Programmazione di Sistema
24 luglio 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***

Soluzione proposta: *si intenda la soluzione come indicativa, in particolare nel caso di domande a risposta aperta.*

1. *(3 punti) Sia data la stringa di riferimenti a pagine 3, 4, 1, (3, 1, 4, 4, 3, 1, 1)*10, in cui la sintassi (…)*n indica che la stringa tra parentesi viene ripetuta/iterata n volte (la stringa può ad esempio, derivare da un costrutto iterativo).*

   Si utilizzi un algoritmo di sostituzione pagine di tipo ***working-set*** (versione ***esatta***) con finestra di durata $\Delta = 3$. Si supponga di denominare *page-out* la rimozione di una pagina dal resident set (in quanto esce dal working set). Si visualizzino nello schema che segue i riferimenti e il resident set dopo ogni riferimento, indicando i page-fault (accessi a pagine non presenti nel resident set) e i page-out. Si richiede la visualizzazione solo fino alle prime due iterazioni della sotto-stringa ripetuta 10 volte. *ATTENZIONE: ogni riga del resident set rappresenta un frame, quindi una pagina presente in un frame non può cambiare riga quando rimane nel resident set. Nel caso di page-fault senza page-out, si utilizzi il primo frame libero dall'alto. Nel caso di page-out e (contemporaneo) page-fault, viene riutilizzato il frame appena liberato (da page-out). Qualora page-out e page-fault siano relativi alla stessa pagina, l'algoritmo di sostituzione ne tiene conto, evitando sia page-out che page-fault.*

Tempo	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Riferimenti	3	4	1	3	1	4	4	3	1	1	3	1	4	4	3	1	1
Resident Set	3	3	3	3	3	3			1	1	1	1	1	1	3	3	3
		4	4	4		4	4	4	4				4	4	4	4	
			1	1	1	1	1	3	3	3	3	3	3			1	1
Page Fault	x	x	x			x		x	x				x		x	x	
Page Out					x		x	x		x				x	x		x

   Il comportamento del resident set nelle 8 iterazioni non riportate sarà
   - identico a quello dell'intervallo 10-16 (ripetuto 8 volte)? (SI/NO motivare)
     No, in quanto, pur se il periodo è di 7 tempi, le pagine 1 e 3 non si trovano nello stesso frame all'inizio e alla fine del periodo
   - oppure ripeterà 4 volte l'intervallo 3-16? (SI/NO motivare)
     No, in quanto, pur essendo i 14 istanti l'unione dei primi due periodi di 7 istanti, il primo periodo ha un comportamento iniziale diverso per la presenza della pagina 4 al tempo 3 (il quarto)
   - oppure avrà una diversa configurazione (quale?)?
     Si ripete 4 volte l'intervallo 3-16, a patto di eliminare la pagina 4 dal tempo 3 e il page-out al tempo 4. Oppure (detto in un altro modo) si ripete 8 volte l'intervallo 10-16, alternando ad ogni iterazione i contenuti del primo e del terzo frame.

   Quanti page fault e page-out ci saranno in totale (comprese le 8 iterazioni mancanti?
   9 PF e 7 PO nella parte visualizzata, 8*(3 PF e 3 PO) nella parte mancante: in totale 33 PF e 31 PO

2. *(4 punti)* Sia dato un file system Unix, basato su inode aventi 13 puntatori (10 diretti, 1 indiretto singolo 1 doppio e 1 triplo). I puntatori/indici hanno dimensione 32 bit e i blocchi hanno dimensione 1KB. Si sa che il file system contiene N file ($1000 \leq N \leq 10000$), la cui dimensione complessiva *(ATTENZIONE: la somma delle dimensioni dei singoli file, quindi indipendente dal tipo di file system)* è 501MB. La frammentazione interna complessiva è 3 MB.

