<td><strong>AddPreAction</strong>(self, files, action)</td>
<td></td>
</tr>
<tr>
<td><strong>Alias</strong>(self, target, source=[], action=None, **kw)</td>
<td></td>
</tr>
<tr>
<td><strong>AlwaysBuild</strong>(self, *targets)</td>
<td></td>
</tr>
<tr>
<td><strong>Append</strong>(self, **kw)</td>
<td></td>
</tr>
</tbody>
</table>
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).

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.

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, dump the contents of the scons build environment to stdout.

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\((self, name, *args, **kw)\)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FindFile(self, file, dirs)</code></td>
<td></td>
</tr>
<tr>
<td><code>FindInstalledFiles(self)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>returns the list of all targets of the Install and InstallAs Builder.</td>
</tr>
<tr>
<td><code>FindIxes(self, paths, prefix, suffix)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Search a list of paths for something that matches the prefix and suffix.</td>
</tr>
<tr>
<td></td>
<td>paths - the list of paths or nodes. prefix - construction variable for the prefix.</td>
</tr>
<tr>
<td></td>
<td>suffix - construction variable for the suffix.</td>
</tr>
<tr>
<td><code>FindSourceFiles(self, node='.' )</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>returns a list of all source files.</td>
</tr>
<tr>
<td><code>Flatten(self, sequence)</code></td>
<td></td>
</tr>
<tr>
<td><code>GetBuildPath(self, files)</code></td>
<td></td>
</tr>
<tr>
<td><code>Glob(self, pattern, ondisk=True, source=False, strings=False)</code></td>
<td></td>
</tr>
<tr>
<td><code>Ignore(self, target, dependency)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ignore a dependency.</td>
</tr>
<tr>
<td><code>Literal(self, string)</code></td>
<td></td>
</tr>
<tr>
<td><code>Local(self, *targets)</code></td>
<td></td>
</tr>
<tr>
<td><code>NoCache(self, *targets)</code></td>
<td>Tags a target so that it will not be cached.</td>
</tr>
</tbody>
</table>
NoClean(self, *targets)

Tags a target so that it will not be cleaned by -c

ParseConfig(self, command, function=\texttt{None}, unique=1)

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.

ParseDepends(self, filename, must_exist=\texttt{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\( (self, *targets) \)

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.

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.

**TargetSignatures**(*self*, *type*)

**Tool**(*self*, *tool*, *toolpath=None, **kw*)
Class Base

Module SCons.Environment

**Value** *(self, value, built_value=None)*

**VariantDir** *(self, variant_dir, src_dir, duplicate=1)*

**WhereIs** *(self, prog, path=None, pathext=None, reject=[])*

Find prog in the path.

**__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)

Inherited from object

8.9.2 Properties

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

8.9.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memoizer_counters</td>
<td>Value: []</td>
</tr>
</tbody>
</table>

Inherited from SCons.Environment.SubstitutionEnvironment (Section 8.6)

__metaclass__
9 Module SCons.Errors

SCons.Errors

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

9.1 Functions

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

Convert any return code a BuildError Exception.

‘status’ can either be a return code or an Exception. The buildError.status we set here will normally be used as the exit status of the “scons” process.

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 2014/03/02 14:18:15 garyo'</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 occuring 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

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

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

```python
__str__(self)
```

```
str(x) Overrides: object.__str__
```

Inherited from exceptions.Exception

```python
__new__()
```

Inherited from exceptions.BaseException

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

Inherited from object

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

9.3.2 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>

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

<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>

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**
9.5.2 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>

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
9.7 Class EnvironmentError

Inherited from exceptions.BaseException

<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>

9.7.1 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__()

9.7.2 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>Inherited from object</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

  __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></td>
<td>Inherited from exceptions.BaseException</td>
</tr>
<tr>
<td></td>
<td>Inherited from object</td>
</tr>
</tbody>
</table>
9.9 Class ExplicitExit

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.ExplicitExit

9.9.1 Methods

```python
__init__(self, node=None, status=None, *args)
```

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

9.9.2 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>
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>rfile(node)</td>
<td>A function to return the results of a Node’s rfile() method, if it exists, and the Node itself otherwise (if it’s a Value Node, e.g.).</td>
</tr>
</tbody>
</table>

GetBatchExecutor(key)

AddBatchExecutor(key, executor)

get_NullEnvironment()

Use singleton pattern for Null Environments.

10.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Executor.py 2014/03/02 14:18:15 garyo’</td>
</tr>
<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

```
__init__(self, targets=[], sources=[])  
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__()
```

10.3.2 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>

10.4 Class TSList

```
object  
  _abcoll.Sized  
object  
  _abcoll.Iterable  
object  
  _abcoll.Container  
    _abcoll.Sequence  
    _abcoll.MutableSequence  
UserList.UserList  
SCons.Executor.TSList
```

A class that implements $TARGETS or $SOURCES expansions by wrapping an executor.

Method. This class is used in the Executor.Ivars() to delay creation of NodeList objects until
they’re needed.

Note that we subclass collections.UserList purely so that the isSequence() 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>init</strong>(self, func)</td>
<td>x.<strong>init</strong>(...) initializes x; see help(type(x)) for signature. Overrides: object.<strong>init</strong>.</td>
</tr>
<tr>
<td><strong>getattr</strong>(self, attr)</td>
<td></td>
</tr>
<tr>
<td><strong>getitem</strong>(self, i)</td>
<td>Overrides: _abcoll.Sequence.<strong>getitem</strong>.</td>
</tr>
<tr>
<td><strong>getslice</strong>(self, i, j)</td>
<td>Overrides: UserList.UserList.<strong>getslice</strong>.</td>
</tr>
<tr>
<td><strong>str</strong>(self)</td>
<td>str(x) Overrides: object.<strong>str</strong>.</td>
</tr>
<tr>
<td><strong>repr</strong>(self)</td>
<td>repr(x) Overrides: object.<strong>repr</strong>.</td>
</tr>
</tbody>
</table>

**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></td>
</tr>
<tr>
<td><code>__class__</code></td>
<td></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 <code>UserList.UserList</code></td>
<td></td>
</tr>
<tr>
<td><code>__abstractmethods__</code>, <code>__hash__</code></td>
<td></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

**`__init__`(self, func)**

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

**`__getattr__`(self, attr)**

**`__str__`(self)**

str(x)  
Overrides: object.`__str__`  
(inherited documentation)
Class Executor Module SCons.Executor

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

**Inherited from object**

```
__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><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>

### 10.6 Class Executor

```
object └── SCons.Executor.Executor
```

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__ (inherited documentation)
```

```
get_lvars(self)
```

```
get_action_targets(self)
```

```
set_action_list(self, action)
```

```
get_action_list(self)
```

101
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_all_targets(self)</td>
<td>Returns all targets for all batches of this Executor.</td>
</tr>
<tr>
<td>get_all_sources(self)</td>
<td>Returns all sources for all batches of this Executor.</td>
</tr>
<tr>
<td>get_all_children(self)</td>
<td>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.</td>
</tr>
<tr>
<td>get_all_prerequisites(self)</td>
<td>Returns all unique (order-only) prerequisites for all batches of this Executor.</td>
</tr>
<tr>
<td>get_action_side_effects(self)</td>
<td>Returns all side effects for all batches of this Executor used by the underlying Action.</td>
</tr>
<tr>
<td>get_build_env(self)</td>
<td>Fetch or create the appropriate build Environment for this Executor.</td>
</tr>
<tr>
<td>get_buildScannerPath(self, scanner)</td>
<td>Fetch the scanner path for this executor’s targets and sources.</td>
</tr>
</tbody>
</table>
get_kw(self, kw={})

do_nothing(self, target, kw)

do_execute(self, target, kw)

Actually execute the action list.

__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)
Class Executor

Module SCons.Executor

```
__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)

Fetch 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__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
```
10.6.2 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>

10.6.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memoizer_counters</td>
<td>Value: []</td>
</tr>
</tbody>
</table>

10.6.4 Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>my_str</td>
<td>Value: &lt;function my_str at 0x95f14fc&gt;</td>
</tr>
</tbody>
</table>

10.7 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.

10.7.1 Methods

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

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

```
get_build_env(self)
```

```
get_build_scanner_path(self)
```
Class Null Module SCons.Executor

```
cleanup(self)
prepare(self)
get_unignored_sources(self, *args, **kw)
get_action_targets(self)
get_action_list(self)
get_all_targets(self)
get_all_sources(self)
get_all_children(self)
get_all_prerequisites(self)
get_action_side_effects(self)
__call__(self, *args, **kw)
get_contents(self)
ad_pre_action(self, action)
ad_post_action(self, action)
set_action_list(self, action)
```

Inherited from object

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

10.7.2 Properties

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

Inherited from object

continued on next page

106
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
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 2014/03/02 14:18:15 garyo'</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

```python
__init__(self)
```

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

```python
set(self)
```

```python
__call__(self)
```

Inherited from object

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

```python
__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__

```python
run(self, postfunc=<function <lambda> at 0x98b909c>)
```

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></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></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__
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__()
```

11.4.2 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><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

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

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

_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>
</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__
class Parallel Module SCons.Job

\textbf{put}(self, task)

Put task into request queue.

\textbf{get}(self)

Remove and return a result tuple from the results queue.

\textbf{preparation\_failed}(self, task)

\textbf{cleanup}(self)

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

\textit{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>_class_</td>
<td></td>
</tr>
</tbody>
</table>

11.7 Class Parallel

\texttt{object} \quad \texttt{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

```python
__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__`

```python
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__()
```

11.7.2 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>

114
Module SCons.Memoize

Memoizer

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

```python
memoizer_counters = []  # Memoization
memoizer_counters.append(SCons.Memoize.CountValue('foo'))  # Memoization
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
memoizer_counters.append(SCons.Memoize.CountDict('bar', _bar_key))  # Memoization
def bar(self, argument):
    memo_key = argument  # Memoization
    try:  # Memoization
        memo_dict = self._memo['bar']  # Memoization
    except KeyError:  # Memoization
        memo_dict = {}  # Memoization
    self._memo['dict'] = memo_dict  # Memoization
    return result
```
else:
    try:
        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  # Memoization

At one point we avoided replicating this sort of logic in all the methods by putting it right into this module, but we’ve moved away from that at present (see the "Historical Note," below.).

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.

Historical Note: The initial Memoizer implementation actually handled the caching of values for the wrapped methods, based on a set of generic algorithms for computing hashable values based on the method’s arguments. This collected caching logic nicely, but had two drawbacks:

Running arguments through a generic key-conversion mechanism is slower (and less flexible) than just coding these things directly. Since the
methods that need memoized values are generally performance-critical, slowing them down in order to collect the logic isn’t the right tradeoff.

Use of the memoizer really obscured what was being called, because all the memoized methods were wrapped with re-used generic methods. This made it more difficult, for example, to use the Python profiler to figure out how to optimize the underlying methods.

12.1 Functions

```python
Dump(title=None)
```

```python
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 2014/03/02 14:18:15 garyo'</td>
</tr>
<tr>
<td><strong>doc</strong></td>
<td>Value: &quot;&quot;&quot;Memo... &quot;&quot;&quot;</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

```python
object
```

```
SCons.Memoize.Counter
```

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

Base class for counting memoization hits and misses.

We expect that the metaclass initialization will have filled in the .name attribute that represents the name of the function being counted.
12.3.1 Methods

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

```python
display(self)
```

```python
__cmp__(self, other)
```

*Inherited from object*

```python
__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 class for each of the class’s methods that memoizes its return value by simply storing the return value in its _memo dictionary.

We expect that the metaclass initialization will fill in the .underlying_method attribute with the method that we’re wrapping. We then call the underlying_method method after counting whether its memoized value has already been set (a hit) or not (a miss).
12.4.1 Methods

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

_Inherited from SCons.Memoize.Counter (Section 12.3)_

```python
__cmp__(), __init__(), display()
```

_Inherited from object_

```python
__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><strong>class</strong></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 class 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.

We expect that the metaclass initialization will fill in the .underlying_method attribute with the method that we’re wrapping. We then call the underlying_method method after counting whether the computed key value is already present in the memoization dictionary (a hit) or not (a miss).
### 12.5.1 Methods

```python
__init__(self, method_name, keymaker)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides:
object.__init__
```

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

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

```python
__cmp__(), display()
```

**Inherited from object**

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

### 12.5.2 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>

### 12.6 Class Memoizer

```python
object

SCons.Memoize.Memoizer
```

Object which performs caching of method calls for its `primary` instance.

#### 12.6.1 Methods

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

**Inherited from object**

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

Module SCons.Memoize

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

12.6.2 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>

12.7 Class Memoized_Metaclass

object

type

SCons.Memoize.Memoized_Metaclass

12.7.1 Methods

```
__init__(cls, name, bases, cls_dict)
```

x.__init__(...) initializes x; see help(type(x)) for signature  
the object’s type

Overrides: object.__init__  
(inherited documentation)

Inherited from type

```
__call__(), __delattr__(), __eq__(), __ge__(), __getattribute__(), __gt__(),
__hash__(), __instancecheck__(), __le__(), __lt__(), __ne__(), __new__(),
__repr__(), __setattr__(), __subclasscheck__(), __subclasses__(), mro()
```

Inherited from object

```
__format__(), __reduce__(), __reduce_ex__(), __sizeof__(), __str__(),
__subclasshook__()
```

12.7.2 Properties

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

continued on next page
## Class Memoized_Metaclass

### Module SCons.Memoize

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...abstractmethods...</td>
<td>Inherited from object</td>
</tr>
<tr>
<td>...base...</td>
<td></td>
</tr>
<tr>
<td>...bases...</td>
<td></td>
</tr>
<tr>
<td>...basicsize...</td>
<td></td>
</tr>
<tr>
<td>...dictoffset...</td>
<td></td>
</tr>
<tr>
<td>...flags...</td>
<td></td>
</tr>
<tr>
<td>...itemsize...</td>
<td></td>
</tr>
<tr>
<td>...mro...</td>
<td></td>
</tr>
<tr>
<td>...name...</td>
<td></td>
</tr>
<tr>
<td>...weakrefoffset...</td>
<td></td>
</tr>
<tr>
<td><strong><strong>class</strong></strong></td>
<td></td>
</tr>
</tbody>
</table>
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

- **Alias**: scons.Node.Alias
  
  *(Section 14, p. 128)*

- **FS**: scons.Node.FS
  
  *(Section 15, p. 135)*

- **Python**: scons.Node.Python
  
  *(Section 16, p. 180)*

13.2 Functions

- `classname(obj)`

- `Annotate(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

```python
__init__(self, node=None)
```

x.__init__(...) initializes x; see help(type(x)) for signature. Overrides: object.__init__(inherited documentation)
convert(self, node, val)

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

merge(self, other)

update(self, node)

Inherited from object

_delattr_(), _format_(), _getattribute_(), _hash_(), _new_(),
_reduce_(), _reduce_ex_(), _repr_(), _setattribute_(), _sizeof_(),
_str_(), _subclasshook_()

13.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Inherited from object
| __class__ |

13.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 1</td>
</tr>
</tbody>
</table>

13.5 Class BuildInfoBase

```
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.
13.5.1 Methods

```python
__init__(self, node=None)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides:
object.__init__ extit(inherited documentation)
```

```python
merge(self, other)
```

*Inherited from object*

```python
__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>
</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: 1</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

```
Decider(self, function)
```
```python
__init__(self)

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

```python
add_dependency(self, depend)

Adds dependencies.
```

```python
add_ignore(self, depend)

Adds dependencies to ignore.
```

```python
add_prerequisite(self, prerequisite)

Adds prerequisites
```

```python
add_source(self, source)

Adds sources.
```

```python
add_to_implicit(self, dep)  
add_to_waiting_parents(self, node)

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...)
```

```python
add_to_waiting_seq(self, node)

```

```python
add_wkid(self, wkid)

Add a node to the list of kids waiting to be evaluated
```
### all_children(self, scan=1)

Return a list of all the node’s direct children.

### alter_targets(self)

Return a list of alternate targets for this Node.

### 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().

### builder_set(self, builder)

### 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()

**changed_since_last_build** *(self, 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.

**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)

### do_not_store_info(self)

### env_set(self, env, safe=0)

### executor_cleanup(self)

Let the executor clean up any cached information.

### exists(self)

Does this node exist?

### 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.
**Class Node**

**get_builder**(self, default_builder=None)

Return the set builder, or a specified default value.

**get_cachedir_csig**(self)

**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.

**get_implicit_deps**(self, env, scanner, path)

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)
**get_source_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.

**get_state**(*self*)

**get_stored_implicit**(*self*)

Fetch the stored implicit dependencies

**get_stored_info**(*self*)

**get_string**(*self, for_signature*)

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.
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.

get_suffix(self)

get_target_scanner(self)

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.

### `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.

### `missing(self)`
**multiple_side_effect_has_builder**\( (sself) \)

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.

**new\_\_binfo**\( (sself) \)

**new\_\_ninfo**\( (sself) \)

**postprocess**\( (sself) \)

Clean up anything we don’t need to hang onto after we’ve been built.

**prepare**\( (sself) \)

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.
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>push_to_cache(self)</code></td>
<td>Try to push a node into a cache</td>
</tr>
<tr>
<td><code>release_target_info(self)</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>remove(self)</code></td>
<td>Remove this Node: no-op by default.</td>
</tr>
<tr>
<td><code>render_include_tree(self)</code></td>
<td>Return a text representation, suitable for displaying to the user, of the include tree for the sources of this node.</td>
</tr>
<tr>
<td><code>reset_executor(self)</code></td>
<td>Remove cached executor; forces recompute when needed.</td>
</tr>
</tbody>
</table>
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.Scanner that’s configured for the target.

set.always_build(self, always_build=1)

Set the Node’s always_build value.

set.executor(self, executor)

Set the action executor for this node.

set.explicit(self, is_explicit)

138
```
set_nocache(self, nocache=1)
```
Set the Node’s nocache value.

