Esame di Sistemi Operativi
AA 2018/19
16 Luglio 2019
[soluzione]

Nome	Cognome	Matricola

Esercizio 1

Sia data la seguente tabella che descrive il comportamento di un insieme di processi.

Process	T_{start}	**CPU Burst**	**IO Burst**
P1	0	6	5
P2	0	5	3
P3	3	1	10
P4	5	3	1

Domanda Si assuma di disporre di uno scheduler preemptive con politica di selezione dei processi Round Robin (RR) con quanto di tempo $T = 2$. Si assuma inoltre che:

- i processi in entrata alla CPU dichiarino il numero di burst necessari al proprio completamento;
- l'operazione di avvio di un processo lo porti nella coda di ready, ma **non necessariamente** in esecuzione.
- il termine di un I/O porti il processo che termina nella coda di ready, ma **non necessariamente** in esecuzione.

Si illustri il comportamento dello scheduler in questione nel periodo indicato, avvalendosi degli schemi di seguito riportati (vedi pagina seguente). Si supponga che i processi **si ripresentino con le stesse specifiche** una volta finito l'I/O.

Soluzione Date le specifiche dell'algoritmo di scheduling, la traccia di esecuzione dei processi e' illustrata in Figura 1. Nella soluzione ivi riportata, nel caso in cui due processi siano entrambi nelle stesse condizioni, viene valutato per prima quello con `pid` minore.

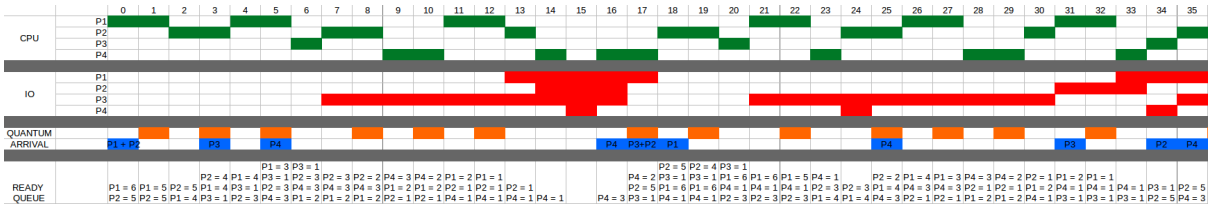


Figure 1: Traccia di esecuzione dei processi usando l'algoritmo di scheduling richiesto. In verde ed in rosso sono riportati rispettivamente i cicli di CPU e IO; in arancione ed in blu, invece, sono scanditi il time quantum e l'arrivo di un processo.

Nome	Cognome	Matricola

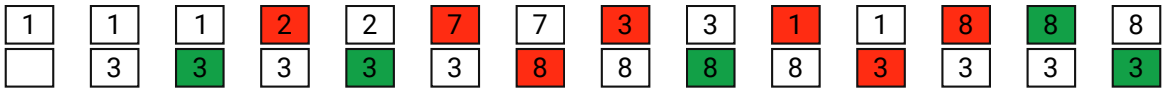
Esercizio 2

Sia data la seguente traccia di accesso alle pagine di memoria:

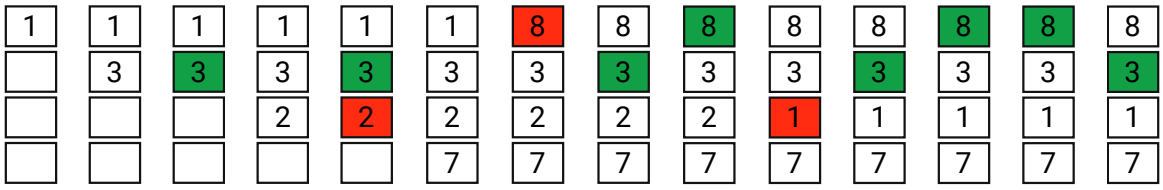
1 3 3 2 3 7 8 3 8 1 3 8 8 3

Si assima di avere un Translation Lookaside Buffer (TLB) di 2 elementi, gestito con politica Last Recently Used (LRU). Si assuma che T_{fetch} e T_{TLB} siano rispettivamente i tempi di fetch e di accesso al TLB. Di quanto aumentano le prestazioni incrementando la dimensione del TLB a 4 elementi?

Soluzione Le tracce di accesso alle pagine nei due casi sono illustrate rispettivamente in Figura 2a e Figura 2b. Come e' possibile notare dalle figure, raddoppiare il numero di entry del TLB portera'



(a) Traccia di accesso alle pagine con TLB a due elementi.



(b) Traccia di accesso alle pagine con TLB a quattro elementi.

