Sistemas Operacionais

Prof. Me. Pietro M. de Oliveira

Aula de Hoje

Unidade II

Definição de processos e threads.

Comunicação entre processos.

Escalonamento de processos.

Impasses e *Deadlock.*

Introdução

Estrutura de um SO:

- Gerência de processador/processos.
- Gerência de memória.
- Gerência de dispositivos.
- Gerência de Arquivos.
- Gerência de proteção.

Introdução

Gerência de processos

- Sistemas em lote (batch): 1 job por vez.
- Multiprogramação.
  - Várias atividades simultâneas.
- Processamento muito rápido.
  - Falsa sensação de paralelismo.
- SO:
  - Múltiplas requisições de processos.
  - Coordenar acesso a recursos.

Processos

Processo é um programa em execução

Exemplo:

Navegar na internet.

Duplo clique sobre o ícone do navegador.

SO solicita utilização do processador.

Navegador está sendo processado.

Programa em execução: processo!

Em computadores pessoais:

Processamento compartilhado entre processos

CTRL + ALT + DEL

Processos

Endereçamento em memória

- Ler e gravar dados.

Programa em execução:

- Registradores e outras informações.

Troca de contexto:

- SO decide “entregar” o processador.
- Todas as informações devem ser salvas.

Identificadores:

- User Id (UID), Process ID (PID), Group ID (GID).

Processos

Tabela de Processos:

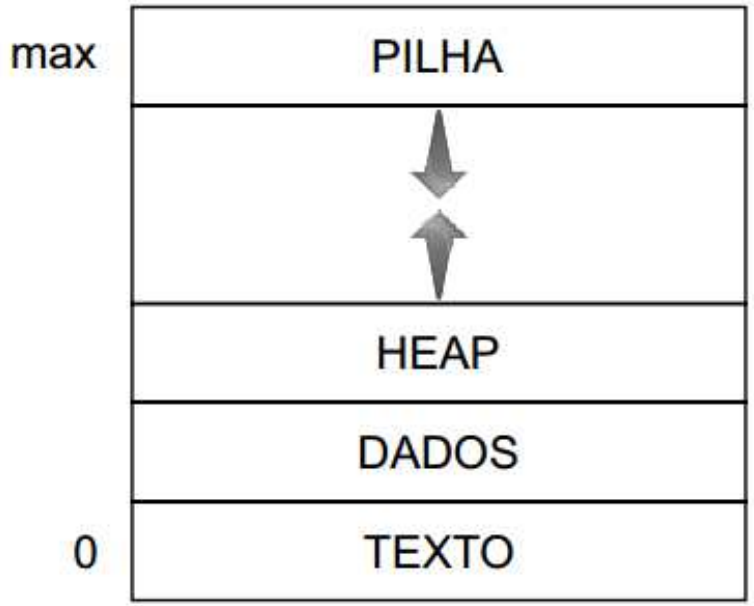
- Registradores.
- Contador do programa.
- Estado do processo.
- Ponteiro de pilha.
- Tempo:
  - Que iniciou, CPU utilizado, próxima troca.
- ID do processo.
- Bits de sinalização.

Estrutura de um processo

- Seção de texto:
  - Código do programa.
  - Contador de programa.
  - Conteúdo dos registradores.
- Seção de dados: variáveis globais.
- *Heap*: alocação dinâmica de memória.
- Pilha: parâmetros de função, variáveis locais.

Pilha: dados temporários

Parâmetros de função, variáveis locais, endereços de retorno.



Processo em memória

Fonte: Silberschatz (2011, p.57)

Recursos

Recursos são componentes necessários para a execução de um processo:

- Tempo de CPU (Processador)
  - Registradores, processamento, cachê.
- Memória.
- Arquivos.
- Dispositivos de entrada/saída.

SO:

- Alocar recursos aos processos.

Situações de criação de processos

Início do sistema.

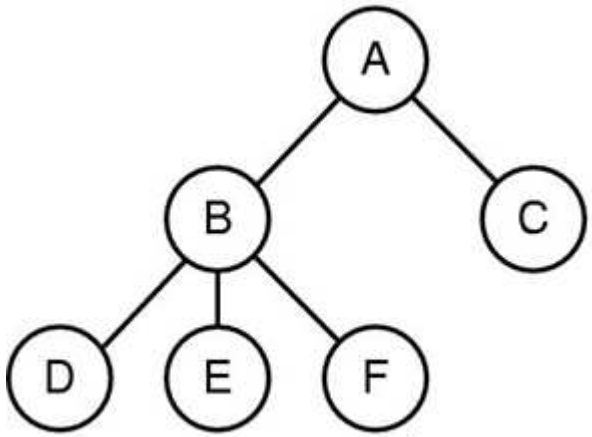
Requisição de um usuário.

Criação de um processo por outro processo.

