Sistemas Operacionais

Caroliny Abreu Teixeira

1. O que são Processos? Como são classificados? Qual a diferença entre um Processo e um Programa? Explique;

   R: Processos são entidades fundamentais em um sistema operacional. Um processo representa a execução de um programa em um ambiente controlado pelo sistema operacional. Ele inclui o código do programa, dados, pilha de execução e recursos associados, como arquivos abertos e informações de contexto.

   Classificação de Processos: Os processos podem ser classificados de várias maneiras, incluindo:

   Processos em Primeiro Plano e Segundo Plano:

   Processo em Primeiro Plano: Interativo, recebe entrada do usuário diretamente e está visível para o usuário.

   Processo em Segundo Plano: Executa em segundo plano, sem interação direta com o usuário. Também chamado de processo em segundo plano ou processo em background.

   Processos de Usuário e Processos do Sistema:

   Processos de Usuário: Iniciados e controlados por usuários.

   Processos do Sistema: Iniciados e controlados pelo sistema operacional para tarefas internas.

   Diferença entre Processo e Programa:

   Programa: É um conjunto de instruções e dados armazenados em um arquivo no disco rígido. É uma entidade estática.

   Processo: É uma instância em execução de um programa. Ele inclui o programa, mas também o contexto de execução, como valores de registradores, contador de programa e outros recursos.

2. Quais são os possíveis estados de um Processo? Explique sucintamente como o processo se encontra em cada estado;
   R: **Pronto (Ready):**
   Neste estado, o processo está pronto para ser executado, mas ainda não foi escalonado pela CPU.
   Ele aguarda na fila de processos prontos para ser escolhido para a execução.

   Execução (Running):
   O processo está sendo executado na CPU.
   Ele está ocupando o tempo da CPU e executando suas instruções.

Bloqueado (Blocked ou Waiting):
Um processo entra neste estado quando precisa aguardar por algum evento, geralmente uma operação de entrada/saída (E/S) como leitura de arquivo.
O processo não pode continuar sua execução até que o evento necessário ocorra.

Terminado (Terminated):
Neste estado, o processo concluiu sua execução ou foi encerrado por alguma razão.
O sistema operacional pode liberar os recursos alocados pelo processo.

3. O que são ***Filas de Scheduling***? Como são subdivididas? Explique;
   R: Filas de Scheduling são estruturas de dados usadas pelos sistemas operacionais para organizar e gerenciar os processos em diferentes estados de execução. Elas permitem que o sistema operacional faça a seleção adequada de qual processo deve ser executado a seguir, com base em algoritmos de escalonamento. As Filas de Scheduling são subdivididas principalmente em duas categorias:

   Fila de Prontos (Ready Queue):
   Essa fila mantém os processos que estão prontos para serem executados, mas ainda aguardam a CPU.
   Os processos nesta fila estão no estado de "Pronto" e aguardam sua vez de serem escalonados para a execução.
   O escalonador do sistema operacional seleciona processos dessa fila para serem executados com base em políticas de escalonamento.

   Fila de Espera (Blocked Queue ou Wait Queue):
   Essa fila contém os processos que foram bloqueados e estão aguardando a ocorrência de algum evento externo, como uma operação de entrada/saída (E/S).
   Os processos nesta fila estão no estado "Bloqueado" e não podem executar até que o evento necessário ocorra.
   Quando o evento ocorre, o processo é movido de volta para a fila de prontos.

4. O que você entende por ***Context Switching***? E ***Overhead***?
   R: **Context Switching:**
   O Context Switching (troca de contexto) é o processo pelo qual o sistema operacional muda o controle da CPU de um processo em execução para outro processo. Isso envolve salvar o contexto do processo atual, que inclui informações como os valores dos registradores, o contador de programa e outros dados relevantes. Em seguida, o sistema operacional restaura o contexto do próximo processo na fila de prontos e permite que ele continue a execução. O Context Switching é uma operação essencial para o gerenciamento eficiente de processos multitarefa, permitindo que vários processos compartilhem a CPU de maneira aparentemente simultânea.

   Overhead:
   Overhead refere-se às atividades ou recursos adicionais necessários para realizar uma determinada tarefa, além do trabalho principal que está sendo realizado. No contexto de sistemas operacionais e processos, o overhead pode ser causado por operações que não contribuem diretamente para a execução da tarefa principal de um processo, mas são necessárias para administrar o processo e o sistema como um todo.

5. Como o Sistema Operacional consegue gerenciar cada processo em execução de forma adequada? Que tipo de estrutura é mantida por ele a fim de prover esta organização?
   R: O sistema operacional gerencia cada processo em execução por meio de uma estrutura de dados chamada Tabela de Processos (Process Control Block - PCB), que é mantida para cada processo. A Tabela de Processos contém informações detalhadas sobre o estado e o contexto de cada processo, permitindo ao sistema operacional gerenciar e controlar os processos de maneira adequada.

