# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package SCons</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Modules</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Variables</td>
<td>4</td>
</tr>
<tr>
<td>Module SCons.Action</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Functions</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Variables</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Class ActionBase</td>
<td>6</td>
</tr>
<tr>
<td>2.3.1 Methods</td>
<td>7</td>
</tr>
<tr>
<td>2.3.2 Properties</td>
<td>7</td>
</tr>
<tr>
<td>2.4 Class CommandAction</td>
<td>8</td>
</tr>
<tr>
<td>2.4.1 Methods</td>
<td>8</td>
</tr>
<tr>
<td>2.4.2 Properties</td>
<td>9</td>
</tr>
<tr>
<td>2.5 Class CommandGeneratorAction</td>
<td>9</td>
</tr>
<tr>
<td>2.5.1 Methods</td>
<td>10</td>
</tr>
<tr>
<td>2.5.2 Properties</td>
<td>11</td>
</tr>
<tr>
<td>2.6 Class LazyAction</td>
<td>11</td>
</tr>
<tr>
<td>2.6.1 Methods</td>
<td>11</td>
</tr>
<tr>
<td>2.6.2 Properties</td>
<td>12</td>
</tr>
<tr>
<td>2.7 Class FunctionAction</td>
<td>13</td>
</tr>
<tr>
<td>2.7.1 Methods</td>
<td>13</td>
</tr>
<tr>
<td>2.7.2 Properties</td>
<td>14</td>
</tr>
<tr>
<td>2.8 Class ListAction</td>
<td>14</td>
</tr>
<tr>
<td>2.8.1 Methods</td>
<td>14</td>
</tr>
<tr>
<td>2.8.2 Properties</td>
<td>15</td>
</tr>
<tr>
<td>2.9 Class ActionCaller</td>
<td>15</td>
</tr>
<tr>
<td>2.9.1 Methods</td>
<td>16</td>
</tr>
<tr>
<td>2.9.2 Properties</td>
<td>16</td>
</tr>
<tr>
<td>2.10 Class ActionFactory</td>
<td>17</td>
</tr>
<tr>
<td>2.10.1 Methods</td>
<td>17</td>
</tr>
<tr>
<td>2.10.2 Properties</td>
<td>17</td>
</tr>
<tr>
<td>Module SCons.Builder</td>
<td>18</td>
</tr>
<tr>
<td>3.1 Functions</td>
<td>19</td>
</tr>
<tr>
<td>3.2 Variables</td>
<td>20</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>3.3</td>
<td>Class DictCmdGenerator</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Methods</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Class Variables</td>
</tr>
<tr>
<td>3.4</td>
<td>Class CallableSelector</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Methods</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Class Variables</td>
</tr>
<tr>
<td>3.5</td>
<td>Class DictEmitter</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Methods</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Class Variables</td>
</tr>
<tr>
<td>3.6</td>
<td>Class ListEmitter</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Methods</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Properties</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Class Variables</td>
</tr>
<tr>
<td>3.7</td>
<td>Class OverrideWarner</td>
</tr>
<tr>
<td>3.7.1</td>
<td>Methods</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Class Variables</td>
</tr>
<tr>
<td>3.8</td>
<td>Class EmitterProxy</td>
</tr>
<tr>
<td>3.8.1</td>
<td>Methods</td>
</tr>
<tr>
<td>3.8.2</td>
<td>Properties</td>
</tr>
<tr>
<td>3.9</td>
<td>Class BuilderBase</td>
</tr>
<tr>
<td>3.9.1</td>
<td>Methods</td>
</tr>
<tr>
<td>3.9.2</td>
<td>Properties</td>
</tr>
<tr>
<td>3.10</td>
<td>Class CompositeBuilder</td>
</tr>
<tr>
<td>3.10.1</td>
<td>Methods</td>
</tr>
<tr>
<td>3.10.2</td>
<td>Properties</td>
</tr>
<tr>
<td>4</td>
<td>Module SCons.CacheDir</td>
</tr>
<tr>
<td>4.1</td>
<td>Functions</td>
</tr>
<tr>
<td>4.2</td>
<td>Variables</td>
</tr>
<tr>
<td>4.3</td>
<td>Class CacheDir</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Methods</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Properties</td>
</tr>
<tr>
<td>5</td>
<td>Module SCons.Conftest</td>
</tr>
<tr>
<td>5.1</td>
<td>Functions</td>
</tr>
<tr>
<td>5.2</td>
<td>Variables</td>
</tr>
<tr>
<td>6</td>
<td>Module SCons.Debug</td>
</tr>
<tr>
<td>6.1</td>
<td>Functions</td>
</tr>
<tr>
<td>6.2</td>
<td>Variables</td>
</tr>
<tr>
<td>7</td>
<td>Module SCons.Defaults</td>
</tr>
<tr>
<td>7.1</td>
<td>Functions</td>
</tr>
<tr>
<td>7.2</td>
<td>Variables</td>
</tr>
<tr>
<td>7.3</td>
<td>Class NullCmdGenerator</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Methods</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Properties</td>
</tr>
<tr>
<td>7.4</td>
<td>Class Variable_Method_Caller</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Methods</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Properties</td>
</tr>
<tr>
<td>8</td>
<td>Module SCons.Environment</td>
</tr>
</tbody>
</table>
## 8.1 Functions

8.1.1 Functions

8.1.2 Variables

8.2 Variables

8.3 Class MethodWrapper

8.3.1 Methods

8.3.2 Properties

8.4 Class BuilderWrapper

8.4.1 Methods

8.4.2 Properties

8.5 Class BuilderDict

8.5.1 Methods

8.5.2 Class Variables

8.6 Class SubstitutionEnvironment

8.6.1 Methods

8.6.2 Properties

8.7 Class Base

8.7.1 Methods

8.7.2 Properties

8.8 Class OverrideEnvironment

8.8.1 Methods

8.8.2 Properties

8.9 Class Base

8.9.1 Methods

8.9.2 Properties

## 9 Module SCons.Errors

9.1 Functions

9.2 Variables

9.3 Class BuildError

9.3.1 Methods

9.3.2 Properties

9.4 Class InternalError

9.4.1 Methods

9.4.2 Properties

9.5 Class UserError

9.5.1 Methods

9.5.2 Properties

9.6 Class StopError

9.6.1 Methods

9.6.2 Properties

9.7 Class EnvironmentError

9.7.1 Methods

9.7.2 Properties

9.8 Class MSVCError

9.8.1 Methods

9.8.2 Properties

9.9 Class ExplicitExit

9.9.1 Methods

9.9.2 Properties

## 10 Module SCons.Executor

10.1 Functions

10.2 Variables
## CONTENTS

### 10.3 Class Batch
- 10.3.1 Methods .................................................. 83
- 10.3.2 Properties ................................................ 83

### 10.4 Class TSLList
- 10.4.1 Methods .................................................. 84
- 10.4.2 Properties ................................................ 85

### 10.5 Class TSOObject
- 10.5.1 Methods .................................................. 86
- 10.5.2 Properties ................................................ 86

### 10.6 Class Executor
- 10.6.1 Methods .................................................. 86
- 10.6.2 Properties ................................................ 90

### 10.7 Class NullEnvironment
- 10.7.1 Methods .................................................. 91
- 10.7.2 Properties ................................................ 91

### 10.8 Class Null
- 10.8.1 Methods .................................................. 91
- 10.8.2 Properties ................................................ 92

### 11 Module SCons.Job

#### 11.1 Variables ................................................. 94

#### 11.2 Class InterruptState
- 11.2.1 Methods .................................................. 94
- 11.2.2 Properties ................................................ 95

#### 11.3 Class Jobs
- 11.3.1 Methods .................................................. 95
- 11.3.2 Properties ................................................ 96

#### 11.4 Class Serial
- 11.4.1 Methods .................................................. 96
- 11.4.2 Properties ................................................ 97

#### 11.5 Class Worker
- 11.5.1 Methods .................................................. 97
- 11.5.2 Properties ................................................ 99

#### 11.6 Class ThreadPool
- 11.6.1 Methods .................................................. 99
- 11.6.2 Properties ................................................ 100

#### 11.7 Class Parallel
- 11.7.1 Methods .................................................. 100
- 11.7.2 Properties ................................................ 101

### 12 Module SCons.Memoize

#### 12.1 Functions ................................................. 102

#### 12.2 Variables ................................................. 103

#### 12.3 Class Counter
- 12.3.1 Methods .................................................. 104
- 12.3.2 Properties ................................................ 105

#### 12.4 Class CountValue
- 12.4.1 Methods .................................................. 105
- 12.4.2 Properties ................................................ 106

#### 12.5 Class CountDict
- 12.5.1 Methods .................................................. 106
## 15.5.1 Methods

15.5.2 Properties

15.5.3 Class Variables

15.6 Class Base

15.6.1 Methods

15.6.2 Properties

15.6.3 Instance Variables

15.7 Class Entry

15.7.1 Methods

15.7.2 Properties

15.7.3 Instance Variables

15.8 Class LocalFS

15.8.1 Methods

15.8.2 Properties

15.9 Class FS

15.9.1 Methods

15.9.2 Properties

15.10 Class DirNodeInfo

15.10.1 Methods

15.10.2 Properties

15.10.3 Class Variables

15.11 Class DirBuildInfo

15.11.1 Methods

15.11.2 Properties

15.11.3 Class Variables

15.12 Class Dir

15.12.1 Methods

15.12.2 Properties

15.12.3 Instance Variables

15.13 Class RootDir

15.13.1 Methods

15.13.2 Properties

15.13.3 Instance Variables

15.14 Class FileInfo

15.14.1 Methods

15.14.2 Properties

15.14.3 Class Variables

15.15 Class FileBuildInfo

15.15.1 Methods

15.15.2 Properties

15.15.3 Class Variables

15.16 Class File

15.16.1 Methods

15.16.2 Properties

15.16.3 Class Variables

15.16.4 Instance Variables

15.17 Class FileFinder

15.17.1 Methods

15.17.2 Properties

16 Module SCons.Node.Python

16.1 Variables
19.3.1 Methods ................................................................. 209
19.3.2 Properties .......................................................... 209
19.3.3 Class Variables ...................................................... 209
19.4 Class Base ............................................................ 210
19.4.1 Methods ............................................................. 210
19.4.2 Properties .......................................................... 211
19.5 Class DB ............................................................... 211
19.5.1 Methods ............................................................. 211
19.5.2 Properties .......................................................... 211
19.6 Class Dir ............................................................... 212
19.6.1 Methods ............................................................. 212
19.6.2 Properties .......................................................... 212
19.7 Class DirFile .......................................................... 213
19.7.1 Methods ............................................................. 213
19.7.2 Properties .......................................................... 213
19.8 Class DB ............................................................... 214
19.8.1 Methods ............................................................. 214
19.8.2 Properties .......................................................... 214

20 Package SCons.Scanner .................................................. 215
20.1 Modules ................................................................ 215
20.2 Functions ................................................................ 215
20.3 Variables ................................................................ 215
20.4 Class FindPathDirs .................................................... 216
  20.4.1 Methods ............................................................. 216
  20.4.2 Properties .......................................................... 216
20.5 Class Base ............................................................ 216
  20.5.1 Methods ............................................................. 217
  20.5.2 Properties .......................................................... 219
20.6 Class Selector ......................................................... 219
  20.6.1 Methods ............................................................. 222
  20.6.2 Properties .......................................................... 223
20.7 Class Current .......................................................... 223
  20.7.1 Methods ............................................................. 225
  20.7.2 Properties .......................................................... 226
20.8 Class Classic ........................................................... 226
  20.8.1 Methods ............................................................. 228
  20.8.2 Properties .......................................................... 229
20.9 Class ClassicCPP ...................................................... 229
  20.9.1 Methods ............................................................. 230
  20.9.2 Properties .......................................................... 230

21 Module SCons.Scanner.C .................................................. 231
21.1 Functions ................................................................ 231
21.2 Variables ................................................................ 231
21.3 Class SConsCPPScanner .............................................. 231
  21.3.1 Methods ............................................................. 232
  21.3.2 Properties .......................................................... 232
21.4 Class SConsCPPScannerWrapper .................................. 233
  21.4.1 Methods ............................................................. 233
  21.4.2 Properties .......................................................... 233
22 Module SCons.Scanner.D ........................................... 234
  22.1 Functions .................................................. 234
  22.2 Variables .................................................. 234
  22.3 Class D ..................................................... 234
    22.3.1 Methods ........................................... 236
    22.3.2 Properties ........................................ 237

23 Module SCons.Scanner.Dir ........................................ 238
  23.1 Functions .................................................. 238
  23.2 Variables .................................................. 238

24 Module SCons.Scanner.Fortran .................................... 240
  24.1 Functions .................................................. 240
  24.2 Variables .................................................. 240
  24.3 Class F90Scanner .......................................... 240
    24.3.1 Methods ........................................... 243
    24.3.2 Properties ........................................ 244

25 Module SCons.Scanner.IDL ........................................ 245
  25.1 Functions .................................................. 245
  25.2 Variables .................................................. 245

26 Module SCons.Scanner.LaTeX ..................................... 246
  26.1 Functions .................................................. 246
  26.2 Variables .................................................. 246
  26.3 Class FindENVPathDirs .................................... 246
    26.3.1 Methods ........................................... 247
    26.3.2 Properties ........................................ 247
  26.4 Class LaTeX ................................................ 247
    26.4.1 Methods ........................................... 250
    26.4.2 Properties ........................................ 251
    26.4.3 Class Variables ................................... 251

27 Module SCons.Scanner.Prog ...................................... 253
  27.1 Functions .................................................. 253
  27.2 Variables .................................................. 253

28 Module SCons.Scanner.RC ........................................ 254
  28.1 Functions .................................................. 254
  28.2 Variables .................................................. 254

29 Package SCons.Script ........................................... 255
  29.1 Modules .................................................... 255
  29.2 Functions .................................................. 255
  29.3 Variables .................................................. 255
  29.4 Class TargetList ......................................... 262
    29.4.1 Methods ........................................... 262
    29.4.2 Properties ........................................ 263
    29.4.3 Class Variables ................................... 263

30 Module SCons.Script.Interactive ................................ 264
  30.1 Functions .................................................. 264
  30.2 Variables .................................................. 264
30.3 Class SConsInteractiveCmd .................................................. 264
30.3.1 Methods ................................................................. 265
30.3.2 Class Variables ......................................................... 266

31 Module SCons.Script.Main .................................................... 267
31.1 Functions ........................................................................... 267
31.2 Variables ........................................................................... 268
31.3 Class SConsPrintHelpException ......................................... 269
  31.3.1 Methods ...................................................................... 269
  31.3.2 Properties .................................................................... 269
31.4 Class Progressor ................................................................. 270
  31.4.1 Methods ...................................................................... 270
  31.4.2 Properties .................................................................... 270
  31.4.3 Class Variables ............................................................ 270
31.5 Class BuildTask ................................................................. 271
  31.5.1 Methods ...................................................................... 271
  31.5.2 Properties .................................................................... 273
  31.5.3 Class Variables ............................................................ 273
31.6 Class CleanTask ................................................................. 273
  31.6.1 Methods ...................................................................... 274
  31.6.2 Properties .................................................................... 275
31.7 Class QuestionTask ............................................................. 275
  31.7.1 Methods ...................................................................... 276
  31.7.2 Properties .................................................................... 277
31.8 Class TreePrinter ............................................................... 277
  31.8.1 Methods ...................................................................... 277
  31.8.2 Properties .................................................................... 277
31.9 Class FakeOptionParser ..................................................... 278
  31.9.1 Methods ...................................................................... 278
  31.9.2 Properties .................................................................... 278
  31.9.3 Class Variables ............................................................ 278
31.10 Class Stats ....................................................................... 278
  31.10.1 Methods .................................................................... 279
  31.10.2 Properties .................................................................... 279
31.11 Class CountStats ............................................................... 279
  31.11.1 Methods .................................................................... 279
  31.11.2 Properties .................................................................... 280
31.12 Class MemStats ............................................................... 280
  31.12.1 Methods .................................................................... 280
  31.12.2 Properties .................................................................... 280

32 Module SCons.Script.SConscript’ .............................................. 281
32.1 Functions ........................................................................... 281
32.2 Variables ........................................................................... 282
32.3 Class SConscriptReturn ...................................................... 282
  32.3.1 Methods ...................................................................... 282
  32.3.2 Properties .................................................................... 283
32.4 Class Frame ....................................................................... 283
  32.4.1 Methods ...................................................................... 283
  32.4.2 Properties .................................................................... 283
32.5 Class SConsEnvironment .................................................... 284
  32.5.1 Methods ...................................................................... 284
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.5.2</td>
<td>Properties</td>
<td>285</td>
</tr>
<tr>
<td>32.6</td>
<td>Class DefaultEnvironmentCall</td>
<td>286</td>
</tr>
<tr>
<td>32.6.1</td>
<td>Methods</td>
<td>286</td>
</tr>
<tr>
<td>32.6.2</td>
<td>Properties</td>
<td>286</td>
</tr>
<tr>
<td>33</td>
<td>Module SCons.Sig</td>
<td>287</td>
</tr>
<tr>
<td>33.1</td>
<td>Variables</td>
<td>287</td>
</tr>
<tr>
<td>33.2</td>
<td>Class MD5Null</td>
<td>287</td>
</tr>
<tr>
<td>33.2.1</td>
<td>Methods</td>
<td>287</td>
</tr>
<tr>
<td>33.2.2</td>
<td>Properties</td>
<td>288</td>
</tr>
<tr>
<td>33.3</td>
<td>Class TimeStampNull</td>
<td>288</td>
</tr>
<tr>
<td>33.3.1</td>
<td>Methods</td>
<td>288</td>
</tr>
<tr>
<td>33.3.2</td>
<td>Properties</td>
<td>288</td>
</tr>
<tr>
<td>34</td>
<td>Module SCons.Subst</td>
<td>290</td>
</tr>
<tr>
<td>34.1</td>
<td>Functions</td>
<td>290</td>
</tr>
<tr>
<td>34.2</td>
<td>Variables</td>
<td>291</td>
</tr>
<tr>
<td>34.3</td>
<td>Class Literal</td>
<td>292</td>
</tr>
<tr>
<td>34.3.1</td>
<td>Methods</td>
<td>292</td>
</tr>
<tr>
<td>34.3.2</td>
<td>Properties</td>
<td>293</td>
</tr>
<tr>
<td>34.4</td>
<td>Class SpecialAttrWrapper</td>
<td>293</td>
</tr>
<tr>
<td>34.4.1</td>
<td>Methods</td>
<td>293</td>
</tr>
<tr>
<td>34.4.2</td>
<td>Properties</td>
<td>294</td>
</tr>
<tr>
<td>34.5</td>
<td>Class CmdStringHolder</td>
<td>294</td>
</tr>
<tr>
<td>34.5.1</td>
<td>Methods</td>
<td>294</td>
</tr>
<tr>
<td>34.5.2</td>
<td>Properties</td>
<td>295</td>
</tr>
<tr>
<td>34.5.3</td>
<td>Class Variables</td>
<td>295</td>
</tr>
<tr>
<td>34.6</td>
<td>Class NLWrapper</td>
<td>296</td>
</tr>
<tr>
<td>34.6.1</td>
<td>Methods</td>
<td>296</td>
</tr>
<tr>
<td>34.6.2</td>
<td>Properties</td>
<td>296</td>
</tr>
<tr>
<td>34.7</td>
<td>Class Targets_or_Sources</td>
<td>297</td>
</tr>
<tr>
<td>34.7.1</td>
<td>Methods</td>
<td>297</td>
</tr>
<tr>
<td>34.7.2</td>
<td>Properties</td>
<td>298</td>
</tr>
<tr>
<td>34.7.3</td>
<td>Class Variables</td>
<td>298</td>
</tr>
<tr>
<td>34.8</td>
<td>Class Target_or_Source</td>
<td>299</td>
</tr>
<tr>
<td>34.8.1</td>
<td>Methods</td>
<td>299</td>
</tr>
<tr>
<td>34.8.2</td>
<td>Properties</td>
<td>299</td>
</tr>
<tr>
<td>34.9</td>
<td>Class NullNodeList</td>
<td>300</td>
</tr>
<tr>
<td>34.9.1</td>
<td>Methods</td>
<td>300</td>
</tr>
<tr>
<td>34.9.2</td>
<td>Properties</td>
<td>300</td>
</tr>
<tr>
<td>35</td>
<td>Module SCons.Taskmaster</td>
<td>301</td>
</tr>
<tr>
<td>35.1</td>
<td>Functions</td>
<td>301</td>
</tr>
<tr>
<td>35.2</td>
<td>Variables</td>
<td>301</td>
</tr>
<tr>
<td>35.3</td>
<td>Class Stats</td>
<td>302</td>
</tr>
<tr>
<td>35.3.1</td>
<td>Methods</td>
<td>302</td>
</tr>
<tr>
<td>35.3.2</td>
<td>Properties</td>
<td>302</td>
</tr>
<tr>
<td>35.4</td>
<td>Class Task</td>
<td>303</td>
</tr>
<tr>
<td>35.4.1</td>
<td>Methods</td>
<td>303</td>
</tr>
<tr>
<td>35.4.2</td>
<td>Properties</td>
<td>307</td>
</tr>
<tr>
<td>35.5</td>
<td>Class AlwaysTask</td>
<td>307</td>
</tr>
<tr>
<td>35.5.1</td>
<td>Methods</td>
<td>308</td>
</tr>
</tbody>
</table>
36 Module SCons.Util

36.1 Functions ................................................. 312
36.2 Variables ................................................. 312
36.3 Class NodeList ............................................ 318
  36.3.1 Methods ............................................. 320
  36.3.2 Properties ........................................... 320
  36.3.3 Class Variables ..................................... 321
36.4 Class DisplayEngine .................................... 321
  36.4.1 Methods ............................................. 322
  36.4.2 Properties ........................................... 322
  36.4.3 Class Variables ..................................... 322
36.5 Class Proxy .............................................. 322
  36.5.1 Methods ............................................. 323
  36.5.2 Properties ........................................... 323
36.6 Class Delegate ........................................... 324
  36.6.1 Methods ............................................. 324
  36.6.2 Properties ........................................... 324
36.7 Class _NoError ........................................... 324
  36.7.1 Methods ............................................. 325
  36.7.2 Properties ........................................... 325
36.8 Class WindowsError ..................................... 325
  36.8.1 Methods ............................................. 325
  36.8.2 Properties ........................................... 326
36.9 Class CLVar ............................................... 327
  36.9.1 Methods ............................................. 327
  36.9.2 Properties ........................................... 328
  36.9.3 Class Variables ..................................... 328
36.10 Class OrderedDict ...................................... 328
  36.10.1 Methods ............................................. 329
  36.10.2 Class Variables ................................... 330
36.11 Class Selector .......................................... 330
  36.11.1 Methods ............................................. 330
  36.11.2 Class Variables ................................... 330
36.12 Class LogicalLines .................................... 331
  36.12.1 Methods ............................................. 331
  36.12.2 Properties ........................................... 331
36.13 Class UniqueList ....................................... 332
  36.13.1 Methods ............................................. 332
  36.13.2 Properties ........................................... 335
  36.13.3 Class Variables ................................... 335
36.14 Class Unbuffered ....................................... 335
  36.14.1 Methods ............................................. 336
  36.14.2 Properties ........................................... 336
36.15 Class Null .............................................. 336
36.15.1 Methods .............................................. 336
36.15.2 Properties ........................................... 337
36.16 Class NullSeq ........................................... 338
36.16.1 Methods .............................................. 338
36.16.2 Properties ........................................... 338

37 Package SCons.Variables .................................. 339
37.1 Modules ................................................... 339
37.2 Variables .................................................. 339
37.3 Class Variables .......................................... 339
  37.3.1 Methods .............................................. 340
  37.3.2 Properties ........................................... 342
  37.3.3 Class Variables ..................................... 342

38 Module SCons.Variables.BoolVariable’ .................. 343
  38.1 Functions ............................................... 343

39 Module SCons.Variables.EnumVariable’ .................. 344
  39.1 Functions ............................................... 345

40 Module SCons.Variables.ListVariable’ .................. 346
  40.1 Functions ............................................... 346

41 Module SCons.Variables.PackageVariable’ ............... 347
  41.1 Functions ............................................... 347

42 Module SCons.Variables.PathVariable’ .................. 348
  42.1 Variables ............................................... 349

