UNOESC Campus Chapecó

Curso: Sistemas de Informação e Engenharia de Computação

Disciplina: Sistemas Operacionais

Profº: Jean Carlos Hennrichs

Processos

(MACHADO, MAIA 2002 e FROSI, 2010, STUART, 2011)

O conceito de processo é a base para a implementação de um sistema operacional multitarefa. O processador é projetado apenas para executar instruções, não sendo capaz de distinguir qual programa se encontra em execução. Sendo assim, a gerência de um ambiente multiprogramável (multitarefa) é uma função exclusiva do sistema operacional, que deve controlar a execução dos diversos programas e o uso concorrente do processador. Com isso, para ser executado, um programa deve estar sempre associado a um **processo**. Um processo pode ser definido como uma abstração para representar um programa em execução.

O termo processo ganhou unanimidade após o surgimento dos sistemas multiprogramáveis, tornando-se um dos conceitos mais importantes no estudo e nos projetos de sistemas operacionais. Apesar de denominações como tarefa ou job ainda serem encontradas com o mesmo sentido, o termo processo é atualmente utilizado por grande parte da bibliografia.

A gerência de processos é uma das principais funções de um sistema operacional. Através de processos, um programa pode alocar recursos, compartilhar dados, trocar informações e sincronizar sua execução com outros programas. Nos sistemas multiprogramáveis, os processos são executados concorrentemente, compartilhando, dentre outros recursos, o uso do processador, da memória principal e dos dispositivos de E/S. Nos sistemas com múltiplos processadores, não só existe a concorrência de processos pelo uso do processador, como também a execução simultânea de processos nos diferentes processadores.

5.1 – Estrutura do Processo

Um processo pode ser inicialmente entendido como um **programa em execução**, mas seu conceito é mais abrangente. Esta idéia torna-se mais clara quando pensamos em como os sistemas multiprogramáveis atendem os diversos usuários e mantém informações a respeito dos vários programas que estão sendo executados concorrentemente.

Em um sistema multiusuário, cada programa em execução é associado a um usuário. Ao executar um programa, o usuário tem a impressão de possuir o processador e todos os demais recursos reservados exclusivamente para seu uso. De fato isto não é verdade, visto que todos os recursos estão sendo compartilhados, inclusive a CPU. Nesse caso, o processador executa o programa de um usuário durante um intervalo de tempo e, no instante seguinte, poderá estar processando um outro programa, de outro usuário.

Para que a troca de programas ocorra sem problemas, é necessário que todas as informações do programa interrompido sejam guardadas para que, quando este retornar a ser executado, não lhe

falte nenhuma informação necessária à continuação do processamento. Todas as informações importantes e necessárias à execução de um programa fazem parte do processo.

Essa informações nos levam à idéia de que um processo pode ser melhor definido como o **ambiente onde um programa é executado**. Este ambiente, além das instruções do programa propriamente dito, possui também informações adicionais de apoio ao S.O., como o quanto de recursos do sistema esse programa pode utilizar, qual o espaço de endereçamento utilizado, tamanho do *time-slice*, quotas de disco, identificação do processo, entre muitas outras informações.

A execução de um mesmo programa pode então variar dependendo do processo no qual ele é executado, ou seja, em função dos recursos disponíveis. A falta de recursos pode impedir a execução com sucesso de um programa. Caso um programa, por exemplo, necessite utilizar uma área em disco superior ao seu limite, o sistema operacional irá interromper sua execução por falta de recursos disponíveis.

Um processo é formado por três partes, conhecidas como contexto de hardware, contexto de software e espaço de endereçamento, que juntas mantêm todas as informações necessárias à execução de um programa (Figura 5.1).

Figura 5.1: Estrutura do processo.

5.1.1 – Contexto de Hardware

O **contexto de hardware** armazena o conteúdo dos registradores gerais da CPU, além dos registradores de uso específico, como program counter (PC), stack pointer (SP) e registrador de status (PSW). Quando um programa está em execução, o seu contexto de hardware está armazenado nos registradores do processador; no momento em que o programa perde a utilização da CPU, o sistema salva as informações no contexto de hardware do processo (em memória).

O contexto de hardware é fundamental para a implementação dos sistemas multiprogramáveis, onde os processos se revezam na utilização da CPU, podendo ser interrompidos e, posteriormente, restaurados. A troca de um processo por outro no processador, comandada pelo sistema operacional, é denominada **mudança de contexto**. A mudança de contexto consiste em salvar o conteúdo dos registradores do programa que está deixando a CPU e carregá-los com os valores referentes ao do novo programa que será executado. Essa operação resume-se em substituir o contexto de hardware de um processo pelo de outro (Figura 5.2).

