Esame del 20/01/2023 ([GitHub](https://github.com/davide-dipierro/Prove-svolte-LSO/tree/main/20012023_A))

**Esercizio 1**

*Si supponga di avere un file “sup\_groups.txt” le cui righe contengono informazioni su gruppi unix strutturate nel seguente modo: "nome\_gruppo:password:id\_gruppo:utente1,utente2..."*

*Di seguito viene riportato un esempio:*

adm:x:4:syslog,adm1

admins:x:1006:adm2,adm12,manuel

ssl-cert:x:122:postgres

alan2:x:1009:aceto,salvemini conda:x:1011:giovannelli,galise,aceto,caputo,haymele,salvemini,scala,adm2,adm12

adm1Group:x:1022:adm2,adm1,adm3

docker:x:998:manuel

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

1. *Elencare i nomi di tutti gli utenti del file*

awk -F : '{ print $4 }' sup\_groups.txt

1. *Contare il numero di utenti appartenenti al gruppo “admins” e “adm”*

awk -F : '{ if($1=="admins" || $1=="adm") print $4 }' sup\_groups.txt | awk -F , '{ SOMMA+=NF } END { print SOMMA }'

1. *Elencare il GID dei gruppi con almeno 2 utenti*

awk -F , '{ if(NF>2) print }' sup\_groups.txt | awk -F : '{ print $3 }'

1. *Elencare il gruppo con il maggior numero di utenti*

awk -F , 'BEGIN{ MAX\_USER=0; ID\_GROUP=0 } { if(NF>MAX\_USER) { MAX\_USER=NF; ID\_GROUP=$1 } print ID\_GROUP } END { print ID\_GROUP}' sup\_groups.txt | awk -F : 'END { print $3 }'

**Esercizio 2**

Si realizzi uno script di shell BASH “groups”, che accetta come argomento un file “groups\_file.txt” strutturato nel seguente modo:

adm:x:4:syslog,adm1

admins:x:1006:adm2,adm12,manuel

ssl-cert:x:122:postgres

alan2:x:1009:aceto,salvemini conda:x:1011:giovannelli,galise,aceto,caputo,haymele,salvemini,scala,adm2,adm12

adm1Group:x:1022:adm2,adm1,adm3

docker:x:998:manuel

che:

1. stampa il numero massimo di campi di una linea in un dato file
2. crea una sottodirectory per ogni gruppo presente nel file, dando accessi di lettura e scrittura agli utenti dei gruppi “adm” e “admins”
3. crea un file per ogni sotto directory contenente gli utenti che appartengono a quel gruppo riga per riga

#!/bin/bash

# Verifica che sia stato fornito un file come argomento

if [ "$#" -ne 1 ]; then

    echo "Usage: $0 <groups\_file>"

    exit 1

fi

groups\_file="$1"

# (a) Stampa il numero massimo di campi di una linea in un dato file

max\_fields=$(awk -F':' '{print NF}' "$groups\_file" | sort -n | tail -n 1)

echo "(a) Numero massimo di campi in una linea: $max\_fields"

# (b) Crea una sottodirectory per ogni gruppo e assegna permessi agli utenti di "adm" e "admins"

while IFS=":" read -r group\_name x gid users; do

    dir\_name="subdir\_$group\_name"

    mkdir -p "$dir\_name"

    group\_members=$(echo "$users" | sed 's/,/ /g')

    # Imposta i permessi sulla directory

    chmod 770 "$dir\_name"

    chown :"$group\_name" "$dir\_name"

    # Imposta i permessi sugli utenti di "adm" e "admins"

    for user in $group\_members adm admins; do

        usermod -aG "$group\_name" "$user"

        chmod +rw "$dir\_name" # Aggiunge permessi di lettura e scrittura

    done

done < "$groups\_file"

# (c) Crea un file per ogni sottodirectory contenente gli utenti del gruppo riga per riga

while IFS=":" read -r group\_name x gid users; do

        dir\_name="subdir\_$group\_name"

        echo "$users" | tr ',' '\n' > "$dir\_name/users.txt"

done < "$groups\_file"

echo "Script completato con successo."

**Esercizio 3**

Si implementi un programma C che accetta come argomento il path di un file “f\_input”, una stringa “tabu”, una coppia di interi positivi “i” e “j” con i < j.

Il programma stampa a video il contenuto del file di input sostituendo tutte e sole le occorrenze di “tabu”, a partire dalla linea “i-esimo” fino al “j-esimo”, con una stringa contenente tanti asterischi quanti sono i caratteri della stringa “tabu”. Se il file di input contiene meno di “j” linee, la sostituzione avviene fino alla fine del file.

#include "unistd.h"

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

#include "stdbool.h"

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

    FILE \*file = fopen(argv[1], "rw+");

    char \*tabu = argv[2];

    size\_t len = strlen(tabu);

    int i = atoi(argv[3]);

    int j = atoi(argv[4]);

    if (file == NULL || i >= j || i < 0 || j < 0 || argc != 5) printf("Errore\n"), exit(1);

    fseek(file, i, SEEK\_SET);

    while(1) {

        if (ftell(file) == j) break;

        char c = fgetc(file);

        if (c == EOF) break;

        if (c == tabu[0]) {

            bool found = true;

            for (int k = 1; k < len; k++) {

                if (fgetc(file) != tabu[k]) {

                    fseek(file, -k, SEEK\_CUR);

                    found = false;

                    break;

                }

            }

            if (found) {

                printf("Trovato %s\n", tabu);

                fseek(file, -len+1, SEEK\_CUR);

                for (int k = 0; k < len; k++) {

                    fprintf(file, "\*");

                }

            }

        }

    }

    fclose(file);

    return 0;

}