# Programmazione di Sistema
## 6 settembre 2019 (teoria)

*Si prega di rispondere in maniera leggibile, descrivendo i passaggi e i risultati intermedi. Non è possibile consultare alcun materiale. Durata della prova: 70 minuti. Sufficienza con punteggio >= 8. Le due parti possono essere sostenute in appelli diversi. La presenza a una delle due parti annulla automaticamente un'eventuale sufficienza già ottenuta (per la stessa parte): viene intesa come rifiuto del voto precedente.* ***LE RISPOSTE SI/NO VANNO MOTIVATE***

***Le risposte sono state aggiornate e completate, dopo la correzione del compito, per segnalare problemi comuni e aggiungere ulteriori chiarimenti***

***1.*** *(4 punti)* Sia dato il frammento di programma rappresentato, che effettua un calcolo matriciale*:*

```
...
float M[512][512],V[512];
...
for (i=0; i<512; i++) {
 V[i]=0;
 for (j=0; j<=i; j++) {
   if (i%2==0) {
     V[i] += M[i][j];
   }
   else {
     V[i] -= M[i][511-j];
   }
 }
}
...
```

Il codice macchina generato da tali istruzioni viene eseguito in un sistema con memoria virtuale gestita mediante paginazione, con **pagine di 2Kbyte**, utilizzando come politica di sostituzione pagine la **SECOND CHANCE**. Si sa che un `float` ha dimensione **32 bit** e che le istruzioni in codice macchina, corrispondenti al frammento di programma visualizzato, sono contenute in una sola pagina. Si supponga che `M` e `V` siano allocati ad indirizzi logici contigui (prima `M`, poi `V`), a partire dall'indirizzo logico 0x5524AE00. La matrice `M` è allocata secondo la strategia "row major", cioè per righe (prima riga, seguita da seconda riga, …).

- Quante pagine (e frame) sono necessarie per contenere la matrice e il vettore ?

|V| = 512*sizeof(float) = 2KB = 2KB/2KB pagine = 1 pagina
|M| = 512*512*sizeof(float) = 1MB = 1MB/2KB pagine = 0.5K pagine = 512 pagine
L'indirizzo di partenza 0x5524AE00 NON è multiplo di pagina (dovrebbe terminare con 11 bit a 0), ma inizia a ¾ di pagina. quindi sono necessarie correzioni: V sta su 2 pagine e M su 513 (la prima in comune con V).

- Ipotizzando che le variabili i, j siano allocate in registri della CPU, quanti accessi in memoria (in lettura e scrittura) fa il programma proposto, per accedere a dati (non vanno conteggiati gli accessi a istruzioni) ?

Notazione: $N_i$ numero iterazioni del for esterno
$N_j$ numero iterazioni del for interno

Soluzione

```
for (i=0;i<512;i++){ // Ni = 512 iterazioni
V[i]=0;              // 1 Write per iterazione, in totale Ni(512) write
for (j=0;j<=i;j++){  // Nj = i+1 iterazioni, ripetute Ni(512) volte (i cambia)
V[i] += M[i][j];     // Nj = Σi=0..511(i+1) = 512*(512+1)/2 = 128K+256 = 131328
    oppure           // per ogni iterazione 2 Read e 1 Write, in totale
V[i] += M[i][511-j]; // 2*Nj read, Nj write
```

***ATTENZIONE:*** Il calcolo fatto sopra è ***sufficiente***. Per completezza, si effettua sotto il calcolo dettagliato delle iterazioni e degli accessi differenziati tra `i` pari e `i` dispari.

```
V[i] += M[i][j];      // iterazioni con i pari: Nj,0 = 131328/2 = 65664 (*)
                      // per ogni iterazione 2 Read, 1 Write, in totale
                      // 2*Nj,0 read, Nj,0 write
V[i] += M[i][511-j];  // iterazioni con i dispari: Nj,1 = 131328/2 65664 (*)
```

```
                        // per ogni iterazione 2 Read, 1 Write, in totale
                        // 2*N_{j,1} read, N_{j,1} write
```

