Programmazione e Laboratorio di Programmazione

Gestione dei files in linguaggio “C”

Un progetto completo

Attività:

Sviluppare in linguaggio “C”:

- **un programma che copi un file in un altro**
- **un programma che trasformi una immagine a colori nella sua equivalente in bianco e nero**
- **un programma che metta in evidenza le tonalità rosse più intense di una immagine**
- **un programma in grado di miscelare due immagini aventi la stessa risoluzione**
- **un programma che estragga dati posizionali da un file testuale e ne generi un altro in formato CSV**

mycp

Cosa farà il nostro programma nelle varie iterazioni?

- **MP1:** lettura di un file in formato noto
- **MP2:** file di uscita e uso della redirectione
- **MP3:** parametri su linea di comando
- **MP4:** lettura di un file di formato sconosciuto
- **MP5:** gestione degli errori
- **MP6:** scrittura corretta del file di uscita

MP1

- **Creare un file chiamato FileNoto.dat (utilizzando un text editor) contenente le seguenti righe:**

Mario Rossi 22

Giulia Bianchi 30

Marco Verdi 27

- **Creare un file sorgente “C” chiamato mycp1.c**

MP1(2)

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

```
main() {
```

```
    FILE *FileIn;
```

```
    char FileName [80];
```

```
    char var1 [80], var2 [80];
```

```
    int var3;
```

```
    printf ("Nome del file sorgente ?");
```

```
    scanf ("%s", FileName);
```

MP1(3)

```
FileIn = fopen (FileInName, "r");
```

```
fscanf (FileIn, "%s %s %d", var1, var2, &var3);  
printf ("%s %s %d\n", var1, var2, var3);
```

```
fscanf (FileIn, "%s %s %d", var1, var2, &var3);  
printf ("%s %s %d\n", var1, var2, var3);
```

```
fscanf (FileIn, "%s %s %d", var1, var2, &var3);  
printf ("%s %s %d\n", var1, var2, var3);
```

```
}
```

MP1(3)

- **A che serve modificare la linea:**
`printf ("Nome del file sorgente ?\n");`
in quest'altra?:
`printf ("Nome del file sorgente : ");`
- **All'interno del file FileNoto.dat cancellate la votazione di Giulia Bianchi e rilanciate il programma; cosa succede?**

MP2

- **Il programma mycp nella sua prima versione legge dati da un file e li presenta sullo schermo**
- **Esiste un modo molto semplice per inserire i dati che vengono mostrati sullo schermo all'interno di un nuovo file ... ricordate il meccanismo della redirectione?**

MP2 (2)

- **Provate il comando seguente:**

```
./mycp1 > FileNotoCopia.dat
```

Cosa succede???

- **La richiesta del nome del file è presentata sul canale **stdout** che è lo stesso canale al quale vengono inviati i dati letti dal file ...**

MP3

- **Un modo per uscire dall'impasse puo' essere quello di evitare di inviare richieste di inserimento dati alla console e utilizzare i parametri a linea di comando**
- **A questo scopo studiamo il seguente codice "C":**

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

```
main (int argc, char *argv []) {  
    int i;
```

```
    for (i=0; i<argc; i++) {  
        printf ("<Arg%d|%s>\n", i, argv [i]);  
    }  
}
```

MP3 (2)

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

```
main (int argc, char *argv []) {
```

```
    FILE *FileIn;
```

```
    char var1 [80], var2 [80];
```

```
    int var3;
```

```
    FileIn = fopen (argv [1], "r");
```

```
    fscanf (FileIn, "%s %s %d", var1, var2, &var3);
```

```
    printf ("%s %s %d\n", var1, var2, var3);
```

```
    . . . . .
```

MP3 (3)

Le cose ora sembrano funzionare ma ...

- **Controllate la lunghezza dei files origine e destinazione**
- **La soluzione proposta come si comporta se il formato del file viene cambiato?**
- **Che succede se il programma lavorava su files binari?**

MP4

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

```
main (int argc, char *argv []) {
```

```
    FILE *FileIn;  
    unsigned char ch;
```

```
    FileIn = fopen (argv [1], "rb");
```

```
    for (;;) {
```

```
        ch = fgetc (FileIn);
```

```
        if (feof (FileIn)) break;
```

```
        printf ("%c", ch);
```

```
    }
```

```
}
```

... quanto ci soddisfa?

MP4 (3)

Quella che segue è una versione piu' corretta e pulita. Per quali motivi?

```
ch = fgetc (FileIn);  
while (! feof (FileIn)) {  
    fputc (ch, FileOut);  
    Nby++;  
    ch = fgetc (FileIn);  
}
```

A cosa serve la prima **fgetc**?

MP5

- Che succede a **mycp** se viene dato come nome del file di ingresso quello di un file che non esiste?
- Che succede a **mycp** se il file di ingresso appartiene ad un altro utente o è protetto?
- Come vengono gestite eventuali anomalie di funzionamento?

MP5 (2)

Apertura del file di ingresso:

```
FileIn = fopen (FileName, "rb");  
if (FileIn == NULL) {  
    printf ("Impossibile aprire il file %s\n", FileName);  
    return (1);  
}
```

Non è ancora del tutto corretto ... **cosa non va?**

MP5 (2)

```
FileIn = fopen (FileName, "rb");  
if (FileIn == NULL) {  
    fprintf (stderr, "Impossibile aprire il file %s\n", FileName);  
    return (1);  
}
```

MP6

L'ultimo passo consiste nell'inserire nel codice sorgente le istruzioni che ci permetteranno di scrivere il file di uscita facendo a meno della redirectione

- **Inserire l'apertura del file di uscita utilizzando il secondo parametro su linea di comando**
- **Verificare la corretta scrittura dei singoli caratteri in uscita**
- **Abbiamo aperto i files ma dobbiamo anche chiuderli!!!**

MP6 (2)

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

```
int main (int argc, char *argv []) {
```

```
    FILE *FileIn, *FileOut;
```

```
    unsigned char ch;
```

```
    long int Nby = 0;
```

```
    int status;
```

MP6 (3)

```
FileIn = fopen (argv [1], "rb");  
if (FileIn == NULL) {  
    fprintf (stderr, "Impossibile aprire il file %s\n", argv [1]);  
    return (1);  
}
```

```
FileOut = fopen (argv [2], "wb");  
if (FileOut == NULL) {  
    fprintf (stderr, "Impossibile aprire il file %s\n", argv [2]);  
    return (2);  
}
```

MP6 (4)

```
ch = fgetc (FileIn);
while (! feof (FileIn)) {
    status = fputc (ch, FileOut);
    if (status != ch) {
        fprintf (stderr, "Errore scrivendo il file %s\n", argv [2]);
        return (3);
    }
    Nby++;
    ch = fgetc (FileIn);
}
```

MP6 (4)

```
fprintf (stderr, "Copiati %ld bytes\n", Nby);  
fclose (FileIn);  
fclose (FileOut);  
  
return (0);  
}
```

... Ci siamo quasi!

Ora non resta che fare il controllo di correttezza dei parametri sulla linea di comando e verificare che la funzione **fgetc funzioni correttamente ...**