Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

Sistemas de Operação

Exame NM (Ano Letivo de 2017/18) 15 de janeiro de 2018

Nome: _____ NMec: _____

NOTA: Numa questão em que se peça uma justificação e ela não seja dada, a resposta não será considerada.

. . . . . .

1. Considere o programa apresentado a seguir, que representa excertos dos códigos de 3 processos colaborantes. A sincronização é feita através de 3 semáforos, associados às variáveis `sem1`, `sem2` e `sem3`. As operações de *down* e *up* são realizadas pelas funções `sem_down` e `sem_up`, respetivamente. Considere ainda que, após as devidas inicializações, cada processo/thread executa o ciclo `for` correspondente.

```
1   /* Processo/Thread 1: */
2   for (int i = 0; i < 3; i++)
3   {
4       sem_down(sem1);
5       printf("A"); fflush(stdout);
6       sem_up(sem3);
7   }
8
9   /* Processo/Thread 2: */
10  for (int i = 0; i < 3; i++)
11  {
12      sem_down(sem2);
13      printf("B"); fflush(stdout);
14  }
15
16  /* Processo/Thread 3: */
17  for (int i = 0; i < 3; i++)
18  {
19      sem_down(sem3);
20      printf("C"); fflush(stdout);
21      sem_up(sem1);
22      sem_up(sem2);
23  }
```

b) O wait bloqueia sempre enquanto que o down está associado a uma variável persistente, por isso é qeu é que o wait precisa de um while(cond): wait.
O signal é efímero enquanto que o up não.

A atribuição dos valores tem de ser feito em exclusão mútua

b)
sem_up e se_dpwn com variáveis de condição e uma condição.
A função sem_up que vai ter agora args uma var condição e uma condição. vai fazer um lock. cond++;
signal
unlock

sem_down (var_cond, cond)
lock
cond++
signal
unlock

(a) i. Com que valores mínimos têm de ser inicializados os semáforos `sem1`, `sem2` e `sem3` de modo a que a saída "BCABCABCA" possa ocorrer.

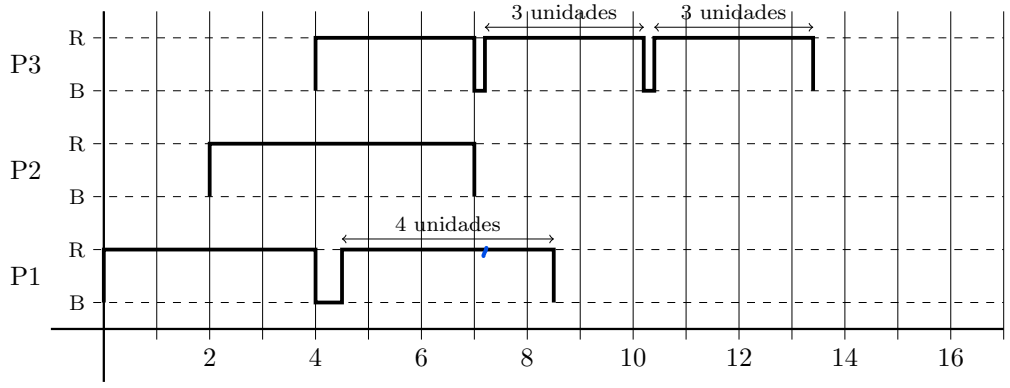
ii. Considerando a inicialização que indicou, apresente mais duas saídas possíveis.

(b) Em ambientes *multithreading* é habitual usar-se variáveis de condição em vez de semáforos para a sincronização entre *threads*. Compare as operações de *wait* e *signal*, aplicáveis às variáveis de condição, com as operações de *down* e *up*, aplicáveis a semáforos.

(c) Considerando uma solução em threads, re-implemente o código dado usando variáveis de condição.

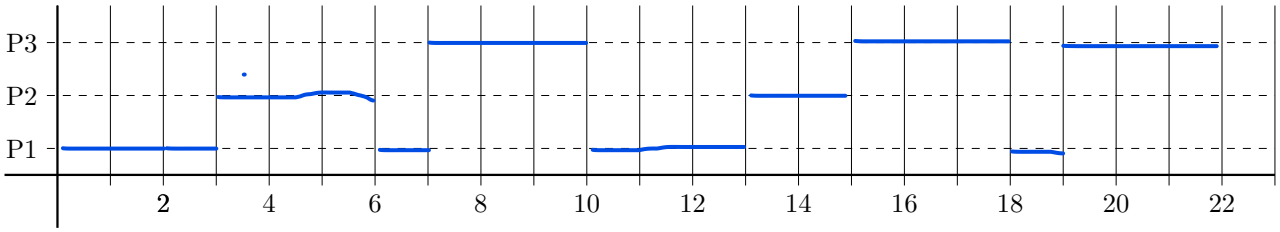
Nos semáforos tem de se por as condições, uma variavel condição. E um mutex que possa ser partilhado entre todos.