(*) il risultato è approssimato. Si assume che il numero di iterazioni su j con `i` pari e con `i` dispari coincidano. Il calcolo esatto sarebbe (assumendo `k=i/2`):

$N_{j,0} = \Sigma_{k=0..255}(2k+1) = 1+3+5\ldots+511 = 2*\Sigma_{k=0..255}(k+1)-256 = 256*(256+1)-256 = 256^2 = 65536$

$N_{j,1} = \Sigma_{k=0..255}(2k+2) = 2+4+6\ldots+512 = 2*\Sigma_{k=0..255}(k+1) = 256*(256+1) = 256*257 = 65792$

***Soluzione alternativa***

Un modo (forse) più semplice di ragionare è il seguente: il numero totale di iterazioni $N_j$ coincide con la dimensione di una sottomatrice triangolare inferiore (comprende la diagonale). In realtà le iterazioni con `i` pari percorrono una riga a partire da sinistra, quelle con `i` dispari a partire da destra. Tuttavia il numero totale di iterazioni non cambia: (dimensione della matrice)/2 + (mezza diagonale) = 512*512/2 + 256.
**(Calcolo dettagliato `i` pari / `i` dispari, NON richiesto)** Le iterazioni da destra sono leggermente inferiori a quelle da sinistra (per ogni coppia i-pari i-dispari, la seconda fa una iterazione in più, complessivamente le iterazioni dispari superano le pari di 256). Quindi vale l'equazione:

$2*N_{j,0}+256 = N_j = 128K+256$
$2*N_{j,0} = 128K$
$N_{j,0} = 64K$
$N_{j,1} = N_{j,0} + 256 = 64K + 256$

- Detto $N_T$ il numero totale di accessi a dati in memoria, ed $N_L$ il numero di accessi a dati nella stessa pagina di uno dei precedenti 10 accessi, si definisca come **località** del programma per i dati il numero $L = N_L/N_T$. Si calcoli la località del programma proposto.

***SI SVOLGE L'ESERCIZIO SUPPONENDO ALLINEAMENTO A PAGINA. LA SOLUZIONE ESATTA RICHIEDEREBBE MINIME CORREZIONI***

Siccome ogni riga di M occupa esattamente una pagina, il fatto di percorrere la riga in avanti o all'indietro non ha alcun impatto su località e page fault. Località e page fault sono quindi gli stessi che si avrebbero con l'algoritmo semplificato:

```
for (i=0; i<512; i++) {
   V[i]=0;
   for (j=0; j<=i; j++) {
     V[i] += M[i][j];
   }
}
```

***Accessi a M (in lettura)***

Per ogni valore di `i` (per ogni riga di M), l'unico accesso non locale è il primo (j=0) perché accede a una nuova pagina (nuova riga), a cui non si era ancora fatto accesso. Gli accessi non locali sono quindi 512

***Accessi a V (in lettura o scrittura)***

V sta tutto in una sola pagina. L'unico accesso non locale è il primo. Quindi 1 accesso non locale

***In totale*** gli accessi NON locali sono 1+512 = 513

$N_T = N_i$ (azzeramenti del vettore) $+ 3*N_j$ (iterazioni interne: 2 Read e 1 Write)

$N_L = N_T - 513$ (accessi NON locali)

$L = L = N_L/N_T = (N_T - 513)/N_T = (512 + 3*(128K + 256) - 513) / (512 + 3*(128K + 256)) =$
$= 1 - 513 / (512 + 3*(128K + 256)) \approx 1 - 512/3*128K = 1 - 1/3*256 =$
$= 1 - 0{,}0013 = 9{,}9987$

- Calcolare il numero di page fault generati dal programma proposto, supponendo che siano allocati per esso **10 frame**, di cui uno utilizzato (già all'inizio dell'esecuzione) per le istruzioni. (motivare la risposta)

1 solo page fault per V *(2 tenendo conto del non allineamento)*
Su M 1 page fault per ogni riga/pagina. Poi si continua a farvi accesso.
In totale 1+512 = 513 page fault *(2 + 512 tenendo conto del non allineamento)*

2. *(3 punti)* Sia dato un file system, basato su FAT. I puntatori hanno dimensione 32 bit, i blocchi hanno dimensione 1KB, la FAT occupa 128 MB.
   Quanti blocchi di dato possono contenere complessivamente, al massimo, i file presenti nel file system?

