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
```