Esame del 13/06/2023 (GitHub)

Esercizio 1

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

A. Utilizzando awk si scriva un comando che stampi solo le linee di un file "parole.txt" che iniziano con la parola "LSO".

```
awk '{if( $0 ~ /^LSO/) print $0}' parole.txt
awk '/^LSO/ {print}' parole.txt
```

B. *Usando grep si stampino tutte le linee che terminano con "700".*

```
grep "700$" parole.txt
```

C. *Dato un file contenente i compleanni di un gruppo di amici, usando sed restituire le linee degli amici aventi il compleanno tra novembre e dicembre.*

```
sed -n '/..-\(11\|12\)-..../p' compleanni.txt
```

D. *Data una file contente nomi, cognome e data di nascita, usando awk restituire tutti i cognomi che iniziano con R.*

```
awk '{if ($2 ~ /^R/) print}' nomicognomi.txt
awk '$2 ~ /^R/' nomicognomi.txt
```

Esercizio 2

Dato un file di testo "paghe.txt" con almeno 6 righe di testo, scrivere uno script "stipendi" che inserisca il titolo "Sig.re" se si tratta di un uomo, e il titolo "Sig.ra" se si tratta di una donna, prima del nome. Calcolare e mostrare a video lo stipendio minimo, massimo e medio del personale, e aggiungere un bonus di x euro (dove x viene inserito dall'utente) allo stipendio minimo. Il file dovrà contenere i seguenti campi: nome, cognome, genere, stipendio, anno di assunzione.

```
#!/bin/bash

aggiungi_titolo(){
    if [ "$1" == "M" ]; then
        echo "Sig.re $2"
    elif [ "$1" == "F" ]; then
        echo "Sign.ra $2"
    else
        echo $2
    fi
}

stipendio_minimo=9999999
stipendio_massimo=0
media_stipendi=0
numero_dipendenti=0
while IFS=, read -r  nome cognome genere stipendio assunzione; do
    nome_completo=$(aggiungi_titolo "$genere" "$cognome")
    if [ "$stipendio" -lt "$stipendio_minimo" ]; then
            stipendio_minimo="$stipendio"
    fi
    if [ "$stipendio" -gt "$stipendio_massimo" ]; then
        stipendio_massimo="$stipendio";
    fi
    media_stipendi=$(( media_stipendi + stipendio ))
    numero_dipendenti=$(( numero_dipendenti + 1 ))
echo "$nome_completo, $nome, $genere, $stipendio, $assunzione"
done < paghe.txt
echo "Stipendio massimo: $stipendio_massimo"
echo "Stipendio minimo: $stipendio_minimo"
media_stipendi=$(( media_stipendi / numero_dipendenti ))
echo "Media stipendi: $media_stipendi"
```

```
read -p "Inserisci il bonus per lo stipendio minimo: " bonus
read -p "Si vuole modificare il file incrementando lo stipendio minimo? (y/n) " modifica

if [ "$modifica" == "y" ]; then
    sed -i "s/$stipendio_minimo/$((stipendio_minimo+bonus))/g" paghe.txt
else
     sed "s/$stipendio_minimo/$((stipendio_minimo+bonus))/g" paghe.txt
fi
```

Esercizio 3

Realizzare un programma C che, utilizzando le system call di UNIX, che prevede: esame F N C:

- *F rappresenta il nome assoluto di un file*
- *N rappresenta un intero*
- *C rappresenta un carattere*