Il numero di blocchi di dato $N_{BL}$ coincide con il numero di righe (celle/indici) nella FAT.
$N_{BL}$ = 128MB/4B = 32M

Quale è la dimensione complessiva massima dei file nel file system?

La dimensione complessiva si ha nel caso in cui la frammentazione interna sia nulla e tutti i blocchi di dato (e la FAT) siano occupati.
$Dim_{MAX} = N_{BL} * |BL| = 32M * 1KB = 32GB$

Quanti blocchi può contenere, al massimo, la free-list?

La dimensione massima della free list si ottiene nel caso di disco completamente vuoto (nessun file).
$|Free\text{-}list|_{MAX} = Dim_{MAX} = 32GB = 32M$ blocchi ($N_{BL}$)
Si noti che in questo caso la free-list è realizzata utilizzando completamente la FAT (per i puntatori/indici)

Sia dato un file "c.exe". Si supponga che "c.exe" sia un file in formato ELF, contenente due segmenti (codice e dati). Il file inizia con un *"executable header"* di 52 Byte, seguito da due *"program header*" di 32 Byte ciascuno, quindi i due segmenti, rispettivamente di 6180 Byte e 8028 Byte.
E' necessario che ognuno degli header e dei segmenti inizi in un nuovo blocco di dati del file? *(motivare la risposta)*

NO.
Il formato del file ELF e l'organizzazione del file in blocchi sono problemi trattati a due livelli completamente diversi: il primo a livello di applicazione, il secondo al livello opportuno del modulo "file system" del sistema operativo.

***ERRORE FREQUENTE:*** *motivare la risposta NO con il fatto che si può fare a meno di frammentare in blocchi, in quanto ci sono tutte le informazioni (puntatori/offset) per trovare i segmenti: La risposta è sbagliata perché presuppone che il file system si preoccupi del contenuto del file (intendendo quindi che in altri casi si sarebbe dovuto partizionare/allineare in base ai blocchi).*

Quanti blocchi di dati occupa il file "c.exe" nel file system proposto?

Dimensione del file:
$|c.exe| = |eh| + 2*|ph| + |codice| + |dati| = 52B + 2*32B + 6180B + 8028B = 14324B$

Occupazione del file:
$N_{BL,c.exe} = \lceil |c.exe| / |BL| \rceil = \lceil 14324B / 1024B \rceil = 14$

Quale è la frammentazione interna di "c.exe"?

$Framm_{c.exe} = N_{BL,c.exe} * |BL| - |c.exe| = 14 * 1024B – 14324B = 12B$

***ERRORI FREQUENTI:*** *considerare come frammentazione la parte OCCUPATA dell'ultimo blocco.*
*Approssimare troppo: passare in base 10 (con la virgola), troncare e calcolare la frammentazione (visto il numero piccolo, si sbaglia di molto, in senso relativo)*
*Fornire una frammentazione in Byte NON interi (tenere il risultato con la virgola: i byte non sono occupati o liberi in parte)*

2. *(3 punti)* Sia dato un disco di tipo "solid state" (SSD).
Quale è l'operazione che ne limita la vita e perché?

La durata di un disco SSD è limitata dal numero massimo di cancellazioni. Le cancellazioni sono effettuate per blocchi/pagine e hanno impatto anche sul numero massimo di scritture, in quanto una scrittura (ri-scrittura) deve essere preceduta da cancellazione.

