Os sistemas computacionais atuais permitem que diversos programas sejam carregados na memória e executados simultaneamente. Essa evolução tornou necessário um controle maior na divisão de tarefas entre os vários programas. Essas necessidades resultaram na noção de processo.

Um sistema é constituído de um conjunto de processos que executam seus respectivos códigos do sistema operacional e processos e códigos de usuários.

Processos

Como vimos, um processo é um programa em execução. A execução de um processo ocorre de maneira seqüencial, ou seja, uma instrução após a outra. A qualquer instante, apenas uma instrução de um determinado processo é executada.

- **Estão associadas a um processo informações sobre seu estado atual.**
  - Representadas pelo valor do contador de instruções e pelos valores contidos nos registradores do processador.
  - Também uma pilha, que contém dados temporários (tais como argumentos de sub-rotinas, endereços de retorno e variáveis temporárias), e uma seção de dados contidas em variáveis globais.

Nota	Um programa por si só não é um processo, mas uma entidade passiva, tal como o conteúdo de um arquivo armazenado em um disco, enquanto um processo é uma entidade ativa, com um contador de instruções que específica à próxima instrução a ser executada e um conjunto de recursos a ele associados.

- Embora dois processos possam estar associados a um mesmo programa, são considerados duas seqüências de execução distintas. Por exemplo, cópias de um programa de correio eletrônico podem estar sendo executadas por vários usuários ou o mesmo usuário pode estar usando diversas cópias de um processador de texto. Cada um desses programas em execução constitui um processo distinto e, embora o texto de alguns programas possa ser o mesmo, o estado de cada processo será diferente. É comum ter um processo que crie muitos processos durante sua execução.

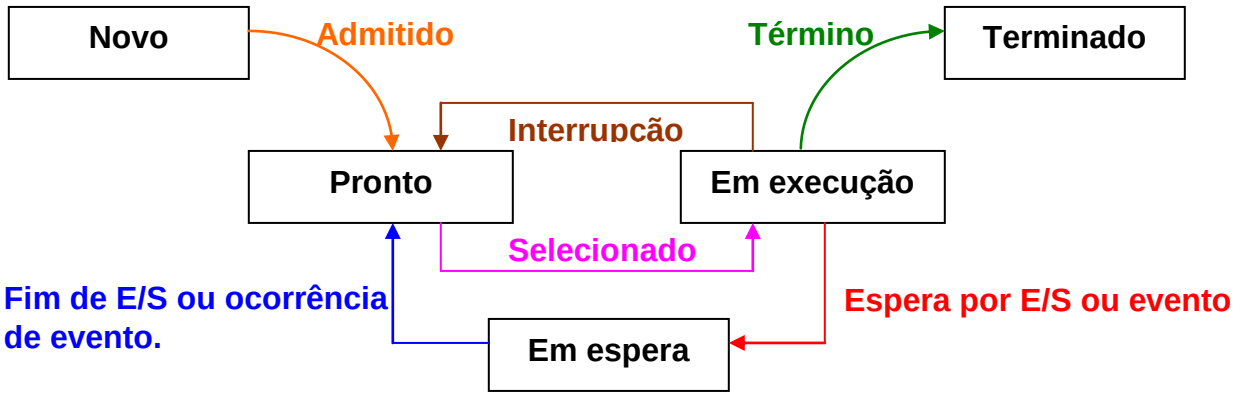


Diagrama de estados de um processo

Estado do Processo

Um processo em execução pode provocar uma mudança de estado. O estado de um processo é definido em parte pelo valor do seu contador de instruções e pelos valores dos registradores do processador.

Podem ser classificados, de acordo com o estado:

- **Novo**
  - **O processo está sendo criado.**
- **Em execução**
  - **Instruções do código do processo estão sendo executadas.**
- **Em espera**
  - **O processo está esperando pela ocorrência de algum evento (tal como a realização de uma operação de E/S ou o recebimento de um sinal).**
- **Pronto**
  - **O processo está pronto para obter o controle do processador.**
- **Terminado**
  - **O processo terminou de ser executado.**

Nota	**É importante reconhecer que apenas um processo pode estar em execução em um processador a cada instante. Entretanto, muitos processos podem estar prontos e em espera para entrar em execução.**

Bloco de Controle de Processos (BCP)

Cada processo é descrito no sistema operacional por um bloco de controle de processos.

Apontador	Estado do processo
Número do processo	
Contador de Instruções	
Registradores	
Limites na memória	
Lista de arquivos abertos	
⋮	

Bloco de Controle de Processos

- **Estado do Processo**
  - Pode ser novo, pronto, em execução, em espera, terminado e assim por diante.
- **Contador de Instruções**
  - Indica o endereço da próxima instrução desse processo a ser executada.
- **Registradores da CPU**
  - Os registradores variam em número e tipo, dependendo da arquitetura do computador. Eles podem ser acumuladores, indexadores, apontadores do topo da pilha e registradores de propósito geral e podem conter também informações sobre o resultado de testes.

Realizados por instruções anteriores, chamadas de códigos de condição. Com o contador de instruções, o valor contido em cada um desses registradores deve ser armazenado quando ocorre uma interrupção, para permitir que o processo possa continuar sua execução normalmente quando obtiver novamente o controle da CPU.

