UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

Rus0013 - Sistemas Operacionais

Aula 03: Processos

Professor Pablo Soares

2022.2

Sumario

- Processos
  - Modelo de Processos
- Criação de Processos
- Término de Processos
- Hierarquia de Processos
- Estado de Processos
- Implementação de Processos

Exemplo Análogo

- Imagine um cientista da computação preparando um bolo, ele tem uma receita e uma cozinha bem suprida com todos os ingrediente
  - Farinha, ovos, açúcar, essência de baunilha etc
- Enquanto prepara o bolo o filho do cientista chega chorando dizendo que uma abelha o picou
- Então o cientista registra onde ele estava na receita, busca um livro de primeiros socorros e começa a seguir as instruções

Exemplo Análogo

- Nessa analogia a receita é o programa (algoritmo expresso por uma notação adequada)
- O cientista é o processador (CPU)
- Os ingredientes são os dados de entrada
- Atender o filho é a alternância de um processo (assar o bolo) para um outro de mais alta prioridade (fornecer cuidados médicos)
- Cada um em um programa diferente (receita X livro de primeiro socorros)
- Quando a picada da abelha for tratada ele voltará ao bolo do ponto em que parou

Processos

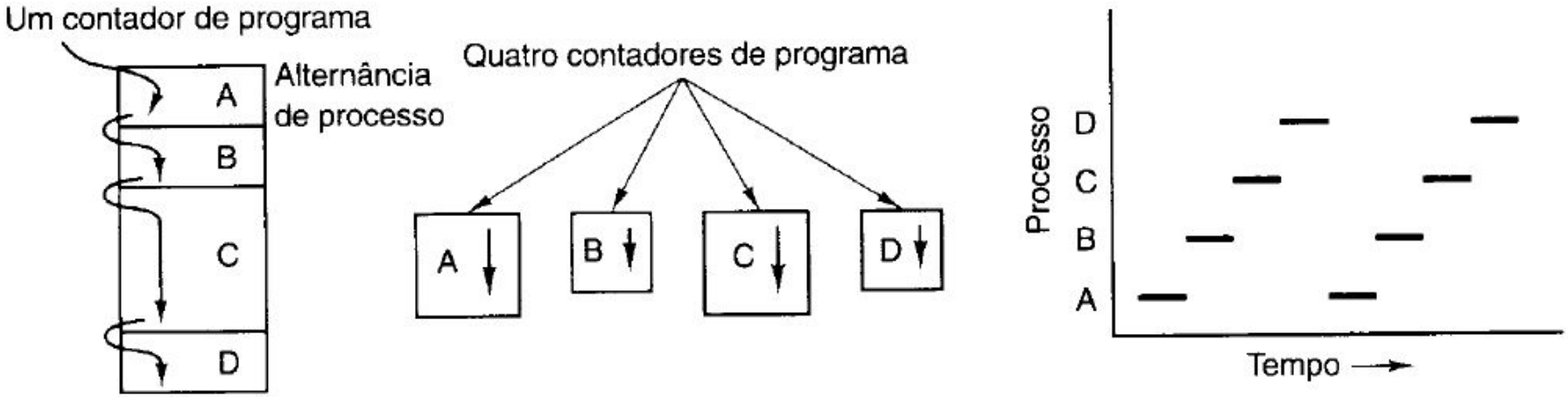
- O conceito mais central em qualquer SO
  - Uma abstração de um programa em execução

- Em um sistema **multiprogramado**, a CPU salta de programa para programa, executando cada um deles por dezenas ou centenas de milissegundos, dando a ilusão de paralelismo (pseudoparalelismo)

- Todos os softwares que podem executar são organizados em processos seqüenciais

Modelo de Processos

- Cada processo tem sua própria CPU virtual, na verdade a CPU troca, a todo momento, de um processo para outro (multiprogramação)



Modelo de Processos

- Cada processo com seu próprio controle de fluxo (isto é, seu próprio PC lógico)

- Mas só há 1 PC físico

Modelo de Processos

- Diferença entre **processo** e **programa**
  - É sutil mas crucial
- **Programa**
  - Algoritmo expresso por uma notação adequada
- **Processo**
  - Constitui uma atividade
    - Programa
    - Entrada e Saída
    - Estado

Criação de Processos

- Há 4 eventos principais que fazem com que processos sejam criados
  - Início do sistema
  - Execução de uma chamada ao sistema de criação por um outro processo
  - Uma requisição do usuário para criar um novo processo
  - Início de uma tarefa em lote

