


**Universidade Federal do Rio Grande do Norte
Unidade Acadêmica Especializada em Ciências Agrárias
Escola Agrícola de Jundiaí
Curso de Análise e Desenvolvimento de Sistemas
TAD0006 - Sistemas Operacionais - Turma 01**

Processos

Antonino Feitosa
antonino.feitosa@ufrn.br

Macaíba, março de 2025



Roteiro

- Modelo
 - Criação
 - Término
 - Hierarquia
 - Estados
 - Implementação
 - Modelagem de Multiprogramação
-

Introdução



Introdução

- Computadores modernos executam várias tarefas ao mesmo tempo.
- Exemplos:
 - Servidor web: tratamento de solicitações e acesso no armazenamento persistente.
 - Computadores pessoais: vários programas são carregados durante a inicialização, além dos programas executados pelo usuário.
- Como isso ocorre?

Introdução

- Processo: uma abstração de um programa em execução.
 - Conceito central em todos sistema operacional!
- Necessário para a multiprogramação.
 - Alternância entre processos rapidamente, permitindo que uma CPU forneça uma ilusão de paralelismo, porém, executando um único processo de cada vez.
 - Pseudoparalelismo: para diferenciar do paralelismo real em sistemas de multiprocessadores.
 - Como a multiprogramação funciona?

Modelo de Processo

Modelo de Processo

- Todos os programas executáveis são organizados como uma série de processos sequenciais.
 - Geralmente inclui o sistema operacional.
- Assumimos que o sistema possui uma única CPU para facilitar o entendimento dos processos.
 - Em geral, os sistemas atuais são multinúcleos.
- Cada processo executa em uma CPU virtual para que seja possível efetuarmos a multiprogramação.

Modelo de Processo

- Processo é uma uma instância do programa em execução.
 - Armazenando todos o contexto associado com a execução do programa.
 - Podemos ter vários processo gerados da execução de um mesmo programa.
 - Analogia com a receita de bolo.
- A CPU é compartilhada entre os diferentes processos.
 - O processo permite resgatar o contexto de execução do programa, dando continuidade à sua tarefa, como se a CPU fosse dedicada.

Modelo de Processo

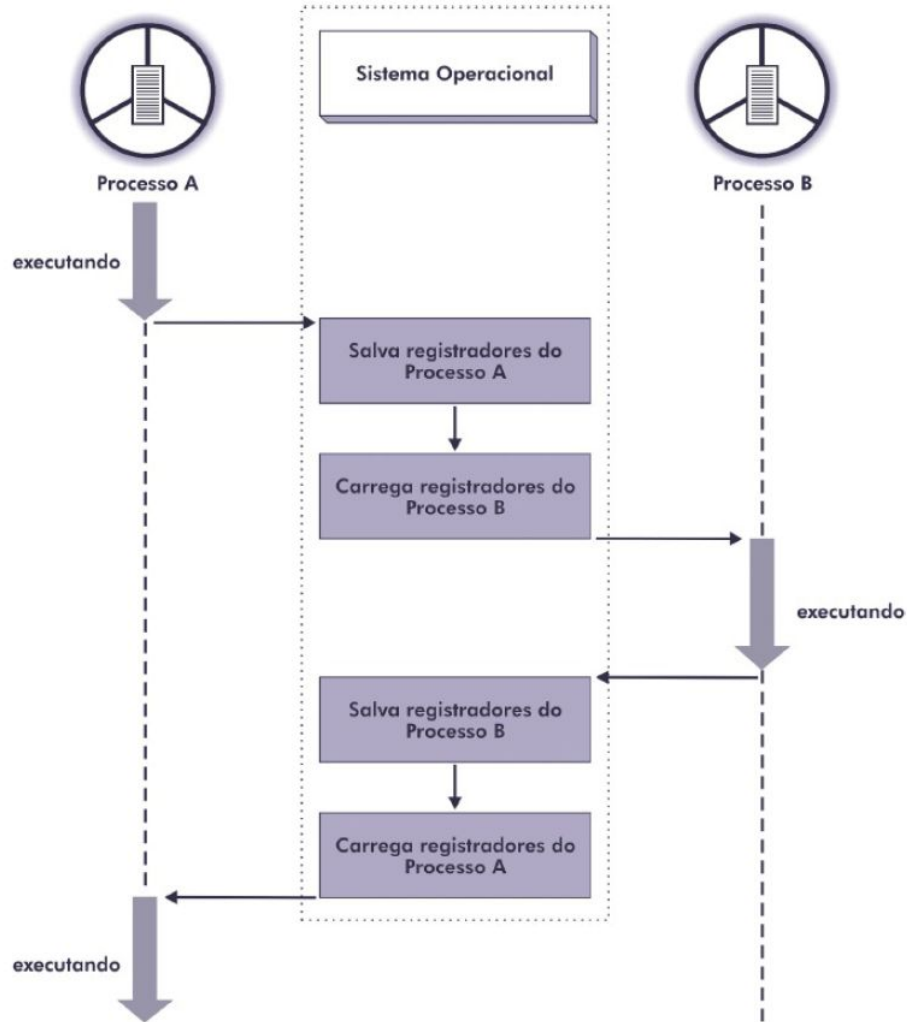
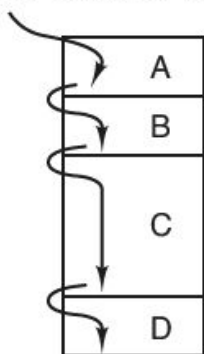


