Número: Nome:

LEIC/LETI – 2018/19 - 2º Teste de Sistemas Operativos (Repescagem)

5 de fevereiro de 2019

Responda no enunciado, apenas no espaço fornecido. Identifique todas as folhas.

Duração: 1h30m

Grupo I [7,5 Val]

1. [4 val] Considere o seguinte excerto de um programa paralelo

```
fechar(m);
f();
abrir(m);
```

a) Considere as seguintes variantes:
Variante A: fechar/abrir são funções do próprio programa, que implementam um trinco com uma instrução atómica em modo utilizador, com espera ativa (tal como estudado nas aulas).
Variante B: fechar/abrir são funções de uma biblioteca do SO, que fazem chamadas sistema sobre um trinco mantido pelo núcleo.

Considere também que: o programa se executa numa máquina com um CPU single-core e muita contenção sobre a secção crítica apresentada acima.

i) [1 val] Testando com uma função f() demorada, cuja duração excede sempre a duração dos *quantums* atribuídos pelo escalonador deste sistema, a variante B conseguiu desempenho muito melhor que a variante A. Indique sucintamente uma razão.

Considerando que uma tarefa T1 obteve o trinco e chamou f(), T1 consumirá o seu quantum e poderá perder o processador para outra tarefa T2 que também tenta obter o trinco.
Na variante A, T2 manter-se-á em execução durante o seu quantum mesmo sem ter condições de avançar (espera ativa); na variante B, T2 é bloqueada mal tenta fechar o trinco, dessa forma permitindo que T1 receba execução e liberte a secção crítica mais cedo.

ii) [1 val] Testando com uma função f() muito curta, cuja duração é várias vezes inferior à duração do *quantum* mínimo atribuído pelo escalonador, deixou de se observar a vantagem de desempenho da variante B sobre a variante A. Indique sucintamente uma razão.

Como f() é muito curta, serão raras as situações em que T1 perde o processador dentro da secção crítica, logo o exemplo descrito na alínea anterior passa a ser raro.

b) A função f() é uma função auxiliar do mesmo programa cujo código se limita a aceder (lê/escreve) a variáveis em memória e executa instruções aritméticas; não chama qualquer função sistema.
Dê um exemplo de uma interrupção que pode ocorrer durante a execução de f():
i) [1 val] Causada devido ao código executado por f().

Excepção (e.g. divisão por zero, falta de página, violação de acesso, etc.)

ii) [1 val] Devido a causas externas ao código executado por f().

Interrupção proveniente do hardware (e.g. do temporizador ou de um periférico).

2. Considere que um processo P1 executa um programa que permite visualizar documentos *pdf* no ecrã. P1 já tem um documento pré-carregado totalmente em memória. P1 executa o seguinte ciclo principal:
   Para cada página do documento:
   1. *Apresenta no ecrã a página atual do pdf (demora 1 unidade de tempo, sempre em execução)*
   2. *Espera até o utilizador premir "enter" (bloqueado, tipicamente durante muitas unidades de tempo); avança para a próxima iteração deste ciclo.*

   a) [1 val] Comparando os algoritmos de escalonamento round-robin e preemptivo com prioridades dinâmicas, estudados nas aulas, qual aquele em que o utilizador sente que P1 tem um **tempo de resposta mais reduzido**?

   Preemptivo com prioridades dinâmicas

   b) [2,5 val] Preencha as grelhas abaixo apresentando um exemplo para cada escalonador, de forma a que esses exemplos ilustrem claramente a sua resposta à alínea anterior.

