

SISTEMAS DE COMPUTADORES – 2024/2025
Exame Época Especial

Versão A

Apenas autorizada a consulta da folha oficial
Duração: 2h

Nome: _____ Nº _____

Grupo I
(10/20 valores)

NOTAS:

- 1. Em todas as questões deverá assinalar apenas uma resposta.**
- 2. Se a resposta assinalada for incorreta sofrerá uma penalização de 1/3 da cotação da pergunta.**
- 3. Devem ser entregues todas as folhas do exame.**

1 – Se um processo filho modificar o valor de uma variável num programa em C, qual das seguintes afirmações descreve o seu efeito?

- a) A modificação afetará apenas o processo filho.
- b) A modificação afetará tanto o processo filho como o processo pai.
- c) A modificação apenas afetará o processo pai se a variável for global.
- d) A modificação não terá nenhum efeito porque as variáveis não podem ser modificadas após a criação de um processo filho.

2 – Com a execução concorrente de processos num sistema com um único CPU podemos ter:

- a) Um processo em execução, enquanto outro está bloqueado numa operação de sincronização usando um mecanismo de espera ativa.
- b) Um processo em execução, enquanto outros estão bloqueados numa operação de sincronização usando um mecanismo de espera passiva.
- c) Mais do que um processo em execução simultânea, quer existam ou não processos bloqueados em operações de sincronização usando mecanismos de espera passiva.
- d) Apenas um processo bloqueado em operações de sincronização usando mecanismos de espera passiva, quer existam ou não processos em execução.

3 – Num sistema operativo com paginação, um endereço virtual usado numa instrução de acesso à memória:

- a) Corresponde à posição exata da instrução na memória física, determinada no momento do carregamento do programa para a memória.
- b) É um índice para uma tabela de páginas, que contém os endereços físicos das páginas correspondentes.
- c) É um deslocamento (*offset*) dentro de uma página, combinado com o número da página, que é utilizado pela MMU para calcular o endereço físico.
- d) É um endereço virtual que é traduzido para um endereço físico pela MMU, mas não possui relação direta com a organização da memória física.

4 – Em programação concorrente, uma secção crítica é uma:

- a) Parte do programa em que são acedidos dados potencialmente partilhados por vários processos que devem ser alterados em exclusão mútua.
- b) Parte do programa em que o processo requisita ao sistema operativo a reserva de mais memória de forma dinâmica.
- c) Parte do programa em que um erro causa garantidamente o término do processo.
- d) Parte do programa que é executada em *kernel space*.

5 – Quando um processo é bloqueado ao invocar a função `sem_wait(&S)` num semáforo `S`, assumindo que `S` tinha o valor 0 antes da invocação, o processo fica no estado de:

- a) Pronto (*ready*).
- b) Em execução (*running*).
- c) Em espera (*waiting*) e liberta a CPU.
- d) Em espera (*waiting*), mas não liberta a CPU.

6 – No escalonamento de processos baseado em prioridades, a inversão de prioridades refere-se a:

- a) Uma redução da prioridade de um processo por este bloquear constantemente em operações de I/O.
- b) Um aumento da prioridade de um processo para que este liberte mais rapidamente um recurso requerido por um processo de maior prioridade.
- c) Uma situação em que um processo de maior prioridade é obrigado a esperar por um processo de menor prioridade.
- d) Uma atribuição de prioridades aos processos pela ordem inversa dos seus tempos de execução.

7 – Considerando as boas práticas de uso de sinais em programação concorrente, podemos afirmar que:

- a) Os sinais devem ser usados para sincronizar o acesso a variáveis partilhadas, substituindo completamente os mecanismos de exclusão mútua, como semáforos.
- b) Os *handlers* devem ser simples, evitando chamadas a funções não reentrantes e operações que possam levar a estados inconsistentes.
- c) Os sinais podem ser ignorados sem consequências em qualquer parte do código, pois o S.O. garante a integridade dos dados durante operações críticas.
- d) Os sinais são destinados exclusivamente à execução de operações de I/O, pois sua manipulação ocorre no mesmo contexto que a função que os gera, permitindo operações complexas sem restrições.