6. Como é estruturado um Processo na memória Principal? Explique em detalhes;

   R: Um processo na memória principal é estruturado em várias seções, cada uma das quais serve a um propósito específico. Essas seções são organizadas de maneira a permitir que o sistema operacional e o hardware do computador gerenciem a execução do processo de maneira eficiente. As principais seções de um processo na memória são:

   Texto (Código):

   Essa seção contém o código executável do programa, ou seja, as instruções que o processador deve executar.

   Geralmente é uma região de memória somente leitura para garantir que o código do programa não seja alterado durante a execução.

   Dados Inicializados:

   Essa seção contém variáveis globais e estáticas que são inicializadas com valores específicos.

   Essas variáveis podem ser acessadas e modificadas ao longo da execução do processo.

   Dados Não Inicializados (BSS - Block Started by Symbol):

   Nesta seção, são armazenadas variáveis globais e estáticas que não são inicializadas explicitamente.

   Alocar memória para essas variáveis requer menos espaço, pois elas são inicializadas com valores padrão (geralmente zero).

   Pilha (Stack):

   A pilha é usada para armazenar informações sobre a execução atual do processo, como endereços de retorno de funções, variáveis locais e parâmetros de função.

   Cresce para baixo na memória, ou seja, à medida que mais elementos são empilhados, o ponteiro de pilha é decrementado.

   Heap:

O heap é usado para alocar memória dinamicamente durante a execução do processo, geralmente por meio de chamadas de alocação de memória como malloc() em C/C++ ou new em outras linguagens.

É necessário gerenciar manualmente a alocação e liberação de memória no heap.

7. O que são **processos filhos**? Por quais razões um processo pai pode finalizar um processo filho?
   R: Processos filhos são processos que são criados por outros processos, chamados de processos pais. A criação de processos filhos é uma maneira de dividir tarefas complexas em tarefas menores e mais gerenciáveis. Os processos filhos geralmente herdam certas propriedades e recursos do processo pai, mas eles também podem ter seu próprio espaço de endereçamento e recursos exclusivos.
   O processo pai tem a responsabilidade de gerenciar seus processos filhos e decidir quando finalizá-los com base em critérios específicos que dependem das necessidades e políticas do sistema e da aplicação, como tarefa concluída, erro na execução, economia de recursos, necessidade de reinicialização, etc

8. O que você entende por ***Sockets***? Quando são utilizados?

   R: Sockets são uma abstração de programação que permite a comunicação entre processos em diferentes máquinas através de uma rede. Eles fornecem uma interface padronizada para trocar dados entre aplicativos, independentemente do sistema operacional ou da linguagem de programação utilizada. Resumindo, um socket é um ponto de extremidade da comunicação em uma rede e são utilizados para permitir a comunicação entre processos de diferentes dispositivos, mesmo que estejam separados por distâncias geográficas. Eles desempenham um papel crucial em diversas aplicações de rede.

9. Diferentes processos podem se comunicar entre si? Em caso afirmativo, quais são as técnicas utilizadas para prover esta comunicação e em que situações devem ser utilizadas?

   R: Sim, diferentes processos podem se comunicar entre si em um sistema operacional.

   Algumas das principais técnicas são:

   Memória Compartilhada:

   Várias aplicações podem compartilhar a mesma área de memória, permitindo a troca direta de dados. É rápida e eficiente, mas requer sincronização cuidadosa para evitar conflitos.

   Situações: Quando processos precisam compartilhar grandes volumes de dados e desempenho é crucial, como em processos cooperativos ou sistemas de banco de dados.

   Pipes e FIFOs (Named Pipes):

Pipes permitem a comunicação entre processos pai e filho, ou entre processos relacionados hierarquicamente. FIFOs são semelhantes, mas permitem comunicação entre processos não relacionados.

Situações: Quando há uma relação hierárquica entre processos, como em um processo pai/filho ou quando é necessária comunicação em uma única direção.

Sockets:

Como mencionado anteriormente, os sockets permitem a comunicação entre processos em diferentes máquinas através de uma rede. Eles são flexíveis e podem ser usados para uma ampla variedade de cenários.

Situações: Para comunicação em redes, entre diferentes máquinas ou processos em uma única máquina, quando a interoperabilidade é importante.

Mensagens (IPC - Inter-Process Communication):

Mensagens permitem que os processos troquem informações indiretamente, através do sistema operacional, utilizando filas de mensagens ou canais de comunicação.

Situações: Quando os processos não precisam interagir diretamente, mas precisam trocar informações de forma assíncrona.

Semáforos, Mutexes e Monitores:

Mecanismos de sincronização que controlam o acesso a recursos compartilhados, evitando condições de corrida e garantindo a coerência dos dados.

Situações: Quando há necessidade de coordenação e controle de acesso a recursos compartilhados.

RPC (Remote Procedure Call) e RMI (Remote Method Invocation):

Mecanismos que permitem que processos chamem funções ou métodos em processos remotos, como se fossem chamadas locais.

Situações: Quando é necessário realizar operações em processos remotos, como em sistemas distribuídos ou aplicações cliente-servidor.