The bash shell and its commands

Stefano Scanzio
(stefano.scanzio [At] gmail (dot) com)

Web: http://wiki.altervista.org/cs/bash_shell

Date: 2014

Outline

Linux Commands

Generic Linux Command Syntax

`command [-options] [arguments]`

- If a command is to long to stay on a single row, it may be continued on the next row using the character "\" as last character of the row
- More than one command can be executed on the same line, separating the commands with ";"
  `command1; command2; ...`
  commands will be executed sequentially

Help

- All linux commands are documented, and their documentation can be accessed using:
  - `man <command>`: gives an extensive description of command
  - `apropos <command>`: search a keyword in the manual page descriptions
  - `whatis <command>` gives a short description of command
  - `command -h` or `command -help`: gives a description of command options

The Unix file system

- The Unix file system is a hierarchical file system organized in directory:
  - `/`: is the symbol that represents the root of the file system
  - `.` `(dot)`: is the actual directory
  - `..` `(dot dot)` is the father directory
  - `~` is the home directory, in my case `/home/scanzio`
- A file that starts with `"."` `(dot)` is an `hidden file`
- Files within a directory can be accessed using:
  - **an absolute path**: `/dir1/dir2/file`
  - **a relative path**: `subdir1/subdir2/file`

Files Commands

- **cp [-fir] src1 src2 . . . dest**
  - copy one or more files, possibly in a directory
- **rm [-fir] file1 file2 . . .**
  - delete *file1*, *file2*, . . .
  - files can be directories if the -r option is used. In this case the entire directory content will be recursively deleted
- **mv [-fi] file1 file2 . . . dest**
  - move one or more file, possibly in a directory
- File commands options are:
  - **-f**: force, no confirmation is required
  - **-i**: interactive, confirmation is required for each file
  - **-r**: recursive, the file command works recursively for each subdirectory

Directories Commands

- **cd dir**
  - goes into the *dir* directory
  - `cd ..`: goes into the father directory
  - `cd` (alone): goes into the home directory
- **pwd**
  - prints the name of the current directory
- **mkdir dir**
  - makes the *dir* directory
- **rmdir dir**
  - removes the *dir* directory
  - *dir* must be empty

Link Commands

Symbolic link

. . . is a file that contains a reference to another file or directory

- **Hard link**: more than one file name point to the same file
- **Soft link**: is a special file that contains the file that must be pointed

- **ln src alias**
  - makes an hard link between src and alias
  - hard links may not normally point to directories and they can not link paths on different volumes, for this reason usually `soft link` are preffered
- **ln -s src alias**
  - makes a symbolic soft link between src and alias

Files and Directories protection

- In Linux, every file or directory has special protections that allow differential access to the different users of the system
- The system users are grouped into three sets:
  - **u**: user, the owner of the file
  - **g**: group users
  - **o**: other users
- Files protections are:
  - **r**: read permission
  - **w**: write permission
  - **x**: execution permission
- Directories protections are:
  - **x**: directory crossing
  - **r**: files list
  - **w**: files creation and/or deletion

Files and Directories protection Commands

- Change the group of the files:
  - `chgrp [-R] group file1 file2 ...`
- Change the owner (and potentially the group) of the files:
  - `chown [-R] user[:group] file1 file2 ...`
- Change the protection of the files:
  - `chmod [-R] protection file1 file2 ...`
  - protection is the sum of: 4 for r, 2 for w and 1 for x respectively for user, group and others
  - Examples:
    - ★ `chmod 777 file`: user has `rwx` access to the file, group has `rwx` access and also others (`-rwxrwxrwx`)
    - ★ `chmod 664 file`: user has `rw-` access to the file, group has `rw-` access and others has `r-` access (`-rw-rw-r-`)
    - ★ `chmod 754 file`: user has `rwx` access to the file, group has `r-x` access and others has `r-` access (`-rwxr-xr-`)
- `-R`: recursive, the command works recursively for each subdirectory

The ls Command

- `ls [-options] [files]`
- list directory contents
- Options are:
  - `-a`: all, list also hidden files
  - `-l`: long, long format output
  - `-r`: reverse, reverse order of output
  - `-t`: time, files are sorted by modification time
  - `-R`: recursive, the command works recursively for each subdirectory

ls command example

```
ls -al ~/tmp
```

```
drwxr-xr-x 3 user user 4096 2009-10-13 15:10 ./
drwxr-xr-x 64 user user 12288 2009-10-13 15:10 ../
drwxr-xr-x 2 user user 4096 2009-10-13 15:10 adir/
-rw-r–r– 1 user user 29 2009-10-13 15:10 afile
lrwxrwxrwx 1 user user 5 2009-10-13 15:24 alink -> afile
```

