Esame di Sistemi Operativi
AA 2018/19
22 Gennaio 2020
[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.

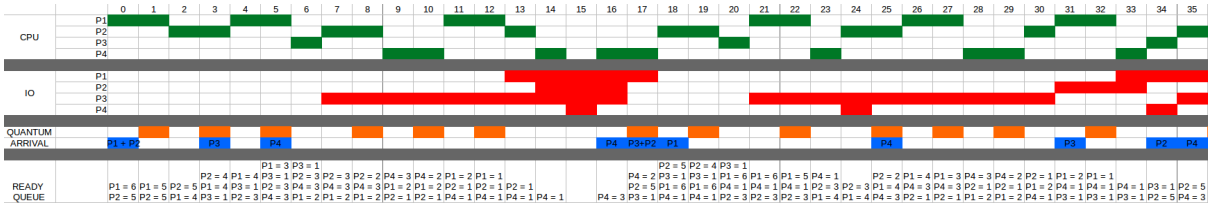


Figure 1: Traccia di esecuzione dei processi usando l'algoritmo di scheduling richiesto. In verde ed in rosso sono riportati rispettivamente i cicli di CPU e IO; in arancione ed in blu, invece, sono scanditi il time quantum e l'arrivo di un processo.

Nome	Cognome	Matricola

Esercizio 2

Sia dato un sottosistema di memoria con paginazione, caratterizzato dalle seguenti dimensioni:

- frame → `2 GB`
- memoria fisica indirizzabile → `256 GB`.

Si calcolino:

A. Il numero di bit minimo per indicizzare tutte le pagine

B. Il valore di p_{fault}, considerando che il tempo di accesso medio ad una pagina e' di 120 ns, $T_{RAM} = 100\ ns$ e $T_{TLB} = 5\ ns$.

Soluzione

A. Data la dimensione di ogni pagina pari a `2 GB`, saranno necessari `31 bit` per indicizzare un elemento all'interno della stessa. La memoria fisica, invece, necessita di almeno `38 bit`. Il numero di bit *minimo* per indicizzare tutte le pagine e' quindi pari a `38-31 = 7 bit`.

B. La formula per il calcolo del tempo di accesso medio alla memoria - *Effective Access Time* - e' data dalla seguente relazione:

$$T_{EAT} = p_{hit}(T_{TLB} + T_{RAM}) + (1 - p_{hit}) \cdot 2(T_{TLB} + T_{RAM}) \tag{1}$$

Sostituendo i dati della traccia nella (1) avremo $p_{fault} = 1 - p_{hit} = \frac{1}{7} \approx 0.14286$.

Nome	Cognome	Matricola

Esercizio 3

Cose 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 2 e' riportata una illustrazione di tale struttura.

