

# Sistemi Operativi

## Esempio di strutturazione della prova scritta (Giugno 2003)

Il compito sarà formato da quattro o cinque quesiti. Ciascun quesito sarà caratterizzato da un punteggio massimo (*max*). A ciascuna risposta sarà assegnato un punteggio fra 0 e *max*, secondo il grado di completezza e correttezza della risposta. Il punteggio max di ciascuna prova sarà pari a 33 punti. Due punti aggiuntivi saranno assegnati a chi risponderà correttamente alla domanda (opzionale) sul sistema UNIX. Il punteggio finale sarà dato dalla somma dei punteggi assegnati alle risposte date ai quesiti. Se il punteggio è  $\geq 30$  e  $\leq 32$ , il voto finale sarà pari a 30. Se il punteggio finale è  $\geq 33$  il voto finale sarà 30 e lode.

Nel seguito sono riportati 5 gruppi di possibili domande. Un possibile compito può essere costruito selezionando una domanda da ciascun gruppo. (Ovviamente non tutte le combinazioni generano compiti di pari grado di difficoltà! Sarà mia cura calibrare le combinazioni di domande in modo tale da ottenere compiti con grado di difficoltà analogo).

### Esercizio 1

Quali sono gli "eventi" che portano alla creazione di un processo? Fornire una spiegazione generale. 2 punti in più per l'esemplificazione della creazione di un processo in UNIX.

Illustrare il modello a 5 stati di un processo spiegando il significato di ciascuno stato.

Quali informazioni vengono memorizzate di solito nel "Process Control Block"?

Illustrare gli approcci principali seguiti dai progettisti di sistemi operativi relativi all'esecuzione del sistema operativo stesso. Illustrare, per ciascun approccio, le differenze fra l'esecuzione del sistema operativo e l'esecuzione di processi utente.

Illustrare le differenze fra il *cambio* di processo e il cambio di *modo*. In particolare, precisare in quali situazioni è necessario effettuare l'una o l'altra operazione e quali sono vantaggi e svantaggi delle due operazioni.

Illustrare il concetto di thread. In particolare illustrare le strutture dati usate per gestire i thread in relazione alle strutture dati utilizzate per gestire i processi. Dare qualche esempio di utilità della esecuzione di thread multipli.

### Esercizio 2

Illustrare il concetto di sezione critica. Commentare il seguente codice (una delle "fasi" di costruzione dell'algoritmo di Dekker)

Garantisce la mutua esclusione? Previene la starvation? Se un processo fallisce dentro o fuori la sezione critica, gli altri processi possono accedere alla risorsa?

Commentare l'algoritmo per la mutua esclusione di Peterson (verrà riportato nel testo).

Illustrare il concetto di semaforo per la mutua esclusione. Scrivere la definizione del tipo di dato "semaforo" in C o pseudocodice) e il codice per l'implementazione delle principali operazioni sul semaforo.

Illustrare il concetto di semaforo per la mutua esclusione. Scrivere la definizione del tipo di dato "semaforo" in C o pseudocodice) e il codice per l'implementazione delle principali operazioni sul semaforo usando l'istruzione speciale di macchina "testset".

Commentare la seguente soluzione del problema del produttore/consumatore, nel caso di buffer di lunghezza infinita, che utilizza semafori binari. (può esserci la versione corretta così come una versione non corretta come quella mostrata a lezione)

Commentare la seguente soluzione del problema del produttore/consumatore, nel caso di buffer limitato, che utilizza semafori generici.

Commentare la seguente soluzione del problema del produttore/consumatore, nel caso di buffer limitato, che utilizza semafori generici. MODIFICARE SECONDO Problema 5.18 in Stallings

Commentare la seguente soluzione del problema dei lettori/scrittori che utilizza semafori generici. (può esserci una delle due soluzioni mostrate a lezione)

Illustrare il concetto di monitor con particolare riferimento al seguente esempio (produttore/consumatore con buffer limitato)

### **Esercizio 3**

Illustrare le principali caratteristiche dei sistemi a “scambio di messaggi” per la comunicazione fra processi. 2 punti in più se si forniscono le caratteristiche salienti delle “pipe” in UNIX.

Illustrare le quattro condizioni che causano lo stallo. Illustrare il funzionamento dell’algoritmo del banchiere per evitare lo stallo (Saranno date le condizioni iniziali)

Illustrare le quattro condizioni che causano lo stallo. Illustrare il funzionamento dell’algoritmo di rilevazione dello stallo (Saranno date le condizioni iniziali)

Esercizio su algoritmo per il rimpiazzamento delle pagine (vedi anche esercizio 8.2 stallings)

Illustrare le principali politiche per la gestione del resident set.

Esercizio su scheduling dei processi (un algoritmo rispetto ad una sequenza di “job”)

### **Esercizio 4**

Esercizio su scheduling del disco (un algoritmo rispetto ad una sequenza di richieste)

Illustrare l’architettura RAID soffermandosi sulle peculiarità dei 7 livelli proposti e i principali campi applicativi di ciascuna architettura.

Illustrare l’architettura del software per la gestione del file system.

Illustrare le principali tipologie di organizzazione dei file

Illustrare le principali strategie utilizzate per allocare i record ai blocchi.

Illustrare le principali strategie per la gestione dei dispositivi di memoria secondaria da parte del gestore del file system.

### **Esercizio 5 (6 punti)**

Illustrare i quattro requisiti che devono essere considerati quando si progetta la sicurezza di un sistema operativo

Illustrare le principali quattro categorie di minacce alla sicurezza cui è esposto tipicamente un sistema operativo

Illustrare i concetti di “access control list” e “capability tickets”

Illustrare le principali differenze fra “anomaly intrusion detection” e “rule-based intrusion detection”

Illustrare le principali tipologie di software “cattivo”

Illustrare il modello di Bell-La Padula per la progettazione di un “trusted operating system”