   Il dato sulla frammentazione interna permette di limitare ulteriormente l'intervallo dei valori possibili per N? (se NO, motivare la risposta, se SI, calcolare l'intervallo):

   La frammentazione complessiva va ripartita sugli N file. Pur potendo calcolare la frammentazione media, ogni singolo file potrebbe avere frammentazione di valore arbitrario (eventualmente 0). Il dato sulla frammentazione NON può quindi ridurre il valore massimo di N (10000) in quanto, dato un qualunque numero di file aventi frammentazione non nulla, se ne potrebbero aggiungere in numero arbitrario con frammentazione 0.

   Il valore minimo può per N può invece essere aumentato. Si noti infatti che, con N=1000 si avrebbe

frammentazione media = 3MB/1000 ≈ 3KB, troppo grande, con blocchi di 1KB.
Per calcolare il valore minimo di N si consideri la frammentazione massima possibile per un file, cioè 1KB-1B = 1023B.

N ≥ 3MB/1023B ≈ *(1023 approssimato con 1024)* 3MB/1KB = 3K = 3072
senza approssimazione (risultato esatto)
N ≥ ⌈3MB/1023B⌉ = 3076

Quale è il massimo numero di file, tra gli N presenti, ad utilizzare indice indiretto triplo?

Un blocco indice contiene 1KB/4B = 256 puntatori/indici.
Per calcolare il numero massimo occorre considerare la dimensione minima.

Si indica con $|F_3|$ la dimensione di un file che richiede indice indiretto triplo e con N_3 il numero di tali file. Si fanno i calcoli sulla dimensione dei file. In alternativa si potrebbe operare sull'occupazione (la differenza tra occupazione e dimensione è la frammentazione interna, che in questo caso può essere considerata trascurabile).

$|F_3| > (10 + 256 + 256^2)$ blocchi = $(266+256^2)$KB

$N_3 * |F_3| < 501MB$

$N_3 < 501MB / ((266+256^2)KB) = 501*1024/(266+256^2) \approx 501*4 / 257 = 2002 / 257 \approx 7.8$

$N_{3,MAX} = 7$ (il numero massimo di file deve essere un intero)

Quale è il massimo numero di file, tra gli N presenti, ad utilizzare indice indiretto singolo?

Si indica con $|F_1|$ la dimensione di un file che richiede indice indiretto singolo e con N_1 il numero di tali file*:*

$|F_1| > 10$ blocchi = 10KB

$N_1 * |F_1| < 501MB$

$N_1 < 501MB / 10KB = 50.1K$

Il dato supera il vincolo del problema: N < 10000. Si utilizza quindi il vincolo più restrittivo: $N_{1,MAX} = 10000$ (tutti i file con indice indiretto singolo)

Sia dato, in questo file system, un file testo "a.txt" contenente una sequenza di 10220 caratteri ASCII (a capo compresi) ripartiti su 200 righe. Quanti blocchi di dato occupa il file?

(Dimensione file) |a.txt| = 10220B = 10220/1024 blocchi
Occupazione in blocchi = ⌈10220/1024⌉ = 10 blocchi

Se si modifica tale file mediante un editor di testi, spostando la prima riga, di 40 caratteri, in fondo al file (operazione *cut-paste*, *taglia-incolla*, cioè cancella all'inizio del file e aggiungi in fondo), la nuova versione di a.txt (dopo modifica e salvataggio della nuova versione del file) avrà un blocco di indici indiretto singolo oppure no (motivare la risposta)?

La dimensione del file non cambia, pur avendo il nuovo file una diversa collocazione di ognuno dei caratteri. L'eventuale ***sola*** aggiunta di una riga di 40 caratteri avrebbe fatto superare la dimensione minima che richiede indice indiretto singolo, ma la contemporanea cancellazione dei 40 caratteri all'inizio lascia invariata la dimensione totale.

3. *(3 punti)* Sia dato un sistema con paginazione basato su Inverted Page Table (IPT).