Fig. 5.3 Mudança de contexto.

Modelo de Processo

FIGURA 2.1 (a) Multiprogramação de quatro programas. (b) Modelo conceitual de quatro processos sequenciais independentes. (c) Apenas um programa está ativo de cada vez.

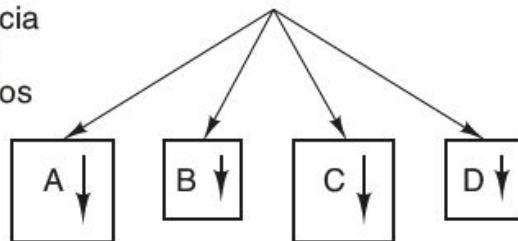
Um contador de programa



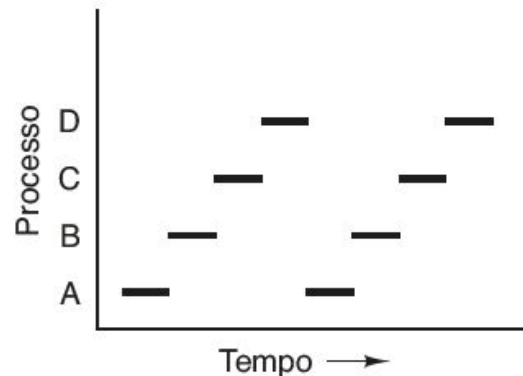
(a)

Quatro contadores de programa

Alternância
entre
processos



(b)

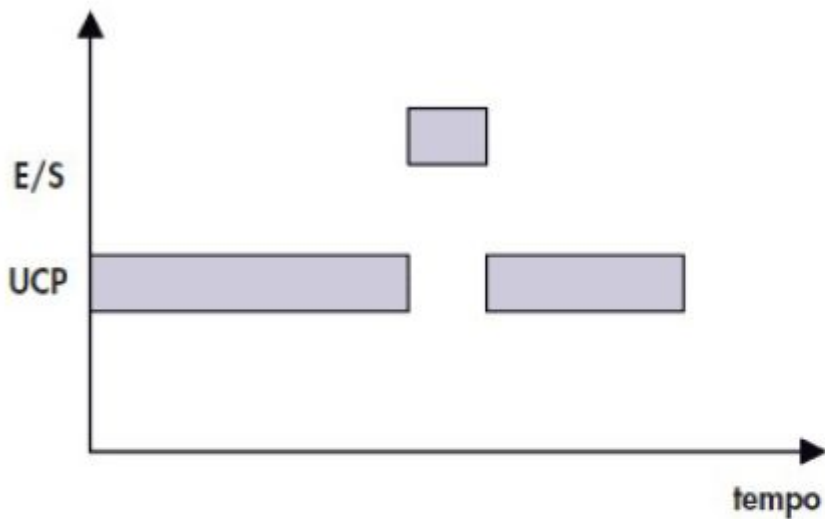


(c)

