Sistemi Operativi
Modulo 1

Alessandro Francucci

31 maggio 2016

Introduzione

In questo pdf, troverete diversi quesiti inerenti al modulo uno di sistemi operativi, distinti in 3 livelli.
Ovviamente, queste domande non saranno tutte quelle possibili all'esame (anche perchè si aggiornano di anno in anno), ma semplicemente quelle che sono riuscito a "racimolare" a mio tempo. Spero che vi siano di aiuto!
NB: Le domande sono particolarmente inerenti all'anno accademico: 2014/2015.

Indice

1 Livello 1 **3**

2 Livello 2 **11**
2.1 Esercizi a risposta multipla 11
2.1.1 Traduzione Indirizzi 16
2.2 Esercizi a risposta semi-aperta 17
2.2.1 DMA 17
2.2.2 Tabella pagine 17
2.2.3 Operazioni I/O 18

3 Livello 3 **19**
3.1 Concorrenza-Risposta multipla 19
3.2 Concorrenza-Codice 21
3.2.1 Semafori 21
3.2.2 Busy wait 22
3.2.3 Lettore-Scrittore 24
3.2.4 Produttore Consumatore 25
3.2.5 Test And Set 26
3.2.6 Doppi Consumatori 27
3.2.7 Doppio Buffer 29
3.2.8 Barrier 30

4 Conclusione **31**

Capitolo 1

Livello 1

Il livello uno tratta semplicemente domande a scelta multipla.(True o False) È il livello più semplice, e richiede un minimo del 50 % di risposte esatte per essere superato.
Consigli: Avere ben chiari i vari tipi di scheduler e il significato di preemption (ovvero di preelazione, che significa che il processo può essere momentaneamente interrotto, per dar spazio ad altri processi.)

Il valore, in termini di 30esimi di questo livello è pari a: 8/30.

Domanda: "Un sistema operativo è un software applicativo che permette ad un utente di gestire il proprio computer."
Risposta: F;

Domanda: "I processi assegnati al processore secondo l'algoritmo FCFS sono soggetti a starvation."
Risposta: F;

Domanda: "Uno scheduler Shortest Process Next può causare Starvation"
Risposta: T;

Domanda: "I processi assegnati al processore secondo l'algoritmo SPN sono soggetti a starvation"
Risposta: T;

Domanda: "I processi assegnati al processore secondo l'algoritmo Round Robin sono soggetti a starvation."
Risposta: F;

Domanda: "Lo scheduling SRT è di tipo preemptive."
Risposta: T;[1]

Domanda: "Lo schduling Round Robin è di tipo non-preemptive."
Risposta: F;

Domanda: "Lo scheduling Highest response Ratio Next (HRRN) è di tipo preemptive."
Risposta: F;

Domanda: "In un sistema multiutente, gli utenti possono operare contemporaneamente sullo stesso calcolatore."
Risposta: T;

Domanda: "In un sistema multiutente, gli utenti condividono sistemi operativi diversi."
Risposta: F;

Domanda: "La politica di disk scheduling chiamata C-Scan riordina le richieste di I/O su disco in arrivo dai processi sulla base delle priorità dei processi richiedenti."
Risposta: F;

Domanda: "La politica di disk scheduling chiamata C-Scan riordina le richieste di I/O su disco in arrivo dai processi in modo da ridurre il numero medio di errori di parità per unità di tempo."
Risposta: F;

Domanda: "La politica di disk scheduling chiamata C-Scan riordina le richieste di I/O su disco in arrivo dai processi permettendo al braccio del disco di muoversi solo in una direzione, quando l'ultima traccia in una direzione è stata visitata, il braccio viene riportato all'estremo opposto e la scansione ricomincia da capo."

[1] Non confondere con "shortest process next"

Risposta: F;

Domanda: "Il kernel di un sistema operativo è costituito dall'insieme delle sole precedure per la getione dei dispositivi di I/O."
Risposta: F;

Domanda: "Il file system gestisce i driver dei dispositivi."
Risposta: F;

Domanda: "Il kernel del sistema operativo è responsabile della gestione del processore."
Risposta: T;

Domanda: "In un elaboratore, il DMA è lo stato a cui passa un processo che inizia un operazione di I/O."
Risposta: F;

Domanda: "In un elaboratore, il DMA è l'attributo dei segmenti di memoria che possono essere acceduti direttamente da ogni processo."
Risposta: F;

Domanda: "In un elaboratore, il DMA è una tecnica che demanda l'esecuzione di un operazione di I/O ad un apposito modulo software."
Risposta: F;

Domanda: "In un elaboratore, il DMA è una tecnica che demanda l'esecuzione di un operazione di I/O ad un apposito dispositivo Hardware."
Risposta: T;

Domanda: "Lo scheduling Round Robin è di tipo preemtive."
Risposta: T;

Domanda: "Nella gestione della memoria, il sistema a partizionamento fisso soffre di frammentazione ESTERNA."