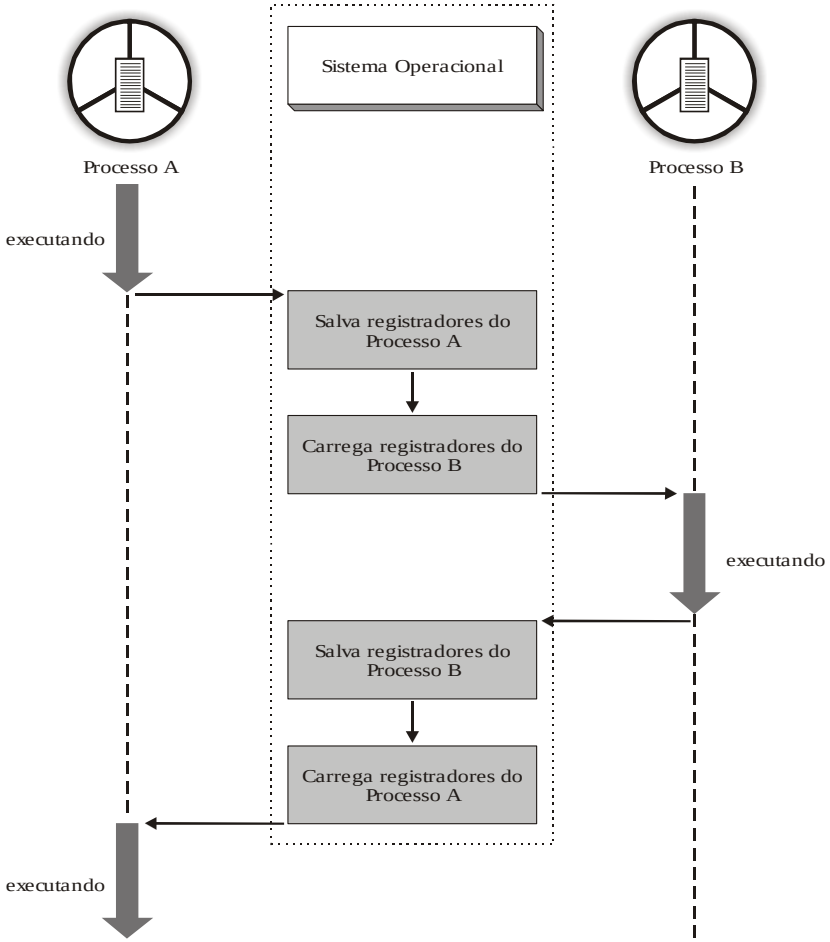

Figura 5.2: Mudança de contexto.

5.1.2 – Contexto de Software

No **contexto de software** são especificadas características e limites dos recursos que podem ser alocados pelo processo, como o número máximo de arquivos abertos simultaneamente, prioridade de execução, quantidade máxima de memória alocada, entre outros. Muitas destas características são determinadas no momento da criação do processo, enquanto outras podem ser alteradas durante sua existência.

A maior parte das informações do contexto de software do processo é proveniente de configurações do próprio sistema operacional, que especifica os limites dos recursos que cada processo pode alocar. Outras informações presentes no contexto de software são geradas dinamicamente ao longo da execução do processo.

O contexto de software é composto por três grupos principais de informações sobre o processo: identificação, quotas e privilégios.

5.1.2.1 – Identificação

Cada processo criado pelo sistema recebe uma identificação única (**PID** – **process identification**) representada por um número. Através do PID, o sistema operacional e outros processos podem fazer referência a qualquer processo existente, consultando seu contexto ou

alterando uma de suas características. Alguns sistemas, além do PID, identificam o processo através de um nome.

O processo também possui a identificação do usuário ou processo que o criou (owner). Cada usuário possui uma identificação única no sistema (**UID – user identification**), atribuída ao processo no momento de sua criação. A UID permite implementar um modelo de segurança, onde apenas os objetos (processos, arquivos, dispositivos, etc.) que possuem a mesma UID do usuário que criou o processo podem ser acessados.

5.1.2.2 – Quotas

As quotas são os limites de cada recurso do sistema que um processo pode alocar. Caso uma quota seja insuficiente, o processo poderá ser executado lentamente, interrompido durante seu processamento ou mesmo não ser executado. Alguns exemplos de quotas presentes na maioria dos sistemas operacionais são:

- Número máximo de arquivos abertos simultaneamente;
- Tamanho máximo de memória principal e secundária que o processo pode alocar;
- Número máximo de operações de *E/S* pendentes;
- Tamanho máximo do buffer para operações de *E/S;*
- Número máximo de processos, subprocessos e threads que podem ser criados.

5.1.2.3 – Privilégios

Os privilégios ou direitos definem as ações que um processo pode fazer em relação a ele mesmo, aos demais processos e ao sistema operacional.

Privilégios que afetam o próprio processo permitem que suas características possam ser alteradas, como prioridade de execução, limites alocados na memória principal e secundária etc. Já os privilégios que afetam os demais processos permitem, além da alteração de suas próprias características, alterar as de outros processos.

Privilégios que afetam o sistema são os mais amplos e poderosos, pois estão relacionados à operação e gerência do ambiente, como a desativação do sistema, alteração de regras de segurança, criação de outros processos privilegiados, modificação de parâmetros de configuração do sistema, entre outros. A maioria dos sistemas operacionais possui uma conta de usuário especial que pode executar processos com todos estes privilégios disponíveis, com o propósito de administrar o sistema operacional. No sistema Unix existe a conta "root", no Windows Server a conta "Administrator" e no Open VMS existe a conta "system" com este perfil.

5.1.3 – Espaço de Endereçamento

O **espaço de endereçamento** é a área de memória pertencente ao processo onde as instruções e os dados do programa são armazenados para execução. Cada processo possui seu

próprio espaço de endereçamento, que deve ser devidamente protegido do acesso dos demais processos. Posteriormente serão analisados diversos mecanismos de implementação e gerência do espaço de endereçamento.

A Figura 5.3 ilustra as características da estrutura de um processo.

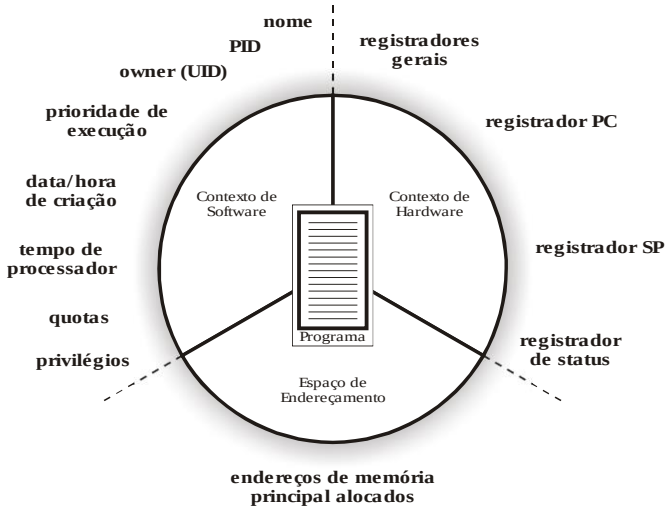

Figura 5.3: Características da estrutura de um processo.

5.1.4 – Bloco de Controle do Processo

O processo é implementado pelo sistema operacional através de uma estrutura de dados chamada **bloco de controle do processo** (**Process Control Block - PCB**). A partir do PCB, o sistema operacional mantém todas as informações sobre o contexto de hardware, contexto de software e espaço de endereçamento de cada processo (Figura 5.4).