2. O gráfico seguinte representa o estado da execução de 3 processos independentes entre si (mesmo em termos de I/O), P1, P2 e P3, assumindo que correm em processadores (virtuais) distintos. R e B indicam, respetivamente, que o processo está no estado `RUN` (a usar o processador) ou no estado `BLOCKED` (bloqueado à espera de um evento).



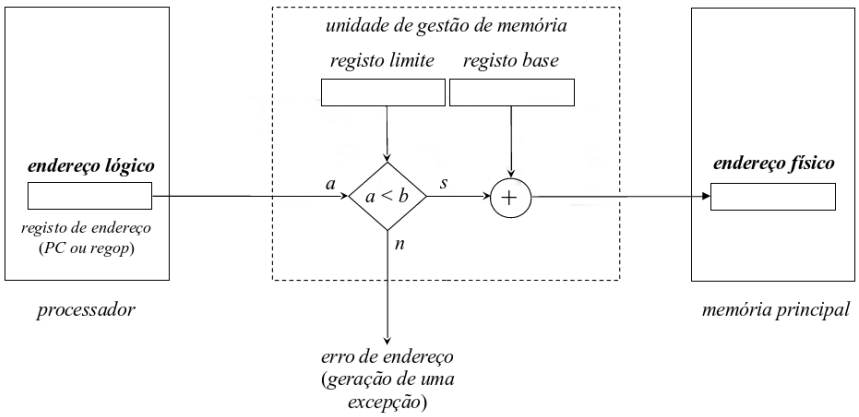
(a) Um escalonador de processador de baixo nível (short-term scheduler) típico possui 3 estados, normalmente designados por `RUN`, `READY-TO-RUN` e `BLOCKED`. Trace o diagrama de estados para um escalonador de baixo nível, considerando os estados anteriores e que se trata de um sistema *preemptive* e com prioridades. Para cada transição considerada, explique o seu papel e em que circunstâncias ocorre.

(b) Considere que os 3 processos representados acima correm num ambiente multiprogramado monoprocessador. Usando o gráfico abaixo, trace o diagrama temporal de escalonamento do processador pelos processos P1, P2 e P3, considerando uma política de escalonamento Round Robin sem prioridades e com um *time quantum* (*time slot* atribuído a cada processo) de 3.



(c) Num sistema *batch*, o tempo de *turnaround* corresponde ao intervalo de tempo entre a submissão de uma tarefa (*job*) e a sua conclusão, incluindo os tempos de espera por recursos. Considerando que os 3 processos representados acima correm num sistema *batch* multiprogramado (não *preemptive*), usando uma disciplina de seleção FCFS (First Come First Served), calcule o tempo de *turnaround* dos processos P1, P2 e P3. Considere que o intervalo de tempo em que o processo P1 está bloqueado é de 0,5 unidades e que cada intervalo de tempo em que o processo P3 está bloqueado é de 0,2 unidades.

3. A figura seguinte representa a unidade de tradução do espaço de endereçamento lógico de um processo para endereços físicos, usado num sistema com organização de memória real.



(a) Caracterize uma arquitetura com organização de memória real e compare as duas abordagens para gestão dessa memória, partições fixas e partições variáveis.

(b) Atendendo à figura,

i. descreva os papeis dos registos base e limite;

ii. diga em que operação do escalonador do processador é que estes registos são alterados;

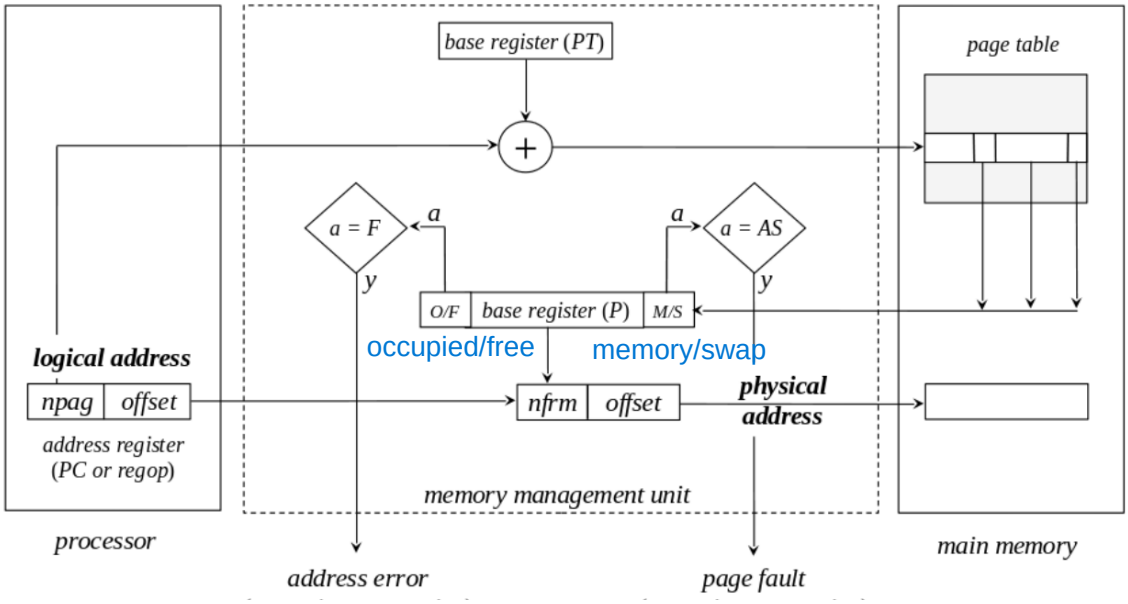
iii. descreva o procedimento de tradução de um endereço lógico num endereço físico.

