

Process

: Object

Runtime environment for the virtual machine and interpreter.

Description

A Process is the runtime environment for the virtual machine and interpreter. It has a subclass named [Main](#) which is where you should override the methods of Process. There are two methods of interest. One is named `startup` and is called after the class library has been compiled. The other is named `run` and is called when the user chooses the Run menu command.

Class Methods

Process.`tailCallOptimize`

Process.`tailCallOptimize` = bool

Get or set tail call optimization. The default is on. Setting this to `false` can help with debugging by including intermediate levels in an error backtrace.

Inherited class methods

14 methods from Object ► [show](#)

Undocumented class methods

Process.`elapsedTime`

Process.`monotonicClockTime`

Instance Methods

.`nowExecutingPath`

.`nowExecutingPath` = value

Usage: `thisProcess.nowExecutingPath`

Returns the full path to the file containing the code that is currently executing *interactively* in the interpreter. Usually this is the current document. If the code block executes another file on disk, using [String: -load](#) or [String: -loadPaths](#), `nowExecutingPath` will be the location of the executed file.

`nowExecutingPath` is valid only for interactive code, i.e., code files with a `.scd` extension. It does not apply to class definitions (`.sc`). For that, use `thisMethod.filenameSymbol` or `this.class.filenameSymbol`.

The method is supported in various environments, including the SuperCollider IDE, Neovim (SCNvim), Emacs (scel), VSCode/VSCodium, and the command line interface (CLI). In unsupported editors, it will return `nil`.

See [Examples](#) for various uses of `thisProcess.nowExecutingPath` with [CmdPeriod](#), [ServerBoot](#) and [ServerTree](#), [Routine](#) and [Task](#).

WARNING: `nowExecutingPath` has a corresponding setter method, `nowExecutingPath_`, for internal use only by the interpreter. Do not call the setter method!

.startup

called after the class library has been compiled. Override this in class [Main](#) to do whatever you want.

.run

called when the user chooses the Run menu command. Override this in class [Main](#) to do whatever you want.

.mainThread

The top-level [Thread](#), i.e the [parent](#) of all other Threads. This instance of Thread always exists and is created with the Process when SuperCollider starts.

Discussion:

All SuperCollider code initially runs in the context of the main Thread:

- Code evaluated in code editor
- Code evaluated on command line
- Tasks scheduled on any [Clock](#)
- Functions evaluated in response to incoming OSC and MIDI messages

This means that `thisThread` will always initially point to the main Thread. However, when some code starts a [Routine](#), the Routine becomes the current Thread, with the main Thread as its parent.

Inherited instance methods

403 methods from [Object](#) ► [show](#)

Undocumented instance methods

.addDocument

From extension in [/Users/prko/Dropbox/prko/___myDocs/Writings/Making Sound using Open Sources/mixed/dev - Bleeding](#)

[edge/SuperCollider.app/Contents/Resources/SCClassLibrary/Common/GUI/PlusGUI/Core/KernelPlusGUI.sc](#)

.archiveAsCompileString

.argv

.getCurrentSelection

.interpretCmdLine

.interpretPrintCmdLine

.interpretPrintSelectedText

.interpreter

.methodReferences

.methodTemplates

.openCodeFile

.openWinCodeFile

.prSchedulerQueue

.shallowCopy

.showHelp

.shutdown

.stop

.tick

Examples

Example 1. Comparison of the path of the evaluated code block in a saved SCD, the loaded SCD and the function in the loaded SCD:

If a code ("fileMain.scd" in the example code below) executes another file on disk ("fileForLoad.scd" in the example code below) using `String: -load` or `String: -loadPaths`, `thisProcess.nowExecutingPath` will be the location of the executed file ("fileForLoad.scd" in the example code below). If a function containing `thisProcess.nowExecutingPath` is called (the function defined in "fileForLoad.scd" in the example code below), `thisProcess.nowExecutingPath` will return the path of the document containing the function call ("fileMain.scd" in the example code below).

Steps:

1. Preparation:

1. Create an SCD file called fileMain.scd in your home folder using the following block of code:

```
(
  ("thisProcess.nowExecutingPath:").postln;
  ("  in fileMain.scd:                " + thisProcess.nowExecutingPath).postln;
  ~test = (thisProcess.nowExecutingPath.dirname ++ "fileForLoad.scd");
  ~test.load.testFunction;
  'fileMain.scd tasks finished'.postln;
)
```

2. Create an SCD file called fileForLoad.scd in your home folder using the following block of code:

```
("  in fileForLoad.scd:                " + thisProcess.nowExecutingPath).postln;
(
  testFunction: {
    ("  in the testFunction in fileForLoad.scd:" +
    thisProcess.nowExecutingPath).postln
  }
)
```

2. Execute SCD file:

- In SC-IDE or other editors:

1. Create a new SCD file (you do not need to save it for this step), then evaluate the following code:

```
"~/fileMain.scd".standardizePath.open0S;
// Note:
// - .openDocument only works in SC-IDE.
// - .open0S will work in other editors and
//   the system default application will open the SCD file.
```

