# Sperimentazioni di Fisica I mod. A – Laboratorio 2

## UNIX Tutorial (Part II)

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

# The BASH I/O Streams

- All the UNIX shells use three standard I/O streams:

stdout , the standard output stream which displays output from commands.
stderr , the standard error stream which displays error output from streams.
stdin , the standard input stream which provides input to commands.

| Stream | File descriptor |
|---|---|
| `stdin` | 0 |
| `stdout` | 1 |
| `stderr` | 2 |

- Input streams provide input to programs, usually from terminals.
- Output streams print text characters, usually to terminals.

# Redirecting output

- There are two ways to redirect output:

`n>`

redirects output from file descriptor `n` to a file.

- You must have write permission to the file.
- If the file does not exist, it is created.
- If the file exists, the existing content is lost without any warning.

`n >>`

redirects output from file descriptor `n` to a file.

- You must have write permission to the file.
- If the file does not exist, it is created.
- If the file exists, the output is appended to the existing file.

# Redirecting output streams - examples

## No redirection

```
$ ls a* z*
/bin/ls: z*: No such file or directory
a1  a2 a3  a4 a5  a.out*  a.ps
```

## `stdout` redirection

```
$ ls a* z* >stdout.txt
/bin/ls: z*: No such file or directory

$ cat stdout.txt
a1
a2
a3
a4
a5
a.out*
a.ps
```

## `stderr` redirection

```
$ ls a* z* 2>errori.txt
a1  a2 a3  a4 a5  a.out*  a.ps

$ cat errori.txt
/bin/ls: z*: No such file or directory
```

## both output streams redirected

```
$ ls a* z* >stdout.txt 2>>errori.txt

$ cat errori.txt
/bin/ls: z*: No such file or directory
/bin/ls: z*: No such file or directory
```

# Redirecting both streams to the same file

- Sometimes both output streams have to be redirected to one file
- The procedure is often used for automated processes or in background jobs.

- Three possibilities:
  1. `command 1> output.log 2> output.log`
  2. `command > output.log 2>&1`
  3. `command &> output.log`

- To append output to a file, replace `>` with `>>`

- Q: How to ignore output streams ?
- A: redirect the appropriate stream to `/dev/null`

## Ex: Ignoring error output stream

```
$ ls a* z* 2>/dev/null
a1  a2 a3  a4 a5  a.out*  a.ps
```

# Input redirection

- `stdin` stream can be redirected from a file, using the `<` operator

### `stdin/stdout` redirect example

```
$ ls -1 > list.out
```

Redirect **stdout** to **list.out**

```
$ cat list.out
a1
a2
a3
```

All entries in ascending order

```
$ sort -r < list.out
a3
a2
a1
```

Process input from file with **sort** giving a reverse ordering

# The cat command

- The cat command, short for *catenate*, allows to display the contents of a file on stdout:

  ```
  $ cat list.out
  a1
  ...
  a3
  ```

- The cat command takes input from stdin if you do not specify a file name; it keeps reading from stdin until the end-of-file. (ctrl-d is used to signal end-of-file).

## Creating a file with cat

```
$ cat >list2.out
a1
...
a3
$
```

**Ctrl-d** is pressed

# Input redirection summary

| | |
|---|---|
| `> file` | Direct standard output to `file` |
| `< file` | Take standard input from `file` |
| `>> file` | Direct standard output to `file`; append to file if it exists |
| `<< label` | Here-document |
| `n> file` | Direct file descriptor `n` to `file` |
| `n< file` | Take file descriptor `n` from `file` |
| `n >> file` | Direct file descriptor `n` to `file`; append to file if it exists |
| `n>&` | Duplicate standard output to file descriptor `n` |
| `n<&` | Duplicate standard input from file descriptor `n` |
| `n>&m` | File descriptor `n` becomes a copy of the output file descriptor |
| `n<&m` | File descriptor `n` becomes copy of the input file descriptor |
| `&>file` | Directs standard output and standard error to file |

# `head` or `tail` ... and `wc`

- While `cat` allows to concatenate files and print them on the standard output

`head` output the first part of files (10 lines by default)

- `head -n K` prints the first K lines of a file;
- `head -c B` prints the first B bytes of a file;

