Instituto Superior de Engenharia do Porto
Departamento de Engenharia Informática

SISTEMAS DE COMPUTADORES – 2014/2015 2ºSemestre
Época recurso

Sem consulta
Duração: 1 hora

Nome:_____ Nº _____

Folha de Respostas

NOTAS:

1. **Em todas as questões deverá assinalar apenas uma resposta.**
2. **Se a resposta assinalada for incorrecta sofrerá uma penalização de 1/3 da cotação da pergunta.**
3. **Apenas as respostas assinaladas na Folha de Respostas serão consideradas.**
4. **Devem ser entregues todas as folhas do exame.**

1 -	a) □	b) □	c) □	d) □		11 -	a) □	b) □	c) □	d) □
2 -	a) □	b) □	c) □	d) □		12 -	a) □	b) □	c) □	d) □
3 -	a) □	b) □	c) □	d) □		13 -	a) □	b) □	c) □	d) □
4 -	a) □	b) □	c) □	d) □		14 -	a) □	b) □	c) □	d) □
5 -	a) □	b) □	c) □	d) □		15 -	a) □	b) □	c) □	d) □
6 -	a) □	b) □	c) □	d) □		16 -	a) □	b) □	c) □	d) □
7 -	a) □	b) □	c) □	d) □		17 -	a) □	b) □	c) □	d) □
8 -	a) □	b) □	c) □	d) □		18 -	a) □	b) □	c) □	d) □
9 -	a) □	b) □	c) □	d) □		19 -	a) □	b) □	c) □	d) □
10 -	a) □	b) □	c) □	d) □		20 -	a) □	b) □	c) □	d) □

1 – Nos sistemas operativos Windows e Linux as chamadas ao sistema (*system calls*):

a) São invocadas e executadas em *user-space*, permitindo a um processo aceder aos serviços disponibilizados pelo sistema operativo.
b) São invocadas em *user-space* e executadas em *kernel-space*, permitindo a um processo aceder aos serviços disponibilizados pelo sistema operativo.
c) Não são usadas porque os processos em *user-space* podem aceder directamente aos serviços disponibilizados pelo sistema operativo.
d) Nenhuma das anteriores.

2 – Um sistema operativo é normalmente grande e complexo, sendo por isso construído através de um conjunto de componentes. A estruturação com base numa abordagem monolítica pura:

a) Agrega todos os componentes num único processo que corre num único espaço de endereçamento.
b) Executa alguns componentes críticos em *user space* por uma questão de performance.
c) Exige um mecanismo de troca de mensagens entre *user space* e *kernel space*.
d) Permite carregar módulos dinamicamente, evitando a recompilação do núcleo quando se adicionam novas funcionalidades.

3 – As mudanças de estado de um processo são determinadas quer pelo seu fluxo de execução, quer pelo escalonador do sistema operativo. Qual das seguintes transições entre estados é possível?

a) "Bloqueado" para "em execução".
b) "Em execução" para "pronto a executar".
c) "Pronto a executar" para "bloqueado".
d) "Bloqueado" para "Terminado".

4 – Durante a execução de um processo qual das seguintes situações pode ocorrer?

a) O processo bloqueia usando um mecanismo de espera activa pelo acesso a um recurso, liberta o CPU e é colocado numa fila de espera associada ao recurso.
b) O tempo que o escalonador tinha atribuído ao processo (*time quantum*) termina, o processo liberta o CPU e mantém-se no estado "em execução".
c) O tempo que o escalonador tinha atribuído ao processo (*time quantum*) termina, o processo liberta o CPU e passa para o estado "pronto a executar".
d) O processo cria um novo processo, ficando de seguida à espera que o seu filho termine invocando a função *wait,* não libertando o CPU e mantendo-se em execução.

5 – A técnica de memória virtual agrega recursos de *hardware* e *software* com três funções básicas: realocação, protecção e paginação. A função de protecção:

a) Delega nos processos o mapeamento de endereços virtuais em endereços físicos.
b) Permite que um processo use mais memória do que a RAM fisicamente existente.
c) Usa registos do CPU para impedir que um processo utilize um endereço que não lhe pertence.
d) Assegura que cada processo tem o seu próprio espaço de endereçamento contínuo que começa no endereço 0.

6 – A gestão do acesso e reserva de memória pelos processos em execução:

a) Não é necessária num sistema operativo que permita a execução concorrente de processos.
b) É substituída pelo conceito de memória virtual em Linux e Windows.
c) É crítica em qualquer sistema operativo que permita a execução concorrente de processos.
d) Obriga os processos a indicarem endereços físicos quando pretendem aceder à memória.