Os PCBs de todos os processos residem na memória principal em uma área exclusiva do sistema operacional. O tamanho desta área geralmente é limitado por um parâmetro do sistema operacional que permite especificar o número máximo de processos que podem ser suportados simultaneamente pelo sistema.

Toda a gerência dos processos é realizada através de system calls, que realizam operações como criação, alteração de características, visualização, eliminação, sincronização, suspensão de processos, dentre outras.

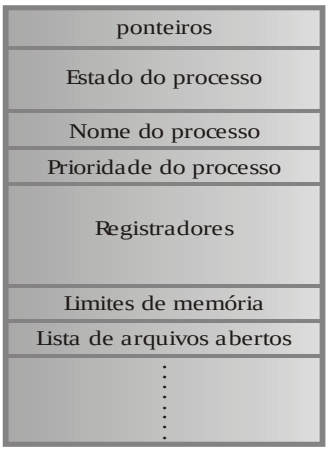

Figura 5.4: PCB – Process Control Block.

5.2 – Estados do Processo

Em um sistema multiprogramável, um processo não deve alocar a CPU com exclusividade, pois deve existir um compartilhamento no uso do processador. Os processos passam por diferentes estados ao longo do seu processamento, em função de eventos gerados pelo sistema operacional ou pelo próprio processo. Um processo ativo pode encontrar-se em três diferentes estados:

5.2.1 – Execução (running)

Um processo é dito no **estado de execução** quando está sendo processado pela CPU. Em sistemas com apenas uma CPU, somente um processo pode estar sendo executado em um dado instante. Os processos se alternam na utilização do processador seguindo uma política estabelecida pelo sistema operacional.

Em sistemas com múltiplos processadores, existe a possibilidade de mais de um processo estar sendo executado ao mesmo tempo. Neste tipo de sistema, também é possível um mesmo processo ser executado simultaneamente em mais de uma CPU, caso seja desenvolvido de acordo.

5.2.2 – Pronto (ready)

Um processo está no **estado de pronto** quando aguarda apenas para ser executado. O sistema operacional é responsável por determinar a ordem e os critérios pelos quais os processos em estado de pronto devem fazer uso do processador. Este mecanismo é conhecido como **escalonamento.** Em geral existem vários processos no sistema no estado de pronto organizados em listas encadeadas (Figura 5.5).

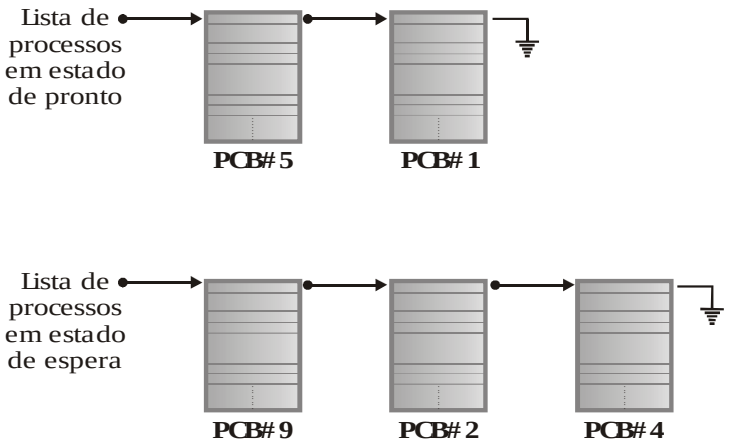

Figura 5.5: Lista de PCBs nos estados de “pronto” e “espera”.

5.2.3 – Espera (wait)

Um processo no **estado de espera** aguarda por algum evento externo ou por algum recurso para prosseguir seu processamento. Como exemplo, podemos citar o término de uma operação de entrada/saída ou a espera de uma determinada data e/ou hora para continuar sua execução. Em alguns sistemas operacionais, o estado de espera pode ser chamado de bloqueado (blocked).

O sistema organiza os vários processos no estado de espera também em listas encadeadas. Em geral, os processos são separados em listas de espera associadas a cada tipo de evento (Figura

5.5). Neste caso, quando um evento acontece, todos os processos da lista associada ao evento são transferidos para o estado de pronto.

5.3 – Mudanças de Estado do Processo

Um processo muda de estado durante seu processamento em função de eventos originados por ele próprio *(eventos voluntários)* ou pelo sistema operacional *(eventos involuntários).* Basicamente, existem quatro mudanças de estado que podem ocorrer a um processo:

Pronto → Execução

Após a criação de um processo, o sistema o coloca em uma lista de processos no estado de pronto, onde aguarda por uma oportunidade para ser executado (Figura 5.6a). Cada sistema operacional tem seus próprios critérios e algoritmos para a escolha da ordem em que os processos serão executados (política de escalonamento).

Execução → Espera

Um processo em execução passa para o estado de espera por eventos gerados pelo próprio processo, como uma operação de *E/S,* ou por eventos externos (Figura 5.6b). Um evento externo é gerado, por exemplo, quando o sistema operacional suspende por um período de tempo a execução de um processo.

Espera → Pronto

Um processo no estado de espera passa para o estado de pronto quando a operação solicitada é atendida ou o recurso esperado é concedido. Um processo no estado de espera sempre terá de passar pelo estado de pronto antes de poder ser novamente selecionado para execução. Não existe a mudança do estado de espera para o estado de execução diretamente (Figura 5.6c).

Execução → Pronto

Um processo em execução passa para o estado de pronto por eventos gerados pelo sistema, como o término da fatia de tempo que o processo possui para sua execução (Figura 5.6d). Nesse caso, o processo volta para a fila de pronto, onde aguarda por uma nova oportunidade para continuar seu processamento.