2. Post window returns:

```
-> /Users/_Your_Account_Folder_/fileMain.scd
thisProcess.nowExecutingPath:
  in fileMain.scd:                /Users/_Your_Account_Folder_/fileMain.scd
  in fileForLoad.scd:
/Users/_Your_Account_Folder_/fileForLoad.scd
  in the testFunction in fileForLoad.scd: /Users/_Your_Account_Folder_/fileMain.scd
fileMain.scd tasks finished
-> fileMain.scd tasks finished
```

- In command line interface (CLI):

```
// Linux:
sclang '/home/_Your_Account_Folder_/fileMain.scd'
```

```
// macOS (/Applications/SuperCollider.app might be changed to properly):
/Applications/SuperCollider.app/Contents/MacOS/sclang ~/fileMain.scd

// Windows (C:\Program Files\SuperCollider_dev\ should be changed to properly):
"C:\Program Files\SuperCollider_dev\sclang.exe" %userprofile%\fileMain.scd

sclang returns to the terminal window (command prompt window on Windows):

... (messages are truncated) ...
*** Welcome to SuperCollider 3.13.0. *** For help press F1.
thisProcess.nowExecutingPath:
    in fileMain.scd: C:\Users\_Your_Account_Folder_\fileMain.scd
    in fileForLoad.scd:
C:\Users\_Your_Account_Folder_\fileForLoad.scd
    in the testFunction in fileForLoad.scd: C:\Users\_Your_Account_Folder_\fileMain.scd
fileMain.scd tasks finished
```

Example 2. `CmdPeriod`, `ServerBoot` and `ServerTree`:

WARNING: The returned path depends on the existence of the `startup.scd` file when the interpreter boots.

When used in `CmdPeriod`, `ServerBoot` and `ServerTree`, it will return `nil` unless `startup.scd` exists.

```
(
  CmdPeriod.add {
    var path = thisProcess.nowExecutingPath;
    ("-- CmdPeriod's thisProcess.nowExecutingPath:" + path).postln
  };
  ServerBoot.add {
    var path = thisProcess.nowExecutingPath;
    ("-- ServerBoot's thisProcess.nowExecutingPath:" + path).postln
  };
  ServerTree.add {
    var path = thisProcess.nowExecutingPath;
    ("-- ServerTree's thisProcess.nowExecutingPath:" + path).postln
  };
  s.reboot
)
```

- If `startup.scd` exists:

- When evaluating the code above, post window returns

```
... (messages are truncated) ...
localhost: keeping clientID (0) as confirmed by server process.
-- ServerBoot's thisProcess.nowExecutingPath:
/Users/_Your_Account_Folder_/Library/Application Support/SuperCollider/startup.scd
-- ServerTree's thisProcess.nowExecutingPath:
/Users/_Your_Account_Folder_/Library/Application Support/SuperCollider/startup.scd
Shared memory server interface initialized
```

- When pressing CMD/control + . after evaluating the code above:
 - CmdPeriod's `thisProcess.nowExecutingPath`: `/Users/_Your_Account_Folder_/Library/Application Support/SuperCollider/startup.scd`
 - ServerTree's `thisProcess.nowExecutingPath`:
`/Users/_Your_Account_Folder_/Library/Application Support/SuperCollider/startup.scd`
- If there is no startup.scd:
 - When evaluating the code above, post window returns


```
... (messages are truncated) ...
localhost: keeping clientID (0) as confirmed by server process.
– ServerBoot's thisProcess.nowExecutingPath: nil
– ServerTree's thisProcess.nowExecutingPath: nil
Shared memory server interface initialized
```
 - When pressing CMD/control + . after evaluating the code above:
 - CmdPeriod's `thisProcess.nowExecutingPath`: `nil`
 - ServerTree's `thisProcess.nowExecutingPath`: `nil`

Example 3. In [Routine](#) and [Task](#):

The following examples return the full path of the SCD file where the evaluated code block is located.

WARNING: Each code should be copied and pasted into an SCD file that has already been saved in a folder.

- [Routine](#): `play`:

```
(
  Routine {
    var path = thisProcess.nowExecutingPath;
    ("– thisProcess.nowExecutingPath in path Routine:" + path).postln
  }.play;

  r {
    var path = thisProcess.nowExecutingPath;
    ("– thisProcess.nowExecutingPath in path r:      " + path).postln
  }.play
)
```

- [Function](#): `fork`:

```
(
  fork {
    var path = thisProcess.nowExecutingPath;
    ("– fork's thisProcess.nowExecutingPath:" + path).postln
  }
)
```

- [Server](#): `waitForBoot`:

```
(
  s.waitForBoot {
    var path = thisProcess.nowExecutingPath;
    ("– .waitForBoot's thisProcess.nowExecutingPath:" + path).postln
  }
)
```

```
}  
)
```

- **Server: doWhenBooted:**

```
(  
s.doWhenBooted {  
    var path = thisProcess.nowExecutingPath;  
    ("– .doWhenBooted's thisProcess.nowExecutingPath:" + path).postln  
};  
s.reboot;  
)
```

- **Task:**

```
(  
Task {  
    var path = thisProcess.nowExecutingPath;  
    ("– Task's thisProcess.nowExecutingPath:" + path).postln  
}.start;  
  
Task {  
    var path = thisProcess.nowExecutingPath;  
    ("– Task's thisProcess.nowExecutingPath:" + path).postln  
}.play  
)
```