Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Sistemas de Computação

SSC 140 - SISTEMAS OPERACIONAIS I
Turmas A e B

Aulas 4 e 5 – Escalonamento de Processos

Profa. Sarita Mazzini Bruschi

Slides de autoria de
Luciana A. F. Martimiano baseados no livro
Sistemas Operacionais Modernos de A. Tanenbaum

Processos

- Multiprogramação:
  - Pseudoparalelismo: coleção de processos sendo executados alternadamente na CPU;
- Um processo é caracterizado por um programa em execução, mas existe uma diferença sutil entre processo e programa:
  - Um processo pode ser composto por vários programas, dados de entrada, dados de saída e um estado (executando, bloqueado, pronto)

Criando Processos

- Processos precisam ser criados e finalizados a todo o momento:
  - Inicialização do sistema;
  - Execução de uma chamada ao sistema de criação de processo realizada por algum processo em execução;
  - Requisição de usuário para criar um novo processo;
  - Inicialização de um processo em *batch* – *mainframes* com sistemas em *batch*;

Criando Processos

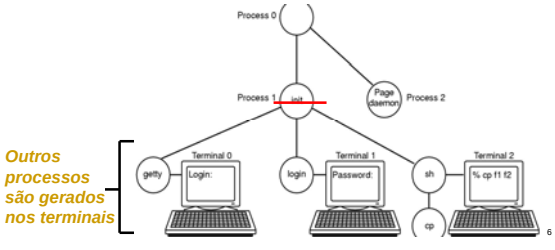
- Processos podem ser:
  - Específicos para usuários específicos:
    - Leitura de um arquivo;
    - Iniciar um programa (linha de comando ou um duplo clique no mouse);
  - Com funções específicas, que independem de usuários, que são criados pelo sistema operacional e que são processados em segundo plano (*daemons*):
    - Recepção e envio de emails;
    - Serviços de Impressão;

Criando Processos

- UNIX:
  - `Fork`;
    - Cria um processo idêntico (filho) ao processo que a chamou (pai), possuindo a mesma imagem de memória, as mesmas cadeias de caracteres no ambiente e os mesmos arquivos abertos;
    - Depois, o processo filho executa uma chamada para mudar sua imagem de memória e executar um novo programa
- Windows:
  - `CreateProcess`
    - Uma única função trata tanto do processo de criação quanto da carga do programa correto no novo processo

Criando Processos

- Exemplo UNIX:
  - Processo `init`: gera vários processos filhos para atender os vários terminais que existem no sistema;



Outros processos são gerados nos terminais

Finalizando Processos

- Condições:
  - Término normal (voluntário):
    - A tarefa a ser executada é finalizada;
    - Chamadas: *exit (UNIX)* e *ExitProcess (Windows)*
  - Término com erro (voluntário):
    - O processo sendo executado não pode ser finalizado: gcc filename.c, o arquivo filename.c não existe;

7

Finalizando Processos

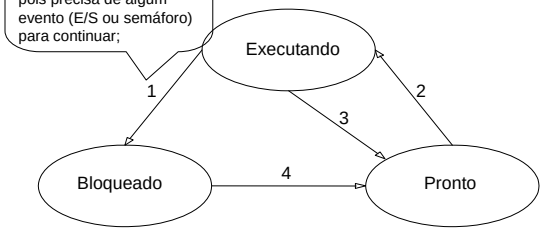
- Condições (continuação):
  - Término com erro fatal (involuntário);
    - Erro causado por algum erro no programa (*bug*):
      - Divisão por 0 (zero);
      - Referência à memória inexistente ou não pertencente ao processo;
      - Execução de uma instrução ilegal;
  - Término causado por algum outro processo (involuntário):
    - `Kill` (UNIX) e `TerminateProcess` (Windows);

8

Estados de Processos

Três estado básicos

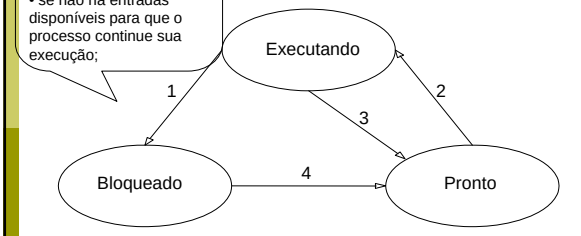
Um processo sendo executado não pode continuar sua execução, pois precisa de algum evento (E/S ou semáforo) para continuar;



