SISTEMI OPERATIVI con LABORATORIO

Progetto della parte di Laboratorio – a.a. 2024-25

"*Incrocio*"

(le indicazioni per la consegna sono alla fine del documento)

Realizzare in linguaggio *C* ed in ambiente *Unix/Linux* un'applicazione denominata "***Incrocio"*** e composta dai seguenti processi:

- processo ***incrocio***
- processo ***garage***
- più processi ***automobile***

Tutti i processi sopra menzionati sono a sé stanti (*task*), cioè NON sono *thread/sottoprocessi/processi leggeri* di un altro di essi.

L'applicazione simula il passaggio di automobili attraverso un quadrivio, cioè un incrocio tra 4 strade, e si avvia eseguendo i processi ***incrocio*** e ***garage***.
Il processo ***garage*** crea 4 processi ***automobile*** una volta al secondo e rimane ogni volta in attesa che tutte e quattro le automobili abbiano atraversato l'incrocio prima di crearne altri.
Il processo ***incrocio*** attende la notifica da parte del processo ***garage*** che ci sono automobili all'incrocio da far passare; dopodiché sceglie, secondo le regole del codice della strada, quale automobile può attraversare e comunica ad essa che può farlo. Ripete questa scelta finché tutte e 4 le automobili hanno atraversato e rimane di nuovo in attesa della notifica dal processo ***garage*** che ci sono altre macchine da far passare.
Strade ed automobili sono identificate dai numeri da 0 a 3: la prima automobile proviene dalla prima strada ed entrambe sono identificate dal numero 0, la seconda automobile proviene dalla seconda strada ed entrambe sono identificate dal numero 1, etc.
I processi ***automobile*** DEVONO essere creati con *fork(2)* dal processo ***garage*** come descritto più avanti.

Scopo dell'applicazione è che lo studente dimostri la capacità di saper utilizzare gli strumenti visti in aula per la creazione e la gestione dei processi e quelli per la comunicazione/sincronizzazione tra di loro.

<u>Di seguito la descrizione più in dettaglio delle operazioni che ogni processo deve compiere.</u>

Il processo ***garage*** esegue in sequenza e di continuo le seguenti operazioni:

1. crea con *fork()* 4 processi ***automobile***, ognuna proveniente da una strada diversa delle 4 del quadrivio. Prima di creare ogni processo ***automobile*** estrae a sorte la strada in cui l'automobile dovrà svoltare attraversando l'incrocio, che deve essere diversa da quella di provenienza. Ogni processo ***automobile*** deve conoscere la strada in cui dovrà svoltare
2. per ogni processo ***automobile*** creato emette un messaggio a video che indica il numero di automobile creata e la strada in cui deve svoltare
3. notifica al processo ***incrocio*** che ci sono 4 auto al quadrivio da far passare
4. rimane in attesa che le automobili abbiano tutte attraversato l'incrocio (ogni processo ***automobile*****,** dopo aver attraversato l'incrocio, termina)
5. attende 1 secondo, per esempio con *sleep()*. Questa attesa deve essere comunque passiva, cioè non deve consumare tempo di CPU

6. riprende dal punto 1.

Il processo ***incrocio*** esegue in sequenza e di continuo le seguenti operazioni:

1. attende che il processo ***garage*** gli comunichi che ci sono 4 automobili che devono attrraversare l'incrocio. L'attesa deve essere comunque passiva, cioè non deve consumare tempo di *CPU*
2. determina quale automobile, secondo le regole di precedenza del codice della strada, può attraversare l'incrocio. Ne consegue che il processo ***incrocio*** deve conoscere la strada da cui ogni automobile proviene e quella in cui deve svoltare
3. aggiunge in coda al file i*ncrocio.txt* il numero della strada da cui proviene l'automobile scelta al punto 2.
4. emette un messaggio a video che indica il numero dell'automobile scelta al punto 2. per attraversare l'incrocio e la strada in cui svolterà
5. comunica all'automobile scelta al punto 2 che può attraversare
6. attende che l'automobile attraversi l'incrocio
7. ripete il punto 2.) fin quando tutte le auto hanno attraversato l'incrocio (in tutto quindi il punto 2. è eseguito per 4 volte
8. riprende dal punto 1

Ogni processo ***automobile*** esegue le seguenti operazioni:

1. attende che il processo ***incrocio*** comunichi che può attraversare l'incrocio
2. attraversa l'incrocio
3. aggiunge in coda al file *auto.txt* una riga contenente il numero della strada da cui proviene
4. comunica al processo ***incrocio*** che ha attraversato e termina

Ad applicazione terminata i due file "*incrocio.txt*" e "*auto.txt*" devono chiaramente essere uguali.

Nella pagina ***Unistudium*** delle specifiche del progetto trovate un file ***incrocio.h*** che contiene:
- la funzione ***EstraiDirezione (int iAutomobile)***, che estrae a sorte un numero di strada in cui dovrà svoltare l'automobile ricevuta come argomento e lo ritorna al chiamante. La strada ritornata è garantito essere diversa da quella da cui l'automobile proviene
- la funzione ***GetNextCar(int *Direzioni)*** , che riceve come argomento l'array in cui sono memorizzate le direzioni che le 4 automobili devono prendere e ritorna il numero dell'auto che può attraversare l'incrocio. Maggiori dettagli nei commenti all'interno del file.
L'automobile che può attraversare è determinata (più o meno...) in base alle regole del codice stradale.
Se volete potete utilizzare queste funzioni.

<u>Terminazione dell'applicazione:</u>

L'applicazione termina quando il processo ***incrocio*** riceve *SIGTERM*.
Ricevuto *SIGTERM* il processo incrocio:

- informa tutti gli altri processi che devono terminare

- attende la terminazione di tutti gli altri processi e termina esso stesso. In altre parole, il processo ***incrocio*** deve essere l'ultimo a terminare

Tutti i processi componenti l'applicazione devono terminare invocando *return()* o *exit()* e non "bruscamente" (per esempio perché hanno ricevuto un *signal* non gestito). Inoltre, ogni processo libererà quanto più possibile eventuali risorse allocate prima di terminare.

Evitare attese attive (es.: cicli continui che impegnino intensivamente la CPU).
Evitare attese passive NON RICHIESTE!

==========

Lo studente dovrà fornire anche uno script *Bash* che compila e avvia l'applicazione.

È gradito l'inserimento di commenti nel codice che ne facilitino la comprensione.

Il progetto per funzionare non deve richiedere l'esecuzione dall'utente *root*.

Indicazioni per la consegna:
Il progetto deve essere spedito via email al docente all'indirizzo ***fabio.rossi@unipg.it***, oppure messo a disposizione su un qualsiasi *cloud* e reso scaricabile tramite *link* da inviare sempre al suddetto indirizzo.

La consegna deve avvenire tassativamente almeno una settimana prima della data dell'esame orale. Il formato può essere tar, tgz o zip.