Esame del 13/07/2023 (GitHub)

Esercizio 1

Utilizzando opportuni comandi in concatenazione si eseguano le seguenti operazioni:

(a) *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./'
```

(b) *Utilizzando sed sostituire tutte le occorrenze (case sensitive) di "LSO" con "LSO-2023" in un file di nome "esami_sostenuti.txt".*

```
sed -i 's/LSO/LSO-2023/g' esami_sostenuti.txt
```

(c) *Usando awk stampare tutte il numero delle linee del "esami_sostenuti.txt" che non contiene il pattern "LSO"*

```
awk 'BEGIN{somma=0} { if($0 ~ /LSO/) somma++ } END{print NR-somma}' esami_sostenuti.txt
```

(d) Usando awk, stampare i nomi dei file aventi dimensione inferiore della directory corrente

```
size=$(ls -l | awk 'BEGIN{somma=0} {if($1 ~ /^-/) somma+=$5} END{print somma}') ;
ls -l | awk -v size=$size '{if($5<size) print $9}'
```

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.

```bash
#!/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

Si scriva un programma in C dove il processo padre P0 crea N processi figli (P1, P2...PN) con N dato in input dall'utente. Tutti i processi figli una volta creati restano in attesa di un segnale dal padre, quando ricevuto il segnale eseguono un comando "ls". Il processo P0 attiva il comportamento in base al proprio pid.

- *Se il ppid è pari attiva i figli con pid pari, e termina subito quelli con pid dispari*
- *Se il ppid è dispari attiva i figli con pid dispari, e termina subito quelli con pid pari*

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 "unistd.h"
#include "stdlib.h"
#include "stdio.h"
#include "signal.h"
#include "stdbool.h"

void signal_handler(int s){
    printf("Svegliato un %s\n", (getpid()%2) ? "dispari" : "pari");
}

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

    int N = atoi(argv[1]);
    int* pids = (int*)malloc(N*sizeof(int));

    for(int i=0; i<N; i++){
        pids[i] = fork();
        if(pids[i]==0){
            signal(SIGUSR1, signal_handler);
            pause();
            execlp("ls", "ls", NULL);
            exit(0);
        }
    }

    bool dispari = (getppid()%2) ? true : false;
    for(int i=0; i<N; i++){
        if(dispari){
            if(i%2) kill(pids[i], SIGUSR1);
            else kill(pids[i], SIGKILL);
        } else {
            if(i%2) kill(pids[i], SIGKILL);
            else kill(pids[i], SIGUSR1);
        }
    }

    sleep(1);
    return 0;
}
```