- **Informações para alocação de CPU**
  - Inclui prioridade do processo, apontadores para filas e outros parâmetros usados na seleção de processos(estudaremos em mais detalhes no próximo capítulo).

- **Informações relativas ao gerenciamento de memória**
  - Podem incluir os valores contidos nos registradores que indicam a base e o limite do espaço de memória reservado ao processo, tabela de páginas ou tabela de segmento usados pelo processo, dependendo do sistema de gerenciamento de memória usado pelo sistema operacional.

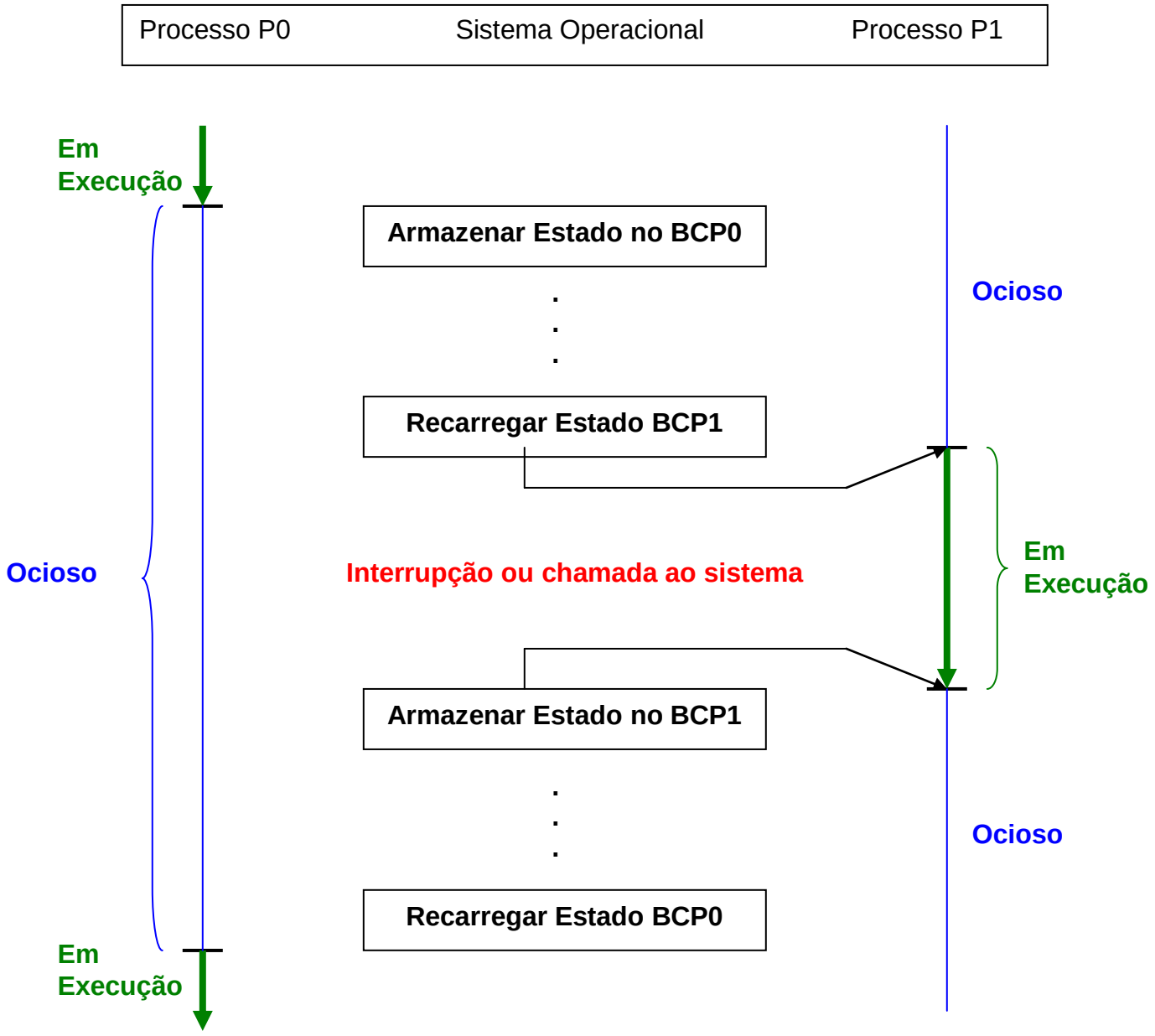


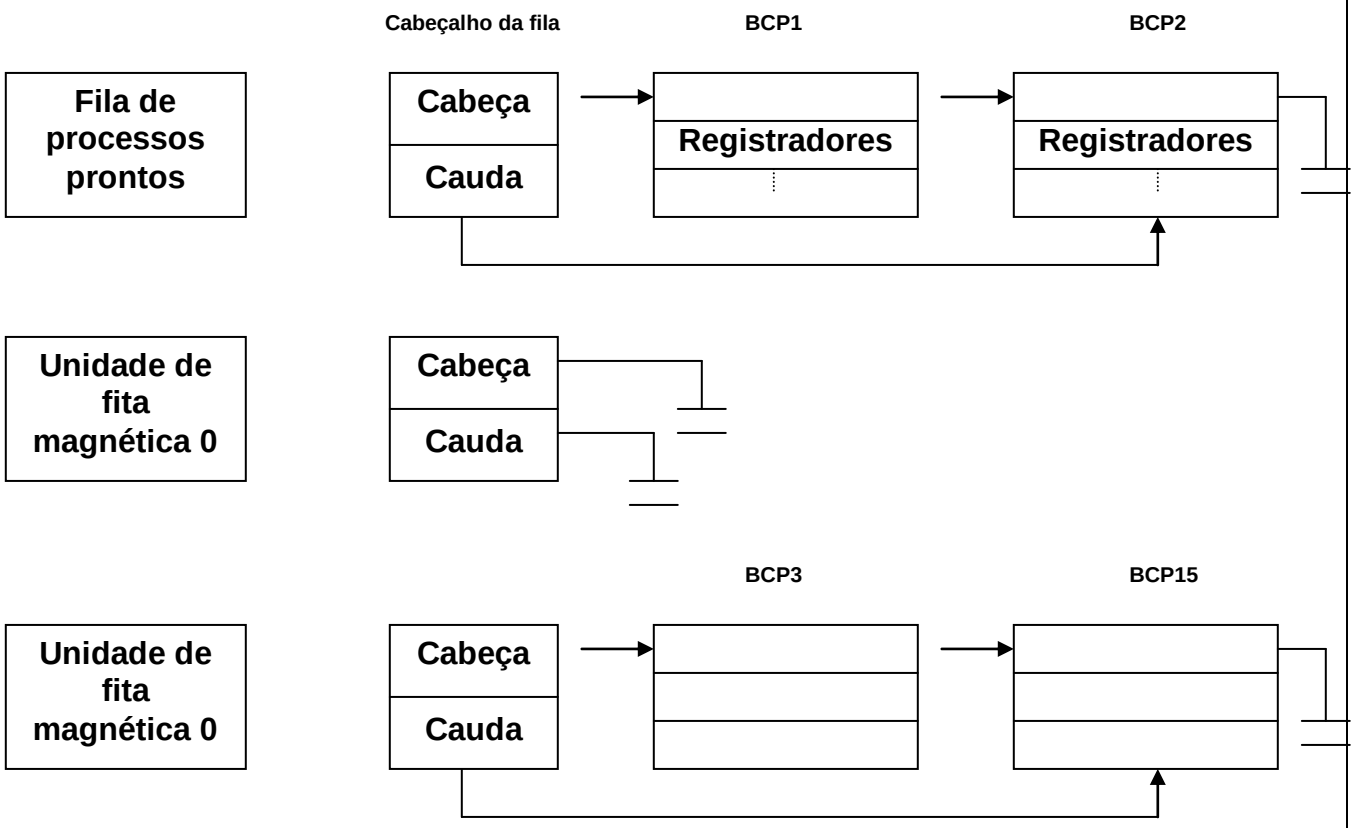
Diagrama mostrando a transferência da CPU entre processos

Nota	**O BCP serve simplesmente como um repositório para quaisquer informações que podem variar de um processo para outro.**

Seleção de Processos

A multiprogramação objetiva ter sempre algum processo sendo executado, para maximizar a utilização da CPU. O objetivo do compartilhamento de tempo é permitir que os usuários possam interagir com seus programas enquanto eles são executados. Para que isso aconteça a CPU é transferida freqüentemente entre os processos. Em um sistema com um único processador, nunca haverá mais de um processo em execução. Se houver mais processos ativos, cada um terá que esperar até que a CPU esteja livre e possa ser alocado para sua execução.

- **Filas para Seleção de Processos**
  - À medida que novos processos vão sendo criados, eles são colocados em uma fila de processos. Essa fila é constituída de todos os processos do sistema. Os processos que estão armazenados na memória principal e estão prontos para entrar em execução são mantidos em uma fila de processos prontos.
