6i Esercizi (laboratorio lez 12)
Progr. Concorrente e Vari

consiglio: installare e provare editor leggero meno buggato di gedit: **geany**
sudo apt-get update ; sudo apt-get install geany

Contenuti:

SIMULAZIONE PROVA PRATICA 0c

Esercizio 1061 Programmazione concorrente semplice
Esercizio 1062 Programmazione concorrente complicato
Esercizio 1063 script bash (stdout, espressioni condizionali if e text utils (sort etc etc))
Soluzioni esercizi 1061, 1062, 1063

ESERCIZI vari, per casa

Esercizio 64 - Programmazione Concorrente
Esercizio 65 - script , stdout, stdin, ridirezionamenti, tail, Soluzioni esercizi 65, 66, 67, 68,
Soluzioni esercizi 64, 65

Simulazione Prova Pratica 0c

Download Materiale:

Scaricare il file con le **dispense** e gli **esempi** svolti a lezione

wget http://www.cs.unibo.it/~ghini/didattica/TREE4OS1617.tgz

Decomprimere l'archivio scaricato: tar xvzf TREE4OS1617.tgz

Viene creata una directory **TREE4OS1617** con dentro una sottodirectory **sistemioperativi** con dentro tutto il **materiale**.

Potete navigare tra il materiale con un normale browser aprendo l' URL

file://home/studente/VOSTRADIRECTORY/TREE4OS1617/sistemioperativi/dispenseSistOp1617.html

Esercizi d'esame: per chi ha difficoltà a superare la prova pratica, ho previsto due tipi di prove:

A. **una prova COMPLICATA**, e' la modalità normale che vi permette di raggiungere un **voto massimo** (nella prova pratica stessa) di **30Lode** ,
B. ed **una prova SEMPLICE**, un po' **meno complicata**, che però vi permette di raggiungere un **voto massimo di 24** perché l'esercizio di programmazione concorrente é meno difficile.

Scegliete voi quale prova svolgere in funzione della vostra preparazione e del grado di angoscia e panico che vi sommerge.

La prova COMPLICATA è composta dagli esercizi **1062 e 1063**,

La prova SEMPLICE è composta dagli esercizi **1061 e 1063**.

Come vedere l'esercizio 1063 è comune alle due prove.

Svolgete SOLO gli esercizi della prova che vi interessa.

I file da consegnare **devono** essere collocati nella directory **CONSEGNA** dentro la home directory dell'utente studente.

Esercizio 1061 - **soccorsi (semplice)**

Una scala in un'università è un po' malridotta.
Ogni professore che passa inciampa e si rompe un piede, cominciando ad urlare imprecazioni irriferibili. Tanto è lo strepitare che nasce il dubbio se si tratti di un ferimento o di una possessione demoniaca.
Nel dubbio, accorrono un medico ed un esorcista. I soccorsi al ferito possono essere effettuati solo dopo che entrambi (medico ed esorcista) hanno raggiunto il malcapitato professore.
L'intervento dei due soccorritori dura 2 secondi. Al termine entrambi se ne vanno.
Dopo un secondo che entrambi se ne sono andati, il professore si rialza, fa 4 secondi di lezione, poi ripassa sulla scala e cade di nuovo, e così via all'infinito.
Ci sono un medico, un esorcista e un professore.

Modellare ed implementare il sistema descritto, **utilizzando dei thread POSIX** per ciascuna figura (professore, esorcista, medico) ed avvalendosi delle opportune strutture dati per la sincronizzazione.
Scrivere il Makefile per compilare e linkare i sorgenti. La mancanza del Makefile viene considerato un errore grave.
Occorre inserire il controllo di errore nelle chiamate a funzione delle librerie dei pthread. In caso di errore grave, terminare il programma producendo un avviso a video.

Esercizio 1062 - **strumenti (complicato)**

In una fabbrica ci sono diversi tipi di strumenti da usare, in particolare:

2 strumenti di tipo S1, 4 strumenti di tipo S2 e 2 strumenti di tipo S3.

Nella fabbrica si effettuano due tipi di lavorazioni, L1 e L2. La lavorazione L1 impiega 5 secondi, la lavorazione L2 solo 4 secondi. Una volta cominciata, la lavorazione non può essere sospesa.

In ciascuna lavorazione si usano alcuni strumenti per alcuni secondi, come specificato nella tabella. Nella tabella la prima colonna da sinistra indica la lavorazione, la seconda colonna da colonna indica uno strumento usato in quella lavorazione, le successive colonne indicano i secondi in cui lo strumento di quella riga viene usato nella lavorazione a cui la riga appartiene.

Ad esempio, alla lavorazione L1 serve lo strumento S1 nel secondi 1, 2, 3, 4 e 5, inoltre serve lo strumento S2 nei secondi 2 e 3. Lo strumento S3 serve nei secondi 4 e 5.

