Bash Guide for Beginners

Chapter 3. The Bash environment

3.1. Shell initialization files

3.1.1. System-wide configuration files

3.1.1.1. /etc/profile

When invoked interactively with the `--login` option or when invoked as **sh**, Bash reads the `/etc/profile` instructions. These usually set the shell variables `PATH`, `USER`, `MAIL`, `HOSTNAME` and `HISTSIZE`.

On some systems, the **umask** value is configured in `/etc/profile`; on other systems this file holds pointers to other configuration files such as:

- `/etc/inputrc`, the system-wide Readline initialization file where you can configure the command line bell-style.

- the `/etc/profile.d` directory, which contains files configuring system-wide behavior of specific programs.

All settings that you want to apply to all your users' environments should be in this file. It might look like this:

```
# /etc/profile

# System wide environment and startup programs, for login setup

PATH=$PATH:/usr/X11R6/bin

# No core files by default
ulimit -S -c 0 > /dev/null 2>&1

USER="`id -un`"
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"

HOSTNAME=`/bin/hostname`
HISTSIZE=1000

# Keyboard, bell, display style: the readline config file:
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

PS1="\u@\h \W"

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC PS1

# Source initialization files for specific programs (ls, vim, less, ...)
for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        . $i
    fi
done

# Settings for program initialization
source /etc/java.conf
export NPX_PLUGIN_PATH="$JRE_HOME/plugin/ns4plugin/:/usr/lib/netscape/plugins"
```

```
PAGER="/usr/bin/less"

unset i
```

This configuration file sets some basic shell environment variables as well as some variables required by users running Java and/or Java applications in their web browser. See [Section 3.2](#).

See [Chapter 7](#) for more on the conditional **if** used in this file; [Chapter 9](#) discusses loops such as the **for** construct.

The Bash source contains sample `profile` files for general or individual use. These and the one in the example above need changes in order for them to work in your environment!

3.1.1.2. /etc/bashrc

On systems offering multiple types of shells, it might be better to put Bash-specific configurations in this file, since `/etc/profile` is also read by other shells, such as the Bourne shell. Errors generated by shells that don't understand the Bash syntax are prevented by splitting the configuration files for the different types of shells. In such cases, the user's `~/.bashrc` might point to `/etc/bashrc` in order to include it in the shell initialization process upon login.

You might also find that `/etc/profile` on your system only holds shell environment and program startup settings, while `/etc/bashrc` contains system-wide definitions for shell functions and aliases. The `/etc/bashrc` file might be referred to in `/etc/profile` or in individual user shell initialization files.

The source contains sample `bashrc` files, or you might find a copy in `/usr/share/doc/bash-2.05b/startup-files`. This is part of the `bashrc` that comes with the Bash documentation:

```
alias ll='ls -l'
alias dir='ls -ba'
alias c='clear'
alias ls='ls --color'

alias mroe='more'
alias pdw='pwd'
alias sl='ls --color'

pskill()
{
        local pid

        pid=$(ps -ax | grep $1 | grep -v grep | gawk '{ print $1 }')
        echo -n "killing $1 (process $pid)..."
        kill -9 $pid
        echo "slaughtered."
}
```

Apart from general aliases, it contains useful aliases which make commands work even if you misspell them. We will discuss aliases in [Section 3.5.2](#). This file contains a function, **pskill**; functions will be studied in detail in [Chapter 11](#).

3.1.2. Individual user configuration files

☞ **I don't have these files?!**

These files might not be in your home directory by default; create them if needed.

3.1.2.1. ~/.bash_profile

This is the preferred configuration file for configuring user environments individually. In this file, users can add extra configuration options or change default settings:

```
franky~> cat .bash_profile
#####################################################################
#                                                                   #
#    .bash_profile file                                             #
#                                                                   #
#    Executed from the bash shell when you log in.                  #
#                                                                   #
#####################################################################

source ~/.bashrc
source ~/.bash_login
case "$OS" in
  IRIX)
    stty sane dec
    stty erase
    ;;
#  SunOS)
#    stty erase
#    ;;
  *)
    stty sane
    ;;
esac
```

This user configures the backspace character for login on different operating systems. Apart from that, the user's .bashrc and .bash_login are read.

3.1.2.2. ~/.bash_login

This file contains specific settings that are normally only executed when you log in to the system. In the example, we use it to configure the **umask** value and to show a list of connected users upon login. This user also gets the calendar for the current month:

```
#####################################################################
#                                                                   #
#    Bash_login file                                                #
#                                                                   #
#    commands to perform from the bash shell at login time          #
#    (sourced from .bash_profile)                                   #
#                                                                   #
#####################################################################
#    file protection
umask 002        # all to me, read to group and others
#    miscellaneous
w
cal `date +"%m"` `date +"%Y"`
```

In the absence of ~/.bash_profile, this file will be read.

3.1.2.3. ~/.profile

In the absence of ~/.bash_profile and ~/.bash_login, ~/.profile is read. It can hold the same configurations, which are then also accessible by other shells. Mind that other shells might not understand the Bash syntax.

