



# Simulazione

---

*Roberto De Virgilio*

*Sistemi operativi - 8 Novembre 2017*

# Simulazionei: gestione file in linguaggio C

Si consideri il file di testo `social_users.txt` contenente le seguenti righe:

```
Paul verde rosso giallo  
Mark giallo marrone verde  
John blu giallo rosso  
Bob blu rosso verde  
Alice giallo verde rosso  
Max marrone nero bianco
```

Ogni riga contiene quattro stringhe separate da spazio con il seguente schema:

**NomePersona Colore1Preferito Colore2Preferito Colore3Preferito**

Scrivere in linguaggio C la seguente funzione

```
int modificaColore (Stringa nome_file, Stringa colore_corrente, Stringa colore_nuovo)
```

Tale funzione deve sostituire nel file di testo `nome_file` tutte le occorrenze di `colore_corrente` con `colore_nuovo` e restituire il numero di sostituzioni effettuate.

Si supponga la dichiarazione

```
typedef char Stringa[30];
```

per definire il tipo `Stringa` da utilizzare all'interno della funzione.

Ad esempio `modificaColore("social_users.txt", "giallo", "viola")` ritornerà il valore **4** e modificherà il file come segue

```
Paul verde rosso viola  
Mark viola marrone verde  
John blu viola rosso  
Bob blu rosso verde  
Alice viola verde rosso  
Max marrone nero bianco
```

# Simulazionei: gestione file in linguaggio C

```
#include <stdio.h>
#include <string.h>

typedef char Stringa[30];
```

# Simulazionei: gestione file in linguaggio C

```
int modificaColore (Stringa nome_file, Stringa colore_corrente, Stringa colore_nuovo) {  
    int num_s = 0;  
    long pos;  
    Stringa nome, c1, c2, c3;  
    FILE *fp = fopen(nome_file, "r+");  
    while (!feof(fp)) {  
        pos = ftell(fp);  
        fscanf (fp, "%s %s %s %s\n", nome, c1, c2, c3);  
    }  
}
```

# Simulazionei: gestione file in linguaggio C

```
if (strcmp(colore_corrente,c1)==0) {  
    fseek (fp,pos,SEEK_SET);  
    fprintf(fp, "%s %s %s %s\n", nome, colore_nuovo, c2, c3);  
    num_s++;  
}  
  
if (strcmp(colore_corrente,c2)==0) {  
    fseek (fp,pos,SEEK_SET);  
    fprintf(fp, "%s %s %s %s\n", nome, c1, colore_nuovo, c3);  
    num_s++;  
}  
  
if (strcmp(colore_corrente,c3)==0) {  
    fseek (fp,pos,SEEK_SET);  
    fprintf(fp, "%s %s %s %s\n", nome, c1, c2, colore_nuovo);  
    num_s++;  
}
```

## Simulazionei: gestione file in linguaggio C

```
    }
    fclose(fp);
    return num_s;
}
```

# SimulazioneI: gestione file tramite comandi Linux

Si consideri il file di testo `indirizzi.txt` contenente informazioni di contatto di vario genere e in particolare indirizzi di posta elettronica come ad esempio:

`daren103@example.org`  
`t35@example.org`  
`a3z@example.org`  
`d10@example.org`

Definire un comando linux per stampare solo le righe che contengono un indirizzo di posta elettronica (quindi contengono il carattere @) tale per cui:

- l'indirizzo di posta finisce con la stringa ".org"
- l'account indicato nell'indirizzo (cioè il nome indicato prima del carattere @) sia composto come segue:
  - abbia solo 3 caratteri
  - il primo carattere è una lettera compresa tra a e d
  - il secondo carattere è un numero compreso tra 1 e 4

Rispetto all'esempio precedente l'esecuzione del comando stamperà su output le seguenti righe:

`a3z@example.org`  
`d10@example.org`

## Simulazionei: gestione file tramite comandi Linux

- ◆ `grep '^a..d][1234].@.*[.]org$'`  
`indirizzi.txt`

a3z@example.org

d10@example.org

# SimulazioneI: manipolazione file tramite script AWK

Si consideri il file [passwords.txt](#) in cui ogni riga segue il seguente schema:

**User:~~\*~~:UID:GID:USER\_DESCRIPTION:Home\_Directory:Shell**

si scriva uno script in linux tale per cui si produca il seguente output

User	UID	GID	Home Directory	Shell
nobody	-2	-2	/var/empty	/usr/bin/false
root	0	0	/var/root	/bin/sh
daemon	1	1	/var/root	/usr/bin/false
_uucp	4	4	/var/spool/uucp	/usr/sbin/uucico
_taskgated	13	13	/var/empty	/usr/bin/false
_networkd	24	13	/var/networkd	/usr/bin/false
_lp	26	13	/var/spool/cups	/usr/bin/false
_scsd	31	31	/var/empty	/usr/bin/false
_ces	32	54	/var/empty	/usr/bin/false
_mcxalr	54	54	/var/empty	/usr/bin/false
_appleevents	55	55	/var/empty	/usr/bin/false
_geod	56	56	/var/db/geod	/usr/bin/false

GID: 13 with 3 users

GID: 54 with 2 users

in cui vengano formattate le prime 12 righe del file [passwords.txt](#) considerando solo i campi

**User, UID, GID, Home\_Directory e Shell**

e a seguire si calcoli per ogni **GID** quanti utenti (**User**) hanno quel **GID**. Si stampino solo i **GID** che hanno per lo meno **2** utenti

# Simulazione: manipolazione file tramite script AWK

```
#!/usr/bin/awk -f

BEGIN {
    FS="."
    print "-----"
    printf "%-14s %-7s %-7s %-15s %-30s\n", "User", "UID", "GID", "Home Directory", "Shell"
    print "-----"
}

NR>=1 && NR<=12{
    printf "%-14s %-7d %-7d %-15s %-30s\n", $1, $3, $4, $6, $7
    GID[$4]++
}

END {
    print ""
    print "-----"
    for (i in GID){
        if (GID[i] >= 2)
            printf("GID: %d with %d users\n", i, GID[i])
    }
    print ""
}
```

# Simulazione2: gestione file in linguaggio C

Si consideri il file di testo `social_users.txt` contenente le seguenti righe:

```
Paul verde rosso giallo
Mark giallo marrone verde
John blu giallo rosso
Bob blu rosso verde
Alice giallo verde rosso
Max marrone nero bianco
```

Ogni riga contiene quattro stringhe separate da spazio con il seguente schema:

```
NomePersona Colore1Preferito Colore2Preferito Colore3Preferito
```

Scrivere in linguaggio C la seguente funzione

```
int creaBinario (Stringa nome_file, Stringa colore)
```

Tale funzione deve creare un file binario di nome "`posizione_colore.dat`" contenente, per ogni riga del file di testo `nome_file` in cui ricorre la stringa `colore` tra i tre colori preferiti, un record a due campi di cui:

- il primo campo è una stringa (il nome della persona sulla riga corrente)
- il secondo campo è un numero intero (corrispondente alla posizione del colore sulla riga - quindi 1, 2 o 3)

e alla fine tale funzione deve restituire la posizione più ricorrente del `colore` sulle righe del file di testo.

Si supponga la dichiarazione

```
typedef char Stringa[30];
```

per definire il tipo `Stringa` da utilizzare all'interno della funzione.

Ad esempio `creaBinario("social_users.txt", "verde")` ritornerà il valore `3` e creerà il file `posizione_colore.dat` come segue:

```
[Paul] [1]
[Mark] [3]
[Bob] [3]
[Alice] [2]
```

## Simulazione2: gestione file in linguaggio C

```
#include <stdio.h>
#include <string.h>

typedef char Stringa[30];

typedef struct {
    Stringa nome;
    int pos;
} record_bin;

int max3(int a, int b, int c) {
    if (a > b && a > c) return a;
    if (b > a && b > c) return b;
    if (c > a && c > b) return c;
    return a;
}
```