43 Module SCons.Warnings .................................... 350
  43.1 Functions ............................................... 350
  43.2 Variables ............................................... 351
  43.3 Class Warning .......................................... 351
    43.3.1 Methods .............................................. 352
    43.3.2 Properties ........................................... 352
  43.4 Class WarningOnByDefault ............................... 352
    43.4.1 Methods .............................................. 353
    43.4.2 Properties ........................................... 353
  43.5 Class TargetNotBuiltWarning ........................... 353
    43.5.1 Methods .............................................. 353
    43.5.2 Properties ........................................... 354
  43.6 Class CacheWriteErrorWarning .......................... 354
    43.6.1 Methods .............................................. 354
    43.6.2 Properties ........................................... 355
  43.7 Class CorruptSConsignWarning .......................... 355
    43.7.1 Methods .............................................. 355
    43.7.2 Properties ........................................... 356
  43.8 Class DependencyWarning ............................... 356
    43.8.1 Methods .............................................. 356
    43.8.2 Properties ........................................... 356
  43.9 Class DevelopmentVersionWarning ....................... 357
    43.9.1 Methods .............................................. 357
    43.9.2 Properties ........................................... 357
43.10 Class DuplicateEnvironmentWarning .......................... 358
  43.10.1 Methods ........................................ 358
  43.10.2 Properties ....................................... 358
43.11 Class FutureReservedVariableWarning ....................... 359
  43.11.1 Methods ........................................ 359
  43.11.2 Properties ....................................... 359
43.12 Class LinkWarning ....................................... 360
  43.12.1 Methods ........................................ 360
  43.12.2 Properties ....................................... 360
43.13 Class MisleadingKeywordsWarning .......................... 361
  43.13.1 Methods ........................................ 361
  43.13.2 Properties ....................................... 361
43.14 Class MissingSConscriptWarning ............................ 362
  43.14.1 Methods ........................................ 362
  43.14.2 Properties ....................................... 362
43.15 Class NoMD5ModuleWarning ................................ 363
  43.15.1 Methods ........................................ 363
  43.15.2 Properties ....................................... 363
43.16 Class NoMetaclassSupportWarning ............................ 364
  43.16.1 Methods ........................................ 364
  43.16.2 Properties ....................................... 364
43.17 Class NoObjectCountWarning ................................ 365
  43.17.1 Methods ........................................ 365
  43.17.2 Properties ....................................... 365
43.18 Class NoParallelSupportWarning ............................ 366
  43.18.1 Methods ........................................ 366
  43.18.2 Properties ....................................... 366
43.19 Class ReservedVariableWarning ............................ 367
  43.19.1 Methods ........................................ 367
  43.19.2 Properties ....................................... 367
43.20 Class StackSizeWarning ................................... 368
  43.20.1 Methods ........................................ 368
  43.20.2 Properties ....................................... 368
43.21 Class VisualCMissingWarning ............................... 369
  43.21.1 Methods ........................................ 369
  43.21.2 Properties ....................................... 369
43.22 Class VisualVersionMismatch .............................. 370
  43.22.1 Methods ........................................ 370
  43.22.2 Properties ....................................... 370
43.23 Class VisualStudioMissingWarning ......................... 371
  43.23.1 Methods ........................................ 371
  43.23.2 Properties ....................................... 371
43.24 Class FortranCxxMixWarning .............................. 372
  43.24.1 Methods ........................................ 372
  43.24.2 Properties ....................................... 372
43.25 Class FutureDeprecatedWarning ......................... 373
  43.25.1 Methods ........................................ 373
  43.25.2 Properties ....................................... 373
43.26 Class DeprecatedWarning .................................. 374
  43.26.1 Methods ........................................ 374
  43.26.2 Properties ....................................... 374
## 43 Module SCons

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MandatoryDeprecationWarning</td>
<td>375</td>
</tr>
<tr>
<td>DeprecatedBuildDirWarning</td>
<td>378</td>
</tr>
<tr>
<td>DeprecatedOptionsWarning</td>
<td>381</td>
</tr>
<tr>
<td>DeprecatedSourceCodeWarning</td>
<td>384</td>
</tr>
<tr>
<td>DeprecatedSourceSignaturesWarning</td>
<td>385</td>
</tr>
<tr>
<td>DeprecatedTargetSignaturesWarning</td>
<td>386</td>
</tr>
<tr>
<td>DeprecatedSourceCodeWarning</td>
<td>386</td>
</tr>
<tr>
<td>DeprecatedOptionsWarning</td>
<td>386</td>
</tr>
<tr>
<td>DeprecatedSourceSignaturesWarning</td>
<td>386</td>
</tr>
<tr>
<td>DeprecatedTargetSignaturesWarning</td>
<td>386</td>
</tr>
<tr>
<td>DeprecatedBuildDirWarning</td>
<td>387</td>
</tr>
</tbody>
</table>

### 44 Module SCons.cpp

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions</td>
<td>387</td>
</tr>
<tr>
<td>Variables</td>
<td>387</td>
</tr>
<tr>
<td>Class FunctionEvaluator</td>
<td>388</td>
</tr>
<tr>
<td>Class PreProcessor</td>
<td>388</td>
</tr>
<tr>
<td>Class DumbPreProcessor</td>
<td>393</td>
</tr>
</tbody>
</table>

### 45 Module SCons.dblite

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions</td>
<td>395</td>
</tr>
</tbody>
</table>
45.2 Variables ................................................................. 395
45.3 Class dblite ................................................................. 395
  45.3.1 Methods .............................................................. 395
  45.3.2 Properties ............................................................ 396

46 Module SCons.exitfuncs ........................................ 397
  46.1 Functions ............................................................... 397
  46.2 Variables ............................................................... 397
1 Package SCons

SCons

The main package for the SCons software construction utility. **Version:** 2.4.0

**Date:** 2015/09/21 14:03:43

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:** SCons.Defaults
  *(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. 94)*

- **Memoize:** Memoizer
  *(Section 12, p. 102)*

- **Node:** SCons.Node
  *(Section 13, p. 108)*
  - **Alias:** scons.Node.Alias
    *(Section 14, p. 131)*
  - **FS:** scons.Node.FS
    *(Section 15, p. 137)*
  - **Python:** scons.Node.Python
    *(Section 16, p. 185)*

- **PathList:** SCons.PathList
  *(Section 17, p. 191)*

- **SConf:** SCons.SConf
  *(Section 18, p. 192)*

- **SConsign:** SCons.SConsign
  *(Section 19, p. 208)*

- **Scanner:** SCons.Scanner
  *(Section 20, p. 215)*
  - **C:** SCons.Scanner.C
    *(Section 21, p. 231)*
  - **D:** SCons.Scanner.D
Modules

- Dir: SCons.Scanner.Dir
  (Section 23, p. 238)
- Fortran: SCons.Scanner.Fortran
  (Section 24, p. 240)
- IDL: SCons.Scanner.IDL
  (Section 25, p. 245)
- LaTeX: SCons.Scanner.LaTeX
  (Section 26, p. 246)
- Prog: SCons.Scanner.Prog
  (Section 27, p. 253)
- RC: SCons.Scanner.RC
  (Section 28, p. 254)
- Script: SCons.Script
  (Section 29, p. 255)
  - Interactive: SCons interactive mode
    (Section 30, p. 264)
  - Main: SCons.Script
    (Section 31, p. 267)
  - SConsscript': SCons.Script.SConsscript
    (Section 32, p. 281)
- Sig: Place-holder for the old SCons.Sig module hierarchy
  (Section 33, p. 287)
- Subst: SCons.Subst
  (Section 34, p. 290)
- Taskmaster: Generic Taskmaster module for the SCons build engine.
  (Section 35, p. 301)
- Util: SCons.Util
  (Section 36, p. 312)
- Variables: engine.SCons.Variables
  (Section 37, p. 339)
  - BoolVariable: engine.SCons.Variables.BoolVariable
    (Section 38, p. 343)
  - EnumVariable: engine.SCons.Variables.EnumVariable
    (Section 39, p. 344)
  - ListVariable: engine.SCons.Variables.ListVariable
    (Section 40, p. 346)
  - PackageVariable: engine.SCons.Variables.PackageVariable
    (Section 41, p. 347)
  - PathVariable: engine.SCons.Variables.PathVariable
    (Section 42, p. 348)
- Warnings: SCons.Warnings
  (Section 43, p. 350)
- cpp: SCons C Pre-Processor module
  (Section 44, p. 387)
- dblite (Section 45, p. 395)
- exitfuncs: SCons.exitfuncs
  (Section 46, p. 397)
### 1.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>build</strong></td>
<td>Value: <code>rel_2.4.0:3365:9259ea1c13d7</code></td>
</tr>
<tr>
<td><strong>builddsys</strong></td>
<td>Value: <code>hpmicrodog</code></td>
</tr>
<tr>
<td><strong>developer</strong></td>
<td>Value: <code>bdbaddog</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</code></td>
</tr>
<tr>
<td></td>
<td><code>rel_2.4.0:3365:9259ea1c13d7...</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 checked 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

\[ \text{rfile}(n) \]

\[ \text{default\_exitstatfunc}(s) \]

\[ \text{remove\_set\_lineno\_codes}(x) \]

\[ \text{Action}(act,*args,**kw) \]

A factory for action objects.

\[ \text{get\_default\_ENV}(env) \]

2.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_revision_</td>
<td>Value: 'src/engine/SCons/Action.py rel_2.4.0:3365:9259ea1c13d7 2...</td>
</tr>
<tr>
<td>print_actions</td>
<td>Value: 1</td>
</tr>
<tr>
<td>execute_actions</td>
<td>Value: 1</td>
</tr>
<tr>
<td>print_actions_presub</td>
<td>Value: 0</td>
</tr>
<tr>
<td>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;\n]([^&quot;\n]</td>
</tr>
<tr>
<td>default_ENV</td>
<td>Value: None</td>
</tr>
<tr>
<td>_package_</td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

2.3 Class ActionBase

object SCons.Action.ActionBase


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

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

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

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

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

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

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

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

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

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

```python
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__init__</code></td>
<td>(<code>self</code>, <code>generator</code>, <code>kw</code>)</td>
<td><strong><strong>init</strong></strong>(...) initializes x; see help(type(x)) for signature. Overrides: <code>object.__init__</code> (inherited documentation)</td>
</tr>
<tr>
<td><code>__str__</code></td>
<td>(<code>self</code>)</td>
<td><strong>str</strong>(x) Overrides: <code>object.__str__</code> (inherited documentation)</td>
</tr>
<tr>
<td><code>get_presig</code></td>
<td>(<code>self</code>, <code>target</code>, <code>source</code>, <code>env</code>, <code>executor=</code>None)</td>
<td>Returns the signature contents of this action’s command line. This strips $(-$) and everything in between the string, since those parts don’t affect signatures.</td>
</tr>
<tr>
<td><code>get_implicit_deps</code></td>
<td>(<code>self</code>, <code>target</code>, <code>source</code>, <code>env</code>, <code>executor=</code>None)</td>
<td></td>
</tr>
<tr>
<td><code>get_targets</code></td>
<td>(<code>self</code>, <code>env</code>, <code>executor</code>)</td>
<td>Returns the type of targets ($TARGETS, $CHANGED_TARGETS) used by this action. Overrides: <code>SCons.Action.ActionBase.get_targets</code> (inherited documentation)</td>
</tr>
</tbody>
</table>
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>
<tbody>
<tr>
<td><code>Inherited from object</code></td>
<td></td>
</tr>
<tr>
<td><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>

#### 2.6 Class LazyAction

**Class Hierarchy**

```
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__` (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
exit(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(), presub_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>
2.7 Class FunctionAction

object
SCons.Action.ActionBase
SCons.Action._ActionAction
SCons.Action.FunctionAction

Class for Python function actions.

2.7.1 Methods

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

function_name(self)

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

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

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

get_presig(self, target, source, env)

Return the signature contents of this callable action.

get_implicit_deps(self, target, source, env)

Inherited from SCons.Action._ActionAction

__call__(), print_cmd_line()

Inherited from SCons.Action.ActionBase(Section 2.3)

__add__(), __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__() 

2.7.2 Properties

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

2.8 Class ListAction

object  
SCons.Action.ActionBase  
\[\text{SCons.Action.ListAction}\]

Class for lists of other actions.

2.8.1 Methods

    __init__(self, actionlist)

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

    genstring(self, target, source, env)


    __str__(self)

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

    presub_lines(self, env)

Overrides: SCons.Action.ActionBase.presub_lines
Class ActionCaller

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><strong>init</strong>(self, parent, args, kw)</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>get_contents(self, target, source, env)</td>
<td></td>
</tr>
<tr>
<td>subst(self, s, target, source, env)</td>
<td></td>
</tr>
<tr>
<td>subst__args(self, target, source, env)</td>
<td></td>
</tr>
<tr>
<td>subst__kw(self, target, source, env)</td>
<td></td>
</tr>
<tr>
<td><strong>call</strong>(self, target, source, env, executor=None)</td>
<td></td>
</tr>
<tr>
<td>strfunction(self, target, source, env)</td>
<td></td>
</tr>
<tr>
<td><strong>str</strong>(self)</td>
<td>str(x) Overrides: object.<strong>str</strong> (inherited documentation)</td>
</tr>
</tbody>
</table>

**Inherited from object**

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

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

31
2.10 Class ActionFactory

object

SCons.Action.ActionFactory

A factory class that will wrap up an arbitrary function as an SCons-executable Action object.

The real heavy lifting here is done by the ActionCaller class. We just collect the (positional and keyword) arguments that we’re called with and give them to the ActionCaller object we create, so it can hang onto them until it needs them.

2.10.1 Methods

```python
__init__(self, actfunc, strfunc, convert=<function <lambda> at 0x7f3ab66dded8>)
```

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

---

32
3 Module SCons.Builder

SCons.Builder

Builder object subsystem.

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

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

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

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

There is also a proxy that looks like a Builder:

CompositeBuilder

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

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

__call__()

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

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

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

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

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

get_name()
Returns the Builder’s name within a specific Environment, primarily used to try to return helpful information in error messages.

adjust_suffix()
get_prefix()
get_suffix()
get_src_suffix()set_src_suffix()
Miscellaneous stuff for handling the prefix and suffix manipulation we use in turning source file names into target file names.

3.1 Functions

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

A factory for builder objects.
is_a_Builder(obj)

'Returns True if the specified obj is one of our Builder classes.
The test is complicated a bit by the fact that CompositeBuilder is a proxy, not a subclass of BuilderBase.

### 3.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Builder.py rel_2.4.0:3365:9259ea1c13d7 ...</td>
</tr>
<tr>
<td>misleading_keywords</td>
<td>Value: {'sources': 'source', 'targets': 'target'}</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

### 3.3 Class DictCmdGenerator

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

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

#### 3.3.1 Methods

```python
__init__(self, dict=None, source_ext_match=1)
```
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><strong>hash</strong></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)
Class DictEmitter Module SCons.Builder

_initsitem_(), __init__(), _setitem_(), clear(), copy(), items(), keys(),
popitem(), setdefault(), update(), values()

Inherited from UserDict.UserDict

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

3.4.2 Class Variables

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

3.5 Class DictEmitter

UserDict.UserDict  
SCons.Util.OrderedDict  
    SCons.Util.Selector  
        SCons.Builder.DictEmitter

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

3.5.1 Methods

__call__(self, target, source, env)

Overrides: SCons.Util.Selector.__call__

Inherited from SCons.Util.OrderedDict(Section 36.10)

_delitem_(), __init__(), _setitem_(), clear(), copy(), items(), keys(),
popitem(), setdefault(), update(), values()

Inherited from UserDict.UserDict

__cmp__(), __contains__(), __getitem__(), __len__(), __repr__(), fromkeys(),
get(), has_key(), iteritems(), iterkeys(), itervalues(), pop()
3.5.2 Class Variables

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

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

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 __abcoll.Sequence
Inherited from __abstract.Sized

Inherited from object

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

3.6.2 Properties

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

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></td>
</tr>
</tbody>
</table>

3.7 Class OverrideWarner

UserDict.UserDict →

SCons.Builder.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

__init__(self, dict)

Overrides: UserDict.UserDict.__init__

warn(self)

Inherited from UserDict.UserDict
3.7.2 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>

3.8 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__` (inherited documentation)

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

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

Inherited from object

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

3.8.2 Properties
### Class BuilderBase

Object structure:

- Inherited from `object`
- `__class__`

**Description**

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

#### 3.9.1 Methods

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

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

  **Overrides:**
  
  - object.**init** inherited documentation

- **`__nonzero__`**
  ```python
  __nonzero__(self)
  ```

- **`get_name`**
  ```python
  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__`**
  ```python
  __cmp__(self, other)
  ```

- **`splitext`**
  ```python
  splitext(self, path, env=None)
  ```
Class BuilderBase

Module SCons.Builder

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

```python
adjust_suffix(self, suff)
```

```python
get_prefix(self, env, sources=[])  # Get the first src_suffix in the list of src_suffixes.
```

```python
set_suffix(self, suffix)
```

```python
set_src_suffix(self, src_suffix)
```

```python
get_src_suffix(self, env)
```

```python
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.

```python
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.

```python
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_builder 'tree.' (This value isn’t cached because there may be changes in a src_builder many levels deep that we can’t see.)

Inherited from object

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

3.9.2 Properties

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

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

3.10.1 Methods

__init__(self, builder, cmdgen)

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

__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__’)

add_action(self, suffix, action)

Inherited from SCons.Util.Proxy(Section 36.5)

__cmp__(), __getattr__(), get()

Inherited from object

__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>
</table>

Inherited from object

__class__
4 Module SCons.CacheDir

CacheDir support

4.1 Functions

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

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

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

4.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/CacheDir.py rel_2.4.0:3365:9259ea1c13d7...</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><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

4.3 Class CacheDir

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

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

This method initializes the CacheDir object. It is called when creating a new instance of the CacheDir class.

Overrides:

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

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

This method is used for debugging purposes and prints information to the console.

```python
is_enabled(self)
```

This method returns a boolean indicating whether the CacheDir is enabled.

```python
is_readonly(self)
```

This method returns a boolean indicating whether the CacheDir is read-only.

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

This method is called to get the path of the cached file.

```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__(),
```

46
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

\[ \texttt{CheckBuilder} (\texttt{context}, \texttt{text=\textit{None}}, \texttt{language=\textit{None}}) \]

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

\[ \texttt{CheckCC} (\texttt{context}) \]

Configure check for a working C compiler.

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

\[ \texttt{CheckSHCC} (\texttt{context}) \]

Configure check for a working shared C compiler.

This checks whether the C compiler, as defined in the $SHCC$ construction variable, can compile a C source file. It uses the current $SHCCCOM$ value too, so that it can test against non working flags.
CheckCXX\((\text{context})\)

Configure check for a working CXX compiler.

This checks whether the CXX compiler, as defined in the $CXX\text{\_construction\_variable}$, can compile a CXX source file. It uses the current $CXX\text{\_COM} variable too, so that it can test against non working flags.

CheckSHCXX\((\text{context})\)

Configure check for a working shared CXX compiler.

This checks whether the CXX compiler, as defined in the $\text{SHCXX\_construction\_variable}$, can compile a CXX source file. It uses the current $\text{SHCXX\_COM}$ value too, so that it can test against non working flags.

CheckFunc\((\text{context}, \text{function\_name}, \text{header}=\text{None}, \text{language}=\text{None})\)

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\((\text{context}, \text{header\_name}, \text{header}=\text{None}, \text{language}=\text{None}, \text{include\_quotes}=\text{None})\)

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 whether the type has the given number of bytes. If not given, will automatically find the size.

**Returns:**

- **status** (int)
  0 if the check failed, or the found size of the type if the check succeeded.
CheckDeclaration(context, symbol, includes=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(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 $LFLAGS 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>
Module SCons.Debug

SCons.Debug

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

6.1 Functions

- `logInstanceCreation(instance, name=None)`
- `string_to_classes(s)`
- `fetchLoggedInstances(classes='*')`
- `countLoggedInstances(classes, file=sys.stderr)`
- `listLoggedInstances(classes, file=sys.stderr)`
- `dumpLoggedInstances(classes, file=sys.stderr)`
- `memory()`
- `caller_stack()`
- `caller_trace(back=0)`
- `dump_caller_counts(file=sys.stderr)`
- `func_shorten(func_tuple)`
- `Trace(msg, file=None, mode='w', tstamp=None)`

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

6.2 Variables
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: <code>src/engine/SCons/Debug.py rel_2.4.0:3365:9259ea1c13d7 20...</code></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: <code>[(‘/scons/SCons/’, 1), (‘/src/engine/SCons/’, 1), (‘/usr/...</code></td>
</tr>
</tbody>
</table>
7 Module SCons.Defaults

SConsDefaults

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultEnvironment(*args, **kw)</td>
<td>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/<strong>init</strong>.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.)</td>
</tr>
<tr>
<td>StaticObjectEmitter(target, source, env)</td>
<td></td>
</tr>
<tr>
<td>SharedObjectEmitter(target, source, env)</td>
<td></td>
</tr>
<tr>
<td>SharedFlagChecker(source, target, env)</td>
<td></td>
</tr>
<tr>
<td>get_paths_str(dest)</td>
<td></td>
</tr>
<tr>
<td>chmod_func(dest, mode)</td>
<td></td>
</tr>
<tr>
<td>chmod_strfunc(dest, mode)</td>
<td></td>
</tr>
</tbody>
</table>
Variables

**Module SCons.Defaults**

```python
# copy_func

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

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

```python
# delete_func

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

```python
# delete_strfunc

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

```python
# mkdir_func

**mkdir_func**(*dest*)
```

```python
# move_func

**move_func**(*dest, src*)
```

```python
# touch_func

**touch_func**(*dest*)
```

```python
# processDefines

**processDefines**(*defs*)
```

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

### 7.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Defaults.py rel_2.4.0:3365:9259ea1c13d7...</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>
</tbody>
</table>
```

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>DAction</td>
<td>Value: SCons.Action.Action(&quot;$DCOM&quot;, &quot;$DCOMSTR&quot;)</td>
</tr>
<tr>
<td>ShDAction</td>
<td>Value: SCons.Action.Action(&quot;$SHDCOM&quot;, &quot;$SHDCOMSTR&quot;)</td>
</tr>
<tr>
<td>ASAction</td>
<td>Value: SCons.Action.Action(&quot;$ASCOM&quot;, &quot;$ASCOMSTR&quot;)</td>
</tr>
<tr>
<td>ShLinkAction</td>
<td>Value: SCons.Action.Action(&quot;$SHLINKCOM&quot;, &quot;$SHLINKCOMSTR&quot;)</td>
</tr>
<tr>
<td>Chmod</td>
<td>Value: SCons.Defaults.Chmod</td>
</tr>
<tr>
<td>Copy</td>
<td>Value: SCons.Defaults.Copy</td>
</tr>
<tr>
<td>Delete</td>
<td>Value: SCons.Defaults.Delete</td>
</tr>
<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>ConstructionEnvironment</td>
<td>Value: {'BUILDERS': {}, 'CONFIGUREDIR': '#/.sconf_temp', 'CONFIG...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

### 7.3 Class NullCmdGenerator

```
object
  SCons.Defaults.NullCmdGenerator
```

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

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

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

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

Overrides:
```python
object.__init__ (inherited documentation)
```

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

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

7.3.2 Properties

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

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

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

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

*Inherited from object*

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

7.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Inherited from object</code></td>
<td></td>
</tr>
<tr>
<td><code>__class__</code></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(\textit{env, target, source})</td>
<td></td>
</tr>
<tr>
<td>apply_tools(\textit{env, tools, toolpath})</td>
<td></td>
</tr>
<tr>
<td>copy_non_reserved_keywords(\textit{dict})</td>
<td></td>
</tr>
<tr>
<td>is_valid_construction_var(\textit{varstr})</td>
<td>Return if the specified string is a legitimate construction variable.</td>
</tr>
<tr>
<td>default_decide_source(\textit{dependency, target, prev Ni})</td>
<td></td>
</tr>
<tr>
<td>default_decide_target(\textit{dependency, target, prev Ni})</td>
<td></td>
</tr>
<tr>
<td>default_copy_from_cache(\textit{src, dst})</td>
<td></td>
</tr>
<tr>
<td>NoSubstitutionProxy(\textit{subject})</td>
<td></td>
</tr>
</tbody>
</table>

8.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>___revision__</td>
<td>Value: `src/engine/SCons/Environment.py rel_2.4.0:3365:9259ea1c1...</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>

continued on next page
8.3 Class MethodWrapper

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

___init___(self, object, method, name=None)

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

___call___(self, *args, **kwargs)

cloned(self, new_object)

Returns an object that re-binds the underlying 'method' to the specified new object.

Inherited from 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)
Class BuilderDict

Module SCons.Environment

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

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

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

**Inherited from SCons.Environment.MethodWrapper (Section 8.3)**

__init__(), clone()

**Inherited from object**

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

### 8.4.2 Properties

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

### 8.5 Class BuilderDict


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

#### 8.5.1 Methods

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

```python
__semi_deepcopy__(self)
```
8.6 Class SubstitutionEnvironment

object — SCons.Environment.SubstitutionEnvironment

**Known Subclasses:** SCons.Environment.Base

Base class for different flavors of construction environments.

