Número: Nome:

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

22 de janeiro de 2019

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

Duração: 1h30m

Grupo I [7,5 Val]

1. Considere que o seguinte excerto de programa é executado por um processo P1 (`pid=34`, criado pelo processo com `pid=33`).

```
int i;
for (i=0; i<100; i++)
  fwrite(a, DIM, 1, f);
fflush(f);
exit(0);
```

Assuma que:

- A variável f é um descritor de ficheiro aberto previamente com *fopen* e a variável *a* é um vetor de bytes previamente preenchido.
- Quando a função fflush é chamada, todos os bytes escritos no ciclo de fwrites estão ainda acumulados no buffer que o *fopen* mantém no espaço de endereçamento de P1. Ou seja, nenhuma das chamadas a *fwrite* resulta em chamadas de sistema.
- Quando está em execução, P1 é mais prioritário que os restantes processos, logo nunca perde o processador por preempção.

a) [1 val] Durante a execução do excerto acima, indique as interrupções que garantidamente ocorrem na máquina em que P1 se executa.
Para cada uma, indique o tipo de interrupção (interrupção hardware, exceção, *trap* para chamada sistema) e a linha de código/acontecimento que origina a interrupção.

Tipo de interrupção	Causa
Trap	Chamada sistema feita na função *fflush* (para escrever o conteúdo do *buffer* no disco)
Interrupção hardware	Quando a escrita ao disco termina, ocorre uma interrupção
Trap	Chamada sistema feita na função *exit*

b) [1 val] Durante a execução do excerto acima, indique os momentos em que o CPU passa do modo de execução utilizador (*user*) para núcleo (*kernel*) devido à execução de alguma linha do excerto acima.
Responda sucinta e objetivamente (1 linha por cada momento).

Em todas as interrupções mencionadas acima, o CPU passa para modo núcleo.

Nota: Aceitámos como correto quem apenas mencionasse as chamadas sistema.

c) [1 val] Descreva um momento em que o processo P1 tenha estado num dos estados indicados abaixo (ou "nunca" caso P1 nunca tenha passado por esse estado).

Estado	Momento em que P1 esteve nesse estado
Execução em modo núcleo	(Ver resposta acima)
Bloqueado	Quando fflush aguarda que a escrita no disco seja terminada
Zombie	Após a chamada a exit

d) Quando P1 chama exit:

a. o núcleo liberta praticamente todo o contexto que mantinha sobre P1, com a exceção de uns poucos atributos.

i. [0,5 val] Indique que atributos são esses e quais os respetivos valores.

PID=34, PPID=33, status= (exited,0)

ii. [0,5 val] Quando é que esses atributos serão finalmente libertados?

Quando o pai chama *wait*.

b. [0,5 val] O processo pai de P1 receberá um *signal* SIGCHLD, que poderá ser ignorado, tratado (por uma função de tratamento) ou terminar o processo pai. Em que estrutura de dados é que o núcleo encontra a informação sobre qual o efeito que o signal terá sobre o processo pai?

No contexto software do processo pai.

2. Em Linux, monitorizou-se a execução de um CPU durante um período de tempo e observou-se que o CPU foi partilhado entre 2 processos, P2 e P3, da seguinte forma:

a) [1 val] Um dos processos corre um programa interativo. Pela informação acima, qual é esse processo?
Nota: resposta errada desconta 1/3 da cotação.

[ ] P2 [X] P3 ☐ Ambos

Nota: na alínea seguinte, as respostas às alíneas i) e ii) só serão cotadas se justificadas em iii).

b) [2 val] Ambos os processos iniciaram a sua execução com prioridade base igual.

i) Ao fim do período monitorizado, qual dos processos é mais prioritário?

[ ] P2 [X] P3 ☐ Ambos

ii) Após o período monitorizado, quando P2 ou P3 voltarem a receber execução, qual receberá um quantum superior?

[ ] P2 [X] P3 ☐ Ambos

iii) Justifique ambas as respostas acima referindo as regras de cálculo de prioridades do Linux (pode assumir qualquer um dos algoritmos de escalonamento usados pelo Linux que foram ensinados nas aulas).

