

**Corso di Sistemi Operativi**  
**A.A. 2020/2021**  
**Arzigogolo 2 – 9 ottobre 2020**  
**Francesco Malferrari - Matricola 142795**

**Esercizio 1.** Si individui un meccanismo per registrare e riprodurre sessioni di lavoro al terminale. La soluzione ideale registra su file l'intera sessione di lavoro e consente la riproduzione esatta della stessa (inclusi i tempi di attesa tra un comando ed un altro) a partire dal file memorizzato. Non è consentito effettuare riprese multimediali di alcun tipo. Si fornisca al docente l'insieme di file necessario per riprodurre la sessione di comandi seguente:

```
whoami
pwd
ls
```

**Soluzione.** Il meccanismo più ottimale per risolvere l'esercizio 1 è attraverso un comando con la seguente sintassi, dove “**arzigogolo.sh**” rappresenta un qualsiasi possibile nome di file:

```
script -a arzigogolo.sh
```

L'esecuzione di questo fa partire la registrazione della sessione dei comandi, digitati dall'utente, sul file **arzigogolo.sh** (come in figura 1), dal quale è possibile riprodurre l'ordine originale delle istruzioni scritte dall'utente.

La riproduzione della sessione avviene attraverso questa istruzione (come in figura 2):

```
cat arzigogolo.sh
```

Il contenuto del file **arzigogolo.sh** è presente nella cartella compressa insieme a questo documento.

```
studente@debian:~$ script -a arzigogolo.sh
Script started, file is arzigogolo.sh
studente@debian:~$ whoami
studente
studente@debian:~$ pwd
/home/studente
studente@debian:~$ ls
1      arzigogolo.sh          histoy_log.txt  palindromi.sh  Scrivania
17644  arzigogolo.txt         Immagini       Pubblici      stdout
18     Documenti            indovinello.sh sas.txt        stringa2
18141  history_arzigogolo.sh  Modelli        Scaricati     terimanl
arz    history_arzigogolo.txt Musica          script.sh     Video
studente@debian:~$ exit
exit
Script done, file is arzigogolo.sh
studente@debian:~$
```

*Figura 1: Registrazione della corrente sessione di lavoro*

```
studente@debian:~$ cat arzigogolo.sh
Script started on 2020-10-10 22:29:29+02:00 [TERM="xterm-256color" TTY="/dev/pts
/0" COLUMNS="80" LINES="24"]
studente@debian:~$ whoami
studente
studente@debian:~$ pwd
/home/studente
studente@debian:~$ ls
1      arzigogolo.sh          histoy_log.txt  palindromi.sh  Scrivania
17644  arzigogolo.txt         Immagini       Pubblici      stdout
18     Documenti            indovinello.sh sas.txt        stringa2
18141  history_arzigogolo.sh  Modelli        Scaricati     terimanl
arz    history_arzigogolo.txt Musica          script.sh     Video
studente@debian:~$ exit
exit
Script done on 2020-10-10 22:29:38+02:00 [COMMAND_EXIT_CODE="0"]
studente@debian:~$
```

*Figura 2: Riproduzione della sessione di lavoro ordinata registrata in **arzigogolo.sh***

---

**Esercizio 2.** Si produca uno script shell di nome **indovinello.sh** che implementa il classico gioco dell'indovinello. Viene generato un numero casuale ed eseguito un ciclo infinito nel quale si chiede un numero all'utente. Se il numero è strettamente minore di quello generato, si stampa "Il numero immesso è minore.". Se il numero è strettamente maggiore di quello generato, si stampa "Il numero immesso è maggiore.". Se il numero è uguale a quello generato, si stampa il messaggio "Complimenti! Hai indovinato il numero in N tentativi.", dove N è il numero di tentativi richiesti all'utente per risolvere l'indovinello. Qual è la strategia ottimale per vincere all'indovinello? Qual è la complessità computazionale di tale strategia?

**Soluzione.** L'esercizio è stato risolto tramite uno script che ripete la richiesta attraverso un while per selezionare un numero e decretare il rapporto di grandezza che ha questa con il numero da indovinare (scelto in modo randomico). Se il numero digitato risulta minore di quello casuale allora stamperà "Il numero immesso è minore" e nel caso opposto, cioè se è maggiore, "Il numero immesso è maggiore". Se risulta che l'utente abbia indovinato apparirà una stringa per congratularsi e poi termina l'esecuzione. La strategia ottimale per vincere l'indovinello è selezionare il numero corrispondente alla metà dei numeri totali su cui scegliere (in questo caso fissato a 100) e, in base al numero scelto se è minore o maggiore di quello estratto a sorte si procederà a sottogruppi numerici grandi la metà del totale. Questo procedimento poi va reiterato fino a quando non hai solo un numero da scegliere (Come in Figura 3).

Lo pseudo-codice che rispecchia la strategia utilizza la ricorsione ed è questo, dove N rappresenta l'insieme dei numeri rimasti e dim la sua dimensione, n è la metà approssimata di N e r il numero da indovinare:

```
indovinello(N,r,dim)
    n = floor(dim/2)
    if n = r then
        return 0
    if n < r then
        stampa stringa
        toglì_parte_sinistra(N)
        indovinello(N,r)
    if n>r then
        stampa stringa
        toglì_parte_destra(N)
        indovinello(N,r)
```

La complessità computazionale dello pseudo-codice è derivata da una ricorsione che ripete le istruzioni al più per il logaritmo del numero degli elementi totali nel caso peggiore, perché uno potrebbe indovinare solo quando è rimasto un unico numero sul totale: da questo si può evincere che svolge  $O(\log_2(\text{dim}))$  dove dim sta per la grandezza dell'intervallo da cui scegliere (in questo caso 100).

