Prova Pratica 018
turno 1 gruppo 2

2017 febbraio 06

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

SUGGERIMENTO PRATICO PRELIMINARE
PER CHI VUOLE FARE
l' esercizio in cui servono i PROCESSI

Se avete in esecuzione tanti processi che hanno tutti nome

processo.exe

e li volete uccidere tutti,

potete killare tutti quei processi utilizzando il comando:

killall processo.exe

Prova Pratica 018 - turno 1 gruppo 2

Download Materiale:

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

wget http://esameso.csr.unibo.it/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/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.

La prova COMPLICATA è composta dagli esercizi **49 e 50**,

La prova SEMPLICE è composta dagli esercizi **48 e 50**.

Come vedere l'esercizio 50 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 Esame Pratica **- 48 - megapizze (semplice)**

5 persone fameliche (identificate da un indice intero tra 0 e 4) mangiano ripetutamente pizze, in una pizzeria dove c'è solo 1 tavolo da 4 posti. La pizzeria fa un solo tipo di pizza, quindi non si può scegliere.
Una megapizza deve essere mangiata da 4 persone contemporaneamente. Fino a che 4 persone non si sono sedute a tavola, nessuno può cominciare a mangiare. All'arrivo della quarta persona al tavolo la megapizza si materializza istantaneamente sul tavolo e tutte le 4 persone sedute al tavolo possono cominciare a mangiare.
Le 4 persone impiegano 6 secondi per mangiare la pizza, poi si alzano tutte quante dal tavolo, vanno via dalla pizzeria e tornano ciascuna dopo altri (2+indice) secondi. Il tavolo viene considerato libero solo dopo che tutte le persone si sono alzate.
Quando una persona entra nella pizzeria, se al tavolo ci sono meno di 4 persone **e queste non hanno ancora mangiato**, la persona si siede a quel tavolo.
Quando al tavolo ci saranno 4 persone, allora le persone potranno cominciare a mangiare.
Quando una persona entra nella pizzeria, se il tavolo è completamente pieno (cioè 4 persone) la persona aspetta che il tavolo si sia completamente liberato (0 persone) e poi si siede a quel tavolo.

Modellare ed implementare il sistema descritto, utilizzando dei PROCESSI per ciascuna figura (persona) ed avvalendosi delle opportune strutture dati per la sincronizzazione. Scrivere il Makefile per compilare e linkare i sorgenti. La mancanza del Makefile, così come un Makefile non funzionante, 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 Esame Pratica **- 49 - megapizze (complicato)**

5 persone fameliche (identificate da un indice intero tra 0 e 4) mangiano ripetutamente pizze, in una pizzeria dove c'è solo 1 tavolo da 4 posti. Una megapizza deve essere mangiata da 4 persone contemporaneamente. Fino a che 4 persone non si sono sedute a tavola con davanti la maxi-pizza, nessuno può cominciare a mangiare. Le 4 persone impiegano 3 secondi per mangiare la pizza, poi si alzano tutte quante dal tavolo, vanno via dalla pizzeria e tornano dopo altri (2+indice) secondi. Il tavolo viene considerato libero solo dopo che tutte le persone si sono alzate.
Il **pizzaiolo** attende di essere chiamato a fare le pizze. Quando viene chiamato, il pizzaiolo prepara una pizza immediatamente e la porta al tavolo subito, poi si rimette in attesa.
Quando una persona entra nella pizzeria, se al tavolo ci sono meno di 4 persone e queste non hanno ancora mangiato, la persona si siede a quel tavolo. Quando al tavolo viene occupato il 3° posto (cioè la maggioranza dei posti) viene deciso che pizza mangiare e il 3° arrivato chiama il pizzaiolo.
Quando al tavolo ci saranno 4 persone e una pizza gigante, allora le persone potranno cominciare a mangiare. Attenzione, la pizza potrebbe arrivare prima o dopo l'arrivo della 4° persona.
Quando una persona entra nella pizzeria, se il tavolo è completamente pieno (cioè 4 persone) la persona aspetta che il tavolo si sia completamente liberato (0 persone) e poi si siede a quel tavolo.

Modellare ed implementare il sistema descritto, utilizzando dei thread POSIX per ciascuna figura (persona, pizzaiolo) ed avvalendosi delle opportune strutture dati per la sincronizzazione. Scrivere il Makefile per compilare e linkare i sorgenti. La mancanza del Makefile, così come un Makefile non funzionante, 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 Esame Pratica **- 50 - selezionarighe**

Scrivere uno script bash **selezionarighe.sh** che accetta due argomenti a riga di comando.
Il primo argomento sarà il percorso, relativo o assoluto, per identificare univocamente un file esistente.
Il secondo argomento sarà una stringa di testo. Se allo script viene passato un numero di argomenti diverso da 2, lo script deve mandare sullo **standard error** il messaggio "numero argomenti errato" e poi terminare restituendo come codice d'errore 1.
Se allo script vengono passati esattamente 2 argomenti, lo script deve controllare se il file specificato dal primo argomento esiste. Se il file non esiste, lo script deve mandare sullo **standard error** il messaggio "argomento non file" e poi terminare restituendo come codice d'errore 2.
Se invece quel **file** esiste, allora lo script deve far eseguire **in background** una sequenza di comandi, o di script, che:

prima aspetta 2 secondi, e poi
seleziona tutte le righe del **file** che contengono la parola passata come secondo argomento allo script, conta quelle righe e scrive il numero di tali righe **aggiungendolo** in una riga di testo in fondo al file OUTPUT.txt, nella directory in cui viene lanciato lo script.

Nel frattempo lo script termina restituendo 0.

Infine, scrivere uno script **chiama.sh** che esegue due volte lo script selezionarighe.sh, passandogli come argomento:
la prima volta il percorso di un file che esiste **/usr/include/stdio.h** e la parola **int**
la seconda volta il percorso di un file che non esiste **./VICSCEMO.txt** e la parola **int**
La seconda chiamata serve a evidenziare se la gestione dell' errore funziona correttamente.

Esercizio Esame Pratica **- suggerimenti per il 50**

Se non sapete come fare output sullo standard error,
cercate di ridirigere l'output del comando echo sullo standard error
prendendo spunto dalla slide intitolata Ridirezionamenti di Stream di I/O (5)
nel file 4_InterfacciaUtenteACaratteri_BashScripting.pdf