  - O cabeçalho de uma fila de processos prontos contém apontadores para o primeiro e o último BCP na lista. Cada BCP tem um campo que contém um apontador para o próximo processo na fila de processos prontos.
  - Existem outros tipos de filas no sistema.
    - Quando a CPU é alocada a um processo, suas instruções são executadas durante algum tempo e depois o processo termina, é interrompido ou passa a esperar pela ocorrência de um determinado evento, tal como o término de uma operação de E/S por ele requisitada. Nesse último caso, a requisição de E/S pode ser uma requisição de uma operação em uma unidade de fita cujo uso foi reservado ao processo ou pode ser uma requisição de uma operação em um dispositivo compartilhado, como um disco. O processo pode ter de esperar para usar o disco.
    - Cada dispositivo tem sua própria fila, chamada fila de dispositivo, com descrições dos processos que estão esperando para usar o dispositivo.



Fila de processos prontos e filas de dispositivos de E/S

Nota

Dois tipos de fila estão presentes: A fila de processos prontos e um conjunto de filas de dispositivos.

- Um novo processo é inicialmente colocado na fila de processos prontos. Ele espera nessa fila até ser selecionado para execução e receber então o controle da CPU. Quando a CPU é alocada a um processo e este está sendo executado, um dos seguintes eventos pode ocorrer.
  - O processo pode fazer um requisição para realização de uma operação de E/S e então ser colocado em uma fila de E/S
  - O processo pode criar um novo subprocesso e esperar pelo seu término.
  - O processo pode perder o controle da CPU, como resultado de uma interrupção, e ser posto de volta na fila de processos prontos.

