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

**Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 26 aprile 2005**

# Compito A

Libri e appunti chiusi. Vietato comunicare con chiunque. Vietato l'uso di cellulari, calcolatrici, palmari e affini. Tempo a disposizione: 60 minuti.

## Esercizio 1

1. Descrivi l'architettura di un sistema operativo "execution within user process" (come in UNIX).

In tale architettura nell'immagine di ciascun processo vi è una parte condivisa da tutti i processi contenete codice, stack e dati del kernel. System call, e interrupt handlers vengono eseguiti nell'immagine del processo running. Solo il process switching viene considerato fuori da qualsiasi processo.

(vedi anche diagramma su libro)

2. Per l'architettura al punto 1, descrivi il concetto di process switch e in che relazione è con il concetto di mode switch.

Il process switch è quella operazione che configura la CPU (in particolare l'immagine accessibile) in modo che si possa eseguire un altro processo. Il processo P precedentemente running viene messo in blocco o in ready e un nuovo processo ready (scelto dallo scheduler) viene messo in running. Nel PCB di P viene salvato tutto ciò che è necessario perché P possa essere ripristinato.

Il process switch può avvenire solo in kernel mode, quindi, per ciascun process switch si hanno due mode switch. Il primo mode switch può essere stimolato da un interrupt (es. time slicing, il processo viene messo in ready) o da una system call bloccante (il processo viene messo in blocco), il secondo è l'ultima operazione del dispatcher.

## Esercizio 2

Descrivi brevemente l'algoritmo di I/O scheduling SCAN. Dai un esempio di starvation o di unfairness.

SCAN muove la testina in una sola direzione soddisfacendo tutte le richieste che fanno riferimento alle tracce su cui la testina passa. All'ultima traccia si cambia direzione.
SCAN provoca starvation solo nel caso molto particolare di lettura sempre sulla stessa traccia.
SCAN favorisce lunghe sequenze di richieste a tracce consecutive che la testina sta per leggere.

## Esercizio 3

Un processo effettua i seguenti riferimenti (in sola lettura) alle sue pagine di memoria: 6 5 4 3 6 4 3 1. Il sistema operativo ha asseganto a tale processo 3 frames gestiti con una politica CLOCK pura. Compila la seguente tabella con lo stato dei frame dopo ciascun riferimento a memoria. Indica con * i frame che hanno bit di uso a 1 e con → la posizione della lancetta.

| | stato | | stato | | stato | | stato | | stato | | stato | | stato | | stato |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | *6 | 5 | *6 | 4 | →*6 | 3 | *3 | 6 | *3 | 4 | *3 | 3 | *3 | 1 | →3 |
| | → | | *5 | | *5 | | →5 | | *6 | | *6 | | *6 | | 6 |
| | | | → | | *4 | | 4 | | →4 | | →*4 | | →*4 | | *1 |

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

**Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 26 aprile 2005**

# Esercizio 4

1. Mostra lo schema di traduzione di indirizzo virtuale in indirizzo fisico per una architettura di memoria virtuale con paginazione e segmentazione. Mostra anche la struttura della tabella delle pagine e dei segmenti.