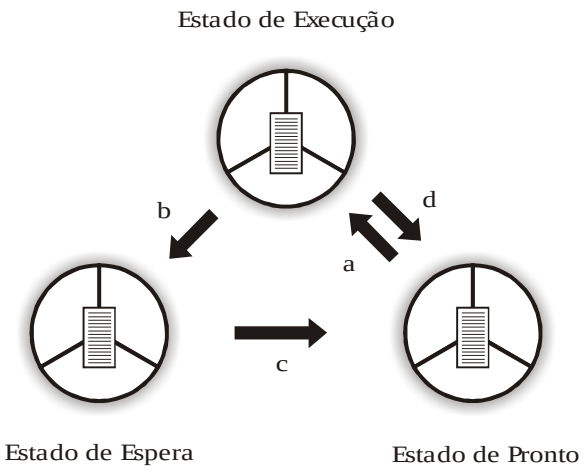

Figura 5.6: Mudanças de estado do processo.

5.4 – Criação e Eliminação de Processos

Processos são criados e eliminados por motivos diversos. A criação de um processo ocorre a partir do momento em que o sistema operacional adiciona um novo PCB à sua estrutura e aloca um espaço de endereçamento na memória para uso. A partir da criação do PCB, o sistema operacional já reconhece a existência do processo, podendo gerenciá-lo e associar programas ao seu contexto para serem executados. No caso da eliminação de um processo, todos os recursos associados ao processo são desalocados e o PCB eliminado pelo sistema operacional.

Além dos três estados apresentados anteriormente para o processo, a maioria dos sistemas operacionais estabelece dois estados adicionais para os momentos de criação e eliminação de um processo. A Figura 5.7 ilustra as mudanças de estado do processo considerando estes dois novos estados.

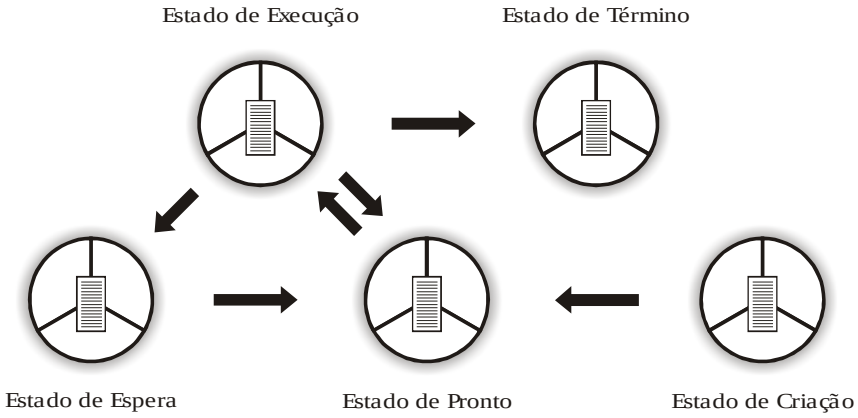

Figura 5.7: Mudanças de estado do processo.

Criação (new):

Um processo é dito no **estado de criação** quando o sistema operacional já criou um novo PCB, porém ainda não pode colocá-lo na lista de processos do estado de pronto. Alguns sistemas operacionais limitam o número de processos ativos em função dos recursos disponíveis ou de desempenho. Esta limitação pode ocasionar que processos criados permaneçam no estado de criação até que possam passar para ativos.

A criação de processos pode ocorrer por diferentes razões, como:

- Login interativo: desta forma, um processo é criado através do estabelecimento de uma sessão interativa por um usuário a partir de um terminal;
- Criação por um outro processo: um processo já existente pode criar outros processos, sendo estes novos processos independentes ou subprocessos;
- Criação pelo sistema operacional: o S.O. pode criar novos processos com o intuito de oferecer algum tipo de serviço.

Terminado (exit):

Um processo no **estado de terminado** não poderá ter mais nenhum programa executado no seu contexto, porém o sistema operacional ainda mantém suas informações de controle presentes em memória. Um processo neste estado não é mais considerado ativo, mas como o PCB ainda

existe, o sistema operacional pode recuperar informações sobre a contabilização de uso de recursos do processo, como o tempo total do processador. Após as informações serem extraídas, o processo pode deixar de existir.

O término de processo pode ocorrer por razões como:

- Término normal de execução;
- Eliminação por um outro processo;
- Eliminação forçada por ausência de recursos disponíveis no sistema.

Exemplos de estados de processo em Sistemas Operacionais

Sistema Operacional Multics

Os estados do processo no Multics são um pouco diferentes(Figura 5.8). O estado tradicional de Espera (quando se faz um I/O), é substituído por dois estados de processo, o Bloqueado e o Esperando. Bloqueado é quando o processo faz uma solicitação e a resposta é relativamente longa para ser entregue, como uma entrada de um usuário por exemplo. Esperando é a espera por um I/O cuja resposta virá rapidamente. O estado Inelegível ocorre quando o processo sai da execução e fica em uma fila de espera (Inelegível), esperando sua vez para ser executado quando vai para o estado de Pronto (Elegível).

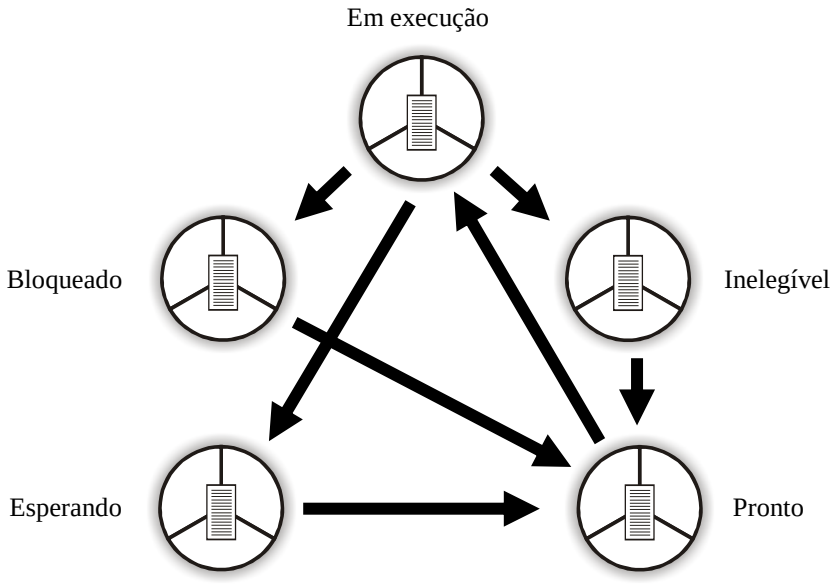

Figura 5.8: Mudanças de estado do processo no Multics.