   Na sua resposta:
   - assuma uma máquina single-core
   - indique, para cada unidade de tempo, o estado de cada processo: E (em execução), P (executável), B (bloqueado)
   - tem liberdade para incluir outros processos que competem pelo mesmo CPU que P1, assim como decidir qual o comportamento desses outros processos (quantas unidades se executam, quantas se bloqueiam, etc)
   - por simplicidade, assuma que, sempre que P1 se bloqueia, fica nesse estado durante exatamente 4 unidades de tempo
   - em ambos os algoritmos, assuma quantum fixo de 5 unidades
   - tem também liberdade para definir parâmetros dos algoritmos de escalonamento que estejam omissos neste enunciado

Nota: diferentes respostas possíveis são aceites como corretas, desde que o escalonamento apresentado seja válido (por exemplo, nunca haja 2 processos em execução ao mesmo tempo; ou, caso haja pelo menos um processo executável, pelo menos um desses processos está em execução); e ilustre um caso em que, sempre que P1 se desbloqueia, o número de unidades de tempo que passam até P1 se executar de novo é menor com o escalonamento preemptivo com prioridades.
A resposta abaixo é aquela que, provavelmente, cumpre os critérios acima da forma mais simples: um processo P2 CPU-intensivo, logo menos prioritário que P1, corre na mesma máquina que P1.

Escalonamento com round-robin:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
P1	E	B	B	B	B	P	E	B	B	B	B	P	E	B	B	B	B	P	E	B	B	B	B
P2	P	E	E	E	E	E	P	E	E	E	E	E	P	E	E	E	E	E	P	E	E	E	E

Escalonamento preemptivo com prioridades dinâmicas:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
P1	E	B	B	B	B	E	B	B	B	B	E	B	B	B	B	E	B	B	B	B	E	B	B
P2	P	E	E	E	E	P	E	E	E	E	P	E	E	E	E	P	E	E	E	E	P	E	E

Grupo I [7,5 Val]

1. [2 val] Responda às seguintes perguntas a escolha múltipla. Cada resposta certa vale 0,5 valores. **NOTA:** Cada resposta errada implica uma **penalização de 0,25 valores**.

   a) Em sistemas baseados em paginação, ao aumentar o tamanho da página a fragmentação interna aumenta?

   SIM X NÃO ☐

   b) Em sistemas baseados em paginação, ao aumentar o tamanho da página a fragmentação externa aumenta?

   SIM ☐ NÃO X

   c) Em sistemas baseados em paginação, ao aumentar o tamanho das páginas reduz-se a latência caso ocorra uma falta de página.

   SIM ☐ NÃO X

   d) Em sistemas baseados em paginação, de cada vez que um processo acede a um endereço de memória, o **núcleo** do sistema operativo efetua a tradução do endereço virtual para o correspondente endereço físico.

   SIM ☐ NÃO X

2. [2 val] Considere um sistema de gestão de memória com 32 bits de espaço de endereçamento virtual, com páginas de 64KB e um processo P1 cuja tabela das páginas contém estas entradas:

Página	Presente	Protecção	Base
0	1	R	0x04 00 00 00
1	1	RW	0x05 00 00 00
2	1	R	0x06 00 00 00
3	0	RW	

Na tabela abaixo, indique qual é o endereço físico correspondente aos seguintes endereços virtuais, caso possível. Caso o acesso der origem a alguma exceção, indique também o tipo da exceção.
Assuma que, em caso de falta de páginas, o SO aloca tramas (page frames) livres a partir do endereço 0x07 00 00 00.

Acesso	End. Virtual	End. Físico	Eventuais exceções
Leitura	0x1000 0000	-	Página inválida
Escrita	0x0001 0000	0x05 00 00 00	-
Escrita	0x0002 FFFF	- (também foi considerada válida: 0x06 00 FF FF)	Violação de proteção
Leitura	0x0003 0011	0x07 00 00 11	Falta de página

3. Considere um sistema com 32 bits de espaço de endereçamento, páginas de 4KB e suporte para o mecanismo de *Copy on Write*.

a) [1 val] Assuma que um processo P1 aloca um vetor de inteiros que ocupa 40KB em memória, cujo endereço inicial é alinhado às páginas (i.e., o endereço do vetor coincide com o endereço inicial de uma página). Indique quais e quantos bits do endereço inicial do vetor têm garantidamente valor 0.