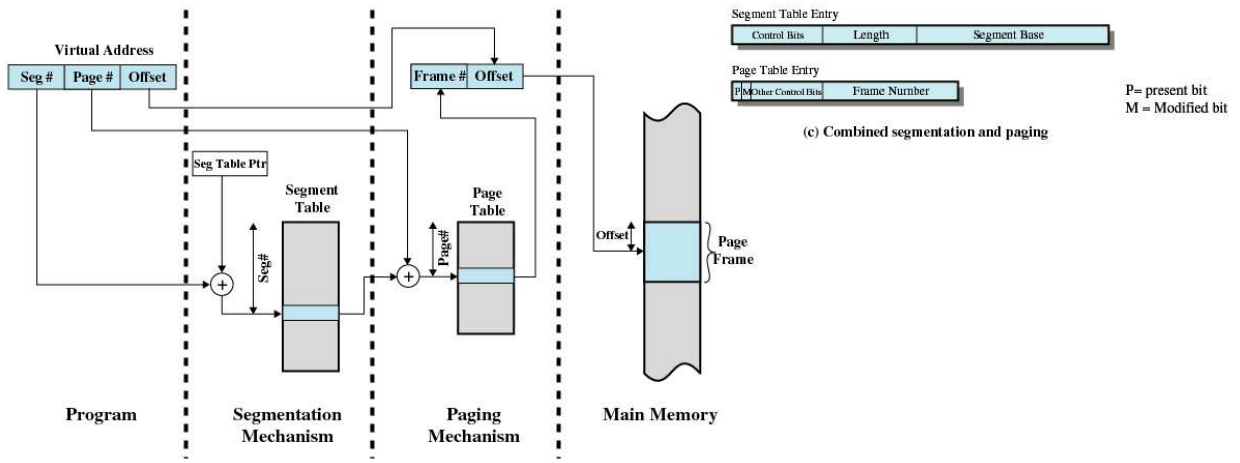


Figure 8.13 Address Translation in a Segmentation/Paging System

2. Considera l'architettura al punto 1 con un indirizzo di memoria virtuale di 32 bit e pagina di 4Kbytes. L'istruzione macchina "copy from 0x00001ffe to 0x00002ffe" occupa 9 bytes a partire dall'inidirizzo 0x00000ffe e copia una parola di 4 bytes. Dati e istruzioni sono tutti nello stesso segmento e tutte le tabelle sono residenti. Quanti page faults può generare al più l'esecuzione di questa sola istruzione? Spiega.

Fetch, indirizzi 0x00000ffe-0x00001006, pagine 0x00000 e 0x00001, 2 page fault.
Lettura, indirizzi 0x00001ffe-0x00002001, pagine 0x00001 e 0x00002, 0x00001 è sicuramente già in memoria, 1 page fault su 0x00002.
Scrittura, indirizzi 0x00002ffe-0x00003001, pagine 0x00002 e 0x00003, la pagina 0x00002 è sicuramente già in memoria, 1 page fault su 0x00003.
Al più possono essere generati 4 page fault

# Esercizio 5

Considera una politica di scheduling feedback con 10 livelli di priorità $p$ (0=alta priorità, 9=bassa priorità) in cui si pone $p=p+1$ ogni volta che scade il quanto $q$ del processo (eventualmente saturando a 9). Ogni volta che il processo va in blocco ritorna in una delle code ready con $p=p-2$ (eventualmente saturando a 0). Il rilascio anticipato della cpu dovuto ad una system call bloccante non dà luogo ad alcun credito di quanto.

1. Supponi che un processo non vada mai in blocco, quale sarà l'evoluzione del valore di $p$ per tale processo?

0 1 2 3 4 5 6 7 8 9 9 9 9 9 9 9....

2. Supponi che un processo vada in blocco ogni 6,5$q$, quale sarà l'evoluzione del valore di $p$ per tale processo?

0 1 2 3 4 5 6 (blocco) 4 5 6 7 8 9 9 (blocco) 7 8 9 9 9 9 9 (blocco) 7 8 9 9 9 9 9 (blocco) ....

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

**Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 26 aprile 2005**

# Compito B

Libri e appunti chiusi. Vietato comunicare con chiunque. Vietato l'uso di cellulari, calcolatrici, palmari e affini. Tempo a disposizione: 60 minuti.

## Esercizio 1

1. Mostra il diagramma di transizione di stato di un processo per il modello a 5 stati (3+new+exit) e descrivi ciascuno dei 3 stati principali in non più di una riga.

(vedi diagramma su libro)

running: il processo è eseguito dalla CPU
ready: processi che possono essere eseguiti dalla CPU ma non sono al momento eseguiti
blocked: processi che sono in attesa di un evento.

2. Per ciascuna possibile transizione nel modello descritto al punto 1, fornisci un esempio di evento che provoca la transizione.

running → ready: quanto di tempo scaduto
ready → running: lo scheduler ha scelto il processo che va in running
running → blocked: il processo ha eseguito una system call bloccante
blocked → ready: l'evento per cui il processo era in attesa è accaduto
(new → ready: il processo è stato lanciato)
(running → exit: il processo ha terminato l'esecuzione)

## Esercizio 2

Descrivi brevemente l'algoritmo di I/O scheduling C-SCAN. Dai un esempio di starvation o di unfairness.

C-SCAN muove la testina in una sola direzione soddisfacendo tutte le richieste che fanno riferimento alle tracce su cui la testina passa. Arrivati all'ultima traccia si ritorna alla prima (ignorando le richieste pendenti) e si ricomincia.
C-SCAN provoca starvation solo nel caso molto particolare di lettura sempre sulla stessa traccia.
C-SCAN favorisce lunghe sequenze di richieste a tracce consecutive che la testina sta per leggere.