Sistema Operacional WindowsNT

No modelo de máquina de estados de processo/thread do Windows NT (Figura 5.9), há a inclusão de dois novos estados o Standby e o Transição. Standby ocorre quando o processo é escalonado para ser executado porem aguarda pela troca de contexto para entrar em execução. Nesse momento ele pode retornar ao estado de pronto se outro processo de maior prioridade precisar ser executado. Já o estado de Transição ocorre quando a resposta da solicitação de I/O é entregue, porém isso foi gravado em Swap de disco e é preciso ler isso e transferir para a memória principal. Ou seja, enquanto ocorre a leitura dessas páginas em disco o processo fica em estado de transição, e assim que tudo é lido e entregue a memória então o processo vai para o estado de Pronto. Outra modificação é o estado de Terminado. Quando o processo termina a sua execução ele é setado como Terminado porém o processo pode ser realmente Terminado ou reinicializado indo ao estado de Criação.

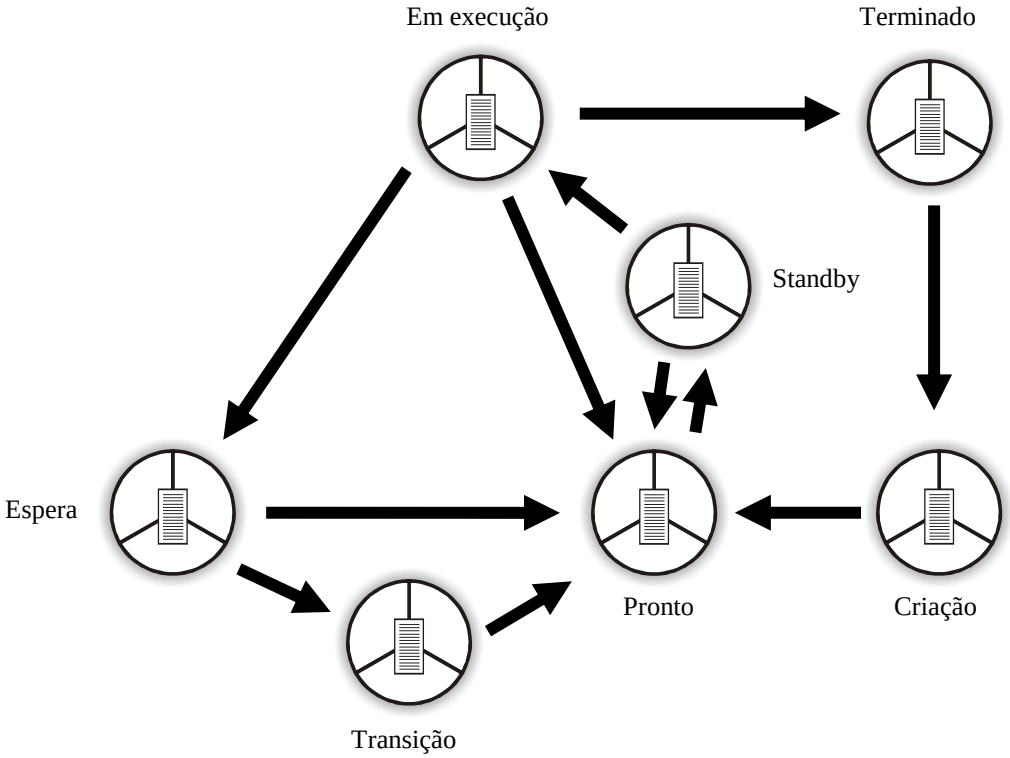

Figura 5.9: Mudanças de estado do processo/thread no Windows NT.

Sistema Operacional Linux

Os estados que um processo pode assumir no Linux é semelhante ao Multics (Figura 5.10). O estado de Espera é substituído por dois estados o *Uninterruptible Sleep* (Dormindo) e o *Interruptible Sleep* (Cochilo). Dormindo ocorre quando é realizada uma solicitação de I/O. Já Cochilo acontece quando um processo precisa aguardar a sinalização de algum evento. Outra introdução nos estados de processo do Linux é a inserção do estado de Zombie (Zumbi) e Dead (Morto) ao invés do estado de Término. Um processo entra em estado de Zumbi quando por algum motivo um processo entrou em estado de término mas suas estruturas de dados, não foram destruídas. Isso ocorre geralmente quando um processo pai deixa de confirmar o término de um processo filho. Por sua vez o estado de Morto, é o estado de término tradicional, que pode ser

visualizado por alguns instantes, como em um velório e depois enterro. O estado de Bloqueado é aqui conhecido no Linux como Stopped.

