PROGETTO SO 2020/21

Autori:

Filippo Zaccari 7030220 [filippo.zaccari@stud.unifi.it](mailto:filippo.zaccari@stud.unifi.it)

Elia Matteini 7033383 [elia.matteini@stud.unifi.it](mailto:elia.matteini@stud.unifi.it)

Ilaria Catone 7020875 [ilaria.catone@stud.unifi.it](mailto:ilaria.catone@stud.unifi.it)

Università degli Studi di Firenze

Corso di laurea in Informatica

Indice

[1 Struttura progetto 3](#_Toc80428773)

## 1 Struttura progetto

Il progetto si presenta con una cartella progettoSO che al suo interno contiene:

* Una cartella SRC
* Due file Avvia.c e makefile

Oltre a questi è presente anche un file readme contenente le istruzioni necessarie per l’esecuzione del programma.

Per eseguire il programma è necessario eseguire questi passi:

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

Il programma a quel punto attraverso il makefile compilerà tutti i file sorgente che sono in SRC, creerà altre quattro cartelle: BIN, LIB, LOG e OBJ, dopodiché sposterà i vari file generati dalla compilazione nelle varie cartelle appena generate, di cui i file .o nella cartella OBJ, il file ConstHeader.h in LIB e tutti gli eseguibili in BIN.

La cartella log conterrà tutti i file richiesti, quindi system\_log e voted\_output, e quelli necessari per l’esecuzione, tra cui il file condiviso fileP3 usato per scambiare messaggi tra InputManager e P3, un filePid dove vengono salvati i PID dei vari processi, un file pipeP1 per la pipe tra InputManager e P1, un file socketP2 per il socket tra InputManager e P2 ed infine socketDF che è il socket usato per scambiare messaggi tra i processi P1, P2 e P3 ed il DecisionFunction.

Per riportare il progetto nelle condizioni iniziali è sufficiente eseguire make clean oppure rieseguire i passi per avviare il programma, che al suo interno eseguirà il comando prima dell’esecuzione.

# 2 Scopo del progetto

Lo scopo del progetto è costruire un meccanismo di N-version programming per realizzare una procedura di conteggio su file di log.

Il lavoro principale è svolto dai processi P1, P2 e P3 che svolgono lo stesso compito ma in maniera diversa.  
L’inputManager legge il file dataset passato all’avvio del programma, scarta la prima riga ed invia ogni riga letta ai processi P1, P2 e P3 in tre modi differenti, rispettivamente con una pipe, un socket e una file condiviso.

Il processo P1 una volta ricevuta la stringa utilizza la funzione strtok() per dividerla in blocchi separati dal carattere “virgola”, poi esegue la somma dei valori interi di ogni carattere del blocco e il risultato lo somma al risultato degli altri blocchi.

P2 invece esegue la somma della stringa partendo dalla fine verso l’inizio scartando il carattere “virgola” e cioè scartando il valore intero 44 che è il corrispondente ASCII.

Infine, P3 è molto simile a P2 con la differenza che esegue la somma dall’inizio verso la fine.

Il controllo dei risultati viene svolto dal programma DecisionFunction che attraverso un socket riceve i risultati delle varie somme, scrive i tre risultati nel file voted\_output, effettua un voto di maggioranza e in base al risultato del voto scrive SUCCESSO o FALLIMENTO nel file system\_log.

Nel caso di un successo il programma continuerà la sua routine altrimenti, se si verifica un fallimento, verrà mandato un segnale SIGUSR1 al processo FailureManager.

Si verifica un fallimento quando non si ha una maggioranza sui valori delle tre somme, questo può avvenire solo quando la modalità random failure viene attivata. Questa fa sì che con probabilità 10% il risultato delle somme venga influenzato con una maggiorazione di 10, 20 e 30 (rispettivamente in P1, P2 e P3).

Inoltre, prima di determinare SUCCESSO o FALLIMENTO invia un segnale al Watchdog per avvisarlo che l’esecuzione sta procedendo.

Il Watchdog attende un segnale I\_AM\_ALIVE con periodo pari a due secondi, che se non viene ricevuto in tempo invia un segnale SIGUSR1 al FailureManager che procede a terminare tutti i processi.

FailureManager è dotato di un handler per gestire l’arrivo di un segnale SIGUSR1 da parte del DecisionFunction che fa terminare tutti i processi.

Inoltre ha un altro handler per il segnale SIGUSR2 che se ricevuto dal DecisionFunction vuol dire che il processo inputManager ha scandito tutto il dataset e quindi può procedere alla terminazione di FailureManager e Watchdog.

Il sorgente Avvia.c

Parlare del file pid e di avvia

Robe sistema operativo

Tabella e come sono stati fatti

esecuzione