- this command lists the `tmp` directory which is in the home directory showing also the hidden files with a long output format
  - the first character is the file type (- standard file, d directory, l link)
  - rwxr-xr-x are the protections
  - the number of the second column is 1 for files and the number of file contained in the directory for directories
  - the first user is the owner, the second is the group
  - the second number is the dimension in byte
  - date and hour are the creation time
  - the last is the filename, in the case of soft link the linked file is reported

Viewing a textual file

- Using visual editor like `gedit`
- `cat file1 file2 ...`
  - print in order file1 and file2 content
- `tail [-n number] file1 file2 ...`
  - print the last number rows of file1 and file2 (default is 10 rows)
  - with +number prints the rows from number to the end of file1 and file2 (ex. tail -n +15)
- `head [-n number] file1 file2 ...`
  - like `tail`, but it prints the first number rows of file1 and file2
- `more file`
  - prints a file page by page. Use space to go to the next page
- `diff file1 file2`
  - displays differences between file1 and file2

The find command

- `find <dir> [-opt]`: it finds files present in the `dir` directory and in each subdirectory
- Options are:
  - `-name pattern`: search `pattern`. Pattern can be a filename or a part of filename using the symbol *to denote the rest of the file
  - examples of pattern:
    - `abc`: the file abc is searched
    - `\*ab`: files that end with ab are searched
    - `bc\*`: files that begin with bc are searched
    - `\*dd\*`: files that contain dd are searched
  - `-type [b c d l]`: b=block file, c=character file, d=directory, l=link
  - `-exec command \;`: for each file found executes `command`
  - `-exec command \{} \;`: for each file found executes `command file_found`. `\{}` is the file found

The grep command

- `grep [-opt] pattern file1 file2 ...`: prints lines matching a pattern found in file1, file2...
- Options are:
  - `-c`: prints the number of matching lines
  - `-i`: case insensitive
  - `-l`: prints only names of files containing the pattern
  - `-n`: prints line number of matched lines
  - `-v`: prints only lines that non match pattern
- Pattern can be a regular expression:
  - `.`: represents any character
  - `^`: the begin of the row
  - `$`: the end of the row
  - `*`: zero or more repetitions
  - `+`: one or more repetitions

The tar command

- `tar [options] tarfile [file1 file2 ...] [dir1 dir2 ...]`: compression or decompression of files and directories
- Compression:
  - `c`: creation of a new tar file
  - `f`: file, must be present both in the compression phase, both in the decompression phase
  - `v`: verbose mode
  - `z`: zipped, the file is compressed
  - Example: `tar cvzf /tmp/file.tgz /home/user/Documents /home/user/a.txt`: creates verbose zipped file. The Documents directory and the file a.txt are compressed in the archive /tmp/file.tgz

The tar command (2)

- Decompression:
  - `x`: extraction of files from an archive
  - `t`: test the content of an archive
  - Example1: `tar tvzf /tmp/file.tgz`: test verbose zipped file. Test the archive previously compressed
  - Example2: `tar xvzf /tmp/file.tgz`: extracts verbose zipped file. Extract in the current directory the archive previously compressed

The sort command

- `sort [-opt] file1 file2 ...`: sorts file1, file2, ...
- Options are:
  - `-b`: ignores blanks spaces at the beginning of a line
  - `-f`: case insensitive
  - `-r`: reverses the result
  - `-n`: numeric sort (for numbers)

The bash shell

- A shell is a textual interface between the user and the operating system
- The shell is not part of the Linux kernel, is a normal program
- Since the shell is a normal process, different shells has been developed. The most commons are Bourne shell (`sh`), C-shell (`csh`), Tahoe C-shell (`tcsh`) and Bourne again shell (`bash`)
  - Our focus will be on `bash`
- Each shell, when opened, search a configuration script in the home directory and run it
  - it can be used to customize the shell behavior or to initialize environment variables
  - in `bash` this file is `.bashrc`

Completion and History

- Using TAB the shell automatically complete file names (using the file present in the actual directory or in the directories listed in the environment variable `$PATH`).
- ↑↓ use arrows to scroll commands previously performed
- use the `history` command to show the history buffer that lists the last executed commands associating them with a number
- `!n` executes a command present in the buffer identified with the number `n`
- `!$` is the last parameter of the command previously performed
- `!*` are all the parameter of the command previously performed
- `!string` perform the last command that begin with `string`