Trarefas em lote (sistemas de grande porte).

Hierarquia de processos:

- Linux: grupos de proc.
- Windows não possui



Fonte: Tanenbaum (2009, p. 26)

Processos

Situações de destruição de processos

- Término normal voluntário.
- Término por erro voluntário.
- Erro fatal involuntário.
- Cancelamento por outro processo involuntário.

Qual tipo de erro é a famosa “Tela Azul”?

Estados de um processo

Novo (*New*)

Processo recém inicializado.

Em execução (*Running*)

Instruções sendo executadas.

Em espera (*Waiting* ou *Blocked*)

Aguardando resposta externa (E/S).

Esperando por algum recurso.

Estados de um processo (continuação)

- Pronto (*Ready*)
  - Está pronto para ser executado.
  - Aguardando por tempo de CPU.
  - Estava em execução, porém seu *quantum* expirou.
  - Estava em espera, porém teve seus recursos liberados ou terminou de fazer E/S.
- Concluído (*Terminated*)
  - Terminou sua execução por algum dos quatro motivos citados.

Interrupções

Interrupção é um sinal de hardware que informa a ocorrência de um evento

- Erros de máquina.
- Relógio (multiprogramação).
- Disco.
- Interface de rede.
- Terminal.
- Interrupção de software.

Exemplo de execução

1. Duplo clique sobre o editor de texto.
2. Tela de carregamento (New).
3. Cursor piscando (Ready).
4. Texto sendo digitado (Waiting - Running).
5. Cursor piscando (Ready).
6. Clique no botão de impressão (Running).
7. Impressora sem papel (Waiting).
8. Papel adicionado à impressora (Ready).

Exemplo de execução

9. Texto sendo impresso (Running).
10. Impressão concluída (Ready).
11. Clique no X (Running-Terminated).
12. Programa encerrando (Terminated).
13. Programa finalizado.

Threads são as atividades (tarefas) que um processo executa

Exemplo:

Editor de texto + Corretor ortográfico.

Processo Monothread

Sistemas de processamento em lote.

1 *batch job* = 1 *thread*

Processo Multithread

Computadores domésticos (Multicomputadores).

Processos × *Threads*

Um processo pode conter **N** *threads.*

Uma *thread* pertence a **Um** processo.

Todo processo contém ao menos uma *thread.*

Permitem ao processo realizar mais de uma tarefa simultaneamente.

Lightweight process (mini processos).

Processos × *Threads*

Programa Total × Tarefas Isoláveis.

Multiprogramação × *Multithreads.*

Processos compartilham:

Memória, HD, recursos de hardware.

Threads compartilham:

Endereçamento, Arquivos abertos.

Comunicação Entre Processos

SO: garantir comunicação

Bem estruturada e sem interrupções.

Condição de corrida:

Dois ou mais processos ou *threads.*

Resultado de um dependendo do outro.

Exemplo:

Dois processos solicitando o mesmo arq.

Disputa: corrupção dos dados.

Comunicação Entre Processos

Como evitar condições de disputa?

Memória de processos:

- Porção alocada para execução.
- Desta porção, um espaço é compartilhado.

Região Crítica (RC):

- Trecho de código que acessa um espaço de memória compartilhado entre diferentes processos.

Exclusão Mútua: Apenas um processo acessa a RC – evita condições de corrida.

Exclusão Mútua

Critérios para uma boa solução:

- Dois ou mais processos nunca podem estar em suas RCs.
- Nada pode ser afirmado sobre velocidade ou número de CPUs.
- Nenhum processo fora de sua RC pode bloquear outros processos.
- Nenhum processo deve esperar indefinidamente para acessar sua RC.

Exclusão Mútua

Técnicas de exclusão mútua:

- Desativação de Interrupções.
- Variável do Tipo Trava (*LOCK*).
- Chaveamento Obrigatório.
- Solução de Peterson.
- Instrução TSL.
- Semáforos e MUTEX.
- Monitores.

Desabilitar Interrupções

SO emite interrupção ao processador para que um processo seja executado

Solução mais simples possível

Funcionamento:

- Processo acessando sua RC.
- Interrupções desabilitadas.
- Nenhum outro processo pode executar.
- Processamento concluído.
- Reabilitam-se interrupções.

Desabilitar Interrupções

Pontos negativos:

- Processo pode “esquecer” de reabilitar.
- Processo de usuário com maior prioridade.
- Não funciona com múltiplas CPUs

  Apenas a CPU do processo é afetada.

Trava (*LOCK*)

Variável binária compartilhada

- Valor 0 permite o acesso.
- Valor 1 região crítica em uso.

Desvantagem:

- LOCK em si, ja é um recurso compartilhado.
- Acesso simultâneo à RC.