```
set_noclean(self, noclean=1)
```
Set the Node’s noclean value.

```
set_precious(self, precious=1)
```
Set the Node’s precious value.

```
set_pseudo(self, pseudo=True)
```
Set the Node’s precious value.

```
set_specific_source(self, source)
```

```
set_state(self, state)
```

```
state_has_changed(self, target, prev_ni)
```

```
store_info(self)
```
Make the build signature permanent (that is, store it in the .sconsign file or equivalent).

```
visited(self)
```
Called just after this node has been visited (with or without a build).

**Inherited from object**

```ruby
__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>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

13.6.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>metaclass</strong></td>
<td>Value: SCons.Memoize.Memoized_Metaclass</td>
</tr>
<tr>
<td>memoizer_counters</td>
<td>Value: []</td>
</tr>
</tbody>
</table>

13.7 Class NodeList

```
object
  _abcoll.Sized
    object
      _abcoll.Iterable
        object
          _abcoll.Container
            _abcoll.Sequence
              _abcoll.MutableSequence
                UserList.UserList
                  SCons.Node.NodeList
```

13.7.1 Methods

```
__str__(self)

str(x) Overrides: object._str_ (inherited documentation)
```

*Inherited from UserList.UserList*
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>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

13.7.3 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>

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

__init__(self, node, kids_func=<function get_children at 0x965802c>, cycle_func=<function ignore_cycle at 0x9658064>, eval_func=<function do_nothing at 0x965809c>)

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

Overrides: object.__init__ (inherited documentation)

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.

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>Inherited from object</td>
<td><strong>class</strong></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 2014/03/02 14:18:15 garyo'</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  ClassAliasNameSpace

UserDict.UserDict  
\[\]
SCons.Node.Alias.AliasNameSpace

14.2.1  Methods

**Alias**(*self*, *name*, **kw*)

**lookup**(*self*, *name*, **kw*)

*Inherited from UserDict.UserDict*

__cmp__(), __contains__(), __delitem__(), __getitem__(), __init__(),
__len__(), __repr__(), __setitem__(), clear(), copy(), fromkeys(), get(), has_key(),
items(), iteritems(), iterkeys(), itervalues(), keys(), pop(), popitem(), setdefault(),
update(), 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>
14.3 Class AliasNodeInfo

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

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

Inherited from SCons.Node.NodeInfoBase(Section 13.4)

```
__init__(), convert(), format(), merge(), update()
```

Inherited from object

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

14.3.2 Properties

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

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: 1</td>
</tr>
<tr>
<td>field_list</td>
<td>Value: ['csig']</td>
</tr>
</tbody>
</table>

144
14.4 Class AliasBuildInfo

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.

14.4.1 Methods

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

__init__(), 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>
<tbody>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

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: 1</td>
</tr>
</tbody>
</table>
14.5 Class Alias

object

SCons.Node.Node


14.5.1 Methods

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

```python
str_for_display(self)
```

```python
__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. Override:
SCons.Node.Node.make_ready extit(inherited documentation)

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().

146
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.

sconsign(self)

An Alias is not recorded in .sconsign files

changed_since_last_build(self, 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. Overrides: SCons.Node.Node.changed_since_last_build extit(inherited documentation)

build(self)

**Class Alias Module SCons.Node.Alias**

**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(), 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(), children(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(), disambiguate(), do_not_store_info(), env_set(), executor_cleanup(), exists(), explain(),
for_signature(), get_abspath(), get_binfo(), get_build_env(), get_build_scanner_path(),
get_builder(), get_cachepath(), 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(), 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(), state_has_changed(), store_info(), 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><strong>Inherited from object</strong></td>
<td></td>
</tr>
</tbody>
</table>

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

14.5.3 Class Variables
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Inherited from SCons.Node.Node (Section 13.6)</em></td>
<td><strong>metaclass</strong>, memoizer_counts</td>
</tr>
</tbody>
</table>
15 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

```plaintext
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)
```

get_MkdirBuilder()

get_DefaultSCCSBuilder()

get_DefaultRCSBuilder()
### Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>do_diskcheck_match(node, predicate, errorfmt)</code></td>
<td>Function to match a diskcheck.</td>
</tr>
<tr>
<td><code>ignore_diskcheck_match(node, predicate, errorfmt)</code></td>
<td>Ignore matching for diskcheck.</td>
</tr>
<tr>
<td><code>do_diskcheck_rcs(node, name)</code></td>
<td>Diskcheck function for RCS.</td>
</tr>
<tr>
<td><code>ignore_diskcheck_rcs(node, name)</code></td>
<td>Ignore diskcheck for RCS.</td>
</tr>
<tr>
<td><code>do_diskcheck_sccs(node, name)</code></td>
<td>Diskcheck function for SCCS.</td>
</tr>
<tr>
<td><code>ignore_diskcheck_sccs(node, name)</code></td>
<td>Ignore diskcheck for SCCS.</td>
</tr>
<tr>
<td><code>set_diskcheck(list)</code></td>
<td>Set diskcheck.</td>
</tr>
<tr>
<td><code>diskcheck_types()</code></td>
<td>Diskcheck types function.</td>
</tr>
<tr>
<td><code>has_glob_magic(s)</code></td>
<td>Check for glob magic in the filename.</td>
</tr>
<tr>
<td><code>get_default_fs()</code></td>
<td>Get the default file system.</td>
</tr>
</tbody>
</table>

#### `find_file(filename, paths, verbose=None)`

Finds 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.
- **verbose**: A boolean indicating whether to print additional output.

Returns the node created from the found file.
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.

### 15.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: ‘`src/engine/SCons/Node/FS.py 2014/03/02 14:18:15 garyo’</td>
</tr>
<tr>
<td>do_store_info</td>
<td>Value: True</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).<em>/.(.</em>)?./.(?:[/:</td>
</tr>
<tr>
<td>Link_Funcs</td>
<td>Value: []</td>
</tr>
<tr>
<td>Link</td>
<td>Value: <code>SCons.Action.Action(LinkFunc, None)</code></td>
</tr>
<tr>
<td>LocalCopy</td>
<td>Value: <code>SCons.Action.Action(LinkFunc, LocalString)</code></td>
</tr>
<tr>
<td>Unlink</td>
<td>Value: <code>SCons.Action.Action(UnlinkFunc, None)</code></td>
</tr>
<tr>
<td>Mkdir</td>
<td>Value: <code>SCons.Action.Action(MkdirFunc, None, presub= None)</code></td>
</tr>
<tr>
<td>MkdirBuilder</td>
<td>Value: None</td>
</tr>
<tr>
<td>DefaultSCCSBuilder</td>
<td>Value: None</td>
</tr>
<tr>
<td>DefaultRCSBuilder</td>
<td>Value: None</td>
</tr>
<tr>
<td>diskcheck_match</td>
<td>Value: `DiskChecker(‘match’, do_diskcheck_match, ignore_diskcheck...</td>
</tr>
<tr>
<td>diskcheck_rcs</td>
<td>Value: <code>DiskChecker(‘rcs’, do_diskcheck_rcs, ignore_diskcheck_rcs)</code></td>
</tr>
</tbody>
</table>

*continued on next page*
### 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

**__init__(self, entry_proxy, attribute)**

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

**__str__(self)**

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

**Module SCons.Node.FS**

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

15.4 Class DiskChecker

```
object —
SCons.Node.FS.DiskChecker
```

15.4.1 Methods

```python
__init__(self, type, do, ignore)
```

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

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

```
set(self, list)
```

Inherited from `object`

```python
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```
15.4.2 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>

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__

___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__/ex
init(inherited documentation)

Inherited from SCons.Util.Proxy (Section 36.5)

___cmp__(), ___init__(), get()

Inherited from object

___delattr__(), ___format__(), ___getattribute__(), ___hash__(), ___new__(), ___reduce__(), ___reduce_ex__(), ___repr__(), ___setattr__(), ___sizeof__(), ___subclasshook__()

15.5.2 Properties
### 15.5.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dictSpecialAttrs</td>
<td>Value: {&quot;base&quot;: __get_base_path, &quot;posix&quot;: __get_posix_path, &quot;win...}</td>
</tr>
</tbody>
</table>

### 15.6 Class Base

**object**


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)

__str__(self)

A Node.FS.Base object’s string representation is its path name. Overrides: object.__str__

rstr(self)

A Node.FS.Base object’s string representation is its path name.

stat(self)

exists(self)

Does this node exists? Overrides: SCons.Node.Node.exists extit(inherited documentation)

rexists(self)

Does this node exist locally or in a repository? Overrides: SCons.Node.Node.rexists extit(inherited documentation)

getmtime(self)

getsize(self)

isdir(self)
**isfile**(self)

**islink**(self)

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

**set_local**(self)

**srcnode**(self)

If this node is in a build path, return the node corresponding to its source file. Otherwise, return ourself.

**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.

**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)

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=<function splitext at 0x95d96bc>)

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** *(self, pathlist)*

Search for a list of directories in the Repository list.

**rentry** *(self)*

**Inherited from SCons.Node.Node (Section 13.6)**

Decider(), 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(), changed_since_last_build(), children(), children_are_up_to_date(), clear(), clear_memorized_values(), del_binfo(), disambiguate(), do_not_store_info(), 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(), 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_executor(), set_explicit(), set_nocache(), set_noclean(), set_precious(), set_pseudo(), set_specific_source(), set_state(), state_has_changed(), store_info(), 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><strong>class</strong></td>
<td>Inherited from object</td>
</tr>
</tbody>
</table>

**15.6.3 Class Variables**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memoizer_counters</td>
<td>Value: []</td>
</tr>
</tbody>
</table>

**Inherited from SCons.Node.Node (Section 13.6)**

__metaclass__
15.6.4 Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Filename with extension as it was specified when the object was created; to obtain filesystem path, use Python str() function</td>
</tr>
<tr>
<td>suffix</td>
<td>Cached filename extension</td>
</tr>
<tr>
<td>fs</td>
<td>Reference to parent Node.FS object</td>
</tr>
</tbody>
</table>

15.7 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.

15.7.1 Methods

**diskcheck_match**(self)

**disambiguate**(self, must_exist=None)

**rfile**(self)

We’re a generic Entry, but the caller is actually looking for a File at this point, so morph into one. Overrides: SCons.Node.FS.Base.rfile

**scanner_key**(self)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get_contents(self)</code></td>
<td>Fetch the contents of the entry. Returns the exact binary contents of the file.</td>
</tr>
<tr>
<td><code>get_text_contents(self)</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>must_be_same(self, klass)</code></td>
<td>Called to make sure a Node is a Dir. Since we’re an Entry, we can morph into one. Overrides: SCons.Node.FS.Base.must_be_same</td>
</tr>
<tr>
<td><code>exists(self)</code></td>
<td>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. Overrides: SCons.Node.Node.exists</td>
</tr>
<tr>
<td><code>rel_path(self, other)</code></td>
<td></td>
</tr>
</tbody>
</table>
changed_since_last_build (self, 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. Overrides: SCons.Node.Node.changed_since_last_build extit(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 extit(inherited documentation)

Inherited from SCons.Node.FS.Base(Section 15.6)

RDirs(), Rfindalldirs(), __init__(), __str__(), for_signature(), get_abspath(),
get_dir(), get_path(), get_suffix(), getmtime(), getsize(), is_under(), isdir(), isfile(), islink(), rentry(), rexists(), rstr(), set_local(), set_src_builder(), src_builder(),
srnode(), stat(), str_for_display(), target_from_source()

Inherited from SCons.Node.Node(Section 13.6)

Decider(), 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_memorized_values(), del_binfo(), do_not_store_info(),
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(), is_derived(), is_literal(), is_up_to_date(),
make_ready(), missing(), multiple_side_effect_has_builder(), new_binfo(), post-
process(), 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_precious(), set_pseudo(),
set_specific_source(), set_state(), state_has_changed(), store_info(), visited()

**Inherited from object**

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>chmod</td>
<td><code>chmod(self, path, mode)</code></td>
</tr>
<tr>
<td>copy</td>
<td><code>copy(self, src, dst)</code></td>
</tr>
<tr>
<td>copy2</td>
<td><code>copy2(self, src, dst)</code></td>
</tr>
<tr>
<td>exists</td>
<td><code>exists(self, path)</code></td>
</tr>
<tr>
<td>getmtime</td>
<td><code>getmtime(self, path)</code></td>
</tr>
<tr>
<td>getsize</td>
<td><code>getsize(self, path)</code></td>
</tr>
<tr>
<td>isdir</td>
<td><code>isdir(self, path)</code></td>
</tr>
<tr>
<td>isfile</td>
<td><code>isfile(self, path)</code></td>
</tr>
<tr>
<td>link</td>
<td><code>link(self, src, dst)</code></td>
</tr>
<tr>
<td>lstat</td>
<td><code>lstat(self, path)</code></td>
</tr>
<tr>
<td>listdir</td>
<td><code>listdir(self, path)</code></td>
</tr>
<tr>
<td>makedirs</td>
<td><code>makedirs(self, path)</code></td>
</tr>
<tr>
<td>mkdir</td>
<td><code>mkdir(self, path)</code></td>
</tr>
<tr>
<td>rename</td>
<td><code>rename(self, old, new)</code></td>
</tr>
<tr>
<td>stat</td>
<td><code>stat(self, path)</code></td>
</tr>
<tr>
<td>symlink</td>
<td><code>symlink(self, src, dst)</code></td>
</tr>
<tr>
<td>open</td>
<td><code>open(self, path)</code></td>
</tr>
<tr>
<td>unlink</td>
<td><code>unlink(self, path)</code></td>
</tr>
<tr>
<td>islink</td>
<td><code>islink(self, path)</code></td>
</tr>
</tbody>
</table>
readlink(self, file)

Inherited from object

__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>

15.8.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>metaclass</strong></td>
<td>Value: SCons.Memoize.Memoized_Metaclass</td>
</tr>
</tbody>
</table>

15.9 Class FS

object

SCons.Node.FS.LocalFS

SCons.Node.FS.FS

15.9.1 Methods

__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:
object.__init__

set_SConstruct_dir(self, dir)
get_max_drift(self)

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.
**Class FS**

**Module SCons.Node.FS**

**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.

**VariantDir**(*self*, *variant_dir*, *src_dir*, *duplicate=1*)

Link the supplied variant directory to the source directory for purposes of building files.

**Repository**(*self*, *dirs*)

Specify Repository directories to search.

**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**(*self*, *pathname*, *ondisk=True*, *source=True*, *strings=False*, *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(), makedirs(), mkdir(), open(), readlink(), rename(), stat(),
symlink(), unlink()

_Inherited from object_

__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>

15.9.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memoizer_counters</td>
<td>Value: []</td>
</tr>
</tbody>
</table>

_Inherited from SCons.Node.FS.LocalFS (Section 15.8)___metaclas___

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

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

_Inherited from SCons.Node.NodeInfoBase(Section 13.4)___init__(), convert(), format(), merge(), update()_

_Inherited from object_
Class DirBuildInfo

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>

Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 1</td>
</tr>
<tr>
<td>fs</td>
<td>Value: None</td>
</tr>
</tbody>
</table>

15.11 Class 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)

__init__(), merge()

Inherited from object

__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 object</td>
<td></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: 1</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)
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entry</strong>(self, name)</td>
<td>Looks up or creates an entry node named ‘name’ relative to this directory.</td>
</tr>
<tr>
<td><strong>Dir</strong>(self, name, create=True)</td>
<td>Looks up or creates a directory node named ‘name’ relative to this directory.</td>
</tr>
<tr>
<td><strong>File</strong>(self, name)</td>
<td>Looks up or creates a file node named ‘name’ relative to this directory.</td>
</tr>
<tr>
<td><strong>link</strong>(self, srcdir, duplicate)</td>
<td>Set this directory as the variant directory for the supplied source directory.</td>
</tr>
<tr>
<td><strong>getRepositories</strong>(self)</td>
<td>Returns a list of repositories for this directory.</td>
</tr>
<tr>
<td><strong>get_all_rdirs</strong>(self)</td>
<td></td>
</tr>
<tr>
<td><strong>addRepository</strong>(self, dir)</td>
<td></td>
</tr>
<tr>
<td><strong>up</strong>(self)</td>
<td></td>
</tr>
<tr>
<td><strong>rel_path</strong>(self, other)</td>
<td>Return a path to “other” relative to this directory.</td>
</tr>
</tbody>
</table>
### get_target_scanner

*Overrides: SCons.Node.Node.get_target_scanner*

```python
get_target_scanner(self)
```

### get_found_includes

*Overrides: SCons.Node.Node.get_found_includes*

```python
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).

### prepare

*Overrides: SCons.Node.Node.prepare extit(inherited documentation)*

```python
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 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.