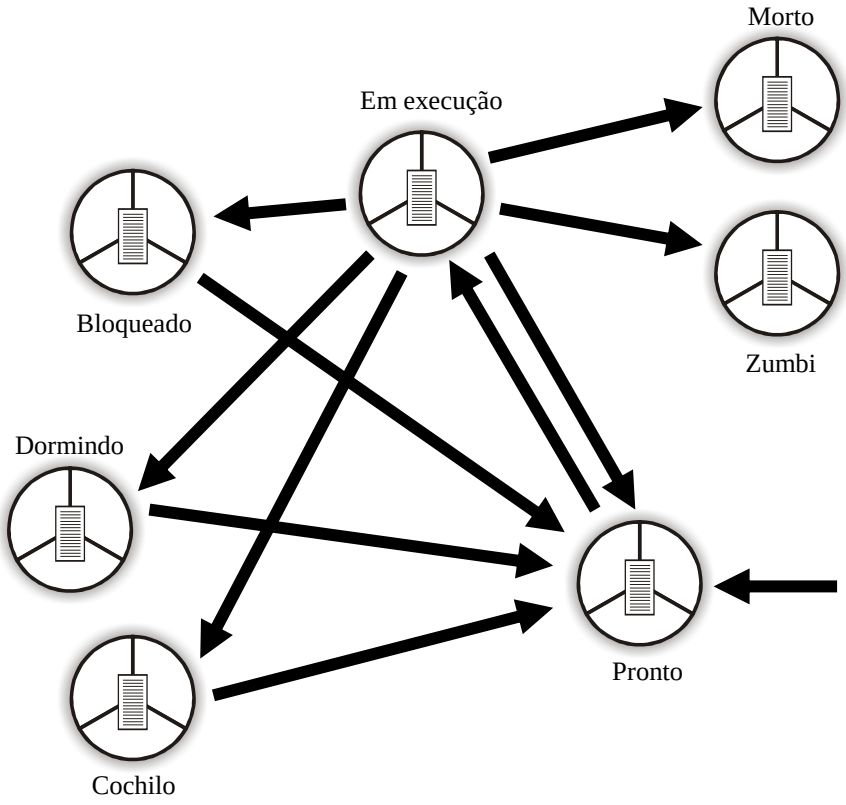

Figura 5.10: Mudanças de estado do processo no Linux.

5.5 – Processos Independentes, Subprocessos e Threads

Processos independentes, subprocessos e threads são maneiras diferentes de implementar a concorrência em sistemas multiprogramáveis. O uso de **processos independentes** é a maneira mais simples de alcançar essa concorrência. Neste caso não existe vínculo do processo criado com o seu criador. A criação de um processo independente exige a alocação de um PCB, possuindo contextos de hardware, contexto de software e espaço de endereçamento próprios.

Subprocesso*s* são processos criados dentro de uma estrutura hierárquica. Neste modo, o processo criador é denominado **processo pai** enquanto o novo processo é chamado de **subprocesso** ou **processo filho**. O subprocesso, por sua vez, pode criar outras estruturas de subprocessos. Uma característica desta implementação é a dependência existente entre o processo criador e o subprocesso. Caso um processo pai deixe de existir, os subprocessos subordinados são automaticamente eliminados. Semelhante aos processos independentes, subprocessos possuem seu próprio PCB. A Fig. 5.11 ilustra cinco processos em uma estrutura hierárquica, cada qual com seu próprio contexto de hardware, contexto de software e espaço de endereçamento.

Além da dependência hierárquica entre processos e subprocessos, uma outra característica neste tipo de implementação é que subprocessos podem compartilhar quotas com o processo pai.

Neste caso, quando um subprocesso é criado, o processo pai cede parte de suas quotas ao processo filho.

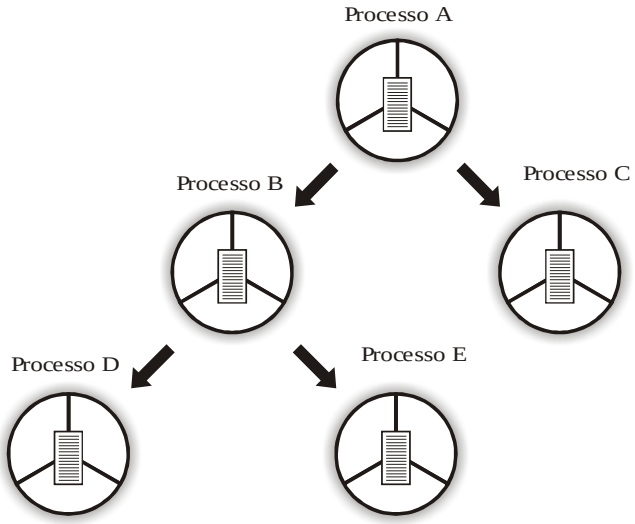

Figura 5.11: Estrutura de processos e subprocessos.

O uso de processos independentes e subprocessos no desenvolvimento de aplicações concorrentes demanda consumo de diversos recursos do sistema. Sempre que um novo processo é criado, o sistema deve alocar recursos (contexto de hardware, contexto de software e espaço de endereçamento), consumindo tempo de CPU neste trabalho. No momento do término dos processos, o sistema operacional também dispensa tempo para desalocar recursos previamente alocados. Outro problema é a comunicação e sincronização entre processos, considerada pouco eficiente, visto que cada processo possui seu próprio espaço de endereçamento.

O conceito de **thread** foi introduzido na tentativa de reduzir o tempo gasto na criação, eliminação e troca de contexto de processos nas aplicações concorrentes, bem como economizar recursos do sistema como um todo. Em um ambiente multithread, um único processo pode suportar múltiplas threads, cada qual associada a uma parte do código da aplicação (Figura 5.12). Neste caso não é necessário haver diversos processos para a implementação da concorrência. Threads compartilham o processador da mesma maneira que um processo, ou seja, enquanto uma thread espera por uma operação de E/S, outra thread pode ser executada.

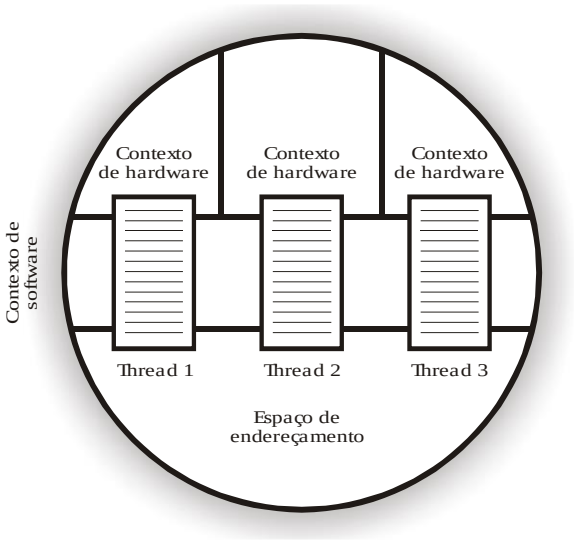

Figura 5.12: Processo mutithread.

Cada thread possui seu próprio contexto de hardware, porém compartilha o mesmo contexto de software e espaço de endereçamento com as demais threads do processo. O compartilhamento do espaço de endereçamento permite que a comunicação de threads dentro de um mesmo processo seja realizada de forma simples e rápida.

