Intro to the Bash shell

Author: Daniele Pizzolli
Contact: ors@tovel.it

Location: Villa Sant'Ignazio — Trento
Version: Draft
Note: Please send your feedback.

Learning bash

> Learning bash the wrong way can be a very confusing process.
>
> [IBMDEV0]

The long way to a shell

Usually, on modern consumer hardware, the boot process involves the following steps:

- hardware initialization, done by the BIOS
- boot loader (grub, grub2, elilo, ...)
- kernel bootstrap (which may use a ramdisk)
- init or upstart
- login and/or {x,g,k}dm

The long way to a shell (2)

Of course there are others way to boot a system, for example, you could boot a system over the network with tftp and PXE and similar.

And init or similar are not needed for a minimal system. See linux-2.6/init/main.c.

```
run_init_process("/sbin/init");
run_init_process("/etc/init");
run_init_process("/bin/init");
run_init_process("/bin/sh");
```

Question

What is the package that contains the first executed binary by the linux kernel?

Answer

```
dpkg -S /sbin/init || dpkg -S /etc/init || dpkg -S /bin/init || dpkg -S /bin/sh
```

In general `dpkg -S [filepath]` outputs `package: filepath`.

Login

So we are at `login` prompt.

This screen comes from the `login` program. It ask a user name and a password and if you insert the right ones it will start your default shell.

Default Shell Choice

The default shell choice is usually stored in `/etc/passwd`.

Default Shell Initialization

Every shell has it owns default settings and files, usually the user could override the settings. We see later where these files are.

A step back: essential packages

Every system has a minimal list of essential packages, a trading off between size and usability.

In Debian/Ubuntu the list is available in the `build-essential` package, in the file: `/usr/share/build-essential/essential-packages-list`.

Note that the packages is architecture dependant.

More on minimal installation

See some packages and files:

- `debootstrap`: `/usr/share/debootstrap/scripts/lenny`
- `live-helper`: `/usr/share/live-helper/lists/minimal`
- `xen-tools`: `/etc/xen-tools/role.d/minimal`

Back to our shell

Once you have inserted you correct user and password you will see the shell prompt.

Usually it's in the form:

```
username@hostname:/current/working/path$
```

And now it'our turn

As you may know, the main interface to a shell is the keyboard.

Usually what you type is echoed back on the console or terminal.

The names console and terminal bring us back to the early day of the unix history as they refers to the names of the physical hardware that was used by the end user of the system.

Screen

The output you see is usually made of lines of text.

It's possible to get some colors on the console.

But it's time to use a better interface.

Virtual Terminal

Inside an `X session` you may choose various terminal-emulator:

```
$ debtags search 'x11::terminal' | wc -l
60
```

Or you can use a local or remote link using `ssh` and event it's name is *Secure SHell* it's not a shell. It is a secure channel for get a shell on a remote system. Basically is the secure version of the `telnet` program.

We will address `ssh` later.

Who

A unix system keep tracks of the logged users, the command `who` displays a list of logged users, the name of the terminal, the login date, and eventually the X screen number.

```
$ who
ors      tty1         2010-02-21 21:53
ors      :0           2010-02-21 15:10
ors      pts/0        2010-02-21 15:10 (:0)
```

In this case the same user is logged in three times, one in the virtual console tty1, that you can reach pressing CTRL-ALT-F1, in a X session, (in this case the first, that usually is configured in the virtual terminal 7: CTRL-ALT-F7).

You could also use the program `chvt`, but only in you are on a tty* terminal, it does not work inside a X session. Why?

More on Who

If you look closer at the open terminal, for example in `/dev/pts` you will notice that there are some terminal session that are not counted as login session.

This is a fine default. In the early days you where billed on login time duration.

Last but not least

The command that summarize the last commands is named unsurprising `last`.

The shell basis

In a shell prompt, you usually type a command and read the output back on the screen.

In the early days, before the screen, you have to print the command and see the result on a sheet printed by a printer.

It's time to be more detailed.

A shell command usually is invoked using this syntax:

```
command_name [options] [arguments]
```

Usually the options or switch starts with one dash or two dashes, the arguments usually are file or directory path on which the command will be executed.

But now a lot of programs, usually the most complex, use a different syntax:

```
command_name [action] [action_options] [arguments]
```

Notables in this list: `apt-cache`, `apt-get`, `lvm` and `git`.

Let's explore some basic commands.

Standard Input, Standard Output, Standard Error and Shell Pipes

But before we need to introduce the `pipe` concept.

For our needs is we treat the `|` as a output redirection mechanism. The output of the command preceding the `|` is connected to the input of the command after the `|`.

GNU coreutils

After the shell itself which contains some built-in commands that we will address later the most used programs come out from the GNU `coreutils` package.

```
$ dpkg -L coreutils | grep bin | sort | wc -l
102
```

Mmm, there are a lot of core programs, if we spend one minute for each program, it will take us more than one hour and half, for only one package...

We will cover only the most *core* programs for time reasons.

Let's start with program that involves the file system.

```
/bin/ls /bin/touch /bin/cp /bin/mkdir /bin/mv /bin/pwd /bin/readlink
/bin/rm /bin/rmdir /bin/df /usr/bin/du /usr/bin/stat /usr/bin/truncate
```

Brief summary

A smart way to get a command summary, at the end of the course you will be able to understand and use the syntax and the commands involved in the example.

```
ls (1)              - list directory contents
touch (1)           - change file timestamps
cp (1)              - copy files and directories
mkdir (1)           - make directories
mv (1)              - move (rename) files
pwd (1)             - print name of current/working directory
readlink (1)        - display value of a symbolic link
rm (1)              - remove files or directories
rmdir (1)           - remove empty directories
df (1)              - report file system disk space usage
du (1)              - estimate file space usage
stat (1)            - display file or file system status
truncate (1)        - shrink or extend the size of a file to the specified size
```

Home sweet home

Every program in the unix environment must follow some well defined rules. One of this rule states that the program has to use a working directory.

For the shell the starting working directory is the user home.

The user home directory is the part of the file system that belong to a well defined user.

```
$ pwd
/home/ors
```

I will assume that you will be familiar with two key concepts in the file system: files and directories. Later we will explore two more unix concepts: hard links and soft links.

type type

In the previous slide the command `pwd` in reality does not executes the `pwd` that belongs to the coreutils packages, but was as **shell builtin**.

To use the `coureutils` version there are various way:

```
$ /bin/pwd
/home/ors
$ command pwd
```

To learn more type `type type` and `type command` and a new entry: `help command`.

How many command could you chain?

What will happen if you type:

```
$ command command
```

and if you type:

```
$ command command command
```

and if you type:

```
$ command command command
```

Help yourself

Ken Thompson once said:

> On Unix you will need only the manual.

Then came the GNU's folk and with info...

```
$ help help
$ man man
$ whatis whatis
$ apropos apropos
$ info info
$ whatis man
$ man whatis
$ example_command -h
$ example_command --help
```

Recap of the first lesson

- Unix has it’s own philosopy, that you my not like
- You should be able to navigate and manipulate the file system using shell commands
- You should be able to search help on command using the command line

Exercises

1. create an empty file using two different programs from the coreutils that we have seen
2. using the man command read the “SHELL BUILTIN COMMANDS” section
3. figure out how to install the package manpages-it if you prefer the italian version
4. create some empty files in a directory named to_delete
5. remove the files and the directory using one command
6. Read (in italian) and try the examples yourself from "Appunti di informatica libera":

    - Capitolo 144. Introduzione alla shell Unix
    - Capitolo 31. La shell
