Eng.ª Informática - Cadeira de Sistemas Operativos

1º Semestre 2007-2008

Avaliação Contínua

1. Identificação do Aluno

Número	Nome	Turma

2. Teste de Selecção Múltipla

Assinale as respostas correctas em cada um dos pontos seguintes. Note que a pontuação máxima só será obtida se todas as respostas correctas forem assinaladas.

2.1 O **Vector de Interrupções** de um processador permite: *(1,0)*

A. Realizar a gestão de periféricos no sistema operativo
B. Associar uma rotina de tratamento a um nível de interrupção
C. Modificar o modo de funcionamento do processador
D. Associar uma rotina de tratamento a um tipo de excepção
E. Realizar transferências de dados directas para a memória do sistema

2.2 O Funcionamento em **Dual Mode** permite ao Sistema Operativo: *(1,0)*

A. Realizar uma comutação de contexto mais rápida
B. Aceder directamente aos dados das aplicações
C. Proteger o espaço do sistema contra acessos erráticos de aplicações
D. Implementar os system calls em modo supervisor
E. Gerir a noção de tempo do sistema

2.3 Os **System Calls** fazem parte de um mecanismo que permite definir: *(1,0)*

A. Uma interface de acesso às funcionalidades aplicacionais
B. Uma forma de gerir o acesso aos periféricos
C. Uma interface de programação fornecida pelo sistema operativo
D. Uma forma de optimização do desempenho do sistema
E. Uma forma padronizada de aceder às funcionalidades do sistema

2.4 A **syscall table** é uma tabela que define: *(1,0)*

A. O número associado a cada system call
B. A localização de cada sytem call
C. A rotina que implementa cada system call
D. O número total de system calls
E. A forma de aceder aos parâmetros de cada system call

2.5 A **arquitectura** dos sistemas **Unix clássicos** é de tipo: *(1,0)*

A. Micro-núcleo
B. Monolítica
C. Modular
D. Máquina Virtual

2.6 O **PCB** - **Process Control Block** é: *(1,0)*
A. Um dos elementos que constituem um processo utilizador
B. Um bloco de controlo de processos
C. Uma zona de memória onde são guardadas as características dos processos
D. Uma zona onde estão armazenados os processos do sistema
E. A lista de todos os processos do sistema

2.7 Um **processo** pode estar nos seguintes **estados**: *(1,0)*
A. New, Idle, Ready, Running, Waiting, Terminated
B. New, Ready, Running, Waiting, Sleeping, Terminated
C. New, Ready, Running, Waiting, Terminated
D. New, Ready, Running, Waiting, Killed, Terminated
E. New, Ready, Active, Running, Waiting, Terminated

2.8 Um **processo** pode passar do estado **Running** para o estado **Ready** por: *(1,0)*
A. Ter pedido a realização de uma operação de I/O
B. Ter terminado a sua execução
C. Ter excedido o seu quantum de tempo
D. Ter entrado numa secção crítica ser ter conseguido acesso exclusivo
E. Ter recebido uma excepção

2.9 O *system call* **fork**() é uma chamada ao sistema que permite
A. Criar um processo filho que partilha o espaço memória com o pai
B. Criar uma nova thread dentro do espaço memória do pai
C. Criar um processo filho que utiliza uma cópia do espaço memória do pai
D. Criar um processo filho que partilha parte do espaço memória do pai
E. Criar um processo filho que tem um espaço memória diferente do pai

2.10 A utilização de **threads** permite: *(1,0)*
A. Criar vários espaços memória distintos
B. Implementar aplicações interactivas de forma mais expedita e ligeira
C. Optimizar o desempenho das aplicações dos utilizadores
D. Criar vários fluxos de execução dentro dum mesmo processo
E. Implementar aplicações mais compactas

2.11 Várias **threads** de um **mesmo processo** partilham: *(1,0)*
A. O segmento de código do processo
B. A pilha de execução
C. O segmento de dados globais
D. O contexto de execução
E. Os ficheiros e periféricos abertos