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

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

8.6.1 Methods

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

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

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

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

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

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

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

Emulates the `get()` method of dictionaries.

```python
has_key(self, key)
```

```python
__contains__(self, key)
```

```python
items(self)
```

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

```python
gvars(self)
```

```python
lvars(self)
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subst</td>
<td><code>subst(self, string, raw=0, target=None, source=None, conv=None, executor=None)</code></td>
<td>Recursively interpolates construction variables from the Environment into the specified string, returning the expanded result. Construction variables are specified by a $ prefix in the string and begin with an initial underscore or alphabetic character followed by any number of underscores or alphanumeric characters. The construction variable names may be surrounded by curly braces to separate the name from trailing characters.</td>
</tr>
<tr>
<td>subst_kw</td>
<td><code>subst_kw(self, kw, raw=0, target=None, source=None)</code></td>
<td></td>
</tr>
<tr>
<td>subst_list</td>
<td><code>subst_list(self, string, raw=0, target=None, source=None, conv=None, executor=None)</code></td>
<td>Calls through to SCons.Subst.scons_subst_list(). See the documentation for that function.</td>
</tr>
<tr>
<td>subst_path</td>
<td><code>subst_path(self, path, target=None, source=None)</code></td>
<td>Substitute a path list, turning EntryProxies into Nodes and leaving Nodes (and other objects) as-is.</td>
</tr>
<tr>
<td>subst_target_source</td>
<td><code>subst_target_source(self, string, raw=0, target=None, source=None, conv=None, executor=None)</code></td>
<td>Recursively interpolates construction variables from the Environment into the specified string, returning the expanded result. Construction variables are specified by a $ prefix in the string and begin with an initial underscore or alphabetic character followed by any number of underscores or alphanumeric characters. The construction variable names may be surrounded by curly braces to separate the name from trailing characters.</td>
</tr>
<tr>
<td>backtick</td>
<td><code>backtick(self, command)</code></td>
<td></td>
</tr>
</tbody>
</table>
**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*
8.6.2 Properties

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

8.7 Class Base

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**(*self*, *targets*)
### Append

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

Append values to existing construction variables in an Environment.

### AppendENVPath

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

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

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

### AppendUnique

```python
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

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

### Builder

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

### CacheDir

```python
CacheDir(self, path)
```

### Clean

```python
Clean(self, targets, files)
```

### Clone

```python
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.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong></td>
<td>(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.</td>
</tr>
<tr>
<td><strong>Configure</strong></td>
<td>(self, *args, **kw)</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>(self, *args, **kw)</td>
</tr>
<tr>
<td><strong>Decider</strong></td>
<td>(self, function)</td>
</tr>
<tr>
<td><strong>Depends</strong></td>
<td>(self, target, dependency)</td>
</tr>
<tr>
<td><strong>Detect</strong></td>
<td>(self, progs)</td>
</tr>
<tr>
<td><strong>Dictionary</strong></td>
<td>(self, *args)</td>
</tr>
<tr>
<td><strong>Dir</strong></td>
<td>(self, name, *args, **kw)</td>
</tr>
<tr>
<td><strong>Dump</strong></td>
<td>(self, key=None)</td>
</tr>
<tr>
<td><strong>Entry</strong></td>
<td>(self, name, *args, **kw)</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td>(self, **kw)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Execute</strong>(<em>self</em>, <em>action</em>, <em>args</em>, <strong>kw</strong>)</td>
<td>Directly execute an action through an Environment</td>
</tr>
<tr>
<td><strong>File</strong>(<em>self</em>, <em>name</em>, <em>args</em>, <strong>kw</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>FindFile</strong>(<em>self</em>, <em>file</em>, <em>dirs</em>)</td>
<td></td>
</tr>
<tr>
<td><strong>FindInstalledFiles</strong>(<em>self</em>)</td>
<td>returns the list of all targets of the Install and InstallAs Builder.</td>
</tr>
<tr>
<td><strong>FindIxes</strong>(<em>self</em>, <em>paths</em>, <em>prefix</em>, <em>suffix</em>)</td>
<td>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.</td>
</tr>
<tr>
<td><strong>FindSourceFiles</strong>(<em>self</em>, <em>node</em>='.'*)</td>
<td>returns a list of all source files.</td>
</tr>
<tr>
<td><strong>Flatten</strong>(<em>self</em>, <em>sequence</em>)</td>
<td></td>
</tr>
<tr>
<td><strong>GetBuildPath</strong>(<em>self</em>, <em>files</em>)</td>
<td></td>
</tr>
<tr>
<td><strong>Glob</strong>(<em>self</em>, <em>pattern</em>, <em>ondisk=True</em>, <em>source=False</em>, <em>strings=False</em>, <em>exclude=None</em>)</td>
<td></td>
</tr>
<tr>
<td><strong>Ignore</strong>(<em>self</em>, <em>target</em>, <em>dependency</em>)</td>
<td>Ignore a dependency.</td>
</tr>
<tr>
<td><strong>Literal</strong>(<em>self</em>, <em>string</em>)</td>
<td></td>
</tr>
</tbody>
</table>
Local(self, *targets)

NoCache(self, *targets)

Tags a target so that it will not be cached

NoClean(self, *targets)

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

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

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

Platform(self, platform)

Precious(self, *targets)

Prepend(self, **kw)

Prepend values to existing construction variables in an Environment.
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).

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.

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

Replace old_prefix with new_prefix and old_suffix with new_suffix.

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

```python
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

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

### Scanner

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

### SetDefault

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

### SideEffect

```python
SideEffect(self, side_effect, target)
```

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

### SourceCode

```python
SourceCode(self, entry, builder)
```

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

### SourceSignatures

```python
SourceSignatures(self, type)
```

### Split

```python
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

```python
TargetSignatures(self, type)
```

### Tool

```python
Tool(self, tool, toolpath=None, **kw)
```

74
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td><code>Value(self, value, built_value=None)</code></td>
<td>Sets the value for a module.</td>
</tr>
<tr>
<td>VariantDir</td>
<td><code>VariantDir(self, variant_dir, src_dir, duplicate=1)</code></td>
<td>Creates a directory with a variant and source directory.</td>
</tr>
<tr>
<td>WhereIs</td>
<td><code>WhereIs(self, prog, path=None, pathext=None, reject=[])</code></td>
<td>Finds the program specified by <code>prog</code> in the given path or path extension.</td>
</tr>
<tr>
<td><strong>init</strong></td>
<td><code>__init__(self, platform=None, tools=None, toolpath=None, variables=None, parse_flags=None, **kw)</code></td>
<td>Initializes a basic SCons construction environment, setting up special variables and searching for tools.</td>
</tr>
<tr>
<td>get_CacheDir</td>
<td><code>get_CacheDir(self)</code></td>
<td>Fetches the cache directory.</td>
</tr>
<tr>
<td>get_builder</td>
<td><code>get_builder(self, name)</code></td>
<td>Fetches the builder with the specified name from the environment.</td>
</tr>
<tr>
<td>get_factory</td>
<td><code>get_factory(self, factory, default='File')</code></td>
<td>Returns a factory function for creating Nodes for the construction environment.</td>
</tr>
<tr>
<td>get_scanner</td>
<td><code>get_scanner(self, skey)</code></td>
<td>Finds the appropriate scanner given a key (usually a file suffix).</td>
</tr>
<tr>
<td>get_src_sig_type</td>
<td><code>get_src_sig_type(self)</code></td>
<td>Fetches the signature type for source files.</td>
</tr>
</tbody>
</table>
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>

### 8.8 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

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

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

Note that we do *not* call the underlying base class (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:

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

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

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

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

Overrides: SCons.Environment.SubstitutionEnvironment.__getitem__

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

Overrides: SCons.Environment.SubstitutionEnvironment.__setitem__

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

Overrides: SCons.Environment.SubstitutionEnvironment.__delitem__

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

Emulates the get() method of dictionaries. Overrides:

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

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

Dictionary(self)


items(self)


gvars(self)


lvars(self)


Replace(self, **kw)

Replace existing construction variables in an Environment with new
collection 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(),

78
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><strong>class</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

Action

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

AddPostAction

```
AddPostAction(self, files, action)
```

AddPreAction

```
AddPreAction(self, files, action)
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alias</strong></td>
<td>( \text{Alias}(\text{self, target, source=}, \text{action=None, **kw}) )</td>
</tr>
<tr>
<td><strong>AlwaysBuild</strong></td>
<td>( \text{AlwaysBuild}(\text{self, *targets}) )</td>
</tr>
<tr>
<td><strong>Append</strong></td>
<td>( \text{Append}(\text{self, **kw}) )</td>
</tr>
<tr>
<td></td>
<td>Append values to existing construction variables in an Environment.</td>
</tr>
<tr>
<td><strong>AppendENVPath</strong></td>
<td>( \text{AppendENVPath}(\text{self, name, newpath, envname='ENV', sep=':', delete_existing=1}) )</td>
</tr>
<tr>
<td></td>
<td>Append path elements to the path <code>name</code> in the <code>ENV</code> 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).</td>
</tr>
<tr>
<td><strong>AppendUnique</strong></td>
<td>( \text{AppendUnique}(\text{self, delete_existing=0, **kw}) )</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td><strong>BuildDir</strong></td>
<td>( \text{BuildDir}(\text{self, *args, **kw}) )</td>
</tr>
<tr>
<td><strong>Builder</strong></td>
<td>( \text{Builder}(\text{self, **kw}) )</td>
</tr>
<tr>
<td><strong>CacheDir</strong></td>
<td>( \text{CacheDir}(\text{self, path}) )</td>
</tr>
<tr>
<td><strong>Clean</strong></td>
<td>( \text{Clean}(\text{self, targets, files}) )</td>
</tr>
</tbody>
</table>
**Clone** \((self, \text{ tools}=[], \text{ toolpath}=\text{None}, \text{ parse\_flags}=\text{None}, **\text{kw})\)

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.

**Command** \((self, \text{ target}, \text{ source}, \text{ action}, **\text{kw})\)

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** \((self, *\text{args}, **\text{kw})\)

**Copy** \((self, *\text{args}, **\text{kw})\)

**Decider** \((self, \text{ function})\)

**Depends** \((self, \text{ target}, \text{ dependency})\)

Explicitly specify that 'target's depend on 'dependency'.

**Detect** \((self, \text{ progs})\)

Return the first available program in progs.

**Dictionary** \((self, *\text{args})\)

**Dir** \((self, \text{ name}, *\text{args}, **\text{kw})\)
Dump(self, key=None)

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

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

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

Environment(self, **kw)

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

Directly execute an action through an Environment

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

FindFile(self, file, dirs)

FindInstalledFiles(self)

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

FindIxes(self, paths, prefix, suffix)

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

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

FindSourceFiles(self, node=’.’)

returns a list of all source files.

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

Prepend values to existing construction variables in an Environment.

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).

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.
Replace(self, **kw)

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

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

Replace old_prefix with new_prefix and old_suffix with new_suffix.

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

Repository(self, *dirs, **kw)

Requires(self, target, prerequisite)

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

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

Scanner(self, *args, **kw)

SetDefault(self, **kw)

SideEffect(self, side_effect, target)

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

SourceCode(self, entry, builder)

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

SourceSignatures(self, type)
**Split**(self, ary)

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 (SCons 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)

86
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.SubstitionEnvironment(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__()

8.9.2 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>

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

Class Base

Module SCons.Environment
9 Module SCons.Errors

SCons.Errors

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

9.1 Functions

\texttt{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>_revision_</td>
<td>Value: 'src/engine/SCons/Errors.py rel_2.4.0:3365:9259ea1c13d7 2...'</td>
</tr>
<tr>
<td>_package_</td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

9.3 Class BuildError

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.BuildError

Errors occurring while building.

BuildError have the following attributes:

Information about the cause of the build error:

\texttt{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))
```

This initializes the `InternalError` class with the given parameters. It overrides the `__init__` method of the base class `Exception`.

```python
__str__(self)
```

This method overrides the `__str__` method of the base class `Exception` to return a string representation of the `InternalError`.

Inherited from `exceptions.Exception`

- `__new__()` method.

Inherited from `exceptions.BaseException`

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

Inherited from `object`

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

9.3.2 Properties

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

9.4 Class InternalError

```
object

exceptions.BaseException

exceptions.Exception

SCons.Errors.InternalError
```
9.4.1 Methods

*Inherited from exceptions.Exception*

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

*Inherited from exceptions.BaseException*

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

*Inherited from object*

\[\text{\_format\_()}, \text{\_hash\_()}, \text{\_reduce\_ex\_()}, \text{\_sizeof\_()}, \text{\_subclassook\_()}\]

9.4.2 Properties

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

9.5 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*

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

*Inherited from exceptions.BaseException*
Inherited from object

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

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

args, message

Inherited from object
__class__

9.7.1 Methods

Inherited from exceptions.Exception

_inherit_(), __new__()  

Inherited from exceptions.BaseException

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

Inherited from object

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

9.7.2 Properties

Inherited from exceptions.BaseException

args, message

Inherited from object
__class__

<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>
9.8 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>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
9.9 Class ExplicitExit

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.ExplicitExit

9.9.1 Methods

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

Overrides:

```python
object.__init__
```

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>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
10 Module SCons.Executor

SCons.Executor

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

10.1 Functions

**rfile(node)**

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.).

**execute_nothing(obj, target, kw)**

**execute_action_list(obj, target, kw)**

Actually execute the action list.

**execute_actions_str(obj)**

**execute_null_str(obj)**

**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>
</table>
| __revision__  | Value: `src/engine/SCons/Executor.py
rel_2.4.0:3365:9259ea1c13d7...` |

continued on next page
**10.3 Class Batch**

```python
targets =SCons.Executor.Batch
```

Remembers exact association between targets and sources of executor.

**10.3.1 Methods**

```python
__init__(self, targets=[], sources=[])  
x.__init__(...) initializes x; see help(type(x)) for signature  
Overrides: object.__init__ extit(inherited documentation)
```

*Inherited from object*

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

**10.3.2 Properties**

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

*Inherited from object*

```python
__class__
```
10.4 Class TSList

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

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

10.4.1 Methods

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

__getattr__(self, attr)

__getitem__(self, i)
Overrides: _abcoll.Sequence.__getitem__
```
Class TSLList Module SCons.Executor

```python
__getslice__(self, i, j)
Overrides: UserList.UserList.__getslice__
```

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

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

Inherited from UserList.UserList

```python
__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 _collections.Sequence

```python
__iter__(), __reversed__()
```

Inherited from _collections.Sized

```python
__subclasshook__()
```

Inherited from object

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

10.4.2 Properties

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

10.4.3 Class Variables

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

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

10.5.1 Methods

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

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

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

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

Inherited from object

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

10.5.2 Properties

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

10.6 Class Executor

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

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

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

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

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

Inherited from object

```python
__delattr__, __format__, __getattribute__, __hash__, __new__,
__reduce__, __reduce_ex__, __setattr__, __sizeof__, __subclasshook__
```
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={})
```

overrides: object.__init__ (inherited documentation)

```
get_lvars(self)
```

```
get_action_targets(self)
```

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

```
get_action_list(self)
```

```
get_all_targets(self)
```

Returns all targets for all batches of this Executor.

```
get_all_sources(self)
```

Returns all sources for all batches of this Executor.

```
get_all_children(self)
```

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

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

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

get_action_side_effects(self)

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

get_build_env(self)

Fetch or create the appropriate build Environment for this Executor.

get_build_scanner_path(self, scanner)

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

get_kW(self, kw={})

call(self, target, **kw)

cleanup(self)

add_sources(self, sources)

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

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

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

prepare(self)

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

add_pre_action(self, action)

add_post_action(self, action)

__str__(self)

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

nullify(self)

get_contents(self)

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

get_timestamp(self)

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 \( N \times M \) scan of the sources for each individual target, which is a hell of a lot more efficient.

get__unignored__sources(self, node, ignore=())

get__implicit__deps(self)

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

Inherited from object

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

10.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action_list</td>
<td></td>
</tr>
<tr>
<td>batches</td>
<td></td>
</tr>
<tr>
<td>builder_kw</td>
<td></td>
</tr>
<tr>
<td>env</td>
<td></td>
</tr>
<tr>
<td>lvars</td>
<td></td>
</tr>
<tr>
<td>override_list</td>
<td></td>
</tr>
<tr>
<td>post_actions</td>
<td></td>
</tr>
<tr>
<td>pre_actions</td>
<td></td>
</tr>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
</tbody>
</table>

___class___
10.7  Class NullEnvironment

object   
SCons.Util.Null   
SCons.Executor.NullEnvironment

10.7.1  Methods

get_CacheDir(self)

Inherited from SCons.Util.Null(Section 36.15)

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

Inherited from object

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

10.7.2  Properties

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

Inherited from object

__class__

10.8  Class Null

object   
SCons.Executor.Null

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

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

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

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

Overrides:

```python
object.__init__
```

(inherited documentation)

```python
get_build_env(self)
```

```python
get_build_scanner_path(self)
```

```python
cleanup(self)
```

```python
prepare(self)
```

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

```python
get_action_targets(self)
```

```python
get_action_list(self)
```

```python
get_all_targets(self)
```

```python
get_all_sources(self)
```

```python
get_all_children(self)
```

```python
get_all_prerequisites(self)
```

```python
get_action_side_effects(self)
```

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

```python
get_contents(self)
```

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

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

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

**Inherited from object**

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

### 10.8.2 Properties

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

*Inherited from object*  
```python
___class___
```
11 Module SCons.Job

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>
</table>
| __revision__      | Value: 'src/engine/SCons/Job.py
rel_2.4.0:3365:9259ea1c13d7 2015... |
| explicit_stack_size | Value: None                                   |
| default_stack_size   | Value: 256                                    |
| interrupt_msg      | Value: 'Build interrupted.'                   |
| __package__        | Value: 'SCons'                                 |

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__ extit(inherited documentation)

```python
set(self)
```

```python
__call__(self)
```

Inherited from object

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

11.3 Class Jobs

object

\[ \text{SCons.Job.Jobs} \]

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

11.3.1 Methods

\[ \_\_\_i\_n\_i\_t\_\_\_ (s\_e\_l\_f, n\_u\_m, t\_a\_s\_k\_m\_a\_s\_t) \]

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__

\[ \_\_\_r\_u\_n\_\_\_ (s\_e\_l\_f, p\_o\_s\_t\_f\_u\_n\_c<\_f\_u\_n\_c<\_l\_a\_m\_b\_d\_a> \ a\_t \ 0x7f3ab63fe1b8>) \]

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

### 11.5 Class Worker

```
object

threading._Verbose

threading.Thread

SCons.Job.Worker
```

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

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

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

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

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

```python
run(self)
```

Method representing the thread’s activity.

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

**Inherited from threading.Thread**

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

**Inherited from object**

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

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

### 11.6 Class ThreadPool

```python
doctree
object

SCons.Job.ThreadPool
```

This class is responsible for spawning and managing worker threads.

#### 11.6.1 Methods

```python
__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__

```python
put(self, task)
```

Put task into request queue.

```python
get(self)
```

Remove and return a result tuple from the results queue.

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

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

*Inherited from object*

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

### 11.6.2 Properties

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

### 11.7 Class Parallel

**object**

**SCons.Job.Parallel**

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

This class is thread safe.
11.7.1 Methods

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

Create a new parallel job given a taskmaster.

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

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

```
start(self)
```

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

Inherited from `object`

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

11.7.2 Properties

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

Memoizer

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

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

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

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

    result = self.compute_foo_value()

    self._memo['foo'] = result # Memoization

    return result
```

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

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

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

result = self.compute_bar_value(argument) # Memoization

memo_dict[memo_key] = result # Memoization

return result # Memoization

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

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

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

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

12.1 Functions

<table>
<thead>
<tr>
<th>Dump(\textit{title=\text{None}})</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>EnableMemoization()</th>
</tr>
</thead>
</table>

118
## CountMethodCall(fn)

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

## CountDictCall(keyfunc)

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

### 12.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| __revision__  | Value: 'src/engine/SCons/Memoize.py
rel_2.4.0:3365:9259ea1c13d7 ... |
| __doc__       | Value: """Memoi...                                    |
| use_memoizer  | Value: None                                       |
| CounterList   | Value: {}                                          |
| __package__   | Value: None                                       |

### 12.3 Class Counter

```
object  ──
       |        
       |        
SCons.Memoize.Counter
```

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

Base class for counting memoization hits and misses.

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

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

```python
key(self)
```

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

12.4 Class CountValue

```
object
```

```
SCons.Memoize.Counter
```

```
SCons.Memoize.CountValue
```

A counter class for simple, atomic memoized values.

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

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

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

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

`__cmp__()`, `__init__()`, `display()`, `key()`

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattribute__()`, `__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><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>

12.5 Class CountDict

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

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

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

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

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

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

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

*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>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</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. 131)*

- **FS**: scons.Node.FS
  
  *(Section 15, p. 137)*

- **Python**: scons.Node.Python
  
  *(Section 16, p. 185)*

13.2 Functions

```
classname(obj)

Annotate(node)

is_derived_none(node)

is_derived_node(node)
```

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

```
exists_none(node)
```
exists_always(node)

exists_base(node)

exists_entry(node)

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

exists_file(node)

rexists_none(node)

rexists_node(node)

rexists_base(node)

get_contents_none(node)

get_contents_entry(node)

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

get_contents_dir(node)

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

get_contents_file(node)

target_from_source_none(node, prefix, suffix, splitext)

target_from_source_base(node, prefix, suffix, splitext)


```python
changed_since_last_build_node(node, target, prev_ni)
```

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

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

```python
changed_since_last_build_alias(node, target, prev_ni)
```

```python
changed_since_last_build_entry(node, target, prev_ni)
```

```python
changed_since_last_build_state_changed(node, target, prev_ni)
```

```python
decide_source(node, target, prev_ni)
```

```python
decide_target(node, target, prev_ni)
```

```python
changed_since_last_build_python(node, target, prev_ni)
```

```python
store_info_pass(node)
```

```python
store_info_file(node)
```

```python
get_children(node, parent)
```

```python
ignore_cycle(node, stack)
```

```python
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
__getstate__(self)
```

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

```python
__setstate__(self, state)
```

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

```python
convert(self, node, val)
```

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

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

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

```python
update(self, node)
```

Inherited from object

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

13.4.2 Properties

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

13.4.3 Class Variables
13.5 Class BuildInfoBase

The generic base class for build information for a Node.

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

13.5.1 Methods

```python
__getstate__(self)
```

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

```python
__init__(self)
```

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

Overrides: object.__init__ extit(inherited documentation)

```python
__setstate__(self, state)
```

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

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

Inherited from object

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

13.5.2 Properties

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

Inherited from object

__class__

13.5.3 Class Variables

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

13.6 Class Node

object

SCons.Node.Node


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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decider</strong>&lt;br&gt;<em>(self, function)</em></td>
<td></td>
</tr>
<tr>
<td><strong>GetTag</strong>&lt;br&gt;<em>(self, key)</em></td>
<td>Return a user-defined tag.</td>
</tr>
<tr>
<td><strong>Tag</strong>&lt;br&gt;<em>(self, key, value)</em></td>
<td>Add a user-defined tag.</td>
</tr>
<tr>
<td><strong><strong>init</strong></strong>&lt;br&gt;<em>(self)</em></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>add_dependency</strong>&lt;br&gt;<em>(self, depend)</em></td>
<td>Adds dependencies.</td>
</tr>
<tr>
<td><strong>add_ignore</strong>&lt;br&gt;<em>(self, depend)</em></td>
<td>Adds dependencies to ignore.</td>
</tr>
<tr>
<td><strong>add_prerequisite</strong>&lt;br&gt;<em>(self, prerequisite)</em></td>
<td>Adds prerequisites</td>
</tr>
<tr>
<td><strong>add_source</strong>&lt;br&gt;<em>(self, source)</em></td>
<td>Adds sources.</td>
</tr>
<tr>
<td><strong>add_to_implicit</strong>&lt;br&gt;<em>(self, deps)</em></td>
<td></td>
</tr>
</tbody>
</table>
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...)

add_to_waiting_se(self, node)

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()

**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_bininfo(self)

Delete the build info from this node.

disambiguate(self, must_exist=None)

evnt_set(self, env, safe=0)

executor_cleanup(self)

Let the executor clean up any cached information.

exists(self)

Does this node exists?

explain(self)

for_signature(self)

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

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

get__binfo(self)

Fetch a node’s build information.

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

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

get__build__env(self)

Fetch the appropriate Environment to build this node.

get__build__scanner__path(self, scanner)

Fetch the appropriate scanner path for this node.

get__builder(self, default__builder=None)

Return the set builder, or a specified default value

get__cachedir__csig(self)

get__contents(self)

Fetch the contents of the entry.

get__csig(self)
### get_env

`get_env(self)`

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

### get_env_scanner

`get_env_scanner(self, env, kw={})`

### get_executor

`get_executor(self, create=1)`

### get_found_includes

`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

`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

`get_ninfo(self)`

### get_source_scanner

`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(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.

**new_binfo(self)**

**new_ninfo(self)**

**postprocess(self)**

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

**Package SCons.Node**

---

**prepare(self)**

Prepare for this Node to be built.

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

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

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

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

**push_to_cache(self)**

Try to push a node into a cache

**release_target_info(self)**

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

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

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

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

**remove(self)**

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

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

reset_executor(self)

Remove cached executor; forces recompute when needed.

retrieve_from_cache(self)

Try to retrieve the node’s content from a cache

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

Returns true if the node was successfully retrieved.

rexists(self)

Does this node exist locally or in a repository?

scan(self)

Scan this node’s dependents for implicit dependencies.

scanner_key(self)

select_scanner(self, scanner)

Selects a scanner for this Node.

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

**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)

**visited**(self)

Called just after this node has been visited (with or without a build).
Inherited from object

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

```python
__str__(self)
```

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

*Inherited from UserList.UserList*

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

*Inherited from __abcoll.Sequence*

__iter__(), __reversed__()

*Inherited from __abcoll.Sized*
__subclasshook__()  

*Inherited from object*  

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

### 13.7.2 Properties

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

### 13.7.3 Class Variables

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

### 13.8 Class Walker

**object**  

SCons.Node.Walker

An iterator for walking a Node tree.

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

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

```python
__init__(self, node, kids_func=<function get_children at 0x7f3ab64d68c0>, cycle_func=<function ignore_cycle at 0x7f3ab64d6938>, eval_func=<function do_nothing at 0x7f3ab64d69b0>)
```

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

```python
get_next(self)
```

Return the next node for this walk of the tree.

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

```python
is_done(self)
```

**Inherited from object**

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

13.8.2 Properties

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

scons.Node.Alias

Alias nodes.

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

14.1 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: <code>src/engine/SCons/Node/Alias.py</code></td>
</tr>
<tr>
<td></td>
<td>rel_2.4.0:3365:9259ea1c13...</td>
</tr>
<tr>
<td>default_ans</td>
<td>Value: <code>{}</code></td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: <code>'SCons.Node'</code></td>
</tr>
</tbody>
</table>

14.2 Class 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

object

SCons.Node.NodeInfoBase


The generic base class for signature information for a Node.

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

14.3.1 Methods

```python
def str_to_node(self, s):
    pass
```

```python
__getstate__(self)
```

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

```python
__setstate__(self, state)
```

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

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

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

*Inherited from object*

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

14.3.2 Properties
### 14.3.3 Class Variables

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

### 14.4 Class AliasBuildInfo

```
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)**

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

**Inherited from object**

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

#### 14.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</td>
<td>continued on next page</td>
</tr>
</tbody>
</table>

148
Class Alias

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bact, bactsig, bdepends, bdependsigs, bimplicit, bimplicitigs, bsources, bsourcesigs</td>
<td>Inherited from object__class__</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: 2</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__ (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)
**really_build(self, **kw)**

Actually build the node.

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

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

**is_up_to_date(self)**

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


**is_under(self, dir)**

**get_contents(self)**

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

**sconsign(self)**

An Alias is not recorded in .sconsign files

**build(self)**


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

Generate a node’s content signature, the digested signature of its content.

node - the node cache - alternate node to use for the signature cache returns -
the content signature  Overrides: SCons.Node.Node.get_csig

Inherited from SCons.Node.Node (Section 13.6)

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

Inherited from object

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

```python
sconsign_none(node)
```

Return the .sconsign file info for this directory, creating it first if necessary.