Risposta: F;

Domanda: "Se si usano istruzioni macchina (atomiche), come compare&swap e exchange, per gestire la mutua esclusione, allora è possibile avere processi in uno stato di stallo (deadlock)"
Risposta: T;

Domanda: "Se si usano istruzioni macchina (atomiche), come compare&swap e exchange, per gestire la mutua esclusione, allora è possibile incorrere nella starvation"
Risposta: T;

Domanda: "Il problema dei lettori/scrittori si risolve utilizzando i semafori."
Risposta: T;

Domanda: "Il descrittore di processo (process control block) include al suo interno le variabili condivise che vengono accedute nelle sezioni critiche."
Risposta: F;

Domanda: "Il descrittore di processo (process control block) include al suo interno le aree dati, codice e stack del processo."
Risposta: F;

Domanda: "Il descrittore di processo (process control block) contiene al suo interno i puntatori alla page table che descrive la memoria virtuale assegnata al processo."
Risposta: T;

Domanda: "Il descrittore di processo (process control block) include al suo interno l'identificatore del processo, ma non quello del processo padre."
Risposta: F;

Domanda: "Il descrittore di processo (process control block) contiene al suo interno la sola area stack del processo,utilizzando puntatori alle page table per tener traccia della memoria assegnata al processo."

Risposta: F;

Domanda: "In un sistema multiutente, gli utenti condividono solo le risorse in rete."
Risposta: F;

Domanda: "In un sistema operativo Linux, qualunque file eseguibile è un processo."
Risposta: F;

Domanda: "Il meccanismo dei semafori non supporta forma di sincronizzazione tra processi."
Risposta: F;

Domanda: "Il problema dei lettori/scrittori si può risolvere utilizzando le primitive di mutua esclusione."
Risposta: T;

Domanda: "In un sistema operativo Linux, la tripletta R - - in notazione simbolica corrisponde al valore ottale 4?"
Risposta: T;[2]

Domanda: "In un sistema operativo Linux, la tripletta RW- in notazione simbolica corrisponde al valore ottale 6?"
Risposta: T;

Domanda: "Il *trashing* nel contesto della memoria virtuale, si riferisce al fatto che il sistema passa più tempo a fare swapping di pagine che ad eseguire istruzioni."
Risposta: T;

[2] NB: il valore delle triplette si calcola esprimendo una terna in binario dove:

- L'1 indica la corrispondente presenza della lettera.
- Lo 0 indica la corrispondente assenza della lettera.

Nell'esempio, essendo presente solo R - -, questo vuol dire che in binario il numero lo rappresentiamo come 100, che equivale in ottale (che è uguale al decimale per ogni possibile esempio in questa situazione, dove il massimo è 7 quando abbiamo 111) al numero:4

Domanda: "Dati due processi concorrenti Pi e Pj, con i meccanismi di sincronizzazione, i due processi non possono imporre il vincolo che l'azione A di Pi preceda l'azione B di Pj, o viceversa"
Risposta: F;

Domanda: "Dati due processi Pi e Pj la velocità relativa di esecuzione di Pi e Pj dipende dagli identificatori di processo assegnati dal sistema operativo."
Risposta: F;

Domanda: "Dati due processi concorrenti Pi e Pj, se A e B sono due comandi eseguiti rispettivamente da Pi e Pj, è sempre stabilita una relazione di precedenza fra A e B."
Risposta: F;

Domanda: "Nella gestione della memoria, il fenomeno della frammentazione esterna riduce la quantità di memoria utilizzabile."
Risposta: T;

Domanda: "Nella gestione della memoria, il fenomeno della frammentazione esterna richiede il vincolo della contiguità dello spazio fisico di un processo in memoria centrale."
Risposta: F;

Domanda: "Il meccanismo di traduzione degli indirizzi nella gestione della memoria virtuale con paginazione richiede un accesso a disco per ogni accesso ad una locazione di memoria principale non utilizzata da più di 1024 cicli di clock"
Risposta: F;

Domanda: "Il meccanismo di traduzione degli indirizzi nella gestione della memoria virtuale con paginazione utilizza la tabella delle pagine che riporta le corrispondenze tra indirizzi astratti e inidirizzi fisici di inzio delle pagine."
Risposta: F;

Domanda: "Uno scheduler Shortest Process Next utilizza la preemption."

