

SISTEMAS DE COMPUTADORES – 2014/2015 2ºSemestre
Época normal

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) ☐

2 - a) ☐ b) ☐ c) ☐ d) ☐

3 - a) ☐ b) ☐ c) ☐ d) ☐

4 - a) ☐ b) ☐ c) ☐ d) ☐

5 - a) ☐ b) ☐ c) ☐ d) ☐

6 - a) ☐ b) ☐ c) ☐ d) ☐

7 - a) ☐ b) ☐ c) ☐ d) ☐

8 - a) ☐ b) ☐ c) ☐ d) ☐

9 - a) ☐ b) ☐ c) ☐ d) ☐

10 - a) ☐ b) ☐ c) ☐ d) ☐

11 - a) ☐ b) ☐ c) ☐ d) ☐

12 - a) ☐ b) ☐ c) ☐ d) ☐

13 - a) ☐ b) ☐ c) ☐ d) ☐

14 - a) ☐ b) ☐ c) ☐ d) ☐

15 - a) ☐ b) ☐ c) ☐ d) ☐

16 - a) ☐ b) ☐ c) ☐ d) ☐

17 - a) ☐ b) ☐ c) ☐ d) ☐

18 - a) ☐ b) ☐ c) ☐ d) ☐

19 - a) ☐ b) ☐ c) ☐ d) ☐

20 - a) ☐ b) ☐ c) ☐ d) ☐

1 – O que é um sistema operativo?

- a) Um gestor de recursos de *hardware*.
- b) Um controlador da execução dos programas de modo a prevenir erros e um uso inadequado do *hardware*.
- c) Um intermediário entre os utilizadores e o *hardware*.
- d) Todos os 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 micro-núcleo pura:

- a) Agrega todos os componentes num único processo que corre num único espaço de endereçamento.
- b) Executa os 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 de um processo não é possível?

- a) “Bloqueado” para “em execução”.
- b) “Pronto a executar” para “em execução”.
- c) “Bloqueado” para “pronto a executar”.
- d) “Em execução” para “bloqueado”.

4 – Os sistemas operativos que suportam a execução concorrente/paralela de processos permitem num sistema com um único processador:

- a) A existência de vários processos em estado de execução.
- b) A existência de vários processos relativos ao mesmo programa executável.
- c) A execução simultânea de mais do que uma instrução do mesmo programa.
- d) Nenhuma das anteriores.

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

- 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) Impede 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 – Assuma que vários processos acedem e manipulam os mesmos dados de forma concorrente e o resultado final depende da ordem particular em que os acessos têm lugar. A este cenário é dado o nome de:

- a) Comunicação entre processos através de memória partilhada.
- b) Sincronização de processos.
- c) Condição de concorrência (*race condition*).
- d) Interbloqueio (*deadlock*).

7 – A exclusão mútua implica que:

- a) Se um processo está a executar a sua zona crítica, então nenhum outro processo pode estar a executar a sua zona crítica.
- b) Se um processo está a executar a sua zona crítica, todos os outros processos têm de estar simultaneamente a executar as suas zonas críticas.
- c) Se um processo está a executar a sua zona crítica, então todos os recursos do sistema têm de estar bloqueados até que o processo termine a sua execução da zona crítica.
- d) Nenhuma das anteriores.

8 – 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.

9 – Interbloqueio (*deadlock*) 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.

10 – A estratégia de escalonamento que força a suspensão temporária de um processo em execução é denominada:

- a) Escalonamento não preemptivo.
- b) Escalonamento preemptivo.
- c) *Shortest job first.*
- d) *First come First served.*

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

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

12 – Considere um sistema com um único processador cujo sistema operativo utiliza um algoritmo de escalonamento *round robin* com um ***time quantum* igual a 3**. 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	11I1	2
P2	22I2I2	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 activa.

Indique a sequência de execução destes processos, sabendo que na solução o símbolo “-” significa que o processador não está a executar qualquer processo.

- a) 33322113312-2
- b) 112233312332
- c) 333-3322-2-211-1
- d) 33322-11-332-21

13 – 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).

Indique a sequência de escalonamento para estes processos (note-se que o símbolo “-” significa que o processador não está a executar qualquer processo).