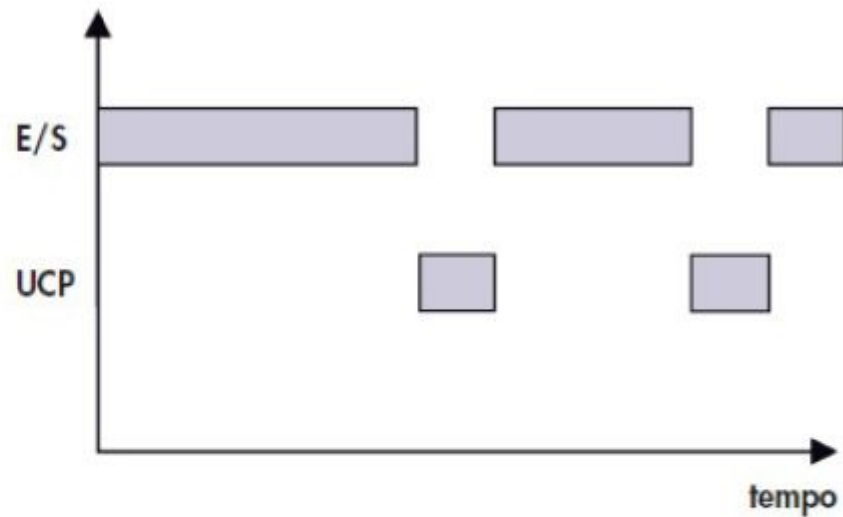
Modelo de Processo

- Processos podem ser classificados conforme a utilização da CPU e dos dispositivos de E/S.
- CPU-bound (ligado à UCP) quando passa a maior parte do tempo no estado de execução, utilizando o processador.
 - Realiza poucas operações de leitura e gravação, e é encontrado em aplicações científicas que efetuam muitos cálculos.
- 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.
 - Aplicações comerciais, que se baseiam em leitura, processamento e gravação.
 - Aplicações interativas.

Modelo de Processo



(a) CPU-bound



(b) I/O-bound

Fig. 5.11 Processos CPU-bound × I/O-bound.

Modelo de Processo

- Como consequência da multiprogramação, o tempo real de uso da CPU será variável.
 - Mesmo se a execução considerar os mesmo programas.
 - Não devemos programar considerando o tempo de processamento.
 - Exemplo: sincronização entre áudio e vídeo; taxa de atualização de elementos de um jogo digital.
 - Tratado pela comunicação entre processos.
- Quem decide qual processo terá acesso à CPU?

Criação de Processos

Criação de Processos

- Sistemas operacionais precisam prover meios para criação de processos.
 - Chamadas de sistema.
- Exemplos:
 - Processos podem ser criados no momento da inicialização do sistema.
 - Em sistemas de propósito geral, processos podem ser criados e encerrados conforme a necessidade de processamento.

Criação de Processos

- Principais eventos de criação de processos.
 - Inicialização do sistema;
 - Execução de uma chamada de sistema de criação de processo por um processo em execução.
 - Solicitação de um usuário para criar um novo processo.
 - Início de uma tarefa em lote.

Criação de Processos: Inicialização

- Processos de primeiro plano: processos que interagem com usuário para execução de alguma tarefa.
 - Exemplo: processador de textos.
- Processos de segundo plano (plano de fundo): não necessitam de interação direta com o usuário.
 - Exemplo: notificação de mensagens de e-mail.
 - Processos daemon: processos de segundo plano que não interagem com o usuário e são executados de forma contínua.
 - Exemplos: sincronização da hora com a rede; servidor de banco de dados.

Criação de Processos: Inicialização

(a) Processo Foreground



(b) Processo Background



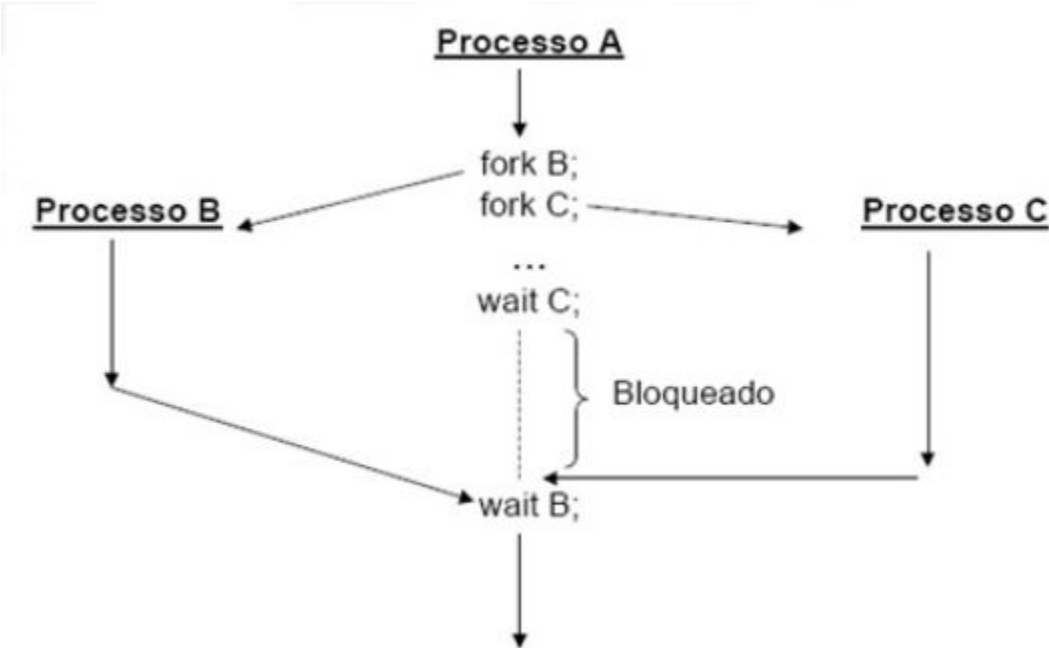
Fig. 5.12 Processos foreground e background.