Risposta: F;

Domanda: "Uno scheduler Shortest Process Next sostituisce il processo in esecuzione qualora sia sottomesso un processo di durata inferiore."
Risposta: F;

Domanda: "La tecnica più efficente per gestire la mutua esclusione è quella di disabilitare gli interrupt. In particolare, questa tecnica viene applicata nei sistemi con più processori."
Risposta: F;

Domanda: "Grazie alla memoria virtuale, un processo può essere più grande di tutta la memoria principale del sistema."
Risposta: T;

Domanda: "Nella memoria virtuale, l'intero processo viene caricato in memoria per rendere l'esecuzione più rapida"
Risposta: F;

Domanda: "Nella memoria virtuale, la tavola delle pagine è contenuta nei registri. In questo modo, il processore può accedere rapidamente alle pagine del processo in esecuzione."
Risposta: F;

Domanda: "Nella memoria virtuale, se il processore non trova un indirizzo logico nella memoria principale, salta l'istruzione e passa alla sucessiva dello stesso processo."
Risposta: F;

Domanda: "Il meccanismo di traduzione degli indirizzi nella gestione della memoria virtuale con la paginazione utilizza la tabella delle pagine che riporta la corrispondenza tra indirizzi astratti e indirizzi fisici di inizio delle pagine."
Risposta: F;

Domanda: "Il meccanismo di traduzione degli indirizzi nella gestione della

memoria virtuale con la paginazione utilizza la tabella delle pagine che riporta la corrispondenza tra indirizzi virtuali e indirizzi astratti di inizio delle pagine."

Risposta: F;

Domanda: "La politica di sostituzione delle pagine LRU (least recently used) sfrutta il principio di località."

Risposta: T;

Capitolo 2

Livello 2

Il livello due, a differenza del primo, tratta domande a risposta semi-aperta, oltre a quelle a risposta multipla. (Che sono pressoché sullo stesso stile del livello 1)[1]
Occorre infatti, per alcuni esercizi di questo livello, capire la metodologia da applicare. (cosa per nulla difficile e alquanto meccanica.)

2.1 Esercizi a risposta multipla

Domanda:"IL translation lookaside buffer (TLB) usa una tecnica chiamata associative mapping per cercare un indirizzo virtuale tra tanti in tempo costante"
Risposta: T;

Domanda:"La gestione della memoria basata sulla segmentazione elimina il problema della frammentazione interna ed esterna."
Risposta: F;

Domanda:"Nella gestione della memoria, il sistema che combina segmentazione e paginazione soffre di frammentazione esterna."
Risposta: F;

[1]Talvolta sono proprio le stesse! E quelle del livello 2 potrebbero essere a loro volta nel livello 1.

Domanda: "Nella gestione della memoria, il sistema a segmentazione soffre di frammentazione esterna."
Risposta: T;

Domanda: "Il prerilascio del processore dovuto alla riattivazione di un processo con priorità superiore a quella del processo in esecuzione fa passare il processo in esecuzione nello stato di pronto."
Risposta: T;

Domanda: "Il problema del lettore/scrittore è una caso particolare del problema dei produttori/consumatori dove si considera un solo scrittore (il produttore) ed un solo lettore (il consumatore)."
Risposta: F;

Domanda: "La gestione della memoria basata sulla segmentazione elimina il problema della frammentazione interna ma non quella esterna."
Risposta: T;

Domanda: "Le transizioni dei processi dallo stato di pronto allo stato di esecuzione competono allo scheduler, che designa di volta in volta il processo in esecuzione selezionandolo tra i processi pronti."
Risposta: T;

Domanda: "Le transizioni dei processi dallo stato di attesa allo stato di pronto competono allo scheduler, che designa di volta in volta il processo in esecuzione selezionandolo tra i processi pronti."
Risposta: F;

Domanda: "La gestione della memoria basata sulla segmentazione elimina il problema della frammentazione interna ed esterna."
Risposta: F;

Domanda: "In un sistema operativo Linux, le chiamate di sistema della famiglia exec() (execl(),execv(), etc..) causano la sostituzione del processo in corso con uno nuovo assegnandogli un nuovo process ID."
Risposta: F;

Domanda: "In un sistema operativo linux, le chiamate di sistema della famiglia exec() (execl(),execv(), etc..) riportano come risultato il valore restituito della funzione main dell'eseguibile specificato. "

Risposta: F;

Domanda: "In un sistema operativo linux, le chiamate di sistema della famiglia exec() (execl(),execv(), etc..) sono i principali meccanismi di creazione di nuovi processi. "

Risposta: F;

Domanda: "Nella gestione della memoria il sistema a partizionamento dinamico soffre di frammentazione interna."

