

Número:

Nome:

LEIC/LETI – 2015/16 - 2º Teste de Sistemas Operativos

30 de Novembro de 2016

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

Duração: 1h

Grupo I [10 Val]

1. Considere o problema de sincronização clássico denominado produtores-consumidores, cujo pseudo-código se apresenta de seguida. Assuma que podem existir múltiplas tarefas produtoras e múltiplas tarefas consumidoras.

```
int buf[N], prodptr=0, consptr=0, count=0;
mutex_t m;
cond_t vazio, cheio;
```

```
1. produtor() {
2.   int item;
3.   while(TRUE) {
4.     item = produz();
5.     pthread_mutex_lock(&m);
6.     while (count == MAX)
7.       pthread_cond_wait(&vazio, &m);
8.     buf[prodptr] = item;
9.     prodptr = (prodptr+1) % N;
10.    count++;
11.    pthread_cond_signal(&cheio);
12.    pthread_mutex_unlock(&m);
13.  }
14. }
```

```
1. consumidor(){
2.   int item;
3.   while(TRUE) {
4.     pthread_mutex_lock(&m);
5.     while (count == 0)
6.       pthread_cond_wait(&cheio, &m);
7.     item = buf[consptr];
8.     consptr = (consptr+1) % N;
9.     count--;
10.    pthread_cond_signal(&vazio);
11.    pthread_mutex_unlock(&m);
12.    consome (item);
13.  }
14. }
```

- a. [1 val] Quais as linhas de código que garantem a exclusão mútua em ambas as funções?

Linhas na função produtor:

Linhas na função consumidor:

- b. Considerando apenas a função produtor:

- i. [1 val] Indique as linhas de código que são garantidamente executadas em exclusão mútua nas funções produtor e consumidor (indicar números de linhas).

Linhas na função produtor:

Linhas na função consumidor:

- ii. [1,5 val] Considerando apenas a função produtor, indique qual o mutex usado, e em que linhas é que o mutex em causa é implicitamente fechado e aberto.

Mutex usado:

O mutex m é implicitamente fechado:

O mutex m é implicitamente aberto:

- c. [1,5 val] Assumindo agora que existe apenas um produtor, diga se poderia retirar as linhas 5 e 12 na função `produtor`? Justifique a sua resposta.

2. Considere um sistema operativo do tipo Unix no qual existem duas classes de processos, P1 e P2, que podem co-existir. Assuma que os processos da classe P1 são "I/O intensive" e os da classe P2 são "CPU intensive" (e.g. não efectua *system calls*). Considere ainda os seguintes algoritmos de escalonamento:

- E1, que suporta prioridades variáveis (mais CPU implica diminuição de prioridade), suporta preempção, e "time-slice" fixo
 - E2, que suporta prioridades fixas (inferiores às dos processos de classe P1), não suporta preempção, e suporta "time-slice" fixo.
- a. [1 val] Diga qual o algoritmo de escalonamento, E1 ou E2, mais adequado para suportar uma mistura de processos das classes P1 e P2.

- b. [1 val] Diga qual o algoritmo de escalonamento, E1 ou E2, mais adequado caso apenas se executem processos da classe P2 onde se pretende maximizar o débito.

c. Considere que o E1 calcula as prioridades de todos os processos existentes de X em X tempo, em que o intervalo X é fixo.

i. [1,5 val] Qual a desvantagem principal?

ii. [1,5 val] Assuma agora que X varia tal como ocorre no Linux (cada intervalo é designado época). Diga como é definido o tempo que cada época demora.

Grupo II [10 Val]

1. [1,5v] Num sistema de ficheiros ext3, listou-se o conteúdo da diretoria /tmp (usando o comando ls) e obteve-se a seguinte informação (simplificada) quanto ao ficheiro /tmp/readme.txt:

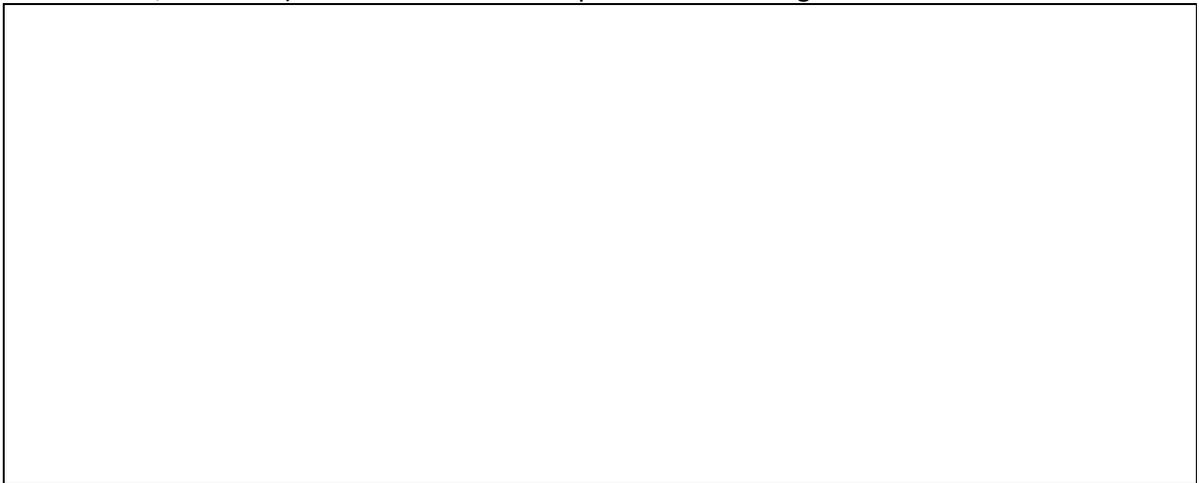
```
Nome do ficheiro:  readme.txt
Permissões:  rw- (dono) r-(grupo) --- (resto)
Dono:  UID=alex GID=alunos
```

