GNU – UNIX command

LINAGORA

Linagora Formation
<formation@linagora.com>
>

Summary

Summary

The shell

- This is the tool of choice for UNIX
- There are six main:

sh	Bourne shell
csh	C shell
tcsh	TENEX C shell
ksh	Korn shell
bash	Bourne-again shell
zsh	Z shell

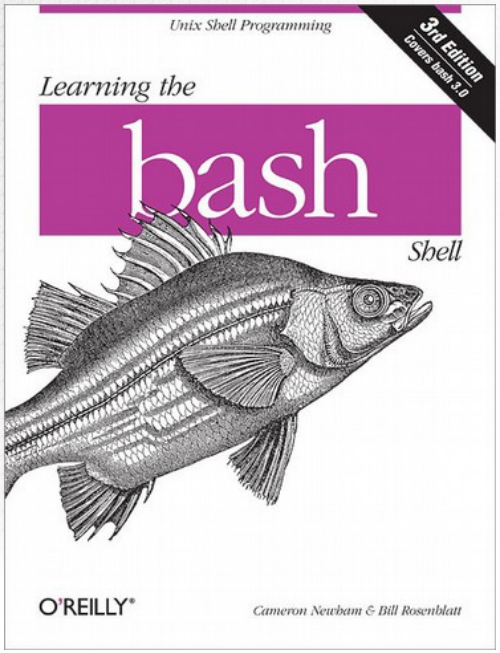
- The shell commands by default in Linux test is bash (this is the one we will use later).

The shell

http://fr.wikipedia.org/wiki/Shell_Unix
http://explainshell.com/
http://en.wikipedia.org/wiki/Bourne_shell
http://en.wikipedia.org/wiki/C_shell
http://www.tcsh.org/Welcome
http://en.wikipedia.org/wiki/Tcsh
http://www.kornshell.com/
http://en.wikipedia.org/wiki/Korn_shell
http://www.gnu.org/software/bash/ http://fr.wikipedia.org/wiki/Bourne-Again_shell
http://zsh.sourceforge.net/
http://fr.wikipedia.org/wiki/Zsh

http://fr.wikipedia.org/wiki/Shellshock_(faille_informatique)

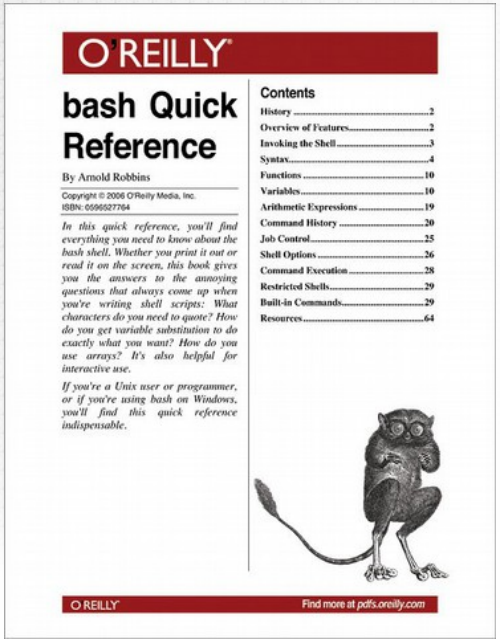
Bibliography



Cameron Newham et Bill Rosenblatt
Learning the bash Shell
3e édition
O'Reilly Media, mars 2005

http://shop.oreilly.com/product/9780596009656.do

Bibliography


O'REILLY

bash Quick Reference

By Arnold Robbins

ISBN: 0596527764

In this quick reference, you'll find everything you need to know about the bash shell. Whether you print it out or read it on the screen, this book gives you the answers to the annoying questions that always come up when you're writing shell scripts: What characters do you need to quote? How do you get variable substitution to do exactly what you want? How do you use arrays? It's also helpful for interactive use.

If you're a Unix user or programmer, or if you're using bash on Windows, you'll find this quick reference indispensable.

Contents

O'REILLY Find more at pdfs.oreilly.com

Arnold Robbins
bash Quick Reference
O'Reilly Media, juin 2006

http://shop.oreilly.com/product/9780596527761.do

The command man

Principe

- The command man will display the manual command specified as argument:

```
$ man bash
[...]
```

- The manual display is managed through the **less** command or the **more** command.

http://fr.wikipedia.org/wiki/Man_(Unix)

The command man
Option

- The -k option searches the keyword specified in an argument in the name and description of the man pages and display the matches:

```
$ man -k bash
```

```
bash (1)              - GNU Bourne-Again SHell
bash-builtins (7)     - bash built-in commands, see bash(1)
bashbug (1)           - report a bug in bash
builtins (7)          - bash built-in commands, see bash(1)
rbash (1)             - restricted bash, see bash(1)
```

The command pwd

- Means: *print working directory*.
- Displays the absolute path of the current directory:

```
$ pwd
/home/formation
```

- Is really useful because the current directory path often figure in the prompt.

http://fr.wikipedia.org/wiki/Pwd

The command uname

Principme

- Means: *UNIX name*.
- Displays the system information

```
$ uname
Linux
```

http://fr.wikipedia.org/wiki/Uname

The command uname
Options

- The -s option (or --kernel-name) displays the kernel name:

```
$ uname -s
Linux
```

- The -n option (or –nodename) displays the computer name:

```
$ uname -n
Linux
```

- The option -r (or –kernel-release) displays the version number of the kernel:

```
$ uname -r
2.6.32-5-686
```

The command uname

Options

- The option -v (or –kernel-version) displays the identification of the kernel:

```
$ uname -v
#1 SMP Mon Oct 3 04:15:24 UTC 2011
```

- The option -m (or --machine) displays the type of material:

```
$ uname -m
i686
```

- The option -o (or --operating-system) displays the operating system:

```
$ uname -o
GNU/Linux
```

- The option -a (or --all) displays all the previous information:

```
$ uname -a
Linux formation 2.6.32-5-686 #1 SMP Mon Oct 3 04:15:24 UTC 2011 i686 GNU/Linux
```

The command history

Principle

- Memory recording mechanism of commands entered in order to reuse all or part later.
- A limited size (500 by default to bash).
- Is saved when leaving the command in ~ / .bash_history, which will be played at the beginning of the next session to initialize history.
- It is easy to navigate in the recent history with the arrow up ↑ and down ↓.
- Using the history command is more effective for longer history.

