Aula 7: Processos: Escalonamento, Operações, Comunicações

Escalonamento de Processos

O objetivo da multiprogramação é garantir que sempre haverá um processo em execução e forma a maximizar a utilização da CPU. O objetivo do compartilhamento de tempo é **partilhar a CPU entre processos**, com uma frequência tal que possibilite aos usuários interagir com cada programa enquanto está executando. Um sistema com um único processador pode ter somente um processo em execução. Se existirem mais processos, eles devem aguardar até que a CPU esteja livre e possa ser redistribuída.

À medida que os processos entram no sistema, são incluídos numa **fila de jobs**. Esta fila consiste em todos os processos que estão no sistema. Os processos residentes na memória principal que estão prontos e em espera para entrar em execução são levados para uma lista chamada **fila pronta**. Esta fila é em geral implementada na forma de uma lista encadeada. A cabeça de uma fila pronta contém ponteiros para o primeiro e o último PCB da lista.

O sistema operacional tem ainda outras filas. Quando um processo está em execução e para por uma interrupção ou entra em espera pela ocorrência de um evento particular, ele deve esperar pela resposta do dispositivo de entrada e saída. A lista de espera por um determinado dispositivo de entrada e saída é chamada de **fila de dispositivos**. Cada dispositivo tem sua própria fila.

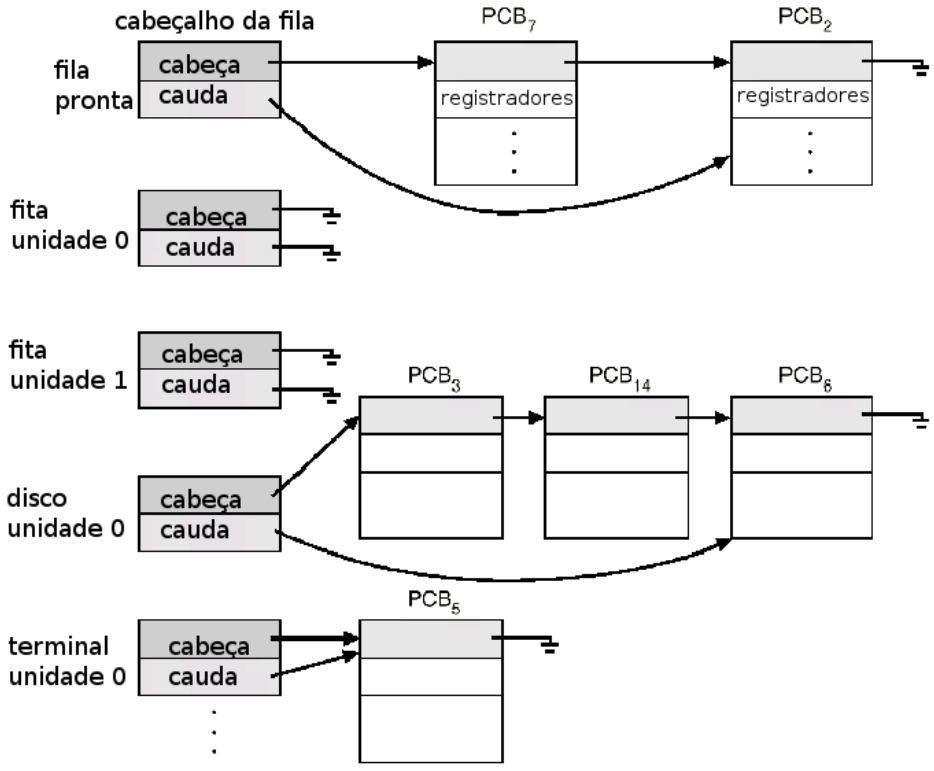

Figura 1: A fila pronta e várias filas de dispositivos de entrada e saída.

Em um sistema operacional, com frequência são submetidos mais processos do que é possível executar de imediato. Estes processos são transferidos para um dispositivo de armazenamento de massa (ordinariamente um disco), onde são reservados para execução posterior. O **escalonador de longo prazo** seleciona os processos da sua cadeia, carregando-os na memória para execução. O **escalonador de curto prazo** seleciona um dos processos prontos para execução e aloca a CPU para ele.