1. Um processo tem permissão concedida.
2. Um segundo processo pede acesso.
3. O primeiro ainda não mudou de 0 p/ 1.

Solução de Peterson

É uma fila de acessos à região crítica

- Um processo solicita a RC.
- Caso já esteja em uso, vai p/ uma fila.
- Região Crítica é liberada.
- O processo da frente ganha acesso.
- O último processo interessado será o último a ser atendido.

Desvantagem

- Espera ociosa.

Test and Set Lock - TSL

Implementação via *Hardware*

- Variável = Registrador.
- Instrução própria - TSL.
- Checagem a cada ciclo de *clock.*
- Desde o processador Intel 8088.

Desvantagem

- Espera ociosa.

Solução p/ desperdício de processamento:

- *SLEEP* e *WAKEUP.*

Semáforos e *Mutex*

Variável especial protegida – abstrata

- Facilita a programação.
- Acesso mediante às operações.
  - *WAIT()* e *SIGNAL()*

Tipos:

- Semáforos binários ou *Mutex.*
- Semáforos de contagem.

Desvantagem

- Erros de programação.

Monitores

Baseado em semáforos e *mutex*:

- Estrutura de mais alto nível.
- Concurrent Pascal, C# e Java.

Considere dois processos em estado PRONTO

- PC com mais de um processador?
- PC com apenas um processador?

Escalonamento de Processos

Um escalonador é um algoritmo que decide qual processo deve utilizar o processador

Quatro situações de escalonamento:

- Processo foi encerrado.
- Novo processo: decidir entre pai e filho.
- Bloqueado por entrada/saída.
- Interrupção de entrada/saída.

Preempção: cada processo tem um tempo máximo de execução (*quantum*).

Escalonamento de Processos

Tipos de algoritmos de escalonamento:

Preemptivos: processo executa até um tempo máximo fixado (*quantum*).

Não-preemptivos: processo executa até que seja bloqueado, ou até que encerre.

Escalonamento de Processos

Objetivos de um escalonador

- Justiça: *quantums* adequados.
- Eficiência: CPU 100% ocupada.
- Tempo de resposta: demora pequena.
- Turnaround: tempo pequeno desde a criação até o término.
- Throughput: alta vazão de processos.

Escalonamento de Processos

Técnicas:

- Primeiro a chegar, primeiro a se servir.
- Tarefa mais curta primeiro.
- Próximo de menor tempo restante.
- Chaveamento circular (Round Robin).
- Escalonamento por prioridade.

FCFS

Primeiro a chegar, primeiro a ser servido

First come, first served – FCFS.

Executa o primeiro processo da fila.

Muito simples, comum em *batch Jobs.*

Desvantagem:

Quando um processo é muito grande, os próximos demorarão a serem executados.

O “dono” do processo terá que aguardar.

SJF

Tarefa mais curta primeiro

Shortest job first – SJF.

Executa o menor processo primeiro.

Comum em *batch jobs.*

Desvantagem:

Quando um processo é muito grande, pode ficar aguardando indefinidamente.

Próximo de menor tempo restante

Considere um sistema preemptivo.

Cada processo possui um ‘tempo total’, estimado à partir do código.

Semelhante ao SJF

> Escolhe-se o processo com menor *tempo de execução total* restante.

Round-Robin

Chaveamento circular ou *Round-Robin.*

Considera preempção.

Semelhante ao FCFS.

- Executa até esgotar *quantum* do processo.

Processo não finalizado, volta pra fila.

- Fica bloqueado até ter acesso novamente.

Processos são atendidos pouco a pouco.

- Falsa sensação de execução múltipla.

Escalonamento por prioridades

Baseado no Round-Robin

- Processos com diferentes prioridades.

Prioridades:

- Estabelecidas internamente (SO).
- Alteradas externamente (Usuário).

Prioridades mais altas terão maior acesso.

Windows: ctrl+alt+del.

Escalonamento por prioridades

Processo com prioridade muito alta:

Monopólio do processador.

Solução:

- Conforme o processo executa, sua prioridade vai caindo.
- Quando houver um processo em ‘pronto’ com prioridade maior, há troca de contexto.

Impasses/*Deadlocks*

Quando dois processos detêm recursos que cada um deles está usando (bloqueando)

Compartilhamento de recursos de uso. **exclusivo**

Considere um conjunto de processos:

Processo A detém um recurso.

Todos os processos dependem deste rec.

Processo B detém um recurso do proc. A

Conjunto de processos em *deadlock.*

Impasses/*Deadlocks*

Segundo Tanembaum (2010) :

“Um conjunto de processos estará em situação de impasse se todo processo que pertence ao conjunto estiver esperando por um evento que somente outro processo desse mesmo conjunto poderá fazer acontecer.”

Impasses/*Deadlocks* - Exemplo

Dois processos A e B

Processo A:

1 – Gravar um DVD.

