Lezione 5

- *Creare una sottodirectory* `bin` *all'interno della propria home directory in cui mettere gli script. Fare in modo che gli script contenuti in* `bin` *possano essere invocati da qualunque directory con il nome del file, senza dover specificare l'intero pathname.*
  Digitare `export PATH=$PATH:~/bin`.

- *Qual è l'effetto della seguente sequenza di comandi? Perché?*

  ```
  > cat >chdir
  cd ..
  Ctrl-d
  > chmod 700 chdir
  > chdir
  > pwd
  ```
  Contrariamente a quanto ci si potrebbe aspettare la directory corrente dopo l'esecuzione dello script non è cambiata. Infatti ogni script viene eseguito in una sottoshell della shell corrente e gli effetti dei comandi come `cd` hanno effetto soltanto nella sottoshell.

- *Creare un alias permanente* `lo` *per il comando* `exit`.
  Aggiungere al file `.bash_profile` nella propria home una linea contenente `alias lo=exit`.

- *Progettare uno script che prende come parametro una stringa e un file di testo e controlla se la stringa compare nel file.*

  ```
  grep "$1" "$2" 2>/dev/null | wc -l
  ```

  Lo script precedente stampa sullo standard output il numero di occorrenze della stringa fornita come primo argomento nel file fornito come secondo argomento. Quindi se il numero stampato è `0` significa che la stringa non occorre nel file.

- *Il comando* `read` *assegna alla variabile speciale* `REPLY` *un testo acquisito da standard input. Qual è l'effetto dello script* `words` *contenente i seguenti comandi?*
  ```
  echo -n 'Enter some text: '
  read one two restofline
  echo 'The first word was: $one'
  echo 'The second word was: $two'
  echo 'The rest of the line was: $restofline'
  exit 0
  ```

  La linea `echo -n 'Enter some text: '` provoca la stampa su standard output della stringa `Enter some text:` senza newline finale. Il comando `read one two restofline` legge dallo standard input una linea di testo memorizzando la prima parola nella variabile `one`, la seconda parola nella variabile `two` ed il resto della linea nella variabile `restofline`. I comandi nelle tre linee successive stampano il valore di queste variabili assieme ai rispettivi messaggi. Lo script termina con un exit status pari a `0`, grazie al comando `exit 0`.

- *Qual è l'effetto della seguente sequenza di comandi? Perché?*

```
> cat >data
echo -n the date today is:
date
Ctrl-d
> chmod 700 data
> data
```

Viene stampato il messaggio `the date today is:` seguito immediatamente dalla data ed ora correnti dato che viene specificata l'opzione `-n` del comando echo che sopprime la stampa del newline alla fine del messaggio da visualizzare.

- *Scrivere uno script che estragga soltanto i commenti dal file con estensione `java` fornito come primo argomento, sostituendo `//` con la stringa `linea di commento del file` <nome del file>:. Inoltre i commenti estratti devono essere salvati nel file fornito come secondo argomento.*

```
sed "s?//?linea di commento del file $1:?w $2" -n $1
```

- *Progettare uno script che prende in input come parametri i nomi di due directory e copia tutti i file della prima nella seconda, trasformando tutte le occorrenze della stringa `SP` in `SU` in ogni file.*

```
if test $# -ne 2
then
  echo 'Utilizzo dello script: SP_to_SU <dir1> <dir2>'
  exit 1
fi

if ! test -d $1 -a -d $2 # Gestione degli errori.
then
  echo "$1 e $2 devono essere delle directory"
  exit 2
fi

for i in $1/*
do
  sed "s/SP/SU/g" "$i" > $2/`basename $i`
done

exit 0
```

- *Progettare uno script `drawsquare` che prende in input un parametro intero con valore da 2 a 15 e disegna sullo standard output un quadrato (utilizzando i caratteri `+`, `-` e `|`) come nel seguente esempio:*

```
> drawsquare 4
+--+
|  |
|  |
+--+
```

```
if test $# -ne 1
then
  echo 'Utilizzo dello script: drawsquare <n>'
  exit 1
fi

if test $1 -le 2 -o $1 -ge 15
then
  echo 'Il parametro deve essere un numero >2 e <15'
  exit 2
fi

x=$1
y=$1

while test $y -gt 0
do
  while test $x -gt 0
  do
    if test $x -eq 1 -o $x -eq $1
    then
      if test $y -eq 1 -o $y -eq $1
      then
        echo -n "+"
      else
        echo -n "|"
      fi
    else
      if test $y -eq 1 -o $y -eq $1
      then
        echo -n "-"
      else
        echo -n " "
      fi
    fi
    x=$[$x-1]
  done
  x=$1
  y=$[$y-1]
  echo
done

exit 0
```

- *Progettare uno script che prende in input come parametro il nome di una directory e cancella tutti i file con nome* `core` *dall'albero di directory con radice la directory parametro.*

```
if test $# -ne 1
then
```

```
  echo 'Utilizzo dello script: rmcore <dir>'
  exit 1
fi

if ! test -d $1 # Gestione degli errori.
then
  echo "$1 deve essere una directory"
  exit 2
fi

find $1 -name core -exec rm {} \; 2>/dev/null

exit 0
```