Che cosa contiene una riga della IPT?
Contiene il PID del processo cui la pagina appartiene e il numero di pagina logica (p), più eventuali flag (protezioni, modify bit, ecc.)

La dimensione della IPT dipende dalla dimensione della RAM oppure dagli spazi di indirizzamento dei processi in esecuzione (motivare la risposta)?
Dipende unicamente dalla dimensione della RAM, in quanto la IPT contiene un entry per ogni frame fisico

Confrontata con una tabella della pagine standard, quali sono i vantaggi egli svantaggi di una IPT?
Vantaggi:

- memoria: contiene solo le pagine associate a frame
- memoria: una sola tabella per tutti i processi

Svantaggi

- tempo: la IPT in versione base base ha tempo di ricerca lineare mentre la PT standard ha tempi O(1)
- difficile realizzare la memoria condivisa (sharing): pagine diverse mappate sullo stesso frame

Perché può essere conveniente combinare una IPT con una tabella di hash?
Per ridurre i tempi di ricerca da lineari a O(1)

Si supponga di usare una tabella di hash basata su chaining (liste di collisione) per effettuare la conversione da `(pid,p)` a `f`, tale tabella può avere meno righe della IPT? (motivare)
Si. Una tabella di hash con chaining non ha vincoli di dimensione, anche se questo ha ovviamente un impatto sulle prestazioni (che dipendono dal fattore di carico, cioè (numero di dati / dimensione del vettore))

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

4a) Si riporta una parte della funzione `as_define_region` (file `dumbvm.c`). Si supponga di ricevere, per i parametri `as` e `vaddr` i valori (esadecimali) `0x80048720` e `0x412370`, e per `sz` il valore (decimale) `4128`. Ricordando che PAGE_SIZE è definito come `4096`, e PAGE_FRAME come `0xfffff000`, si simulino le istruzioni proposte, indicando, per ognuna, in esadecimale, i valori degli operandi e del risultato (il valore assegnato alla variabile).

```
as_define_region(struct addrspace *as, vaddr_t vaddr, size_t sz,
         int readable, int writeable, int executable) {
    size_t npages;

    /* Align the region. First, the base... */
    sz += vaddr & ~(vaddr_t)PAGE_FRAME;
    vaddr &= PAGE_FRAME;
    /* ...and now the length. */
    sz = (sz + PAGE_SIZE - 1) & PAGE_FRAME;
    npages = sz / PAGE_SIZE;
    ...
}
```

	4096 = 0x1000 4128 = 0x1020
`sz += vaddr & ~(vaddr_t)PAGE_FRAME;`	0x1020 += 0x412370 & 0x000FFF sz <- 0x1020+0x370 = **0x1390**
`vaddr &= PAGE_FRAME;`	0x412370 &= 0xFFF000 vaddr <- **0x412000**
`sz = (sz + PAGE_SIZE - 1) & PAGE_FRAME;`	sz <- (0x1390+0xFFF) & 0xFFF000 = **0x2000**
`npages = sz / PAGE_SIZE;`	npages <- 0x2000 / 0x1000 = 2

Si supponga che il valore di `sz` ricevuto sia inferiore alla dimensione di una pagina (es `4090`). E' possibile che si ottenga per `npages` il valore 2? (motivare la risposta)
Si. E' possibile, in quanto il segmento viene allineato a un multiplo di pagina sia all'inizio che alla fine. In questo caso c'è di fatto una forma di frammentazione interna sia sulla prima che sull'ultima pagina. Con 4090 = 0xFFA, sz avrebbe prima assunto il valore `0xFFA+0x370 = 0x126A`, quindi `0x2000`.

4b) Si consideri la realizzazione dei lock in un sistema OS161. Quale thread deve essere considerato owner (proprietario) di un lock?

- Il thread che ha creato il lock?
  NO. Aver creato un lock non significa essere l'owner.