Risposta: F;

Domanda: "In un modello di processo a 7 stati le transizioni dei processi dallo stato ready allo stato running competono allo scheduler, che designa di volta in volta il processo running selezionandolo tra i processi ready."

Risposta: T;

Domanda: "In un modello di processo a 7 stati le transizioni dei processi dallo stato ready allo stato running competono allo scheduler, che designa di volta in volta il processo ready selezionandolo tra i processi blocked."

Risposta: F;

Domanda: "In un modello di processo a 7 stati le transizioni dei processi dallo stato blocked allo stato blocked-suspend competono all'algoritmo di replacement, che designa di volta in volta il processo al quale serve meno memoria."

Risposta: F;

Domanda: "In un modello di processo a 7 stati il prerilascio del processore dovuto alla riattivazione di un processo con priorità superiore a quella del processo running fa passare il processo running allo stato di ready."

Risposta: T;

Domanda: "Una situazione di deadlock (o stallo) si può prevenire utilizzando le comunicazioni per realizzare la mutua esclusione fra i processi."
Risposta: F;

Domanda: "Nella gestione della memoria, il fenomeno della frammentazione esterna, in caso di paginazione, è tanto meno rilevante quanto più la lunghezza media dei programmi è grande rispetto alla dimensione della pagina."
Risposta: F;

Domanda: "Il meccanismo di traduzione degli indirizzi nella gestione della memoria virtuale con paginazione è legato alla nozione di processo, essendo lo spazio degli indirizzi virtuali unico per ogni processo."
Risposta: T;

Domanda: "Uno scheduler Highest Response Ratio Next al momento dello scheduling calcola per ogni processo il rapporto R = (w + s)/s, dove w è il tempo speso in attesa del processore ed s è il tempo stimato per l'esecuzione dell'intero processo."
Risposta: T;

Domanda: "Uno scheduler Highest Response Ratio Next forza la preemption del processo in esecuzione all'arrivo nel sistema di un processo con rapporto R = (w + s)/s maggiore."
Risposta: F;

Domanda: "Un computer esegue 125 istruzioni ogni microsecondo. Dunque, per eseguire una singola istruzione impiega 8 nanosecondi"
Risposta: T[2];

Domanda: "Un computer esegue 125 istruzioni ogni microsecondo. Dunque, per eseguire una singola istruzione impiega 10 nanosecondi."
Risposta: F;

[2] *Come si calcola?* Banalissima proporzione. In generale, per sapere quante istruzioni si fanno al nanosecondo, dividi per 1000 quelle effettuate in un microsecondo.

Domanda: "Un sistema con gestione paginata semplice della memoria ha indirizzi di 32 bit e pagine da 16Kbyte. Il numero massimo di elementi contenuti nella tabella delle pagine di un processo è 512K elementi."

Risposta: F;

Domanda: "Un sistema con gestione paginata semplice della memoria ha indirizzi di 32 bit e pagine di dimensione 16Kbyte. Il numero massimo di elementi contenuti nella tabella delle pagine di un processo è 2^{14} elementi."

Risposta: F;[3]

Domanda: "L'organizzazione della memoria centrale con partizioni fisse elimina la frammentazione esterna ma mantiene il vincolo della contiguità dello spazio fisico in memoria centrale."

Risposta: T;

Domanda: "Una situazione di deadlock (o stallo) si può escludere a priori utilizzando scheduler di processi di tipo non-preemptive."

Risposta: F;

Domanda: "Uno scheduler Highest Response Ratio Next non utilizza la preemption e aspetta che un processo sia completato oppure si blocchi per mettere in esecuzione il processo pronto con il più grande valore di R = (w + s)/s."

Risposta: T;

[3] La risposta corretta è 2^{18}. Per sapere come calcolare questi valori, vedere l'esercizio di questa tipologia aperto, che si affronta a fine livello. Clicca qui per andare all'opportuno paragrafo.