Criação de Processos

- Quando um SO é carregado, em geral criam-se vários processos
  - Alguns em 1º plano (interagem com usuários)
  - Outros em 2º plano que realizam alguma função específica (daemons)
    - Ex: Aviso de mensagens (E-mail)
- No Unix o programa *ps* relaciona os programas em execução,
  - Assim como o CTRL+ALT+DEL no Windows
    - Gerenciador de tarefas

Criação de Processos

- Muitas vezes um processo em execução emitirá chamadas ao sistema para criar um ou mais novos processos para ajudá-lo em seu trabalho
  - Dividir a tarefa entre os processos

- Em sistemas interativos, os usuários podem iniciar um programa digitando um comando ou clicando em um ícone
  - Interação com o mouse

- Em computadores de grande porte o SO cria um novo processo quando julgar existir recursos para executar um outro job, dos jobs em lote

Criação de Processos

- Tecnicamente, em todos os casos, um novo processo é criado por um processo existente executando uma chamada ao sistema
- No Unix há somente 1 chamada ao sistema para criar um novo processo: *fork*
  - Cria uma cópia idêntica do pai, mesma imagem de memória, cadeias de caracteres, arquivos abertos
  - Depois muda a imagem de memória através de *execve*
- No Windows também: CreateProcess (possui 10 parâmetros)

Término de Processos

- Normalmente há 4 razões
  1. Saída normal (voluntária)
  2. Saída por erro (voluntária)
  3. Erro fatal (involuntária)
  4. Cancelamento por um outro processo(involuntária)

1. Na maioria das vezes, os processo terminam por que fizeram seu trabalho.
   - Chamadas ao sistema
     - Unix: exit
     - Windows: ExitProcess

Término de Processos

2. O processo descobre um erro fatal
   - Ex: tentar compilar um arquivo não existente
3. Erro de programa
   - Instrução ilegal, referência à memória inexistente ou divisão por zero
   - No Unix um processo é sinalizado (interrompido) para tentar tratar o erro
4. Um processo executa uma chamada ao sistema para cancelar outro processo quando se tem autorização
   - No Unix: *Kill*
   - No windows: TerminateProcess

Hierarquia de Processo

- Em alguns sistemas, quando um processo cria outro, o pai e o filho continuam, de certa maneira, associados
  - No Unix, um processo, todos os seus filhos e descendentes formam um grupo de processos
  - Quando um usuário envia um sinal de teclado, o sinal é entregue ao grupo, e cada processo decide o que fazer com ele
  - No Windows não existe hierarquia de processos. Quando um processo é criado o pai ainda tem um identificador (handle) que pode controlar o filho

Estados de processos

- Os três estados de um processo
  1. **Em execução** (realmente usando a CPU naquele instante)
  2. **Pronto** (Executável, temporariamente parado para dar lugar a outro processo)
  3. **Bloqueado** (Incapaz de executar enquanto um evento externo não ocorrer)
- Um processo bloqueia por não poder prosseguir ou porque o SO decidiu alocar a CPU para outro processo por algum tempo

Estados de processos

- Logicamente 1 e 2 são similares
- O caso 3 é diferente pois o processo não pode executar, mesmo que a CPU não tenha nada para fazer