Uno strumento può ovviamente essere usato da una sola lavorazione alla volta. Al termine di ciascuna lavorazione, la lavorazione rilascia tutti gli strumenti utilizzati, aspetta 1 secondo e poi ricomincia dall'inizio.

Esistono 2 lavorazioni L1 e 2 lavorazioni L2.

	sec	1	2	3	4	5
L1	S1	1	1	1	1	1
	S2		1	1		
	S3				1	1
L2	S1				1	
	S2		1	1		
	S3	1	1	1	1	

Modellare ed implementare il sistema descritto, **utilizzando dei thread POSIX** per ciascuna figura (lavorazioni L1 e L2) ed avvalendosi delle opportune strutture dati per la sincronizzazione. **Stabilire come assegnare gli strumenti alle lavorazioni, anche in modo non ottimo, ma adatto ad evitare deadlock.**

Scrivere il Makefile per compilare e linkare i sorgenti. La mancanza del Makefile viene considerato un errore grave. Occorre inserire il controllo di errore nelle chiamate a funzione delle librerie dei pthread. In caso di errore grave, terminare il programma producendo un avviso a video.

Esercizio 1063 - **contaseparatamente.sh**

Realizzare uno script bash **contaseparatamente.sh** che effettua le seguenti operazioni:
Lo script prende un numero variabile di argomenti, al massimo 9.
Ciascun argomento è il nome di un file.
Lo script scrive sullo **standard output** il numero totale di righe dei file passati come argomenti di indice pari.
Lo script scrive sullo **standard error** il numero totale di righe dei file passati come argomenti di indice dispari.

Realizzare poi uno script bash **lancia.sh** che:
lancia l'esecuzione dello script contaseparatamente.sh passando allo script come argomenti i primi 7 file che si ottengono come standard output del comando ls -S1 /usr/include/*.h
Il modo in cui lo script lancia.sh lancia l'esecuzione di contaseparatamente.sh deve ridirezionare sia lo standard output che lo standard error di contaseparatamente.sh sullo standard error di lancia.sh stesso.

Mi raccomando, verificare che lo script lancia.sh produca il suo output sullo standard error.

SOLUZIONI

Soluzione Esercizio 1061 es1061_soccorsi_semplice.c
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es1061_soccorsi_semplice.tgz

Soluzione Esercizio 1062 es1062_strumenti_complicato.c
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es1062_strumenti_complicato.tgz

Soluzione Esercizio 1063 es1063_contaseparatamente.sh
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es1063_contaseparatamente.tgz

FINE SIMULAZIONE PROVA PRATICA 0c

da qui in avanti altri esercizi per casa

Esercizio 64 - figliarecomeconigli.c

In un programma non devono mai essere contemporaneamente in esecuzione più di 5 thread (circa) , più il main.
Il main di un programma crea un nuovo thread ogni 5 secondi.
Ciascun thread appena creato stampa a video il numero di thread presenti.
Ciascun thread figlio cerca di creare ogni secondo un nuovo thread nipote, ma se non può crearlo perché ci sarebbero troppi thread allora quel thread figlio termina.
Ciascun thread nipote cerca di creare .... e così via ...

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. La mancanza del Makefile viene considerato un errore grave.
Occorre inserire il controllo di errore nelle chiamate a funzione delle librerie dei pthread. In caso di errore grave, terminare il programma producendo un avviso a video.

Esercizio 65 - ridirezionamentocircolare.sh

Scrivere gli script A.sh B.sh lancia.sh ed altri se occorrono.

Lo script **A.sh** comincia scrivendo il valore 1 sul proprio standard output.
Successivamente lo script A.sh riceve continuamente sul proprio standard input delle righe di testo che contengono un valore numerico.
Per ciascuna riga ricevuta, lo script A.sh legge il valore numerico, lo incrementa di 1, attende 1 secondo, e scrive il valore incrementato sullo standard output.

Lo script **B.sh** riceve continuamente sul proprio standard input delle righe di testo che contengono un valore numerico.
Per ciascuna riga ricevuta, lo script B.sh legge il valore numerico e lo scrive sul proprio standard output senza incrementarlo. Inoltre, lo scrive anche in un file B.txt

Lo script **lancia.sh** deve lanciare i due script A.sh e B.sh (ed eventualmente altri script, se necessario) in modo tale che:

- l'output di A.sh vada a finire nello standard input di B.sh
- l'output di B.sh vada a finire nello standard input di A.sh

In tal modo, A.sh e B.sh continueranno a passarsi quel valore incrementandolo una volta si e una no.

SOLUZIONI

Soluzione Esercizio 64 es64_figliarecomeconigli.c
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es64_figliarecomeconigli.tgz

Soluzione Esercizio 65 es65_ridirizionamentocircolare
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/es65_ridirezionamentocircolare.tgz