Fundamentos de Sistemas de Operação MIEI 2015/2016

2º teste, 7 de dezembro de 2015, 2h

Nº _____ Nome _____

Sem consulta e sem esclarecimento de dúvidas; indique eventuais hipóteses assumidas nas sua respostas.

Questão 1 (2.5 valores) Um dos problemas da gestão da memória central (RAM) em ambientes que suportam um número variável de processos carregados em memória é a *fragmentação externa*.

a) Explique porque é que este problema ocorre quando a memória física é gerida por uma MMU com um registo base e um registo limite.

b) E se a MMU for baseada em páginas o problema ocorre? Justifique.

a) Com uma MMU com um registo base e um registo limite a imagem do processo tem de ser carregada em memória de forma contígua. Isto leva a que, após algum tempo a criar processos e a estes terminarem, a memória livre fique dividida em vários fragmentos que não estão contíguos em memória.

b) Se a MMU for baseada em páginas, as imagens dos processos deixam de necessitar de estar carregadas contíguamente. Isto faz com que a memória livre possa estar fragmentada sem causar problemas.

Questão 2 (2.5 valores) Considere um sistema de gestão de memória baseado em páginas, suportando paginação a pedido. É possível executar um programa cuja imagem tenha uma dimensão superior à da memória física? Se sim, explique como.

É possível, usando paginação a pedido e algoritmos de substituição de páginas.
Isto consegue-se mantendo em memória apenas uma parte das páginas do processo. A tabela de páginas tem o bit de validade a 1 para as páginas da imagem que estão carregadas e LAM e 0 para as páginas que estão apenas no disco. Assim sendo, quando ocorre uma falta de página, é MMU envia uma interrupção e o SO resolve a falta de página. Se necessário utilizando uma página física que pertence ao processo, mantendo assim em memória um nº de páginas físicas do processo limitado.

Questão 3 (2.5 valores) Suponha um sistema de operação que executa numa máquina com uma MMU baseada em páginas. Considere a chamada ao sistema *fork( )* usada nos sistemas operativos da família Linux / UNIX.

a) Explique como seria preenchida a tabela de páginas do processo filho.

b) Diga como poderia otimizar as ações descritas em a) em termos da diminuição do tempo de execução da chamada ao sistema *fork( )* e também da diminuição do nº de páginas físicas ocupadas efetivamente.

a) As entradas de tabela de páginas do processo filho são preenchidas de seguinte forma:

1) páginas de código e de dados imutáveis: cópia das entradas correspondentes da tabela de páginas do pai

2) páginas de dados, heap e stack: páginas novas mas cujo conteúdo é uma cópia dos do pai.

b) Para as páginas referidas em 2) pode ser usada a técnica Copy On Write. Essas entradas na tabela de página são preenchidas com o mesmo valor que está na tabela de páginas do pai. Essas entradas na tabela de página do pai e do filho são marcadas em read-only. Quando um dos processos escreve, a MMU deteta um acesso ilegal e envia uma interrupção ao SO. No seguinte à interrupção, o SO duplica a página, atualizando as tabelas de páginas do processo e marcando ambas as entradas com Read-Write.

Questão 4 (2.5 valores) Suponha um sistema de operação que executa numa máquina com uma MMU baseada em páginas em que cada página tem 4 Kbytes. Executa-se um programa que contém o seguinte fragmento:

```
#define NPAGES 20
#define PAGESIZE 4096
...
unsigned char vec[NPAGES*PAGESIZE]; // 1 char ocupa um byte na RAM
for ( i=0; i<5; i++ )
      for( j=0; j < NPAGES; j++ )
            for( k=0; k < PAGESIZE; k++ )
                        vec[j*PAGESIZE+k] = (unsigned char)i;
...
```

Considere que o código acima já está carregado em RAM, que o vetor *vec* não está carregado em RAM, e que as variáveis *i, j* e *k* estão em registos do CPU; pode assim supor-se que as únicas faltas de página que ocorrem têm a ver com o acesso ao vetor *vec*. Diga quantas faltas de página ocorrem durante a execução do pedaço de código acima indicado quando:

a) é possível ocupar na RAM 20 páginas com o vetor *vec*. Justifique.

b) é apenas possível manter em RAM 5 páginas do vetor *vec*. Suponha que o algoritmo de substituição de páginas é o LRU (*least recently used*). Justifique.

a) Em cada um dos ciclos exteriores o vector é varrido sequencialmente do 1º ao último elemento. Quando $i = 0$, haverá uma falta de página para cada uma das 20 páginas. Para $i = 1$ até $i = 4$ não há faltas de página uma vez que as 20 páginas estão em RAM.

R: 20 faltas de página

b) Quando $i = 0$ ocorrem como em a) 20 faltas de página. Quando $i = 1$ e até $i = 4$ volta a haver 20 faltas de página em cada ciclo porque quando o ciclo se reinicia as páginas presentes na memória são a 15, 16, 17, 18 e 19 e elas já não estão em RAM quando se faz com as últimas páginas.

