Fundamentos de Sistemas de Operação MIEI 2014/2015

2º Teste, 9 de Dezembro 2014, 2 horas – versão A

Número_____ Nome _____

Sem consulta e sem esclarecimento de dúvidas; indique explicitamente nas suas respostas eventuais hipóteses colocadas
A detecção de fraude durante a realização ou correcção do teste implica, no mínimo, a reprovação à cadeira.

Questão 1 (2.0 valores) Considere um sistema operativo em que o algoritmo de escalonamento é Round Robin com uma única fila READY e uma fatia fixa de tempo *Tf*. Suponha que existem no sistema 20 processos *cpu-bound* e que não é aceitável que um processo esteja mais de *500* milisegundos sem que lhe seja atribuído CPU. Faça, justificando, uma estimativa para um valor de *Tf* que permitiria assegurar o requisito anterior. O valor que calculou é aceitável se considerarmos um tempo de comutação entre processos (*Tc*) de 10 microsegundos?

O tempo máximo de espera é $(20-1) * Tf \leq 500$ ms
Podemos escolher por ex. 20 ms para Tf ($20 \times 20\,ms = 400\,ms$)

$Tc = 10\,\mu s = 0.01$ ms

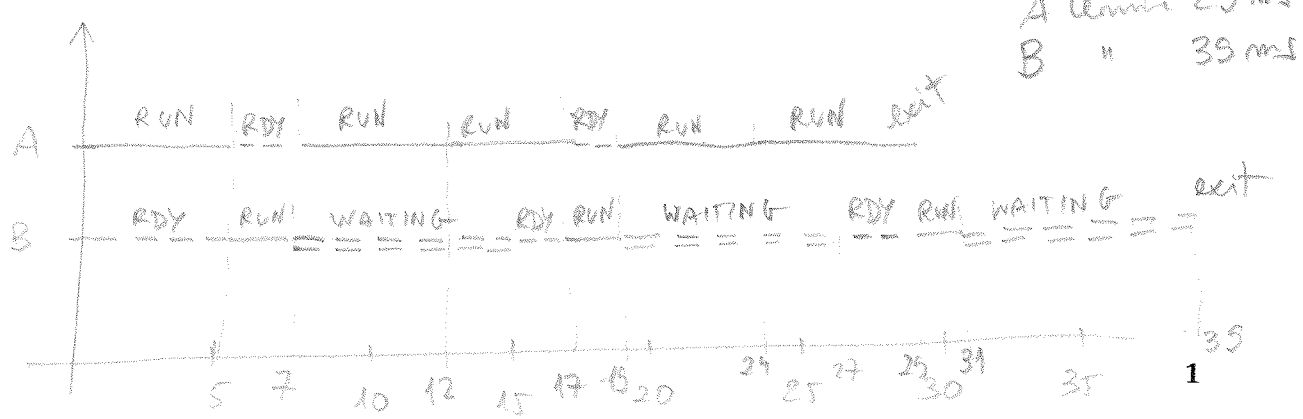
$Tc \ll Tf$ portanto está ok

Questão 2 (2.0 valores) Considere um sistema operativo que utiliza um algoritmo de escalonamento com uma única fila de processos prontos e uma fatia fixa de tempo *Tf*. Admita que dois processos A e B são os únicos no sistema, tendo entrado na fila de processos prontos no instante de tempo 0, pela ordem A, B. Quando um processo passa de WAITING para READY é colocado à cabeça da fila READY. Os processos realizam as seguintes ações:

- Processo A: Usa o CPU durante 25 milisegundos e termina
- Processo B: Faz 3 vezes um conjunto de acções em que usa o CPU durante 2 milisegundos e de seguida faz uma operação de entrada/saída que demora 8 milisegundos; após o ciclo, termina

Supondo que a troca de contexto tem uma duração desprezável, diga, justificando, em que instante de tempo terminam os processos A e B para um algoritmo de escalonamento *Round Robin* com uma fatia de tempo de 5 ms. Justifique a sua resposta fazendo um diagrama em que no eixo dos XX está o tempo e sobre ele estão duas linhas, uma para o processo A e outra para o processo B em que seja possível saber, em cada momento, em que estado (RUNNING, READY, WAITING) estão os processos A e B.