Nota

Nos dois primeiros casos anteriores, o processo mudará em algum momento do estado de espera para o estado pronto e será colocado de volta na fila de processos prontos para entrar em execução. Os processos continuam nesse ciclo até que terminem, quando então serão removidos de todas as filas e terão seus recursos, assim como o espaço reservado ao seu BCP, liberados.

- **Escalonadores**

Um processo passa pelas várias filas de seleção durante sua execução. O sistema operacional deve selecionar processos dessas filas de algum modo, para fins de alocação de recursos. O processo de escolher processos e o de escalar processos para execução é realizado por programas **chamados de escalonadores.**

- Em um sistema batch, é comum que nem todos os processos submetidos possam ser imediatamente executados. Eles são armazenados temporariamente em um dispositivo de armazenamento secundário, para poderem ser executados posteriormente.
- **Escalonador de Processos**
  - Escolhe processos desse repositório e os carrega na memória para execução.

- **Escalonador de CPU**
  - Seleciona um dentre os processos que estão prontos para serem executados e aloca a CPU a esse processo.

- A distinção principal entre esses dois programas é sua freqüência de utilização.

  - O escalonador de CPU é utilizado de maneira bastante freqüente.
  - Enquanto que o escalonador de processos é executado com uma freqüência muito menor. Podem se passar vários minutos entre uma criação de um novo processo e outro.
  - O escalonador de processos controla o grau de multiprogramação(número de processos na memória). Se o grau de multiprogramação for estável, a taxa média de criação de processos deve ser

igual à taxa média de finalização de processos.

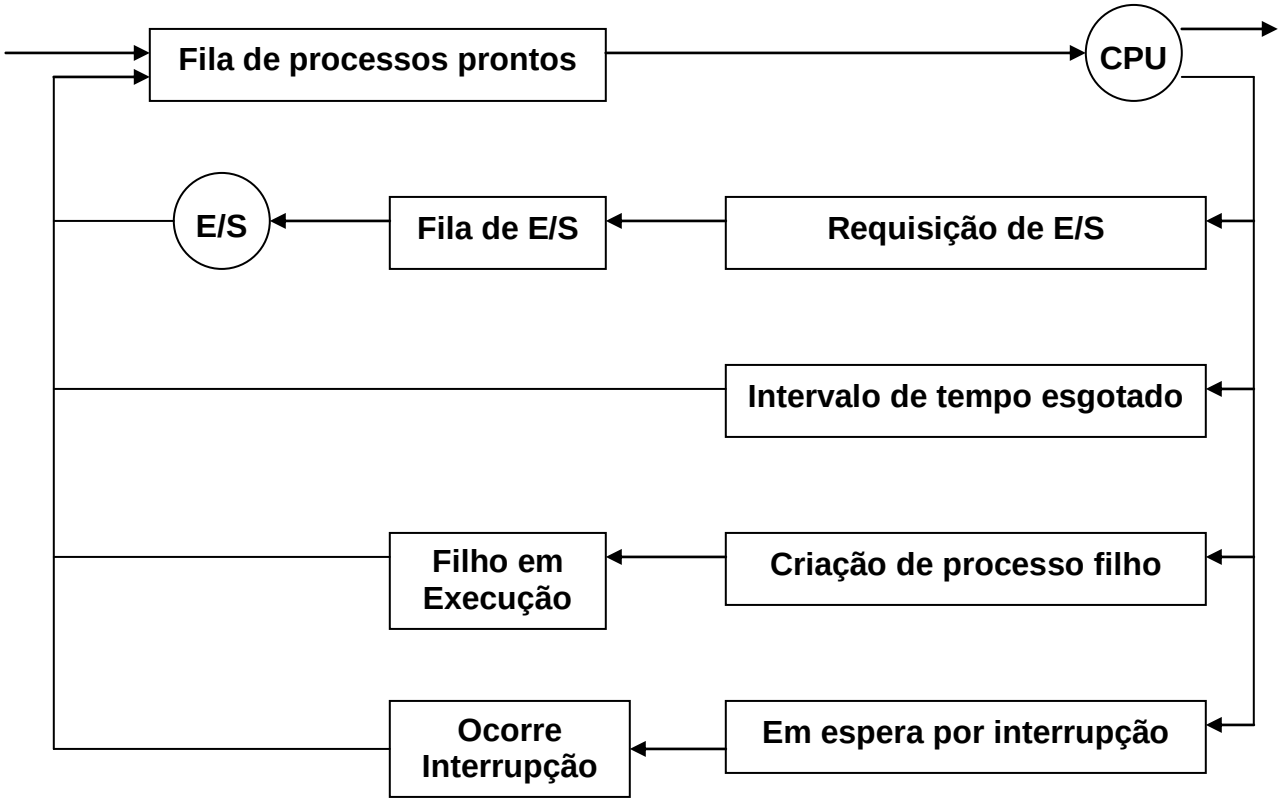
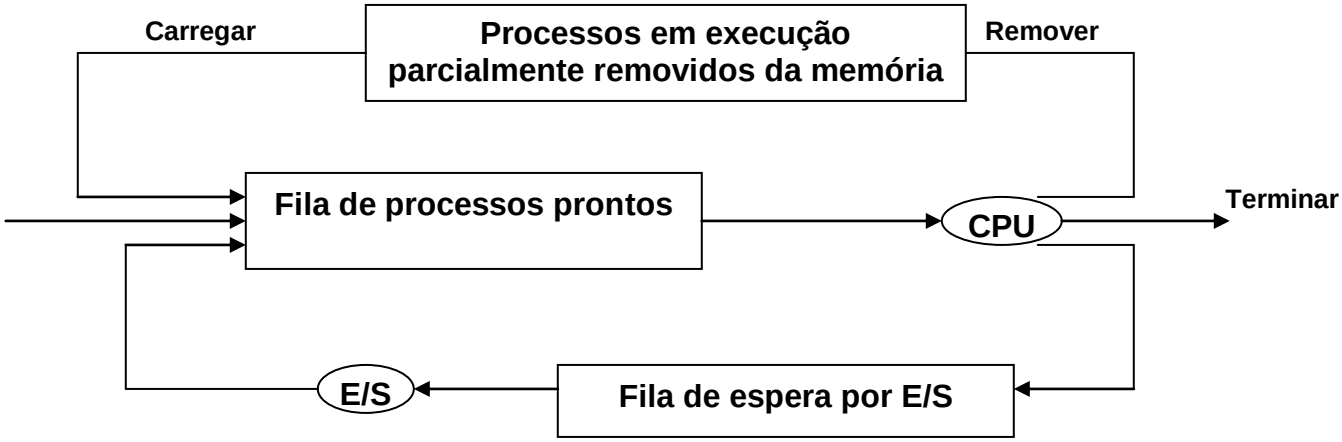