Operações Sobre Processos

O sistema operacional deve fornecer mecanismos para criação e eliminação de processos. A criação e a destruição de processos são realizadas através de chamadas de sistemas. No Unix, um processo é criado pela chamada *fork*(). No Windows, a chamada de criação é a função *CreateProcess*(). Para encerrar um processo no Unix, a chamada é *exit*(). Já no Windows, a função é *ExitProcess*(). No Windows, também existem outras chamadas capazes de terminar um processo, como a *TerminateProcess*().

O processo que cria outros é chamado de **processo pai**. Já os processos gerados são chamados de **processos filhos**. Sendo um processo (como qualquer outro), um processo filho também pode criar outros processos. Os seus filhos, por sua vez, podem criar outros processos. Assim, no sistema é criada uma relação de hierarquia entre processos pais e processos filhos. Essa relação é estrutura através de uma **árvore de processos**.

A peça chave para manter o controle sobre a arvore de processos é identificar os processos de maneira única. Por isso, os processos são identificados por um número inteiro chamado **PID (identificador do processo)**.

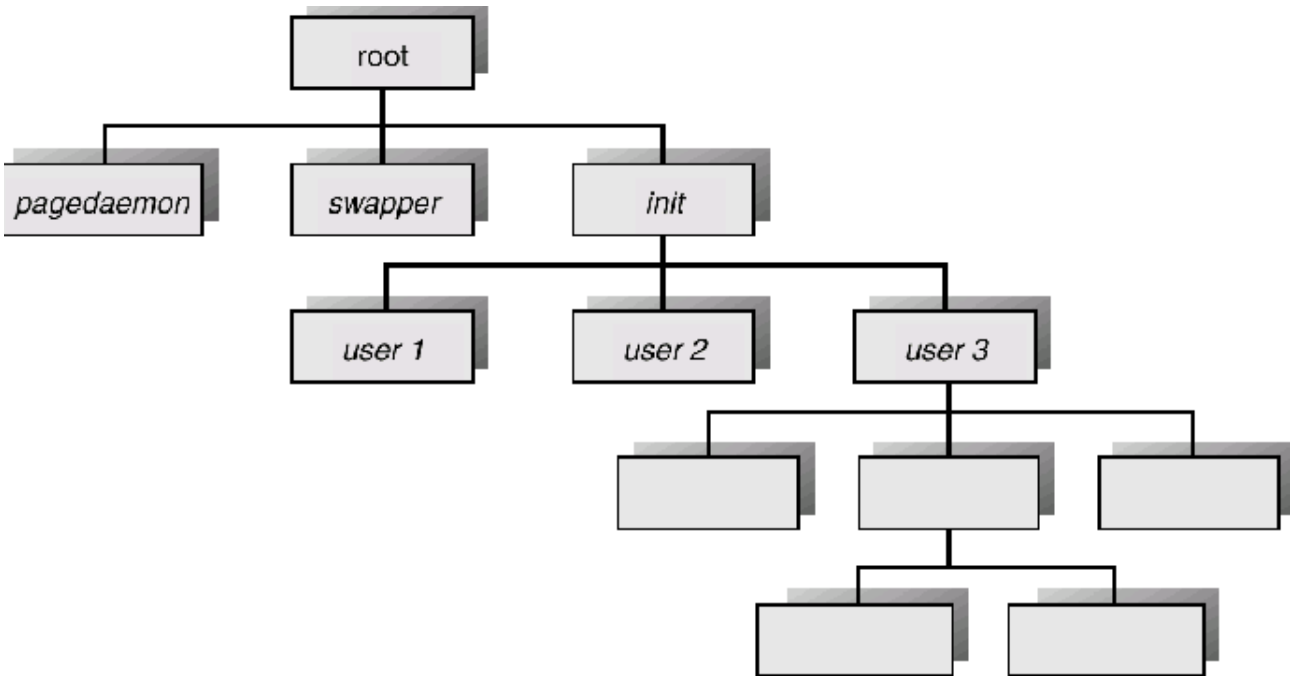

Figura 2: Uma árvore de processos em um sistema UNIX típico.

