Sistemi Operativi CANALE M-Z

Esame Scritto
28 Febbraio 2020

Cognome: **Nome:** **Matricola:**

Votazione massima 30/30
Soglia per superare la prova 18/30

Durata 3 ore

Domanda 1: max 6 punti

Si descrivano in dettaglio le seguenti politiche di scheduling:

- Multilevel Queue Scheduling (MQS), a priorità fissa con prelazione
- Round Robin (RR)
- First Come First Server (FCFS);

Risposta: dare la risposta in un file denominato **domanda1_Cognome_Nome.txt**

Si supponga di avere un Sistema Operativo che utilizza un scheduler di tipo "Multilevel Queue Scheduling (MQS)", a priorità fissa con prelazione con due code gestite rispettivamente con le politiche di scheduling:

- Coda 1: Round Robin (RR) con **quanto di tempo q=3**;
  - Priorità ALTA
- Coda 2: First Come First Server (FCFS);
  - Priorità BASSA

Si supponga che lo scheduler riceva i 7 processi, A, B, C, D, E, F e G con le seguenti caratteristiche:

Processo	Durata CPU–burst	Tempo di arrivo	Priorità
A	5	0	BASSA
B	3	2	ALTA
C	5	4	BASSA
D	8	7	ALTA
E	4	8	ALTA
F	6	11	ALTA
G	2	12	ALTA

Si descriva le sequenza di esecuzione dei processi tramite **diagramma di Gantt**, e si valuti **il tempo medio di attesa dei processi**.

Risposta: Si scriva la risposta in questo foglio continuando sul retro

Domanda 2: max 6 punti

Descrivere in dettaglio le differenze tra processi e threads, con particolare attenzione alla differenza tra la condivisione di: Variabili Statiche, Stack, Heap, Files.

Risposta: dare la risposta in un file denominato **domanda2_Cognome_Nome.txt**

Esercizio al Calcolatore: max 18 punti

Si voglia realizzare un sistema Client/Server con un Server e diversi Client.

Ogni Client invia un vettore di **N** elementi di tipo calcolo, così definito:

```
#define N 3
typedef struct {
        float risultato, a;
        int b;
} calcolo;
calcolo vettore[N];
```

Si suppone che ogni Client invia l'intero vettore riempendo per ciascun elemento, solo i campi **a** e **b**. **ATTENZIONE: il Client riempie prima tutto il vettore e poi una volta riempito lo manda al server.**

Per ciascun vettore ricevuto e per ciascun elemento del vettore, il Server riempie il campo **risultato**, inserendo il valore del prodotto tra i campi **a** e **b**. Una volta effettuato il riempimento dell'intero vettore, il Server risponde al Client inviando il vettore completo. **ATTENZIONE: anche in questo caso, il Server riempie prima tutto il vettore e poi una volta riempito lo manda al client.**

Il Client stampa a video il contenuto di ciascun campo risultato ricevuto dal server.

Quando il Client vuole terminare (su richiesta dell'utente), chiude la connessione. In tal caso, il Server deve automaticamente chiudere anch'esso la connessione con il client. **ATTENZIONE: il Client interrompe la connessine con il Server senza doverlo prima avvertire. Il Server deve capire automaticamente che la connessione con il client è stata interrotta dal client stesso.**

Si definisca un programma Server e un programma Client che realizzi quanto detto utilizzando i Sockets e i threads. Si deve supporre che per ogni client connesso, il server crei un thread apposito che gestisce la comunicazione con il client. Quando il client si disconnette, il thread finisce. Per evitare che il Server debba invocare la pthread_join, si deve cambiare lo stato di ciascun thread in modo detached.

Si supponga che il server non termini mai, e dunque è necessario utilizzare il comando di kill per terminarlo.

Risposta: Si scrivano i due programmi al computer in due file in c, con i seguenti nomi:

Client_COGNOME_NOME.c
Server_COGNOME_NOME.c