Conectividade em Sistemas Ciberfísicos

Lista de Exercícios I

Exercícios sobre Introdução a S.O, Processos e Threads

Exercício 1: Relacione corretamente as definições dos mecanismos de gerenciamento de recursos do S.O.

(4) Cria um modelo de abstração para escrita e leitura em dispositivos HDD (Hard Disk Drive) e SSD (Solid State Drive).

(2) Evita que um programa escreva sobre as variáveis de outro programa.

(1) Determina em qual core de um processador um programa será executado.

(3) Gerencia a escrita e leitura de dispostivos manipulados diretamente pelo usuário.

(1) Permite que mais de um programa seja executado simultaneamente no mesmo core de um processador.

(2) Utiliza a mémória de armazenamento em massa no HDD ou SDD como uma extensão da memória física em RAM.

1. Gerencia de Processos
2. Gerencia de Memória
3. Gerencia de E/S
4. Gerencia de Armazenamento
5. n.d.a

Observação: PROTEÇÃO DE MEMÓRIA

- Programas são movidos do disco para memória antes de serem executados.
- Assim a memória é usada para armazenar variáveis e também o código dos programas.
- Escrever num espaço de memória errado pode alterar o código de um programa que esteja em memória. Isso geralmente leva a um erro fatal para o programa e até mesmo para o sistema.
- Gerenciamento de memória é responsável por evitar que um programa (processo) escreva no espaço de memória usado por outro processo. Esse mecanismo também é denominado PROTEÇÃO DE MEMÓRIA
- Problemas de violação de memória eram comuns em sistemas operacionais antigos. Agora eles são raros em sistemas modernos.

Observação: Memória Virtual e Paginação

- Todo computador tem uma quantidade limitada de RAM (memória de acesso rápido, onde cada posição de memória pode ser endereçada individualmente) e uma grande quantidade de disco (e cujo acesso é lento e sequencial, isto é, não é possível endereçar as posições de memória individualmente).

- O sistema operacional usa uma boa parte da RAM do computador para executar os seus próprios processos.

- Programas dos usuários solicitam memória ao sistema operacional através de um mecanismo denominado alocação dinâmica. Muitos programas, como o Google Chrome, podem solicitar grandes quantidade de memória.

- Para evitar que um programa monopolize toda a memória RAM do computador, o sistema operacional deixa parte da memória solicitada pelo programa em uma **memória virtual**, que está em disco. Essa memória é dividida em blocos denominados **páginas**. As páginas representam uma continuação da memória RAM em disco.

- Quando o programa tenta acessar uma variável que estava na memória virtual, a **página** correspondente é movida da memória virtual para a RAM. Eventualmente, um outro programa tem parte de sua memória movida para o disco para liberar espaço.

Exercício 2. Indique as afirmações corretas sobre o gerenciamento de processos

I. Processos que efetuam operações de E/S, como leitura em disco, são colocados em estado de espera até que a operação seja completada. **VERDADEIRO**

II. A recepção de pacotes pela rede, através de chamadas de sockets, são exemplos de operação que colocam processos em estado de espera. **VERDADEIRO**

III. Processos que realizam muitas operações de E/S consomem muito tempo de CPU. **FALSO:**
As operações de E/S (como leitura do teclado ou escrita em disco) são muito lentas quando comparadas com a execução de instruções pela CPU. Em estado de espera, os processos não consomem CPU.

IV. Processos relacionados a programas de inteligência artificial, como treinamento de redes neurais, realizam poucas operações de E/S e por isso consome muito tempo de CPU enquanto estão ativos. **VERDADEIRO**

V. Programas que estão em estado de espera não consomem CPU, mas eles consomem memória até que sejam terminados. **VERDADEIRO**

Exercício 3. Indique as afirmações corretas sobre a diferença entre escalonamento cooperativo e preemptivo

I. O algoritmo que determina em um dado momento qual processo irá ganhar tempo de CPU é denominado **algoritmo de escalonamento:** **VERDADEIRO**

II. O termo **troca de contexto** refere-se a mudança de estado de uma CPU quando o processo que está sendo executado é substituído por outro. **VERDADEIRO**

III. No escalonamento **cooperativo** (ou não preemptivo), a troca de contexto acontece apenas quando um processo executa uma operação de E/S. **PARCIALMENTE VERDADEIRO.**
O programa pode colocar-se em estado de espera por um certo tempo através de uma chamada do tipo SLEEP. O termo COOPERATIVO refere-se ao fato do programa COOPERAR (passar a vez) para outro programa através de uma chamada de SLEEP.

IV. No escalomanto **preemptivo**, a troca de contexto ocorre através de interrupções, que limitam o tempo que um processo pode ocupar tempo de CPU. **VERDADEIRO**

V. O escalonamento **cooperativo** pode ser **mais eficiente** que o escalonamento preemptivo, uma vez que ocorrem menos trocas de contexto. **VERDADEIRO**
A troca de contexto não é instantânea e precisa que o sistema operacional execute várias operações para remover um processo do contexto e colocar outro no lugar. No escalonamento preemptivo essa troca ocorre muitas vezes por segundo.

Exercício 4. Indique as diferenças entre processos e threads.

(3) Pode ser executado em cores diferentes de um processador.

(2) Compartilham o mesmo espaço de memória.

(2) Podem compartilhar variáveis em um mesmo programa.

(2) Permite que funções de um mesmo programa sejam executadas de forma paralela.

(1) Pode ser executado em computadores diferentes.

(2) Permite que a troca de contexto seja executada de forma relativamente rápida.

1. Processo
2. Thread
3. Ambos
4. Nenhum dos dois

Obsvervação: Programa vs Processo vs Thread

- O termo **programa** designa uma entidade passiva que não está sendo executada.
- Um **processo** é um programa ativo, isto é, um programa em execução.
- Em Python é possível mapear funções em processos, dando a impressão que elas fazem parte de um mesmo programa. Contudo isso é apenas um artifício de programação. Cada função irá se comportar como um **programa independente**, tendo seu próprio espaço de memória.
- **Threads** são divisões de um processo, funções que podem ser executadas de forma paralela, e compartilham o **mesmo espaço de memória** do processo onde foram criadas.
- Quando as threads são criadas usando **chamadas do sistema operacional**, elas podem ser alocadas a diferentes cores de um processador, como se fossem processos independentes.
- Algumas implementações de threads em ambientes de programação, como Python, precisam que threads de um mesmo programa fiquem no mesmo core. Nesse caso, se diz que as threads foram criadas no **espaço do usuário**, e não no **Kernel**.

Exercício 5. Considerando as características dos processos e threads relacione as colunas.

(2) Desenvolver um servidor TCP onde o número de clientes conectados a cada instante é muito variável.
A criação e destruição de processos é muito mais custosa que a das threads.

(5) Desenvolver um programa com um número fixo de componentes (funções) que são executados de forma paralela.
Essa resposta é valida apenas se não houver necessidade de compartilhamento de variáveis entre as funções (como no caso de diferentes janelas em um browser)

(3) Desenvolver um programa com um número fixo de componentes (funções) que são executados de forma paralela, e a falha de um componente não pode afetar os demais.
Por usarem o mesmo espaço de memória, certas falhas causadas por uma thread pode abortar todo o processo.

(5) Utilizar os recursos de um processador com múltiplos cores.

(3) Desenvoler um programa cujos componentes são executados de forma paralela em computadores diferentes.

1. Apenas Threads
2. Melhor Threads
3. Apenas Processos
4. Melhor Processos
5. Indiferente
6. Nenhum dos dois