2.1.1 Traduzione Indirizzi

Domanda:1 Si consideri una memoria di 64 byte suddivisa in 8 pagine da 8 byte ciascuna. La page table di un processo in esecuzione è la seguente:
Parte Logica: 0 1 2 3 4 5 6 7
Parte Fisica: 3 1 7 0 4 6 5 2
La traduzione dell'indirizzo logico 5 corrisponde all'indirizzo fisico 29.
Risposta: T;

Domanda:2 Si consideri una memoria di 64 byte suddivisa in 8 pagine da 8 byte ciascuna. La page table di un processo in esecuzione è la seguente:
Parte Logica: 0 1 2 3 4 5 6 7
Parte Fisica: 3 1 7 0 4 6 5 2
La traduzione dell'indirizzo logico 15 corrisponde all'indirizzo fisico 15.
Risposta: T;

Domanda:3 Si consideri una memoria di 32 byte suddivisa in 8 pagine da 4 byte ciascuna. La page table di un processo in esecuzione è la seguente:
Parte Logica: 0 1 2 3 4 5 6 7
Parte Fisica: 3 1 7 0 4 6 5 2
La traduzione dell'indirizzo logico 5 corrisponde all'indirizzo fisico 29.
Risposta: F;

Come avviene in generale una traduzione da un indirizzo logico ad uno fisico? Il procedimento si può riassumere brevemente come segue:[4]

1. Dalla memoria, bisogna capire il Numero delle pagine (che chiameremo Z), e la dimensione di ognuna (che chiameremo Y).
2. Si deve convertire l'indirizzo logico, nel suo equivalente binario. (esempio: 5 $\rightarrow$ 101) Da questo numero, gli ultimi $log_2 Y$ bit, saranno l'offset della pagina, mentre i primi $log_2 Z$ bit che non intralciano gli ultimi Y (se ci sono), saranno quelli che bisogna matchare nella page table.
3. Trovati l'offset e la pagina virtuale (nella domanda uno sono rispettivamente 5 e 0), l'offset rimarrà uguale, mentre il numero di pagina deve essere matchato grazie alla page table. Nel nostro caso abbiamo 0, quindi la page table ci associa la pagina fisica 3.
4. Sostituendo dunque, (sempre in binario) la pagina logica con la fisica, otteniamo così:

000—101	indirizzo Logico
011—101	indirizzo Fisico

5. Convertiamo adesso in numero binario ottenuto in decimale e rispondiamo alla domanda.
   Nel nostro esempio, abbiamo:

$$11101 = 1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3 + 1 * 2^4 = 1 + 4 + 8 + 16 = 29$$

[4] Eventuali esempi, saranno presi grazie ai dati della domanda 1.

2.2 Esercizi a risposta semi-aperta

Qui sono riportati gli esercizi del secondo livello che non sono a risposta multipla, e che vanno risolti tramite l'ausilio di una calcolatrice. Solitamente questi esercizi sono di 3 tipologie, e sono posti sempre alla fine del livello.

2.2.1 DMA

Domanda:1 Si consideri una unità disco DMA con una velocità di rotazione di 5400 rivoluzioni al minuto (rpm). Ogni traccia del disco contiene 64 Kbytes. Si assuma che una porzione di dimensione 301 Kbytes di un file sia memorizzato sul disco nel modo più compatto possibile (in blocchi vicini). Stimare il tempo totale necessario per il trasferimento di questi 301 Kbytes di dati dal disco in memoria principale. Assumere Trascurabile il seek time ed il rotational delay.
Risposta: 0,05225694 secondi;

Procedimento:
Per risolvere questo quesito bisogna innanzitutto trovare quante rivoluzioni vengono effettuate al secondo. (basta dividere quelle al minuto per 60. chiamiamo questo valore **rs**)
Dopodichè, sia:
L = *Dimensione traccia del disco*, (nell'esempio 64kb) ed Z = *la porzione di memoria da trasferire*; allora il tempo necessario per il trasferimento sarà semplicemente:

$$result = (Z/L)/rs$$

2.2.2 Tabella pagine

Domanda:2 Un sistema con gestione paginata semplice della memoria ha indizi di 43 bit e pagine di dimensione 16K byte. Quale è il numero massimo di elementi contenuti nella Tabella delle pagine di un processo?
Risposta: 536870912 Elementi = 512 Mega Elementi;

Procedimento:
Tutto quello che bisogna fare per risolvere l'esercizio è:

1. prendere $z = log_2|Dimensionepagina|$
2. Sottrarre alla dimensione degli indirizzi in bit (in questo caso 43),il valore z appena ottenuto. Chiamiamo K questo nuovo risultato.
3. Il numero di elementi totali sarà 2^K

2.2.3 Operazioni I/O

Domanda:3 Un calcolatore multiprogrammato esegue 3 processi concorrenti che vengono sottomessi allo stesso istante. Ciascun processo ha le medesime caratteristiche ed è strutturato in fasi di attività di lunghezza 19 millisecondi. Ogni processo utilizza solo 1/4 dei 19 millisecondi in attività di I/O. mentre il resto di ciascuna fase è speso in attività di elaborazione sul processore. Ciascun processo viene eseguito per un totale complessivo di 3 fasi di attività usando un semplice scheduler round robin. Le operazioni di I/O possono essere eseguite in parallelo a quelle del processore. Si calcoli il tempo effettivo di completamento (turnaround time) dell'ultimo processo che termina l'esecuzione.
Risposta:133 millisecondi;

