Università degli Studi di Bologna
Scuola di Ingegneria

Corso di

Reti di Calcolatori T

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

Luca Foschini

Michele Solimando, Giuseppe Martuscelli

Anno accademico 2019/2020

Scritture e letture coordinate

Si consideri l'architettura di un modello che prevede l'**accesso serializzato** a risorse condivise (come i file). Il modello è costituito da due tipi di processi che agiscono su tali risorse:

- il **produttore** è il processo che produce un file;
- il **consumatore** è il processo che legge il file e lo stampa a video.

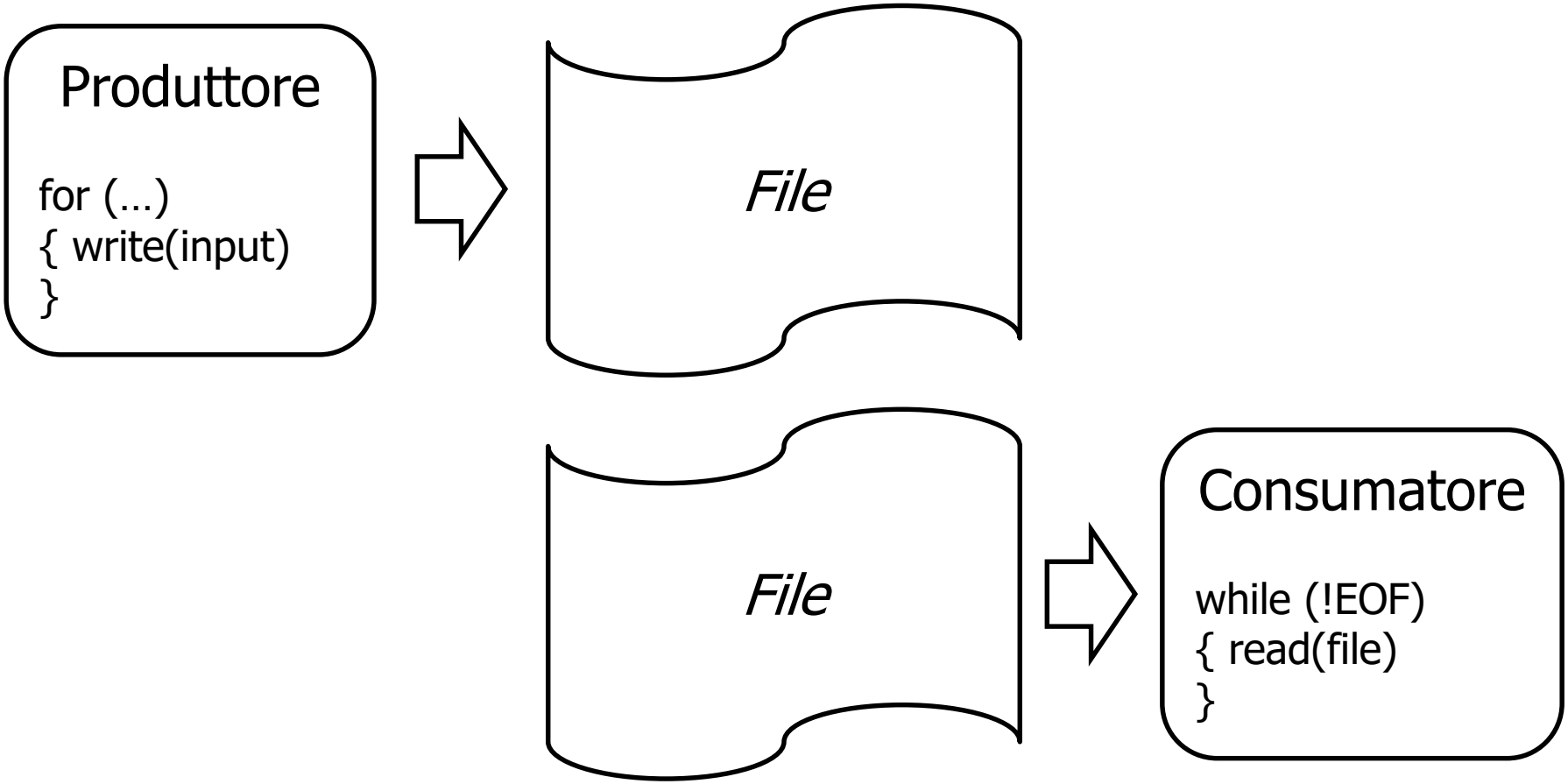
Quando il **produttore** ha terminato di **scrivere** il file**,**

