UNIVERSITÀ DEGLI STUDI FIRENZE

Relazione progetto per il Corso di Sistemi Operativi modulo di Laboratorio – A.A. 2020/2021 – N-version programming

Autori:

Niccolò Rinaldi 7025837 niccolo.rinaldi3@stud.unifi.it

Ilaria Rocchi 7027899 ilaria.rocchi@stud.unifi.it

Chiara Tani 7033107 chiara.tani@stud.unifi.it

Data di consegna: 8 Gennaio 2022

Università degli Studi di Firenze

Corso di laurea in Informatica

INDICE

1. Struttura del progetto

Il progetto è costituito da una cartella denominata Progetto_Rinaldi_Rocchi_Tani, avente al suo interno i seguenti elementi:

- Le cartelle **src** e **data**
- I file **Avvia.c**, **Makefile**, **readme.txt**

Per eseguire il programma è necessario seguire i seguenti passaggi:

1. Compilare il file **Avvia.c**
2. Eseguire il file generato passando i due parametri richiesti, ossia:
   - La modalità di avvio: NORMALE oppure FALLIMENTO
   - Il percorso del file **dataset.csv**

Makefile si occupa di compilare tutti i file presenti nella cartella **src** e di creare le cartelle bin e log.
La cartella log conterrà i file **voted_output.txt** e **system_log.txt** mentre la cartella bin conterrà gli eseguibili di tutti i componenti compilati da **Makefile**.

La cartella **data** contiene una piccola porzione del file dataset.csv, denominata miniDataSet.csv, ed il file dataset.csv stesso.

2. Scopo del progetto

Il fine del progetto è quello di costruire un meccanismo di N-version programming per realizzare una procedura di conteggio.

Per fare ciò viene realizzato un sistema avente come componenti **Input Manager**, **P1**, **P2**, **P3**, **Decision Function**, **Failure Manager** e **Watchdog**. Tali componenti funzionano ed interagiscono tra loro come descritto di seguito.

Input Manager legge il file **dataset.csv** passato all'avvio del programma, ne scarta la prima riga ed invia le righe successive, con periodo 1 secondo, ai processi **P1**, **P2** e **P3**, rispettivamente tramite pipe, socket e scrittura su file condiviso.

P1 utilizza poi la funzione strtok() per estrarre dalla stringa ricevuta le parti di testo separate da virgole; esegue dunque la somma dei valori interi dei caratteri estratti ed il risultato viene sommato a quello dei blocchi di testo precedenti.

P2 effettua invece la somma ignorando, a partire dalla fine della riga in esame, il carattere "virgola".

P3 si avvale dello stesso meccanismo di **P2** per effettuare la somma, con la differenza che preleva i caratteri da sommare a partire dal primo elemento della stringa.

P1, **P2** e **P3** inviano poi il risultato della somma al componente **Decision Function** mediante un socket. **Decision Function** scrive i risultati ricevuti sul file **voted_output.txt**,

effettua su di essi un voto di maggioranza e in base al risultato scrive “SUCCESSO” oppure “FALLIMENTO” sul file **system_log.txt**.

In caso di fallimento viene inoltre inviato a **Failure Manager** un segnale SIGUSR1.

Affinché si verifichi un fallimento è necessario che non esista una maggioranza sui valori delle somme (ossia esso si verifica quando i risultati di tutte le somme risultano diversi tra loro). I componenti **P1**, **P2** e **P3** includono la modalità random_failure, che se abilitata modifica con probabilità 10^{-1} il risultato della somma, aggiungendovi rispettivamente 10, 20 o 30.

Decision Function invia un messaggio I_AM_ALIVE al processo **Watchdog**, cosicché quest’ultimo sappia che l’esecuzione sta continuando.

Il componente **Watchdog** attende il messaggio I_AM _ALIVE con periodo pari a 1 secondo; se il messaggio non viene ricevuto per due periodi, **Watchdog** invia un segnale SIGUSR1 a **Failure Manager**, che procede quindi alla terminazione di tutti i processi.

Il componente **Failure Manager** è inoltre dotato di un handler per il segnale SIGINT (cui ricezione avviene quando l’utente digita il comando Ctrl+C) che se ricevuto comporta anch’esso la terminazione di tutti i processi.