### build

*Overrides: SCons.Node.Node.build*

```python
build(self, **kw)
```
**Class Dir**

**Module SCons.Node.FS**

```python
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

```python
alter_targets(self)
```


```python
scanner_key(self)
```


```python
get_text_contents(self)
```

We already emit things in text, so just return the binary version.

```python
get_contents(self)
```

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

```python
get_csig(self)
```

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
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>do_duplicate</strong>(self, src)**</td>
<td></td>
</tr>
<tr>
<td><strong>changed_since_last_build</strong>(self, target, prev_ni)**</td>
<td>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. Overrides: SCons.Node.Node.changed_since_last_build extit(inherited documentation)</td>
</tr>
<tr>
<td><strong>is_up_to_date</strong>(self)**</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>
</tr>
<tr>
<td><strong>rdir</strong>(self)**</td>
<td></td>
</tr>
<tr>
<td><strong>sconsign</strong>(self)**</td>
<td>Return the .sconsign file info for this directory, creating it first if necessary.</td>
</tr>
<tr>
<td><strong>srcnode</strong>(self)**</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>
</tr>
<tr>
<td><strong>get_timestamp</strong>(self)**</td>
<td>Return the latest timestamp from among our children</td>
</tr>
</tbody>
</table>
entry_abspath(self, name)

entry_labspath(self, name)

entry_path(self, name)

entry_tpath(self, name)

entry_exists_on_disk(self, name)

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).
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 `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 `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 `VariantDir()`). The default behavior is to return Nodes local to the `VariantDir()`.

The “strings” argument, when true, returns the matches as strings, not Nodes. The strings are path names relative to this directory.

The underlying algorithm is adapted from the `glob.glob()` function in the Python library (but heavily modified), and uses `fnmatch()` under the covers.

**Inherited from SCons.Node.FS.Base (Section 15.6)**

RDdirs(), Rfindalldirs(), __str__(), exists(), for_signature(), get_abspath(), get_dir(), get_path(), get_subst_proxy(), get_suffix(), getmtime(), getsize(), is_under(), isdir(), isfile(), islink(), must_be_same(), rentry(), reexists(), rfile(), rstr(), set_local(), set_src_builder(), src_builder(), stat(), str_for_display(), target_from_source()
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(),
state_has_changed(), store__info(), visited()

**Inherited from object**

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

15.12.2 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><strong>class</strong></strong></td>
<td></td>
</tr>
</tbody>
</table>

15.12.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memoizer__counters</td>
<td>Value: []</td>
</tr>
<tr>
<td>Inherited from SCons.Node.Node (Section 13.6)</td>
<td></td>
</tr>
<tr>
<td><strong><strong>metaclass</strong></strong></td>
<td></td>
</tr>
</tbody>
</table>

15.12.4 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>
<tr>
<td>fs, name, suffix</td>
<td></td>
</tr>
</tbody>
</table>

15.13 Class RootDir

object

SCons.Node.Node

SCons.Node.FS.Base

SCons.Node.FS.Dir

SCons.Node.FS.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)

```python
__str__(self)
```

A Node.FS.Base object’s string representation is its path name. Overrides: object.__str__ (inherited documentation)

```python
entry_abspath(self, name)
```

Overrides: SCons.Node.FS.Dir.entry_abspath

```python
entry_labspath(self, name)
```

Overrides: SCons.Node.FS.Dir.entry_labspath

```python
entry_path(self, name)
```

Overrides: SCons.Node.FS.Dir.entry_path

```python
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(), changed_since_last_build(), dir_on_disk(), diskcheck_match(), do_duplicate(), entry_exists_on_disk(), file_on_disk(), getRepositories(), get_all_rdirs(), get_contents(), get_csig(), get_env_scanner(), get_found_includes(), get_target_scanner(), get_text_contents(), get_timestamp(), glob(), is_up_to_date(), link(), multiple_side_effect_has_builder(), prepare(), rdir(), rel_path(), scanner_key(), sconsign(), srcdir_duplicate(), srcdir_find_file(), srcdir_list(), srcnode(), walk()

Inherited from SCons.Node.FS.Base (Section 15.6)

RDirs(), Rfindalldirs(), exists(), for_signature(), get_abspath(), get_path(), get_subst_proxy(), get_suffix(), getmtime(), getsize(), isdir(), isfile(), islink(), reentry(), reexists(), rfile(), rstr(), set_local(), set_src_builder(), stat(), str_for_display(), target_from_source()

Inherited from SCons.Node.Node (Section 13.6)

Decider(), add_dependency(), add_ignore(), add_prerequisite(), add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_sc_e(), add_wkid(), all_children(), builder_set(), built(), changed(), children(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(), disambiguate(), do_not_store_info(), env_set(), executor_cleanup(), explain(), get_binfo(), get_build_env(), get_build_scanner_path(), get_builder(), get_cachepath(), 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(), state_has_changed(), store_info(), 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 object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

15.13.3 Class Variables

<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>memoizer_counters</td>
<td></td>
</tr>
<tr>
<td>Inherited from SCons.Node.Node (Section 13.6)</td>
<td></td>
</tr>
<tr>
<td><strong>metaclass</strong></td>
<td></td>
</tr>
</tbody>
</table>

15.13.4 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>
<tr>
<td>fs, name, suffix</td>
<td></td>
</tr>
</tbody>
</table>

15.14 Class FileNodeInfo

object

SCons.Node.NodeInfoBase

SCons.Node.FS.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)
```

*Inherited from SCons.Node.NodeInfoBase (Section 13.4)*

- `__init__()`, `convert()`, `format()`, `merge()`, `update()`

*Inherited from object*

- `__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__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><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

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: 1</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

```python
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.

```python
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.

```python
format(self, names=0)
```

```python
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)`

```python
__init__(), merge()
```

`Inherited from object`

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

183
15.15.2 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>

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: 1</td>
</tr>
</tbody>
</table>

15.16 Class File

object →
SCons.Node.Node →
SCons.Node.FS.Base →
SCons.Node.FS.File

A class for files in a file system.

15.16.1 Methods

```python
@overload
def diskcheck_match(self):
    pass
```

```python
@overload
def __init__(self, name, directory, fs):
    pass
```

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
def Entry(self, name):
    pass
```

Create an entry node named 'name' relative to the directory of this file.
Dir(self, name, create=True)

Create a directory node named 'name' relative to the directory of this file.

Dirs(self, pathlist)

Create a list of directories relative to the SConscript directory of this file.

File(self, name)

Create a file node named 'name' relative to the directory of this file.

scanner_key(self)


get_contents(self)

get_text_contents(self)

get_content_hash(self)

Compute and return the MD5 hash for this file.

get_size(self)

get_timestamp(self)

store_info(self)

Make the build signature permanent (that is, store it in the .sconsign file or equivalent). Overrides: SCons.Node.Node.store_info (inherited documentation)

convert_old_entry(self, old_entry)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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(). 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). Overrides: SCons.Node.Node.visited (inherited documentation)</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 *.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

**find_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)**

**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 exitt(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 exitt(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`  

**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()  

**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)*
**decide_source**(self, 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.

**decide_target**(self, target, prev_ni)

**changed_since_last_build**(self, 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. Overrides: SCons.Node.Node.changed_since_last_build (inherited documentation)

**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 (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(), __str__(), for_signature(), get_abspath(), get_dir(), get_path(), get_subst_proxy(), get_suffix(), getmtime(), getsize(), is_under(), isdir(), isfile(), islink(), must_be_same(), rentry(), reexists(), set_local(), set_src_builder(), src_builder(), ssrcnode(), stat(), str_for_display(), target_from_source()

*Inherited from SCons.Node.Node (Section 13.6)*

Decider(), add_dependency(), add_ignore(), add_prerequisite(), add_source(),
add_to_implicit(), add_to_waiting_parents(), add_to_waiting_s_e(), add_wkid(),
all_children(), build(), children(), children_are_up_to_date(), clear(), clear_memoized_values(),
derel_bininfo(), disambiguate(), do_not_store_info(), env_set(), executor_cleanup(),
explain(), get_bininfo(), 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_bininfo(), 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(),
state_has_changed()

**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>Inherited from object</td>
<td></td>
</tr>
</tbody>
</table>

### 15.16.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memoizer_counters</td>
<td>Value: []</td>
</tr>
<tr>
<td>md5_chunksize</td>
<td>Value: 64</td>
</tr>
<tr>
<td>convert_copy_attrs</td>
<td>Value: ['bsources', 'bimplicit', 'bdepends', 'bact', 'bactsig', ...]</td>
</tr>
<tr>
<td>convert_sig_attrs</td>
<td>Value: ['bsourcesigs', 'bimplicitsigns', 'bdependsigs']</td>
</tr>
</tbody>
</table>

**Inherited from SCons.Node.Node (Section 13.6)**

| __metaclass__      |             |

### 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>
<tr>
<td>fs, name, suffix</td>
<td></td>
</tr>
</tbody>
</table>

192
15.17   Class FileFinder

```
object  ----
        SCons.Node.FS.FileFinder
```

15.17.1  Methods

```
___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_file(str, [Dir()]) -> [nodes]

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.

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.
```

Class FileFinder

Module SCons.Node.FS

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>

15.17.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>metaclass</strong></td>
<td>Value: SCons.Memoize.Memoized_Metaclass</td>
</tr>
<tr>
<td>memoizer_counters</td>
<td>Value: []</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 2014/03/02 14:18:15 garyo'</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

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

Inherited from SCons.Node.NodeInfoBase(Section 13.4)

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

Inherited from object

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

16.2.2 Properties
Class ValueBuildInfo

<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>

### 16.2.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 1</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)*

```
__init__(), 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>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

196
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: 1</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

```python
__init__(self, value, built_value=None)
```

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

```python
str_for_display(self)
```

```python
__str__(self)
```

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

```python
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)
**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.

**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)**

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.

**changed_since_last_build(self, 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. Overrides: SCons.Node.Node.changed_since_last_build extit(inherited documentation)

**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(), 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(), children(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(), disambiguate(), do_not_store_info(), env_set(), executor_cleanup(), exists(), explain(), for_signature(), get_abspath(), get_binfo(), get_build_env(), get_buildScannerPath(), get_builder(), get_cachefile(), get_contents(), get_contents(self, target, prev_ni), get_csig(), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csig(self, calc=None), get_csg
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_no_clean(), set_precious(), set_pseudo(), set_specific_source(), set_state(), state_has_changed(), store_info(), visited()

**Inherited from object**

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

### 16.4.2 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>

### 16.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.Node (Section 13.6)</td>
<td>metaclas, memoizer_counters</td>
</tr>
</tbody>
</table>
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

\texttt{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.

\texttt{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 2014/03/02 14:18:15 garyo’</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.

18.1 Functions

**SetBuildType**(type)

**SetCacheMode**(mode)

Set the Configure cache mode. mode must be one of “auto”, “force”, or “cache”.

**SetProgressDisplay**(display)

Set the progress display to use (called from SCons.Script)

**CreateConfigHBuilder**(env)

Called 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)

**CheckTypeSize**(context, type_name, includes='', language=None, expect=None)

**CheckDeclaration**(context, declaration, includes='', language=None)

**createIncludesFromHeaders**(headers, leaveLast, include_quotes='')
### Variables

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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></td>
</tr>
<tr>
<td><code>CheckCXX(context)</code></td>
<td></td>
</tr>
<tr>
<td><code>CheckSHCC(context)</code></td>
<td></td>
</tr>
<tr>
<td><code>CheckSHCXX(context)</code></td>
<td></td>
</tr>
<tr>
<td><code>CheckCHeader(context, header, include_quotes='&quot;&quot;')</code></td>
<td>A test for a C header file.</td>
</tr>
<tr>
<td><code>CheckCXXHeader(context, header, include_quotes='&quot;&quot;')</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>
<tr>
<td><code>CheckLibWithHeader(context, libs, header, language, call=None, autoadd=1)</code></td>
<td>Another (more sophisticated) test for a library. Checks, if library and header is available for language (may be 'C' or 'CXX'). Call maybe be a valid expression <em>with</em> a trailing ';' . As in CheckLib, we support library=None, to test if the call compiles without extra link flags.</td>
</tr>
</tbody>
</table>

#### 18.2 Variables
Class `SConfWarning` Module `SCons.SConf`

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__revision__</code></td>
<td>Value: ‘src/engine/SCons/SConf.py 2014/03/02 14:18:15 garyo’</td>
</tr>
<tr>
<td><code>build_type</code></td>
<td>Value: None</td>
</tr>
<tr>
<td><code>build_types</code></td>
<td>Value: [‘clean’, ‘help’]</td>
</tr>
<tr>
<td><code>dryrun</code></td>
<td>Value: 0</td>
</tr>
<tr>
<td><code>AUTO</code></td>
<td>Value: 0</td>
</tr>
<tr>
<td><code>FORCE</code></td>
<td>Value: 1</td>
</tr>
<tr>
<td><code>CACHE</code></td>
<td>Value: 2</td>
</tr>
<tr>
<td><code>cache_mode</code></td>
<td>Value: 0</td>
</tr>
<tr>
<td><code>progress_display</code></td>
<td>Value: <code>DisplayEngine()</code></td>
</tr>
<tr>
<td><code>SConfFS</code></td>
<td>Value: None</td>
</tr>
<tr>
<td><code>sconf_global</code></td>
<td>Value: None</td>
</tr>
<tr>
<td><code>__package__</code></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__(), __subclasshook__()

18.3.2 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>

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

```python
__init__(self, msg)
```

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

**Inherited from object**

```python
__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

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.SConf.SConfError

SCons.SConf.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)
```

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

Inherited from object
18.5.2 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>

18.6 Class ConfigureCacheError

Raised when a user 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 Overrites:
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__()
```
18.6.2 Properties

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

18.7 Class 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
def 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)

- `__init__()`, `merge()`

Inherited from `object`

- `__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>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

18.7.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td>Value: None</td>
</tr>
<tr>
<td>string</td>
<td>Value: None</td>
</tr>
<tr>
<td></td>
<td>Inherited from SCons.Node.FS.FileBuildInfo (Section 15.15)</td>
</tr>
<tr>
<td>current_version_id</td>
<td></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

___init___(self, orig)

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

write(self, str)

writelines(self, lines)

getvalue(self)

Return everything written to orig since the Streamer was created.

flush(self)

Inherited from object
Encryption

<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>

### 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 actually 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)

Log 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 extit(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 extit(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()

Inherited from object

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

18.9.2 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>
18.10 Class SConfBase

object __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 /* and */ will be put automatically.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BuildNodes</strong></td>
<td>Tries to build the given nodes immediately. Returns 1 on success, 0 on error.</td>
</tr>
<tr>
<td><strong>pspawn_wrapper</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>TryBuild</strong></td>
<td>Low level TryBuild implementation. Normally you don’t need to call that - you can use TryCompile / TryLink / TryRun instead.</td>
</tr>
<tr>
<td><strong>TryAction</strong></td>
<td>Tries to execute the given action with optional source file contents &lt;text&gt; and optional source file extension &lt;extension&gt;, Returns the status (0 : failed, 1 : ok) and the contents of the output file.</td>
</tr>
<tr>
<td><strong>TryCompile</strong></td>
<td>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).</td>
</tr>
</tbody>
</table>
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__(), __getattribute__(), __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>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
18.11 Class 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:

```python
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__init__(self, sconf)</code></td>
<td>Constructor. Pass the corresponding SConf instance. Overrides: object.<strong>init</strong></td>
</tr>
<tr>
<td><code>Message(self, text)</code></td>
<td>Inform about what we are doing right now, e.g. 'Checking for SOMETHING ... '</td>
</tr>
<tr>
<td><code>Result(self, res)</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>TryBuild(self, *args, **kw)</code></td>
<td></td>
</tr>
<tr>
<td><code>TryAction(self, *args, **kw)</code></td>
<td></td>
</tr>
<tr>
<td><code>TryCompile(self, *args, **kw)</code></td>
<td></td>
</tr>
</tbody>
</table>
TryLink(self, *args, **kw)

TryRun(self, *args, **kw)

__getattr__(self, attr)

BuildProg(self, text, ext)

CompileProg(self, text, ext)

CompileSharedObject(self, text, ext)

RunProg(self, text, ext)

AppendLIBS(self, lib_name_list)

PrependLIBS(self, lib_name_list)

SetLIBS(self, val)

Display(self, msg)

Log(self, msg)

Inherited from object

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

18.11.2 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>
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><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/SConsign.py 2014/03/02 14:18:15 garyo',</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: '.sconsign'</td>
</tr>
<tr>
<td>DB_sync_list</td>
<td>Value: []</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</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__ extit(inherited documentation)
```

```python
convert_to_sconsign(self)
```

```python
convert_from_sconsign(self, dir, name)
```

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>inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