9

Estados de Processos

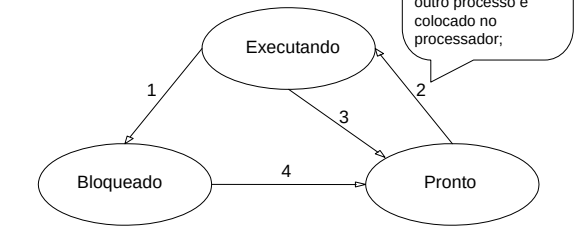
Um processo é bloqueado de duas maneiras:
• chamada ao sistema: `block ou pause`;
• se não há entradas disponíveis para que o processo continue sua execução;



10

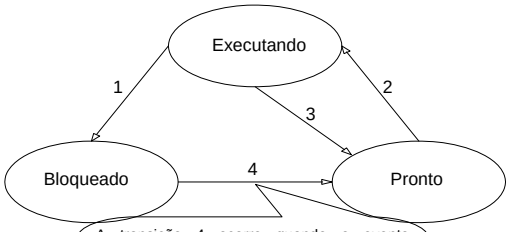
Estados de Processos

As transições 2 e 3 ocorrem durante o escalonamento de processos: o tempo destinado àquele processo acabou e outro processo é colocado no processador;



11

Estados de Processos



A transição 4 ocorre quando o evento esperado pelo processo bloqueado ocorre:
• se o processador está parado, o processo é executado imediatamente (2);
• se o processador está ocupado, o processo deve esperar sua vez;

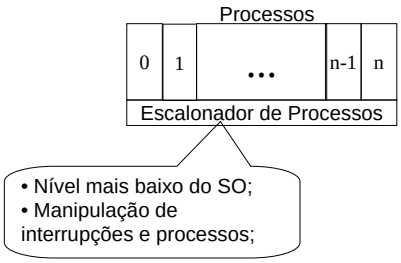
12

Processos

- Processos *CPU-bound* (orientados à CPU): processos que utilizam muito o processador;
  - Tempo de execução é definido pelos ciclos de processador;
- Processos *I/O-bound* (orientados à E/S): processos que realizam muito E/S;
  - Tempo de execução é definido pela duração das operações de E/S;
- **IDEAL**: existir um balanceamento entre processos *CPU-bound* e *I/O-bound*;

13

Escalonador de Processos



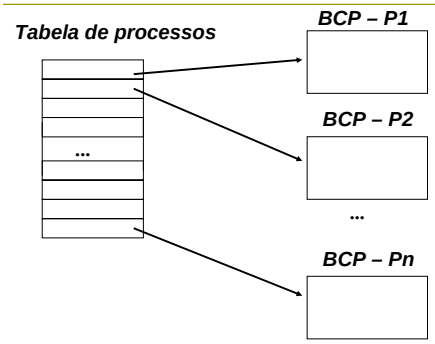
14

Implementação de Processos

- Tabela de Processos:
  - Cada processo possui uma entrada;
  - Cada entrada possui um ponteiro para o bloco de controle de processo (BCP) ou descritor de processo;
  - BCP possui todas as informações do processo → contextos de hardware, software, endereço de memória;

15

Implementação de Processos



16

Implementação de Processos

Process management	Memory management	File management
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Algumas informações do BCP

17

Escalonamento de Processos

- Escalonador de Processos escolhe o processo que será executado pela CPU;
- Escalonamento é realizado com o auxílio do hardware;
- Escalonador deve se preocupar com a eficiência da CPU, pois o chaveamento de processos é complexo e custoso:
  - Afeta desempenho do sistema e satisfação do usuário;
- Escalonador de processo é um processo que deve ser executado quando da **mudança de contexto** (troca de processo);

18

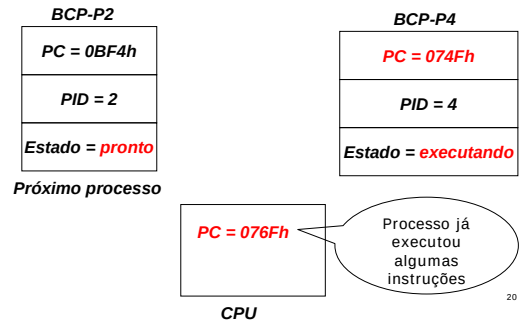
Escalonamento de Processos

- Mudança de Contexto:
  - Overhead de tempo;
  - Tarefa cara:
    - Salvar as informações do processo que está deixando a CPU em seu BCP → conteúdo dos registradores;
    - Carregar as informações do processo que será colocado na CPU → copiar do BCP o conteúdo dos registradores;