A termina 29 ms
B " 39 ms



Questão 3 (2.0 valores) Considere um escalonador de processos que usa múltiplas filas READY ordenadas por prioridade (*MLFQ Multi-level Feedback Queue*). A prioridade máxima P corresponde ao valor numérico 0 e a prioridade mínima ao valor numérico 199. O escalonador recalcula a prioridade P de um processo de 1 em 1 segundos de acordo com a fórmula

$$P = (V / 2) + 60$$

em que V é directamente proporcional ao tempo de CPU atribuído recentemente ao processo. Diga o que acontece à prioridade de um processo *CPU-bound* e explique porque é que é necessário dividir V por 2 a cada segundo que passa.

Um processo CPU-Bound consome as suas fatias de tempo até ao fim. Durante 1s acumula tempo de CPU e portanto V é maior do que o processo IO Bound. Assim sendo, em cada S a sua valor de P cresce e a sua prioridade diminui. A divisão por 2 permite uma recuperação de prioridade mais elevada quando o processo passa de uma fase CPU Bound para a IO Bound

Questão 4 (3 valores) Um sistema em que os endereços virtuais têm 32 bits, usa uma MMU que suporta páginas com dimensão de 1 Mbyte (2^20).

a) Para as condições acima referidas, pretende-se fazer uma simulação do número de faltas de página correspondentes a uma dada sequência de endereços virtuais. O primeiro passo é obter o número da página virtual *npv* e o deslocamento *desc* correspondente a um dado endereço virtual *ev*. Para esse efeito escreveram-se as seguintes linhas de código C que se pretende que complete:

```
#define PAGE_SIZE  2 ^20
npv =  ( ev && 0xfff00000) >> 20;
desc =  ev && 0x000fffff;
```

b) Para um dado processo em execução, as primeiras entradas da tabela de páginas são as seguintes:

Nº página virtual	Nº página física (em base 16)
0	0xBAD
1	0xABC
2	Inválida
3	0xBEE

As outras entradas são todas inválidas. Indique, justificando, os endereços físicos que correspondem aos endereços virtuais (em base 16) abaixo indicados. Responda "Endereço Inválido" se o endereço virtual for inválido.

0x00000000 pv = 0x000 pf = 0xBAD ef = 0xBAD00000

0x00222001 pv = 0x002 pf fim inválido endereço inválido

0x10001001 pv = 0x100 pf fim inválido endereço inválido

0x0033BA11 pv = 0x003 pf = 0xBEE ef = 0xBEE3BA11

Questão 5 (3 valores) Considere um sistema de operação e um CPU+MMU que suportam paginação a pedido.

a) Explique como é que é possível executar programas cuja imagem é superior ao tamanho da memória física disponível.

Usando paginação a pedido, apenas é preciso ter em RAM as páginas referenciadas pelas instruções. As páginas são carregadas em memória RAM à medida que são necessárias. Quando não há páginas físicas livres usa-se um algoritmo de substituição para escolher a página vítima.

b) Para este tipo de sistema, diga como funciona a ligação dinâmica de bibliotecas. Relacione esse funcionamento com a possibilidade que existe de mapear ficheiros no espaço de endereçamento de um processo (como faz a chamada ao sistema *mmap( )* do UNIX).

As bibliotecas dinâmicas contêm código puro e independente de posição (PIC). Quando um símbolo definido numa biblioteca é referenciado, o SO usa uma função como o mmap do UNIX para mapear o ficheiro que contém o código da biblioteca no espaço de endereçamento do processo. A partir daqui funciona o sistema de paginação a pedido

c) Neste contexto, diga o que é uma situação de *thrashing*. Supondo que dispõe permanentemente dos seguintes indicadores sobre o desempenho do sistema

- Taxa de ocupação do CPU (PCPU)
- Número de operações sobre o disco de paginação (NIOP)
- Número de páginas físicas livres (NFL)

que combinações de valores de (PCPU, NIOP e NFL) permitiriam concluir que o sistema estava numa situação de *thrashing* ?

Uma situação de thrashing acontece quando todos os processos têm em memória menos páginas do que o mínimo. Assim, a ratio de faltas de página aumenta muito e o tempo de execução de instruções também muito

$T_{me} = h * T_{RAM} + (1-h) T_{disco}$ e h aproxima-se de 0 ...