19.3.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 1</td>
</tr>
</tbody>
</table>
19.4 Class Base

object  
\[ \text{SCons.SConsign.Base} \]

**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)
```

*Inherited from object*

___delattr___(), ___format___(), ___getattr___(), ___hash___(), ___new___(),
___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(),
___str___(), ___subclasshook___()
19.4.2 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>

19.5 Class DB

object

SCons.SConsign.Base

SCons.SConsign.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.

19.5.1 Methods

```python
__init__(self, dir)
```

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

```python
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

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

19.5.2 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>
19.6 Class Dir

object ─┐
SCons.SConsign.Base ─┐
              SCons.SConsign.Dir

**Known Subclasses:** SCons.SConsign.DirFile

19.6.1 Methods

```
__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>Inherited from object</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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>init</strong>(self, <em>dir</em>)</td>
<td>dir - the directory for the file. Overrides: object.<strong>init</strong></td>
</tr>
<tr>
<td><strong>write</strong>(self, <em>sync=1</em>)</td>
<td>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.</td>
</tr>
</tbody>
</table>

*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__()

### 19.7.2 Properties

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

*Inherited from object*

- __class__
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

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

```python
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

```python
__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>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>


## 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. 225)*
- **D**: SCons.Scanner.D  
  *(Section 22, p. 228)*
- **Dir** *(Section 23, p. 232)*
- **Fortran**: SCons.Scanner.Fortran  
  *(Section 24, p. 234)*
- **IDL**: SCons.Scanner.IDL  
  *(Section 25, p. 239)*
- **LaTeX**: SCons.Scanner.LaTeX  
  *(Section 26, p. 240)*
- **Prog** *(Section 27, p. 247)*
- **RC**: SCons.Scanner.RC  
  *(Section 28, p. 248)*

### 20.2 Functions

\[
\text{Scanner}(\text{function}, *\text{args}, **\text{kw})
\]

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| __revision__ | Value:  
'src/engine/SCons/Scanner/__init__.py  
2014/03/02 14:18:1... |

*continued on next page*
20.4 Class FindPathDirs

A class to bind a specific *PATH variable name to a function that will return all of the *path directories.

20.4.1 Methods

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

Overrides:
```python
object.__init__ extit(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__()
```

20.4.2 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>

20.5 Class 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.

___cmp___(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:

```python
s = Scanner(my_scanner_function)

s = Scanner(function = my_scanner_function)
```
def __str__(self):
    return 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)

recurse_nodes(self, nodes)

select(self, node)

Inherited from object

def __delattr__(self, name):
def __format__(self, format_spec):
def __getattribute__(self, name):
def __new__(cls, *args, **kwds):
def __reduce__(self):
def __reduce_ex__(self, _):
def __repr__(self):
def __setattr__(self, name, value):
def __sizeof__(self):
def __subclasshook__(self):

20.5.2 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>

20.6 Class Selector

object  

SCons.Scanner.Base  

SCons.Scanner.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 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:

```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)
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__ (inherited documentation)

Overrides: SCons.Scanner.Base.select

Overrides: SCons.Scanner.Base.add_scanner

**Inherited from SCons.Scanner.Base (Section 20.5)**

___cmp__(), ___hash__(), ___str__(), add_skey(), get_skeys(), path(), recurse_nodes()

**Inherited from object**

___delattr__(), ___format__(), getattr(), new(), reduce(), reduce_ex(), repr(), setattr(), sizeof(), subclasshook()
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__()`, `__cmp__()`, `__hash__()`, `__str__()`, `add_scanner()`, `add_skey()`, `get_skeys()`, `path()`, `recurse_nodes()`, `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><code>Inherited from object</code></td>
<td></td>
</tr>
<tr>
<td><code>__class__</code></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')
```

Overrides: object.__init__ (inherited documentation)
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__(), __cmp__(), __hash__(), __str__(), add_scanner(), add_skey(),
get_skeys(), path(), recurse_nodes(), 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

```python
find__include(self, include, source_dir, path)
```
Overrides: SCons.Scanner.Classic.find__include

```python
sort__key(self, include)
```
Overrides: SCons.Scanner.Classic.sort__key

**Inherited from SCons.Scanner.Classic (Section 20.8)**

```python
__init__(), find__include__names(), scan()
```

**Inherited from SCons.Scanner.Base (Section 20.5)**

```python
__call__(), __cmp__(), __hash__(), __str__(), add__scanner(), add__skey(),
get__skeys(), path(), recurse__nodes(), select()
```

**Inherited from object**

```python
__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>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dictify_CPPDEFINES(env)</code></td>
<td></td>
</tr>
</tbody>
</table>

`CScanner()`

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>
<tbody>
<tr>
<td><code>__revision__</code></td>
<td>Value: 'src/engine/SCons/Scanner/C.py'</td>
</tr>
<tr>
<td></td>
<td>2014/03/02 14:18:15 garyo'</td>
</tr>
<tr>
<td><code>__package__</code></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>

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__(),
save__(), scons__current__file__(), start__handling__includes__(), stop__handling__includes__(),
tupleize()

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>
</table>

Inherited from object

__class__
21.4 Class SConsCPPScannerWrapper

object

SCons.Scanner.C.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

\_\_\_init\_\_(self, name, variable)

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

\_\_\_call\_\_(self, node, env, path=())

recurse\_nodes(self, nodes)

select(self, node)

Inherited from object

\_\_\_del\_\_attr\_\_(), \_\_\_format\_\_(), \_\_\_get\_\_attribute\_\_(), \_\_\_hash\_\_(), \_\_\_new\_\_(),
\_\_\_reduce\_\_(), \_\_\_reduce\_\_\_ex\_\_(), \_\_\_repr\_\_(), \_\_\_set\_\_attr\_\_(), \_\_\_sizeof\_\_(),
\_\_\_str\_\_(), \_\_\_subclass\_\_hook\_\_() 

21.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Inherited from object
\_\_\_class\_\_ | |
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 2014/03/02 14:18:15 garyo’</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

__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') Overrides:
find\_include(self, include, source\_dir, path)
Overrides: SCons.Scanner.Classic.find\_include

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__(), __cmp__(), __hash__(), __str__(), add\_scanner(), add\_skey(),
get\_skeys(), path(), recurse\_nodes(), 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></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
23  Module SCons.Scanner.Dir

23.1  Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>only_dirs(nodes)</td>
<td></td>
</tr>
<tr>
<td>DirScanner(<strong>kw</strong>)</td>
<td>Return a prototype Scanner instance for scanning directories for on-disk files</td>
</tr>
<tr>
<td>DirEntryScanner(<strong>kw</strong>)</td>
<td>Return a prototype Scanner instance for “scanning” directory Nodes for their in-memory entries</td>
</tr>
<tr>
<td>do_not_scan(k)</td>
<td></td>
</tr>
<tr>
<td>scan_on_disk(node, env, path=())</td>
<td>Scans a directory for on-disk files and directories therein.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>scan_in_memory(node, env, path=())</td>
<td>“Scans” a Node.FS.Dir for its in-memory entries.</td>
</tr>
</tbody>
</table>

23.2  Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| __revision__| Value: `src/engine/SCons/Scanner/Dir.py
2014/03/02 14:18:15 garyo’ |
| skip_entry  | Value: `{'.': 1, '..', ': 1, '.sconsign': 1, '.sconsign.bak': 1, '..s...` |

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', '.sconsign.dblite', '.sconsign.db']</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

\begin{verbatim}
FortranScan(path_variable=\texttt{\textasciitilde FORTRANPATH})
\end{verbatim}

Return a prototype Scanner instance for scanning source files for Fortran USE & INCLUDE statements

24.2 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/Fortran.py</td>
</tr>
<tr>
<td></td>
<td>2014/03/02 14:18:15...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>

24.3 Class F90Scanner

object

SCons.Scanner.Base

SCons.Scanner.Current

SCons.Scanner.Classic

\texttt{SCons.Scanner.Fortran.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

```python
__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:

```python
s = Scanner(my_scanner_function)
s = Scanner(function = my_scanner_function)
```
Class F90Scanner Module SCons.Scanner.Fortran

```
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__(), __cmp__(), __hash__(), __str__(), add_scanner(), add_skey(),
get_skeys(), path(), recurse_nodes(), 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 2014/03/02 14:18:15 garyo'</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</td>
</tr>
<tr>
<td></td>
<td>2014/03/02 14:18:15 g...</td>
</tr>
<tr>
<td>TexGraphics</td>
<td>Value: ['.eps', '.ps']</td>
</tr>
<tr>
<td>LatexGraphics</td>
<td>Value: ['.pdf', '.png', '.jpg', '.gif', '.tif']</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>

26.3 Class FindENVPathDirs

A class to bind a specific *PATH variable name to a function that will return all of the
**path directories.**

### 26.3.1 Methods

```python
__init__(self, variable)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides:
object.__init__ extit(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 "includegraphics" 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)
sort_key(self, include)

find_include(self, include, source_dir, path)

canonical_text(self, text)

Standardize an input TeX-file contents.

Currently:

- removes comments, unwrapping comment-wrapped lines.

scan(self, node)

scan_recurse(self, node, path=())

do a recursive scan of the top level target file. This lets us search for included files based on the directory of the main file just as latex does

Inherited from SCons.Scanner.Base (Section 20.5)

__call__(), __cmp__(), __hash__(), __str__(), add_scanner(), add_skey(), get_skeys(), path(), recurse_nodes(), 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>Inherited from object</td>
<td></td>
</tr>
<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>Value: [&quot;INDEXSTYLE&quot;, &quot;BIBINPUTS&quot;, &quot;TEXINPUTS&quot;, &quot;BSTINPUTS&quot;]</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

\[\text{scan}(\text{node}, \text{env}, \text{libpath}=())\]

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</td>
</tr>
<tr>
<td></td>
<td>2014/03/02 14:18:15 ga...</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**

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>RCScan()</td>
<td>Return a prototype Scanner instance for scanning RC source files</td>
</tr>
</tbody>
</table>

### 28.2 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/RC.py 2014/03/02 14:18:15 garyo'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>
29  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.

29.1  Modules

- **Interactive**: SCons interactive mode
  
  *(Section 30, p. 258)*

- **Main**: SCons.Script
  
  *(Section 31, p. 261)*

- **SConscript**: SCons.Script.SConscript
  
  *(Section 32, p. 275)*

29.2  Functions

<table>
<thead>
<tr>
<th>HelpFunction(text)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Variables(files=[], args={})</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Options(files=[], args={})</th>
</tr>
</thead>
</table>

29.3  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/Script/__init__.py</code></td>
</tr>
<tr>
<td></td>
<td>2014/03/02 14:18:15...</td>
</tr>
<tr>
<td>start_time</td>
<td>Value: 1393788012.03</td>
</tr>
<tr>
<td>call_stack</td>
<td>Value: []</td>
</tr>
<tr>
<td>PathVariable</td>
<td>Value: SCons.Variables.PathVariable</td>
</tr>
<tr>
<td>PathOption</td>
<td>Value: SCons.Options.PathOption</td>
</tr>
<tr>
<td>Chmod</td>
<td>Value: SConsDefaults.Chmod</td>
</tr>
<tr>
<td>Copy</td>
<td>Value: SConsDefaults.Copy</td>
</tr>
<tr>
<td>Delete</td>
<td>Value: SConsDefaults.Delete</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mkdir</td>
<td>Value: SCons.Defaults.Mkdir</td>
</tr>
<tr>
<td>Move</td>
<td>Value: SCons.Defaults.Move</td>
</tr>
<tr>
<td>Touch</td>
<td>Value: SCons.Defaults.Touch</td>
</tr>
<tr>
<td>CScanner</td>
<td>Value: SCons.Defaults.CScan</td>
</tr>
<tr>
<td>DScanner</td>
<td>Value: SCons.Tool.DScanner</td>
</tr>
<tr>
<td>DirScanner</td>
<td>Value: SCons.Defaults.DirScanner</td>
</tr>
<tr>
<td>ProgramScanner</td>
<td>Value: SCons.Tool.ProgramScanner</td>
</tr>
<tr>
<td>SourceFileScanner</td>
<td>Value: SCons.Tool.SourceFileScanner</td>
</tr>
<tr>
<td>CScan</td>
<td>Value: SCons.Defaults.CScan</td>
</tr>
<tr>
<td>ARGUMENTS</td>
<td>Value: {}</td>
</tr>
<tr>
<td>ARGLIST</td>
<td>Value: []</td>
</tr>
<tr>
<td>BUILD_TARGETS</td>
<td>Value: []</td>
</tr>
<tr>
<td>COMMAND_LINE_TARGETS</td>
<td>Value: []</td>
</tr>
<tr>
<td>DEFAULT_TARGETS</td>
<td>Value: []</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: ['CFile', 'CXXFile', 'DVI', 'Jar', 'Java', 'JavaH', 'Libr...</td>
</tr>
<tr>
<td>SConscript</td>
<td>Value: _SConscript.DefaultEnvironmentCall('SConscript')</td>
</tr>
<tr>
<td>Command</td>
<td>Value: _SConscript.DefaultEnvironmentCall('Command', subst= 1)</td>
</tr>
<tr>
<td>AddPostAction</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>AddPreAction</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Alias</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>AlwaysBuild</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>BuildDir</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
</tbody>
</table>

continued on next page

265
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFile</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>CXXFile</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>CacheDir</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Clean</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>DVI</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Decider</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Default</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Depends</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Dir</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>EnsurePythonVersion</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>EnsureSConsVersion</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Entry</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Execute</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Exit</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
</tbody>
</table>

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>File</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>FindFile</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>FindInstalledFiles</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>FindSourceFiles</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>Flatten</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>GetBuildPath</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>GetLaunchDir</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>Glob</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>Help</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>Ignore</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>Import</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>Install</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
<tr>
<td>InstallAs</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...&gt;</code></td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jar</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Java</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>JavaH</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Library</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Literal</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Local</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>M4</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>MSVSPackage</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>NoCache</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>NoClean</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Object</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>PCH</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>PDF</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Package</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
</tbody>
</table>

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParseDepends</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>PostScript</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>Precious</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>Program</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>RES</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>RMIC</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>Repository</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>Requires</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>SConscriptChdir</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>SConsignFile</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>SharedLibrary</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>SharedObject</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>SideEffect</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>SourceCode</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceSignatures</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>Split</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>StaticLibrary</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>StaticObject</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>Tag</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>Tar</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>TargetSignatures</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>TypeLibrary</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>Value</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>VariantDir</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td>Zip</td>
<td>Value: <code>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</code></td>
</tr>
<tr>
<td><strong><strong>package</strong></strong></td>
<td>Value: <code>'SCons.Script'</code></td>
</tr>
</tbody>
</table>
29.4 Class TargetList

object
    _abcoll.Sized

object
    _abcoll.Iterable

object
    _abcoll.Container
        _abcoll.Sequence
            _abcoll.MutableSequence
                UserList.UserList
                    SCons.Script.TargetList

29.4.1 Methods

_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__(), __str__()
## 29.4.2 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><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>

## 29.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from <code>UserList.UserList</code></td>
<td></td>
</tr>
<tr>
<td><code>__abstractmethods__</code>, <code>__hash__</code></td>
<td></td>
</tr>
</tbody>
</table>
30  Module SCons.Script.Interactive

SCons interactive mode

30.1  Functions

\[
\text{interact}(\text{fs, parser, options, targets, target\_top})
\]

30.2  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 2014/03/02 14:18...</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>

30.3  Class SConsInteractiveCmd

```
cmd.Cmd
```

```
SCons.Script.Interactive.SConsInteractiveCmd
```

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>build [TARGETS]</td>
<td>Build the specified TARGETS and their dependencies. 'b' is a synonym.</td>
</tr>
<tr>
<td>clean [TARGETS]</td>
<td>Clean (remove) the specified TARGETS and their dependencies. 'c' is a synonym.</td>
</tr>
<tr>
<td>exit</td>
<td>Exit SCons interactive mode.</td>
</tr>
<tr>
<td>help [COMMAND]</td>
<td>Prints help for the specified COMMAND. 'h' and '?' are synonyms.</td>
</tr>
<tr>
<td>shell [COMMANDLINE]</td>
<td>Execute COMMANDLINE in a subshell. 'sh' and '!' are synonyms.</td>
</tr>
<tr>
<td>version</td>
<td>Prints SCons version information.</td>
</tr>
</tbody>
</table>
30.3.1 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 (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 (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)
```

274
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(), complete getDefault(), completeNames(), emptyline(), get_names(), parseline(), postcmd(), postloop(), precmd(), preloop(), print_topics()

30.3.2 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>synonyms</td>
<td>Value: {'b': 'build', 'c': 'clean',</td>
</tr>
<tr>
<td></td>
<td>'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
31 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.

31.1 Functions

fetch_win32_parallel_msg()

revert_io()

Progress(*args, **kw)

GetBuildFailures()

python_version_string()

python_version_unsupported(version=sys.version_info(major=2, minor=7, micro=3, releaselevel=...))

python_version_deprecated(version=sys.version_info(major=2, minor=7, micro=3, releaselevel=...))

