Simulazione ESAME
1) Prova Pratica 083 settembre 2022

La prova consiste di due esercizi, il primo è un esercizio di programmazione concorrente, il secondo è un esercizio bash.

Avrete a disposizione circa 1 ora e 15 minuti per risolvere i due esercizi e consegnarle la soluzione richiesta. \

Non potete navigare in internet cercando informazioni all'esterno.
Ovviamente, non potete comunicare con nessuno, in nessun modo.

Salvate i files nella directory /home/studente/CONSEGNA

Salvare spesso, per premunirsi in caso di crash.

Esercizio Esame Pratica - **211 – formicai (1/2)**

Ci sono **5 formicai disposti in circolo** e **indicizzati da 0 a 4,** con un palo verticale al centro del circolo.
Ci sono **4 fachiri**, ciascuno disposto inizialmente sui formicai da 0 a 3. **Su ciascun formicaio ci può stare solo un fachiro**. I fachiri si fanno mordere dalle formiche passando da un formicaio all'altro.
Ciascun fachiro passa continuamente da un formicaio al formicaio successivo dove

IndiceFormicaioSuccessivo = (IndiceFormicaioAttuale+1) % NumeroFormicai

Per passare da un formicaio al successivo, il fachiro si aiuta appoggiandosi al palo centrale fino a che non è completamente atterrato sul formicaio successivo. **Solo un fachiro alla volta può usare il palo**.
Per cominciare il passaggio dal formicaio attuale al formicaio successivo, il fachiro deve aspettare che il formicaio successivo sia libero e che anche il palo sia libero. Il passaggio dura un tempo variabile casualmente tra 1 e 3 secondi. Il formicaio, da cui il fachiro parte, viene considerato libero dopo che il fachiro è atterrata completamente sul formicaio successivo cioè allo scadere di quei 1-3 secondi.

Alcuni dettagli implementativi sulle variabili globali: sono messe a disposizione una variabile mutex per ciascun formicaio ed una variabile mutex per il palo. Il main colloca le persone sui formicai iniziali e lascia libero il palo ed il formicaio di indice 4.

Modellare ed implementare il sistema descritto, utilizzando dei thread POSIX per ciascuna figura (**persona**) ed avvalendosi delle opportune strutture dati per la sincronizzazione.
Utilizzare come base dell'esercizio i seguenti files, aggiungendo il codice necessario:

formicai.c DBGpthread.c DBGpthread.c printerror.h Makefile

NOTA BENE: Non potete usare condition variables.

Scrivere il Makefile per compilare e linkare i sorgenti. Inserire il necessario controllo di errore. In caso di errore grave, terminare il programma producendo un avviso a video. Non è obbligatorio utilizzare, nel codice che aggiungerete voi, le funzioni DBG* contenute nei files DBGpthread.* ma è consigliato.
La funzione **attendi**(int min, int max) implementata nel file **formicai.c** genera un numero casuale X compreso tra min e max ed attende X secondi prima di restituire il controllo al chiamante.

Esercizio Esame Pratica **- 211 – formicai (2/2)**

BASE DA COMPLETARE
https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/2022_09_01__BASE_es211_formicai_ONLINE.tgz

SOLUZIONE
https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/2022_09_01__es211_formicai_ONLINE.tgz

Esercizio Esame Pratica - **212 - divani.sh (1/2)**

Esiste un file divani.txt che contiene una riga per ciascun divano prodotto da una ditta. In ogni riga, separati da spazi, ci sono il nome del divano (una sola parola), la larghezza, l'altezza e la profondità espresse in cm.
Scrivere uno script divani.sh che prende esattamente due argomenti, la larghezza minima e l'altezza massima.
Lo script deve stampare a video l'elenco dei divani che rispettano due proprietà:
1) la loro larghezza è maggiore o uguale della larghezza passata come primo argomento;
2) la loro altezza è minore o uguale della altezza passata come secondo argomento allo script;
Usare come file divani.txt il seguente.

```
divano1 200 83 105
Divano2 170 85 115
divaNo3 185 65 99
divano4 170 85 115
divanO5 220 70 145
```

Provare ad eseguire lo script passando come argomenti i seguenti:
A) 200 80
Bisogna ottenere come output:
divanO5 220 70 145

B) 180 71
Bisogna ottenere come output:
divaNo3 185 65 99
divanO5 220 70 145

Esercizio Esame Pratica **- 212 - divani.sh (2/2)**

BASE DA COMPLETARE
https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/2022_09_01__BASE_es212_divani_ONLINE.tgz

SOLUZIONE
https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/2022_09_01__es212_divani_ONLINE.tgz

Simulazione ESAME
2) Prova Pratica giugno 2017
Produttori Consumatori

La prova consiste di due esercizi, il primo è un esercizio di programmazione concorrente, il secondo è un esercizio bash.

Avrete a disposizione circa 1 ora e 15 minuti per risolvere i due esercizi e consegnarle la soluzione richiesta. \

Non potete navigare in internet cercando informazioni all'esterno.
Ovviamente, non potete comunicare con nessuno, in nessun modo.

Salvate i files nella directory /home/studente/CONSEGNA

Salvare spesso, per premunirsi in caso di crash.