Il programma **Avvia.c** effettua un controllo circa il corretto inserimento dei parametri da parte dell’utente ed esegue i comandi make clean, make install e make, che si occupano di creare le cartelle bin e log, di compilare i file nella cartella src (ossia tutti i componenti) e di spostare gli eseguibili così creati nella cartella bin. Viene infine eseguito **Input Manager**, dando inizio all’esecuzione del meccanismo implementato.

3. Specifiche del sistema

Il sistema su cui il programma è stato testato riporta le seguenti caratteristiche:
SO: Ubuntu 21.10
CPU: Intel Core i7-10510U

4. Elementi facoltativi

Elemento Facoltativo	Realizzato (SI/NO)	Metodo o file principale
Invio di I_AM_ALIVE e realizzazione watchdog	SI	Watchdog
Failure Manager comanda il riavvio di P1, P2, P3	NO	

Il progetto presenta la realizzazione di uno degli elementi facoltativi, ossia l’invio del segnale I_AM_ALIVE e dunque il componente **Watchdog**.

Viene inizializzata la variabile ‘flag’, che gestisce il controllo della ricezione del segnale I_AM_ALIVE. Se quest’ultimo viene infatti ricevuto, la variabile viene resettata. Questo avviene poiché nel main è stato implementato un controllo che permette l’invio di

SIGUSR1 a **Failure Manager** qualora il segnale I_AM_ALIVE non venisse ricevuto per due periodi.

5. Scelte progettuali

Implementazione di Makefile:

È stato scelto di implementare **Makefile** in modo da effettuare un'unica compilazione e per organizzare le varie componenti in modo chiaro e definito. In fase di esecuzione, viene stampato il testo dei comandi del Makefile, poiché è stato deciso di non inserire il carattere '@'.

Utilizzo dell'handler per il segnale SIGINT:

Il segnale SIGINT viene ricevuto a seguito dell'esecuzione del comando Ctrl+C da linea di comando. La gestione del segnale mediante l'implementazione dell'handler permette una corretta terminazione del programma, perché l'handler di default associato al suddetto comando non permette di salvare in modo ottimale tutti i file aperti.

Uso di while(1):

Molte delle componenti utilizzano uno while(1) con l'obiettivo di mantenere sempre i processi attivi e non farli quindi terminare a meno di quanto previsto dalle specifiche del progetto.

6. Esecuzione

Esecuzione in modalità NORMALE:

```
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$ cc Avvia.c -o avvia
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$ ./avvia NORMALE /home/
chiara/Progetto_Rinaldi_Rocchi_Tani/data/miniDataSet.csv
rm -r log
rm -r bin
mkdir ./log
mkdir ./bin
cc ./src/InputManager.c -o bin/InputManager
cc ./src/P1.c -o bin/P1
cc ./src/P2.c -o bin/P2
cc ./src/P3.c -o bin/P3
cc ./src/FailureManager.c -o bin/FailureManager
cc ./src/DecisionFunction.c -o bin/DecisionFunction
cc ./src/Watchdog.c -o bin/Watchdog
Compilazione terminata!

---------------------------
MODALITÀ: NORMALE
Esecuzione...
Errore apetura pipe
P3 PRONTO

Problema di connessione P3 -> DecisionFunction, riprovare tra 1 secondo
P2 PRONTO
P1 PRONTO

SIGUSR1 ricevuto, uccisione dei processi in corso...
Ucciso
```

```
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$ cd log
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani/log$ cat system_log.txt

SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani/log$ cat voted_output.t
xt
30622,30622,30622
30624,30624,30624
30623,30623,30623
30613,30613,30613
30614,30614,30614
30617,30617,30617
30619,30619,30619
30604,30604,30604
30626,30626,30626
30620,30620,30620
30622,30622,30622
30615,30615,30615
30610,30610,30610
30612,30612,30612
30619,30619,30619
```