5.6 – Processos Foreground e Background

Um processo possui sempre associado à sua estrutura pelo menos dois canais de comunicação por onde são realizadas todas as entradas e saídas de dados ao longo do seu processamento. Os canais de **entrada padrão** e de **saída padrão** de dados podem estar associados a terminais, arquivos, impressoras e até mesmo outros processos.

Um **processo foreground** é aquele que permite a comunicação direta do usuário com o processo durante o seu processamento. Neste caso, tanto o canal de entrada padrão quanto o de saída padrão estão associados a um terminal com teclado, mouse e monitor, permitindo, assim, a interação com o usuário (Fig. 5.l3a). O processamento interativo tem como base processos foreground.

Um **processo background** é aquele onde não existe a comunicação com o usuário durante o seu processamento (Fig. 5.l3b). Neste caso, os canais de entrada e saída padrão não estão associados a nenhum dispositivo de E/S interativo. O processamento do tipo batch é realizado através de processos background.

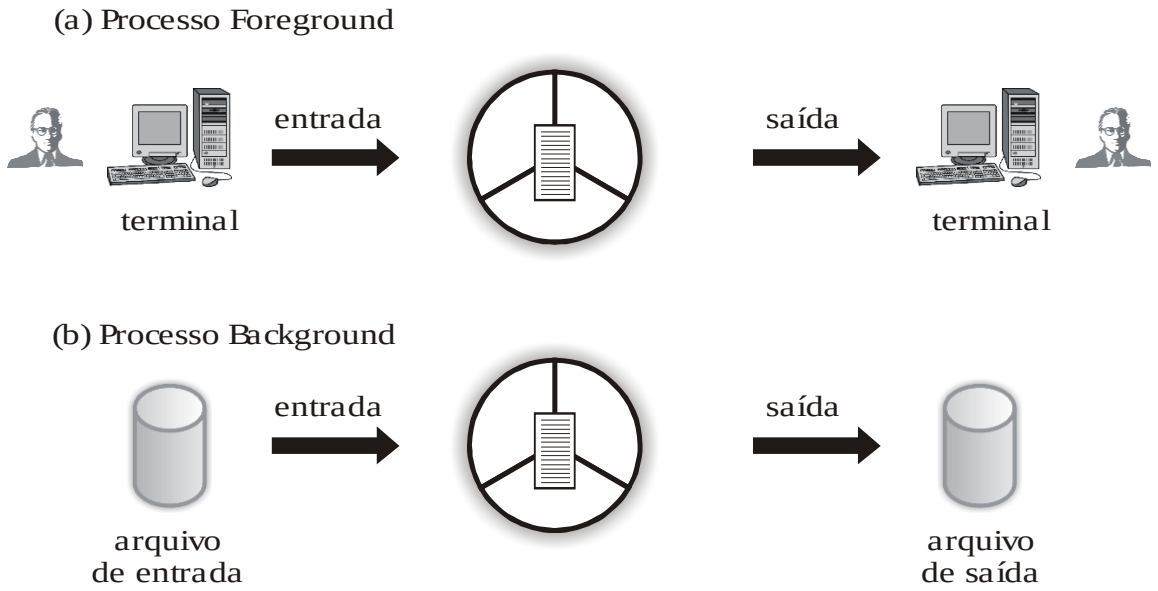

Figura 5.13: Processo foreground e background.

É possível associar o canal de saída padrão de um processo ao canal de entrada padrão de um outro processo. Neste caso dizemos que existe um **pipe** ligando os dois processos. Se um processo A gera uma listagem e o processo B tem como função ordená-la, basta associar o canal de saída padrão do processo A ao canal de entrada padrão do processo B (Fig. 5.14).

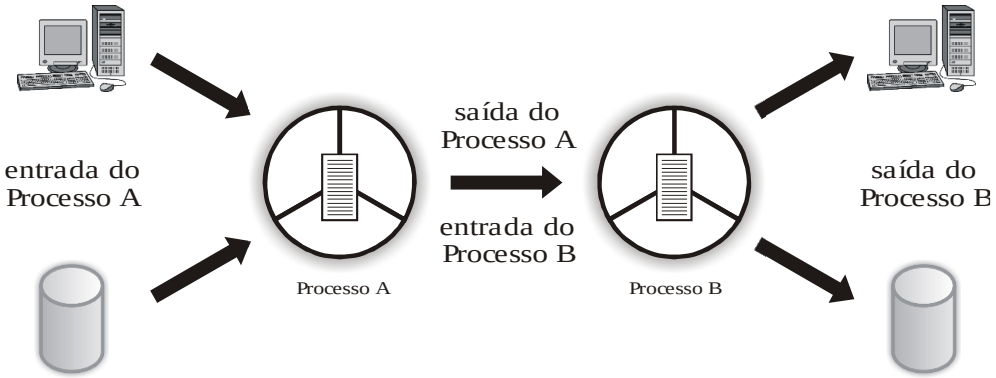

Figura 5.14: Pipe.

5.7 – Processos CPU-Bound e I/O-Bound

Os processos podem ser classificados como CPU-bound ou I/O-bound, de acordo com a utilização do processador e dos dispositivos de E/S.

Um processo é definido como **CPU-bound** (ligado à CPU) quando passa a maior parte do tempo no estado de execução, ou seja, utilizando o processador (Figura 5.15a). Esse tipo de processo realiza poucas operações de leitura e gravação e é encontrado em aplicações científicas que efetuam muitos cálculos.

Um processo é classificado como **I/O-bound** (ligado à E/S*)* quando passa a maior parte do tempo no estado de espera, pois realiza um elevado número de operações de E/S (Figura 5.15b). Esse tipo de processo é encontrado em aplicações comerciais, que se baseiam em leitura, processamento e gravação. Os processos interativos também são bons exemplos de processos I/O-bound, pela forma de comunicação entre o usuário e o sistema, normalmente lenta, devido à utilização de terminais.

