



Nome: Tiago Jorge Coimbra da Silva

Nº aluno: 2022216215

- Qualquer tentativa de fraude conduzirá à anulação da prova para todos os intervenientes.
- Consulta apenas em papel. Durante o exame todos os dispositivos eletrónicos têm que permanecer desligados, com exceção de calculadoras simples.
- Todas as respostas devem ser diretas, objetivas e obrigatoriamente efetuadas na folha fornecida.

1. Diga o que entende por espera ativa quando usada numa solução para o problema da secção crítica. Esta solução pode resolver o problema? Se sim, deve ser usada?

Espera ativa é a consequência de programar com recurso a estruturas de controlo que promovem o uso excessivo de CPU. Não devem ser usados uma vez que existem formas mais inteligentes, seguras e eficientes de resolver o mesmo problema sem esgotar os recursos do computador.

2. Explique em que consiste o *copy-on-write* e de que forma pode otimizar a criação de processos. No Linux, a vantagem do *copy-on-write* mantém-se mesmo que após o `fork()` o utilizador faz logo um `exec()`?

O *copy-on-write* é uma técnica que consiste em criar a cópia de recursos do processo pai para os filhos até que seja absolutamente necessário. No Linux a instrução `exec()` imediatamente a seguir ao `fork()` implica a substituição do espaço de memória do filho com um novo programa. Contudo, existem ainda algumas razões pelas quais se justifica usar COW: janela de tempo entre o `fork()` e o `exec()`; otimização intrínseca do `fork()` diminuindo o overhead na criação de processos.

3. Que vantagens e desvantagens existem no uso de uma tabela de páginas invertida?

Neste modo de paginação há apenas 1 tabela de páginas comum a todos os processos do SO. Dessa forma diminui o espaço de memória necessária para a tabela de páginas porém aumenta o tempo necessário para ver a tabela quando uma página é referenciada. Para além disso, desta forma é promovida a unificação da informação. Contudo, as colissões no hashing e a complexidade de implementação são algumas desvantagens deste modo

$$\begin{array}{r} 32 \\ -13 \\ \hline 19 \end{array}$$

4. Acha que é possível existir *Thrashing* num sistema operativo que tenha poucos processos a executar? Justifique.

Sim, é possível a ocorrência de *Thrashing* num sistema operativo com poucos processos a executar. Isto ocorre se:

- Os processos tiverem um *working set* tão grande que ultrapassa a memória física disponível e ocorrerem muitos *page faults*.
- A memória física for muito pequena.
- Acesso a memória aleatória que não respeite os princípios de localidade temporal.

5. Considere que tem um sistema de memória paginada hierárquico, com 2 níveis, com um espaço de endereçamento virtual de 4GByte (2^{32} bytes). Sabendo que cada página de memória ocupa 8KBytes e que cada PTE ocupa 64 bits, responda as seguintes questões:

$$2^6 / 2^3 = 2^3 \text{ bytes} \quad 2^{13}$$

- 5.1. Se cada tabela de páginas de 2.º nível tiver de caber numa única página de memória, qual a divisão de bits no endereço lógico? Apresente os cálculos que realizar.

P_1	P_2	offset
9	10	13

$$\frac{2^{13}}{2^3} = 2^{10}$$

numero de endereços possíveis

- 5.2. Qual o tamanho ocupado pela tabela de páginas de 1.º nível?

$$2^9 \times 2^3 = 2^{12} \text{ bytes} = 4 \text{ KB} \rightarrow \text{meia página}$$

- 5.3. No total, quantas páginas de 8KB são ocupadas por todas as tabelas de páginas (1.º e 2.º nível)?

$$2^{10} \times 2^3 = 2^{13} \text{ } \left. \begin{array}{l} \text{Cada tabela de páginas 2º nível} \\ 2^9 \times 2^{13} = 2^{22} = 4096 \text{ KB} = 512 \end{array} \right\}$$

