Università degli Studi di Bologna
Scuola di Ingegneria

Corso di
Reti di Calcolatori T

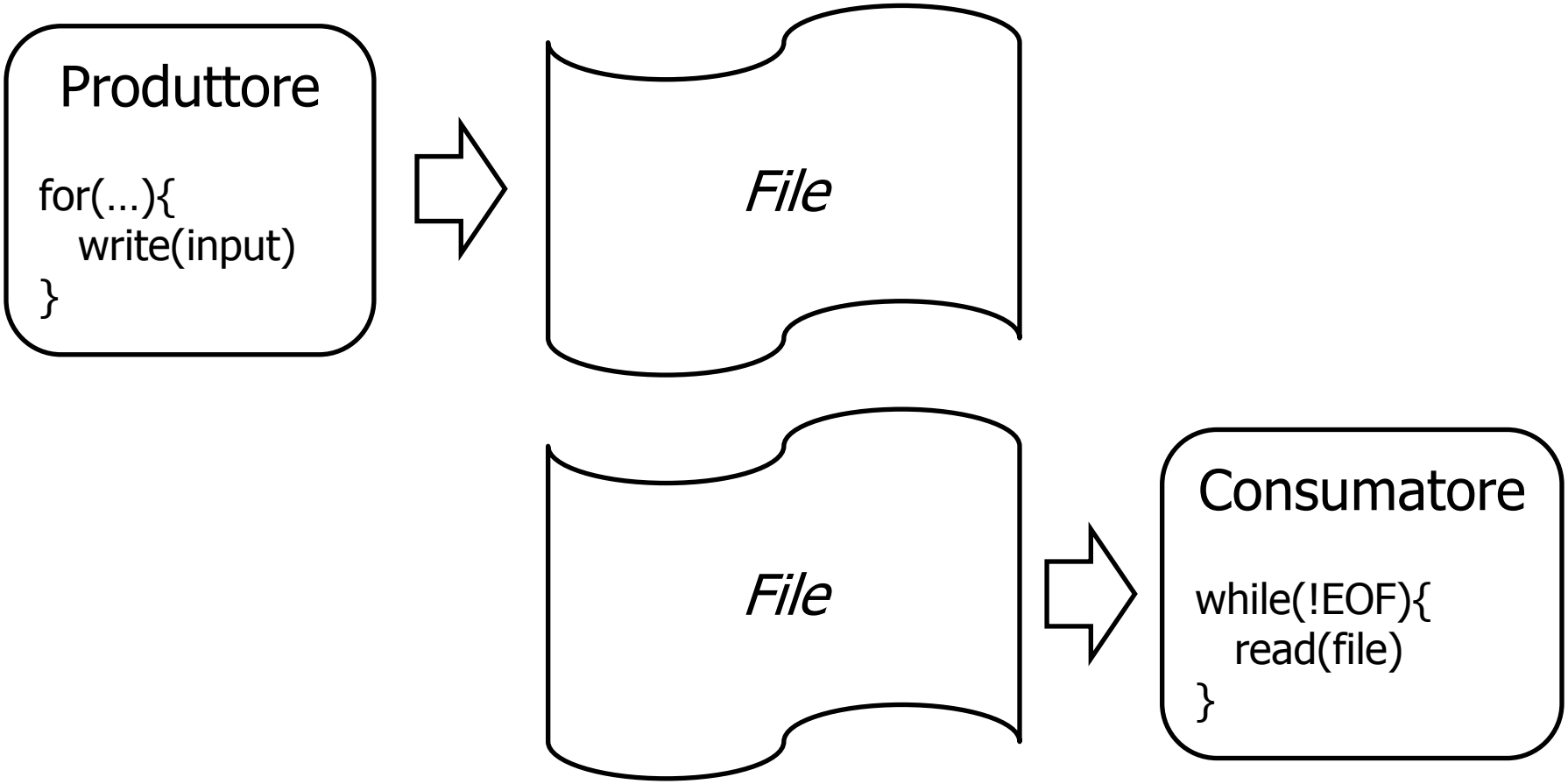
Esercitazione 0 (proposta)
Lettura e Scrittura File in Java e C

Luca Foschini

Anno accademico 2015/2016

Architettura di riferimento

Si consideri lo schema in figura…



Proposta

Modificare **la soluzione precedente** facendo in modo che nel processo **produttore** non venga chiesto all'utente quante righe scrivere, ma si legga fino a quando l'utente immette EOF (end of file) per terminare l'inserimento. Interfaccia d'uso produttore:

```
>java Produttore fileName.txt
```