AddOption(*args, **kw)

GetOption(name)

SetOption(name, value)
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()

31.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsupported_python_version</td>
<td>Value: (2, 3, 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 2014/03/02 14:18:15 garyo'</td>
</tr>
<tr>
<td>display</td>
<td>Value: DisplayEngine()</td>
</tr>
<tr>
<td>progress_display</td>
<td>Value: SCons.Util.DisplayEngine()</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(0x09D4948C)</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>
<tr>
<td>OptionsParser</td>
<td>Value: FakeOptionParser()</td>
</tr>
</tbody>
</table>
### 31.3 Class `SConsPrintHelpException`

```
object               
exceptions.BaseException       
                      
exceptions.Exception       
                      
SCons.Script.Main.SConsPrintHelpException
```

#### 31.3.1 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__()`

#### 31.3.2 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>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>count_stats</code></td>
<td><code>Value: CountStats()</code></td>
</tr>
<tr>
<td><code>memory_stats</code></td>
<td><code>Value: MemStats()</code></td>
</tr>
<tr>
<td><code>__package__</code></td>
<td><code>Value: 'SCons.Script'</code></td>
</tr>
</tbody>
</table>
31.4 Class Progressor

object → SCons.Script.Main.Progressor

31.4.1 Methods

___init___(self, obj, interval=1, file=None, overwrite=False)

ox.__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

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

31.4.2 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>

31.4.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prev</td>
<td>Value: ' '</td>
</tr>
</tbody>
</table>
### Class BuildTask

Object diagram:

- `object` 
  - `SCons.Taskmaster.Task` 
  - `SCons.Taskmaster.OutOfDateTask` 
  - `SCons.Script.Main.BuildTask`

An SCons build task.

#### 31.5.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>display(self, message)</code></td>
<td>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 actually 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: <code>SCons.Taskmaster.Task.display</code> (inherited documentation)</td>
</tr>
<tr>
<td><code>prepare(self)</code></td>
<td>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: <code>SCons.Taskmaster.Task.prepare</code> (inherited documentation)</td>
</tr>
<tr>
<td><code>needs_execute(self)</code></td>
<td>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: <code>SCons.Taskmaster.Task.needs_execute</code></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Value: 0</td>
</tr>
<tr>
<td>target_string</td>
<td>Value: <code>$TARGET</code></td>
</tr>
</tbody>
</table>
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(), 
executed_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__()

31.5.2 Properties

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

31.5.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>progress</td>
<td>Value: Null(0x09D4948C)</td>
</tr>
</tbody>
</table>

31.6 Class CleanTask

object

SCons.Taskmaster.Task

SCons.Taskmaster.AlwaysTask

SCons.Script.Main.CleanTask

An SCons clean task.
31.6.1 Methods

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fs_delete</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 prepare(), executed() or failed().</td>
</tr>
<tr>
<td><code>show</code></td>
<td>Called to execute the task.</td>
</tr>
<tr>
<td><code>remove</code></td>
<td>Called to execute the task.</td>
</tr>
<tr>
<td><code>execute</code></td>
<td>Called to execute the task.</td>
</tr>
<tr>
<td><code>executed</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.</td>
</tr>
<tr>
<td><code>make_ready</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 “scons -c” option.</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__()`

31.6.2 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><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

31.7 Class QuestionTask

object

SCons.Taskmaster.Task

SCons.Taskmaster.AlwaysTask

SCons.Script.Main.QuestionTask

An SCons task for the -q (question) option.
31.7.1 Methods

**prepare**(*self*)

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to un-link underlying files and make all necessary directories before the Action is actually called to build the targets. Overrides: SCons.Taskmaster.Task.prepare

**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

**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 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__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), 285
31.7.2 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>

31.8 Class TreePrinter

```
object  └── SCons.Script.Main.TreePrinter
```

31.8.1 Methods

```
__init__(self, derived=False, prune=False, status=False)
```

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

```
ger_all_children(self, node)
```

```
ger_derived_children(self, node)
```

```
display(self, t)
```

**Inherited from object**

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

31.8.2 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>

286
31.9 Class 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.

31.9.1 Methods

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

Inherited from object

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

31.9.2 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>

31.9.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>values</td>
<td>Value: FakeOptionValues()</td>
</tr>
</tbody>
</table>

31.10 Class Stats

31.10.1 Methods

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

```python
enable(self, outfp)
```

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

**Inherited from object**

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

31.10.2 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>

31.11 Class CountStats

```
object
```

```
SCons.Script.Main.Stats
```

```
SCons.Script.Main.CountStats
```

31.11.1 Methods

```python
do__append(self, label)
```

```python
do__print(self)
```

**Inherited from SCons.Script.Main.Stats(Section 31.10)**

```
__init__(), do_nothing(), enable()
```

**Inherited from object**
31.11.2 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>

31.12 Class MemStats

```
object
```

```
SCons.Script.Main.Stats
```

```
SCons.Script.Main.MemStats
```

31.12.1 Methods

```
do__append__(self, label)
```

```
do__print__(self)
```

*Inherited from SCons.Script.Main.Stats (Section 31.10)*

```
__init__(), do_nothing(), enable()
```

*Inherited from object*

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

31.12.2 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>

289
32 Module SCons.Script.SConscript

SCons.Script.SConscript
This module defines the Python API provided to SConscript and SConstruct files.

32.1 Functions

get_calling_namespaces()

Return the locals and globals for the function that called into this module in the current call stack.

compute_exports(exports)

Compute a dictionary of exports given one of the parameters to the Export() function or the exports argument to SConscript().

Return(*vars, **kw)

SConscript_exception(file=sys.stderr)

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.

annotate(node)

Annotate a node with the stack frame describing the SConscript file and line number that created it.

Configure(*args, **kw)

get_DefaultEnvironmentProxy()
BuildDefaultGlobals()

Create a dictionary containing all the default globals for SConstruct and SConscript files.

### 32.2 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 2014/03/02 14:18:...</td>
</tr>
<tr>
<td>launch_dir</td>
<td>Value: '/home/garyo/src/scons-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>

### 32.3 Class SConscriptReturn

```
object
    exceptions.BaseException
        exceptions.Exception
            SCons.Script.SConscript'.SConscriptReturn
```

#### 32.3.1 Methods

**Inherited from exceptions.Exception**

- __init__(), __new__()  

**Inherited from exceptions.BaseException**

- __delattr__(), __getattr__(), __getitem__(), __getstate__(), __repr__(), __reduce__(), __reduce_ex__(), __setstate__(), __str__(), __unicode__(), __convert__()
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>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

32.4 Class Frame

object —

SCons.Script.SConscript’.Frame

A frame on the SConstruct/SConscript call stack

32.4.1 Methods

__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__()

32.4.2 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>
32.5 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.

32.5.1 Methods

- **Configure** *(self, *args, **kw)*

- **Default** *(self, *targets)*

- **EnsureSConsVersion** *(self, major, minor, revision=0)*
  Exit abnormally if the SCons version is not late enough.

- **EnsurePythonVersion** *(self, major, minor)*
  Exit abnormally if the Python version is not late enough.

- **Exit** *(self, value=0)*

- **Export** *(self, *vars, **kw)*

- **GetLaunchDir** *(self)*

- **GetOption** *(self, name)*
Class SConsEnvironment

Module SCons.Script.SConscript'

**Help**(*self, text*)

**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(), 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_src_sig_type(), get_tgt_sig_type(), scanner_map_delete()

*Inherited from SCons.Environment.SubstitutionEnvironment (Section 8.6)*

AddMethod(), MergeFlags(), Override(), ParseFlags(), RemoveMethod(), __cmp__(), __contains__(), __delitem__(), __getitem__(), __setitem__(), 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__()

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>Inherited from SCons.Environment.Base (Section 8.9)</td>
<td></td>
</tr>
<tr>
<td>memoizer_counters</td>
<td></td>
</tr>
<tr>
<td>Inherited from SCons.Environment.SubstitutionEnvironment (Section 8.6)</td>
<td></td>
</tr>
<tr>
<td><strong>metaclass</strong></td>
<td></td>
</tr>
</tbody>
</table>

32.6 Class DefaultEnvironmentCall

object

\[ SCons.Script.SConscript'.DefaultEnvironmentCall \]

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).

32.6.1 Methods

```python
___init___(self, method_name, subst=0)
```

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

object.__init__ extit(inherited documentation)

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

Inherited from object

```python
__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>
33 Module SCons.Sig

Place-holder for the old SCons.Sig module hierarchy

This is no longer used, but code out there (such as the NSIS module on the SCons wiki) may try to import SCons.Sig. If so, we generate a warning that points them to the line that caused the import, and don’t die.

If someone actually tried to use the sub-modules or functions within the package (for example, SCons.Sig.MD5.signature()), then they’ll still get an AttributeError, but at least they’ll know where to start looking.

33.1 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Sig.py 2014/03/02 14:18:15 garyo'</td>
</tr>
<tr>
<td><strong>doc</strong></td>
<td>Value: &quot;&quot;&quot;Place-holder for the old SCons.Sig module hierarchy...</td>
</tr>
<tr>
<td>msg</td>
<td>Value: 'The SCons.Sig module no longer exists.\nRemove the f...</td>
</tr>
<tr>
<td>default_calc</td>
<td>Value: None</td>
</tr>
<tr>
<td>default_module</td>
<td>Value: None</td>
</tr>
<tr>
<td>MD5</td>
<td>Value: MD5Null()</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>Value: TimeStampNull()</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

33.2 Class MD5Null

![Class diagram](image)

SCons.Util.Null  
  SCons.Sig.MD5Null

33.2.1 Methods

```python
__repr__(self)
```

repr(x) Overrides: object.__repr__ exitn(inherited documentation)

*Inherited from SCons.Util.Null (Section 36.15)*
Class TimeStampNull

Inherited from object

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

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

33.2.2 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>

33.3 Class TimeStampNull

object
def SCons.Util.Null
    SCons.Sig.TimeStampNull

33.3.1 Methods

___repr__(self)

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

Inherited from SCons.Util.Null(Section 36.15)

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

Inherited from object

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

33.3.2 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
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>
34 Module SCons.Subst

SCons.Subst
SCons string substitution.

34.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetAllowableExceptions(*excepts)</td>
</tr>
<tr>
<td>raise_exception(exception, target, s)</td>
</tr>
<tr>
<td>quote_spaces(arg)</td>
</tr>
</tbody>
</table>

Generic function for putting double quotes around any string that has white space in it.

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>escape_list(mylist, escape_func)</td>
</tr>
</tbody>
</table>

Escape a list of arguments by running the specified escape_func on every object in the list that has an escape() method.

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>subst_dict(target, source)</td>
</tr>
</tbody>
</table>

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\( (strSubst, env, mode=1, target=None, source=None, gvars={}, lvars={}, conv=None) \)

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\( (strSubst, env, mode=1, target=None, source=None, gvars={}, lvars={}, conv=None) \)

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\( (strSubst, env, key) \)

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:

\[
\text{env2} = \text{env.Clone(CCFLAGS = 'CCFLAGS -g')}
\]

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 2014/03/02 14:18:15 garyo'</td>
</tr>
<tr>
<td>AllowableExceptions</td>
<td>Value: (&lt;type 'exceptions.IndexError'&gt;, &lt;type 'exceptions.NameError'&gt;..</td>
</tr>
<tr>
<td>NullNodesList</td>
<td>Value: Null(0x095FA48C)</td>
</tr>
</tbody>
</table>

continued on next page
### 34.3 Class 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>init</strong>(self, lstr)</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><strong>str</strong>(self)</td>
<td>str(x). Overrides: object.<strong>str</strong> (inherited documentation)</td>
</tr>
<tr>
<td>escape(self, escape_func)</td>
<td></td>
</tr>
<tr>
<td>for_signature(self)</td>
<td></td>
</tr>
<tr>
<td>is_literal(self)</td>
<td></td>
</tr>
<tr>
<td><strong>eq</strong>(self, other)</td>
<td></td>
</tr>
<tr>
<td><strong>neq</strong>(self, other)</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object:

- __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>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

34.4 Class SpecialAttrWrapper

object  
SCons.Subst.SpecialAttrWrapper

This is a wrapper for what we call '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__(), __getattribute__(), __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

```plaintext
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:

```python
object.__init__ extit(inherited documentation)
```

```python
is_literal(self)
```
escape(self, escape_func, quote_func=<function quote_spaces at 0x95f917c>)

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__(), __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(),
isspace(), isupper(), join(), ljust(), lower(), lstrip(), partition(), replace(), rfind(),
index(), 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></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></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></td>
</tr>
<tr>
<td><strong>abstractmethods</strong></td>
<td></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*

```plaintext
__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__ extit(inherited documentation)

getattr(self, attr)

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__(),  __del__(),  __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>
</table>
| Inherited from object
| __class__          |             |

34.7.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Inherited from UserList.UserList
| __abstractmethods__,  __hash__ |             |
34.8 Class Target_or_Source

object

SCons.Subst.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

___init__(self, nl)

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

___getattr__(self, attr)

___str__(self)

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

___repr__(self)

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

Inherited from object

___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><em><strong>class</strong></em></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)
str(x) Overrides: object.__str__ extit(inherited documentation)
```

*Inherited from* SCons.Util_NullSeq(*Section 36.16*)

```python
__delitem__(), __getitem__(), __iter__(), __len__(), __setitem__()
```

*Inherited from* SCons.Util_Null(*Section 36.15*)

```python
__delattr__(), __getattr__(), __init__(), __new__(), __nonzero__(), __repr__(), __setattr__()
```

*Inherited from* object

```python
__format__(), __getattr__(), __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>
</tbody>
</table>
## 35 Module SCons.Taskmaster

Generic Taskmaster module for the SCons build engine.

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

```python
dump_stats()

find_cycle(stack, visited)
```

### 35.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>doc</strong></td>
<td>Value: ...</td>
<td></td>
</tr>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Taskmaster.py 2014/03/02 14:18:15 garyo'</td>
<td></td>
</tr>
<tr>
<td>StateString</td>
<td>Value: {0: 'no_state', 1: 'pending', 2: 'executing', 3: 'up_to_d...</td>
<td></td>
</tr>
<tr>
<td>NODE_NO_STATE</td>
<td>Value: 0</td>
<td></td>
</tr>
<tr>
<td>NODE_PENDING</td>
<td>Value: 1</td>
<td></td>
</tr>
<tr>
<td>NODE_EXECUTING</td>
<td>Value: 2</td>
<td></td>
</tr>
<tr>
<td>NODE_UP_TO_DATE</td>
<td>Value: 3</td>
<td></td>
</tr>
<tr>
<td>NODE_EXECUTED</td>
<td>Value: 4</td>
<td></td>
</tr>
<tr>
<td>NODE FAILED</td>
<td>Value: 5</td>
<td></td>
</tr>
</tbody>
</table>

*continued on next page*
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_

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

35.3.2 Properties

```python
<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>
```
35.4 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 subclassing 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

```python
__init__(self, tm, targets, top, node)
```

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

```python
trace_message(self, method, node, description='node')
```

```python
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 actually 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**

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**

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**

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**

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**

Returns info about a recorded exception.
Class AlwaysTask

Module SCons.Taskmaster

```python
exc_clear(self)
```

Clears any recorded exception.
This also changes the “exception_raise” attribute to point to the appropriate
do-nothing method.

```python
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**

```python
__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><code>Inherited from object</code></td>
<td></td>
</tr>
<tr>
<td><code>__class__</code></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)*

__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*

__delattr__(), __format__(), __getattribute__(), __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

object

SCons.Taskmaster.Task

SCons.Taskmaster.OutOfDateTask

Known Subclasses: SCons.Script.Main.BuildTask
35.6.1 Methods

```
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

Inherited from SCons.Taskmaster.Task(Section 35.4)

```
__init__(self, targets=[], tasker=None, order=None, trace=None)
```

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

35.6.2 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>

35.7 Class Taskmaster

```
object
```

SCons.Taskmaster.Taskmaster

The Taskmaster for walking the dependency DAG.

35.7.1 Methods

```
__init__(self, targets=[], tasker=None, order=None, trace=None)
```

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

**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.

**trace_message(self, message)**

**trace_node(self, node)**

**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.
Class Taskmaster

Module SCons.Taskmaster

```python
will_not_build(self, nodes, node_func=<function <lambda> at 0x98b9e9c>)
```

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).

```python
stop(self)
```

Stops the current build completely.

```python
cleanup(self)
```

Check for dependency cycles.

**Inherited from object**

```python
__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>
36  Module SCons.Util

SCons.Util
Various utility functions go here.

36.1  Functions

\begin{enumerate}
\item \texttt{dictify(keys, values, result=\{}\})
\item \texttt{rightmost\_separator(path, sep)}
\item \texttt{containsAny(str, set)}
\begin{itemize}
  \item Check whether sequence str contains ANY of the items in set.
\end{itemize}
\item \texttt{containsAll(str, set)}
\begin{itemize}
  \item Check whether sequence str contains ALL of the items in set.
\end{itemize}
\item \texttt{containsOnly(str, set)}
\begin{itemize}
  \item Check whether sequence str contains ONLY items in set.
\end{itemize}
\item \texttt{splitext(path)}
\begin{itemize}
  \item Same as os.path.splitext() but faster.
\end{itemize}
\item \texttt{updrive(path)}
\begin{itemize}
  \item Make the drive letter (if any) upper case. This is useful because Windows is inconsistent on the case of the drive letter, which can cause inconsistencies when calculating command signatures.
\end{itemize}
\end{enumerate}
Functions Module SCons.Util

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={})

Render a tree of nodes into an ASCII tree view.
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.