```python
save_strings(val)
```

```python
initialize_do_splitdrive()
```

```python
needs_normpath_match(...)
```

match(string[, pos[, endpos]]) --> match object or None. Matches zero or more characters at the beginning of the string

```python
set_duplicate(duplicate)
```

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

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

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

```python
MkdirFunc(target, source, env)
```
get_MkdirBuilder()

get_DefaultSCCSBuilder()

get_DefaultRCSBuilder()

do_diskcheck_match(node, predicate, errorfmt)

ignore_diskcheck_match(node, predicate, errorfmt)

do_diskcheck_rcs(node, name)

ignore_diskcheck_rcs(node, name)

do_diskcheck_sccs(node, name)

ignore_diskcheck_sccs(node, name)

set_diskcheck(list)

diskcheck_types()

has_glob_magic(s)

get_default_fs()
find_file(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.

invalidate_node_memos(targets)

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 rel_2.4.0:3365:9259ea1c13d7 ...</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>Valid_Duplicates</td>
<td>Value: ['hard-soft-copy', 'soft-hard-copy', 'hard-copy', 'soft-c...</td>
</tr>
</tbody>
</table>

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link_Funcs</td>
<td>Value: []</td>
</tr>
<tr>
<td>Link</td>
<td>Value: SCons.Action.Action(LinkFunc, None)</td>
</tr>
<tr>
<td>LocalCopy</td>
<td>Value: SCons.Action.Action(LinkFunc, LocalString)</td>
</tr>
<tr>
<td>Unlink</td>
<td>Value: SCons.Action.Action(UnlinkFunc, None)</td>
</tr>
<tr>
<td>Mkdir</td>
<td>Value: SCons.Action.Action(MkdirFunc, None, presub= None)</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: DiskChecker('rcs', do_diskcheck_rcs, ignore_diskcheck_rcs)</td>
</tr>
<tr>
<td>diskcheck_sccs</td>
<td>Value: DiskChecker('sccs', do_diskcheck_sccs, ignore_diskcheck_sccs)</td>
</tr>
<tr>
<td>diskcheckers</td>
<td>Value: [diskcheck_match, diskcheck_rcs, diskcheck_sccs,]</td>
</tr>
<tr>
<td>node_bwcomp</td>
<td>Value: {'abspath': &lt;unbound method Base.get_abspath&gt;, 'labspath'...</td>
</tr>
<tr>
<td>glob_magic_check</td>
<td>Value: re.compile(r'[^*?[]')</td>
</tr>
<tr>
<td>default_fs</td>
<td>Value: None</td>
</tr>
<tr>
<td>OS_SEP</td>
<td>Value: '/'</td>
</tr>
<tr>
<td>UNC_PREFIX</td>
<td>Value: '//'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Node'</td>
</tr>
<tr>
<td>has_unc</td>
<td>Value: False</td>
</tr>
<tr>
<td>os_sep_is_slash</td>
<td>Value: True</td>
</tr>
</tbody>
</table>
15.3 Class EntryProxyAttributeError

object

exceptions.BaseException

exceptions.Exception

exceptions.StandardError

exceptions.AttributeError

SCons.Node.FS.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)

Inherited from exceptions.AttributeError

__new__()  

Inherited from exceptions.BaseException

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

Inherited from object

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

15.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Inherited from exceptions.BaseException | conti

continued on next page |
Class DiskChecker

object →
SCons.Node.FS.DiskChecker

15.4 Class DiskChecker

object

SCons.Node.FS.DiskChecker

15.4.1 Methods

___init___(self, type, do, ignore)

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

___call___(self, *args, **kw)

set(self, list)

Inherited from object

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

15.4.2 Properties

<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>

157
15.5 Class EntryProxy

```
object
  SCons.Util.Proxy
    SCons.Node.FS.EntryProxy
```

15.5.1 Methods

```python
__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__

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

Retrieve an attribute from the wrapped object. If the named attribute doesn’t exist, AttributeError is raised. Overrides: SCons.Util.Proxy.__getattr__

Inherited from SCons.Util.Proxy (Section 36.5)

__cmp__(), __init__(), get()

Inherited from object

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

15.5.2 Properties

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

15.5.3 Class Variables
15.6 Class Base

object

SCons.Node.Node

SCons.Node.FS.Base


A generic class for file system entries. This class is for when we don’t know yet whether the entry being looked up is a file or a directory. Instances of this class can morph into either Dir or File objects by a later, more precise lookup.

Note: this class does not define `__cmp__` and `__hash__` for efficiency reasons. SCons does a lot of comparing of Node.FS.{Base,Entry,File,Dir} objects, so those operations must be as fast as possible, which means we want to use Python’s built-in object identity comparisons.

15.6.1 Methods

```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__

```python
str_for_display(self)
```

```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.

```python
get_dir(self)
```
get_suffix(self)

rfile(self)

__getattr__(self, attr)

Together with the node_bwcomp dict defined below, this method provides a simple backward compatibility layer for the Node attributes ‘abspath’, ‘labspath’, ‘path’, ‘tpath’, ‘suffix’ and ‘path_elements’. These Node attributes used to be directly available in v2.3 and earlier, but have been replaced by getter methods that initialize the single variables lazily when required, in order to save memory. The redirection to the getters lets older Tools and SConstruct continue to work without any additional changes, fully transparent to the user. Note, that __getattr__ is only called as fallback when the requested attribute can’t be found, so there should be no speed performance penalty involved for standard builds.

__str__(self)

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

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)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getmtime(self)</code></td>
<td>If this node is in a build path, return the node corresponding to its source file. Otherwise, return ourself.</td>
</tr>
<tr>
<td><code>getsize(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>isdir(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>isfile(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>islink(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>is_under(self, dir)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_local(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>srcnode(self)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_path(self, dir=None)</code></td>
<td>Return path relative to the current working directory of the Node.FS.Base object that owns us.</td>
</tr>
<tr>
<td><code>set_src_builder(self, builder)</code></td>
<td>Set the source code builder for this node.</td>
</tr>
<tr>
<td><code>src_builder(self)</code></td>
<td></td>
</tr>
</tbody>
</table>

Fetch the source code builder for this node.

If there isn’t one, we cache the source code builder specified for the directory (which in turn will cache the value from its parent directory, and so on up to the file system root).
get_abspath(self)


get_labspath(self)

Get the absolute path of the file.

get_internal_path(self)

get_tpath(self)

get_path_elements(self)

for_signature(self)

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

get_subst_proxy(self)

This method is expected to return an object that will function exactly like this Node, except that it implements any additional special features that we would like to be in effect for Environment variable substitution. The principle use is that some Nodes would like to implement a __getattr__() method, but putting that in the Node type itself has a tendency to kill performance. We instead put it in a proxy and return it from this method. It is legal for this method to return self if no new functionality is needed for Environment substitution. Overrides: SCons.Node.Node.get_subst_proxy (inherited documentation)
target_from_source(self, prefix, suffix, splitext=<function splitext at 0x7f3ab66bf230>)

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.

Rfindalldirs(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(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(), add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_s_e(), add_wkid(), all_children(), alter_targets(), build(), builder_set(), built(), changed(), children(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(), disambiguate(), env_set(), executor_cleanup(), explain(), get_binfo(), get_build_env(), get_build_scanner_path(), get_builder(), get_cachefile_csig(), get_contents(), get_csig(), get_env(), get_env_scanner(), get_executor(), get_found_includes(), get_implicit_deps(), get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(), get_stored_info(), get_string(), get_target_scanner(), hasBuilder(), 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_nocheckout(), set_pseudo(), set_specific_source(), set_state(), visited()
**Inherited from object**

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

### 15.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwd</td>
<td></td>
</tr>
<tr>
<td>dir</td>
<td></td>
</tr>
<tr>
<td>duplicate</td>
<td></td>
</tr>
<tr>
<td>sbuilder</td>
<td></td>
</tr>
</tbody>
</table>

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

always_build, attributes, binfo, builder, cached,
changed_since_last_build, depends, depends_set, env, executor, ignore,
ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo,
nocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect,
side_effects, sources, sources_set, state, store_info, waiting_parents,
waiting_s_e, wkids

*Inherited from object*

```python
__class__
```

### 15.6.3 Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>fs</td>
<td></td>
</tr>
</tbody>
</table>

### 15.7 Class Entry

```
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

```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)
```

```python
disambiguate(self, must_exist=None)
```


```python
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

```python
scanner_key(self)
```


```python
get_contents(self)
```


```python
get_text_contents(self)
```

Fetch the decoded text contents of a Unicode encoded Entry.

Since this should return the text contents from the file system, we check to see into what sort of subclass we should morph this Entry.
must_be_same(self, klass)

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

exists(self)

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

rel_path(self, other)

new_ninfo(self)


get_subst_proxy(self)

This method is expected to return an object that will function exactly like this Node, except that it implements any additional special features that we would like to be in effect for Environment variable substitution. The principle use is that some Nodes would like to implement a __getattr__() method, but putting that in the Node type itself has a tendency to kill performance. We instead put it in a proxy and return it from this method. It is legal for this method to return self if no new functionality is needed for Environment substitution. Overrides: SCons.Node.Node.get_subst_proxy (inherited documentation)

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

RDirs(), Rfindalldirs(), __getattr__(), __str__(), for_signature(), get_abspath(), get_dir(), get_internal_path(), get_labspath(), get_path(), get_path_elements(), get_suffix(), get_tpath(), getmtime(), getsize(), is_under(), isdir(), isfile(), islink(), rentry(), reexists(), rstr(), set_local(), set_src_builder(), src_builder(), srcnode(), stat(), str_for_display(), target_from_source()  

Inherited from SCons.Node.Node (Section 13.6)

Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(), add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_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(), 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_noclean(), set_precious(), set_pseudo(),
set_specific_source(), set_state(), visited()

**Inherited from object**

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

### 15.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachedir_csig</td>
<td></td>
</tr>
<tr>
<td>cachesig</td>
<td></td>
</tr>
<tr>
<td>contentsig</td>
<td></td>
</tr>
<tr>
<td>dirname</td>
<td></td>
</tr>
<tr>
<td>entries</td>
<td></td>
</tr>
<tr>
<td>on_disk_entries</td>
<td></td>
</tr>
<tr>
<td>rcs_dir</td>
<td></td>
</tr>
<tr>
<td>released_target_info</td>
<td></td>
</tr>
<tr>
<td>repositories</td>
<td></td>
</tr>
<tr>
<td>root</td>
<td></td>
</tr>
<tr>
<td>scanner_paths</td>
<td></td>
</tr>
<tr>
<td>sccs_dir</td>
<td></td>
</tr>
<tr>
<td>searched</td>
<td></td>
</tr>
<tr>
<td>srcdir</td>
<td></td>
</tr>
<tr>
<td>variant_dirs</td>
<td></td>
</tr>
</tbody>
</table>

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

cwd, dir, duplicate, sbuilder

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

always_build, attributes, binfo, builder, cached,
changed_since_last_build, depends, depends_set, env, executor, ignore,
ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo,
ocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect,
side_effects, sources, sources_set, state, store_info, waiting_parents,
waiting_s_e, wkids

*Inherited from object*

__class__
15.7.3 Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs, name</td>
<td>Inherited from SCons.Node.FS.Base (Section 15.6)</td>
</tr>
</tbody>
</table>

15.8 Class LocalFS

object

\[ \text{SCons.Node.FS.LocalFS} \]

Known Subclasses: SCons.Node.FS.FS

15.8.1 Methods

- chmod(self, path, mode)
- copy(self, src, dst)
- copy2(self, src, dst)
- exists(self, path)
- getmtime(self, path)
- getsize(self, path)
- isdir(self, path)
- isfile(self, path)
- link(self, src, dst)
- lstat(self, path)
- listdir(self, path)
- makedirs(self, path)
```python
mkdir(self, path)
rename(self, old, new)
stat(self, path)
symlink(self, src, dst)
open(self, path)
unlink(self, path)
islink(self, path)
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><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></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.

**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, exclude=None, cwd=None)

Globs
This is mainly a shim layer

Inherited from SCons.Node.FS.LocalFS (Section 15.8)

chmod(), copy(), copy2(), exists(), getmtime(), getsize(), isdir(), isfile(), islink(), link(), listdir(), lstat(), 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>

15.10 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)
Class DirBuildInfo

Inherited from SCons.Node.NodeInfoBase(Section 13.4)

__getstate__(), __setstate__(), convert(), format(), merge(), update()

Inherited from object

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

15.10.2 Properties

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

15.10.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
<tr>
<td>fs</td>
<td>Value: None</td>
</tr>
</tbody>
</table>

15.11 Class DirBuildInfo

object

SCons.Node.BuildInfoBase

SCons.Node.FS.DirBuildInfo

The generic base class for build information for a Node.

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

15.11.1 Methods

Inherited from SCons.Node.BuildInfoBase(Section 13.5)

__getstate__(), __init__(), __setstate__(), merge()
Class Dir

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 SCons.Node.BuildInfoBase (Section 13.5)</td>
<td>bact, bactsig, bdepends, bdependsgis, bimplicit, bimplicitsgis, bsources, bsourcesigs</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

15.11.3 Class Variables

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

15.12 Class Dir

object

SCons.Node.Node

SCons.Node.FS.Base

SCons.Node.FS.Dir

Known Subclasses: SCons.Node.FS.RootDir

A class for directories in a file system.
Class Dir

Module SCons.Node.FS

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)
```

```python
Entry(self, name)
```

Looks up or creates an entry node named ‘name’ relative to this directory.

```python
Dir(self, name, create=True)
```

Looks up or creates a directory node named ‘name’ relative to this directory.

```python
File(self, name)
```

Looks up or creates a file node named ‘name’ relative to this directory.

```python
link(self, srcrep, duplicate)
```

Set this directory as the variant directory for the supplied source directory.

```python
getRepositories(self)
```

Returns a list of repositories for this directory.

```python
get_all_rdirs(self)
```

```python
addRepository(self, dir)
```

175
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>up(self)</code></td>
<td>Return a path to 'other' relative to this directory.</td>
</tr>
<tr>
<td><code>rel_path(self, other)</code></td>
<td>Return a path to 'other' relative to this directory.</td>
</tr>
<tr>
<td><code>get_found_includes(self, env, scanner, path)</code></td>
<td>Return this directory’s implicit dependencies. We don’t bother caching the results because the scan typically shouldn’t be requested more than once (as opposed to scanning .h file contents, which can be requested as many times as the files is #included by other files). Overrides: SCons.Node.Node.get_found_includes</td>
</tr>
<tr>
<td><code>prepare(self)</code></td>
<td>Prepare for this Node to be built.</td>
</tr>
</tbody>
</table>

**prepare(self)**

Prepare for this Node to be built.

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

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

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

Overriding this method allows for a Node subclass to remove the underlying file from the file system. Note that subclass methods should call this base class method to get the child check and the BuildInfo structure. Overrides: SCons.Node.Node.prepare exit(inherited documentation)
**build**(*self*, **kw*)


**multiple_side_effect_has_builder**(*self*)

Return whether this Node has a builder or not.

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

**alter_targets**(*self*)


**scanner_key**(*self*)


**get_text_contents**(*self*)

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

**get_contents**(*self*)

Return content signatures and names of all our children separated by new-lines. Ensure that the nodes are sorted. Overrides: SCons.Node.Node.get_contents
**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`

**do_duplicate** *(self, src)*

**is_up_to_date** *(self)*

If any child is not up-to-date, then this directory isn’t, either. Overrides: `SCons.Node.Node.is_up_to_date`

**rdir** *(self)*

**sconsign** *(self)*

Return the `.sconsign` file info for this directory.

**srcnode** *(self)*

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`

**get_timestamp** *(self)*

Return the latest timestamp from among our children

**get_abspath** *(self)*

### get\_labspath(self)

Get the absolute path of the file. Overrides: 
SCons.Node.FS.Base.get\_labspath

### get\_internal\_path(self)

Overrides: SCons.Node.FS.Base.get\_internal\_path

### get\_tpath(self)

Overrides: SCons.Node.FS.Base.get\_tpath

### get\_path\_elements(self)

Overrides: SCons.Node.FS.Base.get\_path\_elements

### entry\_abspath(self, name)

### entry\_labspath(self, name)

### entry\_path(self, name)

### entry\_tpath(self, name)

### entry\_exists\_on\_disk(self, name)

Searches through the file/dir entries of the current directory, and returns True if a physical entry with the given name could be found.

@see entry\_exists\_on\_disk

### reentry\_exists\_on\_disk(self, name)

Searches through the file/dir entries of the current and all its remote directories (repos), and returns True if a physical entry with the given name could be found. The local directory (self) gets searched first, so repositories take a lower precedence regarding the searching order.

@see entry\_exists\_on\_disk
srcdir_list(self)

srcdir_duplicate(self, name)

srcdir_find_file(self, filename)

dir_on_disk(self, name)

file_on_disk(self, name)

walk(self, func, arg)

Walk this directory tree by calling the specified function for each directory in
the tree.

This behaves like the os.path.walk() function, but for in-memory Node.FS.Dir
objects. The function takes the same arguments as the functions passed to
os.path.walk():

    func(arg, dirname, fnames)

Except that 'dirname' will actually be the directory Node, not the string. The
'.' and '..' entries are excluded from fnames. The fnames list may be modified
in-place to filter the subdirectories visited or otherwise impose a specific order.
The "arg" argument is always passed to func() and may be used in any way (or
ignored, passing None is common).
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 'exclude' argument, if not None, must be a pattern or a list of patterns following the same UNIX shell semantics. Elements matching at least one pattern of this list will be excluded from the result.

The underlying algorithm is adapted from the `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)

- `RDirs()`, `Rfindalldirs()`, `__getattr__()`, `__str__()`, `exists()`, `for_signature()`, `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()`

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

- `Decider()`, `GetTag()`, `Tag()`, `add_dependency()`, `add_ignore()`, `add_prerequisite()`, `add_source()`, `add_to_implicit()`, `add_to_waiting_parents()`, `add_to_waiting_s_e()`, `add_wkid()`, `all_children()`, `builder_set()`, `built()`, `changed()`, `children()`, `children_are_up_to_date()`, `clear()`, `clear_memoized_values()`, `del_binfo()`, `disambiguate()`, `env_set()`, `execu-
tor_cleanup(), explain(), get_binfo(), get_build_env(), get_build_scanner_path(),
get_builder(), get_cachedir_csig(), get_env(), get_executor(), get_implicit_deps(),
get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(), get_stored_info(),
get_string(), has_builder(), has_explicit_builder(), is-derived(), is_literal(), make_ready(),
missing(), new_binfo(), new_ninfo(), postprocess(), push_to_cache(), release_target_info(),
remove(), render_include_tree(), reset_executor(), retrieve_from_cache(), scan(),
select_scanner(), set_always_build(), set_executor(), set_explicit(), set_nocache(),
set_noclean(), set_precious(), set_pseudo(), set_specific_source(), set_state(),
visited()

**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>cachedir_csig</td>
<td></td>
</tr>
<tr>
<td>cachesig</td>
<td></td>
</tr>
<tr>
<td>contentsig</td>
<td></td>
</tr>
<tr>
<td>dirname</td>
<td></td>
</tr>
<tr>
<td>entries</td>
<td></td>
</tr>
<tr>
<td>on_disk_entries</td>
<td></td>
</tr>
<tr>
<td>rcs_dir</td>
<td></td>
</tr>
<tr>
<td>released_target_info</td>
<td></td>
</tr>
<tr>
<td>repositories</td>
<td></td>
</tr>
<tr>
<td>root</td>
<td></td>
</tr>
<tr>
<td>scanner_paths</td>
<td></td>
</tr>
<tr>
<td>sccs_dir</td>
<td></td>
</tr>
<tr>
<td>searched</td>
<td></td>
</tr>
<tr>
<td>srcdir</td>
<td></td>
</tr>
<tr>
<td>variant_dirs</td>
<td></td>
</tr>
</tbody>
</table>

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

cwd, dir, duplicate, sbuilder

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

always_build, attributes, binfo, builder, cached,
changed_since_last_build, depends, depends_set, env, executor, ignore,
ignore_set, implicit, implicit_set, includes, is_explicit, linked, ninfo,
ocache, noclean, precious, prerequisites, pseudo, ref_count, side_effect,
side_effects, sources, sources_set, state, store_info, waiting_parents,
waiting_s_e, wkids

*Inherited from object*
15.12.3  Instance Variables

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

```
__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)
**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

**__str__**(self)

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

**entry**__**abspath**(self, name)

Overrides: SCons.Node.FS.Dir.entry_abspath

**entry**__**labspath**(self, name)

Overrides: SCons.Node.FS.Dir.entry_labspath

**entry**__**path**(self, name)

Overrides: SCons.Node.FS.Dir.entry_path

**entry**__**tpath**(self, name)

Overrides: SCons.Node.FS.Dir.entry_tpath

**is**__**under**(self, dir)

Overrides: SCons.Node.FS.Base.is_under

**up**(self)

Overrides: SCons.Node.FS.Dir.up

**get**__**dir**(self)

Overrides: SCons.Node.FS.Base.get_dir

**src**__**builder**(self)

Fetch the source code builder for this node.

If there isn’t one, we cache the source code builder specified for the directory (which in turn will cache the value from its parent directory, and so on up to the file system root). Overrides: SCons.Node.FS.Base.src_builder

Inherited from SCons.Node.FS.Dir (Section 15.12)

Dir(), Entry(), File(), addRepository(), alter_targets(), build(), dir_on_disk(),
diskcheck_match(), do_duplicate(), entry_exists_on_disk(), file_on_disk(), get-
Repositories(), get_abspath(), get_all_rdirs(), get_contents(), get_csig(), get_env_scanner(),
get_found_includes(), get_internal_path(), get_labspath(), get_path_elements(),
get_target_scanner(), get_text_contents(), get_timestamp(), get_tpath(), glob(),
is_up_to_date(), link(), multiple_side_effect_has_builder(), prepare(), rdir(),
rel_path(), rentry_exists_on_disk(), scanner_key(), sconsign(), srcdir_duplicate(),
srcdir_find_file(), srcdir_list(), srcnode(), walk()

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

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

Inherited from SCons.Node.Node (Section 13.6)

Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(),
add_source(), add_toImplicit(), add_to_waiting_parents(), add_to_waiting_se(),
add_wkid(), all_children(), builder_set(), built(), changed(), children(), children_are_up_to_date(),
clear(), clear_memoized_values(), del_binfo(), disambiguate(), env_set(), executor_cleanup(), explain(),
get_binfo(), get_build_env(), get_build_scanner_path(),
get_builder(), get_cachedir_csig(), get_env(), get_executor(), get_implicit_deps(),
get_ninfo(), get_source_scanner(), get_state(), get_stored_implicit(), get_stored_info(),
geetstring(), has_builder(), has_explicit_builder(), is_derived(), is_literal(), make_ready(),
missing(), new_binfo(), new_ninfo(), postprocess(), push_to_cache(), release_target_info(),
remove(), render_include_tree(), reset_executor(), retrieve_from_cache(), scan(),
select_scanner(), set_always_build(), set_executor(), set_explicit(), set_nocache(),
set_noclean(), set_precious(), set_pseudo(), set_specific_source(), set_state(),
visited()

Inherited from object

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

15.13.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.FS.Dir (Section 15.12)</td>
<td></td>
</tr>
<tr>
<td>cachedir_csig, cachesig, contentsig, dirname, entries, on_disk_entries,</td>
<td></td>
</tr>
<tr>
<td>rcs_dir, released_target_info, repositories, root, scanner_paths, scs_dir,</td>
<td></td>
</tr>
<tr>
<td>searched, srcdir, variant_dirs</td>
<td>continued on next page</td>
</tr>
</tbody>
</table>
Class FileNodeInfo

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

15.13.3 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>fs</td>
<td>Reference to parent Node.FS object</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

