MiscSubprocess

Inherits From:	NSObject
Declared In:	<misckit/MiscSubprocess.h>

Class Description

MiscSubprocess facilitates the management of UNIX processes within a NEXTSTEP application.   Methods are provided for the creation, pausing, termination, and communication with the underlying UNIX process.

The subprocess can be run synchronously, meaning that the method call to start the subprocess will not return until the subprocess exits, or asynchronously, in which case the method call returns immediately.

The UNIX subprocess communicates with its parent application through delegation.   Six delegate methods provide notification of pending output, standard error output,   termination,   and errors from the UNIX subprocess.   It is the responsibility of the MiscSubprocess instantiator to implement any desired delegate methods and decide what to do with the resulting data.

In addition to providing a controlled interface to standard UNIX utilities (i.e., `ls`, `find`, `man`, `rdist`), the MiscSubprocess can also provide, on request, the environment necessary for UNIX processes requiring pseudo terminal (or pty) support.   Some UNIX applications that require pty support include `ftp`, `gdb`, `sh`, `csh`, `kermit`, and `tip`.

NOTE: This MiscSubprocess is only usable with NEXTSTEP, as the asynchronous operations are still implemented with DPS functions.

Instance Variables

id **delegate**;
NSMutableDictionary ***environment**;
FILE ***fpToChild**;
FILE ***fpFromChild**;
int **stdoutFromChild**;
int **stderrFromChild**;
int **childPid**;
BOOL **paused**;
BOOL **running**;
BOOL **usePtys**;
BOOL **asynchronous**;
BOOL **stdoutIsDone**;
BOOL **stderrIsDone**;

delegate	MiscSubprocess' delegate object.
environment	NSMutableDictionary that holds the child's environment.

fpToChild	Pipe for sending data to the child process' stdin.
fpFromChild	Pipe for receiving data from the child process' stdout.
stdoutFromChild	File descriptor of pipe for receiving data from the child process' stdout.
stderrFromChild	File descriptor of pipe for receiving data from the child process' stderr.
childPid	PID of the child process.
paused	True if child process is paused.
running	True if the child process is running.
usePtys	True if pty support should be used by degenerate **execute:** methods.
asynchronous	True if degenerate **execute:** methods should run an asynchronous subprocess.
stdoutIsDone	True if standard output has finished.
stderrIsDone	True if standard error has finished.

Method Types

Initializing a MiscSubprocess:	- init - init: - init:asynchronously: - init:withPtys: - init:withPtys:asynchronously: - init:keepEnvironment: - init:keepEnvironment:asynchronously:

	- init:keepEnvironment:withPtys:
	- init:keepEnvironment:withPtys:asynchronously:
	- init:withDelegate:
	- init:withDelegate:asynchronously:
	- init:withDelegate:withPtys:
	- init:withDelegate:withPtys:asynchronously:
	- init:withDelegate:keepEnvironment:
	- init:withDelegate:keepEnvironment:asynchronously:
	- init:withDelegate:keepEnvironment:withPtys:
	- init:withDelegate:keepEnvironment:withPtys:asynchronously:
Sending Data to Child:	- send:
	- send:withNewline:
	- terminateInput
Obtaining Information About Child:	- environment
	- isPaused
	- isRunning
	- pid
Controlling Child Process:	- execChild:
	- execute:
	- execute:asynchronously:
	- execute:withPtys:
	- execute:withPtys:asynchronously:
	- pause:
	- resume:
	- terminate:
Delegate methods:	- delegate
	- setDelegate:

- subprocess:done::
- subprocess:error:
- subprocess:output:
- subprocess:outputData:
- subprocess:stderrOutput:
- subprocess:stderrOutputData:

Instance Methods

delegate

- **delegate**

Returns the MiscSubprocess' delegate object, the object responsible for handling errors and receiving output from the child process.

See also: ±setDelegate: and the delegate methods at the end of this document.

environment

- (NSMutableDictionary *)**environment**

Returns an NSMutableDictionary object which contains the environment that will be set up for the child process when it is started. If you wish to alter the environment, add to or modify the NSMutableDictionary returned by this method.

See also: ± execute:withPtys:asynchronously:

execChild:

- **execChild:**(NSString *)*aString*

This is used internally by MiscSubprocess to actually start the child process, using the UNIX execle(). If you wish to override how a child process is started, you should override this method and not ±**execute:**.

See also: ±environment and **±execute:withPtys:asynchronously:**

execute:
execute:withPtys:
execute:asynchronously:
execute:withPtys:asynchronously:

- **execute:**(NSString *)*aString* **withPtys:**(BOOL)*ptyFlag* **asynchronously:**(BOOL)*async*

Sets up pipes to a child process and starts it executing the command in *aString*. If *async* is NO then this method will not return until the process finishes running. The *ptyFlag* parameter turns PTY support on and off. Returns **self**. The degenerate methods use the instance variables **usePtys** for *ptyFlag* and **asynchronous** for *async*.

See also: ±environment, ±execChild:, ±pause:, ±resume:, and **±terminate:**

init

- **init**

Initializes a new, asynchronous instance of MiscSubprocess which will not use PTYs. The environment is initialized to be a copy of the current environment, and no delegate is assigned. No process is run initially. Returns **self**.