Os últimos 12 bits.

b) Assuma agora que o processo P1 executa fork(), criando um processo P2.
P2 começa por incrementar cada entrada da sua cópia do vetor e, pouco tempo depois, termina.
Após executar fork(), o processo P1 executa wait() e, a seguir, incrementa cada entrada da sua cópia do vetor.

i) [1 val] Observa-se que o tempo que P1 demora para incrementar as entradas do vetor (sem contar com o tempo em que está bloqueado no wait()), é significativamente menor do que no caso do P2.
Justifique sucintamente a razão subjacente a este fenómeno.

P2 executa primeiro e, durante a sua execução, o SO deverá copiar (`copy-on-write`) as páginas associadas ao vetor, cada vez que P2 escreve pela primeira vez sobre uma página. Dado que P1 executa depois do P2 ter já acedido todas as páginas do vetor, não paga este custo.

ii) [0,5 val] Quantas e que tipos de exceções relacionadas com acesso às páginas é garantido que ocorram durante os acessos de P1 e P2 ao vetor, acima descritos?

10 exceções, pois o array ocupa 10 páginas. Cada vez que P2 acede pela primeira vez a uma página é gerada uma exceção de proteção.

4. [1 val] Ilustre um cenário em que o TLB é atualizado pelo SO, e um cenário em que o TLB é atualizado autonomamente pelo hardware, sem nenhuma intervenção do SO

TLB atualizado pelo SO:

Quando o SO efetua um *context switch* que alterna processos diferentes à execução no mesmo processador.

TLB atualizado pelo hardware:

Cada vez que há um "miss" no TLB e o endereço físico é obtido (pelo hardware) através da tabela das páginas, o TLB também é atualizado.

Grupo III [5 Val]

Considere o seguinte programa:

```
int main(void)
{
    char buffer[N];
    int total = 0; int n;
    int fd = open("/home/andre/testfile.txt", O_RDONLY);

    while (n = read(fd, buffer, N)) {
       total += n;
       processaBuffer(buffer);
    }

    printf("total=%d\n", total);
    return 0;
}
```

1. [1 val] Assumindo que se desativam as caches usadas pelo sistema de ficheiros, quantos i-nodes têm de ser lidos do disco para completar a chamada *open("/home/andre/testfile.txt", ...)*?

4 i-nodes (i-node de cada diretoria no caminho + i-node do ficheiro)

2. [1 val] Caso se ativem as caches usadas pelo sistema de ficheiros, indique qual/quais das caches seguintes permite(m) reduzir o número de i-nodes lidos do disco para abrir o ficheiro acima:
NOTA: Cada resposta errada implica uma **penalização de 0,25 valores**.

☐ Cache de i-nodes ☐ Cache de nomes [X] Ambas ☐ Nenhuma

3. [1 val] Num sistema ext3 com blocos de 4KB, executou-se o programa acima e, no final da execução, obteve-se "total=43873". Apresente o conteúdo do vetor de referências do i-node do ficheiro. Assuma que o conteúdo do ficheiro está mantido nos blocos nº 5, 6, 7 e seguintes.

Em ext3, o i-node contém um vetor de 12 referências diretas para blocos.
Neste caso, como o ficheiro ocupa 11 blocos, as primeiras 11 entradas do vetor de referências do i-node são preenchidas assim:

Vetor de referências de blocos do i-node:
0: 5
1: 6
2: 7
...
10: 15
11: -
12: -
13: -

4. [1 val] Considere o mesmo caso que na alínea anterior, mas agora com um sistema FAT. Apresente o conteúdo da tabela FAT que a informação de que dispõe permite conhecer.

Conteúdo da FAT:

...
5: 6
6: 7
7: 8
...
14: 15
15: EOF
...

5. [1 val] Considere agora um programa diferente em que, antes do printf, se chama a função *write(fd, ...).* Naturalmente, esta chamada dará erro. Em que estrutura de dados é que o sistema de ficheiros encontra a informação que lhe permite tomar essa decisão?

Na tabela de ficheiros abertos global.