```
str_to_node(self, s)
```
Return all fields that shall be pickled. Walk the slots in the class hierarchy and add those to the state dictionary. If a '__dict__' slot is available, copy all entries to the dictionary. Also include the version id, which is fixed for all instances of a class. Overrides: SCons.Node.NodeInfoBase._getstate_

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

Inherited from SCons.Node.NodeInfoBase (Section 13.4)

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

Inherited from object

_delattr_(), _format_(), _getattribute_(), _hash_(), _init_(),
_new_(), _reduce_(), _reduce_ex_(), _repr_(), _setattr_(),
sizeof(), _str_(), __subclasshook__()

15.14.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csig</td>
<td></td>
</tr>
<tr>
<td>size</td>
<td></td>
</tr>
<tr>
<td>timestamp</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object

__class__

15.14.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_version_id</td>
<td>Value: 2</td>
</tr>
<tr>
<td>field_list</td>
<td>['csig', 'timestamp', 'size']</td>
</tr>
<tr>
<td>fs</td>
<td>Value: None</td>
</tr>
</tbody>
</table>
15.15 Class FileBuildInfo

The generic base class for build information for a Node.

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

15.15.1 Methods

**convert_from_sconsign**(self, dir, name)

Converts a newly-read FileBuildInfo object for in-SCons use

For normal up-to-date checking, we don’t have any conversion to perform—but we’re leaving this method here to make that clear.

**convert_to_sconsign**(self)

Converts this FileBuildInfo object for writing to a .sconsign file

This replaces each Node in our various dependency lists with its usual string representation: relative to the top-level SConstruct directory, or an absolute path if it’s outside.

**format**(self, names=0)
prepare_dependencies(self)

Prepares a FileBuildInfo object for explaining what changed

The bsources, bdepends and bimplicit lists have all been stored on disk as
paths relative to the top-level SConstruct directory. Convert the strings to
actual Nodes (for use by the --debug=explain code and --implicit-cache).

Inherited from SCons.Node.BuildInfoBase(Section 13.5)

__getstate__(), __init__(), __setstate__(), merge()

Inherited from object

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

15.15.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</td>
<td>bact, bactsig, bdepends, bdependsigs, bimplicit, bimplicitsigs, bsources, bsourcesigs</td>
</tr>
<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: 2</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

**diskcheck_match**(self)

**__init__**(self, name, directory, fs)
Initialize a generic Node.FS.Base object.

Call the superclass initialization, take care of setting up our relative and absolute paths, identify our parent directory, and indicate that this node should use signatures. Overrides: object.__init__ (inherited documentation)

**Entry**(self, name)
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)

**convert_old_entry**(self, old_entry)

**get_stored_info**(self)

**get_stored_implicit**(self)

**rel_path**(self, other)

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

**push_to_cache**(self)
Try to push the node into a cache. Overrides: SCons.Node.Node.push_to_cache
**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. Overrides: SCons.Node.Node.retrieve_from_cache

**visited(self)**

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

**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 (inherited documentation)

prepare(self)

Prepare for this file to be created. Overrides: SCons.Node.Node.prepare

remove(self)


do_duplicate(self, src)

exists(self)

Does this node exists? Overrides: SCons.Node.Node.exists (inherited documentation)
**get_max_drift_csig**(self)

Returns the content signature currently stored for this node if it’s been unmodified longer than the max_drift value, or the max_drift value is 0. Returns None otherwise.

**get_csig**(self)

Generate a node’s content signature, the digested signature of its content.

node - the node cache - alternate node to use for the signature cache returns - the content signature Overrides: SCons.Node.Node.get_csig

**builder_set**(self, builder)


**built**(self)

Called just after this File node is successfully built.

Just like for 'release_target_info' we try to release some more target node attributes in order to minimize the overall memory consumption.

@see: release_target_info Overrides: SCons.Node.Node.built

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

Returns if the node is up-to-date with respect to the BuildInfo stored last time it was built.

For File nodes this is basically a wrapper around Node.changed(), but we allow the return value to get cached after the reference to the Executor got released in release_target_info().

@see: Node.changed() Overrides: SCons.Node.Node.changed

**changed_content**(self, target, prev_ni)

**changed_state**(self, target, prev_ni)
changed_timestamp_then_content(self, target, prev_ni)

changed_timestamp_newer(self, target, prev_ni)

changed_timestamp_match(self, target, prev_ni)

is_up_to_date(self)

Default check for whether the Node is current: unknown Node subtypes are always out of date, so they will always get built. Overrides: SCons.Node.Node.is_up_to_date extit(inherited documentation)

rfile(self)

Overrides: SCons.Node.FS.Base.rfile

rstr(self)

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

get_cachedir_csig(self)

Fetch a Node’s content signature for purposes of computing another Node’s cachesig.

This is a wrapper around the normal get_csig() method that handles the somewhat obscure case of using CacheDir with the -n option. Any files that don’t exist would normally be 'built' by fetching them from the cache, but the normal get_csig() method will try to open up the local file, which doesn’t exist because the -n option meant we didn’t actually pull the file from cachedir. But since the file does actually exist in the cachedir, we can use its contents for the csig. Overrides: SCons.Node.Node.get_cachedir_csig

get_contents_sig(self)

A helper method for get_cachedir_bsig.

It computes and returns the signature for this node’s contents.
**get_cachedir_bsig(self)**

Return the signature for a cached file, including its children.

It adds the path of the cached file to the cache signature, because multiple targets built by the same action will all have the same build signature, and we have to differentiate them somehow.

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

RDirs(), Rfindalldirs(), __getattr__(), __str__(), for_signature(), get_abspath(), get_dir(), get_internal_path(), get_labspath(), get_path(), get_path_elements(), get_subst_proxy(), get_suffix(), get_tpath(), getmtime(), getsize(), is_under(), isdir(), isfile(), islink(), must_be_same(), rentry(), reexists(), set_local(), set_src_builder(), src_builder(), srcnode(), stat(), str_for_display(), target_from_source()

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

Decider(), GetTag(), Tag(), add_dependency(), add_ignore(), add_prerequisite(), add_source(), add_to_implicit(), add_to_waiting_parents(), add_to_waiting_s_e(), add_wkid(), all_children(), build(), children(), children_are_up_to_date(), clear(), clear_memoized_values(), del_binfo(), disambiguate(), env_set(), executor_cleanup(), explain(), get_binfo(), get_build_env(), get_build_scanner_path(), get_builder(), get_env(), get_env_scanner(), get_executor(), get_implicit_deps(), get_ninfo(), get_source_scanner(), get_state(), get_string(), get_target_scanner(), has_builder(), has_explicit_builder(), is Derived(), is_literal(), missing(), multiple_side_effect_has_builder(), new_binfo(), new_ninfo(), postprocess(), render_include_tree(), reset_executor(), scan(), select_scanner(), set_always_build(), set_executor(), set_explicit(), set_nocache(), set_noclean(), set_precious(), set_pseudo(), set_specific_source(), set_state()

**Inherited from object**

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

15.16.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachedir_csig</td>
<td></td>
</tr>
<tr>
<td>cachesig</td>
<td></td>
</tr>
<tr>
<td>contentsig</td>
<td></td>
</tr>
<tr>
<td>dirname</td>
<td></td>
</tr>
<tr>
<td>entries</td>
<td></td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>on_disk_entries</td>
<td></td>
</tr>
<tr>
<td>rcs_dir</td>
<td></td>
</tr>
<tr>
<td>released_target_info</td>
<td></td>
</tr>
<tr>
<td>repositories</td>
<td></td>
</tr>
<tr>
<td>root</td>
<td></td>
</tr>
<tr>
<td>scanner_paths</td>
<td></td>
</tr>
<tr>
<td>sccs_dir</td>
<td></td>
</tr>
<tr>
<td>searched</td>
<td></td>
</tr>
<tr>
<td>srcdir</td>
<td></td>
</tr>
<tr>
<td>variant_dirs</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwd, dir, duplicate, sbuilder</td>
<td></td>
</tr>
</tbody>
</table>

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

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

*Inherited from object*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></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>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', 'bimplicitssigs', 'bdependsigs']</td>
</tr>
</tbody>
</table>

### 15.16.4 Instance Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs, name</td>
<td></td>
</tr>
</tbody>
</table>

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

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

15.17.1 Methods

```python
__init__(self)
```

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

```python
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.

```python
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.
Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(),
__reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(),
__str__(), __subclasshook__()
16  Module SCons.Node.Python

scons.Node.Python

Python nodes.

16.1  Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Node/Python.py rel_2.4.0:3365:9259ea1c1...'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Node'</td>
</tr>
</tbody>
</table>

16.2  Class ValueNodeInfo

object

SCons.Node.NodeInfoBase


The generic base class for signature information for a Node.

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

16.2.1  Methods

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

```python
__getstate__(self)
```

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

Module SCons.Node.Python

```python
__setstate__(self, state)
```

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

Inherited from SCons.Node.NodeInfoBase (Section 13.4)

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

Inherited from object

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

16.2.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csig</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object

__class__

16.2.3 Class Variables

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

16.3 Class ValueBuildInfo

The generic base class for build information for a Node.

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

_Inherited from SCons.Node.BuildInfoBase (Section 13.5)_

__getstate__(), __init__(), __setstate__(), merge()

_Inherited from object_

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

16.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from SCons.Node.BuildInfoBase (Section 13.5)</td>
<td>bact, bactsig, bdepends, bdependsigs, bimplicit, bimplicitsigns, bsources, bsourcesigs</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

16.3.3 Class Variables

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

16.4 Class Value

\[\text{object} \downarrow \]

\[\text{SCons.Node.Node} \downarrow \]

\[\text{SCons.Node.Python.Value} \]

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__
```

```python
str_for_display(self)
```

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

```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
```

```python
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
```

```python
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.
```
### Class Value

**Module SCons.Node.Python**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>is_under</strong>(self, dir)</td>
<td></td>
</tr>
<tr>
<td><strong>write</strong>(self, built_value)</td>
<td>Set the value of the node.</td>
</tr>
<tr>
<td><strong>read</strong>(self)</td>
<td>Return the value. If necessary, the value is built.</td>
</tr>
<tr>
<td><strong>get_text_contents</strong>(self)</td>
<td>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.</td>
</tr>
<tr>
<td><strong>get_contents</strong>(self)</td>
<td>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. Overrides: SCons.Node.Node.get_contents</td>
</tr>
<tr>
<td><strong>changed_since_last_build</strong>(self, target, prev_ni)</td>
<td>Overrides: SCons.Node.Node.changed_since_last_build</td>
</tr>
<tr>
<td><strong>get_csig</strong>(self, calc=None)</td>
<td>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</td>
</tr>
</tbody>
</table>

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

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

Inherited from object

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

16.4.2 Properties

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

Inherited from SCons.Node.Node (Section 13.6)
always_build, attributes, binfo, builder, cached, depends, depends_set,
env, executor, ignore, ignore_set, implicit, implicit_set, includes,
is_explicit, linked, ninfo, nocache, noclean, precious, prerequisites, pseudo,
ref_count, side_effect, side_effects, sources, sources_set, state, store_info,
waiting_parents, waiting_s_e, wkids

Inherited from object
__class__
17 Module SCons.PathList

SCons.PathList

A module for handling lists of directory paths (the sort of things that get set as CPPPATH, LIBPATH, etc.) with as much caching of data and efficiency as we can while still keeping the evaluation delayed so that we Do the Right Thing (almost) regardless of how the variable is specified.

17.1 Functions

**node__conv**(obj)

This is the 'string conversion' routine that we have our substitutions use to return Nodes, not strings. This relies on the fact that an EntryProxy object has a get() method that returns the underlying Node that it wraps, which is a bit of architectural dependence that we might need to break or modify in the future in response to additional requirements.

**PathList**(pathlist)

Returns the cached _PathList object for the specified pathlist, creating and caching a new object as necessary.

17.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value:  'src/engine/SCons/PathList.py rel_2.4.0:3365:9259ea1c13d7...</td>
</tr>
<tr>
<td><strong>doc</strong></td>
<td>Value:  &quot;&quot;&quot;SCons.PathL...</td>
</tr>
<tr>
<td>TYPE_STRING_NO_SUBST</td>
<td>Value:  0</td>
</tr>
<tr>
<td>TYPE_STRING_SUBST</td>
<td>Value:  1</td>
</tr>
<tr>
<td>TYPE_OBJECT</td>
<td>Value:  2</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value:  'SCons'</td>
</tr>
</tbody>
</table>
18 Module SCons.SConf

SCons.SConf

Autoconf-like configuration support.

In other words, SConf allows to run tests on the build machine to detect capabilities of system and do some things based on result: generate config files, header files for C/C++, update variables in environment.

Tests on the build system can detect if compiler sees header files, if libraries are installed, if some command line options are supported etc.

18.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetBuildType</strong> <em>(type)</em></td>
</tr>
<tr>
<td><strong>SetCacheMode</strong> <em>(mode)</em></td>
</tr>
<tr>
<td>Set the Configure cache mode. mode must be one of &quot;auto&quot;, &quot;force&quot;, or 'cache'.</td>
</tr>
<tr>
<td><strong>SetProgressDisplay</strong> <em>(display)</em></td>
</tr>
<tr>
<td>Set the progress display to use (called from SCons.Script)</td>
</tr>
<tr>
<td><strong>NeedConfigHBuilder</strong> ()</td>
</tr>
<tr>
<td><strong>CreateConfigHBuilder</strong> <em>(env)</em></td>
</tr>
<tr>
<td>Called if necessary just before the building targets phase begins.</td>
</tr>
<tr>
<td><strong>SConf</strong> *(args, *<em>kw)</em></td>
</tr>
<tr>
<td><strong>CheckFunc</strong> <em>(context, function_name, header=\text{\texttt{None}}, language=\text{\texttt{None}})</em></td>
</tr>
<tr>
<td><strong>CheckType</strong> <em>(context, type_name, includes='', language=\text{\texttt{None}})</em></td>
</tr>
</tbody>
</table>
**CheckTypeSize**

\[\text{CheckTypeSize}(\text{context}, \text{type\_name}, \text{includes}='', \text{language}=	ext{None}, \text{expect}=	ext{None})\]

**CheckDeclaration**

\[\text{CheckDeclaration}(\text{context}, \text{declaration}, \text{includes}='', \text{language}=	ext{None})\]

**createIncludesFromHeaders**

\[\text{createIncludesFromHeaders}(\text{headers}, \text{leave\_Last}, \text{include\_quotes}='''')\]

**CheckHeader**

\[\text{CheckHeader}(\text{context}, \text{header}, \text{include\_quotes}='<>', \text{language}=	ext{None})\]

A test for a C or C++ header file.

**CheckCC**

\[\text{CheckCC}(\text{context})\]

**CheckCXX**

\[\text{CheckCXX}(\text{context})\]

**CheckSHCC**

\[\text{CheckSHCC}(\text{context})\]

**CheckSHCXX**

\[\text{CheckSHCXX}(\text{context})\]

**CheckCHeader**

\[\text{CheckCHeader}(\text{context}, \text{header}, \text{include\_quotes}='''')\]

A test for a C header file.

**CheckCXXHeader**

\[\text{CheckCXXHeader}(\text{context}, \text{header}, \text{include\_quotes}='''')\]

A test for a C++ header file.

**CheckLib**

\[\text{CheckLib}(\text{context}, \text{library}=	ext{None}, \text{symbol}='\text{main}', \text{header}=	ext{None}, \text{language}=	ext{None}, \text{autoadd}=1)\]

A test for a library. See also CheckLibWithHeader. Note that library may also be None to test whether the given symbol compiles without flags.
**CheckLibWithHeader**(*context, libs, header, language, call=None, autoadd=1*)

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 _with_ a trailing ';'. As in CheckLib, we support library=1None, to test if the call compiles without extra link flags.

### 18.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>__revision__</code></td>
<td>Value: 'src/engine/SCons/SConf.py rel_2.4.0:3365:9259ea1c13d7 20...'</td>
</tr>
<tr>
<td>build_type</td>
<td>Value: None</td>
</tr>
<tr>
<td>build_types</td>
<td>Value: ['clean', 'help']</td>
</tr>
<tr>
<td>dryrun</td>
<td>Value: 0</td>
</tr>
<tr>
<td>AUTO</td>
<td>Value: 0</td>
</tr>
<tr>
<td>FORCE</td>
<td>Value: 1</td>
</tr>
<tr>
<td>CACHE</td>
<td>Value: 2</td>
</tr>
<tr>
<td>cache_mode</td>
<td>Value: 0</td>
</tr>
<tr>
<td>progress_display</td>
<td>Value: DisplayEngine()</td>
</tr>
<tr>
<td>SConfFS</td>
<td>Value: None</td>
</tr>
<tr>
<td>sconf_global</td>
<td>Value: None</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
</tbody>
</table>

### 18.3 Class SConfWarning

```python
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__(), __getattribute__(), __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>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>

18.4 Class SConfError

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.SConf.SConfError

Known Subclasses: SCons.SConf.ConfigureCacheError, SCons.SConf.ConfigureDryRunError
18.4.1 Methods

```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__(), __getattr__(), __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**

```
__new__()
```

**Inherited from exceptions.BaseException**

```
__delattr__(), __getattr__(target), __getstate__(target), __reduce__(target), __reduce_ex__(target), __repr__(target), __setattr__(target), __setstate__(target), __str__(target), __unicode__(target)
```

**Inherited from object**

```
__format__(target), __hash__(target), __reduce_ex__(target), __sizeof__(target), __subclasscheck__(target)
```

18.5.2 Properties

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

Raised when a use explicitly requested the cache feature, but the test is run the first time.

18.6.1 Methods

__init__(self, target)

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

Inherited from exceptions.Exception

__new__()  

Inherited from exceptions.BaseException

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

Inherited from object

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

18.6.2 Properties

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

object

SCons.Node.BuildInfoBase

SCons.Node.FS.FileBuildInfo

SCons.SConf.SConfBuildInfo

Special build info for targets of configure tests. Additional members are result (did the builder succeed last time?) and string, which contains messages of the original build phase.

18.7.1 Methods

```python
__init__(self)
```

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

Overrides: object.__init__ (inherited documentation)

```python
set__build__result(self, result, string)
```

Inherited from SCons.Node.FS.FileBuildInfo (Section 15.15)

convert_from_sconsign(), convert_to_sconsign(), format(), prepare_dependencies()

Inherited from SCons.Node.BuildInfoBase (Section 13.5)

__getstate__(), __setstate__(), merge()

Inherited from object

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

18.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
<td></td>
</tr>
<tr>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

Inherited from SCons.Node.BuildInfoBase (Section 13.5)

bact, bactsig, bdepends, bdependsigs, bimplicit, bimplicitsigs, bsources, bsourcesigs

Inherited from object
18.7.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>current_version_id</code></td>
<td>Inherited from <code>SCons.Node.FS.FileBuildInfo (Section 15.15)</code></td>
</tr>
</tbody>
</table>

18.8 Class Streamer

`object` ↘

`SCons.SConf.Streamer`

'Sniffer' for a file-like writable object. Similar to the unix tool tee.

18.8.1 Methods

```python
__init__(self, orig)
```

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

```python
write(self, str)
```

```python
writelines(self, lines)
```

```python
getvalue(self)
```

Return everything written to `orig` since the Streamer was created.

```python
flush(self)
```

*Inherited from object*

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

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

18.9 Class SConfBuildTask

object

SCons.Taskmaster.Task

SCons.Taskmaster.AlwaysTask

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)

Logs the original builder messages, given the SConfBuildInfo instance bi.
**failed(self)**

Default action when a task fails: stop the build.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using `Configure()`.Overrides: `SCons.Taskmaster.Task.failed` 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><code>Inherited from object</code></td>
<td></td>
</tr>
<tr>
<td><code>__class__</code></td>
<td></td>
</tr>
</tbody>
</table>
18.10 Class SConfBase

This is simply a class to represent a configure context. After creating a SConf object, you can call any tests. After finished with your tests, be sure to call the Finish() method, which returns the modified environment. Some words about caching: In most cases, it is not necessary to cache Test results explicitly. Instead, we use the scons dependency checking mechanism. For example, if one wants to compile a test program (SConf.TryLink), the compiler is only called, if the program dependencies have changed. However, if the program could not be compiled in a former SConf run, we need to explicitly cache this error.

18.10.1 Methods

```python
__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__

```python
Finish(self)
```

Call this method after finished with your tests: env = sconf.Finish()

```python
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>Function</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 <code>&lt;text&gt;</code> and optional source file extension <code>&lt;extension&gt;</code>, 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><em>Inherited from object</em></td>
<td></td>
</tr>
<tr>
<td><strong><strong>class</strong></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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>corupt_dblite_warning(filename)</td>
<td>Get_DataBase(dir)</td>
</tr>
<tr>
<td></td>
<td>Reset()</td>
</tr>
<tr>
<td></td>
<td>write()</td>
</tr>
<tr>
<td></td>
<td>File(name, dbm_module=None)</td>
</tr>
</tbody>
</table>

Reset global state. Used by unit tests that end up using SConsign multiple times to get a clean slate for each test.

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 rel_2.4.0:3365:9259ea1c13d7...'</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

```python
object
SCons.SConsign.SConsignEntry
```

Wrapper class for the generic entry in a .sconsign file. The Node subclass populates it with attributes as it pleases.

XXX As coded below, we do expect a `.binfo` attribute to be added, but we'll probably generalize this in the next refactorings.

19.3.1 Methods

```python
__init__(self)
```

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

Overrides:
object.__init__ (inherited documentation)

```python
convert_to_sconsign(self)
```

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

```python
__getstate__(self)
```

```python
__setstate__(self, state)
```

**Inherited from object**

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

19.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binfo</td>
<td></td>
</tr>
<tr>
<td>ninfo</td>
<td></td>
</tr>
</tbody>
</table>

**Inherited from object**

```python
__class__
```

19.3.3 Class Variables

224
### Name Description

| current_version_id | Value: 2 |

#### 19.4 Class Base

```
object __init__(self)
```

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

```
get_entry(self, filename)
```

Fetch the specified entry attribute.

```
set_entry(self, filename, obj)
```

Set the entry.

```
do_not_set_entry(self, filename, obj)
```

```
store_info(self, filename, node)
```

```
do_not_store_info(self, filename, node)
```

```
merge(self)
```

---

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.

### Known Subclasses

SCons.SConsign.DB, SCons.SConsign.Dir
19.4.2 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>

19.5 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.

19.5.1 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)
```

19.5.2 Properties

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__()
```
19.6 Class Dir

object

SCons.SConsign.Base

SCons.SConsign.Dir

Known Subclasses: SCons.SConsign.DirFile

19.6.1 Methods

```python
__init__(self, fp=None, dir=None)
```

fp - file pointer to read entries from  Overrides: object.__init__

Inherited from SCons.SConsign.Base (Section 19.4)

- do_not_set_entry(), do_not_store_info(), get_entry(), merge(), set_entry(), store_info()

Inherited from object

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

19.6.2 Properties

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

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

dir - the directory for the file  Overrides: object.__init__

```python
write(self, sync=1)
```

Write the .sconsign file to disk.

Try to write to a temporary file first, and rename it if we succeed. If we can’t write to the temporary file, it’s probably because the directory isn’t writable (and if so, how did we build anything in this directory, anyway?), so try to write directly to the .sconsign file as a backup. If we can’t rename, try to copy the temporary contents back to the .sconsign file. Either way, always try to remove the temporary file at the end.

_Inherited from SCons.SConsign.Base (Section 19.4)_

do_not_set_entry(), do_not_store_info(), get_entry(), merge(), set_entry(), store_info()

_Inherited from object_

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

19.7.2 Properties
### 19.8 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.

#### 19.8.1 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. 231)*
- **D**: SCons.Scanner.D  
  *(Section 22, p. 234)*
- **Dir**: SCons.Scanner.Dir  
  *(Section 23, p. 238)*
- **Fortran**: SCons.Scanner.Fortran  
  *(Section 24, p. 240)*
- **IDL**: SCons.Scanner.IDL  
  *(Section 25, p. 245)*
- **LaTeX**: SCons.Scanner.LaTeX  
  *(Section 26, p. 246)*
- **Prog**: SCons.Scanner.Prog  
  *(Section 27, p. 253)*
- **RC**: SCons.Scanner.RC  
  *(Section 28, p. 254)*

20.2 Functions

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

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>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: \n'src/engine/SCons/Scanner/<strong>init</strong>.py rel_2.4.0:3365:9259...'</td>
</tr>
</tbody>
</table>

*continued on next page*
### 20.4 Class FindPathDirs

object \[ \rightarrow \]
SCons.Scanner.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

___init___(self, variable)

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

___call___(self, env, dir=None, target=None, source=None, argument=None)

Inherited from object

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

#### 20.4.2 Properties

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

## 20.5 Class Base

object \[ \rightarrow \]
SCons.Scanner.Base

**Known Subclasses:** SCons.Scanner.Current, SCons.Scanner.Selector, SCons.Scanner.LaTeX.LaTeX
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__ (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 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)
__str__(self)

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

add_scanner(self, skey, scanner)

add_skey(self, skey)

Add a skey to the list of skeys

get_skeys(self, env=None)

path(self, env, dir=None, target=None, source=None)

recurse_nodes(self, nodes)

select(self, node)

Inherited from object

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

20.5.2 Properties

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

20.6 Class Selector

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 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)
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__, __getattribute__, __new__, __reduce__,
__reduce_ex__, __repr__, __setattr__, __sizeof__, __subclasshook__

20.6.2 Properties

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

20.7 Class Current

object

SCons.Scanner.Base

SCons.Scanner.Current

Known Subclasses: SCons.Scanner.Classic

A class for scanning files that are source files (have no builder) or are derived files and are current (which implies that they exist, either locally or in a repository).
20.7.1 Methods

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

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path_function.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the SConscript file that defined the primary target, list of target nodes, list of source nodes, and optional argument for this instance) and returns a tuple of the directories that can be searched for implicit dependency files. May also return a callable() which is called with no args and returns the tuple (supporting Bindable class).

'node_class' - the class of Nodes which this scan will return. If node_class is None, then this scanner will not enforce any Node conversion and will return the raw results from the underlying scanner function.

'node_factory' - the factory function to be called to translate the raw results returned by the scanner function into the expected node_class objects.

'scan_check' - a function to be called to first check whether this node really needs to be scanned.

'recursive' - specifies that this scanner should be invoked recursively on all of the implicit dependencies it returns (the canonical example being #include lines in C source files). May be a callable, which will be called to filter the list of nodes found to select a subset for recursive scanning (the canonical example being only recursively scanning subdirectories within a directory).

The scanner function’s first argument will be a Node that should be scanned for dependencies, the second argument will be an Environment object, the third argument will be the tuple of paths returned by the path_function, and the fourth argument will be the value passed into 'argument', and the returned list should contain the Nodes for all the direct dependencies of the file.

Examples:

```python
s = Scanner(my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function)
```

```python
s = Scanner(function = my_scanner_function, argument = 'foo')
```
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>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></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__`
Class ClassicCPP

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

20.9 Class ClassicCPP

object

SCons.Scanner.Base

SCons.Scanner.Current

SCons.Scanner.Classic

SCons.Scanner.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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>find_include</code></td>
<td><code>self, include, source_dir, path</code></td>
</tr>
<tr>
<td></td>
<td>Overrides: <code>SCons.Scanner.Classic.find__include</code></td>
</tr>
<tr>
<td><code>sort_key</code></td>
<td><code>self, include</code></td>
</tr>
<tr>
<td></td>
<td>Overrides: <code>SCons.Scanner.Classic.sort__key</code></td>
</tr>
</tbody>
</table>

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

- `__init__()`, `find__include_names()`, `scan()`

*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__()`

### 20.9.2 Properties

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

*Inherited from `object`*
21 Module SCons.Scanner.C

SCons.Scanner.C
This module implements the dependency scanner for C/C++ code.

21.1 Functions

```python
dictify_CPPDEFINES(env)
```

**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><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/C.py rel_2.4.0:3365:9259ea1c13d...'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>

21.3 Class SConsCPPScanner

SCons.cpp.PreProcessor -> SCons.Scanner.C.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__ (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 (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 ifndef(), 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>
<tbody>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

*Inherited from object*
21.4 Class SConsCPPScannerWrapper

The SCons wrapper around a cpp.py scanner.

