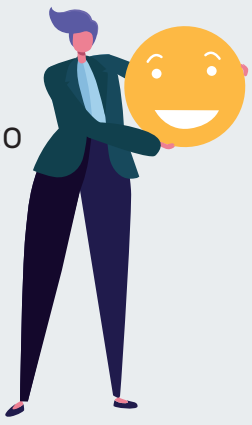
PROGETTO FLY BY WIRE

Autore:

Bardeli Edmond
Matricola 6001066
edmond.bardeli@stud.unifi.it
Data 04/02/2021

Indice:



Progetto di Sistemi Operativi assegnato per l'anno accademico 2019-2020 dal Prof. **Ceccarelli Andrea**.

AMBIENTE DI SVILUPPO E SCELTE PROGETTUALI

Distribuzione di Linux utilizzata:
Ubuntu 20.10 Groovy Gorilla.

L'ambiente di sviluppo utilizzato:
Eclipse IDE for C/C++.

Per monitorare l'esecuzione dell'applicazione posso aprire il software "**Monitor di sistema**". All'interno possiamo trovare tutti i processi generati, in più posso interagire inviando segnali (con la finestrina tasto destro del mouse), come mostra l'immagine:

Processi | Risorse | File system

Nome del processo	Stato	Memoria vir	Memoria res	Memoria	Totale lettura disco
bash	Sleeping	10,6 MiB	4,0 MiB	1,7 MiB	393,1 KiB
MainProject	Sleeping	3,8 MiB	1,4 MiB	112,0 KiB	38,6 MiB
PFC1	Sleeping	3,8 MiB	104,0 KiB	104,0 KiB	1,4 MiB
PFC2	Sleeping	3,8 MiB	104,0 KiB	104,0 KiB	1,4 MiB
PFCdisconSwitch	In esecuzione	3,8 MiB	1,2 MiB	112,0 KiB	53,8 MiB
GenFailures	Fermato	3,8 MiB	1,1 MiB	116,0 KiB	2,2 MiB
NextLine	Sleeping	3,8 MiB	860,0 KiB	104,0 KiB	1,7 MiB
Wes	Fermato	3,8 MiB	800,0 KiB	116,0 KiB	1,4 MiB
TransducerPFC1	In esecuzione	3,8 MiB	104,0 KiB	104,0 KiB	*N/D*
TransducerPFC2	Sleeping	3,8 MiB	104,0 KiB	104,0 KiB	*N/D*
TransducerPFC3	In esecuzione	3,8 MiB	104,0 KiB	104,0 KiB	2,3 KiB
PFC3	Sleeping	3,8 MiB	104,0 KiB	104,0 KiB	1,4 MiB

La scelta progettuale per cui ho optato è quella di creare moduli contente uno (o più similari) processi, che usufruissero dei metodi interni. Per ridurre il legame fra gli script e facilitare la generazione dei processi.

ISTRUZIONI DI COMPILAZIONE ED ESECUZIONE

Per iniziare la compilazione, aprire una shell nella directory della cartella del progetto.

Per la generazione dei file oggetto eseguire il comando:

```
~$ make all
```

L'esecuzione puo' iniziare con il seguente comando:

```
~$ ./mainProject  /home/scrivania/G18.txt
```

Se la directory del file G18.txt non è corretta, il programma chiede ricorsivamente la directory fino a quando non risulta esatta.

Dopo l'inserzione della directory corretta, inizia l'esecuzione e non si conclude finché tutte le PFC sono terminate oppure ci troviamo in una situazione in cui non ci sono più righe $GPGLL da leggere nel file G18.txt.

Per la pulizia dei file oggetto eseguire il comando:

```
~$ make clean
```

TABELLA DEGLI ELEMENTI FACOLTATIVI

Elementi Facoltativi	Reallizzato (SI/NO)	Metodo o File Principale
Utilizzo Makefile per la compilazione	SI	ProgettoEB/makefile
Organizzazione in folder, e creazionedei folder al memento della compilazione	SI	presetFile(); killAllProcess();
Riavvio di PFC1, PFC2 e PFC3 alla ricezione di Emeregnza	SI	ProgettoEB/src /disconnectedSwitch.c (metodo: disconnectedSwitch())
Utilizzo macro nel Generare Fallimenti per indicare le probabilità	SI	ProgettoEB/src /disconnectedSwitch.c (metodo: processGenFailures())
PFC Disconnect Switch: - sblocca il processo se bloccato. - riavvia il processo se interrotto In entrambi i casi, il Processo inquestione deve riprendere a leggere dal punto giusto di G18.txt	SI	ProgettoEB/src /disconnectedSwitch.c (metodo: disconnectedSwitch())

SCHEMA DI RAPPRESENTAZIONE DELLA SOLUZIONE ADOTTATA

mainProject.c

nextLine ();

Il metodo genera il processo **nextLine** ogni secondo incrementare il contatore della linea GPGLL, inoltre manda un segnale al Wes e GenFailures di tipo SIGCONT.

processPFC1(); processPFC2(); processPFC3();

Il metodo genera i processo **PFC1**, **PFC2** e **PFC3**.
Ogni secondo i processi preleva il contatore di riga, preleva la riga da G18. Infine calcola la distanza, la velocita e comunica al Transducer il risultato.