Em termos dos indicadores

PCPU - baixo, porque os processos estão todos bloqueados

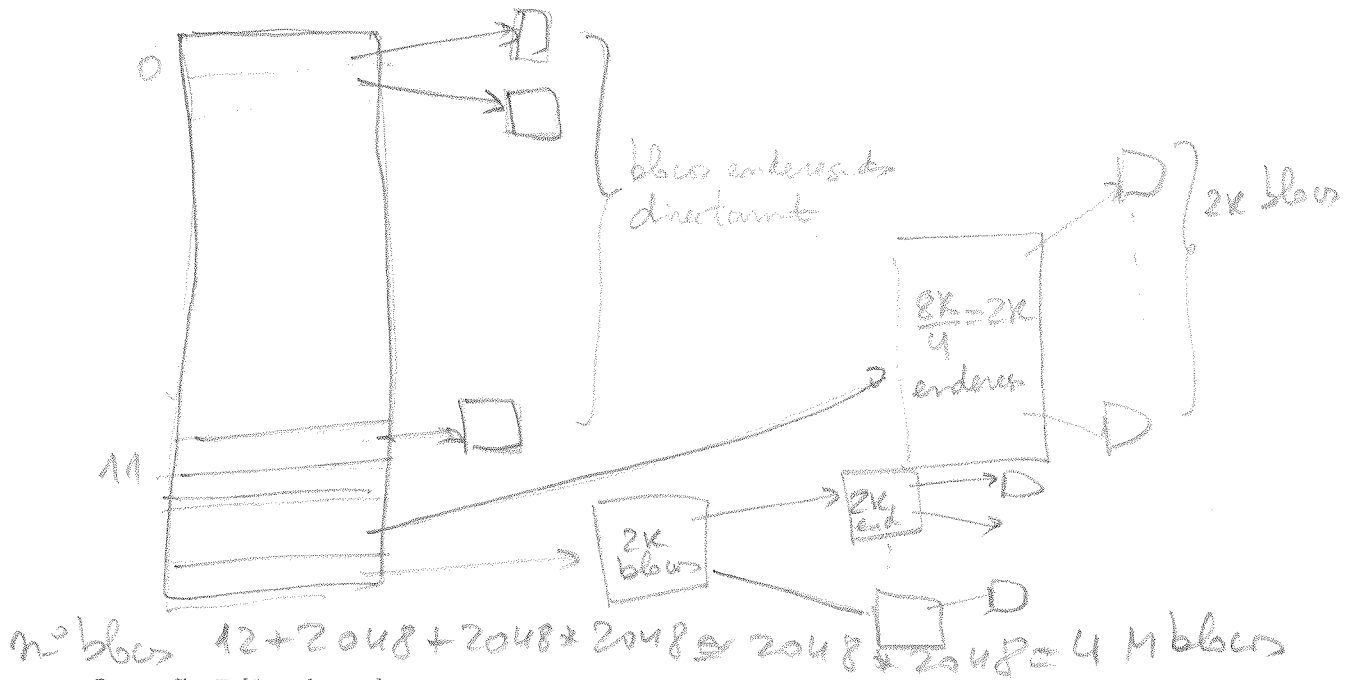
NIOP - alto, porque estão a ocorrer muitas transferências entre a RAM e o disco

NFL - baixo, há poucas páginas livres

Questão 6 (2 valores) Suponha uma variante do sistema de ficheiros UNIX em que os blocos têm 8 Kbytes e os endereços de blocos têm 4 bytes. Cada i-node tem:

- 12 endereços directos
- 1 endereço indirecto - isto é que contém o endereço de um bloco com endereços
- 1 endereço duplamente indirecto - contém endereços de blocos que contêm endereços de blocos com endereços

Diga, justificando, qual é máxima dimensão em blocos de um ficheiro neste sistema.



Questão 7 (3 valores) O sistema de ficheiros XPTO usa uma forma de designação dos ficheiros igual à do UNIX - */dir1/dir2/dir3/.../ficheiro*. Estão disponíveis as chamadas ao sistema tradicionais sobre ficheiros (*open / read / write /close*). Na zona dos meta-dados de um disco formatado com o sistema de ficheiros XPTO há

- Um *super bloco* que contém a dimensão do disco e o número do bloco onde se encontra a directoria raiz.
- Um *bitmap* de ocupação de blocos
- Uma *tabela geral de ficheiros* (TGF) que tem uma entrada para cada ficheiro existente no disco e em que cada entrada tem os campos:
  - Em uso/livre
  - Comprimento do ficheiro em bytes
  - Lista de blocos onde estão os dados do ficheiro
- As directorias são ficheiros que podem ser vistos como um vector de registos em que cada registo tem os seguintes campos:
  - Tipo da entrada: ficheiro, directoria, livre
  - Nome (cadeia de caracteres)
  - Índice na TGF onde está a informação sobre o ficheiro.

