Operating Systems 2017/2018 | Exercises

Ricardo Rocha

Department of Computer Science

Faculty of Sciences

University of Porto

Exercises

1.1

- Considere as seguintes afirmações:

  A1: Vários processos podem ser executados simultaneamente em diferentes CPUs

  A2: A execução de processos é alternada tão frequentemente (mudança de contexto) que permite que os utilizadores interajam com os seus processos enquanto estes executam, aumentando assim o tempo de resposta do sistema como um todo

  A3: Um novo processo é escalonado para executar logo que o processo em execução bloqueie à espera de um evento (e.g., operação de I/O)

  A4: Um novo processo é escalonado para executar logo que um processo termine a sua execução

  - Quais definem um sistema de operação com suporte para *multiprogramming*?
  - Quais definem um sistema de operação com suporte para *multitasking*?

2.1

- Considere alguns dos eventos que podem ocorrer durante a existência de um processo:
  - **E1:** Ser criado por invocação da chamada *fork()* por parte do processo pai
  - **E2:** Executar a chamada ao sistema *pause()*
  - **E3:** Executar a chamada ao sistema *exit()*
  - **E4:** Executar em modo de *kernel*
  - **E5:** Receber interrupção indicando a conclusão duma operação de I/O

  - Quais colocam o processo no estado de *Ready*?
  - Quais colocam o processo no estado de *Waiting*?

3.1

- Considere os seguintes algoritmos de escalonamento de processos:
  - **A1:** FCFS (*First Come First Served*)
  - **A2:** RR (*Round Robin*)
  - **A3:** SJF (*Shortest Job First*)
  - **A4:** SRTF (*Shortest Remaining Time First*)

  - Quais podem levar a inanição (*starvation*)?

3.2

- Considere a seguinte sequência de pedidos de tempo de execução (*CPU burst*), por ordem de chegada, em que os pedidos são representados por tuplos do tipo <processo,cpu_burst>.

<P1,20>, <P2,5>, <P3,10>, <P4,4>

Ignorando o tempo de mudança de contexto, indique qual o tempo médio passado na fila de espera (*waiting time*), desses pedidos utilizando o algoritmo de escalonamento de processos:

  - FCFS (*First Come First Served*)
  - RR (*Round Robin*) com *time quantum* 5

4.1

- Uma solução para o problema da região crítica deve satisfazer os seguintes requisitos:
  - **R1:** Garantir exclusão mútua, i.e., apenas um processo pode estar a executar numa zona crítica em cada momento
  - **R2:** Garantir progresso, i.e., apenas os processos à entrada duma zona crítica podem participar na decisão de qual o próximo processo a entrar nessa zona crítica e essa decisão não pode ser adiada indefinidamente
  - **R3:** Garantir um limite no tempo de espera, i.e., desde que um processo se apresenta à entrada duma zona crítica, deve existir um limite no número de outros processos que entram na zona crítica, até que o processo ganhe esse acesso

4.1

Considere a solução que se segue em que cada processo executa a função **proc()** e em que **test_and_set()** implementa a instrução atómica de *test_and_set*:

```
boolean lock = false;  // variável partilhada

void proc() {
   while(1) {
      while (test_and_set(&lock));
      regiao_critica();
      lock = false;
      regiao_nao_critica();
   }
}
```

- Quais requisitos a solução acima satisfaz?

4.2

- Considere um sistema com recursos do tipo A, B e C dos quais existem respetivamente 1, 2 e 3 instâncias. Considere ainda a tabela que se segue como representativa do estado de alocação desses recursos pelos 5 processos existentes (P1, P2, P3, P4 e P5).

	Atribuído (em posse)			Pedido (à espera)		
	A	B	C	A	B	C
P1	0	1	1	0	1	0
P2	1	0	0	0	0	1
P3	0	1	0	0	1	1
P4	0	0	0	1	0	0
P5	0	0	1	0	0	0

  - Quais os processos que se encontram em *deadlock*?