7 – A comunicação entre processos permite:

a) A troca de dados e a sincronização de acções apenas quando os processos partilham o mesmo espaço de endereçamento.
b) A troca de dados e a sincronização das acções mesmo sem partilha do mesmo espaço de endereçamento.
c) Apenas a troca de dados e não a sincronização das suas acções, com ou sem partilha de espaço de endereçamento.
d) Apenas a sincronização das acções mas não a troca de dados, com ou sem partilha de espaço de endereçamento.

8 – A utilidade do uso de sinais como método de comunicação entre processos é limitada porque:

a) Só podem ser usados entre as várias *threads* do mesmo processo.
b) Funcionam de forma assíncrona.
c) Não é possível associar mais dados ao sinal para além do seu número.
d) Um processo tem sempre a opção de ignorar a recepção de todos os sinais.

9 – Uma solução eficiente para o problema da secção crítica deve garantir:

a) Diferentes prioridades para os processos, envelhecimento, ausência de interbloqueio (*deadlock*).
b) Acesso exclusivo, preempção, progressão.
c) Acesso exclusivo, progressão, espera limitada.
d) Ausência de preempção, *starvation*, inversão de prioridades.

10 – Os mecanismos de sincronização de processos são implementados:

a) Apenas em *software*.
b) Apenas em *hardware*.
c) Tanto em *software* como em *hardware*.
d) Apenas para garantir exclusão mútua.

11 – Em sistemas concorrentes, *resource starvation* é uma situação em que:

a) Dois processos (P1 e P2) se bloqueiam mutuamente devido a P1 ter bloqueado o semáforo S1, P2 ter bloqueado o semáforo S2, P1 necessitar de aceder a uma zona crítica protegida por S2 (sem libertar S1) e P2 necessitar de aceder a uma zona crítica protegida por S1 (sem libertar S2).
b) Dois processos (P1 e P2) se bloqueiam mutuamente devido a P1 ter bloqueado o semáforo S1, P2 ter bloqueado o semáforo S2, P1 necessitar de aceder a uma zona crítica protegida por S2 (depois de libertar S1) e P2 necessitar de aceder a uma zona crítica protegida por S1 (depois de libertar S2).
c) Em consequência da política de escalonamento do CPU, um recurso passa alternadamente dum processo P1 para um outro processo P2, deixando um terceiro processo P3 indefinidamente bloqueado sem acesso ao recurso.
d) Nenhuma das anteriores.

12 – Decidir o número de semáforos necessários para uma correcta sincronização de um conjunto de processos pode ser um exercício difícil. A abordagem de granularidade abrangente tem como consequência:

a) Diminuir o grau de concorrência das aplicações e o *overhead* do protocolo de sincronização.
b) Aumentar o grau de concorrência das aplicações e o *overhead* do protocolo de sincronização.
c) Diminuir o grau de concorrência das aplicações, mas aumentar o *overhead* do protocolo de sincronização.
d) Um maior número de dependências entre os semáforos, por vezes subtis, que podem originar *deadlocks*.

13 – Considere um processo com diversas *threads*. Qual das seguintes secções do espaço de endereçamento de um processo é privada para cada uma das *threads* desse processo:

a) *Heap*.
b) *Stack*.
c) Variáveis globais.
d) Código do programa.

14 – Considere o código seguinte, usado iterativamente pelos processos P1 e P2 para acederem à suas secções críticas sempre que necessário. Os processos partilham o acesso às variáveis inteiras *turn* e *flag[2]* numa zona de memória partilhada apontada por *shm*.

P1	**P2**

shm->flag[0] = 1;
shm->turn = 1;
while(shm->flag[1]==1 && shm->turn == 1);
/* Executa secção crítica */
shm->flag[0] = 0;
	shm->flag[1] = 1;
	shm->turn = 0;
	while(shm->flag[0]==1 && shm->turn == 0);
	/* Executa secção crítica */
	shm->flag[1] = 0;

Quais das seguintes afirmações descreve as propriedades conseguidas pela solução?

a) Exclusão mútua, progressão, mas não espera limitada.
b) Progressão, mas não exclusão mútua nem espera limitada.
c) Nem exclusão mútua, nem progressão, nem espera limitada.
d) Exclusão mútua, progressão e espera limitada.