This is the actual glue between the calling conventions of generic SCons scanners, and the (subclass of) cpp.py class that knows how to look for #include lines with reasonably real C-preprocessor-like evaluation of #if/#ifdef/#else/#elif lines.

21.4.1 Methods

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

```python
call__(self, node, env, path=())
```

```python
recurse_nodes(self, nodes)
```

```python
select(self, node)
```

Inherited from object

```python
_delattr_(), _format_(), _getattribute_(), _hash_(), _new_(),
_reduce_(), _reduce_ex_(), _repr_(), _setattr_(), _sizeof_(),
_str_(), _subclasshook_()
```

21.4.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
22 Module SCons.Scanner.D

SCons.Scanner.D
Scanner for the Digital Mars 'D' programming language.
Coded by Andy Friesen 17 Nov 2003

22.1 Functions

DScanner()

Return a prototype Scanner instance for scanning D source files

22.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: ‘src/engine/SCons/Scanner/D.py rel_2.4.0:3365:9259ea1c13d...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: ‘SCons.Scanner’</td>
</tr>
</tbody>
</table>

22.3 Class D

object

SCons.Scanner.Base

SCons.Scanner.Current

SCons.Scanner.Classic

SCons.Scanner.D.D
22.3.1 Methods

```python
__init__(self)
```

Construct a new scanner object given a scanner function.

'function' - a scanner function taking two or three arguments and returning a list of strings.

'name' - a name for identifying this scanner object.

'argument' - an optional argument that, if specified, will be passed to both the scanner function and the path_function.

'skeys' - an optional list argument that can be used to determine which scanner should be used for a given Node. In the case of File nodes, for example, the 'skeys' would be file suffixes.

'path_function' - a function that takes four or five arguments (a construction environment, Node for the directory containing the 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)

s = Scanner(function = my_scanner_function, argument = 'foo')
```

Overrides: object.__init__ (inherited documentation)
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(), getattr(), 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

```python
only_dirs(nodes)

DirScanner(**kw)

Return a prototype Scanner instance for scanning directories for on-disk files

DirEntryScanner(**kw)

Return a prototype Scanner instance for "scanning" directory Nodes for their in-memory entries

do_not_scan(k)

scan_on_disk(node, env, path=())

Scans a directory for on-disk files and directories therein.
Looking up the entries will add these to the in-memory Node tree representation of the file system, so all we have to do is just that and then call the in-memory scanning function.

scan_in_memory(node, env, path=())

'Scans" a Node.FS.Dir for its in-memory entries.
```

23.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/Dir.py rel_2.4.0:3365:9259ea1c1...</td>
</tr>
<tr>
<td>skip_entry</td>
<td>Value: {'.': 1, '..': 1, '.sconsign': 1, '.sconsign.bak': 1, '..s...</td>
</tr>
</tbody>
</table>

continued on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip_entry_list</td>
<td>Value: ['.','..','.sconsign','.sconsign.dblite','.sconsign.d...']</td>
</tr>
<tr>
<td><strong>package</strong>____</td>
<td>Value: 'SCons.Scanner'</td>
</tr>
<tr>
<td>skip</td>
<td>Value: '.sconsign.db'</td>
</tr>
</tbody>
</table>
24 Module SCons.Scanner.Fortran

SCons.Scanner.Fortran

This module implements the dependency scanner for Fortran code.

24.1 Functions

FortranScan(path_variable='FORTRANPATH')

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 rel_2.4.0:3365:9259e...'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>

24.3 Class F90Scanner

A Classic Scanner subclass for Fortran source files which takes into account both USE and INCLUDE statements. This scanner will work for both F77 and F90 (and beyond) compilers.

Currently, this scanner assumes that the include files do not contain USE statements. To enable the ability to deal with USE statements in include files, add logic right after the module names are found to loop over each include file, search for and locate each USE statement, and append each module name to the list of dependencies. Caching the search
results in a common dictionary somewhere so that the same include file is not searched multiple times would be a smart thing to do.
24.3.1 Methods

```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 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)
```
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></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
Module SCons.Scanner.IDL

SCons.Scanner.IDL

This module implements the dependency scanner for IDL (Interface Definition Language) files.

25.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLScan()</td>
<td>Return a prototype Scanner instance for scanning IDL source files</td>
</tr>
</tbody>
</table>

25.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/IDL.py rel_2.4.0:3365:9259ea1c1...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Scanner'</td>
</tr>
</tbody>
</table>
26 Module SCons.Scanner.LaTeX

This module implements the dependency scanner for LaTeX code.

26.1 Functions

```python
modify_env_var(env, var, abspath)
```

LaTeXScanner()

Return a prototype Scanner instance for scanning LaTeX source files when built with latex.

PDFLaTeXScanner()

Return a prototype Scanner instance for scanning LaTeX source files when built with pdflatex.

26.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/LaTeX.py rel_2.4.0:3365:9259ea1...</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
26.3.1 Methods

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

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

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

Inherited from object

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

26.3.2 Properties

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

26.4 Class LaTeX

```
object
```

```
SCons.Scanner.Base
```

```
SCons.Scanner.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
pdftex 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=())

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><strong>class</strong></td>
<td>Inherited from object</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: ['INDEXSTYLE', 'BIBINPUTS', 'TEXINPUTS', 'BSTINPUTS']</td>
</tr>
</tbody>
</table>
27 Module SCons.Scanner.Prog

27.1 Functions

**ProgramScanner(**kw*)**

Return a prototype Scanner instance for scanning executable files for static-lib dependencies

**scan(n**ode, env, libpath=())**

This scanner scans program files for static-library dependencies. It will search the LIBPATH environment variable for libraries specified in the LIBS variable, returning any files it finds as dependencies.

27.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: 'src/engine/SCons/Scanner/Prog.py rel_2.4.0:3365:9259ea1c...</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 rel_2.4.0:3365:9259ea1c13...</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. 264)*
- **Main**: SCons.Script
  *(Section 31, p. 267)*
- **SConscript**: SCons.Script.SConscript
  *(Section 32, p. 281)*

29.2 Functions

<table>
<thead>
<tr>
<th>HelpFunction(\textit{text})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables(\textit{files=[]}, \textit{args={}})</td>
</tr>
<tr>
<td>Options(\textit{files=[]}, \textit{args={}})</td>
</tr>
</tbody>
</table>

29.3 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: ‘src/engine/SCons/Script/<strong>init</strong>.py’ rel_2.4.0:3365:9259e...</td>
</tr>
<tr>
<td>start_time</td>
<td>Value: 1442869709.5</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: SCons.Defaults.Chmod</td>
</tr>
<tr>
<td>Copy</td>
<td>Value: SCons.Defaults.Copy</td>
</tr>
<tr>
<td>Delete</td>
<td>Value: SCons.Defaults.Delete</td>
</tr>
</tbody>
</table>

continued on next page
### Variables

<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>CSScan</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*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| CFile      | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| CXXFile    | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| CacheDir   | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| Clean      | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| DVI        | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| Decider    | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| Default    | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| Depends    | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| Dir        | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| EnsurePythonVersion | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| EnsureSConsVersion | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| Entry      | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| Execute    | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`
| Exit       | Value: `<SCons.Script.SConscript.DefaultEnvironmentCall object at...`

*continued on next page*
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>File</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>FindFile</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>FindInstalledFiles</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>FindSourceFiles</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Flatten</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>GetBuildPath</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>GetLaunchDir</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Glob</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Help</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Ignore</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Import</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Install</td>
<td>&lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>InstallAs</td>
<td>&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>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>MSVSP</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>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<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: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Split</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>StaticLibrary</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>StaticObject</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Tag</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Tar</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>TargetSignatures</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>TypeLibrary</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Value</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>VariantDir</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td>Zip</td>
<td>Value: &lt;SCons.Script.SConscript.DefaultEnvironmentCall object at...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Script'</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 object</td>
<td><strong>class</strong></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 UserList.UserList</td>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
</tr>
</tbody>
</table>
30 Module SCons.Script.Interactive

SCons interactive mode

30.1 Functions

```
interact(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:</td>
</tr>
<tr>
<td></td>
<td>'src/engine/SCons/Script/Interactive.py' rel_2.4.0:3365:92...</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
build [TARGETS]  Build the specified TARGETS and their dependencies. 'b' is a synonym.
clean [TARGETS]  Clean (remove) the specified TARGETS and their dependencies. 'c' is a synonym.
extit        Exit SCons interactive mode.
help [COMMAND]  Prints help for the specified COMMAND. 'h' and '?' are synonyms.
shell [COMMANDLINE]  Execute COMMANDLINE in a subshell. 'sh' and '!' are synonyms.
version       Prints SCons version information.
```
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 extit(inherited documentation)
```

```python
onecmd(self, line)

Interpret the argument as though it had been typed in response to the prompt.

This may be overridden, but should not normally need to be; see the precmd() and postcmd() methods for useful execution hooks. The return value is a flag indicating whether interpretation of commands by the interpreter should stop. Overrides: cmd.Cmd.onecmd extit(inherited documentation)
```

```python
do_build(self, argv)

build [TARGETS] Build the specified TARGETS and their dependencies. 'b' is a synonym.
```

```python
do_clean(self, argv)

clean [TARGETS] Clean (remove) the specified TARGETS and their dependencies. 'c' is a synonym.
```

```python
do_EOF(self, argv)
```
do_exit(self, argv)

```
exit Exit SCons interactive mode.
```

do_help(self, argv)

```
help [COMMAND] Prints help for the specified COMMAND. ‘h’ and ‘?’ are synonyms. Overrides: cmd.Cmd.do_help
```

do_shell(self, argv)

```
shell [COMMANDLINE] Execute COMMANDLINE in a subshell. ‘sh’ and ‘!’ are synonyms.
```

do_version(self, argv)

```
version Prints SCons version information.
```

**Inherited from cmd.Cmd**

```
cmdloop(), columnize(), complete(), complete_help(), completedefault(), complete_names(), emptyline(), get_names(), parseline(), postcmd(), postloop(), precmd(), preloop(), print_topics()
```

### 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', '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
```

281
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=6, releaselevel=...))
- python_version_deprecated(version=sys.version_info(major=2, minor=7, micro=6, 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 rel_2.4.0:3365:9259ea1c1...</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(0x7F3AB5FBB310)</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>

*continued on next page*
### 31.3 Class SConsPrintHelpException

```python
doctest
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>Inherited from exceptions.BaseException</td>
<td>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></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)

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

write(self, s)

erase_previous(self)

spinner(self, node)

string(self, node)

replace_string(self, node)

__call__(self, node)

Inherited from object

__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><strong>class</strong></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>

continued on next page
### Class BuildTask

<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>

#### 31.5 Class BuildTask

An SCons build task.

#### 31.5.1 Methods

**display**(self, message)

Hook to allow the calling interface to display a message.

This hook gets called as part of preparing a task for execution (that is, a Node to be built). As part of figuring out what Node should be built next, the 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)

**prepare**(self)

Called just before the task is executed.

This is mainly intended to give the target Nodes a chance to unlink underlying files and make all necessary directories before the Action is actually called to build the targets. Overrides: SCons.Taskmaster.Task.prepare extit(inherited documentation)

**needs__execute**(self)

Returns True (indicating this Task should be executed) if this Task’s target state indicates it needs executing, which has already been determined by an earlier up-to-date check. Overrides: SCons.Taskmaster.Task.needs__execute
execute(self)

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in prepare(), executed() or failed(). Overrides: SCons.Taskmaster.Task.execute (inherited documentation)

do_failed(self, status=2)

executed(self)

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node's callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node's state before deciding whether it was 'built', in which case we call the appropriate Node method. In any event, we always call 'visited()', which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node. Overrides: SCons.Taskmaster.Task.executed (inherited documentation)

failed(self)

Default action when a task fails: stop the build.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure(). Overrides: SCons.Taskmaster.Task.failed (inherited documentation)

postprocess(self)

Post-processes a task after it's been executed.

This examines all the targets just built (or not, we don't care if the build was successful, or even if there was no build because everything was up-to-date) to see if they have any waiting parent Nodes, or Nodes waiting on a common side effect, that can be put back on the candidates list. Overrides: SCons.Taskmaster.Task.postprocess (inherited documentation)
make_ready(self)

Make a task ready for execution  Overrides: SCons.Taskmaster.Task.make_ready

Inherited from SCons.Taskmaster.Task(Section 35.4)

__init__(), exc_clear(), exc_info(), exception_set(), executed_with_callbacks(),
executed_without_callbacks(), fail_continue(), fail_stop(), get_target(), make_ready_all(),
make_ready_current(), trace_message()

Inherited from object

__delattr__(), __format__(), __getattr__(), __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><strong>class</strong></td>
<td>Inherited from object</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(0x7F3AB5FBB310)</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>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| `fs_delete`          | 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()`.
| `show`               | 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) |
| `executed`           | Called when the task has been successfully executed and the Taskmaster instance doesn’t want to call the Node’s callback methods. Overrides: SCons.Taskmaster.Task.executed |
| `make_ready`         | 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. Overrides: SCons.Taskmaster.Task.make__ready |
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 exitt(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>Inherited from object</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 unlink underlying files and make all necessary directories before the Action is actually called to build the targets. Overrides: SCons.Taskmaster.Task.prepare extit(inherited documentation)

**execute**(self)

Called to execute the task.

This method is called from multiple threads in a parallel build, so only do thread safe stuff here. Do thread unsafe stuff in prepare(), executed() or failed(). Overrides: SCons.Taskmaster.Task.execute extit(inherited documentation)

**executed**(self)

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node’s callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node’s state before deciding whether it was "built", in which case we call the appropriate Node method. In any event, we always call 'visited()', which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node. Overrides: SCons.Taskmaster.Task.executed extit(inherited documentation)

*Inherited from SCons.Taskmaster.AlwaysTask*(Section 35.5)

needs_execute()

*Inherited from SCons.Taskmaster.Task*(Section 35.4)

__init__(), display(), exc_clear(), exc_info(), exception_set(), executed_with_callbacks(), executed_without_callbacks(), fail_continue(), fail_stop(), failed(), get_target(), make_ready(), make_ready_all(), make_ready_current(), postprocess(), trace_message()

*Inherited from object*

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

\[
\textbf{add\_local\_option}(\textit{self}, *\textit{args}, **\textit{kw})
\]

Inherited from object

\[
\text{\_delattr\_()}, \text{\_format\_()}, \text{\_getattribute\_()}, \text{\_hash\_()}, \text{\_init\_()}, \text{\_new\_()}, \text{\_reduce\_()}, \text{\_reduce\_ex\_()}, \text{\_repr\_()}, \text{\_setattr\_()}, \text{\_sizeof\_()}, \text{\_str\_()}, \text{\_subclasshook\_()}
\]

31.9.2 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>

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*

```python
__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)*

```python
__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

```
__init__(self, label)
do_append(self, label)
do_print(self)
```

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

```
__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>
32 Module SCons.Script.SConscript’

SCons.Script.SConscript

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

32.1 Functions

```python
get_calling_namespaces()
```

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

```python
compute_exports(exports)
```

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

```python
Return(*vars, **kw)
```

```python
SConscript_exception(file=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.

```python
annotate(node)
```

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

```python
Configure(*args, **kw)
```

```python
get_DefaultEnvironmentProxy()
```

296
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 rel_2.4.0:3365:925...</td>
</tr>
<tr>
<td>launch_dir</td>
<td>Value: '/home/bdbaddog/scons/as_scons'</td>
</tr>
<tr>
<td>GlobalDict</td>
<td>Value: None</td>
</tr>
<tr>
<td>global_exports</td>
<td>Value: {}</td>
</tr>
<tr>
<td>sconscript_chdir</td>
<td>Value: 1</td>
</tr>
<tr>
<td>call_stack</td>
<td>Value: []</td>
</tr>
<tr>
<td>stack_bottom</td>
<td>Value: '% Stack boTTom %'</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons.Script'</td>
</tr>
</tbody>
</table>

### 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__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()
Inherited from object

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

32.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td>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__()
32.5 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default(self, *targets)</td>
<td></td>
</tr>
<tr>
<td>EnsureSConsVersion(self, major, minor)</td>
<td>Exit abnormally if the SCons version is not late enough.</td>
</tr>
<tr>
<td>EnsurePythonVersion(self, major, minor)</td>
<td>Exit abnormally if the Python version is not late enough.</td>
</tr>
<tr>
<td>Exit(self, value=0)</td>
<td></td>
</tr>
<tr>
<td>Export(self, *vars, **kw)</td>
<td></td>
</tr>
<tr>
<td>GetLaunchDir(self)</td>
<td></td>
</tr>
<tr>
<td>GetOption(self, name)</td>
<td></td>
</tr>
</tbody>
</table>
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()
32.6 Class DefaultEnvironmentCall

A class that implements "global function" calls of Environment methods by fetching the specified method from the DefaultEnvironment’s class. Note that this uses an intermediate proxy class instead of calling the DefaultEnvironment method directly so that the proxy can override the subst() method and thereby prevent expansion of construction variables (since from the user’s point of view this was called as a global function, with no associated construction environment).

32.6.1 Methods

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

Overrides: object.__init__ (inherited documentation)

```python
__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><strong>class</strong></td>
<td>Inherited from object</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 rel_2.4.0:3365:9259ea1c13d7 2015...</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. Remove 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

```
object
    SCons.Util.Null
        SCons.Sig.MD5Null
```

33.2.1  Methods

```python
___repr__(self)
```

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

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

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

Inherited from object

__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

SCons.Util.Null

SCons.Sig.TimeStampNull

33.3.1 Methods

```python
___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><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
34 Module SCons.Subst

SCons.Subst
SCons string substitution.

34.1 Functions

SetAllowableExceptions(*excepts)
raise_exception(exception, target, s)
quote_spaces(arg)

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

escape_list(mylist, escape_func)

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

subst_dict(target, source)

Create a dictionary for substitution of special construction variables.

This translates the following special arguments:

target - the target (object or array of objects), used to generate the TARGET and TARGETS construction variables

source - the source (object or array of objects), used to generate the SOURCES and SOURCE construction variables
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.

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.

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:

```
env2 = 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 rel_2.4.0:3365:9259ea1c13d7 20...</td>
</tr>
<tr>
<td>AllowableExceptions</td>
<td>Value: (&lt;type 'exceptions.IndexError'&gt;, &lt;type 'exceptions.NameEr...</td>
</tr>
<tr>
<td>NullNodesList</td>
<td>Value: Null(0x7F3AB66D80D0)</td>
</tr>
</tbody>
</table>

continued on next page
34.3 Class Literal

object

SCons.Subst.Literal

A wrapper for a string. If you use this object wrapped around a string, then it will be interpreted as literal. When passed to the command interpreter, all special characters will be escaped.

34.3.1 Methods

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

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

escape(self, escape_func)

for_signature(self)

is_literal(self)

__eq__(self, other)

__neq__(self, other)
```

Inherited from object

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

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

```
__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__`

```
__str__(self)
```

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

```
escape(self, escape_func)
```

```
for_signature(self)
```

```
is_literal(self)
```

Inherited from `object`

```
__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><em><strong>class</strong></em></td>
<td></td>
</tr>
</tbody>
</table>

34.5 Class CmdStringHolder

This is a special class used to hold strings generated by scons_subst() and scons_subst_list(). It defines a special method escape(). When passed a function with an escape algorithm for a particular platform, it will return the contained string with the proper escape sequences inserted.

34.5.1 Methods

```python
__init__(self, cmd, literal=None)
```

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

```python
is_literal(self)
```
escape(self, escape_func, quote_func=<function quote_spaces at 0x7f3ab66ca5f0>)

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(), istitle(), isupper(), join(), ljust(), lower(), lstrip(), partition(), replace(), rfind(),
index(), rjust(), rpartition(), rsplit(), rstrip(), split(), splitlines(), startswith(),
strip(), swapcase(), title(), translate(), upper(), zfill()

Inherited from abc.Sequence

__iter__(), __reversed__()

Inherited from abc.Sized

__subclasshook__() 

Inherited from object

__delattr__(), __format__(), __getattr__(), __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 \texttt{abspath}, \texttt{.source}, etc. This way, we avoid creating those proxies just 'in case' someone is going to use $\texttt{TARGET}$ 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

\begin{verbatim}
__init__(self, list, func)
\end{verbatim}

\texttt{x.__init__(...)} initializes \texttt{x}; see \texttt{help(type(x))} for signature  \texttt{Overrides: object.__init__} exit(inherited documentation)

\textit{Inherited from object}

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

34.6.2 Properties

\begin{tabular}{|c|c|}
\hline
Name & Description \\
\hline
_Inherited from object_ & \\
__class__ & \\
\hline
\end{tabular}
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__(), __delslice__(), __eq__(), __ge__(), __gt__(), __iadd__(), __imul__(), __le__(), __len__(), __lt__(), __mul__(), __ne__(), __radd__(), __rmul__(), __setitem__(), __setslice__(), append(), count(), extend(), index(), insert(), pop(), remove(), reverse(), sort()

Inherited from __abcoll.Sequence
__iter__() , __reversed__() 

Inherited from __abcoll.Sized
__subclasshook__() 

Inherited from object
__delattr__() , __format__() , __getattribute__() , __new__() , __reduce__() , __reduce_ex__() , __setattr__() , __sizeof__() 

34.7.2 Properties

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

34.7.3 Class Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from UserList.UserList</td>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
</tr>
</tbody>
</table>
### 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 NLLWrapper. This class handles the different methods used to access an individual proxy Node, calling the NLLWrapper to create a proxy on demand.

#### 34.8.1 Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>init</strong></td>
<td><code>__init__(self, nl)</code></td>
<td>x.<strong>init</strong>(...) initializes x; see help(type(x)) for signature. Overrides: object.<strong>init</strong> (inherited documentation)</td>
</tr>
<tr>
<td><strong>getattr</strong></td>
<td><code>__getattr__(self, attr)</code></td>
<td></td>
</tr>
<tr>
<td><strong>str</strong></td>
<td><code>__str__(self)</code></td>
<td>str(x) Overrides: object.<strong>str</strong> (inherited documentation)</td>
</tr>
<tr>
<td><strong>repr</strong></td>
<td><code>__repr__(self)</code></td>
<td>repr(x) Overrides: object.<strong>repr</strong> (inherited documentation)</td>
</tr>
</tbody>
</table>

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><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
34.9 Class NullNodeList

object

SCons.Util.Null

SCons.Util.NullSeq

SCons.Subst_NullNodeList

34.9.1 Methods

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

Overrides: SCons.Util.Null.__call__

```python
__str__(self)
```

str(x) Overrides: object.__str__ (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__(), __getattribute__(), __init__(), __new__(), __nonzero__(), __repr__(),
__setattr__()
```

Inherited from object

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

34.9.2 Properties

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

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>dump_stats()</td>
</tr>
<tr>
<td>find_cycle(stack, visited)</td>
</tr>
</tbody>
</table>

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: <code>/src/engine/SCons/Taskmaster.py rel_2.4.0:3365:9259ea1c13...</code></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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td>Inherited from object</td>
</tr>
<tr>
<td><strong>delattr</strong></td>
<td></td>
</tr>
<tr>
<td><strong>format</strong></td>
<td></td>
</tr>
<tr>
<td><strong>getattribute</strong></td>
<td></td>
</tr>
<tr>
<td><strong>hash</strong></td>
<td></td>
</tr>
<tr>
<td><strong>new</strong></td>
<td></td>
</tr>
<tr>
<td><strong>reduce</strong></td>
<td></td>
</tr>
<tr>
<td><strong>reduce_ex</strong></td>
<td></td>
</tr>
<tr>
<td><strong>repr</strong></td>
<td></td>
</tr>
<tr>
<td><strong>setattr</strong></td>
<td></td>
</tr>
<tr>
<td><strong>sizeof</strong></td>
<td></td>
</tr>
<tr>
<td><strong>str</strong></td>
<td></td>
</tr>
<tr>
<td><strong>subclasshook</strong></td>
<td></td>
</tr>
</tbody>
</table>
35.4 Class Task

object → SCons.Taskmaster.Task

**Known Subclasses:** SCons.Taskmaster.AlwaysTask, SCons.Taskmaster.OutOfDateTask

Default SCons build engine task.

This controls the interaction of the actual building of node and the rest of the engine.

This is expected to handle all of the normally-customizable aspects of controlling a build, so any given application *should* be able to do what it wants by sub-classing this class and overriding methods as appropriate. If an application needs to customize something by sub-classing Taskmaster (or some other build engine class), we should first try to migrate that functionality into this class.

Note that it’s generally a good idea for sub-classes to call these methods explicitly to update state, etc., rather than roll their own interaction with Taskmaster from scratch.