Criação de Processos

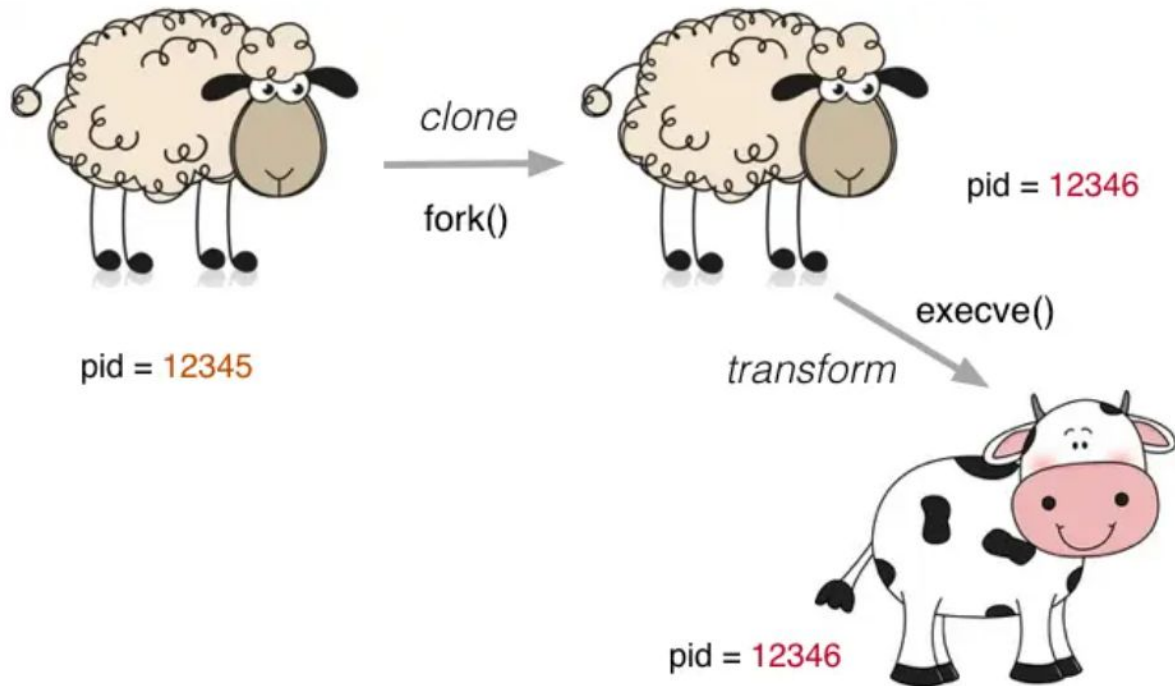
- Os processos são criados por chamadas de sistema.
 - Sistemas UNIX: **fork** - cria um clone do processo que o invocou.
 - Mesma imagem de memória, variáveis de ambiente e recursos.
 - Geralmente seguida da chamada **execve**: alteração da imagem de memória.
 - Sistemas Windows: CreateProcess - possui 10 parâmetros indicando programa, parâmetros de linha de comando, opções de segurança, prioridade, acesso aos recursos.

Criação de Processos

- Os processos são criados por chamadas de sistema.
- Os processos criados possuem espaço de endereçamento distintos.
 - Regiões diferentes na memória principal.
 - Modificações na memória são independentes entre os processos.
- Cada processo possui um identificador único chamado de PID.



Criação de Processos



Término de Processos

Término de Processos

- Principais condições de término:
 - Saída normal (voluntária).
 - Chamada de sistema para encerrar o processo (Exit ou Exit-Process).
 - Erro fatal (involuntário).
 - Situação adversa que não foi tratada pelo programa.
 - Saída por erro (voluntária).
 - O programa pode lidar com o erro, mas opta por encerrar.
 - Morto por outro processo (involuntário).
 - Processo solicita o encerramento de outro (Kill ou TerminateProcess).
 - Necessita de autorização.
 - O que acontece com os processo filho do processo morto (hierarquia)?

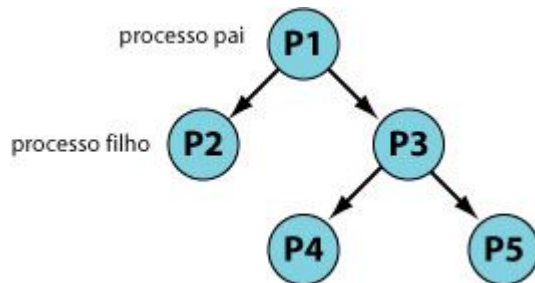
Hierarquia de Processos

Hierarquia de Processos

- Geralmente, os processo criados estão associados aos processos que efetuaram a chamada de sistema.
 - Processo pai e filho.
 - Hierarquia de processos.
- A hierarquia pode formar um grupo de processos.
 - A comunicação, sinais, são enviados para o grupo.

