



## Programmazione di sistema

### Esame 20 Giugno 2022 - OS Internals



**Iniziato** lunedì, 20 giugno 2022,

**Terminato** lunedì, 20 giugno 2022,

**Tempo impiegato**

**Valutazione**

*Per CBM, la valutazione soprastante è relativa alla valutazione massima per risposte tutte corrette con  $C=1$ . ?*

#### Risultati per l'intero quiz (0 domande)

**Punteggio CBM medio** 0,00

**Accuratezza** 0,0%

**CBM bonus** 0,0%

**Accuratezza + Bonus** 0,0%

#### Ripartizione per confidenza

**C=3** Nessuna risposta

**C=2** Nessuna risposta

**C=1** Nessuna risposta

#### Domanda 1

Completo

Punteggio ottenuto ,00 su 3,00

**TUTTE LE RISPOSTE SÌ / NO DEVONO ESSERE MOTIVATE. QUANDO I RISULTATI SONO NUMERI, SONO NECESSARI IL RISULTATO FINALE E I RELATIVI PASSI INTERMEDI (O FORMULE)**

Considerare un file system Unix-like, basato su inode, con 13 puntatori / indici (10 diretti, 1 singolo indiretto, 1 doppio indiretto e 1 triplo indiretto). I puntatori / indici hanno una dimensione di 32 bit e i blocchi del disco hanno una dimensione di 2 KB. Il file system risiede su una partizione del disco di 800 GB, che include sia blocchi di dato che blocchi di indice.

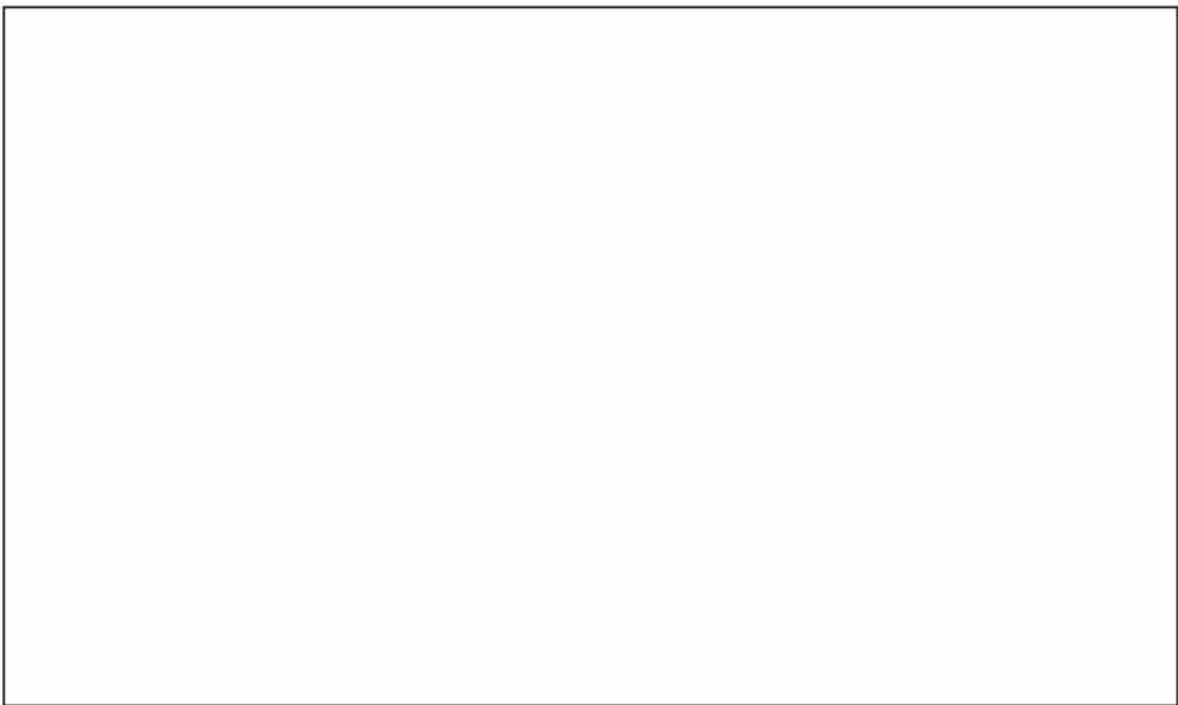
A) Supponendo che tutti i metadati (eccetto i blocchi indice) abbiano dimensioni trascurabili, calcolare il numero massimo di file che il file system può ospitare, utilizzando l'indicizzazione indiretta doppia (N2) e l'indicizzazione indiretta tripla (N3).