Diagrama de transição entre filas para seleção de processos

Em alguns sistemas, o escalonador de processos pode não existir ou ser muito simples. Por exemplo, os sistemas de tempo compartilhado freqüentemente não tem um escalonador de processos, mas simplesmente colocam cada novo processo na memória, para o escalonador da CPU.

- Alguns sistemas operacionais, como sistemas de tempo compartilhado, podem introduzir uma seleção intermediária adicional.
    - A idéia principal é a de que pode ser algumas vezes vantajoso remover processos da memória(e de uma disputa ativa pela CPU), reduzindo assim o grau de multiprogramação. Algum tempo depois, o processo pode ser recarregado na memória e sua execução pode continuar a partir do ponto em que foi interrompida. Esse esquema é comumente chamado de Swapping(troca de processos). O processo é removido e recarregado na memória por um escalonador. Essas transferências podem ser necessárias para melhorar o balanceamento entre processos que dependem mais de E/S e da CPU ou por que uma mudança em alguma requisição de memória

fez com que se tornasse necessário liberar um certo espaço de memória.



Seleção intermediária de processos adicionada ao diagrama de transição entre filas de processos.

- **Mudança de Contexto**

Para transferir o controle da CPU de um processo para outro, é necessário armazenar o estado do processo antigo e carregar o estado do novo processo, anteriormente armazenado. A essa tarefa se dá o nome de **mudança de contexto**. O tempo gasto nesta tarefa é considerado como um trabalho adicional pelo sistema, durante o qual ele não realiza qualquer trabalho útil.

Uma mudança de contexto consiste em mudar o apontador para o conjunto de registradores do processo a ser executado. Se houver mais processos ativos do que conjuntos de registradores, o sistema terá de transferir dados de registradores para a memória e vice-versa.

Nota	• **Mudanças de contexto são um ponto crítico no desempenho de sistemas.** • **Novas estruturas (fluxos de execução – threads) estão sendo usadas para evita-las, sempre que**

Operações sobre Processos

Processos podem ser executados simultaneamente e devem ser criados e removidos dinamicamente. Portanto, o sistema operacional deve oferecer mecanismos para a criação e a finalização de processos.

- **Criação de Processos**
  - Usando uma chamada ao sistema para criar processos, um processo pode criar vários outros durante sua execução.
    - Processo Pai
      - O processo criado.
    - Processo Filho
      - Processos criados a partir do processo Pai.
    - Cada um desses processos pode criar outros, formando uma árvore de processos.
  - Um processo precisará utilizar (tempo de CPU, memória, arquivos, dispositivos de E/S) para realizar sua tarefa.
  - Quando um processo cria um subprocesso, este pode ser capaz de obter recursos diretamente do sistema operacional ou pode estar restrito ao uso de um subconjunto dos recursos do processo pai.
  - O pai pode ter que dividir os seus recursos entre seus filhos ou pode compartilhar recursos(tais como memória ou arquivos) entre alguns de seus filhos.
  - **A restrição dos recursos de um filho a um subconjunto dos recursos do pai evita que um processo possa sobrecarregar o sistema ao criar subprocessos**
  - **em demasia.**
  - Quando um processo cria outro, existem duas possibilidades
    - **em termos de execução**:
      - O processo pai continua a ser executado simultaneamente com seu filho.
      - O processo pai espera até que algum ou todos os seus filhos tenham terminado.
    - **Em termos do espaço de endereçamento**
      - O processo filho é uma cópia do processo pai.
      - O processo filho tem um programa que deve ser carregado para sua execução.