Hierarquia de Processos

- Sistemas UNIX utilizam o conceito de hierarquia de processo.
 - Processos podem alterar o seu grupo.
- Os sistemas Windows não trabalha com hierarquia.
 - Processos independentes.



Estados de Processos

Estados de Processos

- Cada processo possui um contexto independente dos demais.
- Processos podem interagir.
 - Solicitar dados ou processamento.
 - O processo pode optar por esperar pela interação (bloqueado).
 - Depende da lógica do programa.
- O sistema operacional pode bloquear um processo.
 - O processo está pronto e capaz de executar, mas o SO decidiu alocar o CPU para outra tarefa.

Estados de Processos: Estados

- **Em execução:** realmente usando a CPU naquele instante.
- **Pronto:** executável, temporariamente parado para deixar outro processo ser executado.
- **Bloqueado:** incapaz de ser executado até que algum evento externo aconteça.



Estados de Processos: Estados

FIGURA 2.2 Um processo pode estar nos estados em execução, bloqueado ou pronto. Transições entre esses estados ocorrem como mostrado.



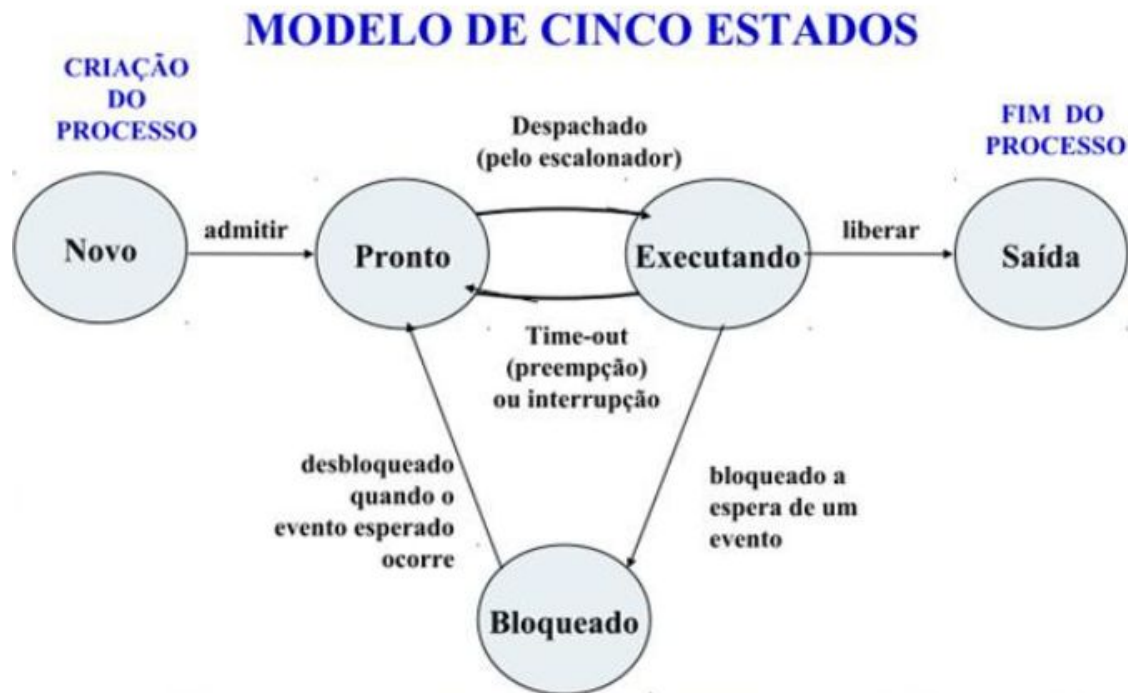
1. O processo é bloqueado aguardando uma entrada
2. O escalonador seleciona outro processo
3. O escalonador seleciona esse processo
4. A entrada torna-se disponível

Estados de Processos: Estados

Transições:

1. SO detecta que o processo não pode continuar.
 - a. Solicitação de dados ou então chamada de sistema pause ou sleep.
 - b. Interrupções.
2. SO decide que o processo executou por tempo suficiente.
 - a. Escalonador: controla a alternância entre os processos.
3. SO decide que o processo deve ser executado.
4. Ocorrência de evento externo.
 - a. Dados disponíveis.