(c) Considerando uma arquitetura de partições variáveis, considere que a memória real tem 200000 unidades de memória, das quais as primeiras 10000 são reservadas para o *kernel* do sistema de operação. Partindo da situação inicial (nenhum processo está alocado em memória), 4 processos (A, B, C e D) entram em jogo da seguinte forma:

- A, usando 10000 unidades de memória, é alocado;
- B, usando 40000 unidades de memória, é alocado;
- C, usando 20000 unidades de memória, é alocado;
- B, sai, libertando a memória que usava;
- D, usando 20000 unidades de memória, é alocado;
- A, sai, libertando a memória que usava;

Considerando que a política de alocação usada é a *worst fit*, e que o estado da memória é representado por duas listas ligadas simples (uma de blocos livres e outra de blocos ocupados), mostre o estado destas listas após a sequência de ações anterior. Considere que cada bloco é caracterizado pelo PID do processo a que foi atribuído (se estiver ocupado), pelo endereço inicial e pelo tamanho em número de unidades de memória.

4. A figura seguinte ilustra o processo de tradução do espaço de endereçamento lógico de um processo para endereços físicos, usado num sistema com organização de memória virtual com arquitetura paginada.



(a) Na figura acima, o bloco do meio representa uma peça de hardware (MMU) que faz o acoplamento entre o barramento (bus) de dados da CPU e o barramento de dados da memória principal, sendo uma peça essencial na implementação da organização de memória virtual.

- descreva os papéis dos registos base PT e P;
- diga em que circunstâncias é que estes registos são alterados;
- descreva o procedimento de tradução de um endereço lógico que conduz a uma falha de página (*page fault*).

(b) Considere uma hipotética implementação de um sistema de gestão de memória em que a cada processo são apenas atribuídos 5 *frames* de memória (F1..F5). A primeira linha da tabela seguinte representa, ao longo do tempo, as páginas a que um determinado processo acede. Preencha a linhas F1 a F5, indicando, ao longo do tempo, para cada *frame*, que páginas lá serão colocadas, considerando que o algoritmo de substituição de páginas utilizado é o LRU (Least Recently Used). Pode preencher apenas as células da tabela em que há mudança de página.

	1	2	1	2	3	1	4	5	4	6	7	1	8	9	2	7	8	9	5	10	9
F1	1																				
F2		2																			
F3																					
F4																					
F5																					

(c) Estudou concerteza outros algoritmos de substituição de páginas, que não o LRU. Escolha um deles e descreva o seu princípio de funcionamento.

5. Considere que 4 processos (P1, P2, P3 e P4) partilham recursos de 3 categorias diferentes (R1, R2 e R3). Os recursos são geridos por uma entidade que exige aos processos a declaração inicial das quantidades máximas de cada tipo de recurso que podem eventualmente necessitar. A seguir, os processos podem pedir recursos e a entidade gestora apenas os atribui se o sistema se mantiver, após a atribuição, num estado seguro. A tabela `Estado dos processos` ilustra as necessidades máximas de recursos declaradas pelos vários processos, os já adquiridos e os ainda por adquirir. A tabela `Recursos disponíveis` indica os recursos que a entidade de gestão ainda tem disponíveis para atribuição.

Estados dos processos

	Recursos declarados			Recursos já adquiridos			Recursos por adquirir		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	5	1	2	3	1	2	2	0	0
P2	2	2	2	2	0	0	0	2	2
P3	4	2	0	1	1	0	3	1	0
P4	2	1	0	1	1	0	1	0	0

Recursos disponíveis

R1	R2	R3
1	1	1

(a) Estudou políticas de prevenção de *deadlock* em sentido estrito (*deadlock prevention*) e em sentido lato (*deadlock avoidance*). Em que categoria coloca o sistema apresentado acima? Justifique a sua resposta.

(b) Um sistema deste tipo pode encontrar-se nos estado *safe*, *unsafe* ou em *deadlock*. Mostre que na situação representada o sistema se encontra num estado *safe*, apresentando uma sequência de execução (incluindo os correspondentes estados do sistema) que o prove.

(c) Se o processo P3 pede um recurso do tipo R2, o sistema pode ou não atribuir-lho imediatamente? Justifique a sua resposta.

b)
O P4 é o único que pode ser atendido primeiro, pelos recursos que pede e os recursos disponíveis

1 1 1 -> 2 2 1 (recursos disponíveis + recursos já adquiridos)

P2, P3 : 2 2 1 -> 5 3 3

Não é obrigatório seguir esta linha, mas através desta é como é evitado o deadlock

Portanto através deste "caminho" é safe. "Se atingiu este estado, então é safe"

c) em re- dispo passamos a ter 1 0 1