Prova Pratica n. 054 - 17 luglio 2020 online

La prova consiste di due esercizi, il primo è un esercizio di programmazione concorrente, il secondo è un esercizio bash.
I files da utilizzare e da completare per l'esercizio di programmazione concorrente si trovano qui:

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/2020_07_17_ONLINE/BASE_2020_07_17__es153_piattello_ONLINE.tgz
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/2020_07_17_ONLINE/BASE_2020_07_17__es154_voti_ONLINE.tgz

Le soluzioni si trovano qui:

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/2020_07_17_ONLINE/SOLUZIONE_2020_07_17__es153_piattello_ONLINE.tgz
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/2020_07_17_ONLINE/SOLUZIONE_2020_07_17__es154_voti_ONLINE.tgz

Avrete a disposizione circa 1 ora per risolvere i due esercizi e consegnare la soluzione richiesta.
Potete usare il materiale contenuto in http://www.cs.unibo.it/~ghini/didattica/TREE4OS1920.tgz che dovete già avere scaricato prima di questo esame

Non potete navigare in internet cercando informazioni all'esterno.
Ovviamente, non potete comunicare con nessuno, in nessun modo.
Salvate i files spesso, per premunirsi in caso di crash.

Esercizio Esame Pratica Online **- 153 - piattello**

In un poligono di tiro 10 tiratori si allenano sparando a dei piattelli lanciati in volo.
Ogni 8 secondi una macchina (il main) lancia in aria un piattello, il piattello vola per 3 secondi e infine cade a terra. Nel momento in cui il piattello viene lanciato, tutti i tiratori cercano di colpirlo. ogni tiratore impiega un tempo variabile tra 2 e 4 secondi per mirare e poi, SE IL PIATTELLO E' ANCORA IN VOLO, spara impiegando 0 secondi e SICURAMENTE colpisce il piattello. Se invece il piattello è già caduto a terra il tiratore grida di rabbia e non spara. Il piattello se colpito comunque continua il volo e può essere colpito da altri tiratori. I tiratori possono mirare e sparare anche tutti assieme.
Ogni piattello è un diverso thread. Circa ogni 8 secondi la macchina (il main) crea un nuovo thread piattello. Ogni thread piattello inizia il volo e alla fine del volo cade a terra e termina. Prima di terminare ogni piattello stampa un messaggio e dice se è stato colpito o no.
Ciascuno dei 10 tiratori è un thread che aspetta il lancio dei piattelli e cerca di colpirli, senza mai terminare.
Modellare ed implementare il sistema descritto, utilizzando dei thread POSIX per ciascna figuhra (**piattello** e **tiratore**) ed avvalendosi delle opportune strutture dati per la sincronizzazione.
Utilizzare come base dell'esercizio i seguenti files, aggiungendo il codice necessario:

piattello.c DBGpthread.c DBGpthread.h printerror.h

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.
Implementare la soluzione a partire dai files .c e .h che vi ho consegnato.
Per implementare l'attesa di un tempo variabile impiegato da un tiratore per mirare e sparare, usare la funzione **attendi(int min, int max)**. La funzione attendi(int min, int max) implementata nel file **piattello.c** genera un numero casuale X compreso tra min e max ed attende X secondi prima di restituire il controllo al chiamante.
Non è obbligatorio utilizzare, nel codice che aggiungerete voi, le funzioni DBG* contenute nei files DBGpthread.* ma è fortemente consigliato.

Esercizio Esame Pratica - **154 - voti.sh**

Nell'archivio che vi ho messo a disposizione, nella directory BASE_es154, ci sono due file esame1.txt ed esame2.txt che contengono ciascuno i risultati di una prova pratica di sistemi operativi.
In ciascuna riga dei file c'é il numero di matricola di uno studente presente alla prova ed il voto ottenuto in quella prova, separati da spazi bianchi.

La prova pratica, i cui risultati sono nel file esame2.txt, è la prova più recente.

Qualche studente potrebbe essere stato presente solo ad una delle due prove.
Se uno studente è stato presente alla seconda prova allora il risultato della prima prova non è più valido e vale solo il voto della seconda.