Comunicação Interprocessos

Os processos concorrentes em execução no sistema operacional podem ser tanto processos independentes como processos cooperativos. Um processo é **independente** se não puder afetar os outros processos em execução no sistema ou ser afetados por eles. Claramente, qualquer processos que não compartilhe dado (temporário ou permanente) com qualquer outro processo é independente. Por outro lado, um processo é **cooperativo** se puder afetar outros processos em execução no sistema ou ser afetados por ele.

A comunicação entre processos, ou **Inter-Process Communication (IPC)**, é o grupo de mecanismos que permite aos processos transferirem informação entre si. De forma mais abstrata, a comunicação entre processos pode ser implementada por duas primitivas básicas: enviar (dados, destino), que envia os dados relacionados ao destino indicado, e receber (dados, origem), que recebe os dados previamente enviados pela origem indicada. Essa abordagem, na qual o emissor identifica claramente o receptor e vice e versa, é denominada **comunicação direta**.

Poucos sistemas empregam a comunicação direta; na prática são utilizados mecanismos de **comunicação indireta**, por serem mais flexíveis. Na comunicação indireta, emissor e receptor não precisam se conhecer, pois não interagem diretamente entre si. Eles se relacionam através de um canal de comunicação, que é criado pelo sistema operacional, geralmente a pedido de uma das partes. Assim, as primitivas de comunicação não designam diretamente tarefas, mas canais de comunicação aos quais as tarefas estão associadas: enviar (dados, canal) e receber (dados, canal).

Em relação aos aspectos de sincronismo do canal de comunicação, a comunicação entre tarefas pode ser:

- **Síncrona**: quando as operações de envio e recepção de dados bloqueiam (suspendem) as tarefas envolvidas até a conclusão da comunicação: o emissor será bloqueado até que a informação seja recebida pelo receptor, e vice-versa.
- **Assíncrona**: em um sistema com comunicação assíncrona, as primitivas de envio e recepção não são bloqueantes: caso a comunicação não seja possível no momento em que cada operação é invocada, esta retorna imediatamente com uma indicação de erro. Deve-se observar que, caso o emissor e o receptor operem ambos de forma assíncrona, torna-se necessário criar um *buffer* para armazenar os dados da comunicação entre eles. Sem esse *buffer*, a comunicação se tornará inviável, pois raramente ambos estarão prontos para comunicar ao mesmo tempo.

Comunicação em Sistemas Cliente-Servidor

Uma definição para sistemas Cliente/Servidor seria a existência de uma plataforma base para que as aplicações, onde um ou mais Clientes e um ou mais Servidores, juntamente com o Sistema Operacional e o Sistema Operacional de Rede, executem um processamento distribuído. Um

sistema Cliente/Servidor pode ser entendido como a interação entre Software e Hardware em diferentes níveis, implicando na composição de diferentes computadores e aplicações.

Para melhor se entender o paradigma Cliente/Servidor é necessário observar que o conceito chave está na ligação lógica e não física. O Cliente e o Servidor podem coexistir ou não na mesma máquina. Porém um ponto importante para uma real abordagem Cliente/Servidor é a necessidade de que a arquitetura definida represente uma computação distribuída. Algumas das características do Cliente e do Servidor são descritas a seguir:

- **Cliente** – também denominado de “front-end” e “WorkStation”, é um processo que interage com o usuário através de uma interface gráfica ou não, permitindo consultas ou comandos para recuperação de dados e análise e representando o meio pela qual os resultados são apresentados.
- **Servidor** – também denominado de “back-end”, fornece um determinado serviço que fica disponível para todo Cliente que o necessita. A natureza e escopo do serviço são definidos pelo objetivo da aplicação Cliente/Servidor.

Exercícios

1. Descreva a diferença entre escalonar de curto prazo e longo prazo.

2. Cite 4 aplicações que são executadas sobre sistemas cliente-servidor.

3. Diferencie comunicação direta de comunicação indireta.

4. Diferencie comunicação síncrona de comunicação assíncrona.

5. Qual a relação existente entre processo pai e processos filhos?

6. Um processo ao solicitar um dispositivo de entrada e saída, ele é inserido na cabeça ou na cauda da fila do dispositivo?