## NAME

init−d−script − interpreter for short and simple init.d scripts.

## DESCRIPTION

Generic init.d script framework to reduce the redundant code in */etc/init.d/*. The goal is to create an init.d script that is Debian and LSB compliant. When the Debian policy conflict with the LSB, the Debian policy take preference.

This is a simple example on how init−d−script can be used to start and stop a daemon with PID file support:

```
#!/usr/bin/env /lib/init/init-d-script
### BEGIN INIT INFO
# Provides:          atd
# Required-Start:    $syslog $time $remote_fs
# Required-Stop:     $syslog $time $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: run at jobs
# Description:       Debian init script to start the daemon
#                    running at jobs.
### END INIT INFO
DAEMON=/usr/sbin/atd
```

Following variables affect behaviour of init script:

**DAEMON**
path to daemon being started. If init script is not supposed to start any kind of daemon, functions **do_start_override**, **do_stop_override** and **do_status_override** should be defined instead.

**DAEMON_ARGS**
additional arguments, passed to daemon during start.

**NAME** Additional environment variables are sources from */etc/default/${NAME}*. If unset, this variable defaults to basename of ‘DAEMON’ value.

**COMMAND_NAME**
If this variable is set, it is used as argument to **−−name** option of **start−stop−daemon**. It may be useful if value of **NAME** variable is too long.

**PIDFILE**
path to file, where process identifier of started daemon will be stored during start. If value is verbatim **none**, process identifier will not be stored in any file. If this variable is not set, it gets sensible default value, so is rarely necessary to set this variable explicitly.

Additionally, it is possible to behaviour of resulting shell script by overriding some of internal functions. To do so, define function with **_override** suffix. So, for example, to override **do_status** function one should define **do_status_override** function. **EXCEPT** to redefine **do_reload** function, it should be defined as−is, **without** suffix.

Here is control flow chart, that expalins what functions are called and when:

```
/etc/init.d/script start
  do_start
    do_start_prepare # no-op
    do_start_cmd     # start-stop-daemon is called here
    do_start_cleanup # no-op

/etc/init.d/script stop
  do_stop
    do_stop_prepare # no-op
    do_stop_cmd     # start-stop-daemon is called here
    do_stop_cleanup # no-op
```

```
/etc/init.d/script status
  do_status

/etc/init.d/script reload
  do_reload
    do_usage
    exit 3

/etc/init.d/script force-reload
  do_force_reload
    do_reload   # if overridden
    do_restart
      do_restart_prepare
      do_stop_cmd
      do_start_cmd
      do_restart_cleanup

/etc/init.d/script restart
  do_force_restart

/etc/init.d/script try-restart
  if do_status ; then
    do_restart
      do_restart_prepare
      do_stop_cmd  # start-stop-daemon is called here
      do_start_cmd # start-stop-daemon is called here
      do_restart_cleanup

/etc/init.d/script <arg>
  do_unknown <arg>
    exit 3

/etc/init.d/script
  do_usage
```

As can be seen, by default script does not support **reload** action, it should be implemented by script writer by defining **do_reload** function.

If daemon performs reload action upon receiving **SIGUSR1** signal, generic implementation can be used with following code:

```
alias do_reload=do_reload_sigusr1
```

## SEE ALSO

**inittab**(8), **service**(8), **update−rc.d**(8).

## AUTHORS

Petter Reinholdtsen <pere@debian.org>.