See also: ±init:withDelegate:keepEnvironment:withPtys:asynchronously:

init:
init:asynchronously:
init:withPtys:
init:withPtys:asynchronously:
init:keepEnvironment:
init:keepEnvironment:asynchronously:
init:keepEnvironment:withPtys:
init:keepEnvironment:withPtys:asynchronously:
init:withDelegate:
init:withDelegate:asynchronously:
init:withDelegate:withPtys:
init:withDelegate:withPtys:asynchronously:
init:withDelegate:keepEnvironment:
init:withDelegate:keepEnvironment:asynchronously:
init:withDelegate:keepEnvironment:withPtys:
init:withDelegate:keepEnvironment:withPtys:asynchronously:

- **init:**(NSString *)*aString*
  withDelegate:*theDelegate*
  keepEnvironment:(BOOL)*envFlag*
  withPtys:(BOOL)*ptyFlag*
  asynchronously:(BOOL)*async*

This is the designated initializer for the MiscSubprocess class. Initializes a new instance of MiscSubprocess and runs the command in *aString* with *theDelegate* as the delegate object. If *aString* is **nil**, then no command is run; the instance variables will be set up as specified and then ±**execute:** should be called for anything to happen. If *envFlag* is YES, then the environment is initialized to be a copy of the current environment. If *envFlag* is NO, then the environment starts out empty, which could cause strange things to happen unless you know exactly what you are doing. Use ±**environment** to access and modify the NSMutableDictionary which contains the environment used by the MiscSubprocess. The parameters *ptyFlag* and *async* are used to set up the instance variables *usePtys* and *asynchronous*, respectively. (They are used by the degenerate ±**execute:** methods.) Returns *self*. In the degenerate methods, *aString* defaults to **nil**, *theDelegate* defaults to **nil**, *envFlag* defaults to YES, *ptyFlag* defaults to NO, and *async* defaults to YES.

See also: ±execute:withPtys:asynchronously:, ±init

isPaused

- (BOOL)**isPaused**

Returns YES is the child process is paused. Returns NO otherwise.

See also: ±isRunning, ±pause:, and **±resume:**

isRunning

- (BOOL)**isRunning**

Returns YES if the child process is running, NO otherwise.

See also: ±isPaused, ±pause:, and **±resume:**

pause:

- **pause:***sender*

Pause the child process by sending it a SIGSTOP signal. Returns *self*.

See also: ±isPaused, ±isRunning, ±resume:, and **±terminate:**

pid

- (int)**pid**

Returns the process ID (pid) of the child process, if it exists. If there is no child process running, the return value is undefined and invalid.

resume:

- **resume:***sender*

Resumes a paused child process by sending a SIGCONT signal. Returns *self*.

See also: ±isPaused, ±isRunning, ±pause:, and **±terminate:**

send:
send:withNewline:

- **send:**(NSString *)*string* **withNewline:**(BOOL)*wantNewline*

Sends *string* to the child process. If *wantNewLine* is true, an additional newline character (`\n') is sent to the child process. In the degenerate method, *wantNewline* defaults to YES. Returns *self*.

See also: ±send: and **±terminateInput**

setDelegate:

- **setDelegate:***anObject*

Sets the delegate object. See the delegate methods below for a description of what information is sent to the delegate. Returns *self*.

See also: ±delegate

setExecArgs:::

- **setExecArgs:**(NSString *)*a0* **:**(NSString *)*a1* **:**(NSString *)*a2*

Changes the command that will be used to run the child process. This allows you to choose the shell to execute the child, for example. The path to the shell should be in *a0*. The values of *a1* and *a2* will be used as argv[0] and argv[1], respectively. The default arguments are ª/bin/shº, ªshº, ª-cº. Returns *self*.

See also: ±delegate

terminate:

- **terminate:***sender*

Sends a SIGKILL to the child process, terminating it's execution. Returns *self*.

See also: ±pause:, ±resume:, and **±terminate:**

terminateInput

- **terminateInput**

Terminate the data being sent to the child process. This will send an EOF to the child's stdin. Returns *self*.

See also: ±send:withNewLine:

Delegate Methods

subprocess:done::

- **subprocess:***sender* **done:**(int)*status* **:**(MiscSubprocessEndCode)*code*

Sent to the delegate when the child process completes. The exit code *code* is the reason for the process' termination, and is one of Misc_Exited, Misc_Stopped, Misc_Signaled, or Misc_UnknownEndCode. If the process exited normally, the exit code is returned in *status*. If the process was stopped, then *status* contains the number of the signal that caused the process to stop. If the process was signaled, then *status* contains the number of the signal that caused the process to terminate.

subprocess:output:

- **subprocess:***sender* **output:**(NSString *)*output*

Sent whenever there is data on the process' standard output pipe. The data is passed in *output*.

See also: ±subprocess:stderrOutput:

subprocess:outputData:

- **subprocess:***sender* **outputData:**(NSData *)*output*

Sent whenever there is data on the process' standard output pipe. The data is passed in *output*. This will be sent before ±**subprocess:output:**, and is slightly more efficient since there is one fewer copy of the data done. If both ±**subprocess:output:** and ±**subprocess:outputData:** are implemented, then both will be called.

See also: ±subprocess:stderrOutput:

subprocess:stderrOutput:

- **subprocess:***sender* **stderrOutput:**(NSString *)*output*

Sent whenever there is data on the process' standard error pipe. The data is passed in *output*.

See also: **±subprocess:output:**

subprocess:stderrOutputData:

- **subprocess:***sender* **stderrOutputData:**(NSData *)*output*

Sent whenever there is data on the process' standard error pipe. The data is passed in *output*. This will be sent before ±**subprocess:stderrOutput:**, and is slightly more efficient since there is one fewer copy of the data done. If both ±**subprocess:stderrOutput:** and ±**subprocess:stderrOutputData:** are implemented, then both will be called.

See also: **±subprocess:output:**

subprocess:error:

- **subprocess:***sender* **error:**(NSString *)*errorString*

Sent if an error occurs when dealing with the child process. Most errors will occur when trying to start the process.