Sperimentazioni di Fisica I mod. A – Laboratorio 1

UNIX Tutorial (Part I)

Dipartimento di Fisica e Astronomia "G. Galilei",
Università degli Studi di Padova

The BASH shell

- one among the several shells available for LINUX
- also called Bourne-again shell, after Stephen Bourne, the creator of an earlier shell (`/bin/sh`)
- a shell is a program that accepts and executes commands:
  - supports programming constructs, allowing complex commands to be built from smaller parts;
  - commands, or scripts, can be saved as files to become new commands
- shells have some *builtin commands*, (`cd, break, exec`)
- and use three standard I/O streams:

`stdin` is the standard input stream which provides input commands

`stdout` is the standard output stream, which displays output from commands

`stderr` is the standard error stream, which displays error output from commands

Commands and sequences

- On LINUX (and UNIX) systems, commands have:
  1. a command name
  2. none, one or more options
  3. parameters
- some commands have neither options nor parameters

  `pwd`
- some have options but no parameters

  `ls -l`
- others have no options but do have parameters

  `cd /tmp`
- if a line contains the `#` character, all the remaining characters on the line are ignored (a comment).

echo

- The `echo` command prints its agruments to the terminal

```
$ echo Word
Word

$ echo One word
One word

$ echo One word        and   spaces
One word and spaces

$ echo "One word        and   spaces" # a comment
One word        and   spaces
```

- bash uses *white spaces* to separate input line into *tokens* which are passed to the command.
- Quoting preserves additional white space and makes the whole string a single token.

`echo` command options

- The `echo` commands has two options:
  - `-n` to suppress a *newline character* to the output
  - `-e` to enable backslash escaped characters to special meaning:

Escape sequence	Function
`\a`	Alert (ring bell)
`\b`	Backspace
`\c`	Suppress trailing newline (same as `-n` option)
`\f`	Form feed
`\n`	New line
`\r`	Carriage return
`\t`	Horizontal tab

exit

- you can terminate the shell using the `exit` command
- an optional exit code may be given
- if the shell is running in a terminal window, on a graphical desktop, the window will close
- if connected to a remote system, using `ssh`, the connection will end
- in the bash shell, the sequence `CTRL-D` produces the same result.

Environment and SHELL variables

- While running in a bash shell, the *environment* is:
  - the form of your prompt
  - your home directory
  - your working directory
  - the name of the shell
  - files you may have opened, ...
- the *environment* includes many variables (with a name and content) defined by the user or by the shell
- your reference the variables, by prefixing its name with `$`

`USER`	username	`UID`	user identification
`HOME`	user's home directory	`PWD`	current working directory
`SHELL`	name of the shell	`PS1`	user's prompt

```
$ echo Nome: $USER, Shell: $SHELL, User ID: $UID
Nome: agarfa, Shell: /bin/bash, User ID: 3872
```

env

- the `env` command displays the current environment variables

```
$ env
HOSTNAME=spiro4.fisica.unipd.it
TERM=rxvt
SHELL=/bin/bash
USER=agarfa
MAIL=/var/spool/mail/agarfa
PATH=/usr/lib/qt-3.3/bin:/usr/kerberos/bin:/usr/common/bin:
/usr/common/modeltech/linux/:/usr/local/bin:/bin:/usr/bin:
/usr/X11R6/bin:/usr/common/root/bin:/home/agarfa/bin
PWD=/home/agarfa
LANG=en\_US.UTF-8
HOME=/home/agarfa
...
```

Command history

- the bash shell maintains a history of all typed commands.
- the `HISTSIZE` environment variable tells bash how many history lines to keep.

`history`	Displays the entire history
`history N`	Displays the last `N` entries of your history
`history -d N`	Deletes line `N` from your history
`!!`	The most recent command
`!N`	The `N-th` history command
`!string`	The most recent command that starts with `string`
`!?string?`	The most recent command that contains `string`

Paths

- on LINUX and UNIX systems, all files are accessed as part of a single large tree that is rooted at `/`
- all standard users have a home directory whithin the `/home` directory, such as `/home/agarfa`.
- if you type a command name, bash looks for that command on your *path*, which is a colon-separated list of directories in the `PATH` environment variable

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/home/agarfa/bin
```

- if you want to know what command will be executed if you type a particular string, use the `which` or `type` commands

```
$ which echo
/bin/echo

$ type which
which is aliased to `alias | /usr/bin/which -tty-only
-read-alias -show-dot -show-tilde'
```

Absolute and Relative paths

- If a command is not in your `PATH` specification, a path has to be given with the command name. There are two types of paths:
  1. absolute paths : staring with `/`, such as
     `/bin/echo`
  2. relative paths : relative to your current directory (as reported by the `pwd` command),
     `./esercizi_c++/my_program`

- two special names can be used in paths:
  - `.` a *single dot*, which refers to the current directory;
  - `..` a *pair of dots*, referring to the parent of the current directory.

- the special character ~ (tilde) means your own home directory;
  `~username` refers to the user `username` home directory
  (i.e. `/home/username`)

Changing the working directory

- After login, the user will be in his/her home directory
- Since you can execute programs from various directories, you can also change the working directory with the `cd` command
- The argument to `cd` must be an *absolute* or *relative* path to a directory.
- The characters `.`, `..`, and `~` can be used in paths.
- `cd` with no arguments changes to your *home directory*
- `cd -` means a change to the previous working directory

- the user home directory is stored in the `HOME` environment variable
- the previous directory is stored in `OLDPWD`:
- `cd -` is equivalent to `cd $OLDPWD`

The UNIX MANual pages

- The primary source of documentation is the *manual pages*, which can be accesed using the `man` command

```
$ man man
man(1)                                                          man(1)

NAME
       man - format and display the on-line manual pages

SYNOPSIS
       man  [-acdfFhkKtwW]  [--path]  ... [-S section_list] [section] name ...