Para o sistema de ficheiros XPTO pretende-se que descreva em detalhe como seriam implementadas as chamadas ao sistema referidas nas alíneas abaixo. Repare que o que se pretende é que indique o que é feito em termos de preenchimento das tabelas do sistema operativo (quer as globais quer as associadas a cada processo) e as leituras e escritas que são feitos na zona de meta-dados do disco e não o que fazem as chamadas ao sistema do ponto de vista do programador.

a) Abertura para escrita de um ficheiro que se supõe que já existia antes mas que está vazio

f = open("/dir1/f", O_WRONLY)

- leitura do bloco que contém a dir. raiz
- extracção da directoria raiz do bloco onde está a dir. dir1
- leitura da directoria dir1
- extracção da directoria dir1 do índice no TCAF do ficheiro f
- obtenção de uma entrada livre L na tabela de ficheiros em uso no RAM. Preenchimento da entrada L com a informação sobre f
- ocupação da 1ª entrada livre da tabela de canais do processo com, entre outras coisas, uma referência para L

b) Seja BLOCK_SIZE o tamanho de um bloco do disco. Escrita de BLOCK_SIZE bytes no final do ficheiro e em que o *offset* corrente é um múltiplo de BLOCKSIZE.

nw = write(f, buf, BLOCKSIZE)

- A partir da entrada f na tabela de canais obtem-se o offset no ficheiro e qual a entrada L na tabela geral de ficheiros abertos
- se necessário obtém-se um bloco livre B no bitmap de ocupação do bloco. Actualiza-se o bitmap
- escreve-se o conteúdo de buf no bloco B
- actualiza-se a informação sobre o ficheiro (comprimento, bloco) na entrada L da tabela geral de ficheiros abertos
- actualiza-se a entrada f da tabela de canais abertos

c) Fecho do ficheiro *close(f).*

- actualizam a info em disco a partir de RAM, se no write isto não foi feito.
- liberta-se a entrada L da tabela geral de ficheiros abertos
- liberta-se a entrada f da tabela de canais abertos do processo

Questão 8 (3 valores) Considere o sistema de ficheiros descrito na pergunta 7. Suponha que o mapa de blocos ocupa apenas um bloco no disco e que cada directoria também ocupa no máximo um bloco. Imagine que vai remover o ficheiro */dir1/f7* que se supõe que existe.

a) A operação referida implica a escrita em mais do que um bloco do disco. Que blocos são esses?

É preciso escrever:
A no bloco com o bitmap dos blocos para libertar os blocos que pertenciam ao ficheiro
B no TOF para indicar que a entrada está livre
C no bloco que contém a directoria dir1 para assinalar que a entrada está livre.

b) O que aconteceria se houvesse uma falha de energia ou um erro no sistema operativo que se manifestasse pela terminação abrupta do sistema operativo enquanto a operação de remoção do ficheiro estava a decorrer. Quais seriam as consequências para o sistema de ficheiros?

A ordem pela qual as três operações acima são efectuadas não é indiferente. Supondo por exemplo que B e C foram feitas e A não: há blocos ocupados que não são referenciados por nenhuma entrada em directoria mas que estão ocupados. O sistema de ficheiros está incoerente.

c) Suponha que se pretendia corrigir o erro anterior da próxima vez que se fizesse *mount( )* do disco. O que haveria a fazer nessa ocasião? Refira apenas o que fazer em relação a remoções de ficheiros que não correram bem.

Usando uma estratégia "à fsck" ter-se-ia de:
- Verificar se todos os blocos referenciados por entradas na TOF estão marcados como ocupados no bitmap. Marcar como livres os blocos não referenciados na TOF
- Percorrer a árvore de directorias e verificar quais são as entradas na TOF que estão referenciadas por entradas em directorias. Marcar como livres as que não são referenciadas e fazer algo com os blocos (colocá-los como livres, colocar no lost+found?)