R: 100 faltas de página

Questão 5 (3.0 valores) Considere o sistema de ficheiros do UNIX. Explique o que se passa, em termos de alterações nos meta-dados e nas diretorias quando se

a) Muda o nome a um ficheiro com o comando

```
mv ./f1 ./f2
```

O 1° argumento é o nome antigo e o 2° argumento é o nome novo. Suponha que antes de dar o comando o ficheiro *f1* existe e o ficheiro *f2* não existe.

b) Se apaga um ficheiro com o comando

```
rm /d1/f3
```

Suponha que antes de dar o comando a diretoria *d1* e o ficheiro *f3* existem

a) - obtém-se o i-node do directório corrente e lê-se o seu conteúdo para RAM
- verifica-se se f2 existe e se tal acontecer remove-se f2 (como descrito em b))
- na entrada correspondente a f1, muda-se o campo nome para f2

b) - lê-se o conteúdo do directório raiz. Aí obtém-se o i-node do directório d1
- lê-se o conteúdo do directório d1 e obtém-se o i-node do ficheiro f3
- apaga-se a entrada f3 no directório d1
- actualiza-se o link count no i-node de f3
- se o link count se tornar 0
  - declaram-se livres os blocos que estão listados no i-node de f3
  - liberta-se a entrada correspondente a f3 na tabela geral de ficheiros abertos (em RAM)
  - liberta-se a entrada de f3 na tabela de i-nodes

Questão 6 (2.0 valores) Considere um sistema de ficheiros que usa o método indexado para atribuir blocos a um ficheiro. A localização dos blocos é definida por um vetor *a[ ]* com 12 endereços

- Os endereços *a[0]* a *a[9]* são endereços diretos
- O endereço *a[10]* é o endereço de um bloco que contém endereços de blocos (indireto simples)
- O endereço *a[11]* é o endereço de um bloco que contém endereços de blocos que contêm por sua vez endereços de blocos (indireto duplo)

Supondo que em cada bloco do disco cabem 1024 endereços e cada bloco tem 4 Kytes, qual é o maior tamanho de ficheiro que pode existir neste sistema de ficheiros? Justifique indicando a expressão usada no cálculo.

nº de blocos = 10 endereç directos + 1024 endereç indirecto

+ 1024 * 1024 endereç indirecto ≃ 1024 * 1024 blocos

tamb max ≃ 1024 * 1024 * 4K ≃ 4 G bytes

Questão 7 (3 valores) Considere o sistema de ficheiros FATX-32 com as seguintes características principais:

- o bloco 0 é o superbloco e contém o número mágico, o número de blocos NB do disco e um vetor de inteiros *V* sem sinal com NB posições. Cada entrada *V[i]* tem o seguinte conteúdo:
  - 0x00000000 bloco i está livre
  - 0xFFFFFFFF bloco i é o fim da lista ligada de blocos em que o ficheiro está organizado
  - Próximo bloco do ficheiro, constituindo a lista ligada de blocos que começa na entrada da diretoria correspondente.
- O bloco1 contém a única diretoria existente. Cada entrada desta diretoria tem
  - Uma marca a dizer se está ocupada ou livre
  - O nome do ficheiro
  - Um inteiro com o comprimento do ficheiro em bytes
  - O número do bloco onde começa o ficheiro. Isto é, se este campo da entrada da diretoria é *c*, *V[c]* é o 1º bloco do ficheiro.
- Os restantes blocos são blocos de dados

Suponha que quando um disco é montado se pretende verificar se o sistema de ficheiros está coerente. O que seria necessário fazer? Note que não se pretende corrigir as inconsistências eventualmente existentes.

(1) Verificação de consistência do mapa de blocos com o conteúdo da diretoria

a) para todas as entradas verificar se a lista termina e se todos os apontadores estão entre 1 e NB-1 (salvo o último)

b) Construir um vector de inteiros com NB posições todo inicializado a 0x00000000. Percorrer todas as entradas na diretoria marcando a 1 os blocos ocupados. Aproveitar para detectar se há blocos reivindicados por mais do que um ficheiro.
Após a construção do mapa auxiliar, verificar se o mapa do bloco ∅ e o mapa construído têm ∅ nas mesmas posições

(3) Verificar se o comprimento do ficheiro é coerente com o nº de blocos NB que fazem parte da lista que se inicia no bloco inicial indicado na diretoria e que termina com o bloco marcado a 0xFFFFFFFF.
Assinalar os casos em que NB é diferente de (comp. ficheiro / tamb de bloco) arredondado para o multiplo de tamb de bloco mais próximo.

Questão 8 (2 valores) Suponha um disco com as seguintes características:

- *seek time* médio: 12 ms
- tempo de uma rotação: 6 ms
- 500 sectores por pista; o controlador do disco tem memória suficiente para ler uma pista inteira

Apresente, justificando, uma estimativa para o tempo que demora a leitura de 20 blocos nas seguintes duas situações:

a) os 20 blocos estão espalhados aleatoriamente pelo disco

b) os 20 blocos estão todos contíguos na mesma pista.

Note que não precisa de apresentar um valor numérico, mas apenas as expressões que permitem calcular os dois valores.

a)
tempo total = 20 * (seek time + latência rotacional + tempo de transferência)

seek time → 12 ms
latência rotacional → 6 ms / 2
tempo de transferência → 6 ms / 500

b) tempo total = seek time + latência rotacional + 20 * tempo de transferência

também poderia ser

tempo total = seek time + tempo de leitura da pista
tempo de 1 rotação