Esame del 09/02/2023 (GitHub)

Esercizio 1

A. *Utilizzando awk si scriva un comando che stampi una lista dei file presenti nella directory corrente mostrando solo dimensione e nome.*

```
ls -l | awk '{print $5, $9}'
```

B. *Si calcoli la dimensione occupata in totale dai file regolari con dimensione maggiore di 1024 byte nella directory corrente*

```
ls -l | awk 'BEGIN {sum=0} { if ($5 > 1024 && substr($1,1,1) == "-") sum+=$5} END {print "sum="sum}'
ls -l | awk 'BEGIN {sum=0} { if ($5 > 1024 && $1 ~ /^-/) sum+=$5} END {print "sum="sum}'
```

C. *Si faccia in modo che il comando stampi solo i file maggiori di 1024 byte*

```
ls -l | awk '{ if ($5 > 1024) print $0 }'
```

D. *Trovare i file non acceduti negli ultimi 30 giorni*

```
ls -l --time-style=+%s | awk 'BEGIN { now=systime() } { if(now-$6 < 60*60*24*30) print $0 }'
```

Esercizio 2

Si realizzi uno script di shell BASH "`menu`", che accetta come argomento un file "`listino.txt`" strutturato nel seguente modo:

```
codice   quantità   costo
01953       2        15
07934       1        20
084Gd      10        30
9038H       1         5
```

e che implementi le seguenti funzioni accessibili da un menu:

a) *Cerca - Chiede all'utente una stringa da ricercare all'intero del listino ed effettua la ricerca*
b) *Aggiungi - Chiede all'utente il codice del prodotto da aggiungere (primo campo del listino) e la quantità di articoli desiderati, verifica le scelte effettuate e le memorizza in un file carrello*
c) *Elimina - un prodotto dal carrello*

```
#!/bin/bash
echo "Benvenuto in menù"
if [ $# -ne 1 ]
then
     echo "Utilizzo: $0 <nomefile>"
     exit 1
fi

if [ ! -f $1 ]
then
     echo "Il file $1 non esiste"
     exit 2
fi

echo "Cosa vuoi fare?"
echo "1) Cerca"
echo "2) Aggiungi"
echo "3) Elimina"
echo "4) Esci"
read scelta

case $scelta in
     # Cerca
     1) echo "Inserisci il codice"
         read codice
         grep $codice $1
         ;;
     # Aggiungi
     2) echo "Inserisci il codice"
```

```
        read codice
        if [ $(grep -c $codice $1) -eq 0 ]
        then
            echo "Il codice non esiste"
            exit 3
        fi
        echo "Inserisci la quantità"
        read quantita
        quantita_file=$(grep "$codice" $1 | awk '{print $2}')
        if [ $quantita_file -lt $quantita ]
        then
            echo "Non ci sono abbastanza prodotti"
            exit 4
        fi
        echo $codice $quantita $costo >> carrello.txt
        ;;
    # Elimina
    3) echo "Inserisci il codice da eliminare"
        read codice
        sed -i "/$codice/d" carrello.txt
        ;;
    # Esci
    4) echo "Arrivederci"
        exit 0
        ;;
    # Scelta non valida
    *) echo "Scelta non valida"
        exit 3
        ;;
esac
```

Esercizio 3

Si scriva un programma in C che prende in input i seguenti argomenti: esame i j f C1 C2
Dove esame è il nome dell'eseguibile, i e j sono due interi positivi, f il nome del file, C1 e C2 due caratteri. Il processo padre dovrà creare due processi P1 e P2, il processo P1 dovrà cercare le istanze di C1 nel file f, mentre il processo P2 dovrà cercare le istanze di C2 in f. Quando uno dei due processi trova il carattere, lo segnala al padre. Il padre si mette in attesa che i figli cercano i caratteri, quando riceve il segnale chiamerà una funzione "notifica" che stampa il PID del figlio. Quando uno dei figli trova il carattere, l'applicazione termina.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>

void P_Func(char* f, char C){
    printf("Il processo %d inizia\n", getpid());
    int fd = open(f, O_RDONLY);
    char c;
    do read(fd, &c, sizeof(char));
    while(c!=C && c!=EOF);
}

void notifica(pid_t P_Winner){
    printf("Il processo %d ha vinto\n", P_Winner);
}

void Padre_Func(pid_t P1_PID, pid_t P2_PID){
    pid_t P_Winner = wait(NULL);
    if(P_Winner==P1_PID) kill(P2_PID, SIGKILL);
    else kill(P1_PID, SIGKILL);
    notifica(P_Winner);
}

int main(int argc, char** argv){
    if(argc!=6) printf("Usage: %s <i> <j> <f> <C1> <C2>\n", argv[0]), exit(1);
```

```
    char* f = argv[3];
    char C1 = argv[4][0];
    char C2 = argv[5][0];

    pid_t P1_PID = fork();
    if(P1_PID==0) P_Func(f, C1);
    else{
        pid_t P2_PID = fork();
        if(P2_PID==0) P_Func(f, C2);
        else Padre_Func(P1_PID, P2_PID);
    }

}
```