Il codice di **indovinello.sh** è il seguente:

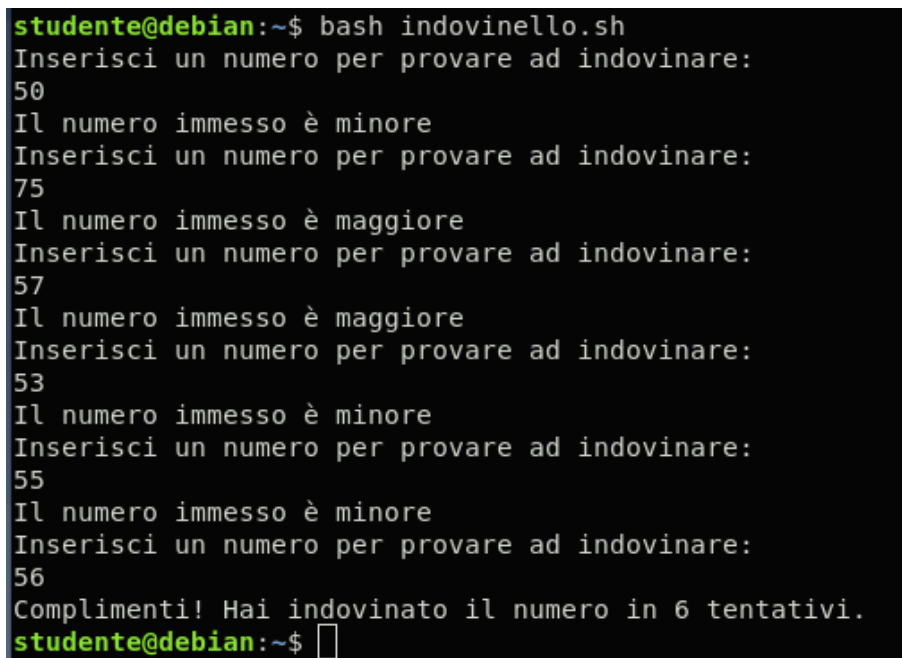
```
declare -i tentativi=0
declare -i random=$(( 1 + RANDOM % 100))
declare -i numero

while [ true ];
do
```

```

tentativi=$((tentativi+1))
echo -e "Inserisci un numero per provare ad indovinare\t"
read numero
if [ $numero -lt $random ]; then
    echo "Il numero immesso è minore"
elif [ $numero -gt $random ]; then
    echo "Il numero immesso è maggiore"
else
    echo "Complimenti! Hai indovinato il numero in
        $tentativi tentativi."
    exit 0;
fi
done

```



```

studente@debian:~$ bash indovinello.sh
Inserisci un numero per provare ad indovinare:
50
Il numero immesso è minore
Inserisci un numero per provare ad indovinare:
75
Il numero immesso è maggiore
Inserisci un numero per provare ad indovinare:
57
Il numero immesso è maggiore
Inserisci un numero per provare ad indovinare:
53
Il numero immesso è minore
Inserisci un numero per provare ad indovinare:
55
Il numero immesso è minore
Inserisci un numero per provare ad indovinare:
56
Complimenti! Hai indovinato il numero in 6 tentativi.
studente@debian:~$

```

*Figura 3: Immagine esecuzione dello script **indovinello.sh** con la strategia ottimale*

---

**Esercizio 3.** Si consideri il problema di determinare se una data stringa sia palindroma oppure no. Si chiede di progettare e dettagliare una soluzione ricorsiva a tale problema. Inoltre, si chiede di implementare la soluzione proposta in uno script shell di nome **palindromi.sh**. Tale script riceve in argomento la stringa da controllare, effettua il controllo ed esce con lo stato di uscita 0 se è palindroma, o 1 se non è palindroma. Si chiede infine di verificare tramite **palindromi.sh** la palindromia della stringa “angolo bar a bologna”.

**Soluzione.** Il programma bash sfrutta la ricorsione (come richiesto dalla consegna) per controllare se la prima lettera combacia con l'ultima, in caso negativo è sicuro che la stringa passata su linea di comando non è palindroma. Nel caso positivo invece riparte la funzione con la stringa privata del primo e dell'ultimo carattere.

Lo pseudo-codice corrispondente a questo ragionamento è questo, dove “S” è la stringa e “n” la sua dimensione:

```

palindroma(S,n)
  if n <= 1 then
    return 0
  if S[1] != S[n-1] then
    return 1

  togliere_prima_lettera S
  togliere_ultima_lettera S
  palindroma(S,n-2)

```

La stringa “angolo bar a bologna” se passata allo script nominato porterà il valore di “\$?” a 1, il che vuol dire che la frase risulta non palindroma.

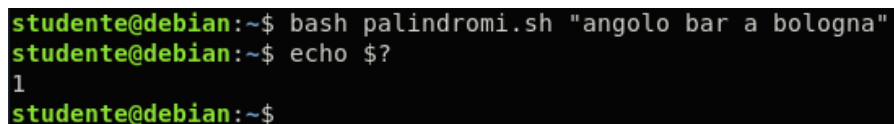
Il codice di **palindromi.sh** è il seguente:

```

palindroma() {
  if [ ${#1} -le 1 ]; then
    return 0;
  elif [ ${1:0:1} != ${1: -1} ]; then
    return 1;
  fi
  rimanente=${1:1}
  palindroma ${rimanente%?}
}

palindroma $(echo $1)

```



```

studente@debian:~$ bash palindromi.sh "angolo bar a bologna"
studente@debian:~$ echo $?
1
studente@debian:~$

```

Figura 4: Esecuzione dello script **palindromi.sh** passando come stringa “angolo bar a bologna”