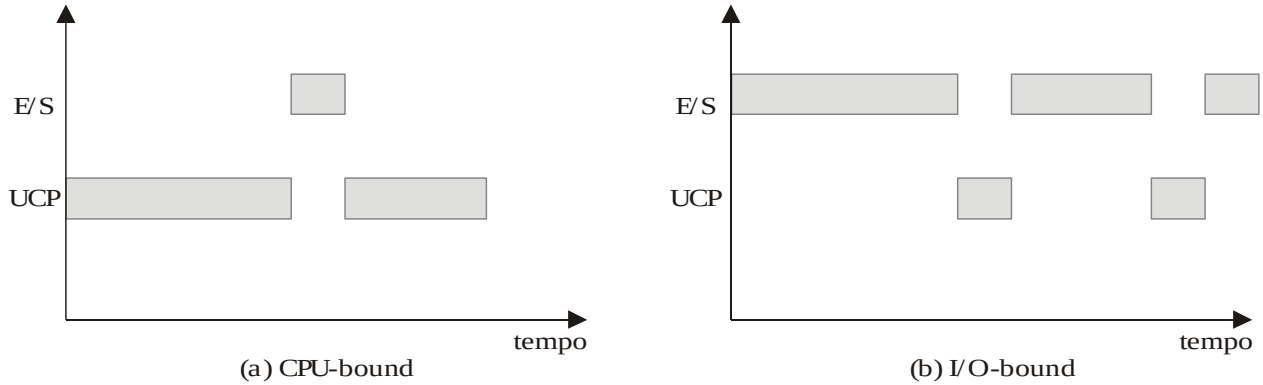

Figura 5.15: Processos CPU-Bound e I/O-Bound/

5.8 – Sinais

Sinais são um mecanismo que permite notificar processos de eventos gerados pelo sistema operacional ou por outros processos. O uso de sinais é fundamental para a gerência de processos, além de possibilitar a comunicação e sincronização entre processos.

Um exemplo de uso de sinais é quando um usuário utiliza uma seqüência de caracteres do teclado, como [Ctrl+C], para interromper a execução de um programa. Neste caso, o sistema operacional gera um sinal a partir da digitação desta combinação de teclas, sinalizando ao processo a ocorrência do evento. No momento em que o processo identifica a chegada do sinal, uma rotina específica de tratamento deve ser executada (Figura 5.16).

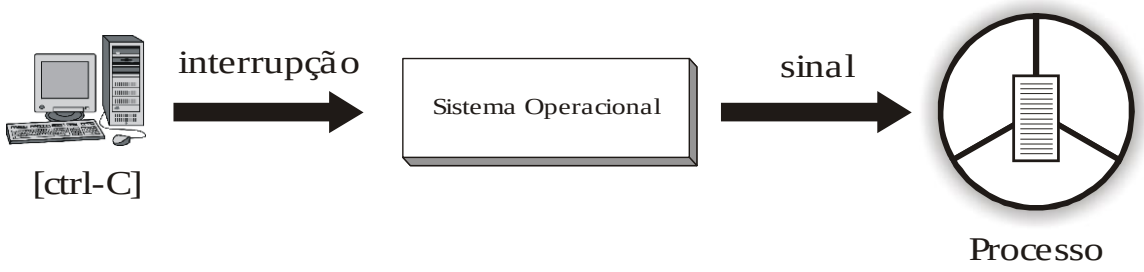

Figura 5.16: Uso de sinais.

A maior parte dos eventos associados a sinais são gerados pelo sistema operacional ou pelo hardware, como a ocorrência de exceções, interrupções geradas por terminais, limites de quotas excedidos durante a execução dos processos e alarmes de tempo. Em outras situações, os eventos são gerados a partir de outros processos com o propósito de sincronizar suas execuções.

A geração de um sinal ocorre quando o sistema operacional, a partir da ocorrência de eventos síncronos ou assíncronos, notifica o processo através de bits de sinalização localizados no seu PCB. Um processo não responde instantaneamente a um sinal. Os sinais ficam pendentes até que o processo seja escalonado, quando então serão tratados. Por exemplo, quando um processo é eliminado, o sistema ativa o bit associado a este evento. O processo somente será excluído do sistema quando for selecionado para execução. Neste caso, é possível que o processo demore algum período de tempo até ser eliminado de fato.

O tratamento de um sinal é muito semelhante ao mecanismo de interrupções. Quando um sinal é recebido pelo processo, o contexto do processo é salvo e a execução desviada para um código de tratamento de sinal (signal handler), geralmente no núcleo do sistema. Após a execução do tratador de sinais, o programa pode voltar a ser processado do ponto onde foi interrompido. Em certas implementações, o próprio processo pode tratar o sinal através de um tratador de sinais definido no código do programa. É possível também que um processo bloqueie temporariamente ou ignore por completo alguns sinais.

O mecanismo de sinais assemelha-se ao tratamento de interrupções e exceções vistos no capítulo "Concorrência", porém com propósitos diferentes. O sinal está para o processo assim como as interrupções e exceções estão para o sistema operacional (Figura 5.17).

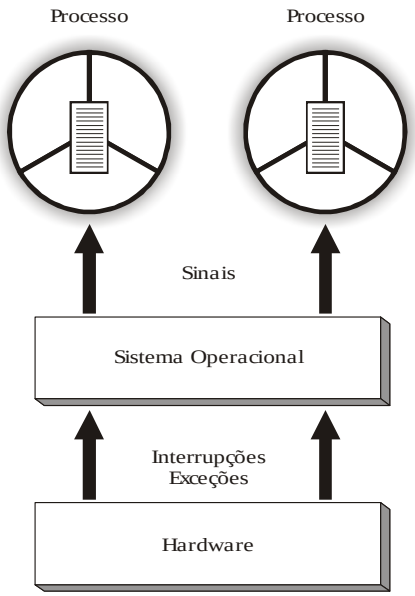

Figura 5.17: Sinais, interrupções e exceções.

Referências

FROSI, Luciano Carlos. **Sistemas Operacinais**. Apostila Word. 2010.

MACHADO, Francis Berenger. MAIA, Luiz Paulo. **Arquitetura de Sistemas Operacionais**. 3 ed. LTC – Livros Técnicos e Científicos Editora AS, 2002.

STUART, Brian L. **Princípios de Sistemas Operacionais –** Projetos e Aplicações. Cengage Learning, 2011.