Implementare uno script voti.sh che legge i due file esame1.txt ed esame2.txt e mette in output delle righe; ciascuna riga contiene il voto più recente ottenuto da uno studente preceduto dal numero di matricola di quello studente.
Il pratica lo script voti.sh deve mettere in output il voto più recente di ciascuno studente, nello stesso formato dei file di input.
Nella directory BASE_es154 c'è un file voti.sh contenente qualche suggerimento.
Se volete potete utilizzare dei file temporanei per salvare risultati parziali, ma non è necessario.

Prova Pratica 000 sync

Un main genera 5 pthread e ne attende la fine.
Ciascun thread aspetta 1 secondo e poi aspetta che tutti i 5 thread siano dentro la funzione che li implementa e solo allora i thread possono terminare.
Però, ogni thread ottiene il permesso di terminare nello stesso ordine con cui i thread si sono messi in attesa dell'arrivo degli altri thread.

Modellare ed implementare il sistema descritto, utilizzando dei thread POSIX ed avvalendosi delle opportune strutture dati per la sincronizzazione.

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.

SOLUZIONI
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/SYNCPOINT/SYNCPOINT.tgz

Prova Pratica n. 055 - 10 settembre 2020 online

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

I files da utilizzare e da completare per l'esercizio di programmazione concorrente si trovano qui:

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/2020_09_10_ONLINE/BASE_2020_09_10__es155_vacche_ONLINE.tgz

Le soluzioni si trovano qui:

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/2020_09_10_ONLINE/SOLUZIONE_2020_09_10__es155_vacche_ONLINE.tgz

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/2020_09_10_ONLINE/SOLUZIONE_2020_09_10__es156_signal_ONLINE.tgz

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

Potete usare il materiale contenuto in http://www.cs.unibo.it/~ghini/didattica/TREE4OS1920.tgz che dovete già avere scaricato prima di questo esame

Non potete navigare in internet cercando informazioni all'esterno.
Ovviamente, non potete comunicare con nessuno, in nessun modo.
Salvate i files spesso, per premunirsi in caso di crash.

Esercizio Esame Pratica Online **- 155 - vacche**

In un allevamento di bovini, su una parete ci sono 3 mangiatoie una a fianco all'altra, numerate da 0 a 2. Le mangiatoie sono sempre piene di mangime. Solo una vacca per volta può mangiare in una mangiatoia. Circa ogni 20 secondi, all'esterno della stalla un bovaro pazzo incendia della paglia, la fa bruciare per 5 secondi e poi la spegne e così all'infinito.

Ciascuna vacca impiega per mangiare un tempo casuale compreso tra 5 e 7 secondi, poi va in giro per 10 secondi e poi torna alla mangiatoia e aspetta che ci sia posto per mangiare ancora, e così via. Però, solo mentre sta mangiando, ad ogni secondo la vacca alza la testa, se vede che la paglia sta bruciando si spaventa, smette di mangiare e va in giro per 10 secondi, poi ritorna e cerca di trovare ancora una mangiatoia in cui ricominciare a mangiare.

Ci sono 6 vacche ed 1 bovaro pazzo.

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

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

vacche.c DBGpthread.c DBGpthread.h printerror.h

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.

Implementare la soluzione a partire dai files .c e .h che vi ho consegnato.

Per implementare l'attesa di un tempo variabile impiegato da una vacca per mangiare, usare la funzione **attendi(int min, int max)**. La funzione attendi(int min, int max) implementata nel file **vacca.c** genera un numero casuale X compreso tra min e max ed attende X secondi prima di restituire il controllo al chiamante.

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

Esercizio Esame Pratica **- 156 - signal**

Realizzare due script bash, denominati padre.sh e figlio.sh.

Lo script padre.sh mette in esecuzione in background lo script figlio.sh e si predispone per ricevere un signal SIGUSR2.
Dopo avere ricevuto il signal SIGUSR2 il padre termina se stesso.

Il figlio.sh attende 5 secondi, poi manda al padre il signal SIGUSR2 e poi termina se stesso.