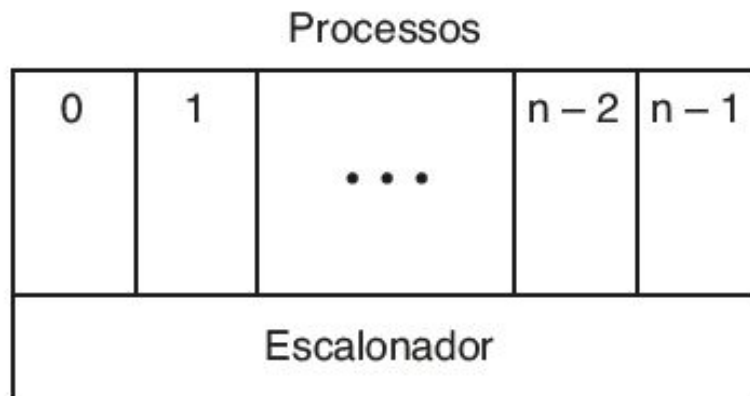
Estados de Processos



Estados de Processos: Fila de Processos

FIGURA 2.3

O nível mais baixo de um sistema operacional estruturado em processos controla interrupções e escalonamento. Acima desse nível estão processos sequenciais.



Implementação de Processos

Implementação de Processos

- O sistema operacional armazena o contexto de cada processo em uma **tabela de processos**.
 - Estrutura de dados.
 - Uma entrada para cada processo.
- Contêm as informações sobre o estado do processo.
 - Informação de contexto (registradores)
 - Informações de escalonamento (estado, tempo de execução, etc.)
 - Entre outras informações.

Implementação de Processos

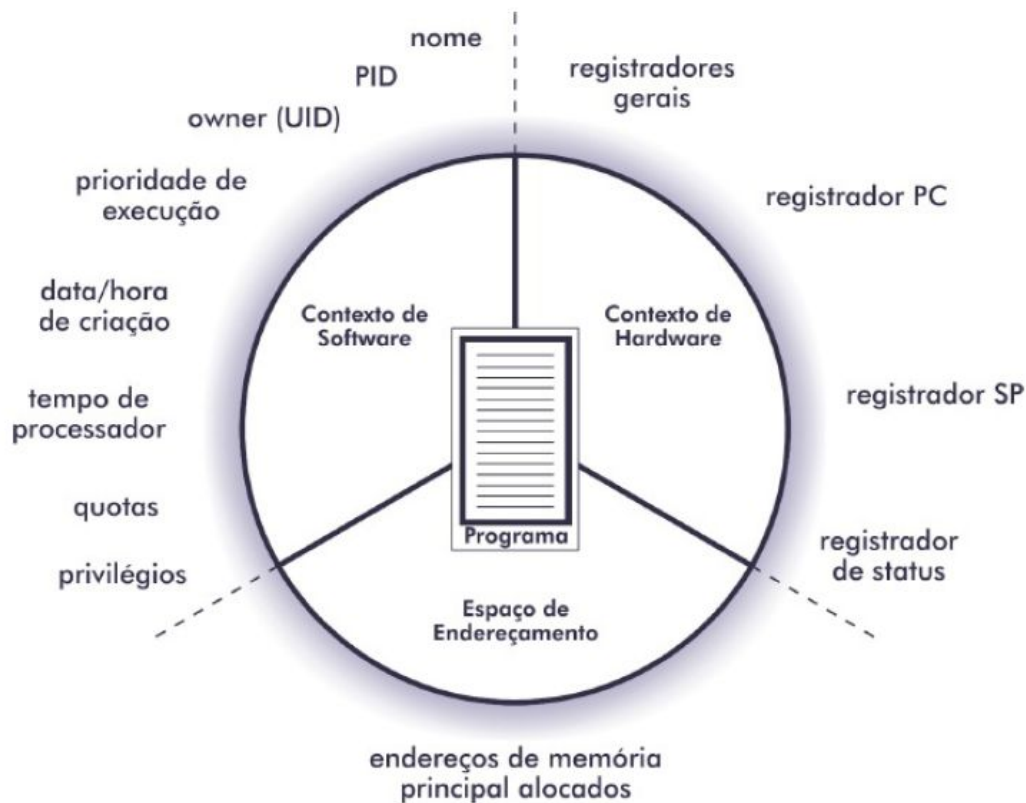


Fig. 5.4 Características da estrutura de um processo.

Implementação de Processos

FIGURA 2.4 Alguns dos campos de uma entrada típica na tabela de processos.