processoWes();

Il metodo genera il processo **Wes** dopo il segnale proveniente dal processo nextLine, effettua il confronto delle velcoità nei file speedPFC.log, e stampa il tipo di messaggio in base alle condizioni.

processTransducer();

Il metodo genera il processo **TransducerPFC1**, **TransducerPFC2** e **TransducerPFC3**. Ogni processo si occupa di ricevere la stringa proveniente dalla propria PFC, scrivere nei file .log e stampa a video il contenuto.

disconnectSwitch();

Il metodo genera il processo **PFCdisconSwitch**.
Il processo preleva il pid delle PFC, verifica lo stato.
Se una delle PFC si trova in uno stato di Z(zombie) o T(stop), la PFC viene rigenerata o mandato un SIGCONT.

genFailures();

Il metodo genera il processo **GenFailures**.
Ogni secondo il processo scegliere in modo causare la PFC a cui inviare il segnale(SIGINT, SIGSTOP, SOGCONT o SIGUSR1).

Invoca il metodo che genera il processo.

Invia segnali

Lo Script principale da cui ha inizio l'applicazione è mainProject.c. Questo invoca tutti i metodi descritti sopra; le PFC hanno un timing indipendente l'una dall'altra. Il processo Nextline detta il timing di esecuzione del processo Wes e GenFailures. Gli altri processi sono invece indipendenti, aspettano comunicazioni da altri processi per l'attivazione.

SIMULAZIONE DELL'ESECUZIONE CON COMMENTI

```
Gen Signal (Random) => SIGSTOP
Transucer(PFC2)=> Riga,214,Velocità,0.000000[m/s]
Transucer(PFC1)=> Riga,214,Velocità,0.000000[m/s]
Segnale(Wes)=> [ ERRORE PFC1 ]


Transucer(PFC3)=> Riga,215,Velocità,0.000000[m/s]
DisconnectSwitch =>    Ricontinua PFC3
Transucer(PFC2)=> Riga,215,Velocità,0.000000[m/s]
Transucer(PFC1)=> Riga,215,Velocità,0.000000[m/s]
Segnale(Wes)=>[ OK ]
[...]

Gen Signal (Random) => SIGUSR1
Transucer(PFC2)=> Riga,719,Velocità,1.395258[m/s]
Transucer(PFC3)=> Riga,719,Velocità,1.395258[m/s]
Transucer(PFC1)=> Riga,719,Velocità,1.395258[m/s]
Segnale(Wes)=>[ OK ]


Gen Signal (Random) => SIGCONT
Transucer(PFC3)=> Riga,720,Velocità,1.178116[m/s]
Transucer(PFC1)=> Riga,720,Velocità,1.178116[m/s]
Transucer(PFC2)=> Riga,720,Velocità,4.000000[m/s]
Segnale(Wes)=> [ ERRORE PFC2 ]
[...]

Gen Signal (Random) => SIGINT
Transucer(PFC3)=> Riga,919,Velocità,0.189871[m/s]
Transucer(PFC2)=> Riga,919,Velocità,0.189871[m/s]
Segnale(Wes)=> [ ERRORE PFC1 ]


DisconnectSwitch =>    Riavvia PFC1
Transucer(PFC1)=> Riga,920,Velocità,0.000000[m/s]
Transucer(PFC3)=> Riga,920,Velocità,0.342294[m/s]
Transucer(PFC2)=> Riga,920,Velocità,0.342294[m/s]
Segnale(Wes)=> [ ERRORE PFC1 ]
[...]

Gen Signal (Random) => SIGSTOP
Transucer(PFC1)=> Riga,1208,Velocità,1.119271[m/s]
Transucer(PFC3)=> Riga,1208,Velocità,4.000000[m/s]
Segnale(Wes)=> [ EMERGENZA ]

EMERGENZA catturata, l'applicazione riparte fra 5 secondi...

Transucer(PFC2)=> Riga,1209,Velocità,0.000000[m/s]
Transucer(PFC1)=> Riga,1209,Velocità,0.000000[m/s]
Transucer(PFC3)=> Riga,1209,Velocità,0.000000[m/s]
Segnale(Wes)=>[ OK ]
```

Esempio di Segnale SIGSTOP
Come possiamo vedere a sinistra, il processo PFC2 riceve un segnale di SIGSTOP dal processo GenSignal. Subito dopo riprende a leggere da punto giusto di G18.txt.

Esempio di Segnale SIGUSR1
Come possiamo vedere a sinistra il processo PFC2 riceve un segnale di SIGUSR1 dal processo GenSignal, effettua il cast a int della velocità e lo shoft di due bit. Infine riprende a leggere da punto giusto di G18.txt.

Esempio di Segnale SIGINT
Come possiamo vedere a sinistra il processo PFC1 riceve un segnale di SIGINT dal processo GenSignal. Il processo PFCdisconSwitch si accorge che la PFC1 non è presente e la genera.

Esempio di Emeregnza
Come possiamo vedere a sinistra il tutte le PFC hanno comunicato valori diversi. PFCdisconSwitch riavvia le PFC e continua a leggere dal punto in cui ha ricevuto il segnale EMERGENZA.