Si supponga che un disco SSD da 1TB (TeraByte) sia garantito per 3 anni, supportando un massimo di 5TB scritti al giorno. Quale è il numero massimo (complessivo) di Byte che possono essere letti e scritti, durante l'intera vita del disco? (due risposte, una per le letture e una per le scritture, motivando la risposta e indicando i passaggi per calcolare il risultato)

Letture: non c'è limite, le letture non sono un problema
Scritture: il limite è dato dal limite giornaliero, moltiplicato per i giorni in 3 anni.
$N_{write,max} = 5TB * 3*365 = 5*1095\ TB \approx 5.5*10^3 TB$

Sia dato un disco organizzato con struttura RAID. Che cosa si intende, in relazione a tale disco e alla probabilità di guasto e/o tempo medio tra guasti (MTTF: Mean Time To Failure), con i termini seguenti?

- Mean Time To Repair (MTTR):

Tempo Medio necessario per Riparare (e ripristinare) il disco una volta che si è guastato.

- Mean Time To Data Loss (MTTDL):

*(Risposta integrata e completata rispetto alla prima versione)* Tempo Medio (inverso di frequenza/probabilità) tra due perdite di dato (oppure fino alla prima perdita di dato): un dato viene perso perché interviene un secondo guasto dopo un primo, prima che la riparazione sia terminata.

Si consideri una struttura RAID con 2 dischi in configurazione *"mirrored"*. Se ognuno dei dischi può guastarsi in modo indipendente dall'altro, con MTTF = 50000 ore e MTTR = 20 ore, quanto sarà il MTTDL? (non basta il risultato, indicare i passaggi per calcolarlo)

*(per una spiegazione dettagliata si vedano i lucidi Silberschatz modificati:* ***ch11 (con aggiunte su dischi RAID)****)*

MTTDL = $MTTF^2/(2*MTTR) = 50000^2/(2*20)$ ore = $25/4*10^7$ ore = $6.25*10^7$ ore

4. *(4 punti)* Sia dato un sistema operativo OS161.

4.a) Perché, in un contesto multi-core (sono presenti più CPU) non è possibile realizzare la mutua esclusione semplicemente disabilitando e riabilitando l'interrupt ?

Perché l'interrupt verrebbe disabilitato sulla sola CPU corrente, questo non precluderebbe quindi l'esecuzione di altri thread o processi sulle altre CPU.

***ATTENZIONE:*** *Non sarebbe neppure sufficiente estendere la disabilitazione degli interrupt alle altre CPU, in quanto su queste ci potrebbero già essere altri thread in esecuzione. La disabilitazione degli interrupt serve quindi solo nel caso di un single core, in quanto significa che il thread corrente è l'unico in esecuzione sul sistema.*

Dato il codice (ridotto alle parti essenziali) delle funzioni di semaforo P e V, riportate in seguito

```
void P(struct semaphore *sem) {
   spinlock_acquire(&sem->sem_lock);
   while (sem->sem_count == 0) {
     wchan_sleep(sem->sem_wchan,
                 &sem->sem_lock);
   }
   sem->sem_count--;
   spinlock_release(&sem->sem_lock);
}
```

```
void V(struct semaphore *sem) {
   spinlock_acquire(&sem->sem_lock);
   sem->sem_count++;
   KASSERT(sem->sem_count > 0);
   wchan_wakeone(sem->sem_wchan,
                 &sem->sem_lock);
   spinlock_release(&sem->sem_lock);
}
```

Si risponda alle seguenti domande:

- A cosa serve lo spinlock (in entrambe le funzioni)?

Lo spinlock è necessario per poter utilizzare un wait-channel, quindi per chiamare correttamente `wchan_sleep` e `wchan_wakeone`. Serve a garantire la gestione in mutua esclusione della condizione (`sem->sem_count`)

***ATTENZIONE:*** *Si valuta corretta solo in parte una risposta "generica", in cui si spieghi cosa sia uno spinlock. La domanda è relativa all'utilità dello spinlock in queste due funzioni.*

- Perché la P contiene un ciclo `while` anziché un `if (sem->sem_count == 0)`, mentre il ciclo non è presente nella V?