- l'ultimo thread che ha chiamato la funzione `lock_acquire`?
  Non necessariamente. Solo nel caso in cui la lock_acquire sia stata fatta sul lock in questione (non su un altro lock) e il thread abbia effettivamente acquisito il lock (non sia ancora in attesa)
- altro…(completare)
  Si veda la risposta precedente: l'owner di un lock è il thread che ha effettuato `lock_acquire` sul lock ed ha superato l'eventuale attesa.

Sono date le funzioni `lock_release` e `lock_do_i_hold` proposte in figura. Nelle funzioni sono presenti errori: li si identifichi e li si corregga (occorre motivare/spiegare)

```
void loc_release(struct lock *lock) {
  KASSERT(lock != NULL);
  spinlock_acquire(&lock->lk_lock);
  KASSERT(lock_do_i_hold(lock));
  lock->lk_owner=NULL;
  wchan_wakeone(lock->lk_wchan, &lock->lk_lock);
  spinlock_release(&lock->lk_lock);
}
```

```
bool
lock_do_i_hold(struct lock *lock) {
  spinlock_acquire(&lock->lk_lock);
  if (lock->lk_owner==curthread)
    return true;
  spinlock_release(&lock->lk_lock);
  return false;
}
```

L'errore nella lock_do_i_hold è l'istruzione return senza rilascio dello spinlock. Una possibile correzione è data dall'uso di una variabile booleana

```
bool lock_do_i_hold(struct lock *lock) {
  bool ret;
  spinlock_acquire(&lock->lk_lock);
  ret = lock->lk_owner==curthread;
  spinlock_release(&lock->lk_lock);
  return ret;
}
```

L'errore nella `lock_release` è dato dall'acquisizione dello spinlock prima della `lock_do_i_hold`, che tenterà di acquisire lo stesso spinlock. Si tratta di un problema di deadlock. Una possibile soluzione consiste nello spostare la `spinlock_acquire` dopo la chiamata a `lock_do_i_hold`.

E' possibile realizzare la `sys_waitpid` utilizzando per l'attesa un lock, su cui fare `lock_acquire`, mentre la segnalazione, da parte della `sys_exit` viene realizzata con `lock_release` dello stesso lock? (motivare la risposta)
NO. Un lock non può essere usato per trasmettere sincronizzazioni di tipo wait-signal, proprio per il problema dell'ownership: il lock serve per mutua esclusione. Per wait-signal si consigliano semafori o condition variable (l'eventuale lock associato serve per mutua esclusione, non per gestire wait-signal).

4c) Si vogliono realizzare le system calls `sys_open` e `sys_close`. E' necessario associare a un file descriptor il concetto di ownership da parte di un thead, in modo tale che solo il thread che ha fatto la `open` di un file sia autorizzato a chiamare la `close` del file? (motivare la risposta)
NO. Un file non ha concetti di ownership di questo tipo: un file può essere chiuso da un thread diverso da quello che lo ha aperto. Indirettamente invece, un file ha un concetto di ownership legato al processo, in quanto un file descriptor è associato al contesto di un processo (e della relativa tabella dei file aperti).

A cosa servono le funzioni OS161 `copyin` e `copyout`?
Servono a effettuare copie di dati tra memoria user e kernel, copyin ha come destinazione la memoria kernel, dualmente la copyout. La principale differenza rispetto ad altre forma di copia da memoria a memria consiste nel fatto che vengono gestite in modo consistente eventuali errori/eccezioni legate a indirizzi/puntatori user non validi, impedendo così al kernel di terminare in modo anomalo (es. crash/panic).

SI supponga di sostituire una chiamata `copyin(src,dst,size);`
con `memmove(dst,src,size);`
Si tratta di una sostituzione lecita? Si perde o guadagna qualcosa, oppure sono istruzioni equivalenti?
SI. La sostituzione è lecita (sono possibili altre soluzioni) ma si perde la protezione da eccezioni/errori.