

Relazione Progetto A.A. 2020/2021- Meccanismi di software fault tolerance: un semplice esempio di N-Version programming

Informazioni autore del progetto

Autore: Jacopo Manetti

Matricola: 7003888

data di consegna: 10/01/2022

mail: jacopo.manetti1@stud.unifi.it

introduzione

L'obiettivo di questo progetto è quello di costruire un meccanismo di N-version programming in cui 3 programmi C ricevono righe di dati CSV e per ciascuna di esse si produce una somma. (vengono effettivamente prodotte 3 somme che denotano 3 voti). Ora il piano è che questi 3 programmi "votino" la somma corretta. In questo caso la somma corretta deve avere il sostegno di almeno due "elettori".

L'idea è di rimuovere gli errori che si verificano dal processo di sviluppo all'esecuzione e avere un sistema più robusto.

Il sistema fault tolerant è composto da:

- 3 elettori - i programmi P1, P2 e P3

- Una decision function - accetta i voti, effettua le elezioni

- input manager - legge i file CSV e consegna le righe agli elettori

- Faiure Manager - arresta il sistema quando viene segnalato

- Watchdog timer - garantisce che i voti arrivino in tempo, mantiene il sistema sotto controllo

Come output il programma produce due file "voted_output.txt", che contiene i valori ricevuti per riga e "System_log.txt", che contiene informazioni sullo svolgimento dei turni elettorali, successo o no.

compilazione ed esecuzione

Ho fornito un makefile con istruzioni di compilazione codificate.

Tutto ciò che serve è un sistema che abbia gcc e make utility

Per compilare... all'interno della cartella eseguire make

```
$ make
```

Successivamente verranno creati 7 programmi: P1, P2, P3, input_manager, failure_manager e watchdog

Per avviare il programma:

```
$ ./input_manager FALLIMENTO dataset.csv
```

Dove "dataset.csv" è il file da elaborare riga per riga

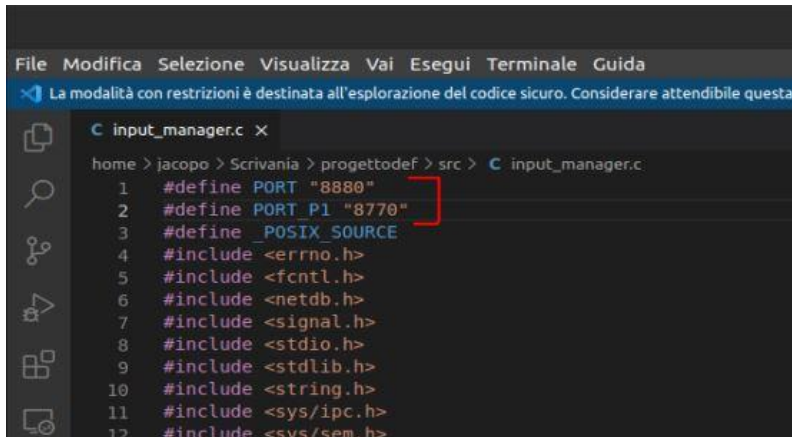
Il programma stamperà le informazioni di debug sullo schermo man mano che i dati vengono prodotti e consumati come aggiornamento dei file "voted_output.txt" e "System_log.txt" riga per riga

Nota bene

Se si vuole eseguire il programma più volte prima è necessario cambiare le porte, per farlo basta cambiare le prime due righe del file: ``input_manager.c`` => definite:

```
```#define PORT "8880"
```

```
#define PORT_P1 "8770"```
```



```
File Modifica Selezione Visualizza Vai Esegui Terminale Guida
x La modalit  con restrizioni   destinata all'esplorazione del codice sicuro. Considerare attendibile questa
C input_manager.c x
home > jaco... > Scrivania > progettodef > src > C input_manager.c
1 #define PORT "8880"
2 #define PORT_P1 "8770"
3 #define POSIX_SOURCE
4 #include <errno.h>
5 #include <fcntl.h>
6 #include <netdb.h>
7 #include <signal.h>
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <string.h>
11 #include <sys/ipc.h>
12 #include <sys/sem.h>
```

regole:

basta cambiare i numeri tra le virgolette della stringa, devono essere numeri => non lettere o altri caratteri, preferire numeri a pi  di 4 cifre.

quindi eseguire

```
```$ make```
```

idealmente le porte dovrebbero essere fornite come parametri della riga di comando al programma ma l'assegnazione era molto specifica sui parametri che il programma input_manager deve prendere.

