



# Final Exercises

---

*Roberto De Virgilio*

*Sistemi operativi - 6 Novembre 2017*

# Esercizio (gestione file)

- *si vogliono gestire file di testo su cui siano memorizzate una serie di record composti da un **nome** e da un **voto** tra 0-10; nel file i nomi e i voti devono essere separati da uno spazio.*

file-1

giuseppe 7

lara 6

annalisa 10

...

file-2

lorena 4

antonio 3

lorena 9

...

.....

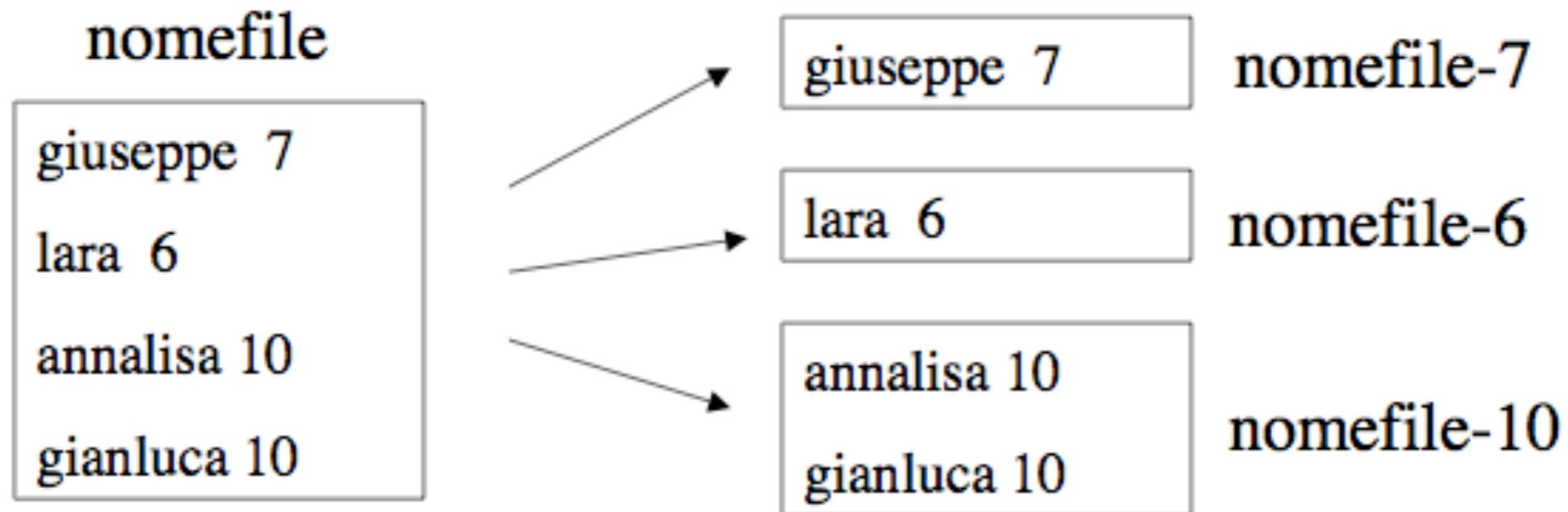
# Esercizio (gestione file)

- ◆ L'insieme di operazioni che si vogliono poter effettuare su un file sono:
  - (1) aggiunta di un **nome** e **voto** specificati dall'utente e restituzione del numero di record presenti in un dato file (compreso il nuovo elemento)
  - (2) sostituzione in un dato file del **voto** per una occorrenza di cui viene specificato **nome** e corrente **voto** dall'utente; anche il nuovo voto da assegnare è dato dall'utente
  - (3) stampa del voto massimo
  - (4) creazione (a partire da un file unico) di file più piccoli in ciascuno dei quali ci siano tutte occorrenze con lo stesso voto; il nome di ciascun file deve far riferimento al nome del file di partenza e al voto a cui fa riferimento  
**(esempio: *a.txt* → *a-10.txt*, *a-9.txt*, *a-8.txt*, ...)**

# Esercizio (gestione file)

- ◆ L'insieme di operazioni che si vogliono poter effettuare su un file sono:

(4) *creazione (a partire da un file unico) di file più piccoli in ciascuno dei quali ci siano tutte occorrenze con lo stesso voto; il nome di ciascun file deve far riferimento al nome del file di partenza e al voto a cui fa riferimento (esempio: **a.txt** → **a-10.txt**, **a-9.txt**, **a-8.txt**, ...)*



# Esercizio (gestione file)

- ◆ *L'insieme di operazioni che si vogliono poter effettuare su un file sono:*

```
int inserisci (Stringa nome, int voto, Stringa nomefile)
```

```
void cambiaVoto (Stringa nome, int vecchio_voto,
```

```
                int nuovo_voto, Stringa nomefile)
```

```
int maxVoto (Stringa nomefile)
```

```
void creaFiles (Stringa nomefile)
```

# Esercizio (gestione file)

*PREMESSA:*

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

typedef char Stringa[30];

Stringa voti[11] = {"0","1","2","3","4","5","6","7","8","9","10"};
```

# Esercizio (gestione file)

```
/* Inserisci un nome e un voto e ritorna il numero di elementi */

int inserisci (Stringa nome, int voto, Stringa nomefile) {
    int numelem = 0; Stringa temp;
    if ( (voto <= 10) && (voto >= 0) ) {
        FILE *fp = fopen(nomefile, "a+");
        fprintf (fp, "%s %d\n", nome, voto);

        /* conta il numero di elementi nel file */
        fseek (fp, 0, SEEK_SET);
        while (!feof(fp)) {
            fscanf (fp, "%s %d\n", temp, &voto);
            numelem++;
        }
        fclose(fp);
    }
    else numelem = -1; /* errore di inserimento */
    return numelem;
}
```

# Esercizio (gestione file)

```
/* cambia il voto all'occorrenza col nome e voto corrente specificati */

void cambiaVoto (Stringa nome, int vecchio_voto, int nuovo_voto, Stringa nomefile) {
    long pos; int voto; int test =0; Stringa temp;
    if ( (nuovo_voto <= 10) && (nuovo_voto >=0) ) {
        FILE *fp = fopen(nomefile, "r+");
        while ((!feof(fp))&&(!test)) {
            pos = ftell(fp);
            if (fscanf (fp, "%s %d\n", temp, &voto) != EOF)
                if (strcmp(nome,temp)==0 && voto == vecchio_voto) {
                    fseek (fp,pos,SEEK_SET);
                    fprintf(fp, "%s %d\n", nome, nuovo_voto);
                    test=1;
                }
        }
        fclose(fp);
    }
    else printf("errore specifica voto");
}
```

# Esercizio (gestione file)

```
/* ritorna il voto max */

int maxVoto (Stringa nomefile) {
    int voto, max = 0; Stringa nome;
    FILE *fp = fopen(nomefile,"r");
    if (!fp) return -1; /* errore di lettura file */
    while (!feof(fp) && max < 10){
        fscanf(fp,"%s %d\n", nome, &voto);
        if (voto > max) max = voto;
    }
    fclose(fp);
    return max;
}
```

# Esercizio (gestione file)

```
/* crea file piccoli */
```

```
void creaFiles(Stringa nomefile) {
    Stringa temp; int voto;
    FILE *fp = fopen(nomefile,"r");
    while (!feof(fp)) {
        fscanf (fp, "%s %d\n", temp, &voto);
        Stringa newfile;
        strcpy(newfile,nomefile);
        strcat(newfile,"-");
        strcat(newfile,voti[voto]);
        strcat(newfile,".txt");
        inserisci(temp,voto,newfile);
    }
}
```

# Esercizio (espressioni regolari)

- ◆ *Dato il file di testo esempio.txt*

15 fifteen

14 fourteen

13 thirteen

12 twelve

.....

1 one

- ◆ *scrivere un comando linux per mostrare tutte le righe di esempio.txt*

# Esercizio (espressioni regolari)

◆ *grep '.\*' esempio.txt*

15 fifteen

14 fourteen

13 thirteen

12 twelve

.....

1 one

# Esercizio (espressioni regolari)

- ◆ *Dato il file di testo esempio.txt*

15 fifteen

14 fourteen

13 thirteen

12 twelve

.....

1 one

- ◆ *scrivere un comando linux per mostrare tutte le righe di esempio.txt contenenti i caratteri 2 oppure 4 indipendentemente dalla loro collocazione nella stringa*

# Esercizio (espressioni regolari)

- ✿ *grep '[24]' esempio.txt*

14 fourteen

12 twelve

4 four

2 two

# Esercizio (espressioni regolari)

- ◆ *Dato il file di testo esempio.txt*

15 fifteen

14 fourteen

13 thirteen

12 twelve

.....

1 one

- ◆ *scrivere un comando linux per mostrare tutte le righe di esempio.txt che contengono due “e” di seguito, seguite da un carattere qualsiasi.*

# Esercizio (espressioni regolari)

- ◆ `grep 'ee.' esempio.txt`
- ◆ `grep -E 'e{2}.' esempio.txt`

15 fifteen

14 fourteen

13 thirteen

# Esercizio (espressioni regolari)

- ◆ *Dato il file di testo esempio.txt*

15 fifteen

.....

1 one

- ◆ *scrivere un comando linux per mostrare tutte le righe che:*
  - ◆ *Iniziano con il carattere “1”*
  - ◆ *Terminano con il carattere “e”*
  - ◆ *Presentano al loro interno zero o più caratteri*

# Esercizio (espressioni regolari)

- ◆ `grep '^1.*e$' esempio.txt`

12 twelve

1 one

# Esercizio (espressioni regolari)

- ◆ *Dato il file di testo esempio.txt*

15 fifteen

.....

1 one

- ◆ *scrivere un comando linux per mostrare tutte le righe che:*

- ◆ *Iniziano con una qualsiasi sequenza di caratteri*
- ◆ *deve essere presente una parola che inizia con la lettera 'f' seguita da un carattere compreso tra 'i' e 'p'*

# Esercizio (espressioni regolari)

◆ `grep -E '^.*\<f[i-p]' esempio.txt`

15 fifteen

14 fourteen

5 five

4 four

# Esercizio (espressioni regolari)

- ◆ *Dato il file di testo esempio.txt*

15 fifteen

.....

1 one

- ◆ *scrivere un comando linux per rimuove i primi 3 caratteri ad inizio linea*

# Esercizio (espressioni regolari)

◆ ***sed 's/.../' esempio.txt***

fifteen

fourteen

thirteen

twelve

...

wo

ne

# Esercizio (AWK)

- ◆ Si consideri il file 1975.txt che, riga per riga, contiene: nome di una nazione, la sua superficie in migliaia di km<sup>2</sup>, la sua popolazione in milioni di abitanti e il continente di riferimento:

|           |       |     |         |
|-----------|-------|-----|---------|
| USSR      | 86250 | 262 | Asia    |
| USA       | 3615  | 219 | America |
| Cina      | 3692  | 866 | Asia    |
| Canada    | 3852  | 24  | America |
| Brasile   | 3286  | 116 | America |
| Australia | 2968  | 14  | Oceania |
| India     | 1269  | 637 | Asia    |
| Argentina | 1072  | 26  | America |
| Sudan     | 968   | 19  | Africa  |
| Algeria   | 920   | 18  | Africa  |

# Esercizio (AWK)

- ◆ *scrivere uno script basato sull'utilizzo di AWK che stampi esclusivamente nazione e continente.*
- ◆ *scrivere uno script basato sull'utilizzo di AWK che stampi solo i dati relativi alle nazioni asiatiche.*
- ◆ *scrivere uno script basato sull'utilizzo di AWK che stampi Continente - Numero Medio di Abitanti (calcolato come somma del numero di abitanti diviso il numero di nazioni presenti nel continente).*

# Esercizio (AWK)

- ◆ *Creiamo il programma AWK che stampi esclusivamente nazione e continente:*

```
#!/bin/awk -f

BEGIN {
    printf("[Nazione] [Continente]\n");
}

{
    printf("%s %s\n", $1, $4);
}
```

# Esercizio (AWK)

- ◆ *Creiamo il programma AWK che stampi solo i dati relativi alle nazioni asiatiche:*

```
#!/bin/awk -f

BEGIN {

    printf("[Nazioni Asiatiche]\n");

    printf("[nazione] [superficie] [popolazione]\n");

}

($4 == "Asia") {

    printf("%s %d %d\n", $1, $2, $3);

}
```

# Esercizio (AWK)

- ✿ *Creiamo il programma AWK che stampi Continente - Numero Medio di Abitanti:*

```
#!/bin/awk -f
BEGIN {
    printf("[Continente] [Numero Medio Abitanti]\n");
}
{
    abitanti[$4] += $3;
    nazioni[$4] += 1;
}
END {
    for (i in abitanti) {
        printf("%s %lf\n", i, abitanti[i]/nazioni[i]);
    }
}
```