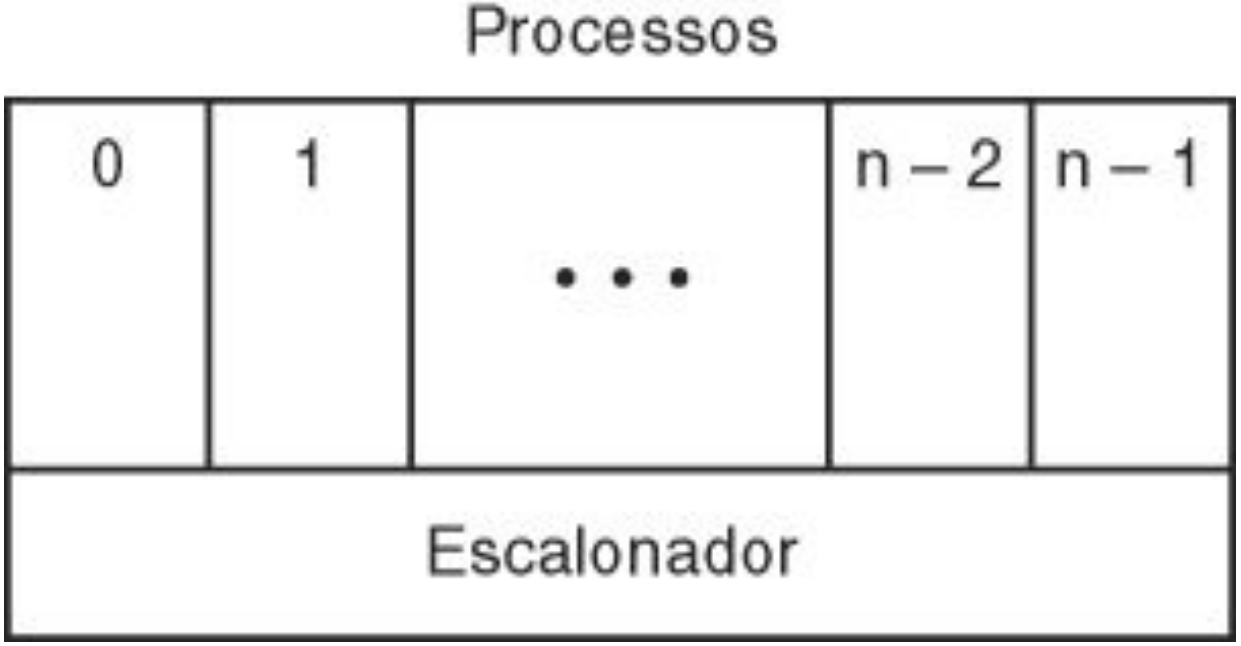
1. O processo bloqueia aguardando uma entrada
2. O escalonador seleciona outro processo
3. O escalonador seleciona esse processo
4. A entrada torna-se disponível

Estados de processos

- A transição 1 ocorre quando um processo descobre que ele não pode prosseguir
- As transições 2 e 3 são causadas pelo escalonador de processos sem que o processo saiba disso
- A transição 4 ocorre quando acontece um evento externo pelo qual o processo estava aguardando

Estados de processos

- O nível mais baixo do SO é o escalonador
  - Todo o tratamento de interrupção e detalhes sobre a inicialização e o bloqueio de processos estão ocultos no escalonador



Implementação de processo

- Para implementar o modelo de processos, o SO mantém uma **tabela de processos**, com uma entrada para cada processo

- Essa tabela mantém informações sobre o estado dos processos (registradores, pilha, memória alocada, arquivos abertos, contabilidade, escalonamento etc.)

Implementação de processo

- Campos de entrada de uma tabela de processos

Gerenciamento de processos	Gerenciamento de memória	Gerenciamento de arquivos
Registradores	Ponteiro para o segmento de código	Diretório-raiz
Contador de programa	Ponteiro para o segmento de dados	Diretório de trabalho
Palavra de estado do programa	Ponteiro para o segmento de pilha	Descritores de arquivos
Ponteiro de pilha		Identificador (ID) do usuário
Estado do processo		Identificador (ID) do grupo
Prioridade		
Parâmetros de escalonamento		
Identificador (ID) do processo		
Processo pai		
Grupo do processo		
Sinais		
Momento em que o processo iniciou		
Tempo usado da CPU		
Tempo de CPU do filho		
Momento do próximo alarme		

Implementação de processo

- Em uma máquina com 1 CPU e muitos dispositivos de E/S, há uma locação de memória para cada classe de dispositivo (**vetor de interrupções**)

- Esse vetor contem os endereços dos procedimentos dos serviços de interrupção

- Quando acontece uma interrupção, o hardware de interrupção coloca na pilha atual as informações do processo interrompido, daí por diante é papel do software fazer o procedimento de serviços da interrupção

Implementação de processo

- Esqueleto do que o nível mais baixo do SO 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.

Modelando a multiprogramação

- Em geral o processo computa
  - 20% do seu tempo em que está na memória
  - Com 5 processos na memória
    - CPU ocupada o tempo todo
      - Modelo Otimista
  - Do ponto de vista probabilístico
    - Fração *p* do tempo esperando E/S
    - Com *n* processos na memória
      - p^n tempo ocioso
      - Utilização CPU = 1 - p^n
- Suponha computador com 512MB
  - 3 processos de 128MB cada
  - 80% do tempo em espera E/S
    - Qual o tempo de utilização de CPU?
    - E se aumentarmos 512MB ?

Referências

- Andrew S. Tanenbaum. “Sistemas Operacionais Modernos”. 3ª Edição, Prentice Hall, 2010.

- Francis B. Machado e Luiz P. Maia. “Arquitetura de Sistemas Operacionais”. 3ª. Edição. LTC, 2004.

UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

Rus0013 - Sistemas Operacionais

Aula 03: Processos

Professor Pablo Soares

2022.2