solo allora il **consumatore** può **effettuare la lettura** del file scritto

In questo modo il file diventa un tramite di **comunicazione** tra due processi indipendenti.

Architettura di riferimento

Si consideri lo schema in figura…



Ulteriori dettagli tecnologici

Nella realizzazione proposta qui, le entità sopra definite sono implementate come **due processi** che vengono eseguiti in tempi diversi.

Solo una volta terminato il produttore è possibile che il consumatore esegua.

In questa esercitazione, viene utilizzato come **canale di comunicazione il file (di testo)**, che viene acceduto prima in scrittura e poi in lettura dai due processi in momenti diversi e successivi.

Specifica dell'entità Produttore

Il **Produttore** è il **processo** che **crea il file** e comincia a scrivere il contenuto inserito dall'utente **riga per riga**, rappresentato da stringhe, e le inserisce nel file di testo.

Il processo **chiede preliminarmente all'utente** quante righe ha intenzione di inserire. Poi mette i contenuti nel file.

ATTENZIONE: il Produttore **NON è un filtro**.

Specifica dell'entità Consumatore

Il **Consumatore** è il **processo** che **legge dal file** passato come **argomento** e legge il contenuto dello stesso **fino a EOF**.

Stampa il contenuto a video senza nessun tipo di elaborazione dell'input.

ATTENZIONE: il Consumatore **è un filtro**.

Lettura fino a EOF di file di testo in Java

`String Java.io.BufferedReader.readLine()`

Return Value

A String containing the contents of the line, not including any line-termination characters, **or null if the end of the stream has been reached**.

`int Java.io.FileReader.read()`

Return Value

The character read, **or -1 if the end of the stream has been reached**.

Codice JAVA: Produttore

```
public class Produttore
{public static void main(String[] args)
 { BufferedReader in = null; int res = 0;
   // fare controllo argomenti
   if (args.length != 1){
        System.out.println("Utilizzo: produttore <inputFilename>");
        System.exit(0);
   }
   in = new BufferedReader(new InputStreamReader(System.in));

   FileWriter fout;
   try { fout = new FileWriter(args[0]);
        System.out.println("Quante righe vuoi inserire?");
        res = Integer.parseInt(in.readLine());
        for (int i =0; i<res; i++)
        {  System.out.println("Inserisci la nuova riga");
            String inputl = in.readLine()+"\n";
            fout.write(inputl, 0, inputl.length());
        }
        fout.close();
       }
       catch (NumberFormatException nfe) {e.printStackTrace(); System.exit(1);}
       catch (IOException e) {e.printStackTrace(); System.exit(2);}
 }
}
```

Codice JAVA: Consumatore

```
public class Consumatore
{ public static void main(String[] args)
{
   FileReader r = null;
   // fare controllo argomenti
   if (args.length != 1){
        System.out.println("Utilizzo: produttore <inputFilename>");
        System.exit(0);
   }
    try { r = new FileReader(args[0]); }
    catch(FileNotFoundException e)
        { System.out.println("File non trovato"); System.exit(1);}

    try { int x; char ch;
          while ((x = r.read()) >=0)
           {ch = (char) x;
            System.out.print(ch);
            }
          r.close();
         }
    catch(IOException ex)
        {System.out.println("Errore di input"); System.exit(2);}
 }
}
```