Pipe

- `command1 | command2`
- A `pipe` is a connection between the first command stdout and the second command stdin
- This linking is performed by the operator `|`
- Examples:
  - `ls | head -n 5`
  - `cat file | tail -n 25` that does the same thing of `tail -n 25 file`
  - `cat file | sort -n -u | head -n 5`

Process Management

- Batch execution: `comand1; command2` command1 is executed. When command1 is finished command2 is executed.
- Concurrent execution: `command1 &; command2` command1 and command2 are executed concurrently
  - command1 is executed in background (symbol &).
  - it returns immediately the control to the shell that executes command2.
- Summariziong:
  - normally a command is run foreground (`fg`)
  - with the symbol & after the command it can be run background (`bg`)
  - A program that run fg can be suspended pressing CTRL-Z
  - A suspended program can be restarted foreground using the command fg or background using the command `bg`

Process Management

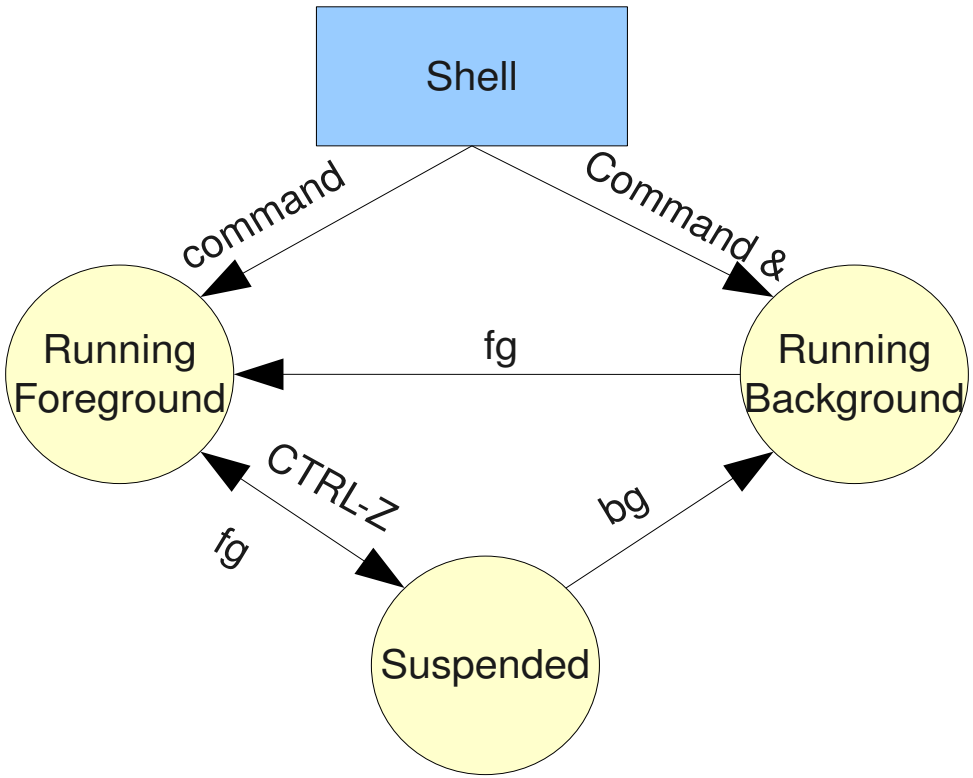

Figure : Process states and transitions between them

Scripts - Shell commands in a File

- A sequence of commands can be written in a file and they can be executed directly calling it
- It can be executed indirectly:
  - `source <sciptfile> <args>`
- It can be executed directly:
  - the script file must have the execution permissions
  - the first row must begin with `#!` and with the absolute path of the shell used (example: `#!/bin/tcsh`)
  - to run the script type `./scriptfile` in the shell prompt

Shell Variables

- `variableName=value`
  - instantiates a variable with the name `variableName`, setting it to `value`
  - `a=10, b="pippo"`
- `$variableName`
  - used to access the variable
  - to print a variable the command `echo $variableName` can be used
  - to print the value of variable `b` for example:

```
echo $b
pippo
```

Shell Environment Variables

- Environment variables are written in upper-case and are used for the shell configuration
- `env`: displays shell environment variables
- `printenv var`: prints the value of `var` shell environment variables
- `export var`: makes the variable `var` available to all the processes launched with the shell

Useful Environment Variables

- `HOME`: the home directory
- `PATH`: directory where the shell finds shell commands
- `SHELL`: the used shell
- `LD_LIBRARY_PATH`: it shows the directories where shared libraries are located
- Example: add to the environment variable `PATH` the directory `/home/scanzio/bin`:

```
export PATH=$PATH:/home/scanzio/bin
```