35.4.1 Methods

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

Override:

```python
object.__init__ (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.
**Class Task**

**Module SCons.Taskmaster**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>prepare</strong>(self)</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.</td>
</tr>
<tr>
<td><strong>get_target</strong>(self)</td>
<td>Fetch the target being built or updated by this task.</td>
</tr>
<tr>
<td><strong>needs_execute</strong>(self)</td>
<td></td>
</tr>
<tr>
<td><strong>execute</strong>(self)</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><strong>executed_without_callbacks</strong>(self)</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><strong>executed_with_callbacks</strong>(self)</td>
<td>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 &quot;built&quot;, in which case we call the appropriate Node method. In any event, we always call 'visited()&quot;, 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.</td>
</tr>
</tbody>
</table>
### executed(self)

Called when the task has been successfully executed and the Taskmaster instance wants to call the Node's callback methods.

This may have been a do-nothing operation (to preserve build order), so we must check the node's state before deciding whether it was "built", in which case we call the appropriate Node method. In any event, we always call 'visited()’, which will handle any post-visit actions that must take place regardless of whether or not the target was an actual built target or a source Node.

### failed(self)

Default action when a task fails: stop the build.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().

### fail_stop(self)

Explicit stop-the-build failure.

This sets failure status on the target nodes and all of their dependent parent nodes.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().

### fail_continue(self)

Explicit continue-the-build failure.

This sets failure status on the target nodes and all of their dependent parent nodes.

Note: Although this function is normally invoked on nodes in the executing state, it might also be invoked on up-to-date nodes when using Configure().
**make_ready_all**(self)

Marks all targets in a task ready for execution.

This is used when the interface needs every target Node to be visited--the canonical example being the "scons -c" option.

**make_ready_current**(self)

Marks all targets in a task ready for execution if any target is not current.

This is the default behavior for building only what’s necessary.

**make_ready**(self)

Marks all targets in a task ready for execution if any target is not current.

This is the default behavior for building only what’s necessary.

**postprocess**(self)

Post-processes a task after it’s been executed.

This examines all the targets just built (or not, we don’t care if the build was successful, or even if there was no build because everything was up-to-date) to see if they have any waiting parent Nodes, or Nodes waiting on a common side effect, that can be put back on the candidates list.

**exc_info**(self)

Returns info about a recorded exception.
exc_clear(self)

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

exception_set(self, exception=None)

Records an exception to be raised at the appropriate time.
This also changes the 'exception_raise' attribute to point to the method that will, in fact

Inherited from object

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

```
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__(), 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__()
```

35.6.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inherited from object</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></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__ (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.
will_not_build(self, nodes, node_func=<function <lambda> at 0x7f3ab64109b0>)

Perform clean-up about nodes that will never be built. Invokes a user defined function on all of these nodes (including all of their parents).

stop(self)

Stops the current build completely.

cleanup(self)

Check for dependency cycles.

Inherited from object

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

35.7.2 Properties

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

SCons.Util
Various utility functions go here.

36.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dictify</strong> <em>(keys, values, result=</em>)</td>
<td>Check whether sequence str contains ANY of the items in set.</td>
</tr>
<tr>
<td><strong>rightmost_separator</strong> <em>(path, sep)</em></td>
<td></td>
</tr>
<tr>
<td><strong>containsAny</strong> <em>(str, set)</em></td>
<td></td>
</tr>
<tr>
<td><strong>containsAll</strong> <em>(str, set)</em></td>
<td>Check whether sequence str contains ALL of the items in set.</td>
</tr>
<tr>
<td><strong>containsOnly</strong> <em>(str, set)</em></td>
<td>Check whether sequence str contains ONLY items in set.</td>
</tr>
<tr>
<td><strong>splitext</strong> <em>(path)</em></td>
<td>Same as os.path.splitext() but faster.</td>
</tr>
<tr>
<td><strong>updrive</strong> <em>(path)</em></td>
<td>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.</td>
</tr>
</tbody>
</table>
**get_environment_var(varstr)**

Given a string, first determine if it looks like a reference to a single environment variable, like "$FOO" or "${FOO}". If so, return that variable with no decorations ("FOO"). If not, return None.

**render_tree(root, child_func, prune=0, margin=[0], visited={})**

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.

**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>,
StringTypes=(<type 'str'>, <type 'unicode'>, <class 'UserString.UserS...)

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

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

flatten(obj, isinstance=<built-in function isinstance>,
StringTypes=(<type 'str'>, <type 'unicode'>, <class 'UserString.UserS...),
SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>),
do_flatten=<function do_flatten at 0x7f3ab66bfcf8>)

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**

`flatten_sequence(sequence, isinstance=<built-in function isinstance>, StringTypes=(<type 'str'>, <type 'unicode'>), <class 'UserString.UserString'>, SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>), do_flatten=<function do_flatten at 0x7f3ab66bfaf8>)`

Flatten a sequence to a non-nested list.

Same as `flatten()`, but it does not handle the single scalar case. This is slightly more efficient when one knows that the sequence to flatten can not be a scalar.

**to_String**

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

**to_String_for_subst**

`to_String_for_subst(s, isinstance=<built-in function isinstance>, str=<type 'str'>, to_String=<function to_String at 0x7f3ab66bfaf8>, BaseStringTypes=(<type 'str'>, <type 'unicode'>), SequenceTypes=(<type 'list'>, <type 'tuple'>, <class 'UserList.UserList'>), UserString=<class 'UserString.UserString'>)`

**to_String_for_signature**

`to_String_for_signature(obj, to_String_for_subst=<function to_String_for_subst at 0x7f3ab66bfe8>, AttributeError=<type 'exceptions.AttributeError'>)`

**semi_deepcopy_dict**

`semi_deepcopy_dict(x, exclude=[])`

**semi_deepcopy**

`semi_deepcopy(x)`

**RegGetValue**

`RegGetValue(root, key)`

**RegOpenKeyEx**

`RegOpenKeyEx(root, key)`

**WhereIs**

`WhereIs(file, path=None, pathext=None, reject=[])`
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.

generic._native_path(path)

Transforms an absolute path into a native path for the system. Non-Cygwin version, just leave the path alone.
**Functions**

**Module SCons.Util**

- **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:
```
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.UserS...</td>
</tr>
<tr>
<td>BaseStringTypes</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><code>self</code></td>
</tr>
<tr>
<td><code>__str__</code></td>
<td><code>str(x)</code> Overrides: object.<strong>str</strong> extit(inherited documentation)</td>
</tr>
<tr>
<td><code>__iter__</code></td>
<td><code>self</code></td>
</tr>
<tr>
<td><code>__call__</code></td>
<td><code>self, *args, **kwargs</code></td>
</tr>
</tbody>
</table>
```
getattr__(self, name)
```

**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**

```
__reversed__() 
```

**Inherited from _abcoll.Sized**

```
__subclasshook__() 
```

**Inherited from object**

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

### 36.3.2 Properties

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

### 36.3.3 Class Variables

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

### 36.4 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>Inherited from object</td>
<td><strong>class</strong></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

```python
object
```

```python
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><strong>init</strong></td>
<td>Wrap an object as a Proxy object</td>
</tr>
<tr>
<td><strong>getattr</strong></td>
<td>Retrieve an attribute from the wrapped object. If the named attribute doesn’t exist, AttributeError is raised.</td>
</tr>
<tr>
<td>get</td>
<td>Retrieve the entire wrapped object</td>
</tr>
<tr>
<td><strong>cmp</strong></td>
<td></td>
</tr>
</tbody>
</table>

Inherited from object

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()</td>
</tr>
</tbody>
</table>

### 36.5.2 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>
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
__init__(self, attribute)
```

Overrides: object.__init__ (inherited documentation)

```python
__get__(self, obj, cls)
```

Inherited from object

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

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

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __replac

Inherited from object

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

36.7.2 Properties

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

36.8 Class WindowsError

object

exceptions.BaseException

exceptions.Exception

exceptions.StandardError

exceptions.EnvironmentError

exceptions.OSError

SCons.Util.WindowsError

36.8.1 Methods

Inherited from exceptions.OSError
Class WindowsError

__init__(), __new__()

Inherited from exceptions.EnvironmentError

__reduce__(), __str__()

Inherited from exceptions.BaseException

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

Inherited from object

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

36.8.2 Properties

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

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

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

__coerce__(self, other)
```
__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 object</td>
<td><strong>class</strong></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 UserList.UserList</td>
<td><strong>abstractmethods</strong>, <strong>hash</strong></td>
</tr>
</tbody>
</table>

36.10 Class OrderedDict

UserDict.UserDict

SCons.Util.OrderedDict

Known Subclasses: SCons.Util.Selector
## 36.10.1 Methods

- `__init__(self, dict=None)`
  Overrides: `UserDict.UserDict.__init__`

- `__delitem__(self, key)`
  Overrides: `UserDict.UserDict.__delitem__`

- `__setitem__(self, key, item)`
  Overrides: `UserDict.UserDict.__setitem__`

- `clear(self)`
  Overrides: `UserDict.UserDict.clear`

- `copy(self)`
  Overrides: `UserDict.UserDict.copy`

- `items(self)`
  Overrides: `UserDict.UserDict.items`

- `keys(self)`
  Overrides: `UserDict.UserDict.keys`

- `popitem(self)`
  Overrides: `UserDict.UserDict.popitem`

- `setdefault(self, key, failobj=None)`
  Overrides: `UserDict.UserDict.setdefault`

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

- `values(self)`
  Overrides: `UserDict.UserDict.values`

**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

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

Inherited from SCons.Util.OrderedDict(Section 36.10)

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

Inherited from UserDict.UserDict

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

36.11.2 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>

345
36.12 Class LogicalLines

```
object ->
SCons.Util.LogicalLines
```

### 36.12.1 Methods

**`__init__`**

```
__init__(self, fileobj)
```

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

**`readline`**

```
readline(self)
```

**`readlines`**

```
readlines(self)
```

**Inherited from object**

```
__delattr__, __format__, __getattribute__, __hash__, __new__,
__reduce__, __reduce_ex__, repr, getattr, setattr, sizeof, str,
subclasshook
```

### 36.12.2 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>
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__ extit(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.<strong>gt</strong></td>
</tr>
<tr>
<td><code>__ge__</code></td>
<td>(self, other)</td>
<td>UserList.UserList.<strong>ge</strong></td>
</tr>
<tr>
<td><code>__cmp__</code></td>
<td>(self, other)</td>
<td>UserList.UserList.<strong>cmp</strong></td>
</tr>
<tr>
<td><code>__len__</code></td>
<td>(self)</td>
<td>abcoll.Sized.<strong>len</strong></td>
</tr>
<tr>
<td><code>__getitem__</code></td>
<td>(self, i)</td>
<td>abcoll.Sequence.<strong>getitem</strong></td>
</tr>
<tr>
<td><code>__setitem__</code></td>
<td>(self, i, item)</td>
<td>abcoll.MutableSequence.<strong>setitem</strong></td>
</tr>
<tr>
<td><code>__getslice__</code></td>
<td>(self, i, j)</td>
<td>UserList.UserList.<strong>getslice</strong></td>
</tr>
<tr>
<td><code>__setslice__</code></td>
<td>(self, i, j, other)</td>
<td>UserList.UserList.<strong>setslice</strong></td>
</tr>
<tr>
<td><code>__add__</code></td>
<td>(self, other)</td>
<td>UserList.UserList.<strong>add</strong></td>
</tr>
<tr>
<td><code>__radd__</code></td>
<td>(self, other)</td>
<td>UserList.UserList.<strong>radd</strong></td>
</tr>
<tr>
<td><code>__iadd__</code></td>
<td>(self, other)</td>
<td>abcoll.MutableSequence.<strong>iadd</strong></td>
</tr>
<tr>
<td><code>__mul__</code></td>
<td>(self, other)</td>
<td>UserList.UserList.<strong>mul</strong></td>
</tr>
</tbody>
</table>
`__rmul__`(self, other)
Overrides: UserList.UserList.__rmul__

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

`append`(self, item)
append object to the end of the sequence  Overrides: _abcoll.MutableSequence.append extit(inherited documentation)

`insert`(self, i)
insert object before index  Overrides: _abcoll.MutableSequence.insert extit(inherited documentation)

`count`(self, item)
return number of occurrences of value  **Return Value**
  integer
Overrides: _abcoll.Sequence.count extit(inherited documentation)

`index`(self, item)
return first index of value. Raises ValueError if the value is not present.  **Return Value**
  integer
Overrides: _abcoll.Sequence.index extit(inherited documentation)

`reverse`(self)
reverse **IN PLACE**  Overrides: _abcoll.MutableSequence.reverse extit(inherited documentation)

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

`extend`(self, other)
extend sequence by appending elements from the iterable  Overrides: _abcoll.MutableSequence.extend extit(inherited documentation)
Inherited from UserList.UserList

__contains__(), __delitem__(), __delslice__(), __repr__(), pop(), remove()

Inherited from __abstract.Sequence

__iter__(), __reversed__()

Inherited from __abstract.Sized

__subclasshook__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(),
__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><strong>abstractmethods</strong>, <strong>hash</strong></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

<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>

36.15 Class Null

```python
object
    SCons.Util.Null
```

**Known Subclasses:** SCons.Executor_NullEnvironment, SCons.Sig_MD5Null, SCons.Sig_TimeStampNull, SCons.Util_NullSeq

Null objects always and reliably "do nothing."

36.15.1 Methods

```python
__new__(cls, *args, **kwargs)
```

*Return Value*

a new object with type S, a subtype of T

*Overrides:* object.__new__ extit(inherited documentation)
Class Null

Module SCons.Util

```python
__init__(self, *args, **kwargs)

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

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

```python
__repr__(self)

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

```python
__nonzero__(self)
```

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

```python
__setattr__(self, name, value)

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

```python
__delattr__(self, name)

x.__delattr__('name') <===> del x.name  Overrides: object.__delattr__ 
extit(inherited documentation)
```

**Inherited from object**

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

### 36.15.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
36.16 Class 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__(),
__repr__(), __setattr__()

Inherited from object

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

36.16.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>class</strong></td>
<td>Inherited from object</td>
</tr>
</tbody>
</table>
37  Package SCons.Variables

gine.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. 343)*
- **EnumVariable** *(Section ??, p. ??)*
- **EnumVariable**: engine.SCons.Variables.EnumVariable *(Section 39, p. 344)*
- **ListVariable** *(Section ??, p. ??)*
- **ListVariable**: engine.SCons.Variables.ListVariable *(Section 40, p. 346)*
- **PackageVariable** *(Section ??, p. ??)*
- **PackageVariable**: engine.SCons.Variables.PackageVariable *(Section 41, p. 347)*
- **PathVariable** *(Section ??, p. ??)*
- **PathVariable**: SCons.Variables.PathVariable *(Section 42, p. 348)*

37.2  Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>revision</strong></td>
<td>Value: <code>src/engine/SCons/Variables/__init__.py</code></td>
</tr>
<tr>
<td></td>
<td>rel_2.4.0:3365:92...</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: <code>'SCons.Variables'</code></td>
</tr>
</tbody>
</table>

37.3  Class Variables

```
object
```

SCons.Variables.Variables
37.3.1 Methods

___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__

keys(self)

Returns the keywords for the options

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(self, *optlist)

Add a list of options.

Each list element is a tuple/list of arguments to be passed on to the underlying method for adding options.

Example:

```python
opt.AddVariables(
    ('debug', '', 0),
    ('CC', 'The C compiler'),
    ('VALIDATE', 'An option for testing validation', 'notset',
     validator, None),
)
```

Update(self, env, args=None)

Update an environment with the option variables.

env - the environment to update.

UnknownVariables(self)

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

Save(self, filename, env)

Saves all the options in the given file. This file can then be used to load the options next run. This can be used to create an option cache file.

filename - Name of the file to save into env - the environment get the option values from
GenerateHelpText(self, env, sort=None)

Generate the help text for the options.

env - an environment that is used to get the current values of the options.

FormatVariableHelpText(self, env, key, help, default, actual, aliases=[])"
38  Module SCons.Variables.BoolVariable

engine.SCons.Variables.BoolVariable

This file defines the option type for SCons implementing true/false values.

Usage example:

    opts = Variables()
    opts.Add(BoolVariable('embedded', 'build for an embedded system', 0))
    ...
    if env['embedded'] == 1:
        ...

38.1  Functions

<table>
<thead>
<tr>
<th>BoolVariable(key, help, default)</th>
</tr>
</thead>
</table>

The input parameters describe a boolean option, thus they are returned with the correct converter and validator appended. The 'help' text will by appended by '(yes|no) to show the valid valued. The result is usable for input to opts.Add().
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

```
EnumVariable(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).
40  **Module SCons.Variables.ListVariable**

engine.SCons.Variables.ListVariable

This file defines the option type for SCons implementing 'lists'.

A 'list' option may either be 'all', 'none' or a list of names separated by comma. After the option has been processed, the option value holds either the named list elements, all list elements or no list elements at all.

Usage example:

```python
list_of_libs = Split('x11 gl qt ical')

opts = Variables()
opts.Add(ListVariable('shared',
                  'libraries to build as shared libraries',
                  'all',
                  elems = list_of_libs))

...  
for lib in list_of_libs:
    if lib in env['shared']:
        env.SharedObject(...) 
    else:
        env.Object(...)
```

40.1  **Functions**

<table>
<thead>
<tr>
<th>ListVariable(key, help, default, names, map={})</th>
</tr>
</thead>
</table>

The input parameters describe a 'package list' option, thus they are returned with the correct converter and validator appended. The result is usable for input to opts.Add().

A 'package list' option may either be 'all', 'none' or a list of package names (separated by space).
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 existence)

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

<table>
<thead>
<tr>
<th>PackageVariable(key, help, default, searchfunc=None)</th>
</tr>
</thead>
</table>

The input parameters describe a 'package list' option, thus they are returned with the correct converter and validator appended. The result is usable for input to opts.Add().

A 'package list' option may either be 'all', 'none' or a list of package names (seperated by space).
Module 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:

```python
Examples:
  prefix=/usr/local

opts = Variables()

opts = Variables()
opts.Add(PathVariable('qtdir',
                   'where the root of Qt is installed',
                   qtdir, PathIsDir))
opts.Add(PathVariable('qt_includes',
                   'where the Qt includes are installed',
                   '$qtdir/includes', PathIsDirCreate))
opts.Add(PathVariable('qt_libraries',
                   ```
'where the Qt library is installed',
'${qtdir/lib}')

42.1 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathVariable</td>
<td>Value: <code>SCons.Variables.PathVariable</code></td>
</tr>
</tbody>
</table>
Module SCons.Warnings

This file implements the warnings framework for SCons.

43.1 Functions

\begin{verbatim}
suppressWarningClass(clazz)

Suppresses all warnings that are of type clazz or derived from clazz.

enableWarningClass(clazz)

Enables all warnings that are of type clazz or derived from clazz.

warningAsException(flag=1)

Turn warnings into exceptions. Returns the old value of the flag.

warn(clazz, *args)
\end{verbatim}
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>
</table>
| __revision__   | Value: 'src/engine/SCons/Warnings.py
|                | rel_2.4.0:3365:9259ea1c13d7... |
| __package__    | Value: 'SCons'          |

43.3 Class Warning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

Known Subclasses: SCons.SConf.SConfWarning, SCons.Warnings.CacheWriteErrorWarning,
43.3.1 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__()`

43.3.2 Properties

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

43.4 Class `WarningOnByDefault`

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

43.4.1 Methods

*Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__reduce_ex__()`, `__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><em>Inherited from <code>exceptions.BaseException</code></em> args, message</td>
<td></td>
</tr>
<tr>
<td><em>Inherited from <code>object</code></em> <strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

43.5 Class `TargetNotBuiltWarning`

```
object
  └── exceptions.BaseException
      └── exceptions.Exception
          └── SCons.Errors.UserError
              └── SCons.Warnings.Warning
                  └── SCons.Warnings.TargetNotBuiltWarning
```

43.5.1 Methods

*Inherited from `exceptions.Exception`*

`__init__()`, `__new__()`
**Class CacheWriteErrorWarning**

**Inherited from exceptions.BaseException**

\_delattr\_(), \_getattribute\_(), \_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>_class_</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\_(), \_getattribute\_(), \_getitem\_(), \_getslice\_(), \_reduce\_(), \_repr\_(), \_setattr\_(), \_setstate\_(), \_str\_(), \_unicode\_()
Class CorruptSConsignWarning

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__() , __getattribute__() , __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>args, message</td>
</tr>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></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__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __reduce_ex__(), __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>
### 43.9 Class DevelopmentVersionWarning

**object**

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.DevelopmentVersionWarning

#### 43.9.1 Methods

*Inherited from exceptions.Exception*

__init__(), __new__()

*Inherited from exceptions.BaseException*

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

*Inherited from object*

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

#### 43.9.2 Properties

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

*continued on next page*
<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>

43.10 Class DuplicateEnvironmentWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.DuplicateEnvironmentWarning

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>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>
43.11  Class FutureReservedVariableWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.FutureReservedVariableWarning

43.11.1  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__()`

43.11.2  Properties

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

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.LinkWarning

**Known Subclasses:** SCons.Warnings.FortranCxxMixWarning

43.12.1 Methods

*Inherited from exceptions.Exception*

```python
__init__(), __new__()
```

*Inherited from exceptions.BaseException*

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

*Inherited from object*

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

43.12.2 Properties

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

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.MisleadingKeywordsWarning

43.13.1 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__()  

43.13.2 Properties

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

| Inherited from object | __class__ |
### 43.14 Class MissingSConscriptWarning

```
object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.MissingSConscriptWarning
```

#### 43.14.1 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__()  
```

#### 43.14.2 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>
<tr>
<td>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.15 Class NoMD5ModuleWarning

object —

exceptions.BaseException —

exceptions.Exception —

SCons.Errors.UserError —

SCons.Warnings.Warning —

SCons.Warnings.WarningOnByDefault —

SCons.Warnings.NoMD5ModuleWarning

43.15.1 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__()

43.15.2 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 NoMetaclassSupportWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.NoMetaclassSupportWarning

43.16.1 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__()

43.16.2 Properties

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

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.NoObjectCountWarning

43.17.1 Methods

_Inherited from exceptions.Exception_

__init__(), __new__() 

_Inherited from exceptions.BaseException_

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

_Inherited from object_

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

43.17.2 Properties

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

Inherited from exceptions.BaseException

__init__(), __new__()  

Inherited from exceptions.Exception

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

Inherited from object

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

43.18.2 Properties

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

Inherited from exceptions.Exception

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

Inherited from exceptions.BaseException

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

Inherited from object

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

43.19.2  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>
</tbody>
</table>

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

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.StackSizeWarning

43.20.1  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__()

43.20.2  Properties

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

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.VisualCMissingWarning

43.21.1 Methods

*Inherited from exceptions.Exception*

```python
__init__(), __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__()
```

43.21.2 Properties

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

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.VisualVersionMismatch

43.22.1 Methods

_Inherited from exceptions.Exception_

__init__(), __new__()

_Inherited from exceptions.BaseException_

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

_Inherited from object_

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

43.22.2 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>
43.23 Class VisualStudioMissingWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.VisualStudioMissingWarning

43.23.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

_delattr__(), _getattribute__(), _getitem__(), _getslice__(), _reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

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

43.23.2 Properties

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

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.WarningOnByDefault

SCons.Warnings.LinkWarning

SCons.Warnings.FortranCxxMixWarning

43.24.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

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

Inherited from object

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

43.24.2 Properties

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

387
43.25  Class FutureDeprecatedWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.FutureDeprecatedWarning

Known Subclasses: SCons.Warnings.DeprecatedSourceCodeWarning

43.25.1  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__()

43.25.2  Properties

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

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning


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><em>Inherited from exceptions.BaseException</em></td>
<td></td>
</tr>
<tr>
<td>args, message</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>

<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>
43.27 Class MandatoryDeprecatedWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.MandatoryDeprecatedWarning


43.27.1 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__()

43.27.2 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>Inherited from object</td>
<td><strong>class</strong></td>
</tr>
</tbody>
</table>
43.28 Class PythonVersionWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.PythonVersionWarning

43.28.1 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__()

43.28.2 Properties

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

<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>
### 43.29 Class DeprecatedSourceCodeWarning

```
```

#### 43.29.1 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__()
```

#### 43.29.2 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 DeprecatedBuildDirWarning

object\

exceptions.BaseException\

exceptions.Exception\

SCons.Errors.UserError\

SCons.Warnings.Warning\

SCons.Warnings.DeprecatedWarning\

SCons.Warnings.DeprecatedBuildDirWarning

43.30.1  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__()  

43.30.2  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>
43.31 Class TaskmasterNeedsExecuteWarning

object

exceptions.BaseException

exceptions.Exception

SCons.Errors.UserError

SCons.Warnings.Warning

SCons.Warnings.DeprecatedWarning

SCons.Warnings.TaskmasterNeedsExecuteWarning

43.31.1 Methods

Inherited from exceptions.Exception

__init__(), __new__() 

Inherited from exceptions.BaseException

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

Inherited from object

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

43.31.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inherited from exceptions.BaseException</td>
<td></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>
43.32 Class DeprecatedCopyWarning

Inherited from exceptions.BaseException

__init__(), __new__()  

Inherited from exceptions.Exception

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

Inherited from object

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

43.32.2 Properties

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

```
object
```

```
extceptions.BaseException
```

```
extceptions.Exception
```

```
SCons.Errors.UserError
```

```
SCons.Warnings.Warning
```

```
SCons.Warnings.DeprecatedWarning
```

```
SCons.Warnings.MandatoryDeprecatedWarning
```

```
SCons.Warnings.DeprecatedOptionsWarning
```

43.33.1 Methods

**Inherited from exceptions.Exception**

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

**Inherited from exceptions.BaseException**

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

**Inherited from object**

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

43.33.2 Properties

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


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>
43.35  Class DeprecatedTargetSignaturesWarning

object  

exceptions.BaseException  

exceptions.Exception  

SCons.Errors.UserError  

SCons.Warnings.Warning  

SCons.Warnings.DeprecatedWarning  

SCons.Warnings.MandatoryDeprecatedWarning  

SCons.Warnings.DeprecatedTargetSignaturesWarning

43.35.1  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__()

43.35.2  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>
</tbody>
</table>

| Inherited from object |
| __class__ |
43.36 Class DeprecatedDebugOptionsWarning

object

exceptions.BaseException

    exceptions.Exception

        SCons.Errors.UserError

            SCons.Warnings.Warning

                SCons.Warnings.DeprecatedWarning

                    SCons.Warnings.MandatoryDeprecatedWarning

                        SCons.Warnings.DeprecatedDebugOptionsWarning

43.36.1 Methods

_Inherited from exceptions.Exception_

    __init__(), __new__()  

_Inherited from exceptions.BaseException_

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

_Inherited from object_

    __format__(), __hash__(), reduce_ex(), __sizeof__(), __subclasshook__()  

43.36.2 Properties

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


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>
</table>
43.38  Class DeprecatedBuilderKeywordsWarning

Inherited from exceptions.BaseException

__init__(), __new__()

Inherited from exceptions.Exception

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

Inherited from object

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

43.38.2  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>
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>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>doc</strong></td>
<td>Value: ...</td>
</tr>
<tr>
<td>cpp_lines_dict</td>
<td>Value: {{'define'}: \s+([-A-Za-z][-A-Za-z0-9_]<em>)((([^)]</em>)))(^[^]*^[^])?...</td>
</tr>
<tr>
<td>Table</td>
<td>Value: {'define': re.compile(r'\s+([-A-Za-z][-A-Za-z0-9_]<em>)((([^)]</em>)))(^[^]*^[^])?...</td>
</tr>
<tr>
<td>e</td>
<td>Value: ^\s*#<em>\s</em>(elif</td>
</tr>
<tr>
<td>CPP_Expression</td>
<td>Value: re.compile(r'(?m)\s*#*(elif</td>
</tr>
<tr>
<td>CPP_to_Python_Ops_Dict</td>
<td>Value: {'\r': '', '!': ' != ', '&amp;&amp;': ' and ', '||': ' or ', ':?': ' if'</td>
</tr>
<tr>
<td>CPP_to_Python_Ops_Expression</td>
<td>Value: re.compile(r'(?m)\s*#*(elif</td>
</tr>
<tr>
<td>CPP_to_Python_Eval_List</td>
<td>Value: [[re.compile(r'defined\s+(\w+)'), '', '\1' in <strong>dict</strong>], [],...</td>
</tr>
<tr>
<td>line_continuations</td>
<td>Value: re.compile(r'\r?\n')</td>
</tr>
<tr>
<td>function_name</td>
<td>Value: re.compile(r'(\S+)(([^])]*))')</td>
</tr>
<tr>
<td>function_arg_separator</td>
<td>Value: re.compile(r'(\S+)(([^])]*))')</td>
</tr>
<tr>
<td><strong>package</strong></td>
<td>Value: 'SCons'</td>
</tr>
<tr>
<td>x</td>
<td>Value: 'if'</td>
</tr>
</tbody>
</table>

402
44.3 Class FunctionEvaluator

```
object ├── SCons.cpp.FunctionEvaluator
```

Handles delayed evaluation of a #define function call.