- **Finalização de Processos**

  Um processo se completa quando a execução de seu último comando termina e uma chamada ao sistema é feita para que ele seja removido do sistema.

  - O processo pode retornar dados para seu processo pai (por meio de uma chamada ao sistema).
  - Os recursos do processo, incluindo espaço de memória física e virtual, arquivos abertos e áreas de armazenamento de dispositivos de E/S, são liberados pelo sistema operacional, podendo agora ser usados por outros processos.
  - Um processo também pode ocasionar a terminação de um outro processo por meio de uma determinada chamada ao sistema (por exemplo, abort).
  - Normalmente, essa chamada pode ser feita apenas pelo pai do processo cuja execução deve terminar. Caso contrário, os usuários poderiam provocar arbitrariamente a terminação de processos de outros usuários. Um PAI precisa conhecer o identificador de seus FILHOS.
  - Quando um processo cria outro, o identificador do processo criado é passado ao seu PAI.
  - Motivos para um processo PAI encerrar um processo FILHO:
    - O FILHO excedeu um limite estabelecido para o uso de algum dos recursos que foram a ele alocados.
    - A tarefa atribuída a um determinado filho não é mais necessária.
    - A execução do pai está terminando e o sistema operacional não permite que a execução de um processo filho continue se a de seu pai já terminou.

Processos Cooperativos

Os processos concorrentes em execução em um sistema operacional podem ser tanto independentes quanto cooperativos:

- **Independente**
  - Se um processo não pode afetar ou ser afetado por outros processos em execução no sistema.
  - Qualquer processo que não compartilhe dados com qualquer outro processo.
- **Cooperativo**
  - Se um processo pode afetar ou ser afetado por outros processos em execução no sistema, ou melhor, qualquer processo que compartilhe dados com outros processos.

- **Razões para permitir a cooperação entre processos:**
  - **Compartilhamento de informações**
    - Podem existir vários usuários interessados em uma mesma informação. Por exemplo, um arquivo compartilhado, devemos permitir acesso concorrente a esse tipo de recurso.
  - **Aumento da velocidade de processamento**
    - Uma tarefa será executada mais rapidamente de puder ser dividida em subtarefas, cada uma das quais deverá ser executada em paralelo com as outras. Esse aumento de velocidade de processamento só pode ser alcançado se o computador tiver várias unidades de processamento (tais como CPUs ou canais de E/S).
  - **Modularidade**
    - Podemos querer construir o sistema de maneira modular, dividindo as funções do sistema em processos separados.
  - **Conveniência**
    - Mesmo um único usuário pode ter muitas tarefas para serem realizadas simultaneamente. Por exemplo, um usuário pode estar editando, imprimindo e compilando em paralelo.

Nota	**Uma execução concorrente que requer cooperação entre os processos requer também mecanismos que permitam aos processos comunicarem-se uns com os outros e sincronizarem suas ações.**

Fluxos de Execução

Nota	**Fork** **No UNIX cada processo é identificado por um identificador de processo, um valor inteiro único que o distingue dos outros processos. Um novo processo é criado pela chamada ao sistema fork. O espaço de endereçamento do novo processo é uma cópia do espaço de endereçamento do processo original. Esse mecanismo permite ao processo pai comunicar-se facilmente com seu processo filho, com uma diferença: o código retornado pela chamada fork é zero para o processo novo (FILHO), enquanto o identificador de processo (diferente de zero) do filho é retornado ao pai.**

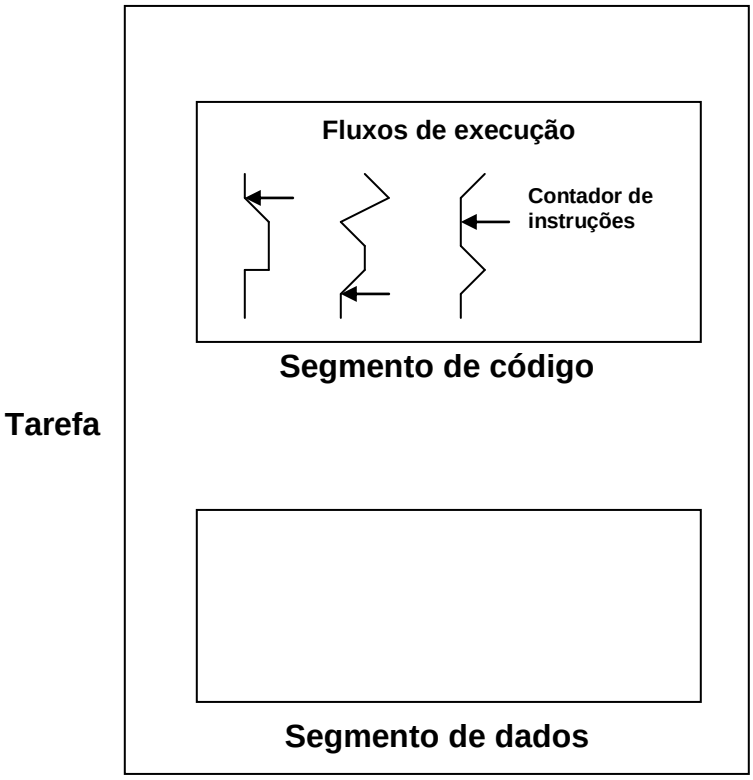
O estado de um processo é definido pelos recursos que ele está usando e pela posição na qual está sendo executado. Existem muitos casos, entretanto, em que seriam úteis o acesso e o compartilhamento simultâneo de recursos. Essa situação é similar ao caso em que a execução de uma chamada ao sistema fork é realizada com um novo contador de instruções, ou **fluxo de execução (thread), com o processamento compartilhando o mesmo espaço de endereçamento.**