## Esercizio 3

Un processo effettua i seguenti riferimenti (in sola lettura) alle sue pagine di memoria: 1 2 3 4 2 3 5 6. Il sistema operativo ha asseganto a tale processo 3 frames gestiti con una politica CLOCK pura. Compila la seguente tabella con lo stato dei frame dopo ciascun riferimento a memoria. Indica con * i frame che hanno bit di uso a 1 e con → la posizione della lancetta.

| | stato | | stato | | stato | | stato | | stato | | stato | | stato | | stato |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | *1 | 2 | *1 | 3 | →*1 | 4 | *4 | 2 | *4 | 3 | *4 | 5 | 4 | 6 | →4 |
| | → | | *2 | | *2 | | →2 | | →*2 | | →*2 | | *5 | | *5 |
| | | | → | | *3 | | 3 | | 3 | | *3 | | →3 | | *6 |

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

**Sistemi Operativi 1 — A.A. 2004-2005, prova scritta del 26 aprile 2005**

# Esercizio 4

1. Mostra lo schema di traduzione di indirizzo virtuale in indirizzo fisico per una architettura di memoria virtuale con paginazione a due livelli. Mostra anche la struttura delle tabella delle pagine.

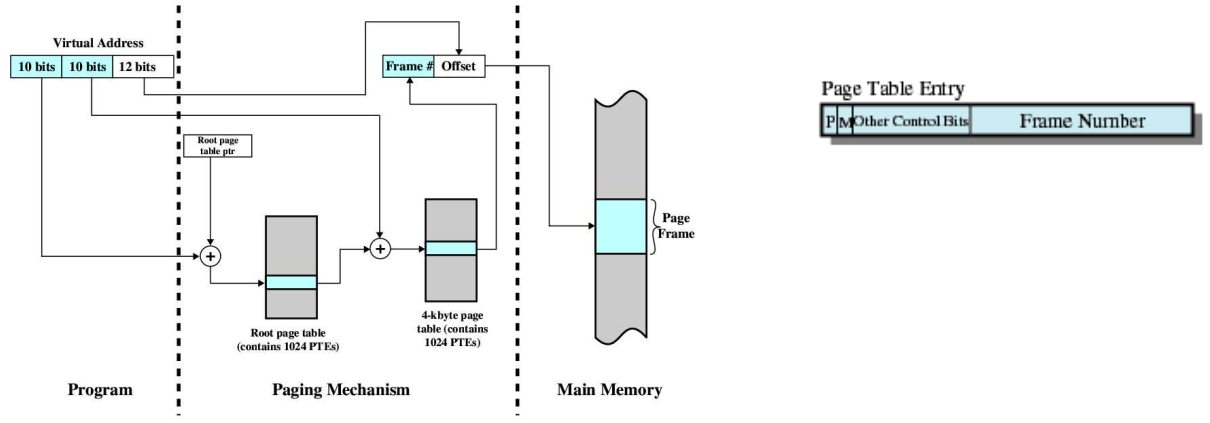


Figure 8.5 Address Translation in a Two-Level Paging System

2. Considera l'architettura al punto 1 con un indirizzo di memoria virtuale di 32 bit e pagina di 4Kbytes. Assumi che tutte le tabelle delle pagine siano sempre residenti. L'istruzione macchina "call 0x00050000" occupa 5 bytes a partire dall'inidirizzo 0x00000ffe ed è una chiamata di procedura. Lo stack pointer contiene il valore 0x80001001 che punta alla prima locazione libera dello stack. Lo stack cresce verso indirizzi di memoria bassi (cioè lo stack pointer viene decrementato a seguito di una operazione di push). Quanti page faults può generare l'esecuzione di tale istruzione? Spiega.

Fetch, indirizzi 0x00000ffe-0x00001002, pagine 0x00000 e 0x00001, 2 page fault.
Scrittura sullo stack di un indirizzo di 4 bytes, indirizzi 0x80000ffe-0x80001001, pagine 0x80000 e 0x80001, 2 page fault.
Nota che non si fa alcun accesso a 0x00050000, tale valore viene solo caricato nel program counter.
Al più possono essere generati 4 page fault

# Esercizio 5

Considera una politica di scheduling feedback con 7 livelli di priorità $p$ (0=alta priorità, 6=bassa priorità) in cui si pone $p=p+2$ ogni volta che scade il quanto $q$ del processo (eventualmente saturando a 6). Ogni volta che il processo va in blocco ritorna in una delle code ready con $p=p-5$ (eventualmente saturando a 0). Il rilascio anticipato della cpu dovuto ad una system call bloccante non dà luogo ad alcun credito di quanto.

1. Supponi che un processo vada sempre in blocco dopo 1,5$q$, quale sarà l'evoluzione del valore di $p$ per tale processo?

0 2 (blocco) 0 2 (blocco).....

2. Supponi che un processo vada in blocco ogni 6,5$q$, quale sarà l'evoluzione del valore di $p$ per tale processo?

0 2 4 6 6 6 6 (blocco) 1 3 5 6 6 6 6 (blocco) 1 3 5 6 6 6 6 (blocco) ....