NAME

Dpkg::IPC − helper functions for IPC

DESCRIPTION

Dpkg::IPC offers helper functions to allow you to execute other programs in an easy, yet flexible way, while hiding all the gory details of IPC (Inter-Process Communication) from you.

FUNCTIONS

`$pid` = spawn(%opts)

Creates a child process and executes another program in it. The arguments are interpreted as a hash of options, specifying how to handle the in and output of the program to execute. Returns the pid of the child process (unless the wait_child option was given).

Any error will cause the function to exit with one of the Dpkg::ErrorHandling functions.

Options:

exec
: Can be either a scalar, i.e. the name of the program to be executed, or an array reference, i.e. the name of the program plus additional arguments. Note that the program will never be executed via the shell, so you can't specify additional arguments in the scalar string and you can't use any shell facilities like globbing.

  Mandatory Option.

from_file, to_file, error_to_file
: Filename as scalar. Standard input/output/error of the child process will be redirected to the file specified.

from_handle, to_handle, error_to_handle
: Filehandle. Standard input/output/error of the child process will be dup'ed from the handle.

from_pipe, to_pipe, error_to_pipe
: Scalar reference or object based on IO::Handle. A pipe will be opened for each of the two options and either the reading (`to_pipe` and `error_to_pipe`) or the writing end (`from_pipe`) will be returned in the referenced scalar. Standard input/output/error of the child process will be dup'ed to the other ends of the pipes.

from_string, to_string, error_to_string
: Scalar reference. Standard input/output/error of the child process will be redirected to the string given as reference. Note that it wouldn't be strictly necessary to use a scalar reference for `from_string`, as the string is not modified in any way. This was chosen only for reasons of symmetry with `to_string` and `error_to_string`. `to_string` and `error_to_string` imply the `wait_child` option.

wait_child
: Scalar. If containing a true value, **wait_child()** will be called before returning. The return value of **spawn()** will be a true value, not the pid.

nocheck
: Scalar. Option of the **wait_child()** call.

timeout
: Scalar. Option of the **wait_child()** call.

chdir
: Scalar. The child process will chdir in the indicated directory before calling exec.

env
: Hash reference. The child process will populate `%ENV` with the items of the hash before calling exec. This allows exporting environment variables.

delete_env
: Array reference. The child process will remove all environment variables listed in the array before calling exec.

sig Hash reference. The child process will populate `%SIG` with the items of the hash before calling exec. This allows setting signal dispositions.

delete_sig
Array reference. The child process will reset all signals listed in the array to their default dispositions before calling exec.

wait_child($pid, `%opts`)
Takes as first argument the pid of the process to wait for. Remaining arguments are taken as a hash of options. Returns nothing. Fails if the child has been ended by a signal or if it exited non-zero.

Options:

cmdline
String to identify the child process in error messages. Defaults to ''child process''.

nocheck
If true do not check the return status of the child (and thus do not fail it has been killed or if it exited with a non-zero return code).

timeout
Set a maximum time to wait for the process, after that kill the process and fail with an error message.

CHANGES

Version 1.02 (dpkg 1.18.0)
Change options: **wait_child()** now kills the process when reaching the 'timeout'.

Version 1.01 (dpkg 1.17.11)
New options: **spawn()** now accepts 'sig' and 'delete_sig'.

Version 1.00 (dpkg 1.15.6)
Mark the module as public.

SEE ALSO

Dpkg, Dpkg::ErrorHandling