Esame del 06/03/2023 ([GitHub](https://github.com/davide-dipierro/Prove-svolte-LSO/tree/main/06032023_B))

**Esercizio 1**

*Utilizzando opportuni comandi in concatenazione si eseguano le seguenti operazioni:*

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

ls -l | awk '{print $9" "$3}'

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

ls -l | awk '{ if($1 ~ /^-/ && $5<1024) somma=somma+$5 } END {print somma}'

1. *Dato un file “parole.txt” stampa solo le linee con più di 10 caratteri.*

awk --field-separator= '{if(NF>10) print $0}' parole.txt

1. *Impostando una variabile d’ambiente LIST: perm, link, user, group, date (può anche essere inizializzata al di fuori del comando stesso) visualizzi il listato dei file nella directory corrente con il campo corrispondente*

export LIST="perm,link,user,group,date"

ls -l | awk -v list="$LIST" '{split(list, fields, ","); print $fields[1], $fields[2], $fields[3], $fields[4], $fields[5]}'

**Esercizio 2**

Si realizzi uno script di shell BASH “menu”, che implementi le seguenti funzioni accessibili da un menu:

1. Aggiungi verifica - Chiede all’utente gli elementi: giorno, mese, anno, nome studente, voto. E inserirli in un file di nome “verifica”. Il file deve essere creato sola la prima volta, chiamate successive, dovranno aggiungere nuove righe allo stesso file.
2. Conta - Chiede all’utente il mese e lo studente, e conta il numero di prove effettuate nel mese per lo studente dato.
3. Media - Chiede all’utente lo studente, e calcola la media dei voti delle verifiche date dallo studente.

Si rappresentino i mesi con una stringa di tre caratteri (gen, feb, mar, ecc.)

#!/bin/bash

scelta=0

while [ $scelta -ne 4 ];

do

    echo "Scegli un'opzione"

    echo "1) Aggiungi verifica"

    echo "2) Conta"

    echo "3) Media"

    echo "4) Uscita"

    read scelta

    case $scelta in

        1) echo "Inserisci il giorno: "

            read giorno

            echo "Inserisci il mese nel formato di 3 lettere (gen feb mar): "

            read mese

            echo "Inserisci l'anno: "

            read anno

            echo "Inserisci il nome dello studente: "

            read nome

            echo "Inserisci il voto: "

            read voto

            echo $giorno,$mese,$anno,$nome,$voto >> verifica

            ;;

        2) echo "Inserisci il nome dello studente: "

            read nome

            echo "Inserisci il mese in cui contare le prove: "

            read mese

            risultato=$(awk -F, -v mese="$mese" -v nome="$nome" 'BEGIN{somma=0} {if($2==mese && $4==nome) somma=somma+1} END {print somma}' verifica)

            echo "In $mese ci sono $risultato prove per $nome"

            ;;

        3) echo "Inserisci il nome dello studente: "

            read nome

            somma=$(awk -F, -v nome="$nome" 'BEGIN{somma=0} {if($4==nome) somma=somma+$5} END {print somma}' verifica)

            numero=$(awk -F, -v nome="$nome" 'BEGIN{somma=0}{if($4==nome) somma=somma+1} END {print somma}' verifica)

            media=$((somma/numero))

            echo "La media è: $media"

            ;;

        4) echo "Uscita"

            ;;

        \*) echo "Scelta non valida"

            ;;

    esac

done

**Esercizio 3**

*Si scriva un programma in C che prende in input i seguenti valori: N N1 N2 C, dove:*

* *N, N1, N2 sono interi positivi*
* *C è il nome di un file eseguibile (presente nel PATH)*

*Il processo iniziale ‘padre’ (P0) deve creare 2 processi figli: P1 e P2, dopodiché:*

* *Il figlio P1 deve aspettare N1 secondi e successivamente eseguire il comando C;*
* *Il figlio P2 dopo N2 secondi dalla sua creazione dovrà provocare la terminazione del processo fratello P1 e successivamente terminare; nel frattempo P2 deve periodicamente sincronizzarsi con il padre P0 (si assuma la frequenza di 1 segnale al secondo).*
* *Il padre P0, dopo aver creato i figli, si pone in attesa di segnali da P1: per ogni segnale ricevuto, dovrà stampare il proprio pid; al N-simo segnale ricevuto dovrà attendere la terminazione dei figli e successivamente terminare*

*Scegliere un comando semplice da eseguire, es. ls o pwd. Stampare a video i diversi comportamenti.*

#include "stdio.h"

#include "stdlib.h"

#include "sys/wait.h"

#include "unistd.h"

#include "signal.h"

void signal\_handler(int s){ ; }

int main (int argc, char\*\* argv) {

    printf("\nINIZIO PROGRAMMA\n");

    int N = atoi(argv[1]);

    int N1 = atoi(argv[2]);

    int N2 = atoi(argv[3]);

    char\* comando = argv[4];

    int pid1 = fork();

    if(pid1==0){

        printf("P1: Inizia l'esecuzione.\n");

        fflush(stdout);

        sleep(N1);

        printf("P1: Terminata l'attesa.\n");

        execlp(comando, comando, NULL);

        printf("Errore nell'esecuzione del comando.\n");

        return 0;

    }

    int pid2 = fork();

    if(pid2==0){

        printf("P2: Inizia l'esecuzione.\n");

        while(N2){

            kill(getppid(), SIGUSR1);

            N2--;

            printf("P2: Segnale USR1 inviato a P0.\n");

            sleep(1);

        }

        kill(pid1, SIGKILL);

        printf("P2: Segnale KILL inviato a P1.\nP2: Termino.\n");

        return 0;

    }

    signal(SIGUSR1, signal\_handler);

    printf("P0: Inizia l'esecuzione.\n");

    while(N) {

        pause();

        N--;

    }

    printf("P0: Ricevuti tutti o segnali. Attendo P1 e P2.\n");

    while(wait(NULL)>0) ;

    printf("P0: Termino.\n");

    return 0;

}