B) Dato un file binario di dimensione 10240.5KB, calcolare esattamente quanti blocchi indice e blocchi di dato occupa il file.

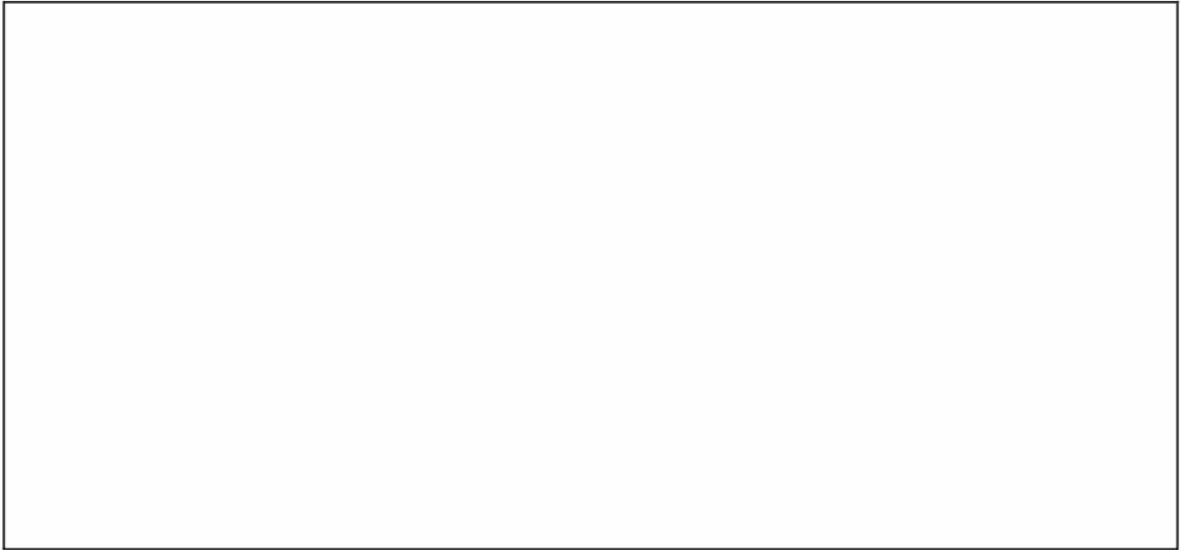
C) Considerare lo stesso file della domanda B, su cui viene chiamata l'operazione lseek (fd, offset, SEEK\_END) per posizionare l'offset del file per la successiva operazione di lettura / scrittura (SEEK\_END significa che l'offset viene calcolato dalla fine del file). Supponiamo che fd sia il descrittore di file associato al file (già aperto), e offset = 0x00800000. Calcolare il numero di blocco logico (numerazione relativa ai blocchi del file, numerati a partire da 0) in corrispondenza del quale viene spostata la posizione (da lseek). Se il file utilizza un'indicizzazione indiretta singola (o doppia / tripla), calcolare quale riga del blocco di indici esterno contiene l'indice del blocco di dati (o l'indice del blocco di indici più interno).

---

A) Supponendo che tutti i metadati (eccetto i blocchi indice) abbiano dimensioni trascurabili, calcolare il numero massimo di file che il file system può ospitare, utilizzando l'indicizzazione indiretta doppia (N2) e l'indicizzazione indiretta tripla (N3).



B) Dato un file binario di dimensione 10240.5KB, calcolare esattamente quanti blocchi indice e blocchi di dato occupa il file.



C) Considerare lo stesso file della domanda B, su cui viene chiamata l'operazione lseek (fd, offset, SEEK\_END) per posizionare l'offset del file per la successiva operazione di lettura / scrittura (SEEK\_END significa che l'offset viene calcolato dalla fine del file). Supponiamo che fd sia il descrittore di file associato al file (già aperto), e offset = 0x00800000. Calcolare il numero di blocco logico (numerazione relativa ai blocchi del file, numerati a partire da 0) in corrispondenza del quale viene spostata la posizione (da lseek). Se il file utilizza un'indicizzazione indiretta singola (o doppia / tripla), calcolare quale riga del blocco di indici esterno contiene l'indice del blocco di dati (o l'indice del blocco di indici più interno).



