

# Sistemi Operativi II - Corso di Laurea in Ingegneria Informatica

Appello d'esame del 29/3/2006 – Docente: Francesco Quaglia

Cognome \_\_\_\_\_ Nome \_\_\_\_\_

*Si raccomanda di scrivere il proprio cognome e nome su questo foglio e di utilizzarlo come cartellina per contenere i fogli con le risposte. Se si considera ambigua una domanda, scrivere la propria interpretazione e rispondere conseguentemente. Chi viene scoperto che copia o consulta appunti verrà bocciato.*

**AVVERTENZA: rispondere in modo schematico e sintetico, senza essere prolissi. La capacita' di sintesi verra' premiata.**

## Domanda 1 (5 punti)

Descrivere le modalita' di passaggio dei parametri (sia di input che di output) nel contesto delle RPC (Remote Procedure Call).

## Domanda 2 (5 punti)

Descrivere lo schema di paginazione a livelli multipli in sistemi a memoria virtuale, discutendo (1) la relazione tra tale schema ed la strutturazione del microcodice (in termini di cicli macchina) per la risoluzione di un "TLB miss" e (2) gli effetti di tale schema nel caso in cui il kernel del sistema operativo sia completamente residente in memoria di principale.

## Domanda 3 (5 punti)

Descrivere gli algoritmi di schedulazione del disco SCAN, CSCAN ed FSCAN, discutendone vantaggi e svantaggi. Si consideri inoltre la sequenza di operazioni di I/O corrispondenti alle tracce: 16, 63, 49, 7, 67 e 44. Le operazioni di I/O vengono richieste nell'ordine delle tracce nella sequenza. Supponendo che (1) il disco abbia 100 tracce (numerata da 1 a 100), (2) la testina sia inizialmente posizionata sulla traccia 90, (3) la direzione di movimento della testina per SCAN sia inizialmente decrescente, (4) CSCAN ed FSCAN lavorino con scansione crescente, (5) per FSCAN la sequenza di scheduling corrente comprenda solo le richieste per le tracce 16, 63 e 49, calcolare la lunghezza media di scansione dei tre algoritmi.

## Domanda 4 (5 punti)

Considerare un sistema formato da tre processi concorrenti che competono nell'accesso ai MUTEX M1, M2 ed M3, per gestire una sezione critica. Determinare se è possibile il verificarsi di un deadlock. In caso affermativo, fornire una sequenza di interleaving che lo produce ed un algoritmo per la prevenzione dello stato di deadlock.

Mutex M1, M2, M3

```

P1
while (true) {
1.1   NonCriticalReg();
1.2   MutexLock(&M1);
1.3   MutexLock(&M2);
1.4   CriticalRegion();
1.5   MutexUnlock(&M2);
1.6   MutexUnlock(&M1);
}

P2
while (true) {
2.1   NonCriticalReg();
2.2   MutexLock(&M2);
2.3   MutexLock(&M3);
2.4   CriticalRegion();
2.5   MutexUnlock(&M3);
2.6   MutexUnlock(&M2);
}

P3
while (true) {
3.1   NonCriticalReg();
3.2   MutexLock(&M3);
3.3   MutexLock(&M1);
3.4   CriticalRegion();
3.5   MutexUnlock(&M1);
3.6   MutexUnlock(&M3);
}
```

### Domanda 5 (10 punti – la soglia per la sufficienza e' pari ad almeno 5 punti su questa domanda)

Il codice che segue implementa un server TCP che mantiene un contatore numerico. Tale contatore viene incrementato ad ogni richiesta di connessione da parte di un qualsiasi client. Ogni volta che il contatore viene incrementato, il nuovo valore viene stampato a video. Il comportamento atteso e' che venga stampata a video sul server una sequenza di numeri crescente senza "salti" o "doppioni". Alla terminazione del programma server (tramite la pressione dei tasti Ctrl+C), dovra' essere stampato a video il valore attuale del contatore.

Si chiede allo studente di completare il programma in tutte le sue parti mancanti, e di controllare in relazione all'uso dei segnali se questo sia corretto rispetto alla specifica. In caso negativo, spiegare i motivi e fornire una implementazione che soddisfi la specifica.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <signal.h>

int contatore=0;

void incrementa(){
    signal(SIGUSR1, incrementa);
    printf("Incremento il contatore.\n", contatore);
    contatore=contatore+1;
    printf("Il nuovo valore del contatore Ã¨ %d.\n", contatore);
}

int main(int argc, char *argv[]) {

    int servsocket, connsocket, len;
    struct sockaddr_in cliaddr, servaddr;

    printf("Server avviato.\n");
    memset ((char *)&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(3333);

    printf("Inizializzazione socket.\n");

    printf("Armo il segnale.\n");
    signal(SIGUSR1, incrementa);

    while (1) {
        len = sizeof(struct sockaddr_in);

        if (fork()==0) {
            close(servsocket);
            printf("Client connesso.\n");
            kill(getppid(), SIGUSR1);
            close(connsocket);
            exit(0);
        }

        close(connsocket);
    }
}
```

Il Sottoscritto, ai sensi della legge 675 del 31/12/96, autorizza il Docente a pubblicare in bacheca e su web i risultati della prova di esame. In fede

Firma leggibile: \_\_\_\_\_