19

Escalonamento de Processos

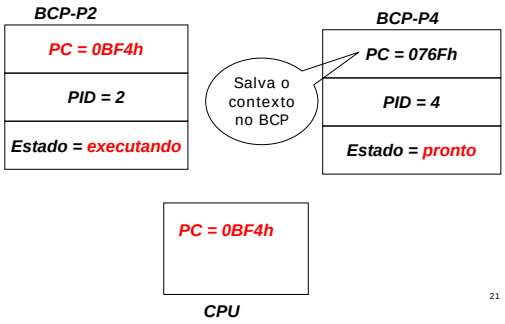
***Antes* da Mudança de Contexto**



20

Escalonamento de Processos

***Depois* da Mudança de Contexto**



21

Escalonamento de Processos

- Situações nas quais escalonamento é necessário:
  - Um novo processo é criado;
  - Um processo terminou sua execução e um processo pronto deve ser executado;
  - Quando um processo é bloqueado (semáforo, dependência de E/S), outro deve ser executado;
  - Quando uma interrupção de E/S ocorre o escalonador deve decidir por: executar o processo que estava esperando esse evento; continuar executando o processo que já estava sendo executado ou executar um terceiro processo que esteja pronto para ser executado;

22

Escalonamento de Processos

- Hardware de relógio fornece interrupções de relógio e a decisão do escalonamento pode ser tomada a cada interrupção ou a cada *k* interrupções;
- Algoritmos de escalonamento podem ser divididos em duas categorias dependendo de como essas interrupções são tratadas:
  - Preemptivo: escolhe um processo e o deixa executando por um tempo máximo;
  - Não-preemptivo: estratégia de permitir que o processo que está sendo executado continue sendo executado até ser bloqueado por alguma razão (semáforos, operações de E/S-interrupção) ou que libere a CPU voluntariamente;

23

Escalonamento de Processos

- Categorias de Ambientes:
  - **Sistemas em Batch**: usuários não esperam por respostas rápidas; algoritmos não-preemptivos ou preemptivos com longo intervalo de tempo;
  - **Sistemas Interativos**: interação constante do usuário; algoritmos preemptivos; Processo interativo → espera comando e executa comando;
  - **Sistemas em Tempo Real**: processos são executados mais rapidamente;; tempo é crucial → sistemas críticos;

24

Escalonamento de Processos

- Características de algoritmos de escalonamento:
  - Qualquer sistema:
    - **Justiça** (*Fairness*): cada processo deve receber uma parcela justa de tempo da CPU;
    - **Balanceamento**: diminuir a ociosidade do sistema;
    - **Políticas do sistema** – prioridade de processos;

25

Escalonamento de Processos

- Características de algoritmos de escalonamento:
  - Sistemas em *Batch*:
    - **Vazão** (*throughput*): maximizar o número de *jobs* executados por hora;
    - **Tempo de retorno** (*turnaround time*): tempo no qual o processo espera para ser finalizado;
    - **Eficiência**: CPU deve estar 100% do tempo ocupada;
  - Sistemas Interativos:
    - **Tempo de resposta**: tempo esperando para iniciar execução;
    - **Proporcionalidade**: satisfação do usuários;

26

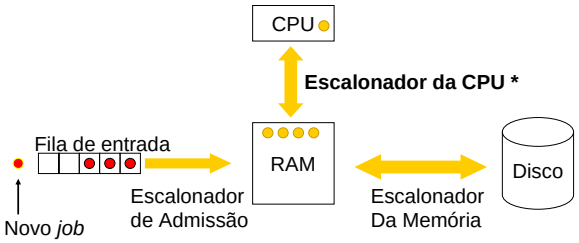
Escalonamento de Processos

- Características de algoritmos de escalonamento:
  - Sistemas em Tempo Real:
    - **Cumprimento dos prazos**: prevenir perda de dados;
    - **Previsibilidade**: prevenir perda da qualidade dos serviços oferecidos;

27

Escalonamento de Processos Sistemas em *Batch*

- Escalonamento *Three-Level*



28

Escalonamento de Processos Sistemas em *Batch*

- Escalonamento *Three-Level*
  - **Escalonador de admissão**: decide qual job será admitido no sistema. Por exemplo, uma mescla de jobs orientados a CPU e orientados à E/S; processos com menor tempo de acesso à CPU e maior tempo de interação com dispositivos de E/S;
  - **Escalonador da Memória**: decisões sobre quais processos vão para a MP:
    - A quanto tempo o processo está esperando?
    - Quanto tempo da CPU o processo já utilizou?
    - Qual o tamanho do processo?
    - Qual a importância do processo?
  - **Escalonador da CPU**: seleciona qual o próximo processo a ser executado;