Perché la sincronizzazione tra `wchan_wakeone` e `wchan_sleep` (il risveglio, realizzato con semantica "Mesa", anziché "Hoare") non garantisce che la condizione, vera alla chiamata di `wchan_wakeone`, lo sia ancora al ritorno da `wchan_sleep`. Altri thread potrebbero modificarla nel frattempo.

**ERRORE FREQUENTE:** *pensare che esistano* ***risvegli "spuri"****errati.* ***NON CI SONO RISVEGLI SBAGLIATI****. Il problema è semplicemente che in un contesto di programmazione concorrente ci potrebbero essere altri thread/processi svegliati (****correttamente****, da altri thread, in questo caso da altre chiamate a V), SENZA*

*GARANTIRE (semantica Mesa) LA CRONOLOGIA STRETTA (garantita dalla semantica di Hoare) dell'esecuzione (stato RUN) dei thread, corrispondente ai risvegli (che non mandano in stato di RUN ma solo nella coda READY)*

- Perché la `wchan_sleep` riceve come parametro lo spinlock? Vale lo stesso motivo per la `wchan_wakeone`?

La `wchan_sleep` deve rilasciare lo spinlock, prima di mettere il thread in stato "wait", per poi riprenderlo al risveglio (prima di ritornare al chiamante). La `wchan_wakeone` non deve fare nulla sullo spinlock (le versioni Unix/Linux di tale finzione, ad esempio, NON hanno questo parametro): in OS161 il parametro viene solo verificato (in una KASSERT, per prevenire eventuali errori del programmatore) perché al momento delle chiamata il thread deve essere owner dello spinlock.

***ERRORI FREQUENTI:*** *pensare che lo spinlock serva per garantire la mutua esclusione all'interno di `wchan_sleep` o `wchan_wakeone`: NO. Le funzioni al loro interno non avrebbero bisogno dello spinlock. Si asserisce l'ownership dello spinlock per verificare che CHI CHIAMA le funzioni stia facendo le cose bene, NON per far funzionare correttamente il wait channel (che, questo si, deve rilasciare e riacquisire lo spinlock in `wchan_sleep`).*
*Secondo errore (peggiore): pensare che un thread si metta in attesa sullo spinlock. Non ci si mette in attesa su uno spinlock, lo spinlock serve per mutua esclusione, non per sincronizzazione di tipo wait-signal.*

- E' possibile che la chiamata alla `wchan_wakeone` svegli più di un **semaforo** *(improprio: meglio dire **thread**, vedi spiegazione sotto)* in attesa su `wchan_sleep`? Se NO, perché? Se SI, come si fa in modo di rilasciare un solo thread in attesa su `P`?

**ATTENZIONE:** *La domanda è posta male (errore mio). La formulazione corretta sarebbe "… la chiamata alla `wchan_wakeone` svegli più di un **thread** in attesa …". Infatti siamo nel contesto di un solo semaforo, sul quale ci potrebbero essere molteplici thread che fanno P, come pure V. Fatta questa premessa, ne ho tenuto conto nella correzione, ma non mi pare di aver riscontrato problemi di interpretazione.*

NO. C'è la garanzia che la `wchan_wakeone` svegli un solo thread in attesa su `wchan_sleep`. La funzione che sveglia più thread in attesa (tutti) è la `wchan_wakeall`.

4b) Si riporta una possibile realizzazione della funzione `getfreeppages`, che alloca un intervallo di `npages` pagine contigue di memoria ***fisica*** libera.
La funzione realizza una politica di allocazione best-fit, worst-fit, first-fit o altro (motivare la risposta) ?