A) Supponendo che tutti i metadati (eccetto i blocchi indice) abbiano dimensioni trascurabili, calcolare il numero massimo di file che il file system può ospitare, utilizzando l'indicizzazione indiretta doppia (N2) e l'indicizzazione indiretta tripla (N3).

**Osservazioni generali**

Un blocco indice contiene  $2KB / 4B = 512$  puntatori / indici.

La partizione contiene  $800 GB / 2 KB = 400 M$  blocchi

**Calcolo dei numeri massimi  $N_2$  /  $N_3$**  (per file con indicizzazione indiretta doppia / tripla)

Per calcolare il numero massimo di file, dobbiamo considerare l'occupazione minima. Dobbiamo considerare sia i blocchi di dati che i blocchi di indice.

Usiamo  $MIN_2$  e  $MIN_3$  per l'occupazione minima dei due tipi di file:

$$MIN_2 = (10 + 512 + 1) \text{ blocchi di dati} + \\ 1 \text{ (singolo)} + 2 \text{ (doppio)} \text{ blocchi indice} = 526$$

$$MIN_3 = (10 + 512 + 512^2 + 1) \text{ blocchi di dati} + \\ 1 \text{ (singolo)} + 1 + 512 \text{ (doppio)} + 1 + 1 + 1 \text{ (triplo)} \text{ blocchi indice} \\ = 16 + 1024 + 512^2 = 1040 + 512^2$$

$$N_2 = \text{floor}(400M / 526) = 797397 = 0,76 M$$

$$N_3 = \text{floor}(400M / (1040 + 512^2)) = 1594$$

- B) Dato un file binario di dimensione 10240.5KB, calcolare esattamente quanti blocchi indice e blocchi di dato occupa il file.

Blocchi dati:  $\text{ceil}(10240.5KB/2KB) = 5121$

Framm. Int. =  $1 - 0.25 \text{ blocks} = 0.75 \text{ blocks} = (1024+512)B = 1536B$

Blocchi indice

Blocchi indice singolo: 1

Blocchi indice interni:  $\text{ceil}((5121-10-512)/512) = 9$

Blocco indice esterno: 1 (è sufficiente il doppio indiretto)

Blocchi indice totali:  $1+9+1 = 11$

- C) Considerare lo stesso file della domanda B, su cui viene chiamata l'operazione lseek (fd, offset, SEEK\_END) per posizionare l'offset del file per la successiva operazione di lettura / scrittura (SEEK\_END significa che l'offset viene calcolato dalla fine del file). Supponiamo che fd sia il descrittore di file associato al file (già aperto), e offset = 0x00800000. Calcolare il numero di blocco logico (numerazione relativa ai blocchi del file, numerati a partire da 0) in corrispondenza del quale viene spostata la posizione (da lseek). Se il file utilizza un'indicizzazione indiretta singola (o doppia / tripla), calcolare quale riga del blocco di indici esterno contiene l'indice del blocco di dati (o l'indice del blocco di indici più interno).

l'offset viene posto a  $10240.5K - 0x00800000 = 0x00A00200 - 0x00800000 = 0x00200200$   
Indice del blocco dati:  $0x00200200/2K = (2M+515)/2K = 1K = 0x400$   
10 blocchi di dati sono diretti  
512 blocchi di dati sono a singoli indiretti  
Il blocco è al doppio livello indiretto  
Indici esterni =  $(1K-512-10)/512 = 0$

Commento:

## Domanda 2

Completo

Punteggio ottenuto ,00 su 3,00

**SE I RISULTATI SONO NUMERI, RIPORTARE PASSAGGI INTERMEDI RILEVANTI E/O FORMULE USATE**  
**LE RISPOSTE SI/NO VANNO MOTIVATE**

Un processo user vuole effettuare un input da un dispositivo di IO a caratteri (character device) mediante una strategia basata su polling del dispositivo. Per evitare un'attesa troppo lunga (nel loop di polling) da parte del driver del dispositivo, si chiede all'autore del programma eseguito dal processo user, di fare direttamente polling mediante un loop di lettura del registro di stato del dispositivo.