6. Considere um sistema com 5 processos em execução (P1 a P5) e 3 tipos de recursos (R1 a R3). O número máximo de recursos de cada tipo existentes no sistema é: R1=9, R2=2, R3=9 (NOTA: este não é o número dos que estão disponíveis nesse momento, mas sim os que o sistema tem!). Num determinado momento o sistema está no seguinte estado:

	Recursos em uso Pelos processos			Recursos máximos que os processos precisam		
	R1	R2	R3	R1	R2	R3
P1	0	1	0	2	2	2
P2	2	0	0	3	2	6
P3	3	0	3	9	0	4
P4	2	1	1	2	1	2
P5	2	0	3	4	2	4

No total não
precisa 513 páginas

6.1. Prove que o sistema está num estado seguro (*safe state*).

NEED:					
	R ₁	R ₂	R ₃		
P ₁	2	1	2	✓	existing: [9, 2, 9] available: [0, 0, 2] → [2, 1, 3] → [2, 2, 3] → [4, 2, 6] → [6, 2, 6]
P ₂	4	2	6	✓	
P ₃	6	0	4	✓	
P ₄	0	0	1	✓	
P ₅	2	2	1	✓	

R: P₄, P₁, P₅, P₂, P₃

6.2. O processo P4 fez o seguinte pedido (R₁=0, R₂=0, R₃=1). Qual será a decisão do gestor de recursos se aplicar o algoritmo do banqueiro (*Banker's Algorithm*)? Apresente o processo que usou para dar a resposta.

ALDC

NEED:					
	R ₁	R ₂	R ₃		
P ₁	0	1	0	available [0, 0, 1] → [2, 1, 3] → [4, 2, 4] → [6, 2, 8]	R: P ₄ , P ₁ , P ₅ , P ₂ , P ₃
P ₂	2	0	0		
P ₃	3	0	3		
P ₄	2	1	2		
P ₅	2	0	3		

Safe State

7. Considere um sistema onde existem 8 páginas de processos (1 a 8) e 4 *page frames* em RAM. Vão ser feitos acessos à memória usando a *string* de referência seguinte:

W(1) ; W(3) ; R(2) ; R(3) ; R(8) ; W(5) ; R(3) ; R(8) ; R(1) ; W(6)

Supondo que inicialmente todas as *frames* estão vazias, que R(...) é uma operação de leitura, que W(...) é uma operação de escrita e que o sistema faz uso do *modify-bit*, preencha a tabela abaixo considerando os algoritmos pedidos:

	FIFO	LRU	CLOCK
Page-faults	7		
Swap-outs	2		
Estado final das <i>frames</i> (indique a <i>frame</i> , a página que cada <i>frame</i> contém e o <u>estado do modify bit</u>)			

Nota: no início, com as *frames* vazias, as páginas ocupam primeiro a *frame* 0, depois a 1 e assim sucessivamente.

W(2) R(1) W(1) W(3) R(4) R(2) W(1) W(3) W(2) R(4)

NOTA: Esta folha não é para entregar R(1) W(2) R(1)

Potência de 2

2 ⁶	2 ⁷	2 ⁸	2 ¹⁰	2 ¹²	2 ¹⁴	2 ¹⁶	2 ¹⁸	2 ²⁰	2 ³⁰
64	128	256	1024	4096	16384	65536	262144	1048576	1073741824

3 Page Frames

Grelha para algoritmos de substituição de páginas.

LRU	W2	R(1)	W1	W3	R4	R2	W1	W3	W2	R4	R1	W2	R1
	2M	2M	2M	2M	4	4	4	3M	3M	3M	1	1	1
		1	1M	1M	1M	2	2	2	2M	2M	2M	2M	2M
				3M	3M	3M	1M	1M	1M	4	4	4	4

Page Faults	1	2	2	3	4	5	6	7	7	8	9	9	9
Swap-outs	0	0	0	0	1	2	3	3	3	4	5	5	5

CLOCK W2 R1 W1 W3 (R4) (R2) (W1) (W3) (W2) (R4) (R1) (W2) (R1)

Page Faults													
Swap-outs													

Page Faults	1	1	1										
Swap-outs													