Esercizio Esame Pratica **- 211 – Prod Cons di due tipi**

Dobbiamo risolvere un problema di Produttori e Consumatoti come quello descritto a lezione ma con le seguenti differenze:

Ci sono **6 produttori**, **4 consumatori di tipo A** e **3 consumatori di tipo B. Tutti producono e consumano dei numeri interi e li depositano in un buffer con un solo posto.**

La caratteristica del problema è che per poter prelevare, occorre che prelevino contemporaneamente mezzo buffer ciascuno un consumatore di tipo A ed uno di tipo B.

Per semplicità si aspetta prima l'arrivo del consumatore A e poi l'arrivo del consumatore B. Quando sono arrivati entrambi può partire il prelievo contemporaneo. Quando entrambi hanno finito di prelevare la loro parte si può considerare il buffer vuoto.

Completare il codice contenuto nel file prod_ConsA+B.c

Il codice del produttore è già completo, quello del consumatore va completato.

Aggiungete quel che serve.

Modellare ed implementare il sistema descritto, utilizzando dei thread POSIX per ciascuna figura (**produttore consumatore**) ed avvalendosi delle opportune strutture dati per la sincronizzazione.

La figura del consumatore si comporta come tipo A o tipo B a seconda di quali parametri gli vengono passati.

Utilizzare come base dell'esercizio i seguenti files, aggiungendo il codice necessario:

prod_ConsA+B.c DBGpthread.c DBGpthread.c printerror.h Makefile

Scrivere il Makefile per compilare e linkare i sorgenti. Inserire il necessario controllo di errore. In caso di errore grave, terminare il programma producendo un avviso a video. Non è obbligatorio utilizzare, nel codice che aggiungerete voi, le funzioni DBG* contenute nei files DBGpthread.* ma è consigliato.

Codice da completare

Esercizio Esame Pratica **- 211 – Prod Cons di due tipi**

BASE DA COMPLETARE
https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/prod_ConsA+B.DACOMPLETARE.tgz

SOLUZIONE
https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/prod_ConsA+B.SOLUZIONE.tgz

Esercizio Esame Pratica - **59 - conta multe uguali**

Sia dato un file di testo **multe.txt** in cui, in ciascuna riga, e' contenuto il nome e il cognome di un guidatore, l'ammontare della multa e la data (giorno, mese, anno) in cui la multa gli è stata comminata.
I diversi campi in ciascuna riga sono separati da TAB o da spazi bianchi, scegliete voi.
Le righe del file **sono ordinate in modo crescente secondo il valore della multa**.
Ovviamente lo stesso valore può esistere più volte nel file.
Scrivere **uno** script bash, avente nome **contamulte.sh**, che viene eseguito lanciando questa riga di comando **./contamulte.sh < multe.txt**
e che produce il seguente risultato:

per ciascun valore di multa presente in quel file **multe.txt**, stampare una riga di testo contenente : il valore della multa e il numero di volte che quel valore si ripete nel file.

NB: un esempio di file multe.txt e' contenuto qui:
https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/2017_06_12__BASE_es59_contamulte.tgz
Se volete potete sostituire i caratteri bianchi con dei TAB.

esempio di file multe.txt

Luca Andreucci	**57**	11 maggio 2014
Giovanni Pau	**57**	20 aprile 2015
Luca Andreucci	**123**	28 luglio 2015
Astolfo Isoardi	**160**	21 dicembre 2015
Vittorio Ghini	**1000**	17 febbraio 2016
Gilles Villeneuve	**1000**	1 gennaio 1978

output da produrre

57	**2**
123	**1**
160	**1**
1000	**2**

Soluzione
https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/2017_06_12_es59_contamulte.SOLUZIONE.tgz

Simulazione ESAME
3) Altro esercizio Prova Pratica
5 filosofi

Questo è un esercizio di programmazione concorrente.

Esercizio Esame Pratica – **5 filosofi**

Questo esercizio vuole realizzare il sistema dei 5 filosofi a cena in cui:

NON si può adottare l'approccio in cui il filosofo prende le due forchette ASSIEME con un'unica operazione atomica.

Bisogna prendere le forchette una alla volta.

Per prendere ciascuna forchetta si impiega un tempo di circa 2 secondi.

Una volta prese le forchette il filosofo mangia per circa 11 secondi.

Il rilascio delle forchette invece può essere istantaneo e svolto assieme.

Utilizzare come base dell'esercizio i seguenti files, aggiungendo il codice necessario:

https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/5filosofiNo2ForchetteAssieme.DACOMPLETARE.tgz

Suggerimento: usate un vettore di mutex (già definito), una mutex per ciascuna forchetta ed un vettore con lo stato di occupazione di ciascuna forchetta.

Potete aggiungere altre variabili globali, se vi servono.

Scrivere il Makefile per compilare e linkare i sorgenti. Inserire il necessario controllo di errore. In caso di errore grave, terminare il programma producendo un avviso a video.

Non è obbligatorio utilizzare, nel codice che aggiungerete voi, le funzioni DBG* contenute nei files DBGpthread.* ma è fortemente consigliato.

ima di restituire il controllo al chiamante.

SOLUZIONE

https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SIMULAZIONITEMP/5filosofiNo2ForchetteAssieme.SOLUZIONE.tgz