Esecuzione in modalità FALLIMENTO:

```
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$ cc Avvia.c -o avvia
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$ ./avvia FALLIMENTO /ho
me/chiara/Progetto_Rinaldi_Rocchi_Tani/data/miniDataSet.csv
rm -r log
rm -r bin
mkdir ./log
mkdir ./bin
cc ./src/InputManager.c -o bin/InputManager
cc ./src/P1.c -o bin/P1
cc ./src/P2.c -o bin/P2
cc ./src/P3.c -o bin/P3
cc ./src/FailureManager.c -o bin/FailureManager
cc ./src/DecisionFunction.c -o bin/DecisionFunction
cc ./src/Watchdog.c -o bin/Watchdog
Compilazione terminata!

---------------------------
MODALITÀ: FALLIMENTO
Esecuzione...
Errore apetura pipe
P3 PRONTO

Problema di connessione P3 -> DecisionFunction, riprovare tra 1 secondo
P2 PRONTO
P1 PRONTO

SIGUSR1 ricevuto, uccisione dei processi in corso...
Ucciso
```

```
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$ cd log
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani/log$ cat system_log.txt

SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani/log$ cat voted_output.t
xt
30622,30622,30622
30624,30634,30624
30623,30623,30623
30613,30613,30613
30614,30614,30614
30617,30617,30617
30619,30619,30619
30604,30614,30604
30626,30626,30626
30620,30620,30620
30622,30622,30622
30615,30615,30615
30610,30610,30610
30612,30612,30612
30619,30619,30619
```

Esecuzione in modalità FALLIMENTO (con attivazione di random_failure):

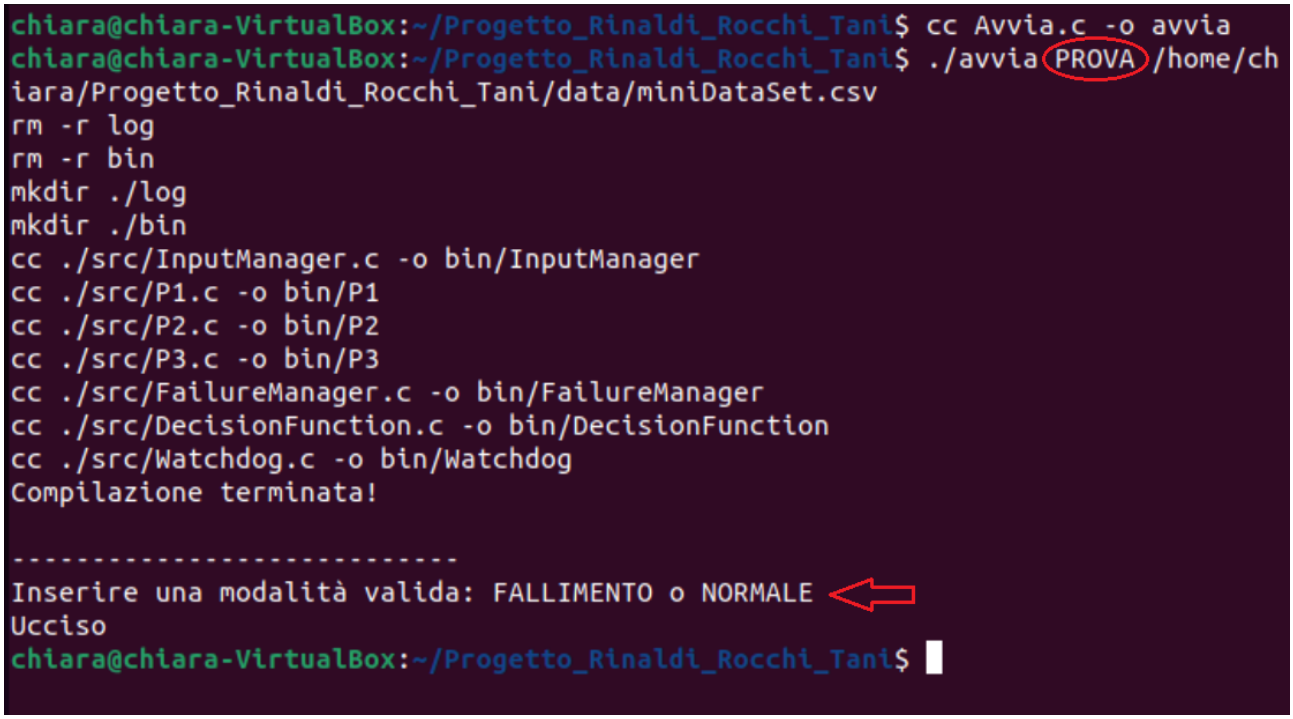
```
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$ cd log
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani/log$ cat system_log.txt

SUCCESSO
SUCCESSO
SUCCESSO
SUCCESSO
FALLIMENTO
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani/log$ cat voted_output.t
xt
30622,30622,30622
30624,30624,30624
30623,30623,30653
30613,30613,30613
30614,30624,30644
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani/log$
```