2 – Escanear uma foto.

Processo B:

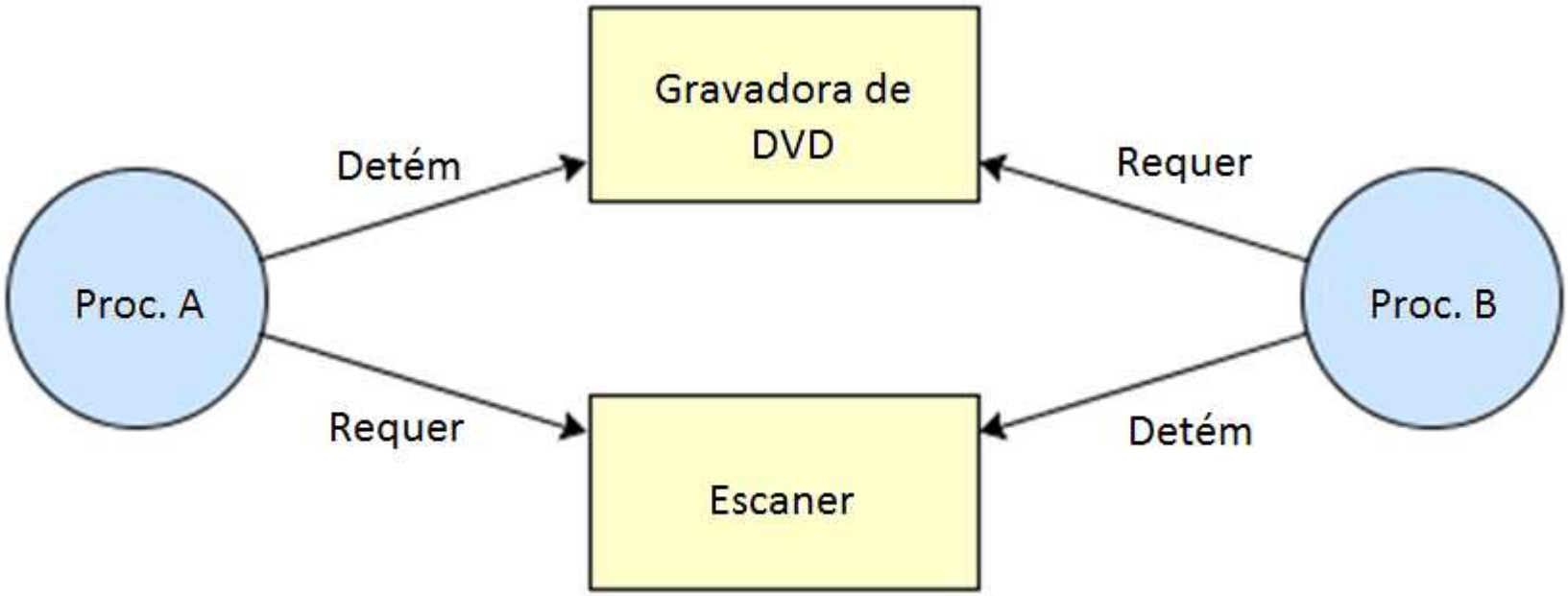
1 – Escanear uma foto.

2 – Gravar em DVD.

Impasses/*Deadlocks* - Exemplo

1. Processo **A** é acionado primeiro.
2. Recurso de DVD é alocado ao processo **A.**
3. Processo **B** é criado.
4. *Quantum* de **A** expira: **B** está em execução.
5. Recurso de scanner é alocado ao processo **B.**
6. *Quantum* de **B** expira: **A** passa a executar.
7. **A** solicita recurso de scanner (**B** bloqueando).
8. *Quantum* de **A** expira: **B** executando.
9. **B** solicita DVD (**A** está bloqueando).

Impasses/*Deadlocks* - Exemplo



Fonte: O autor

Impasses/*Deadlocks*

Condições para impasse (simultâneas)

- Exclusão mútua.
  - Apenas um processo por recurso.
- Posse e espera.
  - Ter um recurso, esperando por outro.
- Inexistência de preempção.
  - Não perde um recurso à força.
- Espera circular.
  - Esperando um recurso do próx. proc.

Prevenção de Impasses

O impasse só ocorre com as quatro condições ocorrendo simultaneamente.

- Garantir que pelo menos uma delas não ocorra:
  - Exclusão mútua.
  - Posse e espera.
  - Inexistência de preempção.
  - Espera circular.

Resumo

Definição de processos

- Estrutura, estados de um processo.

Definição de recursos, interrupções e *threads.*

Comunicação entre processos.

- Região crítica (RC).
- Técnicas de exclusão mútua.

Escalonamento de processos.

Deadlocks e prevenção de impasses.

Sistemas Operacionais

Prof. Me. Pietro M. de Oliveira