29

Escalonamento de Processos Sistemas em *Batch*

- Algoritmos para Sistemas em *Batch*:
  - *First-Come First-Served (ou FIFO)*;
  - *Shortest Job First (SJF)*;
  - *Shortest Remaining Time Next (SRTN)*;

30

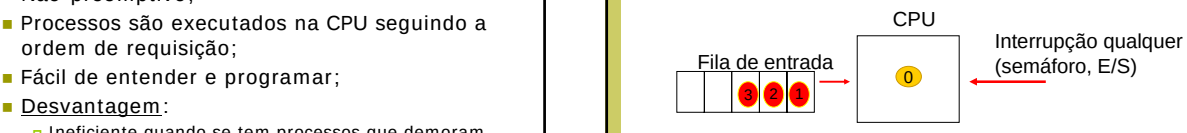
Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo *First-Come First-Served*
  - Não-preemptivo;
  - Processos são executados na CPU seguindo a ordem de requisição;
  - Fácil de entender e programar;
  - Desvantagem:
    - Ineficiente quando se tem processos que demoram na sua execução;

31

Escalonamento de Processos
Sistemas em *Batch*

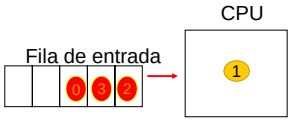
- Algoritmo *First-Come First-Served*



32

Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo *First-Come First-Served*



CPU não controla o tempo dos processos!
(não-preemptivo)

33

Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo *Shortest Job First*
  - Não-preemptivo;
  - Possível prever o tempo de execução do processo;
  - Menor processo é executado primeiro;
  - Menor *turnaround*;
  - Desvantagem:
    - Baixo aproveitamento quando se tem poucos processos prontos para serem executados;

34

Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo *Shortest Job First*

A → a
B → b+a
C → c+b+a
D → d+c+b+a

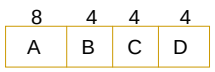
_____
Tempo médio-*turnaround* (4a+3b+2c+d)/4

Contribuição → se a<b<c<d tem-se o mínimo tempo médio;

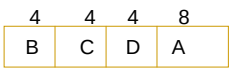
35

Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo *Shortest Job First*



Em ordem:
Turnaround A = 8
Turnaround B = 12
Turnaround C = 16
Turnaround D = 20
Média → 56/4 = 14



Menor *job* primeiro:
Turnaround B = 4
Turnaround C = 8
Turnaround D = 12
Turnaround A = 20
Média → 44/4 = 11

(4a+3b+2c+d)/4 ← Número de Processos

36

Escalonamento de Processos
Sistemas em *Batch*

- Algoritmo *Shortest Remaining Time Next*
  - Preemptivo;
  - Processos com menor tempo de execução são executados primeiro;
  - Se um processo novo chega e seu tempo de execução é menor do que do processo corrente na CPU, a CPU suspende o processo corrente e executa o processo que acabou de chegar;
  - Desvantagem: processos que consomem mais tempo podem demorar muito para serem finalizados se muitos processos pequenos chegarem!

37

Escalonamento de Processos
Sistemas Interativos

- Algoritmos para Sistemas Interativos:
  - *Round-Robin*;
  - Prioridade;
  - Múltiplas Filas;
  - *Shortest Process Next*;
  - Garantido;
  - *Lottery*;
  - *Fair-Share;*
- Utilizam escalonamento em dois níveis (escalonador da CPU e memória);

38

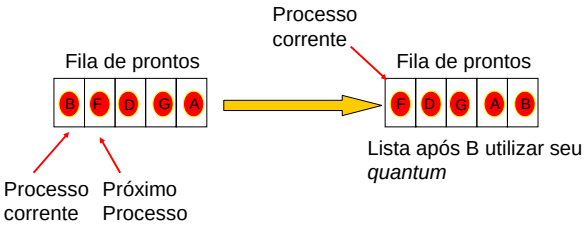
Escalonamento de Processos
Sistemas Interativos

- Algoritmo *Round-Robin*
  - Antigo, mais simples e mais utilizado;
  - Preemptivo;
  - Cada processo recebe um tempo de execução chamado *quantum*; ao final desse tempo, o processo é suspenso e outro processo é colocado em execução;
  - Escalonador mantém uma lista de processos prontos;

39

Escalonamento de Processos
Sistemas Interativos

- Algoritmo *Round-Robin*



40

Escalonamento de Processos
Sistemas Interativos

- Algoritmo *Round-Robin*
  - Tempo de chaveamento de processos;
  - *quantum*: se for muito pequeno, ocorrem muitas trocas diminuindo, assim, a eficiência da CPU; se for muito longo o tempo de resposta é comprometido;