Il processo iniziale Il processo iniziale P0 deve creare un numero due processi figli P1 e P2. P1 legge una parte del file F: in particolare, se L è la lunghezza del file F, il figlio dovrà leggere una frazione di L/N caratteri dal file F, e lo invia al figlio P2. Il processo P1 leggerà quindi una frazione di F allo scopo di calcolare il numero delle occorrenze del carattere C nella parte di file esaminata; al termine della scansione, P1 comunicherà al padre il numero delle occorrenze di C incontrate nella frazione di file assegnatagli. Il figlio P2 conferma al padre il numero di occorrenze di C del pezzo inviatogli. Il padre P0, una volta ottenuti i risultati da tutti i figli, stamperà il numero totale di occorrenze di C nel file F e terminerà. I diversi passaggi devono essere mostrati a video.

```
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include "sys/wait.h"

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


    int N = atoi(argv[2]);
    char C = argv[3][0];

    // Creazione pipe
    int pipe1[2], pipe2[2];
    if(pipe(pipe1) == -1 || pipe(pipe2) == -1) printf("Errore: impossibile creare le pipe\n"), return -1;

    // Creazione processo figlio 1
    int pid1 = fork();
    if(pid1 == 0){
        printf("P1 INIZIA\n");

        // Chiusura pipe inutilizzate
        close(pipe1[0]);
        close(pipe2[0]);

        // Apertura file
        FILE* file = fopen(argv[1], "r");

        printf("P1 CALCOLA LUNGHEZZA DA LEGGERE\n");
        fseek(file, 0, SEEK_END);
        int length = ftell(file);
        fseek(file, 0, SEEK_SET);
        int pieceLength = length / N;
        if(length % N != 0)
            pieceLength++;

        // Allocazione buffer
        char* buffer = malloc(pieceLength * sizeof(char));

        printf("P1 LEGGE IL PEZZO\n");
        fseek(file, 0, SEEK_SET);
        fread(buffer, sizeof(char), pieceLength, file);
        fclose(file);

        printf("P1 INVIA IL PEZZO\n");
        write(pipe1[1], buffer, pieceLength);
```

```
        printf("P1 CONTA LE OCCORRENZE DI %c\n", C);
        int occurrences = 0;
        for(int i = 0; i < pieceLength; i++)
            if(buffer[i] == C)
                occurrences++;
        printf("P1 HA TROVATO %d OCCORRENZE DI %c\n", occurrences, C);

        printf("P1 INVIA IL NUMERO OCCORRENZE A P0\n");
        write(pipe2[1], &occurrences, sizeof(int));
        free(buffer);

        printf("P1 TERMINA\n");
        close(pipe1[1]);
        return 0;
    }

    // Creazione processo figlio 2
    int pid2 = fork();
    if(pid2 == 0){
        printf("P2 INIZIA\n");

        // Chiusura pipe inutilizzate
        close(pipe1[1]);
        close(pipe2[0]);

        // Allocazione buffer
        char* buffer = malloc(1024 * sizeof(char));

        // Lettura pezzo
        printf("P2 LEGGE DALLA PIPE IL PEZZO\n");
        int readLength = read(pipe1[0], buffer, 1024);

        printf("P2 CALCOLA IL NUMERO OCCORRENZE DI %c\n", C);
        int occurrences = 0;
        for(int i = 0; i < readLength; i++)
            if(buffer[i] == C)
                occurrences++;
        printf("P2 HA TROVATO %d OCCORRENZE DI %c\n", occurrences, C);


        printf("P2 INVIA IL NUMERO OCCORRENZE A P0\n");
        write(pipe2[1], &occurrences, sizeof(int));
        free(buffer);

        printf("P2 TERMINA\n");
        close(pipe1[0]);
        close(pipe2[1]);
        return 0;
    }

    printf("P0 INIZIA\n");
    // Chiusura pipe inutilizzate
    close(pipe1[0]);
    close(pipe1[1]);
    close(pipe2[1]);

    printf("P0 ASPETTA CHE P1 E P2 TERMININO\n");
    waitpid(pid1, NULL, 0);
    waitpid(pid2, NULL, 0);

    printf("P0 LEGGE IL NUMERO OCCORRENZE DA P1\n");
    int occurrences1;
    read(pipe2[0], &occurrences1, sizeof(int));
    printf("P0 LEGGE IL NUMERO OCCORRENZE DA P2\n");
    int occurrences2;
    read(pipe2[0], &occurrences2, sizeof(int));
    printf("P1 E P2 HANNO TROVATO LO STESSO NUMERO DI %c? %s\n", C, occurrences1==occurrences2 ? "true" :
"false");

    return 0;
}
```