Procedimento:
Quest'ultima tipologia di esercizio, si risolve invece nel seguente modo:

- Si Calcola la fase di attività di un processo senza l'I/O in termini di secondi. Visto nell'esempio:
  N = 19 ms;
  I/O = (1/4)*N;
  result = 19 - (1/4)*19 = 57/4 ms;
- Trovato questo valore, che chiameremo K, per la soluzione dovremmo semplicemente fare:
  result = (K * —Processi Concorrenti— * —numero fasi di attività—) + I/O[5]
  Nell'esempio, siccome I/O era uguale a 19/4, avremmo che:

$$result = 57/4 * 9 + 19/4 = 513/4 + 19/4 = 532/4 = 133ms;$$

E con questo si conclude l'esercizio.

NB: Non ho mostrato più di un esempio per questi esercizi, in quanto mi interessava mostrare semplicemente il procedimento! Hanno praticamente tutti la stessa struttura :)

[5] I/O è la variabile definita al passo precedente.

Capitolo 3

Livello 3

L'ultimo livello è la parte più difficile dell'esame (rispetto alle altre..). Questa parte sarà infatti strutturata da 3 domande, tutte sulla concorrenza. Le prime 2 saranno però a risposta multipla, con tanto di penality nel caso di risposta errata, mentre l'ultima riguarda la stesura di codice. Arriviamoci però gradualmente.

Il valore, in termini di 30esimi di questo livello è pari a: 14/30; di cui 10 solo per il codice!

3.1 Concorrenza-Risposta multipla

I seguenti due processi concorrenti condividono la variabile comune X:

```
Process A
        int Y;
A1:     Y = X*2;
A2:     X = Y;

Process B
        int Z;
B1:     Z = X+1;
B2:     X = Z;
```

La variabile è inizializzata a 5 prima dell'avvio dell'esecuzione dei processi A e B. Le istruzioni (A1,A2,B1 e B2) all'interno di un processo sono atomiche e vengono eseguite sequenzialmente, ma le istruzioni del processo A possono essere eseguite in qualsiasi ordine rispetto alle istruzioni del processo B. Qual'è un possibile valore di X dopo che entrambi i processi terminano l'esecuzione?
Possibili Risposte:

A1 A2 B1 B2: X = 11
B1 A1 A2 B2: X = 6

Un altra variante di questo esercizio è la seguente:
I seguenti 3 processi concorrenti condividono due semafori:

```
semaphore U = 3;
semaphore V = 0;

[process1]
L1:     wait(U)
        type("C")
        signal(V)
        go to L1

[process2]
L2:     wait(V)
        type("A")
        type("B")
        signal(V)
        go to L2

[process3]
L3:     wait(V)
        type("D")
        goto L3
```

All'interno di un processo le istruzioni vengono eseguite sequenzialmente. Le istruzioni del processo A possono essere eseguite, rispetto alle istruzioni del processo B, in qualsiasi ordine che sia consistente con i vincoli imposti dai semafori.
Nel rispondere alla domanda supporre che una volta iniziata l'esecuzione di ogni processo sarà consentito fino a quando tutti i 3 processi sono bloccati in un comando wait(), a questo punto l'esecuzione dei tre processi viene interrotta e i tre processi terminano l'esecuzione.

Possibile domanda: Qual'è il **minimo** numero di caratteri "A" che potrebbero venir stampati durante l'esecuzione dei tre processi?
– Esattamente 0 (zero) caratteri "A" stampati.
Possibile domanda: Qual'è il **minimo** numero di caratteri "C" che potrebbero venir stampati durante l'esecuzione dei tre processi?
– Esattamente 3 caratteri "C" stampati.

3.2 Concorrenza-Codice

Il codice è la parte che vale di più in tutto l'esame. (ben 10 punti).
Di seguito, vedremo in più sottosezioni, frammenti di codice che effettivamente sono stati richiesti all'esame. Prima però, eccovi anche del codice su argomenti molto importanti sulla concorrenza visti a lezione.

3.2.1 Semafori

Per una nostra maggiore chiarezza, vediamo come sono implementati i semafori che useremo.