IDX(N)

print_tree(root, child_func, prune=0, showtags=0, margin=[0], visited={})

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.
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 isinstance>, 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'>)

is_Scalar(obj, isinstance=<built-in function isinstance>, StringType=(<type 'str'>, <type 'unicode'>, <class 'UserString.UserString'>, SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>))

do_flatten(sequence, result, isinstance=<built-in function isinstance>, StringType=(<type 'str'>, <type 'unicode'>, <class 'UserString.UserString'>, SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>))

flatten(obj, isinstance=<built-in function isinstance>, StringType=(<type 'str'>, <type 'unicode'>, <class 'UserString.UserString'>, SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>)), do_flatten=<function do_flatten at 0x95d9bc4>)

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(sequence, isinstance=<built-in function isinstance>, StringType=(<type 'str'>, <type 'unicode'>), <class 'UserString.UserString'>, SequenceTypes=([<type 'list'>], <type 'tuple'>), <class 'UserList.UserList'>), do_flatten=<function do_flatten at 0x95d9bc4>)

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 cannot be a scalar.

to_String(s, isinstance=<built-in function isinstance>, str=<type 'str'>, UserString=<class 'UserString.UserString'>, BaseStringTypes=(<type 'str'>, <type 'unicode'>))

to_String_for_subst(s, isinstance=<built-in function isinstance>, str=<type 'str'>, to_String=<function to_String at 0x95d9c6c>, BaseStringTypes=(<type 'str'>, <type 'unicode'>), SequenceTypes=([<type 'list'>], <type 'tuple'>), <class 'UserList.UserList'>), UserString=<class 'UserString.UserString'>)

to_String_for_signature(obj, to_String_for_subst=<function to_String_for_subst at 0x95d9ca4>, AttributeError=<type 'exceptions.AttributeError'>)

semi_deepcopy_dict(x, exclude=[])
**PrependPath***(oldpath, newpath, 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***(oldpath, newpath, 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.

**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*)

**uniquer_hashables**(*seq*)

**make_path_relative**(*path*)

makes an absolute path name to a relative pathname.
AddMethod(obj, function, name=None)

Adds either a bound method to an instance or an unbound method 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
AddMethod(lambda self, i: self.l[i], a, "listIndex")
print a.listIndex(5)
```

RenameFunction(function, name)

Returns a function identical to the specified function, but with the specified name.

MD5signature(s)

MD5filesignature(fname, chunksize=65536)

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.

36.2 Variables
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>BaseTypeTypes</td>
<td>Value: (&lt;type 'str'&gt;, &lt;type 'unicode'&gt;)</td>
</tr>
<tr>
<td>d</td>
<td>Value: {}</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>
<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: DisplayEngine()</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

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__nonzero__</code></td>
<td>(self)</td>
</tr>
<tr>
<td><code>__str__</code></td>
<td>(self)</td>
</tr>
<tr>
<td></td>
<td>Overrides: object.<strong>str</strong> extension (inherited documentation)</td>
</tr>
<tr>
<td><code>__iter__</code></td>
<td>(self)</td>
</tr>
<tr>
<td></td>
<td>Overrides: _abcoll.Iterable.<strong>iter</strong></td>
</tr>
<tr>
<td><code>__call__</code></td>
<td>(self, *args, **kwargs)</td>
</tr>
</tbody>
</table>
Inherited from `UserList.UserList`

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

Inherited from `UserList.UserList`

```python
__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 `collections.Sequence`

```python
__reversed__()
```

Inherited from `collections.Sized`

```python
__subclasshook__()
```

Inherited from `object`

```python
__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><em>Inherited from object</em></td>
<td><strong><strong>class</strong></strong></td>
</tr>
</tbody>
</table>

36.3.3 Class Variables

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

36.4 Class DisplayEngine

```
object SCons.Util.DisplayEngine
```
### 36.4.1 Methods

```python
__call__(self, text, append_newline=1)
```

```python
set_mode(self, mode)
```

**Inherited from object**

```python
__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>Inherited from object</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

```
object
```

```
SCons.Util.Proxy
```

**Known Subclasses:** SCons.Builder.CompositeBuilder, SCons.Node.FS.EntryProxy

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
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><em><strong>init</strong></em>(self, subject)</td>
<td>Wrap an object as a Proxy object. Overrides: object. <em><strong>init</strong></em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em><strong>getattr</strong></em>(self, name)</td>
<td>Retrieve an attribute from the wrapped object. If the named attribute doesn’t exist, AttributeError is raised</td>
</tr>
</tbody>
</table>

get(self)

Retrieve the entire wrapped object

___cmp___(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>
</table>

_Inherited from object_

<table>
<thead>
<tr>
<th><strong>class</strong></th>
<th>Description</th>
</tr>
</thead>
</table>
36.6 Class Delegate

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__')
```

36.6.1 Methods

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

def __get__(self, obj, cls):
Inherited from object
```

36.6.2 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>

36.7 Class _NoError

```
object

exceptions.BaseException

exceptions.Exception

SCons.Util._NoError
```
36.7.1 Methods

*Inherited from exceptions.Exception*

\[\texttt{\_\_init\_\_}(), \texttt{\_\_new\_\_}()\]

*Inherited from exceptions.BaseException*

\[\texttt{\_\_delattr\_\_}(), \texttt{\_\_getattribute\_\_}(), \texttt{\_\_getitem\_\_}(), \texttt{\_\_getslice\_\_}(), \texttt{\_\_reduce\_\_}(), \texttt{\_\_repr\_\_}(), \texttt{\_\_setattr\_\_}(), \texttt{\_\_setstate\_\_}(), \texttt{\_\_str\_\_}(), \texttt{\_\_unicode\_\_}()\]

*Inherited from object*

\[\texttt{\_\_format\_\_}(), \texttt{\_\_hash\_\_}(), \texttt{\_\_reduce\_\_ex\_\_}(), \texttt{\_\_sizeof\_\_}(), \texttt{\_\_subclasshook\_\_}()\]

36.7.2 Properties

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

36.8 Class WindowsError

object

exceptions.BaseException

\[\texttt{exceptions.Exception} \quad \texttt{exceptions.StandardError} \quad \texttt{exceptions.EnvironmentError} \quad \texttt{exceptions.OSError} \quad \texttt{SCons.Util.WindowsError}\]

36.8.1 Methods

*Inherited from exceptions.OSError*
Class WindowsError

Module SCons.Util

___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><em><strong>class</strong></em></td>
</tr>
</tbody>
</table>
36.9 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.

36.9.1 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)
Overrrides: UserList.UserList.__add__
```

```python
__radd__(self, other)
Overrrides: UserList.UserList.__radd__
```

```python
__coerce__(self, other)
```
```python
__str__(self)
str(x)  Overrides: object.__str__  Inherited documentation
```

**Inherited from `UserList.UserList`**

- `__cmp__()`, `__contains__()`, `__delitem__()`, `__delslice__()`, `__eq__()`, `__ge__()`,
- `__getitem__()`, `__getslice__()`, `__gt__()`, `__iadd__()`, `__imul__()`, `__le__()`,
- `__len__()`, `__lt__()`, `__mul__()`, `__ne__()`, `__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__()`

### 36.9.2 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><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>

### 36.9.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from <code>UserList.UserList</code></td>
<td></td>
</tr>
<tr>
<td><code>__abstractmethods__</code>, <code>__hash__</code></td>
<td></td>
</tr>
</tbody>
</table>

### 36.10 Class OrderedDict

```
```

**Known Subclasses:** `SCons.Util.Selector`

337
36.10.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__init__</code></td>
<td>(self, dict=None)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.<strong>init</strong></td>
</tr>
<tr>
<td><code>__delitem__</code></td>
<td>(self, key)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.<strong>delitem</strong></td>
</tr>
<tr>
<td><code>__setitem__</code></td>
<td>(self, key, item)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.<strong>setitem</strong></td>
</tr>
<tr>
<td><code>clear</code></td>
<td>(self)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.clear</td>
</tr>
<tr>
<td><code>copy</code></td>
<td>(self)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.copy</td>
</tr>
<tr>
<td><code>items</code></td>
<td>(self)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.items</td>
</tr>
<tr>
<td><code>keys</code></td>
<td>(self)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.keys</td>
</tr>
<tr>
<td><code>popitem</code></td>
<td>(self)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.popitem</td>
</tr>
<tr>
<td><code>setdefault</code></td>
<td>(self, key, failobj=None)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.setdefault</td>
</tr>
<tr>
<td><code>update</code></td>
<td>(self, dict)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.update</td>
</tr>
<tr>
<td><code>values</code></td>
<td>(self)</td>
</tr>
<tr>
<td></td>
<td>Overrides: UserDict.UserDict.values</td>
</tr>
</tbody>
</table>

*Inherited from UserDict.UserDict*

- `__cmp__()`, `__contains__()`, `__getitem__()`, `__len__()`, `__repr__()`, `fromkeys()`, `get()`, `has_key()`, `iteritems()`, `iterkeys()`, `itervalues()`, `pop()`
36.10.2 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>

36.11 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.

36.11.1 Methods

```python
__call__ (self, env, source, ext=None)
```

*Inherited from SCons.Util.OrderedDict*(Section 36.10)*

<table>
<thead>
<tr>
<th>delitem()</th>
<th>init()</th>
<th>setitem()</th>
<th>clear()</th>
<th>copy()</th>
<th>items()</th>
<th>keys()</th>
<th>popitem()</th>
<th>setdefault()</th>
<th>update()</th>
<th>values()</th>
</tr>
</thead>
</table>

*Inherited from UserDict.UserDict*

<table>
<thead>
<tr>
<th>cmp()</th>
<th>contains()</th>
<th>getitem()</th>
<th>len()</th>
<th>repr()</th>
<th>fromkeys()</th>
<th>get()</th>
<th>has_key()</th>
<th>iteritems()</th>
<th>iterkeys()</th>
<th>itervalues()</th>
<th>pop()</th>
</tr>
</thead>
</table>

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 LogicalLines

    object  \hline
              SCons.Util.LogicalLines

36.12.1  Methods

\begin{verbatim}
__init__(self, fileobj)
\end{verbatim}

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

\begin{verbatim}
readline(self)
\end{verbatim}

\begin{verbatim}
readlines(self)
\end{verbatim}

\textit{Inherited from object}

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

36.12.2  Properties

\begin{table}[h]
\begin{tabular}{|c|c|}
\hline
Name & Description \\
\hline
\textit{Inherited from object} & \\
__class__ & \\
\hline
\end{tabular}
\end{table}
36.13 Class UniqueList

object
    _abcoll.Sized
object
    _abcoll.Iterable
object
    _abcoll.Container
        _abcoll.Sequence
            _abcoll.MutableSequence
                UserList.UserList
                    SCons.Util.UniqueList

36.13.1 Methods

___init___(self, seq=[])

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

___lt___(self, other)
Overrides: UserList.UserList.__lt__

___le___(self, other)
Overrides: UserList.UserList.__le__

___eq___(self, other)
Overrides: UserList.UserList.__eq__

___ne___(self, other)
Overrides: UserList.UserList.__ne__
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Overrides</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__gt__</code></td>
<td>(self, other)</td>
<td>UserList.UserList.__gt__</td>
</tr>
<tr>
<td><code>__ge__</code></td>
<td>(self, other)</td>
<td>UserList.UserList.__ge__</td>
</tr>
<tr>
<td><code>__cmp__</code></td>
<td>(self, other)</td>
<td>UserList.UserList.__cmp__</td>
</tr>
<tr>
<td><code>\_\_len\_\_</code></td>
<td>(self)</td>
<td>__abcoll.Sized.__len__</td>
</tr>
<tr>
<td><code>__getitem__</code></td>
<td>(self, i)</td>
<td>__abcoll.Sequence.__getitem__</td>
</tr>
<tr>
<td><code>__setitem__</code></td>
<td>(self, i, item)</td>
<td>__abcoll.MutatableSequence.__setitem__</td>
</tr>
<tr>
<td><code>__getslice__</code></td>
<td>(self, i, j)</td>
<td>UserList.UserList.__getslice__</td>
</tr>
<tr>
<td><code>__setslice__</code></td>
<td>(self, i, j, other)</td>
<td>UserList.UserList.__setslice__</td>
</tr>
<tr>
<td><code>\_\_add\_\_</code></td>
<td>(self, other)</td>
<td>UserList.UserList.__add__</td>
</tr>
<tr>
<td><code>\_\_radd\_\_</code></td>
<td>(self, other)</td>
<td>UserList.UserList.__radd__</td>
</tr>
<tr>
<td><code>\_\_iadd\_\_</code></td>
<td>(self, other)</td>
<td>__abcoll.MutatableSequence.__iadd__</td>
</tr>
<tr>
<td><code>\_\_mul\_\_</code></td>
<td>(self, other)</td>
<td>UserList.UserList.__mul__</td>
</tr>
</tbody>
</table>
Class UniqueList

Module SCons.Util

```python
__rmul__(self, other)
Overrides: UserList.UserList.__rmul__

__imul__(self, other)
Overrides: UserList.UserList.__imul__

append(self, item)
Overrides: _abcoll.MutableSequence.append

insert(self, i)
Overrides: _abcoll.MutableSequence.insert

count(self, item)
Overrides: _abcoll.Sequence.count

index(self, item)
Overrides: _abcoll.Sequence.index

reverse(self)
Overrides: _abcoll.MutableSequence.reverse

sort(self, *args, **kwds)
Overrides: UserList.UserList.sort

extend(self, other)
Overrides: _abcoll.MutableSequence.extend
```

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__(),
Class Unbuffered Module SCons.Util

__reduce_ex__(), __setattr__(), __sizeof__(), __str__()

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>

36.13.3 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>

36.14 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.

36.14.1 Methods

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

36.14.2 Properties
### 36.15 Class Null

object  \[ \rightarrow \]  SCons.Util.Null

**Known Subclasses:** SCons.Sig.MD5Null, SCons.Sig.TimeStampNull, SCons.Util.NullSeq

Null objects always and reliably “do nothing.”

#### 36.15.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| \_
\_new\_\_ | \(\text{Return Value}\) a new object with type \(S\), a subtype of \(T\). Overrides: object.\_
\_new\_\_ (inherited documentation) |
| \_
\_init\_\_ | \(x.\_
\_init\_\_(...)\) initializes \(x\); see help(type(x)) for signature. Overrides: object.\_
\_init\_\_ (inherited documentation) |
| \_
\_call\_\_ | |
| \_
\_repr\_\_ | \(\text{repr(x)}\) Overrides: object.\_
\_repr\_\_ (inherited documentation) |
| \_
\_nonzero\_\_ | |
| \_
\_getattr\_\_ | |
| \_
\_setattr\_\_ | \(x.\_
\_setattr\_\_('name', value) \iff x.name = value\) Overrides: object.\_
\_setattr\_\_ (inherited documentation) |
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 NullSeq

object —
SCons.Util.Null —
SCons.Util.NullSeq

Known Subclasses: SCons.Subst.NullNodeList

36.16.1 Methods

__len__(self)

__iter__(self)

__getitem__(self, i)

__delitem__(self, i)

__setitem__(self, i, v)

Inherited from SCons.Util.Null (Section 36.15)

__call__(), __delattr__(), __getattr__(), __init__(), __new__(), __nonzero__(),
Class NullSeq

Module SCons.Util

`__repr__(), __setattr__()`

*Inherited from object*

`__format__(), __getattr__(), __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>
37 Package SCons.Variables

engine.SCons.Variables

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.Variables.BoolVariable *(Section 38, p. 337)*
- **EnumVariable** *(Section ??, p. ??)*
- **EnumVariable**: engine.SCons.Variables.EnumVariable *(Section 39, p. 338)*
- **ListVariable** *(Section ??, p. ??)*
- **ListVariable**: engine.SCons.Variables.ListVariable *(Section 40, p. 340)*
- **PackageVariable** *(Section ??, p. ??)*
- **PackageVariable**: engine.SCons.Variables.PackageVariable *(Section 41, p. 341)*
- **PathVariable** *(Section ??, p. ??)*
- **PathVariable**: SCons.Variables.PathVariable *(Section 42, p. 342)*

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/Variables/<strong>init</strong>.py</td>
</tr>
<tr>
<td></td>
<td>2014/03/02 14:18...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Variables'</td>
</tr>
</tbody>
</table>

37.3 Class Variables

```
object
   └─ SCons.Variables.Variables
```
37.3.1 Methods

```python
__init__(self, files=[], args={}, 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.

key - the name of the variable, or a list or tuple of arguments
help - optional help text for the options
default - optional default value
validator - optional function that is called to validate the option’s value
    Called with (key, value, environment)
converter - optional function that is called to convert the option’s value before putting it in the environment.
### AddVariables

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 an environment with the option variables.

**env** - the environment to update.

### UnknownVariables

Returns any options in the specified arguments lists that were not known, declared options in this object.