Resposta considerando o algoritmo de escalonamento baseado em épocas (descrito no livro):

Da figura concluímos que o quantum por usar de P3 é superior ao de P2.
O quantum disponível a um processo é calculado em função do quantum que ficou por utilizar na época anterior, sendo que, quanto maior o quantum por usar, maior o quantum disponível e mais prioritário é o processo:
quantum = quantum_base + quantum_por_utilizar_epoca_anterior / 2
Prioridade = prio_base + quantum_por_usar_nesta_época – nice

Grupo II [8 Val]

1. Considere um sistema de gestão de memória com 32 bits de espaço de endereçamento virtual, com 4 bits (os mais significativos) para a identificação do segmento, e onde existem 3 segmentos, S_a, S_b e S_c (a restante memória primária encontra-se livre):
   - S_a: dimensão: 65.536 bytes, que não se encontra carregado em memória principal
   - S_b: dimensão: 32.768 bytes, que se encontra carregado em memória principal no endereço base: 0x00 00 00 00
   - S_c: dimensão: 65.536 bytes, que se encontra carregado em memória principal no endereço base: 0x00 01 00 00

   a) [0,5 Val] Quantos segmentos pode mapear em cada momento um processo no próprio espaço de endereçamento?

$2^4 = 16$

   c) [0,5 Val] Existe fragmentação externa neste caso? Caso a resposta anterior seja sim, qual é quantidade memória, em KB, afetada por fragmentação externa?

[ ] NÃO [X] SIM quantidade de memória afetada: 32 KB (entre S_b e S_c)

b) [1 val] Preencha a tabela de segmentos de um processo, P1, cujo espaço de endereçamento contém S_a e S_b, onde:
- S_a é mapeado como o segmento 0 do processo com permissões de leitura.
- S_b é mapeado como o segmento 1 do processo com permissão de leitura e escrita.

Segment ID	P	Prot	Limite	Base
0	0	R	65536	NA
1	1	RW	32768	0x00 00 00 00

c) [1 val] Preencha a tabela de segmentos de um segundo processo, P2, cujo espaço de endereçamento contém S_a, S_b e S_c, onde:
- S_b é mapeado como o segmento 0 do processo com permissões de leitura.
- S_c é mapeado como o segmento 1 do processo com permissões de leitura e escrita.
- S_a é mapeado como o segmento 2 do processo com permissão de leitura.

Segment ID	P	Prot	Limite	Base
0	1	R	32768	0x00 00 00 00
1	1	RW	65536	0x00 01 00 00
2	0	R	65536	NA

d) [2 val] Considere que os processos P1 e P2 se executaram no mesmo CPU (primeiro P1, depois P2), tendo produzido a sequência de acessos a endereços virtuais listada na tabela abaixo. Complete a tabela especificando em que caso(s):
- o endereço físico correspondente (se possível);
- se é necessário a carregar um segmento da memória secundária; neste caso, assuma que é usado o algoritmo "first-fit" e indique o endereço físico obtido após o carregamento;
- se o acesso dá origem a uma exceção devida a um "acesso fora dos limites" (L) ou "violação de permissão" (P) ou "falta de segmento em memória principal" (FS) ou se não gera alguma exceção (N).

Processo em execução	Acesso	Endereço virtual	Endereço físico	Acesso ao disco (S/N)	Exceção (L/P/FS/N)
P1	Leitura	0x00000011	0x0002 0011	S	FS
P1	Leitura	0x00000021	0x0002 0021	N	N
P1	Leitura	0x10000021	0x0000 0021	N	N
P1	Leitura	0x10001111	0x00 00 11 11	N	N
P2	Leitura	0x00000011	0x0000 0011	N	N
P2	Leitura	0x10000040	0x0001 0040	N	N
P2	Escrita	0x000000AF	0x0000 00AF	N	P

2. [1,5 val] Em sistemas baseados em paginação, com páginas de 16KB, endereços de 32 bits e tabelas de páginas **com um nível** e entradas de 4 bytes, quantos bytes pode ocupar, no máximo, a tabela de páginas de um processo? Indique apenas o valor em bytes.

Como os endereços virtuais são de 32 bits, o espaço de endereçamento virtual é de 2^{32} bytes = 4 GB.
Este espaço é dividido em páginas de 16KB = 2^{14} bytes.
Logo, no caso de um processo que use o espaço de endereçamento completamente, esse processo um número de páginas dado por 4GB/16KB = $2^{(32-14)}$ = 2^{18} páginas.
A tabela de páginas tem uma entrada por página, logo 2^{18} entradas (PTEs), ou seja ocupará 2^{18} x 4 bytes = 2^{20} bytes = 1 MB.

3. [1,5 val] Num sistema baseado em paginação, com páginas de 16KB, endereços de 32 bits e tabelas de pagínas **com dois níveis** e entradas de 4 bytes (em ambos os níveis), quantos bytes pode ocupar, no máximo, a tabela de páginas de um processo? Indique o valor em bytes e justifique de forma sucinta a resposta.

Usam-se 18 bits para endereçar páginas (ver ex. anterior). Assumindo que sejam utilizados b1 bits para o primeiro nível e b2 bits para o segundo nível, com b1+b2=18.
Temos 2^{b1} entradas na tabela de páginas (PT) de primeiro nível, onde cada entrada ocupa 4 bytes. O tamanho máximo da PT de 1º nível é portanto: 2^{b1} * 4 bytes.
Cada entradas na PT de primeiro nível referencia uma PT de 2º nível. No total, o conjunto de PTEs contido nas PTs de 2º nível é igual ao conjunto de PTEs que teríamos caso usássemos uma PT de 1 nível (como na alínea anterior). Ou seja, no total, as PT de 2º nível ocupam 1 MB.
Concluindo, o tamanho total é 2^{b1} * 4 bytes (PT de 1ºnível) + 1MB (PTs de 2º nível)

Nota: também foram consideradas corretas soluções que considerem valores específicos de b1 e b2 (p.e., b1=b2=14, de forma a garantir que o tamanho das PTs de 2º nível coincidam com a dimensão das páginas).

Grupo III [5 Val]

Considere o seguinte excerto de um programa:

```
1.    int socketfd = socket(AF_UNIX, SOCK_DGRAM, 0);
2.    int total = 0;

3.    while (1) {
4.      char buffer[DIM];
5.      n = recvfrom(socketfd, buffer, DIM, 0, 0, 0);
6.      if (n > 0) {
7.        int filefd = open("/usr/so/received", O_WRONLY | O_APPEND);
8.        write(filefd, buffer, n);
9.        close(filefd);
10.       total += n;
11.     }
12.   }
```

Considere que um processo P executa o programa acima e é interrompido por um *breakpoint* do debugger imediatamente antes de executar a linha 9 e os valores das variáveis são os seguintes:

```
socketfd = 5        filefd = 6          total = 4600
```

1. [1 val] Considere que, na tabela de ficheiros abertos **global**, o socket está representado na entrada 402 e o ficheiro está representado na entrada 500.
   Complete a tabela de ficheiros abertos de P abaixo com toda a informação de que dispõe.

Tabela de ficheiros abertos de P

402
500

2. [1 val] Observou-se que, na primeira iteração, a chamada open demorou muito tempo, enquanto que a mesma chamada em iterações seguintes foi muito mais rápida.
   Dê uma explicação sucinta para esta observação, relacionando com as estruturas de dados que o gestor do sistema de ficheiros mantém em memória RAM.

A primeira chamada teve de consultar a árvore de diretorias em disco até encontrar o i-node do ficheiro em causa. À segunda tentativa e restantes, o resultado dessa pesquisa já estava guardado na cache de nomes, em memória primária.

Também se consideram corretas respostas que referissem as caches de blocos ou de i-nodes.

3. [1 val] O sistema de ficheiros em que o ficheiro `/usr/so/received` existe é do tipo ext3.
   Apresente o conteúdo dos campos do i-node do ficheiro que referenciam os blocos de dados deste ficheiro. Na sua resposta, assuma que:
   - A partição do disco em que o sistema de ficheiros existe tem blocos de 1KBytes e os blocos 1200, 1201, 1202 e seguintes estavam livres no início do programa.
   - O ficheiro estava vazio no início do programa.

Em ext3, o i-node contém um vetor de 12 referências diretas para blocos. Neste caso, como o ficheiro ocupa 5 blocos (4600 bytes), as primeiras 5 entradas do vetor de referências do i-node são preenchidas assim:

Vetor de referências de blocos do i-node:
0: 1200
1: 1201
2: 1202
3: 1203
4: 1204
(restantes entradas irrelevantes)

4. [1 val] Responda de novo à mesma questão da alínea anterior, mas assumindo agora uma situação posterior em que P avançou mais iterações e `total=17900`.

Neste caso, o ficheiro ocupa 18 blocos, logo as 12 entradas diretas do i-node não são suficientes e é necessário também referenciar um bloco de índices, que por sua vez referencia os 6 blocos remanescentes.

Vetor de referências de blocos do i-node:
0: 1200
1: 1201
2: 1202
3: 1203
4: 1204
[...]
11: 1211
12: 1212 (referencia bloco de índices)

(restantes entradas irrelevantes)

Conteúdo do bloco de índices (bloco 1212):
0: 1213
1: 1214
2: 1215
3: 1216
4: 1217
5: 1218
(restantes entradas irrelevantes)

5. [1 val] Ponderou-se usar um tamanho de bloco de 4KBytes em vez de 1Kbyte neste sistema de ficheiros. Indique uma vantagem e uma desvantagem que essa mudança traria.

Vantagens: Suporte a ficheiros de maior dimensão; manutenção de menos meta-dados (blocos de índices); entre outras.
Desvantagens: maior fragmentação interna; entre outras.