Semaforo Standard

```
struct semaphore
{
        int count;
        queue coda;
        void semWait(Semaphore s)
        {
                s.cont--;
                if(s.count < 0)
                        // metti il processo in coda e bloccalo
        }

        void semSignal(Semaphore s)
        {
                s.count++;
                if(s.count <= 0)
                        // rimuovi il processo dalla coda, e
                            mettilo
                        // in pronto
        }
}
```

Semaforo Binario

```
struct semaphoreBin
{
        int count; // puo' essere anche boolean.
        queue coda;
        void semWait(Semaphore s)
        {
                if(s.count == 1)
                        s.count = 0;
                else
                        // metti il processo in coda e bloccalo
        }

        void semSignal(Semaphore s)
        {
                if(s.coda.isEmpty())
                        s.count = 1;
                else
                        // rimuovi il processo dalla coda, e
                            mettilo
                        // in pronto
        }
}
```

3.2.2 Busy wait

Di seguito il codice delle 2 semplicissime funzioni più importanti, per avere l'attesa attiva. (più il codice di una loro applicazione.)

Test and Set

```
boolean testAndSet(int i)
{
        if(i == 0) { i=1; return true}
        else return false;
}

int contr;

P (int i)
{
        while(true)
        {
                while(!testAndSet(contr)) {// do nothing}

                /* critical section */
                contr=0;
                /* end of critical section */

                // Resto del programma.
        }
}
```

Exchange

```
void exchange(int i, int k)
{
        int temp = i;
        i = k;
        k = temp;
        return;
}

int contr;

P (int i)
{
        int pass = 1;
        while(true)
        {
                while(pass==1) {exchange(pass,contr);}

                /* critical section */
                exchange(pass,contr);
                /* end of critical section */

                // Resto del programma.
        }
}
```

3.2.3 Lettore-Scrittore

Vediamo anche il codice per questo particolare problema, dando la priorità al lettore.

```
/* inizializzazioni */
semaphore w, ME = 1;
int NR = 0; // number of reader.

Reader() {
        semWait(ME);
        NR ++;
        if (NR==1) then semWait(w);
        signal(ME);
        /* Operazioni in parallelo lettura */
        wait(ME);
        NR--;
        if (NR==0) then semSignal(w);
        signal(ME);
}

writer() {
        semWait(w);
        /* operazioni di scrittura in ME */
        semSignal(w);
}
```

3.2.4 Produttore Consumatore

Riportiamo, prima dei "veri" possibili esercizi, anche il problema del produttore-consumatore visto a lezione, con buffer finito.

```
semaphore n = 0;
semaphore s = 1;
semaphore z = sizeofbuffer;

void producer() {
        while (true) {
                produce();
                semWait(s);
                semWait(z);
                append();
                semSignal(s);
                semSignal(n);
        }
}

void consumer() {
        while (true) {
                semWait(n);
                semWait(s);
                take();
                semSignal(s);
                semSignal(z);
                consume();
        }
}
```

Da qui in poi, verranno proposti alcuni codici che sono stati richiesti all'esame. Provate a svolgerli prima di consultare le soluzioni!

3.2.5 Test And Set

Risolvere Con un programma in pseudo-codice il problema produttore consumatore, con buffer condiviso di lunghezza FINITA N (Un array b[M] adoperato come buffer circolare). Utilizzando per la mutua esclusione l'istruzione test-and-set e l'attesa attiva (busy wait). (NO SEMAFORI!)

```
int i=0;
int cont=0;
queue q;

P()
{
        while(true)
        {
                x = produce();
                while(cont==sizeofbuffer) { /* do nothing */}

                while(!testAndSet(i)) { /* do nothing */ }

                /* Critical section */
                q.append(x);
                cont ++;
                i=0;
                /* End of Critical section */
                ... other operation ...
        }
}

C()
{
        while(true)
        {
                while(cont==0) { /* do nothing */ }

                while(!testAndSet(i)) { /* do nothing */}

                /* Critical section */
                y= q.take();
                cont --;
                i=0;
                /* End of critical section */
                consume(y);
                ... other operation ...
        }
}
```