- **Estrutura de Fluxos de Execução**
  - Um fluxo de execução (thread) é uma unidade básica de utilização da CPU e consiste em um contador de instruções, um conjunto de registradores e um espaço de pilha. **Ele compartilha com fluxos irmãos sua seção de código, a seção de dados e os recursos do sistema operacional**, como arquivos abertos e sinais, conhecidos coletivamente como uma tarefa.
  - Um processo tradicional é igual a uma tarefa com um único fluxo de execução.
  - **Uma tarefa não faz coisa alguma sem um fluxo de execução, enquanto um fluxo de execução deve fazer parte de apenas uma tarefa.**
  - **Compartilhamentos extensivos de recursos são beneficiados pela troca da CPU entre fluxos de execução irmãos e pela criação de novos fluxos de execução, em comparação com mudanças de contexto necessárias entre processos tradicionais.**
  - **Embora uma mudança de contexto entre fluxos de execução ainda exija uma alteração no conjunto de registradores, não é necessário nenhum trabalho relacionado ao gerenciamento de memória.**
  - Como qualquer ambiente de processamento paralelo, a existência de vários fluxos de execução para um processo pode introduzir problemas de controle de concorrência **que necessitam do uso de regiões críticas ou semáforos.**
  - Podemos entender melhor o comportamento de fluxos de execução ao comparar o controle de vários fluxos com o controle de vários processos.
    - Quando diversos processos estão envolvidos, cada processo opera independentemente dos outros, cada um deles tem seu próprio contador de instruções,

registrador apontador de topo da pilha e espaço de endereçamento.

- Esse tipo de organização é útil quando as tarefas realizadas pelos processos não têm relação umas com as outras.
- Diversos processos podem também desempenhar uma mesma tarefa. Por exemplo, eles podem fornecer dados para máquinas remotas na implementação de um sistema de arquivos de uma rede de computadores.
- É mais eficiente ter um processo contendo diversos fluxos de execução que sirvam ao mesmo propósito. Na implementação com diversos processos, cada processo executaria o mesmo código, mas teria espaço em memória e recursos de arquivo próprios.
- Uma implementação com um único processo e vários fluxos de execução usa menos recursos do que no caso de vários processos redundantes, incluindo memória, arquivos abertos e alocação da CPU. Por exemplo, à medida que o Solaris evolui, daemons que implementam mecanismos de comunicação na rede estão sendo reescritos como fluxos de execução do núcleo, para aumentar o desempenho das funções de servidores em redes de computadores.
- **Os fluxos de execução operam, em muitos aspectos, da mesma maneira que os processos. Eles podem estar em um dos vários estados: pronto, bloqueado, em execução ou finalizado.**
- **Assim como os processos, os fluxos de execução compartilham o uso da CPU e apenas um fluxo de execução a cada vez está ativo (em execução).**
- **Um fluxo de execução é processado seqüencialmente e cada um tem sua própria pilha e contador de instruções.**
- **Um fluxo de execução pode criar um fluxo filho e pode ser bloqueado, ficando em espera até o término de uma chamada ao sistema.**
- **Se um fluxo de execução está bloqueado, um outro fluxo pode ser executado.**
- **Em contraste com os processos, os fluxos de execução não são independentes uns dos outros.**
- Como todos os fluxos podem usar os mesmos endereços em uma tarefa, um fluxo pode ler e escrever sobre pilhas de outros fluxos.
- O mecanismo não oferece proteção fluxos de execução. **Essa proteção, entretanto, não deve ser necessária. Os processos podem se originar-se de usuários diferentes, podendo interferir uns com os**

outros de maneira não desejada, ao passo que, no caso de vários fluxos de execução, um único usuário é responsável por uma dada tarefa.



Diversos fluxos de execução em uma tarefa

Vamos tomar o exemplo de um processo servidor de arquivos, que está bloqueado e é executado em um sistema que pressupõe a utilização de apenas um único processo. Nesse cenário, nenhum outro processo servidor pode ser executado até que o primeiro seja desbloqueado. Por outro lado, quando uma tarefa tem diversos fluxos de execução, enquanto um fluxo servidor está bloqueado, em espera pela ocorrência de algum evento, um segundo fluxo da mesma tarefa pode ser executado. Nessa aplicação, a cooperação entre os diversos fluxos é vantajosa, proporcionando uma melhora de produtividade e desempenho.

Nota

O mecanismo de fluxos de execução permite que processos seqüenciais sejam executados paralelamente, apesar de poderem fazer chamadas ao sistema que bloqueiam processos.