Figure 2: Illustrazione raffigurante la traccia di accesso alle pagine. In verde sono evidenziati i page hit.

ad un sostanzioso aumento delle prestazioni. Piu' in particolare, ponendo $T_{miss} = 2(T_{TLB} + T_{RAM})$ e $T_{hit} = T_{TLB} + T_{RAM}$, avremo:

$$T_{EAT}^{[2]} = 9T_{miss} + 5T_{hit} \tag{1}$$

$$T_{EAT}^{[4]} = 6T_{miss} + 8T_{hit} \tag{2}$$

Dalle Equazioni (1) e (2) si evince che la probabilita' di page fault passera' da $p_{fault}^{[2]} \approx 0.64$ a $p_{fault}^{[4]} \approx 0.43$, risultando in un aumento dell'hit ratio del 21%.

Nome	Cognome	Matricola

Esercizio 3

Cos'e' un File Control Block (FCB)? Quali sono le informazioni contenute al suo interno?

Soluzione Il FCB e' una struttura dati che contiene tutte le informazioni relative al file a cui e' associato. Esempi di informazioni possono essere: permessi, dimensione, data di creazione, numero di inode (se esiste), ecc. . Inoltre il FCB contiene informazioni sulla locazione sul disco dei dati del file - ad esempio in un File System (FS) con allocazione concatenata il puntatore al primo blocco del file. In Figura 3 e' riportata una illustrazione di tale struttura.

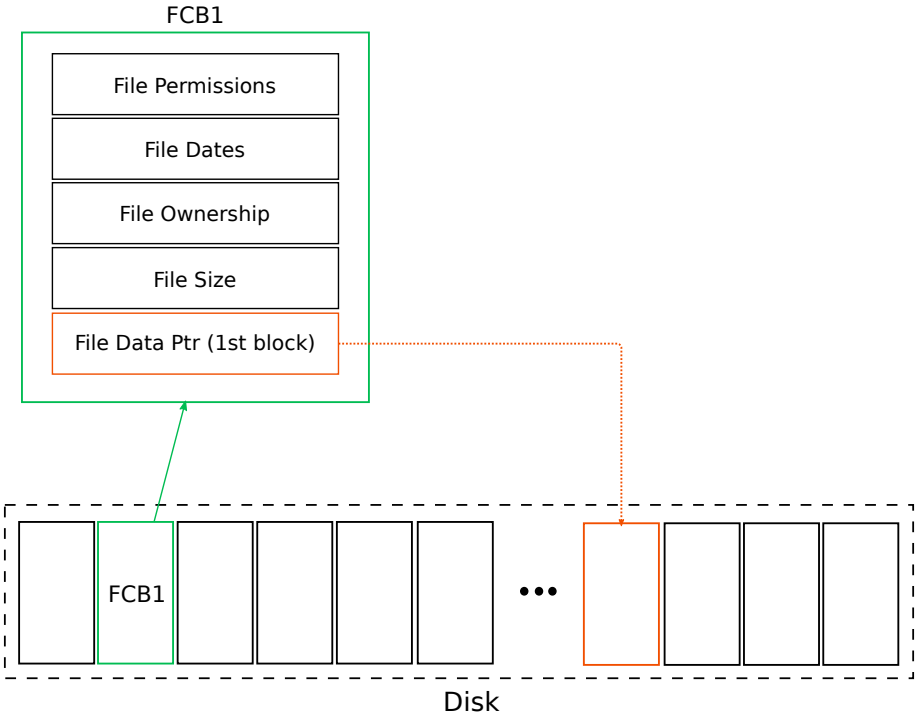


Figure 3: Esempio di FCB. La struttura contiene tutti gli attributi del file compresa la locazione dei dati - qui rappresentato dal puntatore al primo blocco della lista contente i dati, supponendo un FS con Linked List Allocation (LLA).

Nome	Cognome	Matricola

Esercizio 4

Si consideri un file consistente di 60 blocchi e che il suo FCB sia gia' in memoria. Siano dati due file-systems gestiti rispettivamente tramite allocazione a lista concatenata - LLA - e allocazione contigua - Contiguous Allocation (CA). Si assuma che nel caso di CA, eventuale spazio per estendere il file sia disponibile solo alla fine dello stesso - non all'inizio.

Domanda Si calcolino le operazioni di I/O su disco necessarie per eseguire le seguenti azioni in entrambi i file-systems:

- Rimozione di un blocco all'inizio del file
- Rimozione di un blocco ad un terzo del file
- Rimozione di un blocco alla fine del file

Soluzione Nel caso di LLA per effettuare una operazione bisognera' scorrere la lista fino al blocco in questione. Nell'implementazione con CA i blocchi sono allocati in maniera sequenziale sul disco, quindi ogni volta bisognera' ricopiare tutti i blocchi in modo da "compattarli". Date queste premesse, i risultati sono i seguenti:

1. LLA: 1 `IO-ops`; CA: 118 `IO-ops`
2. LLA: 22 `IO-ops`; CA: 78 `IO-ops`
3. LLA: 60 `IO-ops`; CA: 0 `IO-ops`

Si noti che, nell'implementazione con CA la rimozione di un file in posizione $n = 20$ richiede di spostare tutti i blocchi posteriori a quello in questione. Si ricorda che per spostare un blocco e' necessario prima leggerlo (1 `IO-ops`) e poi scriverlo nella posizione giusta (1 `IO-ops`).

