

Gestão do Processador

1. A modelação do ambiente de multiprogramação através da activação e desactivação de um conjunto de processadores virtuais, cada um deles associado a um processo particular, supõe que dois factos essenciais relativos ao comportamento dos processos envolvidos sejam garantidos. Quais são eles?
2. Qual é a importância da tabela de controlo de processos (*PCT*) na operacionalização de um ambiente de multiprogramação? Que tipo de campos devem existir em cada entrada da tabela?
3. O que é o *scheduling* do processador? Que critérios devem ser satisfeitos pelos algoritmos que o põem em prática? Quais são os mais importantes num sistema multiutilizador de uso geral, num sistema de tipo *batch* e num sistema de tempo real?
4. Descreva o diagrama de estados do *scheduling* do processador em três níveis. Qual é o papel desempenhado por cada nível? Num sistema de tipo *batch* multiprogramado fará sentido a existência de três níveis de *scheduling*?
5. Os estados *READY-TO-RUN* e *BLOCKED*, entre outros, têm associadas filas de espera de processos que se encontram nesses estados. Conceptualmente, porém, existe apenas uma fila de espera associada ao estado *READY-TO-RUN*, mas filas de espera múltiplas associadas ao estado *BLOCKED*. Em princípio, uma por cada dispositivo ou recurso. Porque é que é assim?
6. O que é uma *comutação de contexto*? Descreva detalhadamente as operações mais importantes que são realizadas quando há uma comutação de contexto.
7. Classifique os critérios devem ser satisfeitos pelos algoritmos de *scheduling* segundo as perspectivas sistémica e comportamental, e respectivas subclasses. Justifique devidamente as suas opções.
8. Distinga disciplinas de prioridade estática das de prioridade dinâmica. Dê exemplos de cada uma delas.
9. Num sistema de operação multiutilizador de uso geral, há razões diversas que conduzem ao estabelecimento de diferentes classes de processos com direitos de acesso ao processador diferenciados. Explique porquê.
10. Entre as políticas de *scheduling preemptive* e *non-preemptive*, ou uma combinação das duas, qual delas escolheria para um sistema de tempo real? Justifique claramente as razões da sua opção.
11. Justifique claramente se a disciplina de *scheduling* usada em Linux para a classe *SCHED_OTHER* é uma política de prioridade estática ou dinâmica?
12. O que é o *aging* dos processos? Dê exemplos de duas disciplinas de *scheduling* com esta característica, mostrando como ela é implementada em cada caso.
13. Distinga *threads* de processos. Assumindo que pretende desenvolver uma aplicação concorrente usando um dos paradigmas, descreva o modo como cada um afecta o desenho da arquitectura dos programas associados.
14. Indique justificadamente em que situações um ambiente *multithreaded* pode ser vantajoso.
15. Que tipo de alternativas pode o sistema de operação fornecer à implementação de um ambiente *multithreaded*? Em que condições é que num multiprocessador simétrico os diferentes *threads* de uma mesma aplicação podem ser executados em paralelo?
16. Explique como é que os *threads* são implementados em Linux.
17. O principal problema da implementação de *threads*, a partir de uma biblioteca que fornece primitivas para a sua criação, gestão e *scheduling* no nível utilizador, é que quando um *thread* particular executa uma chamada ao sistema bloqueante, todo o processo é bloqueado, mesmo que existam *threads* que estão prontos a serem executados. Será que este problema não pode ser minimizado?