O uso de fluxos de execução está crescendo porque apresentam algumas das características de processos tradicionais, mas podem ser executados de maneira mais eficiente.

Comunicação Entre Processos

Quando tratamos de comunicação entre processos dois mecanismos são possíveis:

- **Comunicação por meio de memória compartilhada**
  - Requer que os processos usem uma área de armazenamento comum e que o código para implementar as operações de armazenagem e leitura de valores da área de armazenamento seja escrito explicitamente pelo programador da aplicação.
- **Sistema de troca de mensagens**
  - Uma outra maneira de se obter a comunicação entre processos é permitida pelo sistema operacional. Esse mecanismo permite que os processos se comuniquem uns com os outros e realizem operações de modo sincronizado, através da troca de mensagens.
- Os esquemas de comunicação por meio de memória compartilhada e de sistema de troca de mensagens não são mutuamente exclusivos e podem ser usados simultaneamente em um único sistema operacional ou mesmo em um único processo.

  - **Estrutura Básica**
    - Os sistemas de troca de mensagens têm a função de permitir que os processos se comuniquem entre si sem a necessidade de usar variáveis compartilhadas.
    - Um mecanismo de suporte a esse sistema deve fornecer pelo menos duas operações, para enviar e receber uma mensagem.
    - Se dois processos P e Q querem se comunicar, eles devem enviar mensagens para receber um do outro; deve existir um canal de comunicação entre eles.
    - Nesse capítulo não vamos nos preocupar com métodos para implementação física desse canal de comunicação (como memória compartilhada, barramento de hardware ou rede de comunicação).
    - Para que dois processos se comuniquem, eles devem ter uma maneira de se referir um ao outro. Eles podem usar tanto comunicação direta quanto comunicação indireta:
      - Comunicação Direta – Cada processo que queira se comunicar com outro deve usar explicitamente o nome do processo receptor ou remetente da mensagem.
      - Comunicação Indireta – As mensagens são enviadas e recebidas de caixas postais. Uma caixa postal pode ser vista abstratamente como um lugar no qual processos podem colocar mensagens e do qual mensagens podem ser retiradas.

- Cada caixa postal possui uma identificação única. Nesse esquema, um processo pode se comunicar com outro por intermédio de diversas caixas postais diferentes.
- Dois processos só podem se comunicar se compartilharem o uso de alguma caixa postal.
- Um canal tem sua capacidade determinada pelo número máximo de mensagens que pode estar na fila de mensagens associada ao canal. Há basicamente três maneiras de implementar essa fila.
    - Capacidade zero
    - Capacidade limitada
    - Capacidade ilimitada

Resumo

Um processo é um programa em execução. Quando um processo é executado, ele muda de estado. O estado de um processo é definido pelas atividades que estão sendo realizadas por esse processo. Cada processo pode estar em um dos seguintes estados: novo, pronto, em execução, em espera ou terminado. Cada processo é descrito no sistema operacional por um bloco de controle de processos.

Quando um processo não está sendo executado, ele é colocado em alguma fila de espera. Existem duas grandes classes de filas em um sistema operacional: fila de requisição de E/S e fila de processos prontos. A segunda contém todos os processos que estão esperando pela CPU. Cada processo é representado por um BCP, e os BCPs podem ser ligados para formar uma fila de processos prontos. O escalonador de processos faz uma seleção de processos que podem passar a competir pela CPU. Normalmente, essa seleção é bastante influenciada por considerações relativas à alocação de recursos, especialmente o gerenciamento de memória. A alocação de CPU consiste em selecionar um processo da fila de processos prontos.

Existem diversas razões para permitir a execução concorrente de processos em um sistema: compartilhamento de informações, aumento na velocidade de processamento, modularidade e conveniência. Para suporte à execução concorrente, é preciso um mecanismo para criação e finalização de processos.

A execução de dois ou mais processos pode ser feita de modo independente ou cooperativo. Dois processos cooperativos devem ter meios de comunicarem um com o outro. Existem basicamente dois esquemas de comunicação complementares: memória compartilhada e troca de mensagens. O método de

memória compartilhada requer que processos em comunicação compartilhem o uso de variáveis. Em um sistema de memória compartilhada, a responsabilidade de estabelecer uma comunicação é dos programadores de uma aplicação; o sistema operacional precisa fornecer apenas a memória compartilhada. No método de troca de mensagens, a responsabilidade de estabelecer uma comunicação é do próprio sistema operacional. Esses dois esquemas não são mutuamente exclusivos e podem ser usados ao mesmo tempo em um único sistema operacional.

Os processos cooperativos que compartilham diretamente um mesmo espaço de endereços lógicos podem ser vistos como fluxos de execução. Um fluxo de execução é uma unidade básica de utilização da CPU e compartilha com fluxos irmãos a seção de código, a seção de dados e os recursos do sistema operacional, que são coletivamente conhecidos como uma tarefa. Uma tarefa não faz nada se não tiver algum fluxo de execução a ela associado, e um fluxo de execução deve estar associado a apenas uma tarefa. O compartilhamento extensivo de recursos entre fluxos de execução irmãos torna mais eficiente a mudança de contexto entre eles, assim como a criação de novos fluxos, em comparação com as trocas de contexto entre processos tradicionais.