2.12 Três processos P1 P2 e P3 com tempos de execução previstos de 80, 60 e 100 ms,

num algoritmo de scheduling **Shortest Job First** são executados na ordem: *(1,0)*

A. P1, P2, P3
B. P3, P2, P1
C. P3, P1, P2
D. P2, P1, P3
E. P1, P3, P2

2.13 Um algoritmo de scheduling de tipo **Round-Robin** com *time-quantum: (1,0)*

A. Executa sempre o processo de maior prioridade
B. Interrompe sempre o processo em curso para dar o CPU ao processo seguinte
C. Espera pelo fim do *time-quantum* em curso e dá o CPU ao processo seguinte
D. Espera pelo fim do processo activo e dá o CPU ao processo seguinte
E. Se o processo activo entrar em espera, dá o CPU ao processo seguinte

2.14 Num sistema **multiprocessador simétrico**, pode ser mais vantajoso utilizar: *(1,0)*

A. Uma única Ready Queue para todos os processadores
B. Uma Ready Queue por processador
C. Uma Ready Queue por cada par de processadores
D. Duas ready Queues por processador
E. Nenhuma das soluções acima indicadas

2.15 No sistema **Linux**, o algoritmo geral de scheduling: *(1,0)*

A. Utiliza Round-Robin com time slice variável em cada nível de prioridade
B. Executa sempre o processo de maior prioridade
C. Utiliza First Come First Served nos processos Real-Time
D. Utiliza 140 níveis de prioridade distintos
E. Permite a modificação de alguns níveis de prioridade pelo utilizador

2.16 Uma **secção crítica** é: *(1,0)*

A. Uma zona de código crítica para o desempenho de um processo
B. Um componente essencial do espaço de endereçamento de um processo
C. Uma zona de código que só pode ser executada por uma thread de cada vez
D. Uma parte do contexto de um processo
E. Nenhuma das opções indicadas

2.17 As soluções para **sincronizar** threads em sistemas **multiprocessador** são: *(1,0)*

A. Utilizar algoritmos baseados unicamente em software
B. Desactivar as interrupções do scheduler
C. Utilizar algoritmos baseados em instruções específicas do hardware
D. Utilizar relógios independentes em cada processador
E. Utilizar ciclos de espera activa

2.18 Um **semáforo binário** é uma abstracção que permite: *(1,0)*

A. Sincronizar dois fluxos de execução concorrentes
B. Sincronizar N fluxos de execução concorrentes
C. Gerir a utilização de N recursos por várias threads concorrentes
D. Gerir a utilização de um recurso por várias threads concorrentes
E. Garantir que não existem *deadlocks* nos programas

2.19 Uma **Unidade de Gestão de Memória** é um dispositivo que permite: *(0,5)*

A. Acelerar o acesso do processador à memória
B. Realizar o mapeamento entre endereços lógicos e físicos
C. Guardar os dados mais utilizados pelo processador numa memória rápida
D. Proteger o espaço de endereçamento dos processos
E. Garantir que os processos possam ser carregados em qualquer zona da memória física

2.20 A **segmentação** é uma técnica de gestão da memória que permite: *(0,5)*

A. Separar os processos em zonas com conteúdos e características distintas
B. Implementar memória virtual
C. Carregar um processo em zonas de memória não contíguas
D. Partilhar zonas de código executável entre vários processos
E. Evitar os problemas da fragmentação

2.21 A **paginação** é uma forma de gestão da memória que permite: *(0,5)*

A. Separar os processos em zonas com conteúdos e características distintas
B. Implementar memória virtual
C. Carregar um processo em zonas de memória não contíguas
D. Partilhar zonas de código executável entre vários processos
E. Evitar os problemas da fragmentação

2.22 Tabelas de **Páginas Hierárquicas** são utilizadas para: *(0,5)*

A. Garantir a coerência do conteúdo das tabelas
B. Evitar o armazenamento de informação sobre todas as páginas possíveis
C. Acelerar o processo de mapeamento de endereços
D. Aumentar o espaço de endereçamento disponível
E. Nenhuma das opções indicadas