Arquitetura e Implementação de Sistemas de Operação

Ano letivo 2014/2015
1º teste – Duração: 2h00

16 de Abril de 2015

Nome: _____ **Número:** _____

Teste sem consulta. Explicite todas as assunções que fizer. A cotação está dividida equitativamente pelas questões.

1. A maior parte dos sistemas operativos (SOs) reservam parte do espaço de endereçamento de cada processo para referenciar estruturas de dados do próprio SO.
   (a) Justifique o porquê desta decisão de desenho.
   (b) Que desvantagens traz ao nível do consumo total - de todos os processos - de memória?
   (c) Que alternativas existem e porque é que não são amplamente utilizadas?
2. Explique as diferenças entre os conceitos de interrupção e de sinal. Faz sentido utilizar sinais como forma de comunicação assíncrona entre dois *threads* utilizador? e entre dois *threads* do SO?
3. No contexto do tratamento de interrupções:
   (a) Porque é que o SO utiliza pilhas (*stacks*) dedicadas para guardar o estado da execução das rotinas de tratamento de interrupções, em vez de utilizar a pilha do *thread* interrompido?
   (b) Porque é que, normalmente, o SO aloca uma das tais pilhas dedicadas por cada *thread*?
   (c) Ilustre graficamente o estado das pilhas utilizador e *kernel* antes, durante e depois da execução de uma rotina de tratamento de interrupção.
4. Muitas plataformas de suporte à virtualização executam sobre um SO de uso geral (ex: Windows ou Linux). Justifique esta opção, apresentando também as suas desvantagens.
5. Por norma os SOs providenciam suporte à criação (e gestão) de processos e de *threads*.
   (a) Genericamente, que informação o SO precisa de guardar para suportar cada uma destas abstrações?
   (b) Que nome se dá às estruturas que guardam essa informação?
   (c) Como é que estas estruturas se relacionam (inter-referenciam) na implementação do SO?
6. Quais das seguintes abstrações não pode ser escalonada pelo SO? (a) processo (b) *thread* do SO (*kernel thread*) (c) *thread* utilizador (*user thread*). Tenha em consideração todos os modelos de mapeamento de *user threads* em *kernel threads* ou processos.
7. Considere um escalonador de threads que implementa uma política *round-robin* suportada por um única fila com comportamento FIFO (First-In-First-Out) e com suporte para preempção. Acha que este escalonador é o mais indicado para escalonar cargas de trabalho (*workloads*) compostas por uma mistura equilibrada de *threads* interativos e não interativos? Justifique. Se a resposta for negativa, que alterações propõe, não negligenciando o requisito de justiça?