- a) 333112212332
- b) 3211212323-33
- c) 1121232323-33
- d) 11-122-2-2333-33

14 – Considere o pseudo-código seguinte, referente aos processos P1 e P2. 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);	down(S2);
/* Executa bloco A */	/* Executa bloco B */
up(S2);	up(S2);
down(S1);	/* Executa bloco D */
/* Executa bloco C */	up(S1);

Indique qual das situações pode ocorrer:

- a) Existe a possibilidade de P1 executar o bloco C antes de P2 executar o bloco D.
- b) P2 pode ficar indefinidamente bloqueado.
- c) O processo P1 nunca executa o bloco C antes de P2 executar o bloco D.
- d) P1 pode ficar indefinidamente bloqueado.

15 – Considere o código seguinte, usado iterativamente pelos processos P1 e P2 para acederem à suas secções críticas sempre que necessário. O valor inicial das variáveis booleanas partilhadas S1 e S2 são inicializadas com valores aleatórios.

P1	P2
...	...
while(S1==S2);	while(S2!=S1);
/* Executa secção crítica */	/* Executa secção crítica */
S1 = !S2;	S2 = S1;
...	...

Quais das seguintes afirmações descreve as propriedades conseguidas pela solução?

- a) Exclusão mútua mas não progressão
- b) Progressão mas não exclusão mútua
- c) Nem exclusão mútua nem progressão
- d) Exclusão mútua e progressão

16 – Considere que vários processos partilham o acesso a um semáforo s , inicializado a 1 (um). Admita que um desses processos usa o semáforo s da seguinte forma, antes e depois de executar a sua secção crítica.

```
...
up(s);
/* Executa secção crítica */
down(s);
...
```

Nesta situação:

- a) Um interbloqueio (*deadlock*) pode ocorrer.
- b) O processo poderá ficar indefinidamente à espera de executar a sua secção crítica.
- c) Mais do que um processo poderá estar a executar a sua secção crítica.
- d) Todas as situações anteriores.

17 - Admita agora que um desses processos usa o semáforo s , inicializado a 1 (um), da seguinte forma, antes e depois de executar a sua secção crítica.

```
...
down(s);
/* Executa secção crítica */
down(s);
...
```

Nesta situação:

- a) Um interbloqueio (*deadlock*) pode ocorrer.
- b) O processo poderá ficar indefinidamente à espera de executar a sua secção crítica.
- c) Mais do que um processo poderá estar a executar a sua secção crítica.
- d) Todas as situações anteriores.

18 – Considere o seguinte programa:

```
for (i = 0; i < 3; i++) {
    pid = fork();
    if (pid > 0) {
        pid = fork();
        printf("S1\n");
    }
    if (pid==0){
        execlp("ls", "ls", NULL);
    }
    printf("S2\n");
}
```

Quantas vezes irá ser impresso "S1"? Assuma que a invocação das funções nunca falha.

- a) 4
- b) 6
- c) 2
- d) 3

19 – O código seguinte apresenta uma solução para o problema dos “Leitores/Escritores” em que o valor inicial de *wrt*=1, *mutex*=1 e *readcount* = 0.

Escritor	Leitor
1. down(wrt);	1. down(mutex);
2. ...	2. readcount++;
3. /* Escreve dados */	3. if (readcount == 1)
4. ...	4. down(wrt);
5. up(wrt);	5. up(mutex);
	6. ...
	7. /* Lê dados */
	8. ...
	9. down(mutex);
	10. readcount--;
	11. if (readcount == 0)
	12. up(wrt);
	13. up(mutex);

Assuma que existe um leitor bloqueado na linha 1 e outro na linha 4. Consequentemente:

- Existe apenas um escritor a aceder à sua zona crítica porque a prioridade é dada aos escritores.
- b) Existe apenas um escritor a aceder à sua zona crítica, apesar da prioridade ser dada aos leitores.
- Existe apenas um escritor a aceder à sua zona crítica e um leitor a executar código entre as linhas 10 e 12.
- O caso apresentado nunca pode acontecer.

20 – Considere os seguintes processos P1 e P2. Assumindo que existem dois semáforos (S1, S2) partilhados entre eles, em que S1 é inicializado a 0 (zero) e S2 é inicializado a 1 (um). As funções *up(s)* e *down(s)* permitem incrementar e decrementar um semáforo, respetivamente.

P1	P2
down(S1);	down(S2)
printf("O");	printf("S");
printf("M");	printf("C");
up(S1);	up(S2);
up(S2);	up(S1);
	down(S2);
	printf("P");

Qual será o resultado da execução dois processos em simultâneo?

- Imprimiria sempre SCOMP.
- Imprimiria sempre OMSCP.
- Imprimiria sempre SCPOM.
- d) É impossível prever o resultado.