- A) E' possibile effettuare questa operazione? (se si, dire come, se no, dire perché)
- B) Supponendo che il dispositivo sia associato alla tastiera e lo si gestisca in interrupt, ci si aspetta che venga generato un interrupt per ogni carattere ricevuto oppure un interrupt per ogni "invio" (cioè enter, fine-riga)? (motivare)

Dato un altro dispositivo di IO, gestito a blocchi:

- C) E' possibile che una sola scrittura (mediante chiamata a write (/)), tenti di fare output di più di una coppia puntatore/dimensione?
- D) Dato il driver che realizza la scrittura, supponendo che il sistema sia dotato di DMA controller, è possibile che venga eseguita "prima" una operazione di lettura e/o scrittura attivata successivamente a questa, da un altro processo? E (stessa domanda) dallo stesso processo? (motivare)

---

A) E' possibile effettuare questa operazione? (se si, dire come, se no, dire perché)

B) Supponendo che il dispositivo sia associato alla tastiera e lo si gestisca in interrupt, ci si aspetta che venga generato un interrupt per ogni carattere ricevuto oppure un interrupt per ogni "invio" (cioè enter, fine-riga)? (motivare)

Dato un altro dispositivo di IO, gestito a blocchi,

C) E' possibile che una sola scrittura (mediante chiamata a write ()), tenti di fare output di più di una coppia puntatore/dimensione?

D) Dato il driver che realizza la scrittura, supponendo che il sistema sia dotato di DMA controller, è possibile che venga eseguita "prima" una operazione di lettura e/o scrittura attivata successivamente a questa, da un altro processo? E (stessa domanda) dallo stesso processo? (motivare)

A) E' possibile effettuare questa operazione? (se sì, dire come, se no, dire perché)

NO. Perché un dispositivo di IO richiede istruzioni privilegiate, con la CPU in modalità kernel. Un processo user non ha accesso al dispositivo e non ne può leggere il registro di stato, a meno che sia implementata una opportuna system call in grado di fornire tale servizio.

B) Supponendo che il dispositivo sia associato alla tastiera e lo si gestisca in interrupt, ci si aspetta che venga generato un interrupt per ogni carattere ricevuto oppure un interrupt per ogni "invio" (cioè enter, fine-riga)? (motivare)

Ci si attende un interrupt per ogni carattere. La tastiera è un dispositivo a caratteri. Una eventuale bufferizzazione dei caratteri in una riga, tale da poter effettuare correzioni prima dell'invio, va gestita via software (a meno di avere una tastiera speciale con queste caratteristiche)

Dato un altro dispositivo di IO, gestito a blocchi,

C) E' possibile che una sola scrittura (mediante chiamata a write ()), tenti di fare output di più di una coppia puntatore/dimensione?

NO: La write riceve come parametro una sola coppia puntatore/dimensione. Il vettore di coppie puntatore/dimensione viene eventualmente gestito a un livello più basso di IO su dispositivo a blocchi (VOP\_READ/VOP\_WRITE).

D) Dato il driver che realizza la scrittura, supponendo che il sistema sia dotato di DMA controller, è possibile che venga eseguita "prima" una operazione di lettura e/o scrittura attivata successivamente a questa, da un altro processo? E (stessa domanda) dallo stesso processo? (motivare)

SI: E' possibile, in quanto l'ordine delle operazioni viene effettuato a un livello più basso, dalla gestione del dispositivo (scheduler del disco). Questo vale anche per il singolo processo, che potrebbe avere più thread attivi oppure aver attivato la write in modalità asincrona ed effettuare una seconda operazione senza che la prima sia terminata.