## Simulazione2: gestione file in linguaggio C

```
int creaBinario (Stringa nome_file, Stringa colore){
    int pos_cur, pos1, pos2, pos3;
    Stringa nome, c1, c2, c3;
    FILE* fp = fopen("nome_file","r");
    FILE* fp2 = fopen("posizione_colore.dat","ab");

    while (!feof(fp)) {
        fscanf (fp, "%s %s %s %s\n", nome, c1, c2, c3);
        record_bin rb;
        pos_cur=0;
```

## Simulazione2: gestione file in linguaggio C

```
if (strcmp(colore,c1)==0) {  
    pos_cur = 1;  
    pos1++;  
}  
  
if (strcmp(colore,c2)==0) {  
    pos_cur = 2;  
    pos2++;  
}  
  
if (strcmp(colore,c3)==0) {  
    pos_cur = 3;  
    pos3++;  
}
```

## Simulazione2: gestione file in linguaggio C

```
if (pos_cur !=0) {  
    strcpy(rb.nome, nome);  
    rb.pos = pos_cur;  
    fwrite(rb, 1, sizeof(record_bin), fp2);  
}  
}  
  
fclose(fp);  
fclose(fp2);  
return max3(pos1,pos2,pos3);  
}
```

## Simulazione2: gestione file tramite comandi Linux

Si consideri il file di testo [indirizzi.txt](#) contenente informazioni di contatto di vario genere e in particolare indicazioni sul paese di residenza

**Anaheimio, VA 92807**

**La Habra, CA 90631**

**Lakewood, CA 90713-2013**

**Placentia, AA 91631**

Definire un comando linux per stampare solo le righe che contengono indicazioni sul paese di residenza tale per cui:

- la riga inizia con una sequenza di minimo **2** lettere e massimo **9** lettere terminata dal carattere **" , "** e seguita da uno **spazio**
- la riga termina con un numero di **5** cifre e tale numero non deve contenere la cifra **0**
- nella riga dopo il carattere **" , "** e lo **spazio non** deve seguire la lettera **"C"**

Rispetto all'esempio precedente l'esecuzione del comando stamperà su output le seguenti righe:

**Placentia, AA 91631**

## Simulazione2: gestione file tramite comandi Linux

- ◆ `grep -E '^([A-z]{2,9}), [^C].*[1-9]{5}$'`  
`indirizzi.txt`

Placentia, AA 91631

## Simulazione2: manipolazione file tramite script AWK

Si consideri il file [passwords.txt](#) in cui ogni riga segue il seguente schema:

**User:~~\*~~:UID:GID:USER\_DESCRIPTION:Home\_Directory:Shell**

si scriva uno script in linux tale per cui si produca il seguente output

User	UID	GID	Home Directory	Shell
root	0	0	/var/root	/bin/sh
daemon	1	1	/var/root	/usr/bin/false
_uucp	4	4	/var/spool/uucp	/usr/sbin/uucico
_taskgated	13	13	/var/empty	/usr/bin/false
_networkd	24	13	/var/networkd	/usr/bin/false

  

GID: 13 with 2 users
GID: 1 with 1 users

in cui vengano formattate le righe comprese tra la seconda e la sesta nel file [passwords.txt](#) considerando solo i campi

**User, UID, GID, Home\_Directory e Shell**

e a seguire si calcoli per ogni GID quanti utenti (**User**) che usano la shell "**/usr/bin/false**" hanno quel GID. Si stampino solo i GID che hanno per lo meno **1** utente

## Simulazione2: manipolazione file tramite script AWK

```
#!/usr/bin/awk -f

BEGIN {
    FS=":"
    print "-----"
    printf "%-14s %-7s %-7s %-15s %-30s\n", "User", "UID", "GID", "Home Directory", "Shell"
    print "-----"
}

NR>=2 && NR<=6 {
    printf "%-14s %-7d %-7d %-15s %-30s\n", $1, $3, $4, $6, $7
    if ($7 == "/usr/bin/false") {
        GID[$4]++
    }
}

END {
    print ""
    print "-----"
    for (i in GID){
        printf("GID: %d with %d users\n",i,GID[i])
    }
}
```