Gerenciamento de processo	Gerenciamento de memória	Gerenciamento de arquivo
Registros Contador de programa Palavra de estado do programa Ponteiro da pilha Estado do processo Prioridade Parâmetros de escalonamento ID do processo Processo pai Grupo de processo Sinais Momento em que um processo foi iniciado Tempo de CPU usado Tempo de CPU do processo filho Tempo do alarme seguinte	Ponteiro para informações sobre o segmento de texto Ponteiro para informações sobre o segmento de dados Ponteiro para informações sobre o segmento de pilha	Diretório-raiz Diretório de trabalho Descritores de arquivo ID do usuário ID do grupo

Interrupções

- São responsáveis pelas transições de estado do processo.
- A memória armazena um local reservado chamado de vetor de interrupção associada a cada classe de E/S.
 - Contém endereço para a rotina adequada para o tratamento da interrupção.

Interrupções

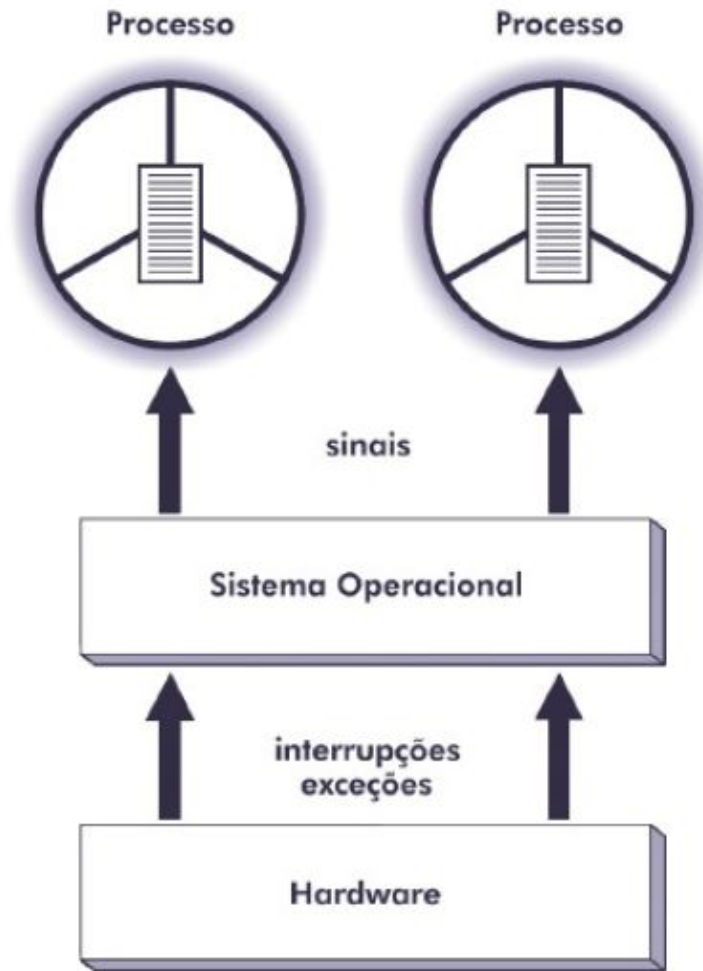


Fig. 5.18 Sinais, interrupções e exceções.

Interrupções

FIGURA 2.5

O esqueleto do que o nível mais baixo do sistema operacional faz quando ocorre uma interrupção.

1. O hardware empilha o contador de programa etc.
2. O hardware carrega o novo contador de programa a partir do arranjo de interrupções.
3. O vetor de interrupções em linguagem de montagem salva os registradores.
4. O procedimento em linguagem de montagem configura uma nova pilha.
5. O serviço de interrupção em C executa (em geral lê e armazena temporariamente a entrada).
6. O escalonador decide qual processo é o próximo a executar.
7. O procedimento em C retorna para o código em linguagem de montagem.
8. O procedimento em linguagem de montagem inicia o novo processo atual.

Interrupções

- Um processo pode ser interrompido milhares de vezes durante sua execução.
- Após cada interrupção, o processo retorna precisamente para o mesmo estado em que se encontrava antes de ser interrompido.
 - Do ponto de vista do processo, nada aconteceu.

Modelando Multiprogramação



Modelando Multiprogramação

- Colocando a questão de maneira direta, se o processo médio realiza computações apenas 20% do tempo em que está na memória, então com cinco processos ao mesmo tempo na memória, a CPU deve estar ocupada o tempo inteiro.
 - Modelo otimista: assume que sempre existirá um processo estado de pronto.

Modelando Multiprogramação

Análise probabilística:

- Um processo passa uma fração de tempo p esperando pelos dispositivos de E/S.
 - Fração no intervalo $[0,1]$, onde 1 é o tempo total.
- Com n processos na memória ao mesmo tempo, a probabilidade de que todos os processos n estejam esperando para E/S é p^n .
 - Caso em que a CPU estará ociosa.