Nome	Cognome	Matricola

Esercizio 5

Sia dato il seguente programma:

```
#define STACK_SIZE 16384
#define ITERATIONS 3

ucontext_t main_context, f1_context, f2_context;
pthread_attr_t t_attr;
pthread_t t_id;
void* f3(void* param);
char f1_stack[STACK_SIZE];char f2_stack[STACK_SIZE];

void f1() {
  printf("f1: start\n");

  pthread_attr_init(&t_attr);
  pthread_create(&t_id,&t_attr,f3,NULL);
  pthread_join(t_id,NULL);

  swapcontext(&f1_context, &f2_context);

  printf("f1: end\n");
  setcontext(&main_context);
}

void f2() {
  printf("f2: start\n");
    for (int i=0; i<ITERATIONS; i++) {
    printf("f2: %d\n", i);
  }
  swapcontext(&f2_context, &f1_context);

  printf("f2: end\n");
  setcontext(&main_context);
}

void* f3(void* param) {
  for (int s = 0; s < ITERATIONS; ++s)
    printf("f3: %d\n", s);
  return param;
}

int main() {
  printf("- program start -\n");

  getcontext(&f1_context);
  f1_context.uc_stack.ss_sp=f1_stack;
  f1_context.uc_stack.ss_size = STACK_SIZE;
  f1_context.uc_stack.ss_flags = 0;
  f1_context.uc_link=&main_context;
  makecontext(&f1_context, f1, 0, 0);

  f2_context=f1_context;
  f2_context.uc_stack.ss_sp=f2_stack;
  f2_context.uc_stack.ss_size = STACK_SIZE;
  f2_context.uc_stack.ss_flags = 0;
  f2_context.uc_link=&main_context;
  makecontext(&f2_context, f2, 0, 0);

  swapcontext(&main_context, &f1_context);

  printf("- program end -\n");
}

```

Domanda Cosa stampa il programma?

Soluzione L'output del programma e' il seguente:

```
1
2     - program start -
3     f1: start
4     f3: 0
5     f3: 1
6     f3: 2
7     f2: start
8     f2: 0
9     f2: 1
10    f2: 2
11    f1: end
12    - program end -
13
```

Si noti che la stampa della riga 30 non verra' mai eseguita, poiche', dopo il ciclo di `f2`, il contesto passera' ad `f1`, che a sua volta fara' uno switch al contesto originario del `main`.

Nome	Cognome	Matricola

Esercizio 6

Che relazione ce tra una *syscall*, un generico *interrupt* e una *trap*? Sono la stessa cosa?

Soluzione Ovviamente le tre cose non sono la stessa cosa.

Una syscall e una chiamata diretta al sistema operativo da parte di un processo user-level - e.g. quando viene fatta una richiesta di IO.

Un interrupt invece e' un segnale asincrono proveniente da hardware o software per richiedere il processo immediato di un evento. Gli interrupt software sono definiti *trap*. A differenza delle syscall, gli interrupt esistono anche in sistemi *privi di Sistema Operativo* - e.g. in un microcontrollore. Quando una syscall viene chiamata, una trap verra' generata (un interrupt software), in modo da poter richiamare l'opportuna funzione associata a tale syscall (attraverso la Syscall Table (ST)).

Nome	Cognome	Matricola

Esercizio 7

A cosa serve la syscall `ioctl`? Come si puo configurare la comunicazione con *devices a caratteri e seriali* in Linux?

Soluzione La syscall `ioctl` permette di interagire con il driver di un device generico - e.g. una webcam. Tramite essa sara' possibile ricavare e settare i parametri di tale device - e.g. ricavare la risoluzione della webcam o settarne la tipologia di acquisizione dati.

Per configurare devices seriali a caratteri - e.g. terminali - e' possibile usare le API racchiuse nella interfaccia `termios`. Tramite di essa, avremo accesso a tutte le informazioni relative al device - e.g. baudrate, echo, ...

Nome	Cognome	Matricola

Esercizio 8

Quali sono le principali differenze tra un indirizzo *logico* ed un indirizzo *fisico*? Da chi vengono generati?

Soluzione Un indirizzo logico non si riferisce ad un indirizzo realmente esistente in memoria. Esso e' in realta' un indirizzo astratto generato dalla CPU, che verra' poi tradotto in un indirizzo fisico tramite la Memory Management Unit (MMU). L'indirizzo fisico, quindi, si riferisce ad una locazione esistente della memoria e *non* e' generato dalla CPU, bensi' dalla MMU.