Commento:

**Domanda 3**

Completo

Punteggio ottenuto ,00 su 3,00

**SE I RISULTATI SONO NUMERI, RIPORTARE PASSAGGI INTERMEDI RILEVANTI E/O FORMULE USATE**  
**LE RISPOSTE SI/NO VANNO MOTIVATE**

Si consideri in OS161 una possibile implementazione delle system call relative al file system, basate su una tabella di processo e su una di sistema, definite come segue

```
/* system open file table */  
struct openfile {  
    struct vnode *vn;  
    off_t offset;  
    unsigned int countRef;  
};  
/* this is a global variable */  
struct openfile systemFileTable[SYSTEM_OPEN_MAX];  
...  
/* user open file table: this a field of struct proc */  
struct openfile *fileTable[OPEN_MAX];
```

Si risponda alle domande seguenti (motivando le risposte):

- A) Perché systemFileTable è una variabile globale, mentre fileTable è un campo di struct proc ?
- B) Si supponga che, dati due processi user P1 e P2 (con p1 e p2 puntatori alle rispettive struct proc) siano vere le seguenti uguaglianze:

- (p1->fileTable[5] == &systemFileTable[20])
- (p2->fileTable[8] == &systemFileTable[20])

a quale/i file descriptor (fd) corrisponde/corrispondono le parti di tabella visualizzate? 5, 8, 20? (motivare)

Se P1 e P2 fanno solo letture sul/sui file di cui sopra, le letture sono indipendenti oppure c'è interazione tra i processi?

C) Si supponga di aver implementato il supporto per la system call SYS\_lseek. Si riporta la descrizione delle funzione lseek:

*off\_t lseek(int fd, off\_t offset, int whence);*

*lseek() repositions the file offset of the open file description associated with the file*



descriptor *fd* to the argument offset according to the directive whence as follows:

*SEEK\_SET*

*The file offset is set to offset bytes.*

*SEEK\_CUR*

*The file offset is set to its current location plus offset bytes.*

*SEEK\_END*

*The file offset is set to the size of the file plus offset bytes.*

Una chiamata a *lseek* effettuata da P1 (si veda domanda B) influenzerà la successiva read effettuata da P2:

- 1) sempre ?
- 2) mai ?
- 2) solo se il parametro offset è *SEEK\_CUR* ?

(sono possibili selezioni multiple, per ogni selezione occorre motivare la risposta)

---

A) Perché *systemFileTable* è una variabile globale, mentre *fileTable* è un campo di struct proc ?



B) Si supponga che, dati due processi user P1 e P2 (con *p1* e *p2* puntatori alle rispettive struct proc) siano vere le seguenti uguaglianze:

- (*p1*->*fileTable*[5] == &*systemFileTable*[20])
- (*p2*->*fileTable*[8] == &*systemFileTable*[20])

a quale/i file descriptor (*fd*) corrisponde/corrispondono le parti di tabella visualizzate? 5, 8, 20? (motivare)

Se P1 e P2 fanno solo letture sul/sui file di cui sopra, le letture sono indipendenti oppure c'è interazione tra i processi?

C) Una chiamata a `lseek` effettuata da P1 (si veda domanda B) influenzerà la successiva read effettuata da P2:

A) Perché `systemFileTable` è una variabile globale, mentre `fileTable` è un campo di struct `proc` ?

`systemFileTable` è una variabile globale perchè è unica e condivisa da tutti i processi, mentre `fileTable` è specifica per ogni processo.

B) Si supponga che, dati due processi user P1 e P2 (con `p1` e `p2` puntatori alle rispettive struct `proc`) siano vere le seguenti uguaglianze:

- `(p1->fileTable[5] == &systemFileTable[20])`
- `(p2->fileTable[8] == &systemFileTable[20])`

a quale/i file descriptor (`fd`) corrisponde/corrispondono le parti di tabella visualizzate? 5, 8, 20?

(motivare)

Se P1 e P2 fanno solo letture sul/sui file di cui sopra, le letture sono indipendenti oppure c'è interazione tra i processi?

Si deve far riferimento alla (per-process) fileTable. Quindi

5 per P1

8 per P2

Le letture NON sono indipendenti in quanto si condivide la stessa entry nella systemFileTable. Questo significa che P1 e o P2 "continuano" a leggere da dove ha lasciato la precedente lettura (se c'è stata) l'altro processo. Potrebbero esser necessario gestire la sincronizzazione, ma si tratta di un altro problema.

