Esercitazione 0 - Lettura e Scrittura File in Java e C

Corso di Reti di Calcolatori T

Bernardini Claudio, Corsetti Luca, Giardina Gianluca, Straccali Leonardo

Anno accademico 2021/2022

Scopo dell'esercitazione

Lo scopo di questa esercitazione è sviluppare un'architettura che prevede la creazione di due entità, Produttore e Consumatore:

- Il Produttore è un'entità che produce un file di testo contenente caratteri. La produzione del file viene fatta richiedendo all'utente le linee da scrivere sul file
- Il Consumatore è un filtro che, preso in ingresso una sequenza di caratteri definita dall'utente, elimina questi caratteri da un file dato e stampa a video quelli restanti

Il Produttore

E' un processo che richiede all'utente di inserire riga per riga stringhe che verranno salvate in un file. Il file in questione viene passato come parametro. La lettura dei parametri inseriti dall'utente continua fino a quando l'utente stesso non invia EOF.

Il Consumatore

E' un processo che funge da **filtro a carattere** il cui scopo è eliminare da un file dato i caratteri contenuti in una sequenza, anch'essa data. La sequenza può essere fornita tramite *passaggio per argomento* o *mediante redirezione in input*. La procedura di eliminazione è eseguita in modo sequenziale dall'inizio del file fino alla fine con il riconoscimento del carattere EOF.

Codice Java

Produttore

```
public class Produttore {
    public static void main(String[] args) {
        BufferedReader in = null;
        int res = 0;

        if (args.length != 1) {
            System.out.println("Utilizzo: produttore <inputFilename>");
            System.exit(0);
        }

        in = new BufferedReader(new InputStreamReader(System.in));

        FileWriter fout = null;
```

```
        String inputl = null;
        try {
            fout = new FileWriter(args[0]);
            while ((inputl = in.readLine()) != null) {
                inputl += System.lineSeparator();
                fout.write(inputl, 0, inputl.length());
            }
            fout.close();
        } catch (NumberFormatException nfe) {
            nfe.printStackTrace();
            System.exit(1); // uscita con errore, intero positivo a livello di
sistema Unix
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(2); // uscita con errore, intero positivo a livello di
sistema Unix
        }
    }
}
```

Consumatore

```
public class Consumatore {
    public static void main(String[] args) {
        String filter = "";
        FileReader r = null;
        int x;
        char ch;

        if (args.length > 2) {
            System.out.println("Utilizzo: consumatore <filterprefix>
<inputFilename> oppure");
            System.out.println("Utilizzo: consumatore <filterprefix> '<'
<inputFilename>");
            System.exit(0);
        }

        // rimuoviamo eventuali caratteri duplicati dal parametro filtro passato
dall'utente
        for (int i = 0; i < args[0].length(); i++) {
            char atIndex = args[0].charAt(i);
            if (filter.indexOf(atIndex) == -1) {
                filter += atIndex;
            }
        }

        System.out.println("Filtro specificato: " + filter + "\n");

        try {
            // se args.length == 2 allora il filtro è passato come parametro
insieme al file
```

```
            // altrimenti il file è passato mediante ridirezione input
            if (args.length == 2) {
                r = new FileReader(args[1]);
            } else {
                r = new FileReader(FileDescriptor.in);
            }
        } catch (FileNotFoundException e) {
            System.out.println("File non trovato");
            System.exit(1);
        }

        try {
            while ((x = r.read()) >= 0) {
                ch = (char) x;
                // se il carattere non è tra quelli da filtrare, printa
                if (filter.indexOf(ch) == -1)
                    System.out.print(ch);
            }
            r.close();
        } catch (IOException ex) {
            System.out.println("Errore di input");
            System.exit(2);
        }
    }
}
```

Codice C

Produttore

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

/**
 * Il produttore ha il compito di:
 * Leggere le linee del fil scritte dal cliente
 * Scrivere sul file passato come parametro
 *
 * Sintassi invocazione: ./produttore input_file.txt
*/
int main(int argc, char* argv[]){
    /**
     * Desrizione delle variabili:
     * fd: File descriptor del file passato come parametro
     * written: numero caratteri scritti sul file
     * file_out: puntatore a stringa nome del file da scrivere passato come
parametro
```

```
     * riga: riga letta dall'utente
    */
    int fd, written;
    char *file_out;
    char riga[MAX_STRING_LENGTH];

    /**
     * Controllo dei parametri di invocazione
     * Verifico la presenza del solo parametro: file da scrivere
     *
     * Sintassi invocazione: ./consumatore stringa_caratteri_da_eliminare
input_file.txt
    */
    if (argc != 2){
        perror(" numero di argomenti sbagliato"); exit(1);
    }

    file_out = argv[1]; //Stringa file input

    fd = open(file_out, O_WRONLY|O_CREAT|O_TRUNC, 00640);
    if (fd < 0){
        perror("P0: Impossibile creare/aprire il file");
        exit(2);
    }

    printf("Inserisci le righe del file: [EOF per terminare l'inserimento] \n");
    while (gets(riga) != NULL) {
        /* 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(3);
        }
    }
    close(fd);
}
```

Consumatore

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

/**
```

```
 * Il consumatore è un filtro a caratteri:
 * prende in input il file passato come parametro o il file passato come
redirezione in input
 * Scopo del programma è la stampa del contenuto del file privata dei caratteri
passati come parametro
 *
 * Sintassi invocazione: ./consumatore stringa_caratteri_da_eliminare
input_file.txt
*/
int main(int argc, char *argv[])
{
    /**
     * Descrizione delle variabili:
     * file_in: puntatore alla stringa nome del file da leggere
     * read_char: carattere letto dal file
     * delete_chars: puntatore alla stringa dei caratteri da rimuovere dalla
stampa file
     * nread: numero carattere letto dal file
     * fd: file descriptor del file di input
    */
    char *file_in, read_char, *delete_chars;
    int nread, fd;

    /**
     * Controllo dei parametri in ingresso
     * Il numero dei parametri possono essere compresi tra 2 e 3 (compresi)
     * Sintassi invocazione: ./consumatore stringa_caratteri_da_eliminare
input_file.txt
    */
    if (argc < 2 || argc > 3)
    {
        perror("numero di argomenti sbagliato");
        exit(1);
    }

    delete_chars = argv[1];

    //Conversione dei caratteri della stringa nel rispettivo carattere maiuscolo
    for (int index = 0; delete_chars[index] != '\0'; ++index){
        delete_chars[index] = toupper(delete_chars[index]);
    }

    printf("Stringa caratteri da rimuovere: %s\n", delete_chars);

    if (argc == 3) //Caso 1: file passato come parametro
    {
        file_in = argv[2];
        fd = open(file_in, O_RDONLY);
        if (fd < 0)
        {
            perror("P0: Impossibile aprire il file.");
            exit(2);
        }
    }
```

```
    else //Caso 2: file passato mediante redirezione input
    {
        fd = 0; //Associo file descriptor dello stdin
    }
    while ((nread = read(fd, &read_char, sizeof(char)))) /* Fino ad EOF*/
    {
        if (nread >= 0)
        {
            if ((strchr(delete_chars, toupper(read_char))) == NULL) //Verifico che
carattere letto non appartenga a stringa caratteri da rimuovere dall'output
                putchar(read_char);
        }
        else
        {
            printf("(PID %d) impossibile leggere dal file %s", getpid(), file_in);
            perror("Errore!");
            close(fd);
            exit(3);
        }
    }
    close(fd);
}
```