Esame del 13/07/2023 ([GitHub](https://github.com/davide-dipierro/Prove-svolte-LSO/tree/main/13072023_A))

**Esercizio 1**

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

1. *Utilizzando sed sostituire tutte le occorrenze (case insensitive) di "LSO" con "LSO-2023" in un file di nome "esami\_sostenuti.txt".*

sed -i 's/LSO/LSO-2023/i' esami\_sostenuti.txt

1. *Usando awk stampare tutte il numero delle linee del "esami\_sostenuti.txt" che contiene il pattern "LSO"*

awk 'BEGIN{somma=0} { if($0 ~ /LSO/) somma++ } END{print somma}' esami\_sostenuti.txt

1. *Usando awk, stampare la somma della dimensione dei file della directory corrente*

ls -l | awk 'BEGIN{somma=0} {if($1 ~ /^-/) somma+=$5} END{print somma}'

1. *Dato un file avente contenuti "nome- cognome" restituire a video o in un altro file, i nomi e cognomi in maniera inversa e con nome proprio abbreviato. Es. "alessandra rossi"-> "rossi, a."*

awk -F'-' '{print $2", "$1}' nomicognomies1.txt | sed -E 's/, ([^ ])([^ ])\*/, \1./'

**Esercizio 2**

*Scrivere uno script BASH che confronta il contenuto di due directory non vuote, e per ogni elemento differente in una directory, questo viene creato nell’altra. Stampare il contenuto di ogni directory, la dimensione di ogni directory, e stampare la lista di file/directory creati.*

#!/bin/bash

# Verifica se sono state fornite due directory come argomenti

if [ "$#" -ne 2 ]; then

    echo "Usage: $0 <directory1> <directory2>"

    exit 1

fi

dir1="$1"

dir2="$2"

lista\_copiati=()

# Funzione per confrontare il contenuto delle directory e copiare i file mancanti

compare\_and\_copy() {

    local source\_dir="$1"

    local dest\_dir="$2"

    # Itera su tutti gli elementi nella directory di origine

    for item in "$source\_dir"/\*; do

        base\_name=$(basename "$item")

        # Verifica se l'elemento esiste nella directory di destinazione

        if [ ! -e "$dest\_dir/$base\_name" ]; then

            # Copia l'elemento mancante nella directory di destinazione

            cp -r "$item" "$dest\_dir/"

            # Aggiungi il file alla lista dei file copiati

            lista\_copiati+=("$base\_name")

        fi

    done

}

# Esegue la funzione per entrambe le directory

compare\_and\_copy "$dir1" "$dir2"

compare\_and\_copy "$dir2" "$dir1"

# Stampa il contenuto di ogni directory

echo -e "\nContenuto di $dir1:"

ls -l "$dir1"

echo -e "\nContenuto di $dir2:"

ls -l "$dir2"

# Stampa le dimensioni delle directory

echo -e "\nDimensione di $dir1:"

ls -l | awk -v dir1="$dir1" '$9 ~ dir1 {print $5}'

echo -e "\nDimensione di $dir2:"

ls -l | awk -v dir2="$dir2" '$9 ~ dir2 {print $5}'

# Stampa i file copiati

echo -e "\nFile copiati:"

for file in "${lista\_copiati[@]}"; do

    echo "$file"

done

**Esercizio 3**

*Realizzare un programma C il cui processo iniziale P0 prende in input un numero N. Il processo P0 crea una gerarchia di processi di profondità N+1 (figlio P1- nipote P2- bisnipote P3- ....-PN+1). A partire dall’ultimo generato, ogni processo della gerarchia avrà il seguente comportamento:*

* *se il suo pid è pari, esegue un comando ls*
* *se il pid è dispari, termina ed invia un segnale al padre P0*

*Il P0 raccoglie lo stato di terminazione di tutti i figli, li stampa a video, e termina la propria esecuzione.*

*Tutte le operazioni rilevanti devono essere stampate a video.*

#include "stdio.h"

#include "unistd.h"

#include "signal.h"

#include "stdlib.h"

#include "sys/wait.h"

void signal\_handler(int signal){

    printf("Padre: Ricevuto segnale da figlio con pid dispari.\n");

}

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

    int N = atoi(argv[1]);

    signal(SIGUSR1, signal\_handler);

    for(int i=0; i<N; i++){

        printf("Nascita figlio %d-esimo.\n", i);

        int pid = fork();

        if(pid==0){

            if(getpid()%2==0){

                execlp("ls", "ls", NULL);

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

                exit(1);

            }else{

                printf("Inviato segnale al padre.\n");

                kill(getppid(), SIGUSR1);

                return 0;

            }

        }

    }

    // Raccogli e stampa lo stato di terminazione dei figli

    int status;

    for(int i=0; i<N; i++){

        wait(&status);

        printf("Figlio %d-esimo terminato con stato %d.\n", i, status);

    }

    return 0;

}