C) Una chiamata a `lseek` effettuata da P1 (si veda domanda B) influenzerà la successiva read effettuata da P2:

1) sempre ? SI perchè se P1 riposiziona l'offset, P2 lo condivide (indipendentemente da quale posizione si prenda come riferimento)

2) mai ? NO (vedi sopra)

2) solo se il parametro offset è `SEEK_CUR`? NO. Perchè anche se il riferimento a inizio/fine del file non dipende da quanto è successo prima (quindi dalla posizione corrente dell'offset) il risultato, il nuovo offset, influenza P2

Commento:

#### Domanda 4

Completo

Punteggio ottenuto ,00 su 3,00

**TUTTE LE RISPOSTE SÌ / NO VANNO MOTIVATE. PER LE RISPOSTE NUMERICHE, SONO RICHIESTI SIA I RISULTATI CHE I PASSAGGI INTERMEDI (O FORMULE) RILEVANTI**

Si considerino tre kernel thread in OS161, i quali implementano un'attività di trasferimento dati basata sul modello produttore/consumatore (due produttori, più consumatori). I thread condividono una struttura C di tipo struct prodCons definita come segue:

```
#define NumP 2
```

```

struct prodCons {
    void *data[NumP];
    int size[NumP];
    int busy[NumP];
    int ready[NumP];
    struct lock *pc_lk;
    struct cv *pc_cv;
    ... /* data buffer handling – omitted */
};

```

Il produttore  $i$  (con  $i = 0$  oppure  $1$ ) aggiorna le  $i$ -esime entry nei vettori `data` e `size` (dato e dimensione, in pratica è un buffer). I consumatori possono leggere indifferentemente una delle due caselle dei vettori. Il lock viene utilizzato per la mutua esclusione sui vettori condivisi.

i vettori di flag `ready` e `busy` indicano, se valgono 1:

- `ready`: dato (e dimensione) corrispondente pronto per essere letto
- `busy`: dato attualmente in lettura o in scrittura

Prima di lavorare sui dati, un consumatore deve quindi attendere/verificare che una condizione sia vera: ossia che Questo avviene chiamando la funzione:

```
int consumerWait(struct prodCons *pc);
```

Si risponda alle seguenti domande:

- A) La struttura condivisa può essere posizionata nello stack di thread o dovrebbe essere una variabile globale o altro?
- B) Dato che i campi `pc_lk` e `pc_cv` sono puntatori, dove dovrebbero essere chiamate `lock_create()` e `cv_create()`? Nei thread produttori, nel thread consumatore? Altrove?
- C) Si fornisca un'implementazione della funzione `consumerWait`. Non è necessario alcun codice produttore e/o consumatore, solo la funzione, tenendo presente che la funzione ha come scopo ritornare l'indice (0 o 1) del buffer (dato e dimensione) da cui leggere (avendone posto a 1 il corrispondente flag `ready`). Il dato non va letto (lo farà chi ha chiamato la funzione `consumerWait`)

- 
- A) La struttura condivisa può essere posizionata nello stack di thread o dovrebbe essere una variabile globale o altro?

- B) Dato che i campi `pc_lk` e `pc_cv` sono puntatori, dove dovrebbero essere chiamate `lock_create()` e `cv_create()`? Nei thread produttori, nel thread consumatore? Altrove?

--

C) Si fornisca un'implementazione della funzione `consumerWait`. Non è necessario alcun codice produttore e/o consumatore, solo la funzione.

$$\left. \begin{array}{l} \} \\ \\ \} \\ \end{array} \right\} ;$$

A) La struttura condivisa può essere posizionata nello stack di thread o dovrebbe essere una variabile globale o altro?

Ogni thread ha il proprio stack di thread, quindi i dati condivisi non possono risiedere lì. Una variabile globale è l'opzione più comune. Altre opzioni includono l'allocazione dinamica (con `kmalloc`), se gestita correttamente.

B) Dato che i campi pc\_lk e pc\_cv sono puntatori, dove dovrebbero essere chiamate lock\_create() e cv\_create()? Nei thread produttori, nel thread consumatore? Altrove?

La chiamata potrebbe essere fatta in ciascuno di essi, a condizione che si sincronizzino correttamente: ad es. un thread esegue le inizializzazioni, gli altri thread attendono.