5.1

- Considere as várias secções que formam o espaço de endereçamento de um processo:
  - **S1:** *Text*, onde reside o código do programa
  - **S2:** *Data*, onde residem as variáveis globais
  - **S3:** *Heap*, onde reside a memória alocada dinamicamente
  - **S4:** *Stack*, onde reside informação da sequência de chamadas em execução

  - Em processos *multithreaded*, quais secções são acessíveis (i.e., a memória é partilhada) por todas as *threads* de um processo?
  - Em processos *multithreaded*, quais secções são replicadas por cada nova *thread*?

6.1

- Tipicamente, para manusear os ficheiros abertos num sistema de ficheiros é comum utilizar-se duas tabelas internas: a *system-wide open-file table* e a *per-process open-file table*. Alguma da informação associada a um ficheiro aberto inclui:
  - **I1:** Descritor do ficheiro (*file descriptor*)
  - **I2:** Tamanho do ficheiro
  - **I3:** Modo de acesso ao ficheiro (*read-only*, *write-only* ou *read/write*)
  - **I4:** Número de processos que têm o ficheiro aberto
  - **I5:** Apontador para a localização da última leitura/escrita

  - Da informação acima, qual é guardada na *system-wide open-file table*?
  - Da informação acima, qual é guardada na *per-process open-file table*?

6.2

- Considere um sistema de ficheiros baseado em *inodes* com 10 apontadores diretos para blocos e 2 apontadores indiretos, um dos quais indireto simples e outro indireto duplo. Cada bloco pode conter até 100 endereços.
  - Qual é o tamanho máximo de um ficheiro em número de blocos?

7.1

- Considere a seguinte sequência de pedidos de acesso a disco em que os pedidos são representados por tuplos do tipo <num_cilindro,num_setor>.

<50,4>, <70,8>, <120,10>, <300,40>, <0,0>

Sabendo que a cabeça de leitura do disco acabou de se movimentar do cilindro 90 para o cilindro 100 (onde se encontra atualmente), indique qual a ordem de serviço desses pedidos utilizando o algoritmo:

- SSTF (*Shortest Seek Time First*)
- SCAN

7.2

- Considere os seguintes tempos de latência duma operação de I/O:
  - **T1:** *Queuing time*
  - **T2:** *Controller time*
  - **T3:** *Seek time*
  - **T4:** *Rotational time*
  - **T5:** *Transfer time*

  - Quais são relevantes no acesso a dispositivos de armazenamento HDD (*Hard Disk Drive*)?
  - Quais são relevantes no acesso a dispositivos de armazenamento SSD (*Solid State Drive*)?

8.1

- Considere os seguintes métodos de alocação de memória:
  - **M1:** Alocação contígua de memória com partições de tamanho fixo
  - **M2:** Alocação contígua de memória com partições de tamanho variável
  - **M3:** Segmentação
  - **M4:** Paginação
  - **M5:** Paginação com segmentação

  - Quais sofrem do problema de fragmentação interna?

8.2

- Considere a tabela de segmentos que se segue num sistema de gestão de memória com segmentação em que os endereços lógicos são representados por tuplos do tipo <num_segmento,deslocamento>.

	Base	Limite
0	7000	500
1	9000	1000
2	5000	500
3	1000	1000
4	3000	500

  - A que endereço físico corresponde o endereço lógico <1,999>?
  - A que endereço lógico corresponde o endereço físico 2222?

8.3

- Considere um sistema multiprocessador com 16 CPUs e um programa P que durante a sua execução cria 10 *threads*. Se num determinado momento, estiverem 3 processos do programa P a executar, cada um com 10 *threads* (i.e., 30 *threads* no total), quantas tabelas de páginas existem no sistema (representando os 3 programas P em execução)?

8.4

- Considere um sistema de gestão de memória com segmentação e paginação em que um endereço lógico tem 32 bits, os quais incluem 12 bits para o deslocamento, 9 bits para o número do segmento e 11 bits para o número da página.