***Nessuna delle tre.***
First-fit NO perché le iterazioni non si fermano alla prima soluzione trovata
Best-fit e Worst fit NO in quanto non c'è una ricerca/gestione di minimo o massimo
***La soluzione ritornata è l'ultima tra quelle trovate. La si potrebbe denominare "last-fit".***

***ERRORE FREQUENTISSIMO:*** *pensare che la soluzione proposta sia **first-fit***
***COMMENTI (alla parte successiva, il programma da modificare):*** *quasi tutti gli studenti che hanno riconosciuto la last-fit, hanno proposto la modifica corretta per la first-fit.*
*Pochissimi sono stati in grado di affrontare il problema della ricerca del **"minimo tra gli intervalli di lunghezza >= np"** (o npages).*
*Si noti che l'algoritmo proposto è O(nRamFrames) e che la versione iniziale controlla la lunghezza (per semplicità) in tutte le caselle intermedie di un intervallo.*
*Per realizzare una worst-fit (ricerca di un massimo) sarebbe sufficiente aggiungere la gestione di una variabile max ad ogni iterazione.*
*La best-fit invece, deve fare in modo di confrontarsi con il minimo provvisorio solo quando si arriva al termine di un intervallo (ad esempio confrontando la cella di indice i+1).*
*Ho visto tentativi di soluzioni O(nRamFrames*npages) basati su doppia iterazione; salvo l'efficienza, possono essere ritenuti corretti.*
*Non ritengo invece accettabili algoritmi che prevedano un vettore aggiuntivo già precaricato con le lunghezze degli intervalli. In effetti questo vettore risolverebbe in parte il problema, ma (al di la dell'occupazione di memoria), con un costo aggiuntivo per mantenerlo aggiornato ad ogni allocazione/de-allocazione (ciò che si vuole evitare con la bitmap). Oppure il vettore sarebbe precaricato con una passata lineare iniziale nella `getfreeppages`, ma senza ottenere nulla di meglio rispetto a calcolare al volo le lunghezze degli intervalli (come proposto). In pratica, tutti coloro che hanno utilizzato questa soluzione hanno cercato di spostare/evitare (senza indicare come risolverlo) un pezzo del problema.*
*Non sono valutate strategie "a parole": è necessario proporre codice in "C".*

Si modifichi la funzione (scrivendo nel riquadro libero le parti modificate delle versioni aggiornate) in modo tale da realizzare una politica first-fit e una best-fit.

```
static paddr_t
getfreeppages(unsigned long npages) {
  paddr_t addr = 0;
  long i, first, found, np = (long)npages;

  if (!isTableActive()) return 0;
  spinlock_acquire(&freemem_lock);
  for (i=0,first=found=-1;i<nRamFrames;i++) {
    if (freeRamFrames[i]) {
      /* controlla ogni frame libero (anche
         interno all'intervallo.
         Se è il primo di un intervallo,
         "ricorda" la posizione. */
      if (i==0 || !freeRamFrames[i-1])
        /* set first free in an interval */
        first = i;
      /* ogni intervallo compatibile con np
         viene assegnato. Quindi si ritorna
         l'ultimo */
      if (i-first+1 >= np)
        found = first;
    }
  }
  if (found>=0) {
    for (i=found; i<found+np; i++) {
      freeRamFrames[i] = (unsigned char)0;
    }
    allocSize[found] = np;
    addr = (paddr_t) found*PAGE_SIZE;
  }
  spinlock_release(&freemem_lock);
  return addr;
}
```

```
/* ATTENZIONE: la soluzione proposta non è
l'unica: ne sono possibili altre (simili) */

/* first-fit: appena trovato esce */
/* variante con uscita strutturata */
...
  for (i=0,first=found=-1;
       i<nRamFrames && found<0; i++) {
...
/* variante con uscita non strutturata */
...
      if (i-first+1 >= np){
        found = first;
        break;
      }
...

/* best fit: ricerca minimo (solo alla fine di
un intervallo) */
int min;
...
   /* si possono in alternativa mettere tutte
      le condizioni in AND (unico if) */
   /* se è l'ultimo frame di un intervallo */
   if (i==nRamFrames-1 || !freeRamFrames[i+1])
     /* se la dimensione va bene */
     if (i-first+1 >= np)
       /* se batte il minimo provvisorio */
       if (found<0 || i-first+1 < min) {
        found = first; min = i-first+1;
       }
...
```