Número: Nome:

LEIC/LERC – 2011/12 - 2º Exame de Sistemas Operativos – 3/Fevereiro/2012

Responda no enunciado, apenas no espaço fornecido.
Identifique todas as folhas. Duração: 2h30m

Grupo I [4v]

1. [0,6v] A comutação entre processos, entre tarefas reais do mesmo processo e entre pseudo-tarefas do mesmo processo tem um desempenho muito diferente. Para cada tipo de comutação, indique: i) se a comutação implica passagem para modo núcleo ou não; ii) que informação de contexto que é necessário trocar.
   a. Comutação entre processos.

   b. Comutação entre tarefas reais do mesmo processo.

   c. Comutação entre pseudo-tarefas do mesmo processo.

2. O Linux suporta preempção, mas não para processos que estejam temporariamente a executar-se em modo núcleo.
   a. [0,6v] Dê um exemplo que ilustre porque é que esta característica é uma limitação. .

   b. [0,6v] De que forma é que o escalonador do Linux tenta minimizar o impacto desta limitação?

3. Considere a noção de processo zombie.
   a. [0,5v] O que é um processo zombie?

   b. [0,5v] Dê um exemplo de uma execução que produza um processo zombie.

4. Considere as seguintes implementações de trincos, tal como estudadas nas aulas: i) trinco hardware usando a instrução compare&swap; e ii) trinco lógico com suporte do núcleo. Assuma que existe um programa multi-tarefa que tem uma secção crítica longa, que se pretende sincronizar um trinco.
   Para cada cenário descrita abaixo, indique qual a implementação de trinco (i ou ii) que considera mais apropriada, justificando.
   a. [0,6v] Cenário de baixa contenção: quando uma tarefa pretende entrar na secção crítica, há uma probabilidade de 0,000001% de outra tarefa querer entrar também na secção crítica.

   b. [0,6v] Cenário de alta contenção: todas as tarefas acedem frequentemente à secção crítica, pelo que há alta probabilidade de haver mais que uma tarefa a tentar entrar em simultâneo.

Grupo II [4v]

Considere o seguinte problema:
- existem múltiplas tarefas, que por vezes podem querer enviar ou receber mensagens para/de um canal partilhado, com capacidade N;
- o canal é implementado como um buffer circular;
- a função *send* permite enviar múltiplas mensagens para o canal de uma só vez, sendo as mensagens colocadas de forma consecutiva no buffer circular; se não houver espaço suficiente, a função *send* espera até que haja;
- a função *receive* permite receber a próxima mensagem disponível no canal; no caso do canal estar vazio, a função espera até chegar uma mensagem.

Estude a seguinte implementação de uma solução para o problema (em pseudo-código):

```
int channel[N];
int free = N;
int sendptr=0, recvptr=0;

send(Message messages[], int numMsg) {
   while (free < numMsg);
   for each (Message m in messages) {
      channel[sendptr] = m;
      sendptr = (sendptr+1) % N;
      free --;
   }
  }
}
```

```
Message receive() {
  while (free == N);
  Message m = channel[recvptr];
  recvptr = (recvptrptr+1) % N;
  free ++;
  return m;
}
```

1. Usando a solução acima apresentada numa máquina mono-processador, houve várias queixas dos utilizadores da solução. Para cada queixa apresentada abaixo, explique porquê pode acontecer e ilustre com um exemplo de execução que produza o mesmo sintoma.
   a. [0,9v] "Um produtor colocou 1 item no buffer; no entanto, esse mesmo item foi consumido por 2 consumidores diferentes."

   b. [0,9v] "Quando um produtor tenta colocar um item no buffer quando este está vazio e há vários consumidores a tentar consumir, o produtor demora muito mais tempo a concluir a operação (que demoraria se não houvesse consumidores à espera de itens)."

2. [2,2v] Proponha uma solução que elimine ou minimize os problemas apontados acima. Na sua solução pode recorrer a trincos lógicos e/ou semáforos, desde que os declare.

   a. Explique se a solução resolve cada um dos problemas apontados acima para a primeira solução (1.a., 1.b, 1.c), justificando.

Grupo III [4v]

1. Considere o sistema operativo Unix/Linux, quando um ficheiro é aberto através da função *open* sendo retornado um inteiro.

a) [0.6v] Qual a motivação fundamental para a operação referida, do ponto de vista do desempenho ?

b) [0.6v] Quais as operações que seriam efectuadas, sempre que se pretendesse ler/escrever de/num ficheiro, caso a função *open* não existisse ?

2. [0.6v] Considere a organização de um sistema de ficheiros em lista ligada. Indique duas desvantagens. Justifique.

3. [0.6v] Considere agora a organização de um sistema de ficheiros FAT. Indique duas desvantagens. Justifique.

4. [0.6v] Considere um processo P1 que abriu um ficheiro XPTO em modo leitura. Em seguida, cria um processo filho P2. Diga se ambos os processos, P1 e P2, partilham o mesmo cursor para o processo em causa. (Atenção: responda SIM ou NÃO e justifique).

5. [1v] Apresente as *tabelas de ficheiros abertos do processo, tabela de ficheiros abertos global* e de *cache de inodes* e a sua interligação ao longo de cada um dos passos seguintes (apresente as estruturas de dados mencionadas correspondentes a cada um dos passos indicados):
   a) Processo P1 inicia a sua execução e cria um ficheiro de nome XXX para escrita e leitura.
   b) Processo P1 abre ficheiro já existente, denominado YYY, apenas para leitura.
   c) Processo P1 cria processo filho P2.
   d) Processo P2 abre ficheiro XXX para leitura.

