Basic Unix Commands

Bash Commands

- man <cmd>
  - Shows manual for a command.

- sudo <cmd>
  - Execute a command as the superuser.

File System Navigation

- cd <path>
  - Change current working directory.

- ls [<opts>] [<path>]
  - List the contents of a directory (current working directory by default).
  - Options:
    - -l: use a long listing format.
    - -a, --all: do not ignore entries starting with “.”
    - -r, --reverse: reverse order while sorting
    - -R, --recursive: list subdirectories recursively

- pwd
  - Print name of the current working directory.

File System Manipulation

- mkdir <dirpath>
  - Create a new directory, if it does not already exist..

- rmdir <dirpath>
  - Remove a directory, if it is empty.

- mv <srcpath> <destpath>
  - Move (rename) the file <srcpath> to <destpath>.

- rm [<opts>] <path>
  - Remove a file.
  - Options:
    - -R: remove directories and their contents recursively.

- cp [<opts>] <srcpath> <destpath>
  - Copy the file <srcpath> to <destpath>.
  - Options:
    - -R: copy directories and their contents recursively.

- ln [<opts>] <target> <destpath>
  - Create an hard link to file <target> in <destpath>.
  - Options:
    - -s: make symbolic links instead of hard links.

Process Management

- ps [<opts>]
  - List active processes.

    - Options:
        - -e: list all processes.
        - -l: use a more detailed listing format.

- top
    - Show a a dynamic real-time view of running processes.

- kill -<signum> <pid>
    - Send the signal <signum> to the process specified by <pid>.
    - Signals:
        - -9: terminate a process.

- <cmd> &
    - Execute command in background.

File Operations

- cat <filepath>
    - Print file on the standard output.

- more <filepath> / less <filepath>
    - Print file on the standard output page by page.

- sort [<opts>] <filepath>
    - Sort lines of text files (alphabetically by default).
    - Options:
        - -r: reverse order.
        - -n: use numerical order.
        - -k <key>: sort via key.

- cut [<opts>] <filepath>
    - Remove sections from each line of a text file.
    - Options:
        - -f <field1,field2,...fieldN>: specify fields to select.
        - -d <delim>: use <delim> instead of TAB for field delimiter.

- tr [<opts>] <set1> [<set2>]
    - Translate standard input characters from <set1> to <set2>.
    - Options:
        - -d: delete characters from <set1> instead of translating them.

- uniq [<opts>] <filepath>
    - Omit repeated lines.
    - Options:
        - -d: report only repeated lines instead of omitting them.

- grep [<opts>] <filepath>
    - Print lines of a file matching a pattern.
    - Options:
        - -e <pattern>: specify a pattern to be matched.

        - -E <pattern>: specify extended regular expression pattern to be matched.
        - -H: print the file name for each match.
        - -n: print the line number for each match.
        - -i: ignore case.
        - -v: invert the sense of matching, to select non-matching lines.
        - -l, --quiet, --silent: no output is produced.
        - -l, --files-with-matches: print out only file names.

- wc [<opts>] <filepath>
    - Print newline, word, and byte counts for a file.
    - Options:
        - -w, --words: print words counts.
        - -c, --bytes: print the byte counts.
        - -m, --chars: print the character counts.
        - -l, --lines: print the newline counts.

File Search

- find [<directory>] [<options>] [<actions>]
    - Search for files in a directory hierarchy (with a specified root, i.e., <directory>).
    - Options:
        - -name <pattern>: search files whose name matches the pattern.
        - -regex <pattern>: search files whose path matches a regular expression.
        - -regextype posix-extended: specify posix-extended format for regular expressions
        - -type <f|l|d>: search files of a specific type.
        - -mindepth <depth>: search files starting from the specified directory tree depth.
        - -maxdepth <depth>: search files up to the specified directory tree depth.
        - -size <[+,-]n[cwkMG]>: search files whose size starts from (+) or goes up to (-) the specified size. (c=bytes, w=words, k=kilobytes, M=megabytes, G=gigabytes).
        - -user uname: File is owned by user uname (numeric user ID allowed).
        - -group gname: File belongs to group gname (numeric group ID allowed).
    - Actions:
        - -exec <cmd>: execute command on each matched file.
            - \{} (or ‘{}’) can be used as a placeholder for the file path.
            - The command must end with \; (or ‘;’).

File Permissions Management

- chmod [<opts>] <mode> <file>
    - Change file permissions. <mode> can be specified symbolically ([ugoa][+-][rwx]) or numerically (octal digits).

    - Options:
        - -R: change permissions of files and directories recursively.

String manipulation

- basename path
    - Strip directory and suffix from path.
- dirname path
    - Strip last component from path.

Redirections

- cmd1 | cmd2
    - Redirect standard output of cmd1 to standard input of cmd2.
- cmd < file
    - Redirect standard input of cmd from file.
- cmd > file
    - Redirect standard output of cmd to file.
- cmd 2> file
    - Redirect standard error of cmd to file.
- cmd &> file
    - Redirect standard output and standard error of cmd to file.
- cmd >> file
    - Append standard output of cmd to file.

Shortcuts

- CTRL+C
    - Terminate the current foreground process.
- CTRL+Z
    - Stop the current foreground process.
- TAB
    - Autocompletion.

Threads and Synchronization

The Pthread Library

- `int pthread_equal (pthread_t tid1, pthread_t tid2);`
- `pthread_t pthread_self (void);`
- `int pthread_create (pthread_t *tid, const pthread_attr_t *attr, void *(*startRoutine)(void *), void *arg);`
- `void pthread_exit (void *valuePtr);`
- `int pthread_join (pthread_t tid, void **valuePtr);`
- `int pthread_cancel (pthread_t tid);`
- `int pthread_detach (pthread_t tid);`

Semaphore POSIX

- `int sem_init (sem_t *sem, int pshared, unsigned int value);`
- `int sem_wait (sem_t *sem);`
- `int sem_trywait (sem_t *sem);`
- `int sem_post (sem_t *sem);`
- `int sem_getvalue (sem_t *sem, int *valP);`
- `int sem_destroy (sem_t *sem);`

Mutex POSIX

- `int pthread_mutex_init (pthread_mutex_t *mutex, const pthread_mutexattr_t *attr);`
- `int pthread_mutex_lock (pthread_mutex_t *mutex);`
- `int pthread_mutex_trylock (pthread_mutex_t *mutex);`
- `int pthread_mutex_unlock (pthread_mutex_t *mutex);`
- `int pthread_mutex_destroy (pthread_mutex_t *mutex);`

The Bash Language

Execution of a script

- Direct:
  - ./scriptname args
    - The file scriptname must include “#!/bin/bash” in the first row
- Indirect
  - source ./scriptname args
    - It is the current shell to execute the script

Variable assigments

var_name=value

Command echo (to print on stdout)

echo [OPTIONS] [STRING]

- Options:
  - -n: to not go into the new line.
  - -e: to interpret escape characters.

Command read (to read from stdin)

read

- with one or more variables passed as argument
- use of the variable $REPLY

Quoting

- Single quoting ''
  - variables are not exapanded
- Double quoting “”
  - variables are expanded
- Ex:
  - a=pippo
  - echo "$a pippo "'$a'" pluto"
  - pippo pippo $a pluto
  - echo $a pippo '$a' pluto
  - pippo pippo $a pluto

Use of { } parenthesis to delimit the name of a variable

- Es.
  - name=Jean
  - echo ${name}paul
  - Jeanpaul

Capture of the stdout of a command

- $(<command>)

Command exit

- exit [numeber]
  - terminate the execution of a process, returning a value to the calling process
- Ex:
  - exit 0
  - return a true value

Execution of arithmetic computations

- a method chosen by the student
- Ex.
  - let s=$n1+$n2
  - Assign to the variable $s the sum of $n1 and $n2

Special shell variables

- $0, $1, $2, ...
  - passing parameters on the command line
- $*
  - complete list of parameters, excluding the name of the script
- $#
  - number of parameters
- $$
  - PID of the process
- $?
  - returned value of the last executed process

Construct if-then-else (and elsif)

```
if condition ; then
  statements
elif condition
then
  statements
else
  statements
fi
```

Construct while (including the redirection of stdin and stdout)

```
while condition
do
  statements
done << $fileIn >> $fileOut
```

Required formats for the condition of the constructs if and while

Only the conditions expressed between [...] are required (instead, the conditions based on the keyword test are not required)

- Numerical comparisons:
  - -eq equal (==)
  - -ne not equal (!=)
  - -gt major (grater) (>)
  - -ge major or equal (greater equal) (>=)
  - -lt minor (less) (<)
  - -le minor or equal (less equal) (>)

- Strings comparisons:
  - = equal
  - != not equal

- Conditions on files:
  - -d <arg> true if <arg> is a directory
  - -f <arg> true if <arg> is a file
  - -r <arg> true if <arg> has read permission
  - -w <arg> true if <arg> has write permission
  - -x <arg> true if <arg> has execution permission

- Logical operators usable within a condition:
  - ! not
  - -a and
  - -o or

- Logical operators usable in a list of conditions:
  - && and
  - || or

Costrutto for

```
for var in [ list ]
do
  statements
done
```

Instructions

- break
- continue

Vectors

```
# Declarations
array[3]="value"
array=( 4 8 7 )
array=( [0]=4 [1]=8 [2]=7 [5]=10 )
# Access
echo ${array[1]} # Access to the element 1 of the vector (value 8)
echo ${array[*]}  # Print of all the elements of the array
echo ${!array[*]} # Print of all the keys of the array
echo ${#array[*]} # Number of elements contained in the array
```

Associative vector

```
# Declarations
declare -A array
array[“key”]=”value”
array=( [pippo]=hello [2]="pluto" ["pluto"]=2 )
# Access
echo ${array[pippo]} # Access to the element “pippo” of the array
(value “hello”)
echo ${array[*]}  # Print of all the elements of the array
echo ${!array[*]} # Print of all the keys of the array
echo ${#array[*]} # Number of elements contained in the array
```