15 – Considere o pseudo-código seguinte, referente aos processos P1 e P2 que executam num único processador. Assuma que existem dois semáforos (S1, S2), em que S1 é inicializado a zero (0) e S2 é inicializado a um (1), e que as funções *up(s)* e *down(s)* permitem incrementar e decrementar um semáforo, respetivamente.

P1	**P2**

…
down(S2);
/* Executa bloco A */
up(S2);
up(S1);
/* Executa bloco C */
	…
	down(S2);
	/* Executa bloco B */
	up(S2);
	down(S1);
	/* Executa bloco D */

Indique qual das seguintes afirmações é verdadeira:

a) P1 executa sempre o bloco C antes de P2 executar o bloco D.
b) P1 nunca executa o bloco C antes de P2 executar o bloco D.
c) Nada pode ser garantido em relação à ordem de execução dos blocos A e B.
d) P2 executa sempre o bloco B antes de P1 executar o bloco A.

16 – A estratégia de escalonamento que permite que qualquer processo em execução monopolize o CPU até ao fim do seu código ou até o libertar voluntariamente é denominada por:

a) Escalonamento não preemptivo.
b) Escalonamento preemptivo.
c) Escalonamento por prioridades fixas.
d) Escalonamento por prioridades dinâmicas.

17 – Aplicar ao algoritmo de escalonamento *round robin*:

a) Um *time quantum* muito pequeno converte-o na prática no algoritmo *First come First served*.
b) Um *time quantum* muito pequeno aumenta consideravelmente a performance do sistema.
c) Um *time quantum* muito pequeno aumenta consideravelmente o *overhead*.
d) Um *time quantum* muito pequeno converte-o na prática no algoritmo *Shortest Job First*.

18 – Considere um sistema com um único processador e um algoritmo de escalonamento *round robin* com um ***time quantum* igual a 2**. Considerando os seguintes tempos de chegada ao sistema e perfis de execução para os processos P1, P2 e P3

Processo	Perfil do processo	Tempo de chegada
P1	111I1	2
P2	2222I2	1
P3	333I33	0

em que um 1, 2 ou 3 representa, respectivamente, o processo P1, P2 ou P3 em execução durante uma unidade de tempo e I representa o bloqueio do processo em I/O, usando espera passiva. Indique a sequência de execução destes processos, sabendo que na solução o símbolo "–" significa que o processador não está a executar qualquer processo.

a) 11223312231233
b) 33221132213321
c) 3322113-221-33-21
d) 32111-1222-233-33

19 – Assuma o mesmo conjunto de processos, respetivos perfis de execução e tempos de chegada, mas um algoritmo de **escalonamento preemptivo de prioridades fixas**, em que os processos têm prioridades (P1=1 (mais alta); P2=2; P3=3) e usam espera passiva para aceder aos recursos. Indique a sequência de escalonamento para estes processos (note-se que o símbolo "–" significa que o processador não está a executar qualquer processo.

a) 32111-1222-233-33
b) 321112122323-33
c) 11223312231233
d) 33221132213321

20 – Considere o seguinte código:

Escritor

```
1. void write(int elem){
2.   pthread_mutex_lock(&buffer_mux);
3.   while(n_elems == MAX)
4.     pthread_cond_wait(&not_full, &buffer_mux);
5.   buffer[write_pos++] = elem;
6.   …
7.   pthread_mutex_unlock(&buffer_mux);
8. }
```

Leitor

```
1. void read(int &elem){
2.   pthread_mutex_lock(&buffer_mux);
3.   …
4.   n_elem--;
5.   if(n_elem == MAX – 1)
6.     pthread_cond_broadcast(&not_full);
7.   pthread_mutex_unlock(&buffer_mux);
8. }
```

Admita que existem duas *threads* escritoras bloqueadas na linha 4 e uma *thread* leitora a executar a linha 6.

a) A correcção das escritas não é garantida porque ambas as *threads* escritoras acordam ao mesmo tempo e executam simultaneamente a sua linha 5.
b) A correcção das escritas é garantida porque, apesar de ambas acordarem ao mesmo tempo, apenas uma *thread* escritora avança de cada vez para a sua linha 5.
c) A correcção das escritas é garantida porque a *thread* leitora apenas consegue acordar uma das *threads* escritoras.
d) Não é possível ter duas *threads* escritoras simultaneamente na linha 4 porque a segunda *thread* só passa da linha 2 quando a anterior executar a sua linha 7.