### Save

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__() , __getattr__(), __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>Inherited from object</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.</td>
</tr>
<tr>
<td>Value:</td>
<td>None</td>
</tr>
</tbody>
</table>
| format | Value: `\n%s: %s
default: %s
actual: %s`                                                                                     |
| format_ | Value: `\n%s: %s
default: %s
actual: %s aliases: ...`                                                                          |
Module SCons.Variables.BoolVariable

engine.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

**BoolVariable(key, help, default)**

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|no) to show the valid values. The result is usable for input to opts.Add().
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

\[
\text{EnumVariable}(\text{key, help, default, allowed\_values, map=\{\}, ignorecase=0})
\]

The input parameters describe a 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 (eg. 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).
Module 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))
```

```python
... 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 validater 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).
41 Module SCons.Variables.PackageVariable

engine.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 existance)

To replace autoconf's --with-xxx=yyy

opts = Variables()
opts.Add(PackageVariable('x11',
    'use X11 installed here (yes = search some places',
    'yes'))

... if env['x11'] == True:
    dir = ... search X11 in some standard places ...
    env['x11'] = dir
if env['x11']:
    ... build with x11 ...

41.1 Functions

PackageVariable(key, help, default, searchfunc=None)

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 (seperated 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',  

Variables

Module SCons.Variables.PathVariable

'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><strong>Value:</strong> SCons.Variables.PathVariable</td>
</tr>
</tbody>
</table>
43 Module SCons.Warnings

SCons.Warnings
This file implements the warnings framework for SCons.

43.1 Functions

```python
suppressWarningClass(clazz)
Suppresses all warnings that are of type clazz or derived from clazz.
```

```python
enableWarningClass(clazz)
Enables all warnings that are of type clazz or derived from clazz.
```

```python
warningAsException(flag=1)
Turn warnings into exceptions. Returns the old value of the flag.
```

```python
warn(clazz, *args)
```
process_warn_strings(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 <warning-class> or
no-<warning-class>. 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 <warning-class> 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><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Warnings.py 2014/03/02 14:18:15 garyo'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

### 43.3 Class Warning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

**Known Subclasses:** SCons.SConf.SConfWarning, SCons.Warnings.CacheWriteErrorWarning,
SCons.Warnings.WarningOnByDefault, SCons.Warnings.DependencyWarning, SCons.Warnings.Deprecated,
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\_\_()
43.4.1 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

\_\_init\_\_(), \_\_new\_\_()
Class CacheWriteErrorWarning

Inherited from exceptions.BaseException

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

Inherited from object

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

43.5.2 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>

43.6 Class CacheWriteErrorWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.CacheWriteErrorWarning

43.6.1 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__()

### 43.6.2 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><em><strong>class</strong></em></td>
</tr>
</tbody>
</table>

### 43.7 Class CorruptSConsignWarning

object  

exceptions.BaseException  

exceptions.Exception  

SCons.Errors.UserError  

SCons.Warnings.Warning  

SCons.Warnings.WarningOnByDefault  

SCons.Warnings.CorruptSConsignWarning

#### 43.7.1 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__()
43.7.2 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>

43.8 Class DependencyWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DependencyWarning

43.8.1 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__()  

43.8.2 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>

continued on next page
### 43.9 Class DuplicateEnvironmentWarning

#### object

- exceptions.BaseException
  - exceptions.Exception
    - SCons.Errors.UserError
    - SCons.Warnings.WarningOnByDefault
      - SCons.Warnings.DuplicateEnvironmentWarning

#### 43.9.1 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__()

#### 43.9.2 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>
</tbody>
</table>

continued on next page
### 43.10 Class FutureReservedVariableWarning

- `object`
  - `exceptions.BaseException`
    - `exceptions.Exception`
      - `SCons.Errors.UserError`
      - `SCons.Warnings.Warning`
      - `SCons.Warnings.WarningOnByDefault`
      - `SCons.Warnings.FutureReservedVariableWarning`

#### 43.10.1 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__()`

#### 43.10.2 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>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>
43.11 Class LinkWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.LinkWarning

Known Subclasses: SCons.Warnings.FortranCxxMixWarning

43.11.1 Methods

Inherited from exceptions.Exception

__init__(), __new__() 

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __reduce_ex__(), __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>

368
43.12 Class MisleadingKeywordsWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.MisleadingKeywordsWarning

43.12.1 Methods

_Inherited from exceptions.Exception_

__init__(), __new__()  

_Inherited from exceptions.BaseException_

__delattr__(), __getattr__(), __getitem__(), __getstate__(), __reduce__(), __reduce_ex__(), __repr__(), __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><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.13 Class MissingSConscriptWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.MissingSConscriptWarning

43.13.1 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__()`

43.13.2 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><code>args</code>, <code>message</code></td>
<td></td>
</tr>
<tr>
<td><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>
43.14 Class NoMD5ModuleWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.NoMD5ModuleWarning

43.14.1 Methods

*Inherited from exceptions.Exception*

__init__(), __new__()  

*Inherited from exceptions.BaseException*

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

*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>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
</tbody>
</table>

| Inherited from object | __class__ |

371
43.15  Class NoMetaclassSupportWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.NoMetaclassSupportWarning

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 NoObjectCountWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.NoObjectCountWarning

43.16.1 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__()

43.16.2 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><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.17   Class NoParallelSupportWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.NoParallelSupportWarning

43.17.1   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__()

43.17.2   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 ReservedVariableWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.ReservedVariableWarning

43.18.1  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__()

43.18.2  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><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.19 Class StackSizeWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.StackSizeWarning

43.19.1 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__()
```

43.19.2 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>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.20 Class VisualCMissingWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.VisualCMissingWarning

43.20.1 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__()

43.20.2 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>
43.21 Class VisualVersionMismatch

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.VisualVersionMismatch

43.21.1 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__()

43.21.2 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 VisualStudioMissingWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.VisualStudioMissingWarning

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>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.23 Class FortranCxxMixWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.LinkWarning

SCons.Warnings.FortranCxxMixWarning

43.23.1 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__(), __setstate__(), __str__(), __subclasshook__()

43.23.2 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>
</tbody>
</table>

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

```
object  
exceptions.BaseException  
    exceptions.Exception  
        SCons.Errors.UserError  
            SCons.Warnings.Warning  
                SCons.Warnings.FutureDeprecatedWarning
```

**Known Subclasses:** SCons.Warnings.DeprecatedSourceCodeWarning

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><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>
43.25 Class DeprecatedWarning

object  

exceptions.BaseException  

exceptions.Exception  

SCons.Errors.UserError  

SCons.Warnings.Warning  

SCons.Warnings.DeprecatedWarning


43.25.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getstate__(), __reduce__(), __reduce_ex__(), __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></td>
<td></td>
</tr>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.26 Class MandatoryDeprecatedWarning

```
object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning
```


43.26.1 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__()  
```

43.26.2 Properties

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.Exception</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.27  Class PythonVersionWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.PythonVersionWarning

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>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.28 Class DeprecatedSourceCodeWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.FutureDeprecatedWarning

SCons.Warnings.DeprecatedSourceCodeWarning

43.28.1 Methods

**Inherited from exceptions.Exception**

`__init__()`, `__new__()`

**Inherited from exceptions.BaseException**

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`, `__setattribute__()`, `__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><strong>Inherited from exceptions.BaseException</strong></td>
<td>args, message</td>
</tr>
<tr>
<td><strong>Inherited from object</strong></td>
<td><code>__class__</code></td>
</tr>
</tbody>
</table>
43.29  Class DeprecatedBuildDirWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors_UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.DeprecatedBuildDirWarning

43.29.1  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__()

43.29.2  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>
43.30 Class TaskmasterNeedsExecuteWarning

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 DeprecatedCopyWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedCopyWarning

43.31.1 Methods

*Inherited from exceptions.Exception*

__init__(), __new__()

*Inherited from exceptions.BaseException*

__delattr__(), __getattr__(), __getitem__(), __getset__(), __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>
</tbody>
</table>

| Inherited from object | __class__ |
43.32  Class DeprecatedOptionsWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedOptionsWarning

43.32.1  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__()

43.32.2  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>
43.33 Class DeprecatedSourceSignaturesWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedSourceSignaturesWarning

43.33.1 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__()

43.33.2 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>
</tbody>
</table>

<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>
43.34 Class DeprecatedTargetSignaturesWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedTargetSignaturesWarning

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></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 DeprecatedDebugOptionsWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedDebugOptionsWarning

43.35.1 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__(), __setattr__(), __sizeof__(), __subclasshook__()

43.35.2 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><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.36 Class DeprecatedSigModuleWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning

SCons.Warnings.DeprecatedSigModuleWarning

43.36.1 Methods

Inherited from exceptions.Exception

_init__(), __new__()  

Inherited from exceptions.BaseException

__delattr__(), __getattr__(), __getitem__(), __getstate__(), __reduce__(), __reduce_ex__(), __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><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
43.37 Class DeprecatedBuilderKeywordsWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons_WARNINGS.Warning

SCons_WARNINGS.DeprecatedWarning

SCons_WARNINGS.MandatoryDeprecatedWarning

SCons_WARNINGS.DeprecatedBuilderKeywordsWarning

43.37.1 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__()  

43.37.2 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>
### 44 Module SCons.cpp

SCons C Pre-Processor module

#### 44.1 Functions

**CPP_to_Python_Ops_Sub(m)**

**CPP_to_Python(s)**

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>Value: ...</td>
</tr>
</tbody>
</table>
| cpp_lines_dict              | Value: \{('define'): '\s+([_A-Za-z][_A-Za-z0-9]*)\((\[^\]*\))*\}...
| Table                       | Value: \{('define': re.compile(r'\s+([_A-Za-z][_A-Za-z0-9]*)\(([^\... |
| e                           | Value: '~=\s*\s*\s*(elif|undef|include_next|endif|else|include|if. |
| CPP_Expression              | Value: re.compile(r'(?m)\s*#\s*(elif|undef|include_next|endif|else|incl |
| CPP_to_Python_Ops_-_Dict    | Value: {"r": '', '': '', '!'': '', 'not': '', '!=': '', '=': '', '&&': ' and', ',': ''... |
| CPP_to_Python_Ops_E-_Expression | Value: re.compile(r'\s*\|\|\|\|!==?!|\r|:|\?') |
| CPP_to_Python_E-_List       | Value: [[re.compile(r'\s*defined\s+(\w+)'), "in __dict__", [...
| line_continuations          | Value: re.compile(r'\s*\r?\n') |
| function_name               | Value: re.compile(r'\s+\(([^\)]*)\)') |
| function_arg_separator      | Value: re.compile(r',\s*') |
| __package__                 | Value: 'SCons' |
| x                           | Value: 'if' |
44.3 Class FunctionEvaluator

object –> SCons.cpp.FunctionEvaluator

Handles delayed evaluation of a #define function call.

44.3.1 Methods

```python
__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__

```python
__call__(self, *values)
```

Evaluates the expansion of a #define macro function called with the specified values.

Inherited from object

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

44.3.2 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>

44.4 Class PreProcessor

object –> SCons.cpp.PreProcessor

The main workhorse class for handling C pre-processing.

44.4.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>call</strong>(self, file)</td>
<td>Pre-processes a file. This is the main public entry point.</td>
</tr>
<tr>
<td><strong>init</strong>(self, current='.', cpppath=(), dict={}, all=0)</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>all__include__(self, t)</td>
<td></td>
</tr>
<tr>
<td>do__define__(self, t)</td>
<td>Default handling of a #define line.</td>
</tr>
<tr>
<td>do__elif__(self, t)</td>
<td>Default handling of a #elif line.</td>
</tr>
<tr>
<td>do__else__(self, t)</td>
<td>Default handling of a #else line.</td>
</tr>
<tr>
<td>do__endif__(self, t)</td>
<td>Default handling of a #endif line.</td>
</tr>
</tbody>
</table>
**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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>eval_expression</strong></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 <code>#define</code> values.</td>
</tr>
<tr>
<td><strong>finalize_result</strong></td>
<td></td>
</tr>
<tr>
<td><strong>find_include_file</strong></td>
<td>Finds the <code>#include</code> file for a given preprocessor tuple.</td>
</tr>
<tr>
<td><strong>initialize_result</strong></td>
<td></td>
</tr>
<tr>
<td><strong>process_contents</strong></td>
<td>Pre-processes a file contents. This is the main internal entry point.</td>
</tr>
<tr>
<td><strong>read_file</strong></td>
<td></td>
</tr>
<tr>
<td><strong>resolve_include</strong></td>
<td>Resolve a tuple-ized <code>#include</code> line. This handles recursive expansion of values without &quot;&quot; or &lt;&gt; surrounding the name until an initial &quot; or &lt; is found, to handle <code>#include FILE</code> where FILE is a <code>#define</code> somewhere else.</td>
</tr>
<tr>
<td><strong>restore</strong></td>
<td>Pops the previous dispatch table off the stack and makes it the current one.</td>
</tr>
</tbody>
</table>
**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*)

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.

**start_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.

**stop_handling_includes**(*self, t=None*)

**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><strong>Inherited from object</strong></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__ extit(inherited documentation)

Inherited from SCons.cpp.PreProcessor (Section 44.4)

__call__(), all_include(), do_define(), do_elif(), do_else(), do endif(), do_if(),

do ifndef(), 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

__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

- corruption_warning(filename)
- is_string(s)
- unicode(s)
- 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