The command history

The command history

- Displays the command history

```
$ history
    1  man bash
    2  pwd
    3  uname
    4  history
```

http://en.wikipedia.org/wiki/History_(Unix)

The command history

Recalling a previous order

- Recalling a previous order is made by an exclamation point followed by one or more characters.
- In all cases, the recalled command is displayed by the shell then immediately executed.
- `!!` allows to execute again the previous command:

```
$ date
thursday  4  July 2013,  14:29:12 (UTC+0200)
$ !!
date
thursday  4  July 2013,  14:29:27 (UTC+0200)
```

The command history

Recalling a previous order

- `!` followed by the number of a command in the history allows to rerun this command

```
$ !2
pwd
/home/formation
```

- `!` followed by one or more characters allows to re-execute the last command whose name begins with these characters:

```
$ !un
uname
Linux
```

The command echo

- The echo command is one of the simplest because that it just prints its arguments:

```
$ echo toto tata
toto tata
$ □
```

- The echo command automatically generates a newline after the last argument. The -n option allows not generate:

```
$ echo –n toto tata
toto  tata$ □
```

- The echo command is mainly used to see the effect of some treatments done by the shell.

http://fr.wikipedia.org/wiki/Echo_(Unix)

The variables

Principle

- A variable can store a value to future use.
- The shell has only one variable, which stores strings (where some programming languages distinguish several types of variables based on the values they can store).
- There are three categories of variables:
  - special variables, whose name consists of a symbol or a figure whose value is set by the shell;
  - environment variables, whose name is traditionally made up of capitals and whose value is fixed in the configuration of the command is inherited from his father;
  - variables defined by the user, whose name is traditionally made up of letters in lower case.

The variables

Creating and initializing a variable defined by the user

- The name of a variable defined by the user can contain letters (it makes the difference between capitals and lower case), numbers or underscores. It can not start with a number.
- A variable is created by assigning a value using the following syntax:

```
$ name=value
```

  We must put absolutely no space around the equal sign.
- If the value contains spaces, each must be preceded by a backslash or the value must be surrounded by double quotation marks or apostrophes:

```
$  name=value\  with\  spaces
$  name="value  with  spaces"
$  name='value  with spaces'
```

The variables
Value of a variable

- The command replaces any symbol of dollar $ followed immediately by a variable name by the value of this variable.
- We can for example display the value of a variable using the echo command with argument in a symbol immediately followed dollar variable name:

```
$ echo $name value
```

The variables

The command `set`

- The command `set` displays :
  - the environment variables;
  - variables defined by the user;
  - the function definitions.

Each line of the display indicates the name of a variable followed by an equal sign followed by the value of the variable:

```
$ set  [...]
name=value  [...]
```

http://www.gnu.org/software/bash/manual/html_node/The-Set-Builtin.html

The variables

The command env

- The command env displays the environment variables.
  Each line of the display indicates the name of a variable followed by an equal sign followed by the value of the variable:

```
$ env [...]
HOME=/home/formation LOGNAME=formation
PATH=/usr/local/bin:/usr/bin:/bin SHELL=/bin/bash
TERM=xterm USER=formation [...]
```

The variables

Some classic environment variables

```
DISPLAY   mean to access the X server:
          $ Echo $ DISPLAY
          0.0

HOME      absolute path to the the user's home
          directory:
          $ echo $HOME
          /home/formation

LANG      language to use for display
          $ echo $LANG
          fr_FR.UTF-8
```

http://fr.wikipedia.org/wiki/Variable_d'environnement

The variables

Some classic environment variables

```
DISPLAY   liste of paths of commans search:
          $ echo $PATH
          PATH=/usr/local/bin:/usr/bin:/bin

PWD       absolute path of the current
          directory:
          $ echo $PWD
          /home/formation

SHELL     chemin d'accès absolu de
          l'interpréteur de commandes :
          $ echo $SHELL
          /bin/bash

USER      ID of user:
          $ echo $USER
          formation
```

http://fr.wikipedia.org/wiki/Variable_d'environnement

The variables

The command export

- Transform any variable into environment variable:

```
$ NAME=value
$  export NAME
```

- both operations can be performed at once:

```
$  export NAME=value
```

The variables

Temporary environment variables

- It is possible to create environment variables that exist only during the execution of an order.
- For this, we can use two syntaxes:
  - by preceding the control (with its options and possible arguments) to initialize environment variables:

```
$  NAME1=value1  NAME2=value2 command
```

  - using the env command and an identical formulation:

```
$  env  NAME1=value1  NAME2=value2 command
```

The variables

The command unset

- Destroy a variable :

```
$ echo $name  value
$  unset name
$ echo $name  [white
line]
```

http://en.wikipedia.org/wiki/Unset

Command search

- When an order is entered simply by name, the corresponding executable is searched for in the directories specified by the PATH environment variable.
- We can also run a command with its path (absolute or relative):

```
$ /bin/pwd
/home/formation
```

http://en.wikipedia.org/wiki/PATH_(variable)