41

Escalonamento de Processos
Sistemas Interativos

- Algoritmo *Round-Robin:*

Exemplos:

quantum

Δt = 4 mseg
x = 1mseg → 25% de tempo de CPU é perdido → menor eficiência

Δt = 100 mseg
x = 1mseg → 1% de tempo de CPU é perdido → Tempo de espera dos processos é maior

chaveamento

quantum razoável: 20-50 mseg

42

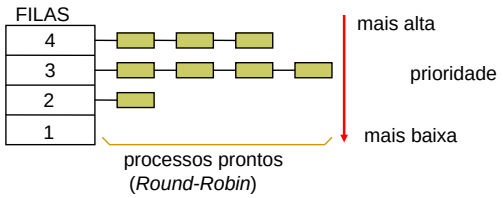
Escalonamento de Processos
Sistemas Interativos

- Algoritmo com Prioridades
  - Cada processo possui uma prioridade → os processos prontos com maior prioridade são executados primeiro;
  - Prioridades são atribuídas dinâmica ou estaticamente;
  - Classes de processos com mesma prioridade;
  - Preemptivo;

43

Escalonamento de Processos
Sistemas Interativos

- Algoritmo com Prioridades



44

Escalonamento de Processos
Sistemas Interativos

- Algoritmo com Prioridades
  - Como evitar que os processos com maior prioridade sejam executado indefinidamente?
    - Diminuir a prioridade do processo corrente a cada interrupção do relógio e trocá-lo pelo próximo processo assim que sua prioridade caia abaixo da prioridade do próximo processo com prioridade mais alta (chaveamento);
    - Atribuir um quantum máximo no qual o processo pode executar;

45

Escalonamento de Processos
Sistemas Interativos

- Múltiplas Filas:
  - CTSS (*Compatible Time Sharing System*);
  - Classes de prioridades;
  - Cada classe de prioridades possui *quanta* diferentes;
  - Assim, a cada vez que um processo é executado e suspenso ele recebe mais tempo para execução;
  - Preemptivo;

46

Escalonamento de Processos
Sistemas Interativos

- Múltiplas Filas:
  - Ex.: um processo precisa de 100 *quanta* para ser executado;
    - Inicialmente, ele recebe um *quantum* para execução;
    - Das próximas vezes ele recebe, respectivamente, 2, 4, 8, 16, 32 e 64 *quanta* (7 chaveamentos) para execução;

47

Escalonamento de Processos
Sistemas Interativo

- Algoritmo *Shortest Process Next*
  - Mesma idéia do *Shortest Job First*;
  - Processos Interativos: não se conhece o tempo necessário para execução;
  - Solução: realizar uma estimativa com base no comportamento passado e executar o processo cujo tempo de execução estimado seja o menor;

48

Escalonamento de Processos
Sistemas Interativo

- Outros algoritmos:
  - Algoritmo Garantido:
    - Garantias são dadas aos processos dos usuários:
      - **n** processos → **1 / n** do tempo de CPU para cada processo;
  - Algoritmo *Lottery*:
    - Cada processo recebe "*tickets*" que lhe dão direito de execução;
  - Algoritmo *Fair-Share*:
    - O dono do processo é levado em conta;
    - Se um usuário A possui mais processos que um usuário B, o usuário A terá prioridade no uso da CPU;

49

Escalonamento de Processos
Sistemas em Tempo Real

- Tempo é um fator crítico;
- Sistemas críticos:
  - Aviões;
  - Hospitais;
  - Usinas Nucleares;
  - Bancos;
  - Multimídia;
- Ponto importante: obter respostas em atraso é tão ruim quanto não obter respostas;

50

Escalonamento de Processos
Sistemas em Tempo Real

- Tipos de STR:
  - ***Hard Real Time***: atrasos não são tolerados;
    - Aviões, usinas nucleares, hospitais;
  - ***Soft Real Time***: atrasos são tolerados;
    - Bancos; Multimídia;
- Programas são divididos em vários processos;
- Eventos causam a execução de processos:
  - **Periódicos**: ocorrem em intervalos regulares de tempo;
  - **Aperiódicos**: ocorrem em intervalos irregulares de tempo;

51

Escalonamento de Processos
Sistemas em Tempo Real

- Algoritmos podem ser estáticos ou dinâmicos;
  - **Estáticos**: decisões de escalonamento antes do sistema começar;
    - Informação disponível previamente;
  - **Dinâmicos**: decisões de escalonamento em tempo de execução;

52