Ma una soluzione più comune potrebbe essere che le inizializzazioni vengano eseguite da un altro thread, ossia il thread padre/principale, il quale sta creando i produttori e il consumatore.

C) Si fornisca un'implementazione della funzione consumerWait. Non è necessario alcun codice produttore e/o consumatore, solo la funzione.

```
/* le istruzioni di controllo/gestione degli errori sono omesse per semplicità */
int consumerWait(struct prodCons *pc) {
    int ret;
    /* lock per mutua esclusione (anche i produttori accedono usando il lock */
    lock_acquire(pc->pc_lk);
    /* un produttore puo' chiamare cv_signal o cv_broadcast ad ogni modifica
       in ogni caso occorre un ciclo, motivato dalla semantica Mesa */
    while ((pc->busy[0]||!pc->ready[0]) && (pc->busy[1] || (!pc->ready[1])) {
        cv_wait(pc->pc_cv, pc->pc_lk);
    }
    if (!pc->busy[0]&&pc->ready[0])
        ret = 0;
    else ret = 1;
    lock_release(pc->pc_lk);
    return ret;
}
```

Commento:

### Domanda 5

Completo

Punteggio ottenuto ,00 su 3,00

**SE I RISULTATI SONO NUMERI, RIPORTARE PASSAGGI INTERMEDI RILEVANTI E/O FORMULE USATE**

**LE RISPOSTE SI/NO VANNO MOTIVATE**

Sia dato un sistema di memoria virtuale con paginazione, nel quale vengono indirizzati i Byte. Il sistema dispone di TLB (Translation Look-aside Buffer). La tabella delle pagine ("page-table") viene realizzata con

uno schema a due livelli, nel quale un indirizzo logico di 64 bit viene suddiviso (da MSB a LSB) in 3 parti:  $p_1$ ,  $p_2$  e  $d$ ,  $p_2$  ha 14 bit e  $d$  13 bit. Non si utilizzano ulteriori strutture dati (quali tabelle di hash o inverted page table) per velocizzare gli accessi. La memoria virtuale viene gestita con paginazione a richiesta.