Em que estrutura(s) de dados do sistema de ficheiros é mantido cada elemento listado acima?
No caso de mais que uma estrutura, indique claramente quais os elementos contidos em cada estrutura distinta.

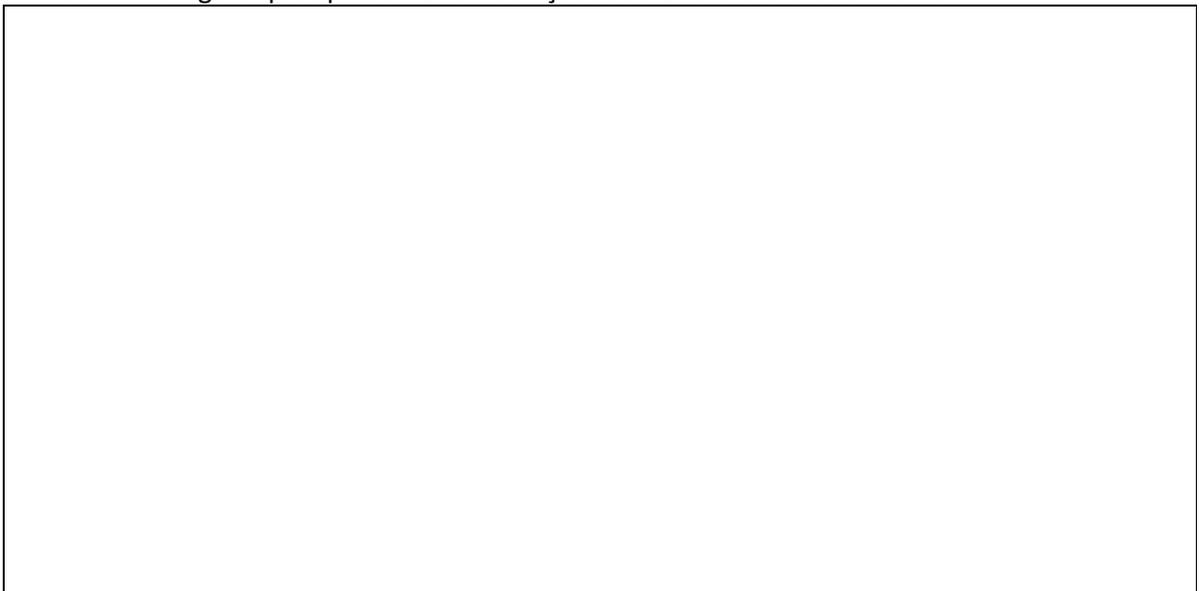
2. No mesmo sistema, considere um processo com UID=jose e GID=alunos que corre o seguinte programa (sobre o mesmo ficheiro da alínea anterior):

```
int f = open("/tmp/readme.txt", O_RDONLY);
if (f==-1) {
    printf("erro no open"); exit(1);
}
if (write(...) == -1) {
    printf("erro no write"); exit(1);
}
close(...);
printf("OK\n");
exit(0);
```

- a. [2v] Que mensagem é impressa pela execução deste programa por este processo? Justifique descrevendo sucintamente toda a execução (explicando porque cada passo teve sucesso/insucesso) até ao momento da impressão da mensagem.



- b. [1,5v] Assuma que a função open tem sucesso (mesmo que a sua resposta à alínea 2.a o contradiga). Que estruturas de dados são modificadas pela função? Ilustre a sua resposta com uma figura que apresente as alterações sobre as estruturas em causa.



- c. [1,5v] Assuma que a função `write` tem sucesso (mesmo que a sua resposta à alínea 2.a o contradiga) e que não aumenta a dimensão do ficheiro. Enumere as estruturas de dados que são alteradas pela função em cada uma destas situações; para cada estrutura, indique o(s) campo(s) que são alterados.

- d. [1,5v] Para abrir este ficheiro, quais i-nodes precisa a função `open` consultar? Assuma que não existe cache de nomes.

3. [2v] Considere que o volume (ou partição, assumindo que ambas coincidem) em causa tem blocos de 1Kbyte (ou seja, os dados do ficheiro ocupam 294 blocos) e em que cada referência (índice) de bloco ocupa 4 bytes. O conteúdo do ficheiro `readme.txt` é de 300.761 bytes. Quantos blocos de índices são usados para representar este ficheiro? Apresente-os numa figura (juntamente com o i-node).