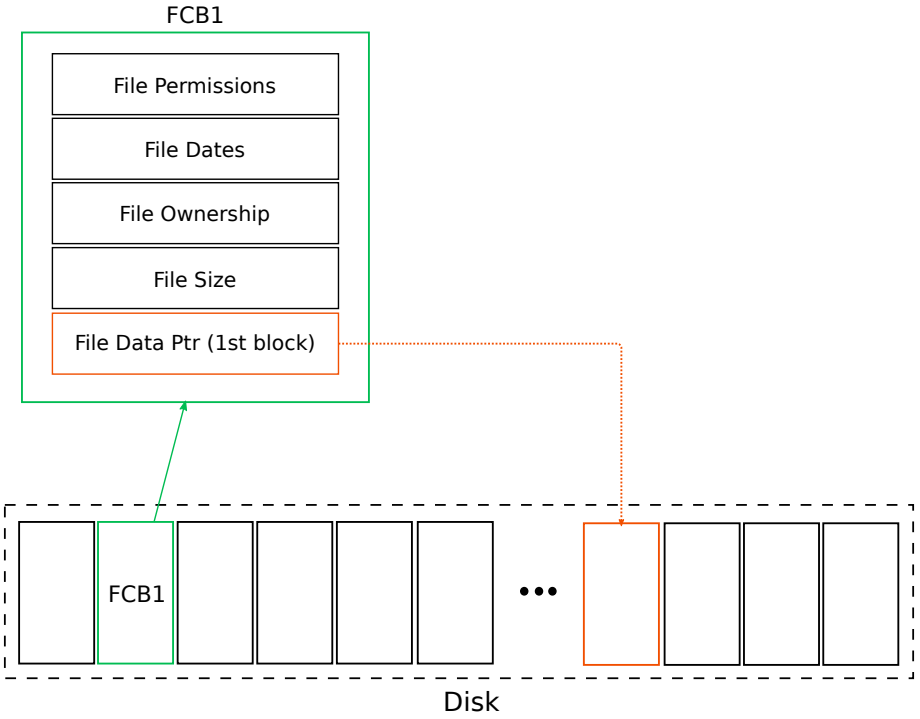


Figure 2: Esempio di FCB. La struttura contiene tutti gli attributi del file compresa la locazione dei dati - qui rappresentato dal puntatore al primo blocco della lista contente i dati, supponendo un FS con Linked List Allocation (LLA).

Nome	Cognome	Matricola

Esercizio 4

Con riferimento agli algoritmi di *Page Replacement*, enumerare i 4 principali algoritmi usati per tale scopo, ordinandoli in base al loro *page-fault rate* (dal piu' alto al piu' basso). Si spieghi brevemente il fenomeno dell'anomalia di Belady, evidenziando gli algoritmi soggetti ad essa.

Soluzione Partendo dall'algoritmo con le peggiori performances in termini di *page-fault rate*, avremo:

#	**Algorithm**	**Belady**
1.	FIFO	si'
2.	Second Chance	si'
3.	LRU	no
4.	Optimal	no

L'anomalia di Belady e' un fenomeno che si presenta in alcuni algoritmi di rimpiazzamento delle pagine di memoria per cui la frequenza dei *page-fault* puo' aumentare con il numero di pagine disponibili (con alcuni pattern di accesso alle pagine).

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;
char f1_stack[STACK_SIZE];
char f2_stack[STACK_SIZE];

void f1(void);
void f2(void);

void f1() {
  printf("f1: start\n");
  int cnt = 0;

  swapcontext(&f1_context, &f2_context);
  if (cnt < ITERATIONS) {
    printf("f1: %d\n", cnt++);
    setcontext(&f1_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);
  }

  printf("f2: end\n");
  swapcontext(&f2_context, &f1_context);
}

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");
}

```

Indicare quale dei seguenti puo essere un possibile output del programma e motivare la risposta:

- Output A:

```
- program start -
- program end -
```

- Output B:

```
- program start -
f1: start
f2: start
f2: 0
f2: 1
f2: 2
f2: end
f1: 0
f1: 1
f1: 2
f1: end
- program end -
```

- Output C:

```
- program start -
f1: start
f2: start
f2: 0
f2: 1
f2: 2
f2: end
f1: 0
f1: end
- program end -
```

- Output D:

```
- program start -
f1: start
f2: start
f2: 0
f2: 1
f2: 2
f2: end
f1: 1
f1: end
- program end -
```

- Output E: Nessuna delle precedenti (specificare quindi l'output di seguito).

Soluzione L'Output B rispecchia cio' che il programma stampa realmente. Piu' nello specifico, in riga 14 la funzione `swapcontext` aggiorna `f1_context` (che ripartira' dunque da riga 15) e "passa il comando" alla funzione `f2` tramite `f2_context`. Una volta finito il suo ciclo, `f2` fara' ripartire il contesto `f1_context`, che ripartira' dalla riga 15. Infine la funzione `setcontext` in riga 17, fara' ripartire il programma ogni volta dalla riga 15, finche' il numero di iterazioni non sara' raggiunto. Cio' poiche' funzione `setcontext` **non** modifica il contesto in argomento (come fa invece la `swapcontext`).

Nome	Cognome	Matricola

Esercizio 6

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.

Nome	Cognome	Matricola

Esercizio 7

Sia dato un sistema con paginazione con 32 `GB` di memoria fisica indirizzabile. Si supponga, inoltre, di avere un totale di `8 388 608` (ovvero 8 `M`) pagine e che il Translation Look-aside Buffer (TLB) abbia 3 entries. Infine, si supponga che il TLB sia gestito con politica di rimozione della pagine Last Recently Used (LRU). Date queste specifiche, si supponga di avere un unico processo in esecuzione, e che la sua dimensione sia 4 `kB`. Si consideri quindi un array bidimensionale `float A[][]` con dimensioni $R \times C$ con $R = 1$ e $C = 50$, allocato come:

```
1
2  float** A = (float**)malloc(sizeof(float*)*R);
3  for (int r = 0; r < R; ++r)
4    A[r] = (float*)malloc(sizeof(float)*C);
5
```

Supponendo che le ultime pagine utilizzate in fase di allocazione siano gia' nel TLB, calcolare il numero di *page fault* dovute all'inizializzazione dell'array nelle seguenti modalita':

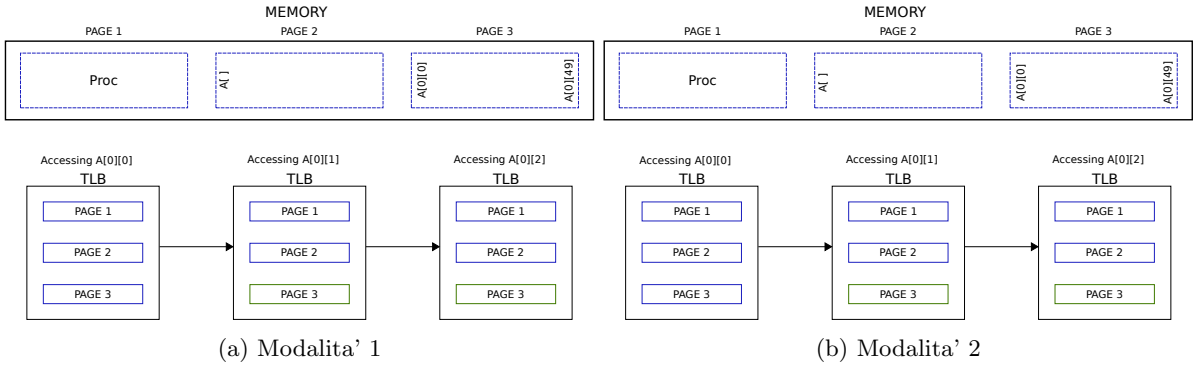
Modalita' 1:

```
1
2  for (int r = 0; r < R; ++r)
3    for (int c = 0; c < C; ++c)
4      A[r][c] = 0;
5
```

Modalita' 2:

```
1
2  for (int c = 0; c < C; ++c)
3    for (int r = 0; r < R; ++r)
4      A[r][c] = 0;
5
```

Soluzione Data la memoria fisica di 32 `GB`, serviranno `35 bit` per indirizzarla correttamente. Inoltre dato che nel sistema ci sono `8 388 608` pagine, serviranno `23 bit` per indicizzarle tutte. Da qui' possiamo calcolare facilmente che ogni pagina avra' dimensioni pari a $2^{35-23} = 2^{12}$ byte, ovvero `4 kB`. In base a tali specifiche e a come avviene l'allocazione della matrice `A[][]` il numero di pagine utilizzate in totale e' 3. Esse entrano tutte nel TLB, percio' in entrambe le modalita' il numero di page fault e' pari a 0 - poiche' quando la matrice viene allocata, le pagine vengano caricate nel TLB. Inoltre, per le specifiche dimensioni dell'array, le due modalita' di accesso si equivalgono. In Figura 3 e' riportata una breve illustrazione del sottosistema di memoria in entrambi i casi.



(a) Modalita' 1 (b) Modalita' 2

Figure 3: Illustrazione della memoria e del TLB in entrambe le esecuzioni.

Nome	Cognome	Matricola

Esercizio 8

Spiegare brevemente la differenza tra `fopen(...)` e `open(...)`.

Soluzione `fopen(...)` una funzione di alto livello che ritorna uno stream, mentre `open(...)` una syscall di basso livello che ritorna un *file descriptor*. `fopen(...)`, infatti, contiene nella sua implementazione una chiamata alla syscall `open(...)`.