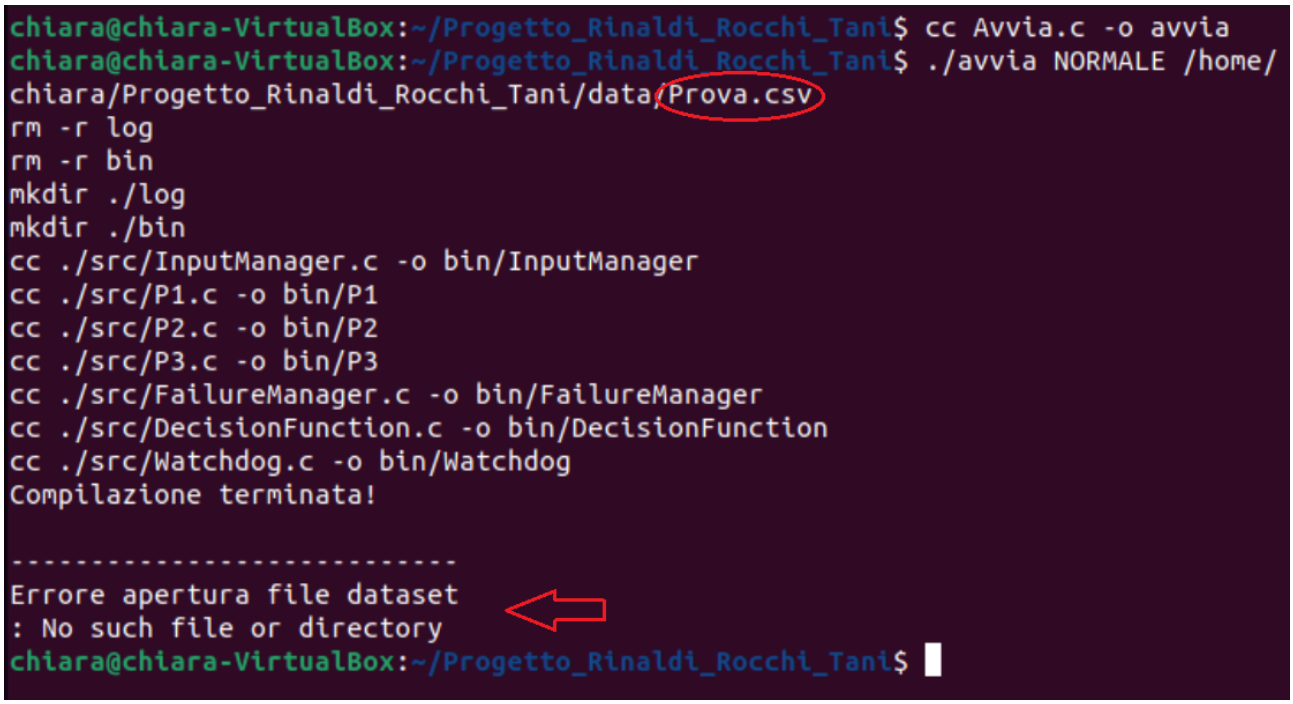
Esecuzione con un parametro mancante:

```
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$ cc Avvia.c -o avvia
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$ ./avvia /home/chiara/P
rogetto_Rinaldi_Rocchi_Tani/data/miniDataSet.csv
Inserire tutti gli argomenti necessari
Ucciso
chiara@chiara-VirtualBox:~/Progetto_Rinaldi_Rocchi_Tani$
```

Esecuzione con una modalità errata:



Esecuzione con path errato:



Le prove sono state effettuate utilizzando una piccola porzione del file dataset.csv.