Modelando Multiprogramação

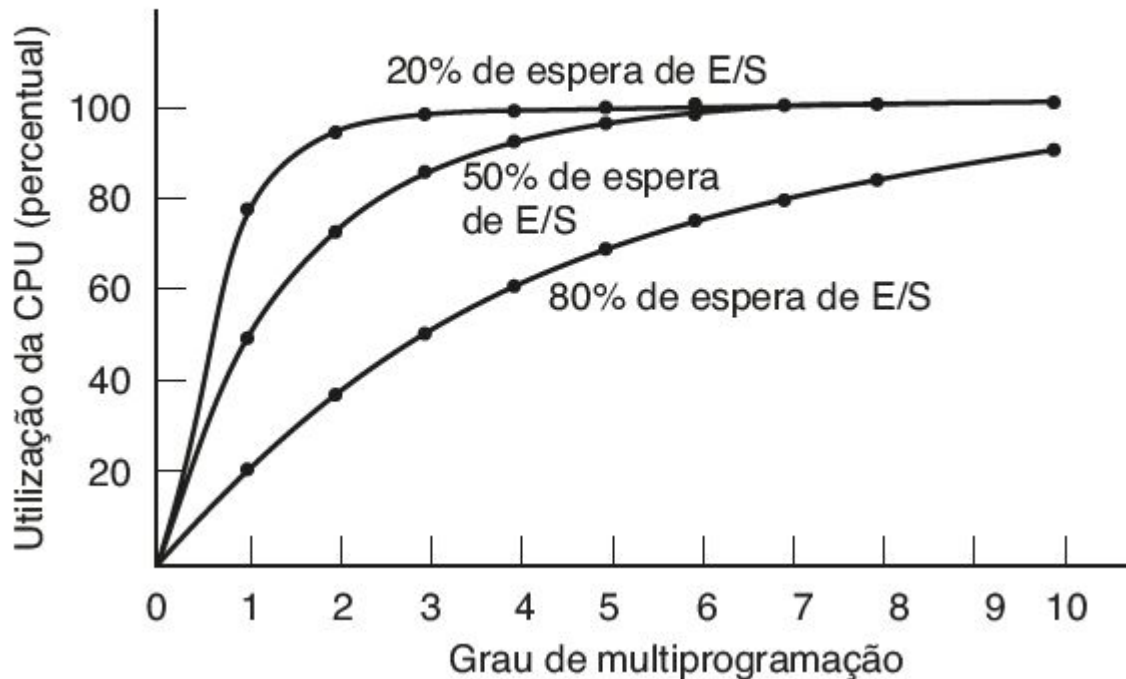
- A utilização da CPU é então dada pela fórmula:

$$\text{Utilização da CPU} = 1 - p^n$$

- n é chamado de grau de multiprogramação.
- O modelo é simples e assume que os processos são independentes.
 - Ignora a comunicação e dependência entre processos.
 - Temos um único processador, logo, um processo em execução e os outros esperando.
 - Pode ser usado para fazer aproximações.

Modelando Multiprogramação

FIGURA 2.6 Utilização da CPU como uma função do número de processos na memória.



Modelando Multiprogramação: Exemplo

- Suponha, por exemplo, que um computador tenha 8 GB de memória, com o sistema operacional e suas tabelas ocupando 2 GB e cada programa de usuário também ocupando 2 GB.
- Esses tamanhos permitem que três programas de usuários estejam na memória simultaneamente.
 - O grau de multiprogramação é 3.

Modelando Multiprogramação: Exemplo

- Com uma espera de E/S média de 80%, temos uma utilização de CPU de $1 - 0,83$ ou em torno de 49%.
 - Ignorando a sobrecarga do sistema operacional.

Modelando Multiprogramação: Exemplo

- Com uma espera de E/S média de 80%, temos uma utilização de CPU de $1 - 0,83$ ou em torno de 49%.
 - Ignorando a sobrecarga do sistema operacional.
- Acrescentar outros 8 GB de memória permite que o sistema aumente seu grau de multiprogramação de três para sete, aumentando desse modo a utilização da CPU para 79%.
 - Em outras palavras, os 8 GB adicionais aumentarão a utilização da CPU em 30%.

Modelando Multiprogramação: Exemplo

- Acrescentar outros 8 GB ainda aumentaria a utilização da CPU apenas de 79% para 91%, desse modo elevando a utilização da CPU em apenas 12% a mais.
- Usando esse modelo, o proprietário do computador pode decidir que a primeira adição foi um bom investimento, mas a segunda, não.

Resumo

Resumo

- Modelo: programa x processo
- Multiprogramação
- Criação: eventos, chamada de sistema, PID
- Término: eventos
- Hierarquia
- Estados: pronto, bloqueado e execução
- Implementação e Interrupções
- Modelagem de Multiprogramação: modelo probabilístico

Dúvidas?

