This document is a draft proposal for a guidelines towards Slackware service files.

Slackware uses a simple BSD style init script system.

Following the Slackware philosophy, init scripts are quite simple and do not perform any kind of black magic. For example, there are no dependencies between init scripts or monitoring.

In this section we will cover the creation of a usual script. Please follow the template as much as possible for consistency with all scripts.

Style

The following guidelines applies to the init scripts.

- Write pure POSIX shell scripts,
- Use two spaces indents,
- Put braces on their own lines,
- Use CAPITALIZED global variables,
- Put user controllable options at the top of the file,
- Start all functions with the program name (same as file name too),
- Avoid braces in variables expansions unless necessary,
- Don't indent case labels.

Functions

At least, a init script should support at least *start* and *stop* arguments.

The recommended list of commands are defined as following:

- **start**: start the script (may perform sanity checks before),
- **stop**: stop the script gracefully,
- **restart**: calls stop and start,
- **reload** (optional): do some reload if applicable,
- **status** (optional): check wether the service is running.

Return 1 from the service file if you were unable to perform the operation requested.

Message formats

For consistency purposes, use the following message formats in your scripts.

Usage

When the script is ran without arguments, use the following message.

```
Usage: /etc/rc.foo {start|stop}
```

Add all commands between braces in alphabetical order.

Command: start

It's usually preferred to show exactly what the command is started with its arguments to help debugging. Thus, please follow the following format for starting scripts:

```
Starting foo: /usr/bin/foo --argument --option
```

If the program is already running, use the following message:

```
foo is already running: 1234
```

Command: stop

The preferred format is:

```
Stoping foo...
```

Don't write any message if the service is already down.

Command: status

Use the following forms if running or not running respectively:

```
foo is running: 1234
```

```
foo does not seem to be running
```

Errors

If your script is unable to perform its operation, use the following form:

```
Simple error message. Aborting.
```

Topics

Handling pid file

If your daemon is able to store its process id from, please honor the default pid location with the init script to increase the out of the box experience.

Note that /var/run is not writable by anyone and some programs try to write the pid file after dropping privileges. If this is the case, use a dedicated subdirectory (e.g. /var/run/program/program.pid) and change permissions/owners on it.

Example, check if a process is running

```
PID=/var/run/$PRGNAM.pid

program_status()
{
  if [ -s $PID ]; then
    echo "$PRGNAM seems to be running"
  fi
}
```

Killing the process

To kill the process, refer to the documentation of your program. It's usually safe to rely on `kill -QUIT` though.

Template

Example and template file for any new service. Substitute NAME and remove `# NOTE:` comments.

```
#
# run control file for NAME
#
# User options:
#
# USER: set an alternative uid (default: www)
# GROUP: set an alternative group (default: www)
#

USER=www
GROUP=www
PRGNAM=magicd
```

```
BIN=/usr/bin/prog
CONF=/etc/prog.conf
ARGS=-c $CONF -u $USER -g $GROUP
PID=/var/run/prog.pid

magicd_start()
{
  if [ ! -r $CONF ]; then
    echo "No configuration file available. Aborting"
    exit 1
  fi

  if [ -s $PID ]; then
    echo "$PRGNAM is already running: $(cat $PID)"
    exit 1
  fi

  if [ -x $BIN ]; then
    echo "Starting $PRGNAM: $BIN $ARGS"
  fi
}

magicd_stop()
{
  if [ -s $PID ]; then
    kill -QUIT $(cat $PID)
  fi
}

magicd_restart()
{
  magicd_stop
  sleep 3         # NOTE: adjust or remove if possible
  magicd_start
}

magic_status()
{
  if [ -s $PID ]; then
    echo "$PRGNAM is running: $(cat $PID)"
  else
    echo "$PRGNAM does not seem to be running"
  fi
}

case $1 in
restart)
    magicd_restart
    ;;
start)
    magicd_start
```

```
    ;;
status)
    magicd_status
    ;;
stop)
    magicd_stop
    ;;
*)
    echo "Usage: $0 {restart|start|status|stop}"
    ;;
esac
```

howtos