Nome: Número: Curso:

Sistemas Operativos - 1ª Frequência 2011/2012

Centro de Competências de Ciências Exactas e Engenharias
Universidade da Madeira
21 de Novembro de 2011, 14 horas

Este exame é **sem consulta**. Acetatos, livros, computadores, calculadoras, PDA's, telemóveis e acesso à Internet **não são permitidos**. Apenas são necessárias esferográficas azuis e/ou pretas.
A duração da frequência é de **90 minutos**, para uma cotação máxima de 20 valores (que corresponde a ¼ da nota final da cadeira). Outras informações sobre a avaliação, consultar a página da cadeira.
Leia as questões com atenção e responda nas folhas do enunciado. Aconselhamos muita atenção para o tempo despendido em cada uma delas. Quando terminar, entregue as suas respostas ao docente/vigilante, tendo a certeza que preencheu correctamente a sua identificação.

Boa Sorte!

[4] Escolha Múltipla

Assinale em cada uma das questões seguinte uma opção. A opção correcta é apenas uma e é aquela que responde ao pedido completamente. Cada questão correcta vale 0,5 valores cada errada desconta ¼. Se o valor final desta secção for negativo, a sua cotação passará para zero.

1. Qual das seguintes opções é uma funcionalidade do despacho:
   - [ ] actualiza o program counter
   - [ ] carregar o contexto de hardware
   - [ ] escolhe o processo da lista de executáveis
   - [ ] guardar o contexto de hardware
   - [ ] todas as opções anteriores

2. Na criação de um processo filho, e relativamente ao espaço de endereçamento:
   - [ ] o processo filho obtém o seu espaço de endereçamento, com dados
   - [ ] o processo pai cede metade do seu espaço de endereçamento
   - [ ] o processo filho obtém o seu espaço de endereçamento, se necessário
   - [ ] o processo filho obtém o seu espaço de endereçamento, vazio
   - [ ] nenhuma das anteriores

3. Qual elemento não pertence a uma arquitectura de um sistema operativo:
   - [ ] gestor de memória
   - [ ] sistema de ficheiros
   - [ ] gestor de periféricos
   - [X] gestor de serviços
   - [ ] todos os anteriores

4. Indique qual das seguintes opções é um ambiente de execução do Windows 2000.
   - [ ] as *Dynamic Loadable Libraries* (DLL)
   - [X] a interface Posix
   - [ ] o *explorer*
   - [ ] um ficheiro *.exe*
   - [ ] nenhuma das anteriores

5. A execução das intruções de um programa entre as primitivas *fechar(trinco)* e *abrir (trinco)*, implica:
   - [ ] apenas um processo a executar essas instruções
   - [ ] a sua execução atomicamente
   - [ ] o processador entrar em modo *kernel*
   - [ ] o sistema operativo esperar
   - [ ] todas as opções anteriores

6. Num sistema mono-processador temos qual das seguintes situações normalmente:
   - ☐ Multiprogramação e Paralelismo real
   - ☐ Monoprogramação e Pseudoconcorrência
   - ☐ Multiprogramação e Pseudoparalelismo
   - ☐ Um processo e Pseudoconcorrência
   - ☐ nenhuma das anteriores

7. A sequência de caracteres '-rw-r--r--' indica que um qualquer utilizador:
   - ☐ pode ler e escrever sobre o ficheiro
   - ☐ ler sobre o ficheiro
   - ☐ pode ler e escrever na directoria
   - ☐ pode aceder à directoria
   - ☐ nenhuma das opções anteriores

8. Um exemplo de uma solução baseada em software para o problema da secção crítica é:
   - ☐ *Compare and Shop*
   - ☐ *Test and Set*
   - ☐ *o algoritmo de Banker*
   - ☐ o algoritmo de *Peterson*
   - ☐ todas as anteriores

[1,5+1,5] Conceitos e Definições

Defina cada um de termos apresentado abaixo e apresente, de forma clara e concisa, as principais diferenças.

9. 'trinco' e 'semáforo'

10. 'processo' e 'tarefa'

[1,5+1,5] Processos e tarefas

11. O que são Rotinas Assíncronas? Qual a sua importância? Indique alguns exemplos da sua utilização.

12. Indique, pelo menos, duas razões ou situações que provoquem a passagem de um processo de 'em execução' para 'executável'. Em que outros estados podem os processos ser colocados?

[1,5+2,5] Gestor de Processos

13. Indique e descreva sucintamente as três principais componentes do Gestor de Processos.

14. Suponha que num sistema operativo estão a correr 2 processos. O processo P2 é CPU-Intensivo (não executa operações de E/S) e o processo P1 executa o seguinte código:

```
main(){
  while(!terminado){
      trabalhoCPU(); // Requer uma unidade de tempo de CPU
      trabalhoCPU(); // Requer uma unidade de tempo de CPU
      efectuaES();   // Operacao de E/S que bloqueia o processo durante
                     //duas unidades de tempo
      trabalhoCPU(); // Requer uma unidade de tempo de CPU
  }
}
```

Os processos têm as seguintes características:

Processo	Prioridade base	Instante do Início da Execução	Tempo Total de CPU que o Processador irá consumir na sua execução
P1	10	0	6
P2	11	3	3

Suponha que o algoritmo de escalonamento utilizado é preemptivo, com prioridades dinâmicas (em que um valor numérico de prioridade mais elevado corresponde a um processo mais prioritário). A prioridade de um processo começa por ser a sua prioridade base, sofrendo um aumento de três unidades sempre que o processo sai do estado de bloqueado. Após este aumento, a prioridade do processo diminui uma unidade após cada *quantum* (*time-slice*) em que o processo tenha estado em execução, até atingir novamente a prioridade base. Se dois processos tiverem igual prioridade, o escalonador escolhe o que não é executado há mais tempo.

Preencha a seguinte tabela, indicando, para cada instante de tempo, qual o estado de cada processo do sistema, e a respectiva prioridade no início do *quantum*.

Use a seguinte notação: Letra E – Em execução; Letra B – Bloqueado; Letra V – Executável; Entrada em Branco – processo não está activo no sistema (não iniciou ou já terminou).

Note que na solução correcta não é necessário utilizar mais espaço do que o fornecido.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P1	**E 10**														
P2															

[6] Sincronização

15. Considere um parque de estacionamento de uma empresa com capacidade máxima para MAX viaturas. Existem dois tipos de utilizadores deste parque: administradores e funcionários. Enquanto há lugares livres no parque, a ordem de entrada é por ordem de chegada. A partir do momento em que o parque fica cheio, a entrada de viaturas fica condicionada à saída de outras, devendo ser dada prioridade às viaturas dos administradores.

```
carro() {
        entrarParque(tipo);
        // o carro esta estacionado
        sairParque();
}
```

Utilizando semáforos, implemente em pseudo-código C as funções `entrarParque(int tipo)` e `sairParque()`. Caso não implemente a solução completa, identifique os aspectos não implementados.