```
o    SCons.dblite.dblite
```

45.3.1 Methods

- __init__(self, file_base_name, flag, mode)
  x.__init__(...) initializes x; see help(type(x)) for signature  Overrides: object.__init__ extit(inherited documentation)
- close(self)
- ___del__(self)
sync(self)

__getitem__(self, key)

__setitem__(self, key, value)

keys(self)

has_key(self, key)

__contains__(self, key)

iterkeys(self)

__iter__(self)

__len__(self)

Inherited from object

__delattr__(), __format__(), __getattr__(), __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 2014/03/02 14:18:15 garyo’</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: ’SCons’</td>
</tr>
</tbody>
</table>
Index

SCons (package), 2–4
  SCons.Action (module), 5–17
  SCons.Builder (module), 18–29
    SCons.Builder.Builder (function), 19
    SCons.Builder.BuilderBase (class), 26–28
    SCons.Builder.CallableSelector (class), 21–22
    SCons.Builder.CompositeBuilder (class), 28–29
    SCons.Builder.DictCmdGenerator (class), 20–21
    SCons.Builder.DictEmitter (class), 22–23
    SCons.Builder.EmitterProxy (class), 25–26
    SCons.Builder.is_a_Builder (function), 19
    SCons.Builder.ListEmitter (class), 23–24
    SCons.Builder.match_splitext (function), 19
    SCons.Builder.Override Warner (class), 24–25
  SCons.CacheDir (module), 30–32
    SCons.CacheDir.CacheDir (class), 30–32
    SCons.CacheDir.CachePushFunc (function), 30
    SCons.CacheDir.CacheRetrieveFunc (function), 30
    SCons.CacheDir.CacheRetrieveString (function), 30
  SCons.Conftest (module), 33–37
  SCons.cpp (module), 380–387
  SCons.dblite (module), 388–389
    SCons.dblite.corruption_warning (function), 388
    SCons.dblite.dblite (class), 388–389
    SCons.dblite.is_string (function), 388
    SCons.dblite.open (function), 388
    SCons.dblite.unicode (function), 388
  SCons.Debug (module), 38–39
    SCons.Debug.caller_stack (function), 38
    SCons.Debug.caller_trace (function), 38
    SCons.Debug.countLoggedInstances (function), 38
    SCons.Debug.dump_caller_counts (function), 38
    SCons.Debug.dumpLoggedInstances (function), 38
    SCons.Debug.fetchLoggedInstances (function), 38
    SCons.Debug.func_shorten (function), 38
    SCons.Debug.listLoggedInstances (function), 38
    SCons.Debug.logInstanceCreation (function), 38
    SCons.Debug.memory (function), 38
    SCons.Debug.string_to_classes (function), 38
    SCons.Debug.Trace (function), 38
  SCons.Defaults (module), 40–44
  SCons.Environment (module), 45–73
    SCons.Environment.alias_builder (function), 45
    SCons.Environment.apply_tools (function), 45
    SCons.Environment.Base (class), 53–61, 64–73
    SCons.Environment.BuilderDict (class), 48–49
    SCons.Environment.copy_non_reserved_keywords (function), 45
    SCons.Environment.default_copy_from_cache (function), 45
    SCons.Environment.default_decide_source (function), 45
    SCons.Environment.default_decide_target (function), 45
    SCons.Environment.is_valid_construction_var
SCons.Environment.MethodWrapper (class), 46–47
SCons.Environment.NoSubstitutionProxy (function), 45
SCons.Environment.OverrideEnvironment (class), 61–64
SCons.Environment.SubstitutionEnvironment (class), 49–53
SCons.Errors (module), 74–81
SCons.Executor (module), 82–92
SCons.Executor.AddBatchExecutor (function), 82
SCons.Executor.Batch (class), 82–83
SCons.Executor.Executor (class), 86–90
SCons.Executor.get_NullEnvironment (function), 82
SCons.Executor.GetBatchExecutor (function), 82
SCons.Executor.Null (class), 90–92
SCons.Executor.rfile (function), 82
SCons.Executor.TSList (class), 83–85
SCons.Executor.TSObject (class), 85–86
SCons.exitfuncs (module), 390
SCons.exitfuncs.register (function), 390
SCons.Job (module), 93–99
SCons.Job.InterruptState (class), 93–94
SCons.Job.Jobs (class), 94–95
SCons.Job.Parallel (class), 98–99
SCons.Job.Serial (class), 95–96
SCons.Job.ThreadPool (class), 97–98
SCons.Job.Worker (class), 96–97
SCons.Memoize (module), 100–107
SCons.Node (package), 108–127
SCons.Node.FS (module), 135–179
SCons.Node.Python (module), 180–185
SCons.PathList (module), 186
SCons.PathList.node__conv (function), 186
SCons.PathList.PathList (function), 186
SCons.Scanner (module)
SCons.Scanner.Base (class), 210–213
SCons.Scanner.Classic (class), 220–223
SCons.Scanner.ClassicCPP (class), 223–224
SCons.Scanner.Current (class), 217–220
SCons.Scanner.FindPathDirs (class), 210
SCons.Scanner.Scanner (function), 209
SCons.Scanner.Selector (class), 213–217
SCons.Scanner (package), 209–224
SCons.Scanner.C (module), 225–227
SCons.Scanner.D (module), 228–231
SCons.Scanner.Dir (module), 232–233
SCons.Scanner.Fortran (module), 234–238
SCons.Scanner.IDL (module), 239
SCons.Scanner.LaTeX (module), 240–246
SCons.Scanner.Prog (module), 247
SCons.Scanner.RC (module), 248
SCons.SConf (module), 187–201
SCons.SConf.CheckCC (function), 188
SCons.SConf.CheckCHeader (function), 188
SCons.SConf.CheckContext (class), 199–201
SCons.SConf.CheckCXX (function), 188
SCons.SConf.CheckCXXHeader (function), 188
SCons.SConf.CheckDeclaration (function), 187
SCons.SConf.CheckFunc (function), 187
SCons.SConf.CheckHeader (function), 187
SCons.SConf.CheckLib (function), 188
SCons.SConf.CheckLibWithHeader (function), 188
SCons.SConf.CheckSHCC (function), 188
SCons.SConf.CheckSHCXX (function), 188
SCons.SConf.CheckType (function), 187
SCons.SConf.CheckTypeSize (function), 187
SCons.SConf.ConfigureCacheError (class), 192–193
SCons.SConf.ConfigureDryRunError (class), 191–192
SCons.SConf.CreateConfigHBuilder (func-
SCons.Action.ActionBase.get_contents (method), (method), 10
SCons.Action.default_exitstatfunc (function),
SCons.Action.ActionBase.get_targets (method), 6
SCons.Action.FunctionAction (class), 12–14
SCons.Action.FunctionAction.execute (method),
SCons.Action.ActionBase.no_batch_key (method)
SCons.Action.FunctionAction.get_implicit_deps
SCons.Action.ActionBase.ppresub_lines (method)
SCons.Action.FunctionAction.get_presig

SCons.Action.ActionCaller (class), 15–16
SCons.Action.ActionCaller.__call__ (method)
SCons.Action.ActionCaller.get_contents (method),
SCons.Action.ActionCaller.strfunction
SCons.Action.ActionCaller.subst (method),
SCons.Action.ActionCaller.subst_kw (method)
SCons.Action.FunctionAction.get_presig (method),
SCons.Action.ListAction (class), 14–15
SCons.Action.MethodAction (class), 8
SCons.Action.MethodAction.get_implicit_deps
SCons.Action.MethodAction.get_presig
SCons.Action.remove_set_lineno_codes (function), 6
SCons.Action.rfile (function)
SCons.Conftest.CheckBuilder (function), 33
SCons.Conftest.CheckCC (function), 33
SCons.Conftest.CheckCXX (function), 33
SCons.Conftest.CheckDeclaration (function), 35
SCons.Conftest.CheckFunc (function), 34
SCons.Conftest.CheckHeader (function), 34
SCons.Conftest.CheckLib (function), 36
SCons.Conftest.CheckSHCXX (function), 34
SCons.Conftest.CheckType (function), 34
SCons.Conftest.CheckTypeSize (function), 35
SCons.cpp.CPP_to_Python (function), 380
SCons.cpp.CPP_to_Python_Ops_Sub (function), 380
SCons.cpp.DumbPreProcessor (class), 386–387
SCons.cpp.FunctionEvaluator (class), 380–381
SCons.cpp.PreProcessor.restore (method),
SCons.cpp.PreProcessor.__call__ (method), 384
SCons.cpp.PreProcessor.save (method), 384
SCons.cpp.PreProcessor.scons_current_file
SCons.cpp.PreProcessor.start_handling_includes
SCons.cpp.PreProcessor.stop_handling_includes
SCons.cpp.PreProcessor.tupleize (method), 385
SCons.cpp.PreProcessor.do_elif (method), 382
SCons.cpp.PreProcessor.do_else (method), 382
SCons.cpp.PreProcessor.do_endif (method), 382
SCons.cpp.PreProcessor.do_if (method), 382
SCons.cpp.PreProcessor.do_ifdef (method), 383
SCons.cpp.PreProcessor.do ifndef (method), 383
SCons.cpp.PreProcessor.do_import (method), 383
SCons.cpp.PreProcessor.do_include (method), 383
SCons.cpp.PreProcessor.do_nothing (method), 383
SCons.cpp.PreProcessor.do_undef (method), 383
SCons.cpp.PreProcessor.eval_expression (method), 384
SCons.cpp.PreProcessor.finalize_result (method), 384
SCons.cpp.PreProcessor.find_include_file (method), 384
SCons.cpp.PreProcessor.initialize_result (method), 384
SCons.cpp.PreProcessor.process_contents (method), 384
SCons.cpp.PreProcessor.read_file (method), 384
SCons.cpp.PreProcessor.resolve_include (method), 384
SCons.cpp.PreProcessor.scons_current_file (method), 385
SCons.Errors.MSVCError (class), 79–80
SCons.Errors.StopError (class), 78–79
SCons.Errors.UserError (class), 77–78
SCons.Memoize.CountDict (class), 104–105
  SCons.Memoize.CountDict.__call__ (method), 105
SCons.Memoize.Counter (class), 102–103
  SCons.Memoize.Counter.__cmp__ (method), 103
  SCons.Memoize.Counter.display (method), 103
SCons.Memoize.CountValue (class), 103–104
  SCons.Memoize.CountValue.__call__ (method), 104
SCons.Memoize.Dump (function), 102
SCons.Memoize.EnableMemoization (function), 102
SCons.Memoize.Memoized_Metaclass (class), 106–107
SCons.Memoize.Memoizer (class), 105–106
SCons.Node.Annotate (function), 108
SCons.Node.BuildInfoBase (class), 110–111
  SCons.Node.BuildInfoBase.merge (method), 111
SCons.Node.classname (function), 108
SCons.Node.do_nothing (function), 108
SCons.Node.get_children (function), 108
SCons.Node.ignore_cycle (function), 108
SCons.Node.Node (class), 111–125
  SCons.Node.Node.add_dependency (method), 112
  SCons.Node.Node.add_ignore (method), 112
  SCons.Node.Node.add_prerequisite (method), 112
  SCons.Node.Node.add_source (method), 112
  SCons.Node.Node.add_to_implicit (method), 112
  SCons.Node.Node.add_to_waiting_parents (method), 112
  SCons.Node.Node.add_to_waiting_s_e (method), 112
  SCons.Node.Node.add_wkid (method), 112
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCons.Node.Node.get_found_includes</td>
<td>117</td>
</tr>
<tr>
<td>SCons.Node.Node.get_implicit_deps</td>
<td>117</td>
</tr>
<tr>
<td>SCons.Node.Node.get_ninfo</td>
<td>117</td>
</tr>
<tr>
<td>SCons.Node.Node.get_state</td>
<td>117</td>
</tr>
<tr>
<td>SCons.Node.Node.get_stored_implicit</td>
<td>117</td>
</tr>
<tr>
<td>SCons.Node.Node.get_stored_info</td>
<td>118</td>
</tr>
<tr>
<td>SCons.Node.Node.get_string</td>
<td>118</td>
</tr>
<tr>
<td>SCons.Node.Node.get_suffix</td>
<td>119</td>
</tr>
<tr>
<td>SCons.Node.Node.has_builder</td>
<td>119</td>
</tr>
<tr>
<td>SCons.Node.Node.has_explicit_builder</td>
<td>119</td>
</tr>
<tr>
<td>SCons.Node.Node.is_derived</td>
<td>119</td>
</tr>
<tr>
<td>SCons.Node.Node.is_literal</td>
<td>120</td>
</tr>
<tr>
<td>SCons.Node.Node.is_up_to_date</td>
<td>120</td>
</tr>
<tr>
<td>SCons.Node.Node.make_ready</td>
<td>120</td>
</tr>
<tr>
<td>SCons.Node.Node.missing</td>
<td>120</td>
</tr>
<tr>
<td>SCons.Node.Node.new_binfo</td>
<td>121</td>
</tr>
<tr>
<td>SCons.Node.Node.new_ninfo</td>
<td>121</td>
</tr>
<tr>
<td>SCons.Node.Node.postprocess</td>
<td>121</td>
</tr>
<tr>
<td>SCons.Node.Node.prepare</td>
<td>121</td>
</tr>
<tr>
<td>SCons.Node.Node.push_to_cache</td>
<td>121</td>
</tr>
<tr>
<td>SCons.Node.Node.remove</td>
<td>122</td>
</tr>
<tr>
<td>SCons.Node.NodeInfoBase.format</td>
<td>110</td>
</tr>
<tr>
<td>SCons.Node.NodeInfoBase.merge</td>
<td>110</td>
</tr>
<tr>
<td>SCons.Node.NodeList</td>
<td>125</td>
</tr>
<tr>
<td>SCons.Node.Walker.get_next</td>
<td>127</td>
</tr>
<tr>
<td>SCons.Node.Walker.is_done</td>
<td>127</td>
</tr>
<tr>
<td>SCons.Scanner.Dir.DirEntryScanner</td>
<td>232</td>
</tr>
<tr>
<td>Function/Method</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>SCons.Scanner.Dir.DirScanner</td>
<td>232</td>
</tr>
<tr>
<td>SCons.Scanner.Dir.do_not_scan</td>
<td>232</td>
</tr>
<tr>
<td>SCons.Scanner.Dir.only_dirs</td>
<td>232</td>
</tr>
<tr>
<td>SCons.Scanner.Dir.scan_in_memory</td>
<td>232</td>
</tr>
<tr>
<td>SCons.Scanner.Dir.scan_on_disk</td>
<td>232</td>
</tr>
<tr>
<td>SCons.Script.Interactive.interact</td>
<td>258</td>
</tr>
<tr>
<td>SCons.Script.Interactive.SConsInteractiveCmd.do_build</td>
<td>259</td>
</tr>
<tr>
<td>SCons.Script.Interactive.SConsInteractiveCmd.do_clean</td>
<td>259</td>
</tr>
<tr>
<td>SCons.Script.Interactive.SConsInteractiveCmd.do_EOF</td>
<td>259</td>
</tr>
<tr>
<td>SCons.Script.Interactive.SConsInteractiveCmd.do_exit</td>
<td>259</td>
</tr>
<tr>
<td>SCons.Script.Interactive.SConsInteractiveCmd.do_shell</td>
<td>260</td>
</tr>
<tr>
<td>SCons.Script.Interactive.SConsInteractiveCmd.do_version</td>
<td>260</td>
</tr>
<tr>
<td>SCons.Subst.CmdStringHolder</td>
<td>288</td>
</tr>
<tr>
<td>SCons.Subst.CmdStringHolder.escape</td>
<td>289</td>
</tr>
<tr>
<td>SCons.Subst.CmdStringHolder.is_literal</td>
<td>293</td>
</tr>
<tr>
<td>SCons.Subst.escape_list</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.Literal</td>
<td>286</td>
</tr>
<tr>
<td>SCons.Subst.Literal.<strong>eq</strong></td>
<td>286</td>
</tr>
<tr>
<td>SCons.Subst.Literal.<strong>neq</strong></td>
<td>286</td>
</tr>
<tr>
<td>SCons.Subst.Literal.escape</td>
<td>286</td>
</tr>
<tr>
<td>SCons.Subst.Literal.is_literal</td>
<td>286</td>
</tr>
<tr>
<td>SCons.Subst.NLWrapper</td>
<td>286</td>
</tr>
<tr>
<td>SCons.Subst.NullNodeList</td>
<td>293</td>
</tr>
<tr>
<td>SCons.Subst.quote_spaces</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.raise_exception</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.scons_subst</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.scons_subst_list</td>
<td>285</td>
</tr>
<tr>
<td>SCons.Subst.scons_subst_once</td>
<td>285</td>
</tr>
<tr>
<td>SCons.Subst.Subst.SetAllowableExceptions</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.SpecialAttrWrapper</td>
<td>287</td>
</tr>
<tr>
<td>SCons.Subst.SpecialAttrWrapper.escape</td>
<td>287</td>
</tr>
<tr>
<td>SCons.Subst.SpecialAttrWrapper.for_signature</td>
<td>287</td>
</tr>
<tr>
<td>SCons.Subst.SpecialAttrWrapper.is_literal</td>
<td>287</td>
</tr>
<tr>
<td>SCons.Subst.subst_dict</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.Target_or_Source</td>
<td>292</td>
</tr>
<tr>
<td>SCons.Subst.Target_or_Source.<strong>getattr</strong></td>
<td>293</td>
</tr>
<tr>
<td>SCons.Subst.Targets_or_Sources</td>
<td>290</td>
</tr>
<tr>
<td>SCons.Subst.CmdStringHolder.escaped</td>
<td>291</td>
</tr>
<tr>
<td>SCons.Subst.CmdStringHolder.is_literal.escaped</td>
<td>291</td>
</tr>
<tr>
<td>SCons.Subst.escape_list</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Util.containsAll</td>
<td>306</td>
</tr>
<tr>
<td>SCons.Util.containsAny</td>
<td>306</td>
</tr>
<tr>
<td>SCons.Util.containsOnly</td>
<td>306</td>
</tr>
<tr>
<td>SCons.Util.DoExit</td>
<td>318</td>
</tr>
<tr>
<td>SCons.Util.Delegate</td>
<td>318</td>
</tr>
<tr>
<td>SCons.Util.DoExit.get</td>
<td>318</td>
</tr>
<tr>
<td>SCons.Util.DisplayEngine</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Subst.NLWrapper</td>
<td>289</td>
</tr>
<tr>
<td>SCons.Subst.NullNodeList</td>
<td>293</td>
</tr>
<tr>
<td>SCons.Subst.quote_spaces</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.raise_exception</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.scons_subst</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.scons_subst_list</td>
<td>285</td>
</tr>
<tr>
<td>SCons.Subst.scons_subst_once</td>
<td>285</td>
</tr>
<tr>
<td>SCons.Subst.Subst.SetAllowableExceptions</td>
<td>284</td>
</tr>
<tr>
<td>SCons.Subst.SpecialAttrWrapper</td>
<td>287</td>
</tr>
<tr>
<td>SCons.Subst.SpecialAttrWrapper.escape</td>
<td>287</td>
</tr>
<tr>
<td>SCons.Subst.SpecialAttrWrapper.for_signature</td>
<td>287</td>
</tr>
</tbody>
</table>
SCons.Warnings.DeprecatedDebugOptionsWarning (class), 376–377
SCons.Warnings.DeprecatedOptionsWarning (class), 361
SCons.Warnings.DeprecatedSigModuleWarning (class), 377–378
SCons.Warnings.DeprecatedSourceCodeWarning (class), 369–370
SCons.Warnings.DeprecatedSourceSignaturesWarning (class), 374–375
SCons.Warnings.DeprecatedTargetSignaturesWarning (class), 376
SCons.Warnings.DeprecatedWarning (class), 366–367
SCons.Warnings.DuplicateEnvironmentWarning (class), 351–352
SCons.Warnings.enableWarningClass (function), 344
SCons.Warnings.FortranCxxMixWarning (class), 364–365
SCons.Warnings.FutureDeprecationWarning (class), 365–366
SCons.Warnings.FutureReservedVariableWarning (class), 352–353
SCons.Warnings.LinkWarning (class), 353
SCons.Warnings.MandatoryDeprecationWarning (class), 367–368
SCons.Warnings.MisleadingKeywordsWarning (class), 353–354
SCons.Warnings.MissingSConscriptWarning (class), 354–355
SCons.Warnings.NoMD5ModuleWarning (class), 355–356
SCons.Warnings.NoMetaClassSupportWarning (class), 356–357
SCons.Warnings.NoObjectCountWarning (class), 357–358
SCons.Warnings.NoParallelSupportWarning (class), 358–359
SCons.Warnings.process__warn__strings (function), 344
SCons.Warnings.PythonVersionWarning (class), 368–369
SCons.Warnings.ReservedVariableWarning (class),