Modificare il processo **consumatore** in modo da eseguire una elaborazione (**N.B.: filtro a carattere!!**) tra la lettura e la stampa a video. In questo caso, si stampino solo le righe che iniziano con una specifico carattere (o sequenza di caratteri) passato come secondo argomento. Interfaccia d'uso consumatore:

```
>java Consumatore fileName.txt prefix
```

Proposta (ancora)

Modificare il processo **consumatore** in modo da eseguire una elaborazione (filtro) tra la lettura e la stampa a video, prendendo il contenuto in input da un file ridiretto su stadanrd input. Come nel caso precedente, si stampino solo le righe che iniziano con una specifico carattere (o sequenza di caratteri) passato come argomento. Interfaccia d'uso:

```
>java Consumatore prefix < fileName.txt
```

Si modifichi il programma **consumatore** in modo da consentire l'invocazione nei due modi, senza ridirezione (2 argomenti) e con ridirezione (1 argomento).

Esempio di man

Manuale: invocato con **`>man read`**

```
READ(2)                          Linux Programmer's Manual                          READ(2)

NAME
       read - read from a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t read(int fd, void *buf, size_t count);

DESCRIPTION
       read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf.

       If  count is zero, read() returns zero and has no other results.  If count is greater than SSIZE_MAX,
       the result is unspecified.

RETURN VALUE
       On success, the number of bytes read is returned (zero indicates end of file), and the file  position
       is  advanced  by  this number.  It is not an error if this number is smaller than the number of bytes
       requested; this may happen for example because fewer bytes are actually available  right  now  (maybe
       because  we were close to end-of-file, or because we are reading from a pipe, or from a terminal), or
       because read() was interrupted by a signal.  On error, -1 is returned, and  errno  is  set  appropri-
       ately.  In this case it is left unspecified whether the file position (if any) changes.

ERRORS
 Manual page read(2) line 1 (press h for help or q to quit)
```

Proposta

open	Apre il file specificato e restituisce il suo file descriptor (fd) Crea una nuova entry nella tabella dei file aperti di sistema (nuovo I/O pointer) Fd è l'indice dell'elemento che rappresenta il file aperto nella tabella dei file aperti del processo (contenuta nella user structure del processo) Possibili diversi flag di apertura, combinabili con OR bit a bit (operatore |)
close	Chiude il file aperto Libera il file descriptor nella tabella dei file aperti del processo Eventualmente elimina elementi dalle tabelle di sistema
read	read(fd, buff, n) legge al più n bytes a partire dalla posizione dell'I/O pointer e li memorizza in buff Restituisce il numero di byte effettivamente letti 0 per end-of-file -1 in caso di errore (perror e errno per sapere quale)
write	write(fd, buff, n) scrive al più n bytes dal buffer buff nel file a partire dalla posizione dell'I/O pointer Restituisce il numero di byte effettivamente scritti o -1 in caso di errore
lseek	lseek(fd, offset, origine) sposta l'I/O pointer di offset posizioni rispetto all'origine. Possibili valori per origine: 0 per inizio del file (SEEK_SET) 1 per posizione corrente (SEEK_CUR) 2 per fine del file (SEEK_END)

Gestione dei processi in UNIX

- Un processo utente in genere viene attivato a partire da un comando (da cui prende il nome). Ad es., dopo aver mandato in esecuzione il comando ‘hw’, verrà visualizzato un processo dal nome ‘hw’.

- ***Tramite ps si può vedere la lista dei processi attivi***

```
pbellavis@lab3-linux:~$ ps
PID TTY STAT TIME COMMAND
4837 p2 S 0:00 -bash
6945 p2 S 0:00 hw
6948 p2 R 0:00 ps
```

Terminazione forzata dei processi

- È possibile ‘terminare forzatamente’ un processo tramite il comando **kill c**he invia un segnale ad un processo

- Ad esempio:
- **kill –9 <PID>** provoca la terminazione del processo
  - Esempio: **kill –9 6945** → termina il processo **‘hw’**

- per conoscere il PID di un determinato processo, si può utilizzare il comando **ps**

Consegna

Chi vuole può inviare via email lo svolgimento dell'estensione ai docenti, in modo da discutere la soluzione ed eventualmente pubblicarla sul sito del corso