8 – Considere o seguinte excerto de código:

<pre>void *f1(void *arg){ printf("S1\n"); pthread_exit(NULL); }</pre>	<pre>for(i = 0; i < 3; i++){ pid = fork(); if (pid > 0){ execlp("ls", "ls", NULL); pid = fork(); } else{ pthread_create(&t[i], NULL, f1, NULL); printf("S1\n"); } printf("S2\n"); }</pre>
---	---

Quantas vezes irá ser impresso “S1”? Assuma que a invocação da função `exec1p()` nunca falha.

- a) 18.
- b) 6.
- c) 36.
- d) 3.

9 – Considere os seguintes processos P1 e P2 que executam num único processador. Assuma que existem dois semáforos (S1, S2), em que S1 é inicializado a um (1) e S2 é inicializado a zero (0), e que as funções `up(s)` e `down(s)` permitem incrementar e decrementar um semáforo, respetivamente, de forma atômica.

P1	P2
...	...
<code>down(S1);</code>	<code>down(S2);</code>
<code>up(S2);</code>	<code>/* Executa bloco B */</code>
<code>/* Executa bloco A */</code>	<code>down(S1);</code>
<code>up(S1);</code>	<code>up(S2);</code>
<code>/* Executa bloco C */</code>	<code>/* Executa bloco D */</code>

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 C e D.
- d) P2 executa sempre o bloco B antes de P1 executar o bloco A.

10 – Considere um sistema com um único processador e um algoritmo de **escalonamento preemptivo de prioridades fixas**, em que os processos têm as seguintes prioridades: P1=1 (mais alta); P2=3 (mais baixa); P3=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, respetivamente, o processo P1, P2 ou P3 em execução durante uma unidade de tempo e I representa o bloqueio do processo em I/O. Indique a sequência de execução destes processos, sabendo que na solução o símbolo “-” significa que o processador não está a executar qualquer processo.

- a) 33111-133-332222-2
- b) 32111212323-333
- c) 3311131332222-2
- d) 33322211133321

Grupo II (10/20 valores)

NOTAS:

1. Responder cada questão numa folha separada, devidamente identificada.
2. Não é necessário indicar o nome das bibliotecas usadas na resolução.

1) Considere um museu que **possui N scanners de bilhetes** distribuídos pelas suas galerias. Sempre que um visitante digitaliza o bilhete num destes scanners, devem ser apresentadas no ecrã as informações do bilhete e a sua validade. Simule este sistema utilizando processos e pipes:

- O processo pai mantém a base de dados com as informações dos bilhetes (pode assumir que a estrutura de dados que decidir usar já está preenchida com essa informação).
- Cada processo filho representa um scanner de bilhetes.
- Existe um **pipe partilhado por todos os N processos filho**, onde cada um solicita as informações relativas a um bilhete específico.
- O processo-pai responde ao processo filho solicitante através de **um pipe que partilha unicamente com esse scanner**, enviando assim os dados do bilhete pretendido.

2) Implemente um programa em C que, através do uso de *threads*, implemente um cenário de múltiplos escritores e múltiplos leitores. Haverá N escritores e M leitores (M e N podem ser estáticos ou definidos pelo utilizador).

Num acesso concorrente a uma área de memória partilhada, é fundamental garantir a sincronização adequada entre as operações de leitura e escrita:

- **leitor** – responsável por ler duas *strings* de uma estrutura presente na *heap*:
 - Os leitores não modificam a área de memória partilhada, permitindo que **vários leitores a acessem simultaneamente**.
 - Os **leitores têm prioridade** sobre os escritores.
 - Cada leitor deve imprimir as *strings* lidas, juntamente com a contagem atual de leitores ativos.
- **escritor** – responsável por escrever na área de memória partilhada na *heap*:
 - Escreve o ID da *thread* e a hora atual.
 - Apenas **um escritor** pode aceder à área de memória partilhada em qualquer momento.
 - Os escritores só podem aceder à área de memória partilhada **quando não há leitores ativos**.
 - Cada escritor imprime a contagem de escritores, bem como a contagem de leitores ativos.