DESCRIPTION
       man formats and displays the on-line manual pages.  If you specify
       section, man only  looks  in  that section  of the manual. name is
...
       See below for a description of where man looks for the manual page files.

OPTIONS
       -C  config_file
...
SEE ALSO
       apropos(1), whatis(1), less(1), groff(1), man.config(5).
```

Listing directory entries

- the `pwd` command *prints the working directory*
- the `ls` command lists a directory content
- on a storage device, a file or directory is contained in a collection of blocks. Information on a file is contained in an `inode` which records
  - the file owner
  - when the file was last accessed
  - how large is the file
  - whether is a directory or not
  - who can read or write it

A directory listing

- the `ls -l` command displays a long format listing of all files in a directory

```
$ ls -l
total 5224
-rwxr-xr-x 1 agarfa fisica   11757 Oct 23  2005 a.out*
drwxr-xr-x 2 agarfa fisica    4096 Oct 26  2004 C00/
drwxr-xr-x 2 agarfa fisica    4096 Oct 26  2004 cprog/
drwxr-xr-x 5 agarfa fisica    4096 Jan 28  2008 Desktop/
drwxr-xr-x 2 agarfa fisica    4096 Oct 23  2007 esercizi/
-rw-r--r-- 1 agarfa fisica    5882 Oct 15  2007 ls.help.txt
drwx------ 2 agarfa fisica    4096 Oct 14 10:12 mail/
-rwxr-xr-x 1 agarfa fisica   11757 Oct 23  2005 sea*
-rw-r--r-- 1 agarfa fisica   14538 Oct 23  2005 sea.after-cpp
-rw-r--r-- 1 agarfa fisica      90 Oct 23  2005 sea.c
-rw-r--r-- 1 agarfa fisica     940 Oct 23  2005 sea.o
-rw-r--r-- 1 agarfa fisica     390 Oct 23  2005 sea.s
-rwxr-xr-x 1 agarfa fisica  962788 Oct 23  2005 sea_static*
```

Directory special entries

- The home directory contains special files, whose names start whit a dot (`.`).
- Every directory has at least two special entries, the directory itself (`./`) and the parent directory (`../`)
- To list all special files, use the `-a` option to `ls` command:

```
[agarfa@spiro4]$ ls -a
./                  .emacs*             .kderc              sea*
../                 .esd_auth           .lesshst            sea.after-cpp
.addressbook        esercizi/           ls.help.txt         sea.c
.addressbook.lu     .fonts.cache-1      mail/               sea_deb*
.adobe/             .forward            .metacity/          sea.o
a.out*              .forward.orig       .mozilla/           sea.s
a.ps                .fullcircle/        mtr/                sea_static*
.bash_history       .gconf/             .pinerc             .spamassassin/
.bash_logout        .gconfd/            pippo               .ssh/
.bash_profile       .gnome/             primo*              .ssh2/
.bashrc             .gnome2/            primo.cxx           temp_data.dat
C00/                .gnome2_private/    primo.o             temp_serial.dat
.config/            .gnome-desktop/     primo_static*       .Trash/
cprog/              .gstreamer-0.10/    prova               ucf07/
Desktop/            .gstreamer-0.8/     .recently-used      .viminfo
.dmrc               .ICEauthority       .redhat/            .Xauthority
.eggcups/           .kde/               .screenrc
```

Copy, move and delete

`cp`

is used to make a copy of one or more files. You must give at least two names, the *source* and the *target* files. Both may contain a path specification.

`mv`

is used to *move* or *rename* one or more files or directories. The same rules as for copying with `cp` may be followed.

`rm`

is used to *remove* one or more files.

making, deleting and listing directories

`mkdir`

is used to create new directories. You must give at least one directory name, and it may contain a relative or absolute path specification.

`rmdir`

is used to delete a directory.
You can only remove a directory with `rmdir` if it is empty.

Recursive file manipulation

listing

The `ls` command has a `-R` option for listing a directory and all its subdirectories.

```
ls -R
```

copying

you can use the `-r` (or `-r` or `-recursive`) option to cause the `cp` command to descend into source directories and copy contents recursively.

```
cp -r esercizi esercizi-backup
```

deleting

the `-r` option tells the `rm` command to remove both directories and included files or subdirectories

```
rm -r esercizi
```

Wildcards and globbing

- Often you may need to perform a single operation on many filesystem objects.
- A wildcard support is built in to the bash shell (also called globbing)

`?` matches any single character

`*` matches any string, including an empty string

`[` introduces a *character class*, a non-empty string terminated by a '`]`'. The '`-`' character between two others represents a range which includes the two other characters.
As an example, `[0-9a-fA-F]` represents any upper or lowrcase hexadecimal digit.