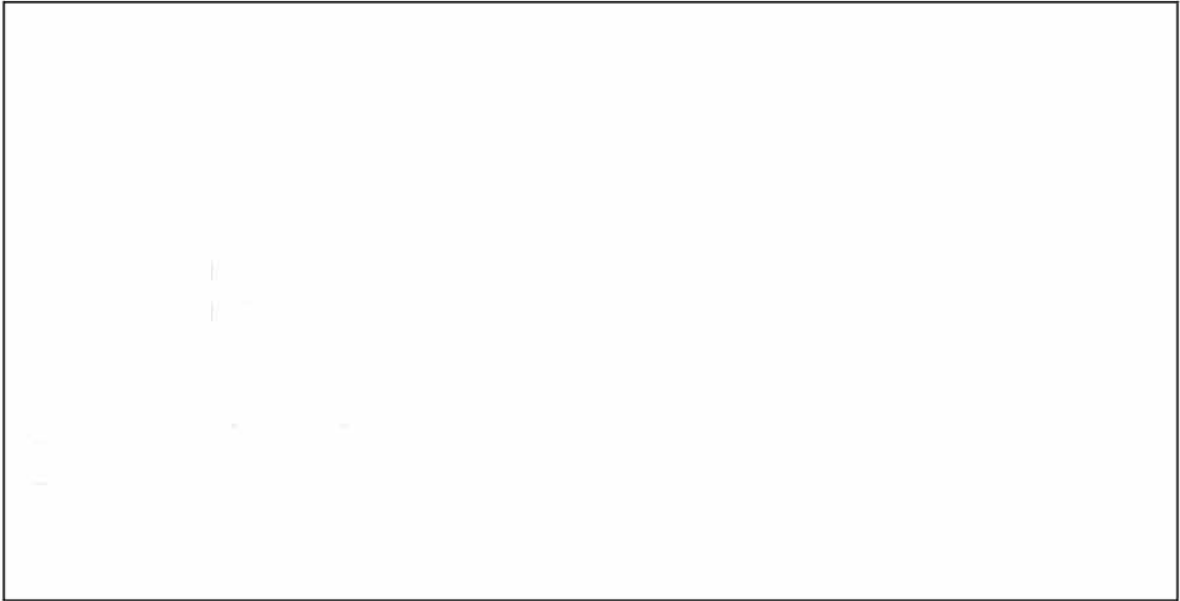
**Si risponda alle seguenti domande:**

- A) Supponendo che la memoria RAM abbia tempo di accesso di 75 ns, si calcoli il TLB miss ratio (frequenza di miss), sapendo che, a causa dei TLB miss, si sa che  $EAT_{PT} \geq 100$  ns, assumendo che il tempo di accesso alla TLB sia trascurabile.  
Si tratta di un lower bound o di un upper bound per la TLB miss ratio?
- B) Si consideri ora la frequenza di page fault  $p_{PF}$ . Una valutazione sperimentale mostra che un page fault viene servito in 2 ms in media, e che il degrado di prestazioni (aumento di tempo di esecuzione) causato dai page fault è compreso tra 10% e 30%. Calcolare l'intervallo di valori di  $p_{PF}$  compatibile con la valutazione sperimentale, assumendo  $EAT_{PT} = 100$  ns.
- C) Una seconda valutazione sperimentale (una simulazione) viene fatta usando due stringhe di riferimento  $w_1, w_2$  aventi lunghezza, rispettivamente,  $len(w_1)=10^6$ ,  $len(w_2)=2 \cdot 10^6$  e probabilità di stringa  $p_1$  e  $p_2$ . Le simulazioni hanno generato 400 page fault in totale (100 su  $w_1$  e 300 su  $w_2$ ) e stimato una probabilità empirica  $f = 0.00012$  (frequenza attesa) che avvenga un page fault nel sistema reale. Si calcolino i valori di  $p_1$  e  $p_2$ .

- 
- A) TLB miss ratio = ...  
upper or lower bound?



- B) Intervallo di valori per  $p_{PF}$



C) Calcolo delle probabilità di stringa  $p_1$  and  $p_2$



**LE RISPOSTE SONO FORMULATE IN INGLESE IN QUANTO COMUNI AL CORSO IN INGLESE DI SYSTEM AND DEVICE PROGRAMMING**

A) TLB miss ratio = ...  
upper or lower bound?



$T_{RAM} = 75 \text{ ns}$  (RAM access time)

$EAT_{PT} = 100 \text{ ns}$

$x = \text{miss}_{TLB} = 1 - h_{TLB}$

2 level (hierarchical) PT  $\Rightarrow$  2 reads for PT lookup

$EAT_{PT} = h_{TLB} * T_{RAM} + (1 - h_{TLB}) * 3T_{RAM} = (1 + 2 * (1 - h_{TLB})) T_{RAM} = (1 + 2x) T_{RAM}$

$2x = EAT_{PT} / T_{RAM} - 1 = 1/3$

$x = 1/6 = 0.17$

Lower bound as a lower miss ratio would decrease  $EAT_{PT}$

B) Range of values for  $p_{PF}$

$T_{PF} = 2 \text{ ms}$  (PF service time)

$p_{PF} * T_{PF}$  = increase in time due to page faults

$0.1 < p_{PF} * T_{PF} / EAT_{PT} < 0.3$

$0.1 < p_{PF} * (2/100) * 10^6 < 0.3$

$0.1 < 2 * 10^{-6} p_{PF} < 0.3$

$5 * 10^{-6} < p_{PF} < 3/2 * 10^{-5}$

$p_{min} = 5 * 10^{-6}$

$p_{Max} = 1.5 * 10^{-5}$

C) Calculation of string probabilities  $p_1$  and  $p_2$

$F_1 = 100, F_2 = 300$  ( $F_1 + F_2 = 400$ )

$p_1 + p_2 = 1$  (we just have two strings)

$f = p_1 * F_1 / \text{len}(w_1) + p_2 * F_2 / \text{len}(w_2)$

$1.2 * 10^{-4} = p_1 * 100 / 10^6 + p_2 * 300 / (2 * 10^6)$

$p_1 = 1 - p_2$

$(1 - p_2) * 10^{-4} + 1.5 * p_2 * 10^{-4} = 1.2 * 10^{-4}$

$1 + 0.5 * p_2 = 1.2$

$p_2 = 0.4$

$p_1 = 0.6$

Commento: