C. Gaibisso
B. Martino

Programmazione e Laboratorio di Programmazione

Lezione XXII.I
La gestione dei file
Fondamenti

Che cosa è un file?

- Un file può essere visto come un contenitore di informazioni simile ad un vettore di bytes
- Quando le informazioni hanno una forma leggibile, ovvero sono costituite da elementi alfanumerici (testo suddiviso in pagine, righe e parole che hanno un significato per un operatore umano) si parla di file di tipo testuale (formattato)
- Quando si ha a che fare con informazioni più legate alla macchina, ad esempio programmi eseguibili, file musicali o immagini la cui struttura non è intuitivamente comprensibile, si parla di file binario (o non formattato)

Che cosa è un file (2)?

- Un file di testo e' costruito a partire da simboli alfanumerici codificati in modo univoco e non ambiguo sotto forma di bytes
- La codifica viene normalmente effettuata utilizzando il codice ASCII anche se è ancora diffuso in ambiente IBM (mainframes) il codice EBCDIC
- Volendo codificare delle informazioni composte da simboli o caratteri specifici di una lingua si utilizza il codice Unicode UTF-8
- UTF-8 usa da 1 a 4 byte per rappresentare un carattere (un solo byte per i caratteri ASCII)

Che cosa è un file (3)?

IMPORTANTE! Documenti scritti con un sistema di videoscrittura come ad esempio Word o OpenOffice sono visti da un operatore come testuali ma si tratta di documenti dotati di informazioni accessorie come il corpo dei caratteri usati, la loro grandezza, la forma di impaginazione e cosi via.

Un modo semplice per verificare se un documento è di tipo testuale o binario è quello di aprirlo con un text editor (ad es. NotePad di windows): se risulta leggibile e' testuale altrimenti è binario.

stdin, stdout, stderr

- Per usare un file occorre necessariamente aprirlo. L'unica eccezione a questa regola è data dai canali di comunicazione predefiniti. I canali di comunicazione predefiniti sono tre:
  - *stdin* (Standard input)
  - *stdout* (Standard output)
  - *stderr* (Standard error)

- Il canale di comunicazione *stdin* è il canale attraverso il quale il sistema operativo riceve dati in ingresso dall'operatore. In particolare, quando si è collegati ad una macchina in sessione interattiva, *stdin* è rappresentato dalla tastiera del computer.

stdin, stdout, stder

- Analogamente, *stdout* rappresenta il canale attraverso il quale il sistema operativo comunica con l'operatore. Richiamando l'esempio precedente, in una sessione intrattiva rappresenta lo schermo di un computer.

- Il canale di errore *stderr* rappresenta il canale diagnostico, ovvero il canale a cui un sistema operativo comunica errori, malfunzionamenti o avvertimenti all'operatore. Appare evidente che di solito *stdout* ed *stderr* puntano allo stesso dispositivo fisico (gli errori di solito compaiono sullo schermo).

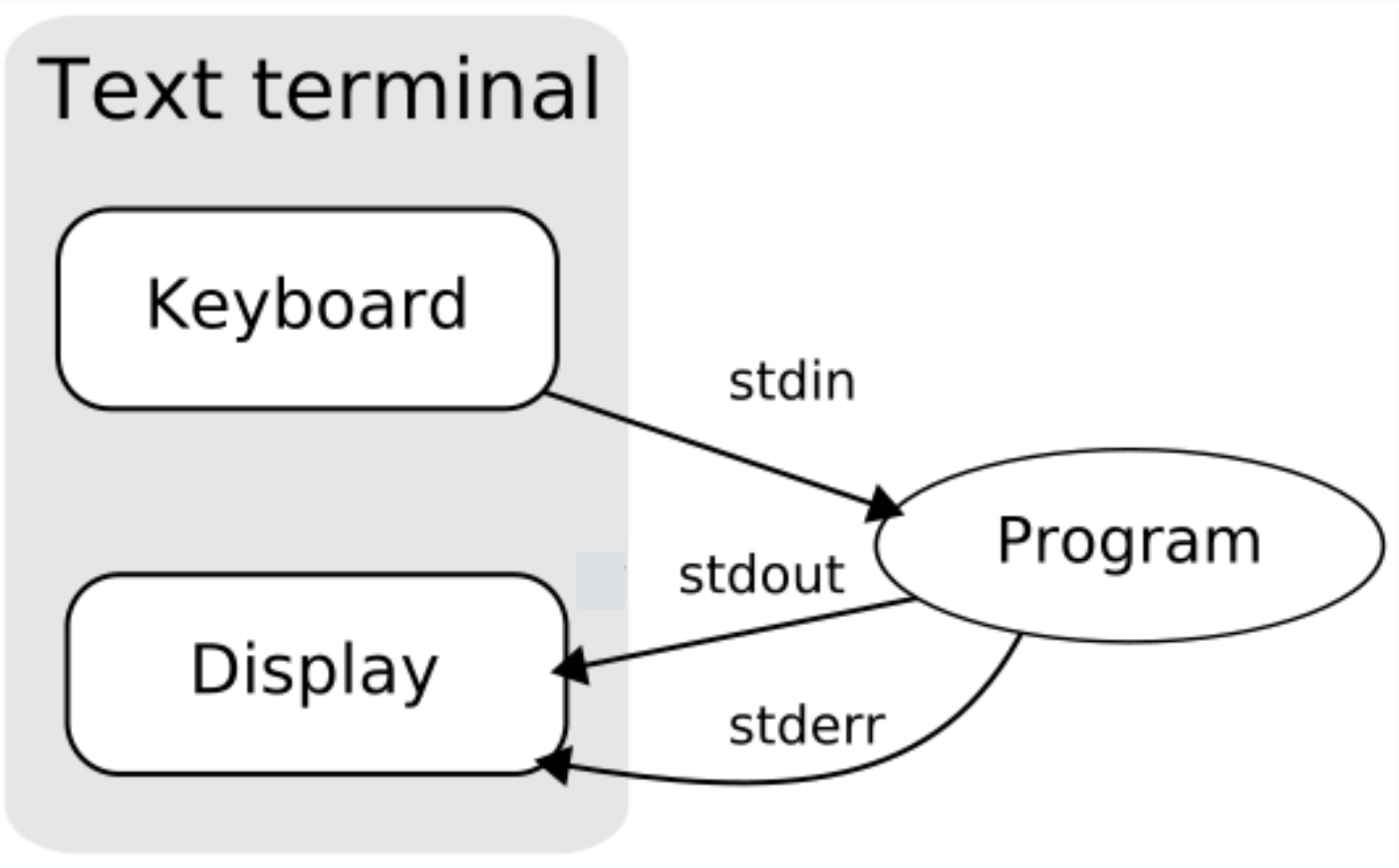
C. Gaibisso
B. Martino

stdin, stdout, stder

- Riassumendo gli standard streams sono canali di ingresso e uscita prestabiliti tra le periferiche e un programma in esecuzione:
  - stdin: standard input
  - stdout: standard output
  - stderr: standard error
- Default:
  - stdin: tastiera (buffer di memoria)
  - stdout: monitor
  - stderr: monitor

C. Gaibisso
B. Martino

Flussi standard di I/O



Redirezione dei flussi std. di I/O

- Si chiama redirezione il meccanismo attraverso il quale è possibile riassegnare i tre canali citati ad altrettanti files o dispositivi.
- Per riassegnare il canale di uscita, occorre aggiungere in coda al comando interessato il simbolo “>” seguito dal nome di un file. Per esempio:

```
# ls -l *.c > elenco
# cat elenco
-rw-r--r--   1 root     root          163 Mar 11 19:32 a.c
-rw-r--r--   1 root     root          175 Mar 11 19:34 std.c
#
```

Redirezione dei flussi std. di I/O

- Un altro esempio significativo:

```
# rm *.dat
# rm *.dat > a.out
rm: cannot remove `*.dat': No such file or directory
# rm *.dat
```

Il primo comando *rm* rimuove tutti i files con estensione .dat e non da segnalazioni sullo schermo. La ripetizione del comando, poiché lanciato sul medesimo tipo di file, provoca una segnalazione di errore. Poiché il comando di redirezione “>” funziona sul canale di uscita e non su quello di errore, la diagnostica esce sullo schermo.

Redirezione dei flussi std. di I/O

- I canali *stdin, stdout, stderr* sono associati rispettivamente agli identificativi 0 1 e 2. Premettendo al simbolo di redirezione il numero appropriato, si puo' selezionare il canale desiderato. Con riferimeto all'esempio precedente:

```
# rm *.dat
# rm *.dat 2> a.out
# cat a.out
rm: cannot remove `*.dat': No such file or directory
#
```

Redirezione dei flussi std. di I/O

stdin (0):

```
ls -1> file.txt
sort -r < file.txt
```

stdout (1):

```
pwd > DoveSono.txt
```

stderr (2):

```
echo "ONE" > one.txt
echo "TWO" > two.txt
chattr -i two.txt
rm -v one.txt two.txt 2> rm.err
```

Pipes

- Una pipe o pipeline (dall'inglese "pipe”o tubatura composta da più elementi collegati) indica un insieme di applicazioni (programmi o comandi) collegati tra loro in cascata
- Si incontra spesso nell'uso della *shell*, dove può essere conveniente riutilizzare i dati uscenti da un programma come dati in ingresso di un altro
- Il collegamento viene stabilito utilizzando il carattere “|” tra una applicazione ed un’altra
- Ad esempio, per conoscere il numero di files presenti all’interno della cartella corrente:

```
ls –l | wc -l
```

Apertura di un file

Prima di potere accedere tramite un programma ad un file è necessario aprirlo; l'istruzione che permette di aprire un file è la fopen. La sintassi di fopen è la seguente:

FileID = fopen (NomeFile, ModOpen);

Con:

FileID: Identificatore del file

NomeFile: Nome del file

ModOpen: Modalità di apertura

Apertura di un file (2)

- L'identificatore del file e' una variabile di tipo *FILE che rappresenta il canale attraverso il quale un programma accede ad un file
- Il nome del file è una stringa che ne determina nome e posizione (ovvero il percorso nel FS)
- La modalità di apertura è una stringa che contiene la lettera "r" per aprire il file in lettura (read) o la lettera "w" per aprire il file in scrittura. Se il file è di tipo binario, occorre aggiungere il carattere "b".

Apertura di un file (3)

- Esempi:
  - FILE *InFile;

    …

    InFile = fopen (“c:\casa\dati.txt”, “rb”);

  - FILE *OutFile;

    …

    OutFile = fopen (“c:\lavori\elenco.txt”, “w”);

Lettura e scrittura su files

Dopo avere aperto un file testuale è possibile leggerlo usando l'istruzione fscanf o scriverci dentro usando l'istruzione fprintf ; il comportamento delle due funzioni è del tutto analogo a quello delle funzioni scanf e printf con l'aggiunta del parametro FileID (prima della stringa di specifica del formato).

- Esempi:
  - fscanf (InFile, "%d", &dato);
  - fprintf (OutFile, "La soluzione è: %f\n", calc);

Caratteri di controllo

All'interno del codice usato per rappresentare i caratteri alfanumerici sono presenti anche semplici caratteri di controllo di formato. Nel codice ASCII i più importanti sono:

▪ CR	*Carriage Return*	A capo senza cambiare riga
▪ LF	*Line Feed*	Alla riga seguente senza andare a capo
▪ FF	*Form Feed*	Salta a pagina nuova
▪ EOF	*End Of File*	Marcatore di fine file

Caratteri di controllo

- Mentre per l'uso della printf e fprintf non ci sono particolari problemi (in quanto e' il programmatore a decidere la struttura dei dati in uscita), scanf e fscanf interpretano quello che leggono in accordo ai caratteri di controllo citati (e non solo…)
- Se nel file in ingresso è presente uno spazio bianco, scanf lo interpreta come un separatore, quindi non è in grado di leggere stringhe contenenti spazi bianchi
- L'unico modo per leggere da tastiera o da file tutti i caratteri in esso contenuti è quello di usare l'istruzione getc o fgetc

C. Gaibisso
B. Martino

Lettura carattere per carattere

- La sintassi di getc è la seguente:

Variabile_Carattere = getc ();

- Mentre quella di fgetc è la seguente:

Variabile_Carattere = fgetc (FileID);

Lettura di un file binario

- La lettura di un file binario è possibile utilizzando fgetc; ovviamente l'organizzazione e la comprensione dei dati in esso contenuti dipende esclusivamente dal programma che lo utilizza

- Supponiamo di dovere scrivere un programma in grado di effettuare la copia di un file in un altro; dato che non è in generale conosciuta a priori la lunghezza del file in questione, è necessario impostare un ciclo condizionato di lettura-scrittura un byte alla volta la cui fine sia legata alla condizione del raggiungimento della fine del file sorgente