3.1.2.4. ~/.bashrc

Today, it is more common to use a non-login shell, for instance when logged in graphically using X terminal windows. Upon opening such a window, the user does not have to provide a user name or password; no authentication is done. Bash searches for ~/.bashrc when this happens, so it is referred to in the files read upon login as well, which means you don't have to enter the same settings in multiple files.

In this user's .bashrc a couple of aliases are defined and variables for specific programs are set after the system-wide /etc/bashrc is read:

```
franky ~> cat .bashrc
# /home/franky/.bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc

fi

# shell options

set -o noclobber

# my shell variables

export PS1="\[\033[1;44m\]\u \w\[\033[0m\] "
export PATH="$PATH:~/bin:~/scripts"

# my aliases

alias cdrecord='cdrecord -dev 0,0,0 -speed=8'
alias ss='ssh octarine'
alias ll='ls -la'

# mozilla fix

MOZILLA_FIVE_HOME=/usr/lib/mozilla
LD_LIBRARY_PATH=/usr/lib/mozilla:/usr/lib/mozilla/plugins
MOZ_DIST_BIN=/usr/lib/mozilla
MOZ_PROGRAM=/usr/lib/mozilla/mozilla-bin
export MOZILLA_FIVE_HOME LD_LIBRARY_PATH MOZ_DIST_BIN MOZ_PROGRAM

# font fix
alias xt='xterm -bg black -fg white &'

# BitchX settings
export IRCNAME="frnk"

# THE END
franky ~>
```

More examples can be found in the Bash package. Remember that sample files might need changes in order to work in your environment.

Aliases are discussed in Section 3.5.

3.1.2.5. ~/.bash_logout

This file contains specific instructions for the logout procedure. In the example, the terminal window is cleared upon logout. This is useful for remote connections, which will leave a clean window after closing them.

```
franky ~> cat .bash_logout
######################################################################
#                                                                    #
#    Bash_logout file                                                #
#                                                                    #
#    commands to perform from the bash shell at logout time          #
#                                                                    #
######################################################################
clear
franky ~>
```

3.1.3. Changing shell configuration files

When making changes to any of the above files, users have to either reconnect to the system or **source** the altered file for the changes to take effect. By interpreting the script this way, changes are applied to the current shell session:

Figure 3-1. Different prompts for different users
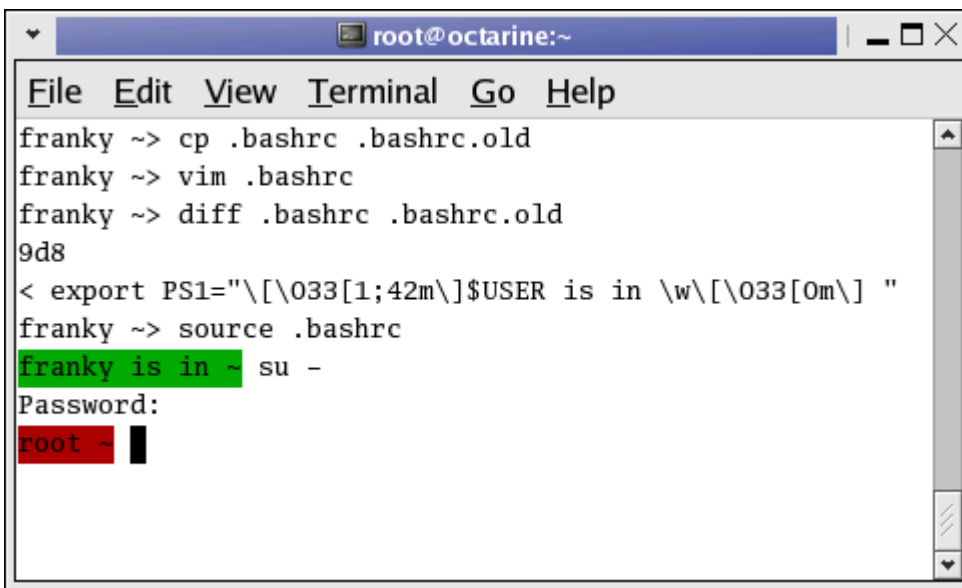
Most shell scripts execute in a private environment: variables are not inherited by child processes unless they are exported by the parent shell. Sourcing a file containing shell commands is a way of applying changes to your own environment and setting variables in the current shell.

This example also demonstrates the use of different prompt settings by different users. In this case, red means danger. When you have a green prompt, don't worry too much.

Note that **source resourcefile** is the same as **. resourcefile**.

Should you get lost in all these configuration files, and find yourself confronted with settings of which the origin is not clear, use **echo** statements, just like for debugging scripts; see Section 2.3.2. You might add lines like this:

```
echo "Now executing .bash_profile.."
```

or like this:

```
echo "Now setting PS1 in .bashrc:"
export PS1="[some value]"
echo "PS1 is now set to $PS1"
```