  - Qual é o tamanho de cada página?
  - Qual é o número máximo de páginas por processo?

8.5 *

- Considere a seguinte tabela de páginas num sistema de gestão de memória com paginação de um nível. O tamanho das páginas é de 1024 bytes, a memória física máxima é de 2 MBytes e o tamanho máximo do espaço de endereçamento é de 16 MBytes.
  - A que endereço físico corresponde o endereço lógico 1524?
  - A que endereço lógico corresponde o endereço físico 4100?
  - Quantos bits são necessários para cada entrada da tabela de páginas?
  - Qual é o número máximo de entradas numa tabela de páginas?

page	frame
0	4
1	2
2	16
3	17
...	...

8.6 *

- Considere um sistema de gestão de memória com paginação de dois níveis em que um endereço lógico tem 32 bits os quais incluem 9 bits para a tabela de primeiro nível, 11 bits para a de segundo nível e os restantes para o deslocamento.
  - Qual é o tamanho de cada página?
  - Qual é o número máximo de páginas por processo?

8.7 *

- Considere um sistema de gestão de memória com segmentação e paginação em que um endereço lógico tem 32 bits os quais incluem 12 bits para o deslocamento, 11 bits para o número do segmento e 9 bits para o número da página.
  - Qual é o tamanho máximo de um segmento?
  - Qual é o número máximo de páginas por segmento?
  - Qual é o número máximo de páginas por processo?
  - Qual é o número máximo de segmentos?

8.8 *

- Considere um sistema de gestão de memória com TLBs e paginação em que o tempo de acesso à TLB é de 2ns e o tempo de acesso à memória é de 50ns.
  - Qual é o tempo de acesso efetivo à memória se estivermos a usar paginação de um nível e 80% das referências forem encontradas na TLB?
  - Qual é o tempo de acesso efetivo à memória se estivermos a usar paginação de dois níveis e 80% das referências forem encontradas na TLB?

9.1

- Considere um sistema de gestão de memória virtual com paginação a pedido (*demand paging*) em que um processo tem 4 molduras (*frames*) de memória física atribuídas, com a seguinte informação:

	Página	Transferida	Última Referência	R	M
0	45	1000	1023	1	1
1	7	1001	1015	1	0
2	100	999	1014	1	1
3	52	1002	1011	1	1

Na ocorrência de uma falha no acesso a uma página (*page fault*), qual página seria substituída utilizando o algoritmo de substituição de páginas:

- FIFO (*First In First Out*)
- NRU (*Not Recently Used*)

9.2

- Considere um sistema de gestão de memória virtual com paginação a pedido (*demand paging*) em que um processo acede à seguinte sequência de páginas:

  A, B, A, C, D, A, E, B, A, C

  Sabendo que ao processo foram atribuídas 3 molduras (*frames*) de memória física, inicialmente vazias, e que o sistema utiliza um esquema de substituição local (i.e., cada processo utiliza apenas o seu conjunto de molduras e não troca molduras com outros processos), indique qual a sequência de páginas transferidas de memória para o disco com o algoritmo de substituição de páginas:

  - LRU (*Least Recently Used*)
  - Segunda Tentativa (assuma que numa falha de página, o bit R é iniciado a 0)

9.3 *

- Considere a tabela de páginas que se segue com 8 molduras (frames) atribuídas. Na ocorrência de uma falha no acesso a uma página (page fault), qual moldura seria substituída pelos seguintes algoritmos:
  - FIFO
  - LRU
  - NRU

Frame	Transferida	Última Referência	R	M
0	129	156	1	1
1	132	143	1	0
2	111	111	0	0
3	96	116	0	1
4	152	162	1	1
5	126	146	1	1
6	90	164	1	0
7	138	154	1	1

9.4 *

- Considere a seguinte sequência de acessos a páginas:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

Para uma memória física dividida em 4 molduras (frames) inicialmente vazias, indique que falhas de página irão ocorrer utilizando os algoritmos:

- FIFO
- LRU
- Segunda tentativa
- OPT