44.3.1 Methods

```
___init___(self, name, args, expansion)
```

Squirrels away the arguments and expansion value of a #define macro function for later evaluation when we must actually expand a value that uses it.

 Overrides: object.__init__

```
___call___(self, *values)
```

Evaluates the expansion of a #define macro function called with the specified values.

**Inherited from object**

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

44.3.2 Properties

```

<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>
```

44.4 Class PreProcessor

```
object └── SCons.cpp.PreProcessor
```

**Known Subclasses:** SCons.cpp.DumbPreProcessor, SCons.Scanner.C.SConsCPPScanner
The main workhorse class for handling C pre-processing.

### 44.4.1 Methods

**__call__**(self, file)

Pre-processes a file.
This is the main public entry point.

**__init__**(self, current=‘.‘, cpppath=(), dict={}, all=0)

Overrides: object.__init__

**all__include**(self, t)

**do__define**(self, t)

Default handling of a #define line.

**do__elif**(self, t)

Default handling of a #elif line.

**do__else**(self, t)

Default handling of a #else line.

**do__endif**(self, t)

Default handling of a #endif line.
do_if(self, t)

Default handling of a #if line.

do_ifdef(self, t)

Default handling of a #ifdef line.

do ifndef(self, t)

Default handling of a #ifndef line.

do_import(self, t)

Default handling of a #import line.

do_include(self, t)

Default handling of a #include line.

do_include_next(self, t)

Default handling of a #include line.

do nothing(self, t)

Null method for when we explicitly want the action for a specific preprocessor directive to do nothing.

do undef(self, t)

Default handling of a #undef line.
**eval_expression**(*self, t*)

Evaluates a C preprocessor expression.
This is done by converting it to a Python equivalent and eval()ing it in the C
preprocessor namespace we use to track #define values.

**finalize_result**(*self, fname*)

**find_include_file**(*self, t*)

Finds the #include file for a given preprocessor tuple.

**initialize_result**(*self, fname*)

**process_contents**(*self, contents, fname=None*)

Pre-processes a file contents.
This is the main internal entry point.

**read_file**(*self, file*)

**resolve_include**(*self, t*)

Resolve a tuple-ized #include line.
This handles recursive expansion of values without "" or <>
surrounding the name until an initial " or < is found, to handle
#include FILE
where FILE is a #define somewhere else.

**restore**(*self*)

Pops the previous dispatch table off the stack and makes it the current one.
### save

```
save(self)
```

Pushes the current dispatch table on the stack and re-initializes the current dispatch table to the default.

### scons_current_file

```
scons_current_file(self, t)
```

### start_handling_includes

```
start_handling_includes(self, t=None)
```

Causes the PreProcessor object to start processing #import, #include and #include_next lines.

This method will be called when a #if, #ifdef, #ifndef or #elif evaluates True, or when we reach the #else in a #if, #ifdef, #ifndef or #elif block where a condition already evaluated False.

### stop_handling_includes

```
stop_handling_includes(self, t=None)
```

Causes the PreProcessor object to stop processing #import, #include and #include_next lines.

This method will be called when a #if, #ifdef, #ifndef or #elif evaluates False, or when we reach the #else in a #if, #ifdef, #ifndef or #elif block where a condition already evaluated True.

### tupleize

```
tupleize(self, contents)
```

This method 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__()
### 44.5 Class DumbPreProcessor

A preprocessor that ignores all `#if/#elif/#else/#endif` directives and just reports back all of the `#include` files (like the classic SCons scanner did).

This is functionally equivalent to using a regular expression to find all of the `#include` lines, only slower. It exists mainly as an example of how the main PreProcessor class can be sub-classed to tailor its behavior.

#### 44.5.1 Methods

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

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

Overrides:

object.__init__ (inherited 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><strong>class</strong></strong></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Inherited from object</td>
<td></td>
</tr>
<tr>
<td><strong>class</strong></td>
<td></td>
</tr>
</tbody>
</table>
45 Module SCons.dblite

45.1 Functions

```python
corruption_warning(filename)
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

```python
object ---- SCons.dblite.dblite
```

45.3.1 Methods

```python
__init__(self, file_base_name, flag, mode)
```

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

```python
close(self)
```

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

45.3.2 Properties

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

SCons.exitfuncs
Register functions which are executed when SCons exits for any reason.

46.1 Functions

**register**(*func, *targs, **kargs*)

register a function to be executed upon normal program termination

*func* - function to be called at exit
*targs* - optional arguments to pass to func
**kargs** - optional keyword arguments to pass to func

46.2 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| __revision__ | Value: 'src/engine/SCons/exitfunecs.py
rel_2.4.0:3365:9259ea1c13d... |
| __package__ | Value: ’SCons’                                   |
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), 28–29
    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.CacheDir *(module), 30–32
  SCons.Conftest *(module), 33–37
  SCons.cpp *(module), 387–394
  SCons.dbLite *(module), 395–396
    SCons.dbLite.corruption_warning *(function), 395
    SCons.dbLite.dbLite *(class), 395–396
    SCons.dbLite.is_string *(function), 395
    SCons.dbLite.open *(function), 395
    SCons.dbLite.unicode *(function), 395
  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.Diagnostics *(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 *(function), 45
    SCons.Environment.NoSubstitutionProxy *(function), 45
    SCons.Environment.OverrideEnvironment *(class), 61–64
    SCons.Environment.SubstitutionEnvironment
SCons.Variables.PathVariable (module), 348–349
SCons.Variables.Variables (class), 339–342
SCons.Warnings (module), 350–386
SCons.Action.Action (function), 6
SCons.Action.ActionBase (class), 6–7
    SCons.Action.ActionBase.__add__ (method), 7
    SCons.Action.ActionBase.__cmp__ (method), 7
    SCons.Action.ActionBase.__radd__ (method), 7
    SCons.Action.ActionBase.genstring (method), 7
    SCons.ActionEvent.get_contents (method), 7
    SCons.ActionEvent.get_targets (method), 7
    SCons.ActionEvent.get_varlist (method), 7
    SCons.ActionEvent.no_batch_key (method), 7
    SCons.ActionEvent.process (method), 8
SCons.ActionEvent.strfunction
    SCons.ActionEvent.get_implicit_deps (method), 8
    SCons.ActionEvent.get_presig (method), 8
    SCons.ActionEvent.CommandAction.execute (method), 8
    SCons.ActionEvent.CommandAction.get_presig (method), 9
    SCons.ActionEvent.CommandAction.get_presig (method), 10
SCons.ActionEvent.CommandGeneratorAction (class), 9–11
    SCons.ActionEvent.CommandGeneratorAction.execute (method), 11
    SCons.ActionEvent.CommandGeneratorAction.get_implicit_deps (method), 11
    SCons.ActionEvent.CommandGeneratorAction.get_presig (method), 11
    SCons.ActionEvent.CommandGeneratorAction.execute (method), 11
SCons.ActionEvent.default_exitstatfunc (function), 6
SCons.ActionEvent.F unctionAction (class), 12–14
    SCons.ActionEvent.F unctionAction.function_name (method), 13
    SCons.ActionEvent.F unctionAction.get_implicit_deps (method), 13
SCons.ActionEvent.get_default_ENV (function), 6
SCons.ActionEvent.LazyAction (class), 11–12
    SCons.ActionEvent.LazyAction.get_parent_class (method), 11
    SCons.ActionEvent.LazyAction.get_implicit_deps (method), 11
SCons.ActionEvent.rfile (function), 6
SCons.ActionEvent.ListAction (class), 14–15
    SCons.ActionEvent.ListAction.execute (method), 15
    SCons.ActionEvent.ListAction.get_implicit_deps (method), 15
SCons.ActionEvent.remove_set_lineno_codes (function), 6

416
SCons.CacheDir.CacheDir (class), 30–32
SCons.CacheDir.CacheDir.CacheDebug (method), 31
SCons.CacheDir.CacheDir.cache_path (method), 31
SCons.CacheDir.CacheDir.is_enabled (method), 31
SCons.CacheDir.CacheDir.is_readonly (method), 31
SCons.CacheDir.CacheDir.push (method), 31
SCons.CacheDir.CacheDir.push_if_forced (method), 31
SCons.CacheDir.CacheDir.retrieve (method), 31
SCons.CacheDir.CachePushFunc (function), 30
SCons.CacheDir.CacheRetrieveFunc (function), 30
SCons.CacheDir.CacheRetrieveString (function), 30
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.CheckSHCC (function), 33
SCons.Conftest.CheckSHCXX (function), 34
SCons.Conftest.CheckType (function), 34
SCons.Conftest.CheckTypeSize (function), 35
SCons.cpp.CPP_to_Python (function), 387
SCons.cpp.CPP_to_Python_Ops_Sub (function), 387
SCons.cpp.DumbPreProcessor (class), 393–394
SCons.cpp.FunctionEvaluator (class), 387–388
  SCons.cpp.FunctionEvaluator.__call__ (method), 388
SCons.cpp.PreProcessor (class), 388–393
  SCons.cpp.PreProcessor.__call__ (method), 389
  SCons.cpp.PreProcessor.all__include (method), 389
  SCons.cpp.PreProcessor.do__define (method), 389
  SCons.cpp.PreProcessor.do__elif (method), 389
  SCons.cpp.PreProcessor.do__else (method), 389
  SCons.cpp.PreProcessor.do__endif (method), 389
  SCons.cpp.PreProcessor.do__if (method), 389
  SCons.cpp.PreProcessor.do__ifdef (method), 390
  SCons.cpp.PreProcessor.do__ifndef (method), 390
  SCons.cpp.PreProcessor.do__import (method), 390
  SCons.cpp.PreProcessor.do__include (method), 390
  SCons.cpp.PreProcessor.do__nothing (method), 390
  SCons.cpp.PreProcessor.do__undef (method), 390
  SCons.cpp.PreProcessor.eval_expression (method), 390
  SCons.cpp.PreProcessor.finalize_result (method), 391
  SCons.cpp.PreProcessor.find__include_file (method), 391
  SCons.cpp.PreProcessor.initialize_result (method), 391
  SCons.cpp.PreProcessor.process__contents (method), 391
  SCons.cpp.PreProcessor.read__file (method), 391
  SCons.cpp.PreProcessor.resolve__include (method), 391
  SCons.cpp.PreProcessor.restore (method), 391
  SCons.cpp.PreProcessor.save (method), 391
  SCons.cpp.PreProcessor.scons__current__file (method), 392
  SCons.cpp.PreProcessor.start__handling__includes (method), 392
  SCons.cpp.PreProcessor.stop__handling__includes
SCons.cpp.PreProcessor.tupleize (method), 392
SCons.Defaults.chmod_func (function), 392
SCons.Defaults.chmod_strfunc (function), 40
SCons.Defaults.copy_func (function), 40
SCons.Defaults.DefaultEnvironment (function), 40
SCons.Defaults.delete_func (function), 41
SCons.Defaults.delete_strfunc (function), 41
SCons.Defaults.get_paths_str (function), 40
SCons.Defaults.mkdir_func (function), 41
SCons.Defaults.move_func (function), 41
SCons.Defaults.NullCmdGenerator (class), 42
SCons.Defaults.NullCmdGenerator.__call__ (method), 43
SCons.Defaults.processDefines (function), 41
SCons.Defaults.SharedFlagChecker (function), 40
SCons.Defaults.SharedObjectEmitter (function), 40
SCons.Defaults.StaticObjectEmitter (function), 40
SCons.Defaults.touch_func (function), 41
SCons.Defaults.Variable_Method_Caller (class), 43–44
SCons.Defaults.Variable_Method_Caller.__call__ (method), 44
SCons.Errors.BuildError (class), 74–76
SCons.Errors.convert_to_BuildError (function), 74
SCons.Errors.EnvironmentError (class), 79
SCons.Errors.ExplicitExit (class), 80–81
SCons.Errors.InternalError (class), 76–77
SCons.Errors.MSVCError (class), 79–80
SCons.Errors.StopError (class), 78–79
SCons.Errors.UserError (class), 77–78
SCons.Node.Annotate (function), 108
SCons.Node.BuildInfoBase (class), 113–114
SCons.Node.BuildInfoBase.__getstate__ (method), 113
SCons.Node.BuildInfoBase.__setstate__ (method), 113
SCons.Node.BuildInfoBase.merge (method), 113
SCons.Node.changed_since_last_build_alias (function), 110
SCons.Node.changed_since_last_build_entry (function), 110
SCons.Node.changed_since_last_build_node (function), 109
SCons.Node.changed_since_last_build_python (function), 110
SCons.Node.classname (function), 108
SCons.Node.decide_source (function), 110
SCons.Node.decide_target (function), 110
SCons.Node.exists_always (function), 108
SCons.Node.exists_base (function), 109
SCons.Node.exists_entry (function), 109
SCons.Node.exists_file (function), 109
SCons.Node.exists_none (function), 108
SCons.Node.get_children (function), 110
SCons.Node.get_contents_dir (function), 109
SCons.Node.get_contents_entry (function), 109
SCons.Node.get_contents_file (function), 109
SCons.Node.ignore_cycle (function), 110
SCons.Node.is_derived_node (function), 108
SCons.Node.is_derived_node (function), 108
SCons.Node.is_derived_node (function), 108
SCons.Node.is_derivative (function), 108
SCons.Node.Node (class), 114–128
SCons.Node.Node.add_dependency (method), 115
SCons.Node.Node.add_ignore (method), 115
SCons.Node.Node.add_prerequisite (method), 115
SCons.Node.Node.add_to_implicit (method), 115
SCons.Node.Node.add_to_waiting_parents (method), 115
SCons.Node.Node.add_to_waiting_s_e (method), 116
SCons.Node.Node.alter_targets (method), 116
SCons.Node.Node.build (method), 116
SCons.Node.Node.children_are_up_to_date (method), 117
SCons.Node.Node.clear (method), 117
SCons.Node.Node.clear_memoized_values (method), 118
SCons.Node.Node.del_binfo (method), 118
SCons.Node.Node.disambiguate (method), 118
SCons.Node.Node.env_set (method), 118
SCons.Node.Node.executor_cleanup (method), 118
SCons.Node.Node.exists (method), 118
SCons.Node.Node.get_abspath (method), 118
SCons.Node.Node.get_contents (method), 119
SCons.Node.Node.get_build_env (method), 119
SCons.Node.Node.get_build_scanner_path (method), 119
SCons.Node.Node.get_builder (method), 119
SCons.Node.Node.get_csig (method), 119
SCons.Node.Node.get_string (method), 121
SCons.Node.Node.get_target_scanner (method), 121
SCons.Node.Node.getTag (method), 115
SCons.Node.Node.get_executor (method), 120
SCons.Node.Node.get_found_includes (method), 120
SCons.Node.Node.get_implicit_deps (method), 120
SCons.Node.Node.get_ninfo (method), 120
SCons.Node.Node.get_source_scanner (method), 120
SCons.Node.Node.get_state (method), 120
SCons.Node.Node.get_stored_implicit (method), 121
SCons.Node.Node.get_stored_info (method), 121
SCons.Node.Node.get_subst_proxy (method), 121
SCons.Node.Node.is_up_to_date (method), 122
SCons.Node.Node.is_derived (method), 122
SCons.Node.Node.is_literal (method), 122
SCons.Node.Node.make_ready (method), 122
SCons.Node.Node.missing (method), 123
SCons.Node.Node.push_to_cache (method), 123
SCons.Node.Node.retrieve_from_cache (method), 125
SCons.Node.Node.rexists (method), 125
SCons.Node.Node.scan (method), 125
SCons.Node.Node.scanner_key (method), 125
SCons.Node.Node.select_scanner (method), 125
SCons.Node.Node.set_always_build (method), 125
SCons.Node.Node.set_executor (method), 126
SCons.Node.Node.set_noclean (method), 126
SCons.Node.Node.set_precious (method), 126
SCons.Node.Node.set_state (method), 126
SCons.Node.NodeInfoBase (class), 111–113
SCons.Node.NodeInfoBase.__getstate__ (method), 112
SCons.Node.NodeInfoBase.__setstate__ (method), 112
SCons.Node.NodeInfoBase.convert (method), 112
SCons.Node.NodeInfoBase.format (method), 112
SCons.Node.NodeInfoBase.merge (method), 112
SCons.Node.rexists_base (function), 109
SCons.Node.rexists_node (function), 109
SCons.Node.rexists_none (function), 109
SCons.Node.store_info_file (function), 110
SCons.Node.store_info_pass (function), 110
SCons.Node.target_from_source_base (function), 109
SCons.Node.target_from_source_none (function), 109
SCons.Node.Walker (class), 129–130
SCons.Node.Walker.get_next (method), 130
SCons.Node.Walker.is_done (method), 130
SCons.Scanner.Dir.DirScanner (function), 238
SCons.Scanner.Dir.do_not_scan (function), 238
SCons.Scanner.Dir.only_dirs (function), 238
SCons.Scanner.Dir.scan_in_memory (function), 238
SCons.Scanner.Dir.scan_on_disk (function), 238
SCons.Scanner.Interactive.interact (function), 264
SCons.Scanner.Interactive.SConsInteractiveCmd (class), 264–266
SCons.Scanner.Interactive.SConsInteractiveCmd.do_build (method), 265
SCons.Scanner.Interactive.SConsInteractiveCmd.do_clean (method), 265
SCons.Scanner.Interactive.SConsInteractiveCmd.do_EOF (method), 265
SCons.Scanner.Interactive.SConsInteractiveCmd.do_exit (method), 265
SCons.Scanner.Interactive.SConsInteractiveCmd.do_shell (method), 266
SCons.Scanner.Interactive.SConsInteractiveCmd.do_version (method), 266
SCons.Subst.CmdStringHolder (class), 294–295
SCons.Subst.CmdStringHolder.escape (method), 294
SCons.Subst.CmdStringHolder.is_literal (method), 294
SCons.Subst.escape_list (function), 290
SCons.Subst.Literal (class), 292–293
SCons.Subst.Literal.__eq__ (method), 292
SCons.Subst.Literal.__neq__ (method), 292
SCons.Subst.Literal (class), 292–293
<table>
<thead>
<tr>
<th>Function/Class</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCons.Util.NullSeq.<strong>getitem</strong> (method)</td>
<td>338</td>
</tr>
<tr>
<td>SCons.Util.NullSeq.<strong>iter</strong> (method)</td>
<td>338</td>
</tr>
<tr>
<td>SCons.Util.NullSeq.<strong>len</strong> (method)</td>
<td>338</td>
</tr>
<tr>
<td>SCons.Util.NullSeq.<strong>setitem</strong> (method)</td>
<td>338</td>
</tr>
<tr>
<td>SCons.Util.OrderedDict</td>
<td>328–330</td>
</tr>
<tr>
<td>SCons.Util.PrependPath (function)</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Util.print_tree (function)</td>
<td>313</td>
</tr>
<tr>
<td>SCons.Util.Proxy (class)</td>
<td>322–324</td>
</tr>
<tr>
<td>SCons.Util.Proxy.<strong>cmp</strong> (method)</td>
<td>323</td>
</tr>
<tr>
<td>SCons.Util.Proxy.<strong>getattr</strong> (method)</td>
<td>323</td>
</tr>
<tr>
<td>SCons.Util.Proxy.get (method)</td>
<td>323</td>
</tr>
<tr>
<td>SCons.Util.RegGetValueme (function)</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Util.RegOpenKeyEx (function)</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Util.RenameFunction (function)</td>
<td>318</td>
</tr>
<tr>
<td>SCons.Util.rightmost_separator (function)</td>
<td>312</td>
</tr>
<tr>
<td>SCons.Util.Selector (class)</td>
<td>330</td>
</tr>
<tr>
<td>SCons.Util.Selector.<strong>call</strong> (method)</td>
<td>330</td>
</tr>
<tr>
<td>SCons.Util.semi_deepcopy (function)</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Util.semi_deepcopy_dict (function)</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Util.Split (function)</td>
<td>316</td>
</tr>
<tr>
<td>SCons.Util.splitext (function)</td>
<td>312</td>
</tr>
<tr>
<td>SCons.Util.to_String (function)</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Util.to_String_for_signature (function)</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Util.to_String_for_subst (function)</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Util.Unbuffered (class)</td>
<td>335–336</td>
</tr>
<tr>
<td>SCons.Util.Unbuffered.<strong>getattr</strong> (method)</td>
<td>336</td>
</tr>
<tr>
<td>SCons.Util.Unbuffered.write (method)</td>
<td>336</td>
</tr>
<tr>
<td>SCons.Util.unique (function)</td>
<td>317</td>
</tr>
<tr>
<td>SCons.Util.UniqueList (class)</td>
<td>331–335</td>
</tr>
<tr>
<td>SCons.Util.uniquer (function)</td>
<td>317</td>
</tr>
<tr>
<td>SCons.Util.uniquer_hashables (function)</td>
<td>317</td>
</tr>
<tr>
<td>SCons.Util.updrive (function)</td>
<td>312</td>
</tr>
<tr>
<td>SCons.Util.Whereis (function)</td>
<td>315</td>
</tr>
<tr>
<td>SCons.Util.WindowsError (class)</td>
<td>325–326</td>
</tr>
<tr>
<td>SCons.Warnings.CacheWriteErrorWarning (class)</td>
<td>361–362</td>
</tr>
<tr>
<td>SCons.Warnings.CorruptSConsignWarning (class)</td>
<td>354–355</td>
</tr>
<tr>
<td>SCons.Warnings.DependencyWarning (class)</td>
<td>355–356</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedBuildDirWarning (class)</td>
<td>356–357</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedBuilderKeywordsWarning (class)</td>
<td>377–378</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedCopyWarning (class)</td>
<td>379–380</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedDebugOptionsWarning (class)</td>
<td>383–384</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedOptionsWarning (class)</td>
<td>380–381</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedSigModuleWarning (class)</td>
<td>384–385</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedSourceCodeWarning (class)</td>
<td>376–377</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedSourceSignaturesWarning (class)</td>
<td>381–382</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedTargetSignaturesWarning (class)</td>
<td>382–383</td>
</tr>
<tr>
<td>SCons.Warnings.DeprecatedWarning (class)</td>
<td>373–374</td>
</tr>
<tr>
<td>SCons.Warnings.DevelopmentVersionWarning (class)</td>
<td>357–358</td>
</tr>
<tr>
<td>SCons.Warnings.DuplicateEnvironmentWarning (class)</td>
<td>358–359</td>
</tr>
<tr>
<td>SCons.Warnings.enableWarningClass (function)</td>
<td>350</td>
</tr>
<tr>
<td>SCons.Warnings.FortranCxxMixWarning (class)</td>
<td>371–372</td>
</tr>
<tr>
<td>SCons.Warnings.FutureDeprecationWarning (class)</td>
<td>372–373</td>
</tr>
<tr>
<td>SCons.Warnings.FutureReservedVariableWarning (class)</td>
<td>359</td>
</tr>
<tr>
<td>SCons.Warnings.LinkWarning (class)</td>
<td>359–360</td>
</tr>
<tr>
<td>SCons.Warnings.MandatoryDeprecatedWarning (class)</td>
<td>374–375</td>
</tr>
<tr>
<td>SCons.Warnings.MisleadingKeywordsWarning (class)</td>
<td>360–361</td>
</tr>
<tr>
<td>SCons.Warnings.MissingSConscriptWarning (class)</td>
<td>361–362</td>
</tr>
</tbody>
</table>
SCons.Warnings.NoMD5ModuleWarning (class), 362–363
SCons.Warnings.NoMetaclassSupportWarning (class), 363–364
SCons.Warnings.NoObjectCountWarning (class), 364–365
SCons.Warnings.NoParallelSupportWarning (class), 365–366
SCons.Warnings.process_warn_strings (function), 350
SCons.Warnings.PythonVersionWarning (class), 375–376
SCons.Warnings.ReservedVariableWarning (class), 366–367
SCons.Warnings.StackSizeWarning (class), 367–368
SCons.Warnings.suppressWarningClass (function), 350
SCons.Warnings.TargetNotBuiltWarning (class), 353–354
SCons.Warnings.TaskmasterNeedsExecuteWarning (class), 378–379
SCons.Warnings.VisualStudioMissingWarning (class), 370–371
SCons.Warnings.VisualVersionMismatch (class), 369–370
SCons.Warnings.warn (function), 350
SCons.Warnings.Warning (class), 351–352
SCons.Warnings.warningAsException (function), 350
SCons.Warnings.WarningOnByDefault (class), 352–353