Codice JAVA: esecuzione

Compilare il file sorgente:

```
>javac Produttore.java Consumatore.java
```

Eseguire il Produttore:

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

Eseguire il Cosumatore:

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

Codice C: Produttore

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

#define MAX_STRING_LENGTH 256

int main(int argc, char* argv[]){ // preparazione programma
int fd, readValues, bytes_to_write, written;  char *file_out;
char buf[MAX_STRING_LENGTH];  int righe, i;   char riga[MAX_STRING_LENGTH];
// fare controllo argomenti
    if (argc != 2) { perror(" numero di argomenti sbagliato …"); exit(EXIT_FAILURE);}

   file_out = argv[1];
   printf("Quante righe vuoi inserire?\n");
   readValues = scanf("%d", &righe);
   if( readValues != 1 ) {printf("Errore: immettere un intero!"); exit(1); }
   gets (buf); // consumare il fine linea
   fd = open(file_out, O_WRONLY|O_CREAT|O_TRUNC, 00640);
    if (fd < 0)
    { perror("P0: Impossibile creare/aprire il file");  exit(EXIT_FAILURE);
    }
```

Codice C: Produttore (2)

```
    // corpo produttore
    for(i=0; i<righe; ++i){
     printf("Inserisci la nuova riga\n");
     gets (riga);
     /* la gets legge tutta la riga, separatori inclusi, e trasforma il fine
        linea in fine stringa */
     // aggiungo il fine linea
     riga[strlen(riga)+1]='\0';
     riga[strlen(riga)]='\n';
     written = write(fd, riga, strlen(riga)); // uso della primitiva
     if (written <= 0)
     { perror("P0: errore nella scrittura sul file"); exit(2);
     }
    }
    close(fd);
}
```

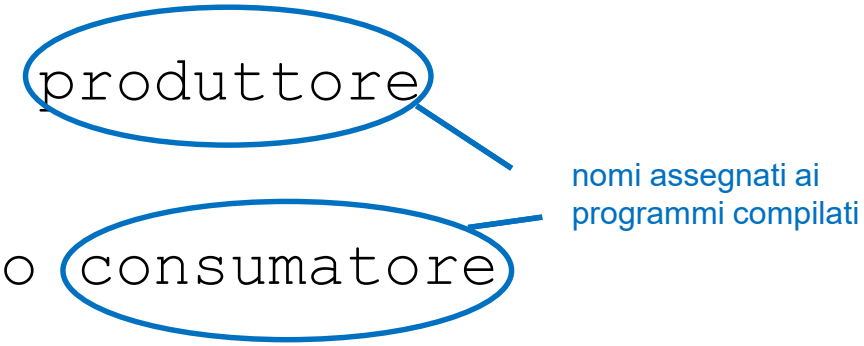
Codice C: Consumatore

```
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#define MAX_STRING_LENGTH 256

int main(int argc, char* argv[]){
    char *file_in, read_char, buf[MAX_STRING_LENGTH]; int nread, fd;
    // fare controllo argomenti
    if (argc != 2) { perror(" numero di argomenti sbagliato"); exit(1);}
    file_in = argv[1];

    fd = open(file_in, O_RDONLY);
    if (fd<0){ perror("P0: Impossibile aprire il file."); exit(2); }

    while(nread = read(fd, &read_char, sizeof(char)))
     /* un carattere alla volta fino ad EOF*/
    {if (nread > 0) putchar(read_char);
     else
     {printf("(PID %d) impossibile leggere dal file %s", getpid(), file_in);
      perror("Errore!"); close(fd); exit(3);
     }
    }
    close(fd);
}
```

Codice C: esecuzione

Compilare i file sorgenti:

```
>gcc produttore.c -o produttore

>gcc consumatore.c -o consumatore
```



Eseguire il file compilato:

```
>produttore fileName.txt

>consumatore fileName.txt
```