In alternativa basta riavviare il sistema.

Sistema di destinazione

Il sistema di destinazione   un computer che esegue il sistema debian/ubuntu linux perch  ci affidiamo alle api posix come fork, sockets e pipes. (in particolare   stato utilizzato Ubuntu 20.04.2)

Progettazione e realizzazione

L'input manager   il processo di root.   responsabile dell'avvio di tutti gli altri processi nel sistema. Questo   utile in quanto ottiene tutti i loro pid ed   in grado di passare loro i parametri. I pid sono usati per terminare tutti i processi quando il sistema ha finito. L'input manager inoltre legge anche il file CSV e lo invia riga per riga agli elettori P1 P2 P3. L'input manager crea e gestisce anche 2 semafori uno da sincronizzare con la decision function (quando la riga precedente   stata elaborata) e l'altro per sincronizzarsi con l'elettore P3 (quando il file   stato scritto)

La decision function   l'arbitro elettorale. Funziona come un server. P1, P2 e P3 sono i suoi client che stabiliscono una connessione e la mantengono attiva per l'invio di messaggi quando disponibile. La connessione socket al decision server, una volta effettuata, non viene pi  chiusa. Sono consentiti solo 3 collegamenti; ogni connessione copia il "voto" in una posizione specifica dell'array (specifica per ogni socket). Quando l'array si riempie (tutti e 3 i valori presenti) i valori vengono scritti nel file "voted_output.txt" e viene condotta un'elezione. Se almeno due dei valori sono uguali scrive SUCCESSO nel file "system_log.txt" e segnala il processo watchdog, altrimenti scrive FALLIMENTO nel "system_log.txt" e lo

segnala al failure manager. Infine il semaforo viene incrementato per consentire all'input manager di rilasciare la riga successiva agli elettori.

Il failure manager è sempre in attesa di un segnale. Questo viene fatto da un ciclo che chiama la funzione "pause()" che attende di essere segnalato, se accade il signal handler a sua volta segnalerà il processo di root (input manager) per terminare il programma.

Anche il watchdog è sempre in attesa di un segnale (I_AM_ALIVE) dal decision manager per resettare il timer dell'allarme. Se l'allarme non viene resettato prima che siano trascorsi due secondi, questo si attiverà e invierà un segnale al failure manager per arrestare il sistema.

Elemento facoltativo

Come si può constatare dalla descrizione fatta qua sopra, dei 2 elementi facoltativi è stato implementato solo il watchdog, il cui funzionamento è gestito utilizzando la funzione alarm().

