Esame di Sistemi Operativi
AA 2018/19
06 Novembre 2019

Nome	Cognome	Matricola

Esercizio 1

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

Process	T_{start}	**CPU Burst 0**	**IO Burst 0**	**CPU Burst 1**	**IO Burst 1**
P1	0	3	2	7	1
P2	2	2	2	2	1
P3	3	4	4	6	6
P4	4	10	1	1	1

Domanda Si assuma di disporre di uno *scheduler preemptive* Shortest Remaining Job First (SRJF) con quanto di tempo pari a $T_q = 3$. 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).

Soluzione La traccia di esecuzione dei processi che soddisfi le specifiche di cui sopra e' riportata in Figura 1. Poiche' lo scheduler considera i burst *rimanenti* di ogni processo, il processo P4 non riuscira' ad essere eseguito fino a che gli altri non avranno finito.

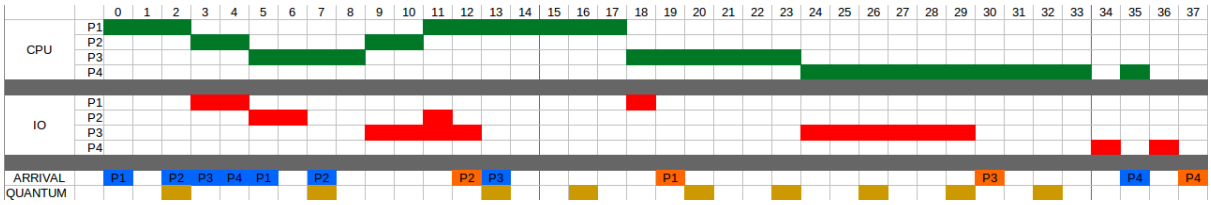


Figure 1: Traccia di esecuzione dei processi con scheduler SRJF e time quantum pari a $T_q = 3$. CPU burst, IO burst, arrivo e fine dei processi sono indicati rispettivamente in verde, rosso, blu e arancione. In giallo viene evidenziato l'intervento del time quantum.

Nome	Cognome	Matricola

Esercizio 2

Si consideri in sottosistema di memoria il caratterizzato dalle seguenti tabelle

Number	Base	Limit
0x00	0x000	0x002
0x01	0x002	0x006
0x02	0x008	0x001
0x03	0x009	0x003
0x04	0x00C	0x001
0x05	0x00E	0x001

Table 1: Segmenti

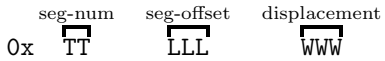
Page	Frame
0x000	0x010
0x001	0x00F
0x002	0x00E
0x003	0x00D
0x004	0x00C
0x005	0x00B
0x006	0x00A
0x007	0x009
0x008	0x008
0x009	0x007
0x00A	0x006
0x00B	0x005
0x00C	0x004
0x00D	0x003
0x00E	0x002
0x00F	0x001
0x010	0x000

Table 2: Pagine

Domanda Assumendo che le pagine abbiano una dimensione di 512 byte, che la tabella delle pagine consista di 512 elementi e che la tabella dei segmenti possa contenere 256 elementi, come vengono tradotti in indirizzi fisici i seguenti indirizzi logici?

- `0x05001FA1`
- `0x01007001`
- `0x00000000`
- `0x01005AAC`
- `0x03F02000`

Soluzione In base alle specifiche del sistema, ogni indirizzo virtuale si potra' scomporre come segue:



I primi due digit piu' significativi individuano il `segment number`. Useremo quindi `base + seg-offset` per individuare il frame all'interno della tabella delle pagine. Ovviamente, se il `seg-offset` eccede il `limit` di quel segmento, si avra' un errore. L'indirizzo finale sara' dato da `frame + displacement`.

Sulla base di quanto detto, avremo i seguenti indirizzi fisici:

- ⊙ `0x05001FA1` → `0x001FA1`
- ⊙ `0x01007001` → invalido
- ⊙ `0x00000000` → `0x010000`
- ⊙ `0x01005AAC` → `0x009AAC`
- ⊙ `0x03F02000` → invalido

Nome	Cognome	Matricola

Esercizio 3

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 4

Spiegare brevemente cose il Direct Memory Access (DMA) e quando viene usato.

Soluzione Alcuni device necessitano di trasferire grandi quantita' di dati a frequenze elevate. Effettuare tali trasferimenti tramite il protocollo genericamente usato per altri tipi di periferiche richiederebbe l'intervento della CPU per trasferire un byte alla volta i dati - tramite un processo chiamato Programmed I/O (PIO). Cio' risulterebbe in un overhead ingestibile per l'intera macchina, consumando inutilmente la CPU. Per consentire il corretto funzionamento di tali device evitando gli svantaggi del PIO, tali periferiche possono avvalersi di controllori dedicati che effettuano DMA, cioe' andando a scrivere direttamente sul bus di memoria. La CPU sara' incaricata soltanto di "validare" tale trasferimento e poi sara' di nuovo libera di eseguire altri task.

Questo tipo di periferiche sono molto comuni ai giorni nostri e sono usate nella maggior parte dei dispositivi elettronici - pc, smartphones, servers, ... Esempi di periferica che si avvalgono di controller DMA sono videocamere, dischi, schede video, schede audio, ecc.

Nome	Cognome	Matricola

Esercizio 5

Sia dato il seguente programma C:

```c
#define NUM_STEPS 5
unsigned int value = 0;
pid_t pid;
pthread_t tid;

void* runner(void* param);

int main(int argc, char *argv[]) {
  pthread_attr_t attr;
  pid = fork();

  if (pid < 0)
    return -1;

  if (pid == 0) {
    pthread_attr_init(&attr);
    pthread_create(&tid,&attr,runner,NULL);
    pthread_join(tid,NULL);
    printf("line C, value = %d\n",value); /* LINE C */
  } else {
    wait(NULL);
    printf("line P, value = %d\n",value); /* LINE P */
  }
  return 0;
}

void* runner(void* param) {
  for (int s = 0; s < NUM_STEPS; ++s) {
    if (!pid) {
      value++;
    }
  }
  return param;
}

```

Domanda Cosa stampa il programma?

Soluzione Il programma in traccia crea un processo `child` tramite `fork`, andando a sdoppiare le variabili del processo `parent`. Inoltre il `child` andra' a creare un nuovo thread, che incrementa - eventualmente - la variabile `value`.

Date queste considerazioni, l'output del programma sara' il seguente:

```

line A, value = 5
line B, value = 0

```

Nome	Cognome	Matricola

Esercizio 6

Si consideri un file consistente di 60 blocchi e che il suo File Control Block (FCB) sia gia' in memoria. Siano dati due File System (FS) gestiti rispettivamente tramite allocazione a lista concatenata - Linked List Allocation (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 FS:

- 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 7

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 8

Che relazione c'e' 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)).