3.2.6 Doppi Consumatori

Un produttore p produce dati che mette in due code FIFO, qa e qb, alternando fra le 2. Due consumatori CA e CB prelevano un dato alla volta, rispettivamente da queste code, e lo elaborano.
I tempi di ogni singola produzione sono molto irregolari, tanto che un consumatore può trovare la sua coda vuota ed andare a consumare nella coda dell'altro. Scrivere lo pseudo codice corrispondente ai processi di P, CA e CB, utilizzando la sincronizzazione tra semafori.

```
inizializzazioni()
{
        queue qa, qb;
                // inizializzo le code

        Sem vuoti_a = size
        Sem vuoti_b = size
                // servono per vedere se ha senso produrre li
                    sopra.
        Sem dati = 0;
                // serve per vedere se abbiamo prodotto qualcosa.
        Sem mutex_a = 1;
        Sem mutex_b = 1;
                // servono per la mutua esclusione
        int dati_a = 0;
        int dati_b = 0;
                // controlla quanti dati abbiamo prodotto per quel
                     consumatore
}

void P()
{
        while (true)
        {
                v = produce();
                semWait(vuoti_a);
                semWait(mutex_a);
                qa.append(v);
                dati_a++;
                semSignal(mutex_a);
                semSignal(dati);

                w = produce();
                semWait(vuoti_b);
                semWait(mutex_b);
                qb.append(w);
                dati_b++;
                semSignal(mutex_b);
                semSignal(dati);
        }
}
```

```
void CA()
{
        while (true) {
                semWait(dati);
                if (dati_a == 0)
                {
                        // vado a consumare da b
                        wait(mutex_b);
                        qb.take();
                        dati_b--;
                        signal(mutex_b);
                        signal(vuoti_b);
                } else {
                        // vado a consumare dalla mia coda
                        wait(mutex_a);
                        qa.take();
                        dati_a--;
                        signal(mutex_a);
                        signal(vuoti_a);
                } consume();
        }
}

// Per quanto riguarda il secondo consumatore, il codice e'
    analogo
// al precedente, con i vari a e b invertiti opportunamente.
```

3.2.7 Doppio Buffer

Il processo P1 produce dei dati e li invia al processo P2 tramite un sistema a doppio buffer, costituito da due buffer (ciascuno di lunghezza unitaria) condivisi fra P1 e P2. Mentre P2 legge da un buffer, P1 può scrivere nell'altro.

Scrivere i codici dei processi P1 e P2, utilizzando i semafori per la sincronizzazione.

```
Sem w1 = 1
Sem w2 = 1
Sem r1 = 0
Sem r2 = 0
dati x, y; /* inizializzazioni dati letti */

void P1()
{
        while(true)
        {
                k = produce();
                semWait(w1);
                append(buffer1, k)
                semSignal(r1)
                k = produce()
                semWait(w2)
                append(buffer2, k)
                semSignal(r2)
        }
}


void P2()
{
        while(true)
        {
                semWait(r1)
                y = take(buffer1)
                semSignal(w1)
                consume(y)
                semWait(r2)
                y = take(buffer2)
                semSignal(w2)
                consume(y)
        }
}
```

3.2.8 Barrier

Talvolta le applicazioni concorrenti sono divise in fasi con la regola che nessun processo può proseguire se prima tutti i suoi simili non sono pronti a farlo. le barriere implementano questo concetto: un processo che ha terminato la sua fase chiama una primitiva *barrier()* e si blocca. Quando tutti i processi coinvolti hanno terminato il loro stadio di esecuzione invocando anch'essi la primitiva *barrier()*, il sistema li sblocca tutti permettendo di passare ad uno stadio successivo. Un'applicazione concorrente composta da N processi utlizza più volte durante la sua esecuzione il meccanismo della *barrier()*. Dovendo migrare l'applicazione in questione ad un sistema operativo che non fornisce tale meccanismo, ma che fornisce condivisione di memoria e semafori, siete costretti a scrivere voi stessi la primitiva *barrier()* in termini dei meccanismi disponibili.

```
/* inizializzazioni */
shared int allabarriera = 0;
sem mutexab = 1;
sem sb = 0;

barriera() {
        int me, i;

        wait(mutexab);
        me = ++allabarriera;
        signal(mutexab);
        if (me==N) {
                allabarriera = 0; /* per preparare alle barriere
                    successive */
                for(i=1; i<N; ++i)
                        signal(sb);
   } else
                wait(sb);
}
```

Capitolo 4

Conclusione

Spero che il pdf vi sia servito e sia stato di vostro gradimento!

Per la segnalazione di errori (o comunicazione), scrivetemi pure alla mia email: alessandrofrancucci@hotmail.it, o, se preferite, sul mio profilo Facebook. :)

Se volete invece consultare altri miei lavori/appunti scritti in LaTeX, vi basta visitare la mia pagina WordPress.

Concludendo, se questo pdf vi è stato di aiuto, potete esprimermi la vostra gratitudine tramite una libera donazione con **PayPal Me**! Ve ne sari molto riconoscente! Anche il semplice gesto di donarmi quanto basta per *un caffè*, mi farebbe capire che tutto il lavoro fatto è stato in qualche modo apprezzato. :')

Detto questo, *Buona fortuna a tutti per l'esame!*