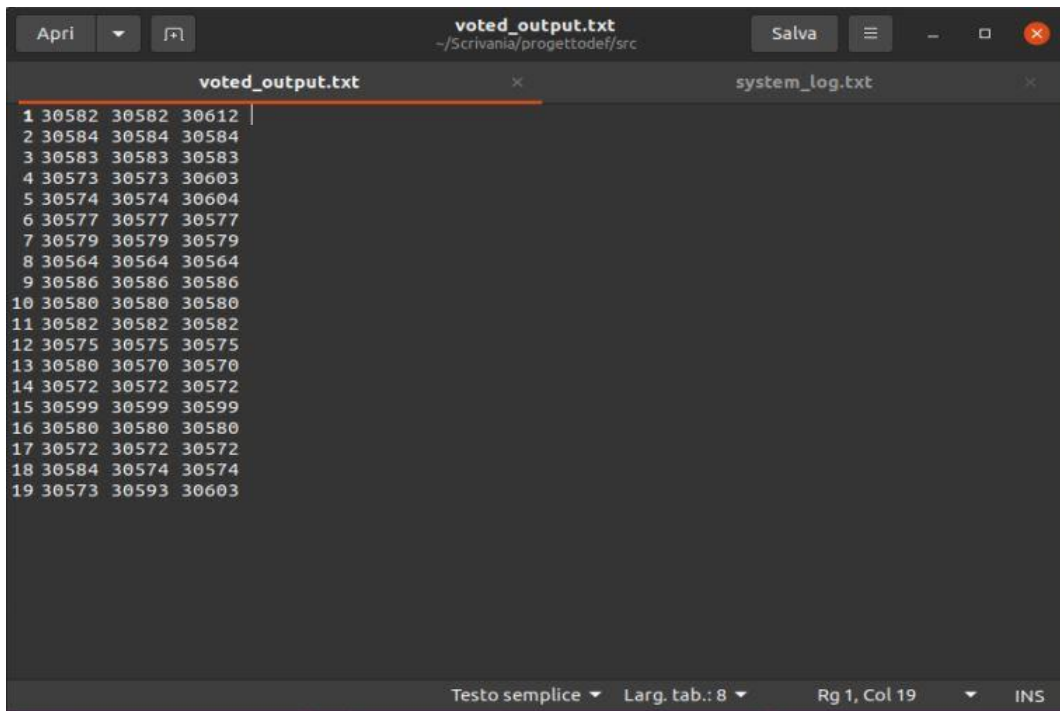
architettura generale: temporizzazione del sistema

Questo è nel complesso un sistema basato su eventi in cui i processi generano e consumano eventi per ottenere i calcoli desiderati, procede in questo modo:

- L'input manager genera l'evento "nuovi dati" e li invia agli elettori in attesa. Attende anche su un semaforo per sapere se può leggere e rilasciare la riga successiva.
- Gli elettori sono fondamentalmente bloccati / in attesa di ricevere questo evento
 - p1 -> fermo su "leggi pipe"
 - p2 -> crea la connessione socket e attende in lettura
 - p3 -> è in loop finché il file condiviso non ha contenuto; è segnalato da un semaforo
- Dopo che gli elettori hanno le somme, trasmettono i loro valori al decisore e riprendono lo stato di attesa
- Il decision manager deve attendere tutti e 3 i voti prima di decidere il risultato delle elezioni e a sua volta invia i segnali al watchdog e al failure manager a seconda dell'elezione risultante. Dopodiché aumenta il valore del semaforo per riattivare l'input manager e torna in attesa dei 3 voti successivi.
- Il watchdog attende di essere segnalato per resettare il suo allarme, se non accade lo segnala al failure manager. Si trova in un ciclo perpetuo.
- A sua volta il failure manager segnalerà il processo di root/input manager dopo che è stato segnalato. Anche l'input manager è impostato per attendere questo segnale di SHUTDOWN.

Esempio di risultati di esecuzione

Di seguito sono riportati i file prodotti da un'esecuzione di esempio:

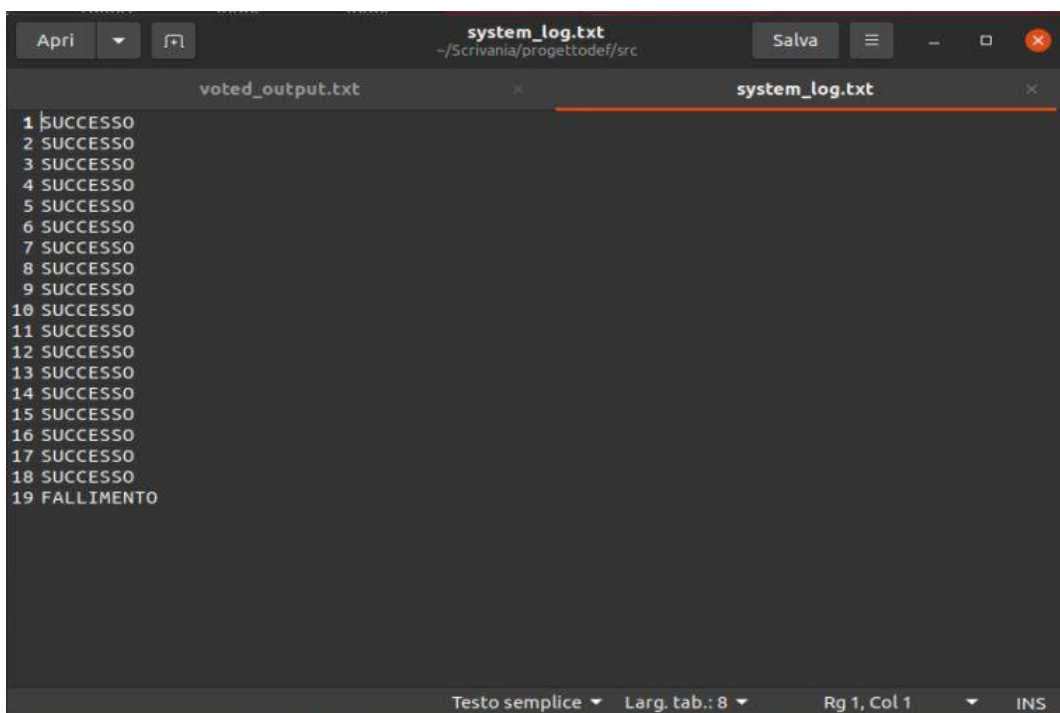


The screenshot shows a text editor window with the title 'voted_output.txt' and the path '~/Scrivania/progettodef/src'. The editor contains 19 lines of data, each with three integers separated by spaces. The values for each line are: 1: 30582 30582 30612; 2: 30584 30584 30584; 3: 30583 30583 30583; 4: 30573 30573 30603; 5: 30574 30574 30604; 6: 30577 30577 30577; 7: 30579 30579 30579; 8: 30564 30564 30564; 9: 30586 30586 30586; 10: 30580 30580 30580; 11: 30582 30582 30582; 12: 30575 30575 30575; 13: 30580 30570 30570; 14: 30572 30572 30572; 15: 30599 30599 30599; 16: 30580 30580 30580; 17: 30572 30572 30572; 18: 30584 30574 30574; 19: 30573 30593 30603. The status bar at the bottom indicates 'Testo semplice', 'Larg. tab.: 8', 'Rg 1, Col 19', and 'INS'.

```
1 30582 30582 30612
2 30584 30584 30584
3 30583 30583 30583
4 30573 30573 30603
5 30574 30574 30604
6 30577 30577 30577
7 30579 30579 30579
8 30564 30564 30564
9 30586 30586 30586
10 30580 30580 30580
11 30582 30582 30582
12 30575 30575 30575
13 30580 30570 30570
14 30572 30572 30572
15 30599 30599 30599
16 30580 30580 30580
17 30572 30572 30572
18 30584 30574 30574
19 30573 30593 30603
```

Si noti che nelle righe di “voted_output.txt” quando tutti gli elettori avanzano lo stesso valore e quando un elettore non è d'accordo con il valore proposto dagli altri due (es. righe: 1,4,5,13,18), viene riportato SUCCESSO nelle righe corrispondenti del file “system_log.txt”.

Ora sull'ultima riga: riga 19 tutti e tre gli elettori hanno valori diversi per la somma. Questo è inaccettabile e quindi la riga 19 del file system_log.txt legge un FALLIMENTO e il sistema si arresta subito



The screenshot shows a text editor window with the title 'system_log.txt' and the path '~/Scrivania/progettodef/src'. The editor contains 19 lines of text, each with the word 'SUCCESSO' or 'FALLIMENTO'. The status bar at the bottom indicates 'Testo semplice', 'Larg. tab.: 8', 'Rg 1, Col 1', and 'INS'.

```
1 SUCCESSO
2 SUCCESSO
3 SUCCESSO
4 SUCCESSO
5 SUCCESSO
6 SUCCESSO
7 SUCCESSO
8 SUCCESSO
9 SUCCESSO
10 SUCCESSO
11 SUCCESSO
12 SUCCESSO
13 SUCCESSO
14 SUCCESSO
15 SUCCESSO
16 SUCCESSO
17 SUCCESSO
18 SUCCESSO
19 FALLIMENTO
```