`tail` output the last part of files (10 lines by default)

- `tail -n K` prints the last K lines of a file;
- `tail -c B` prints the last B bytes of a file;

`wc` prints newline, word and byte counts for each file

- `wc -c` print the byte counts;
- `wc -l` print the newline counts;
- `wc -w` print the word counts;

# Finding text in files

- The `grep` command print lines matching a pattern

```
grep [OPTIONS] PATTERN [FILE...]
```

- `grep` searches the named input FILEs, or standard input if no files are named, for lines containing a match to the given PATTERN.
  - `grep -i` ignore case distinctions in both the PATTERN and the input files.
  - `grep -n` prefix each line of output with the 1-based line number within its input file.

```
[agarfa]$ grep iostream *.cxx
primo.cxx:#include <iostream>
secondo.cxx:#include <iostream>

[agarfa]$ grep -n iostream *.cxx
primo.cxx:1:#include <iostream>
secondo.cxx:1:#include <iostream>
```

# Foregroud and background jobs

- When you run a command in a terminal window, you are running it in the foreground.
- The Bash shell has a *suspend key*, `Ctrl-z`. Pressing this key combination, you get the terminal prompt again.
- The clock is still on your destop but has stopped running.
- It can be restarted with the `fg` command
- `fg` brings the job right back to the foreground, but you no longer have a shell prompt.
- The `bg` command continues running your job in backgorund and giving back the terminal prompt

```
[agarfa]$ xclock -d -update 1
[2]+  Stopped               xclock -d -update 1
[agarfa]$ fg %2
xclock -d -update 1
[1]+  Stopped               xclock -d -update 1
[agarfa]$ bg %1
[1]+ xclock -d -update 1 &
```

# Using &

- To start a new job in *background*, add a & character at the end of the command
- The message shows a *job number* and a *process id* (PID).
- With the `jobs` command is possible to find out what jobs are running
- The `-l` option prints PIDs and the plus sign `(+)` beside the job number indicates that is the current job

```
[agarfa]$ xclock -d -update 2 &
[1] 29696
[agarfa]$ jobs -l
[1]+  29696 Running      xclock -d -update 1 &
```

# ps: inspecting process status information

- The ps command accepts zero or more PIDs as argument and displays the associated process status
- Using ps with no options will list all processes that have our terminal as their controlling terminal

```
[agarfa]$ jobs -l
[1] 29696

[agarfa]$ ps 29696
  PID TTY      STAT   TIME COMMAND
29696 pts/6    S      0:00 xclock -d -update 1

[agarfa]$ ps
  PID TTY          TIME CMD
27082 pts/6    00:00:00 bash
29696 pts/6    00:00:00 xclock
30170 pts/6    00:00:00 ps
```

# top

- The top command displays a continuosly updated process list, along with useful summary information
- the -p option allows to control a single process

```
[agarfa]$ top -p 29696
top - 18:58:50 up 10:32,  7 users,  load average: 0.18, 0.28, 0.27
Tasks:   1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie
Cpu(s):  9.9%us,  1.8%sy,  0.2%ni, 87.2%id,  0.6%wa,  0.1%hi,  0.3%s
Mem:   2074216k total,  2018076k used,    56140k free,    54776k buf
Swap:  3903784k total,    23580k used,  3880204k free,  1501260k cac

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
29696 alberto   20   0  7692 3400 2808 S    0  0.2   0:03.36 xclock
```

# kill

- To `Ctrl-c` sequence terminates a running program.
  The sequence sends a `SIGINT` or interrupt signal to the process.
- The `kill` command sends a signal to a job or process.
  Type `man kill` to get a table of all available signals.

# Esercizi di Oggi

1. Scrivere un programma che chieda all’utente di inserire **due numeri interi**, ne calcoli la **somma** e la **differenza** e stampi a schermo i **due risultati**.

2. Scrivere un programma che chieda all’utente di inserire all’utente la **base** e l’**altezza** di un **triangolo**, ne calcoli l’**area** e stampi a schermo il **risultato**.

3. Scrivere un programma che chieda all’utente di inserire la **lunghezze** dei **tre lati** di un **parallelepipedo**, ne calcoli il **volume** e la **superficie laterale** e stampi a schermi **i due risultati**.