Grupo IV [4v]

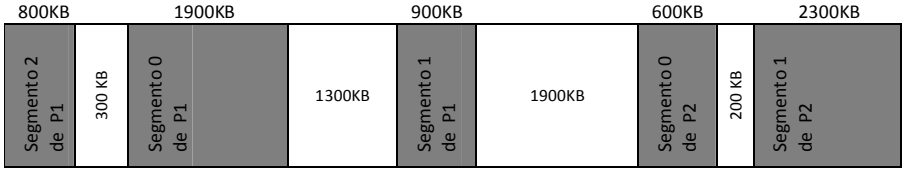
Considere uma máquina com memória virtual segmentada de 32 bits.

1. Quando um programa a correr em modo utilizador pede um acesso a um endereço virtual, este é decomposto em 2 componentes: o nº de segmento, de 8 bits; e o deslocamento, de 24 bits.
   a. [0,5v] Qual a dimensão máxima de um segmento nesta máquina e qual o número máximo de segmentos que um processo pode ter?

   b. [0,5v] A decomposição acima indicada é feita por que entidade e em que modo de execução (núcleo ou utilizador)?

2. [0,6v] Quando um processo em modo utilizador pede um acesso a um dado endereço virtual, esse pedido pode implicar a passagem para modo núcleo devido a várias razões. Enumere 4 razões.

3. Considere que a memória física da máquina está ocupada pelos seguintes segmentos (cinzento: segmentos alocados; a branco: espaço livre):



   a. [0,6v] Assume que P1 tem 5 segmentos no total, sendo que apenas os indicados acima estão presentes em memória física.
   Preencha a tabela de segmentos do processo P1. No caso de não dispor de informação suficiente para preencher alguma(s) coluna(s), indique "ND".

P	Prot	Base	Limite

   b. Assuma que, após o momento ilustrado na figura, P2 pediu a reserva de dois novos segmentos, nesta ordem:
      - Reserva de segmento 2, de 190KB.
      - Reserva de segmento 3, de 90KB.

i. [0,6v] Assumindo que o algoritmo de reserva usado era o best fit, indique qual a base que seria atribuída a cada novo segmento.
Base do segmento 2: Base do segmento 3:

ii. [0,6v] Assumindo agora o algoritmo worst fit, indique qual a base que seria atribuída a cada novo segmento.
Base do segmento 2: Base do segmento 3:

iii. [0,6v] Indique uma desvantagem do algoritmo best fit e uma desvantagem do algoritmo worst fit que as suas respostas às alíneas anteriores exponham.

Grupo V [4v]

1. [0,5v] Considere um programa em Unix/Linux que já está compilado, do qual dispõe apenas do respectivo ficheiro executável e que este, ao executar-se, imprime no ecrã um conjunto de resultados. Diga se é possível executar este mesmo programa mas de forma a que os resultados por ele produzido sejam escritos num ficheiro. (Responda SIM ou NÃO e justifique de forma detalhada).

2. Para cada uma das frases seguintes, diga se está correcta ou não. (Atenção: responda SIM ou NÃO; caso responda NÃO, diga como é que a frase em causa ficaria correcta).

   a) [0,4v] No envio de mensagens, na semântica **síncrona**, o Produtor continua em execução mesmo antes do processo Consumidor receber a mensagem (*unbuffered*).

   b) [0,4v] No envio de mensagens, o Produtor envia a mensagem e continua a execução, assim que esta tenha sido armazenada no canal de forma temporária até que seja recebida pelo Consumidor (*buffered*).

c) [0,4v] No envio de mensagens, na semântica **cliente-servidor (ou pedido-resposta)**, o processo que age inicialmente como Produtor (cliente) desbloqueia-se assim que o Consumidor (servidor) receba a mensagem.

3. [0,4v] Considere o mecanimo de comunicação relativamente ao qual é dita a seguinte frase: "*A sua principal vantagem reside na facilidade de utilização uma vez que a sincronização é implícita, sendo realizada pelas funções sistema*." Qual o mecanismo? (Justifique a sua resposta).

4. [0,4v] Considere o mecanimo de comunicação relativamente ao qual é dito a seguinte frase: "*A transferência de dados é realizada recorrendo a operações directas de leitura e escrita na memória, codificadas numa linguagem de programação*." Qual o mecanismo? (Justifique a sua resposta).

5. Considere o peseudo-código seguinte:

```
#define DIMENSAO 1024
char* adr;
zona zonaPartilhada;
int Mest, Esc;
semaforo SemEscravo, SemMestre;

main() {
 SemEscravo = CriarSemaforo(0);
 SemMestre = CriarSemaforo(0);
 Mest = CriarProcesso(Mestre);
 Esc  = CriarProcesso(Escravo);
}
```

```
void Mestre () {
      zonaPartilhada = CriarZona ("MemPar", DIM);
      adr = MapearZona(zonaPartilhada);
      for (; ;) {
   ProduzirEnviarInformação();
   Assinalar(SemEscravo);
   /* Outras acções */
   Esperar(SemMestre); }
}

void Escravo() {
      zonaPartilhada = AssociarZona ("MemPar");
      adr = MapearZona(zonaPartilhada);
 for (; ;) {
   Esperar (SemEscravo);
   ReceberTratarInformação();
   Assinalar (SemMestre); }
}
```

a) [0,5v] Diga qual o modelo de interação que é suportado indicando de quantos para quantos é que a comunicação é efectuada .

b) [0,5v] Seria possível não usar memória partilhada? Qual ? Como compara ambos no que diz respeito à dificuldade de programação ?

c) [0,5v] Dê um exemplo concreto no Linux de quando é que este modelo de interacção é usado. (Atenção: na sua resposta foque-se na utlização da interface operacional do sistema operativo).