GSI018 – Sistemas Operacionais

Escalonamento de Processos

Universidade Federal de Uberlândia
Faculdade de Computação
Prof. Dr. rer. nat. Daniel D. Abdala

Na Aula Anterior...

- Utilização de Processos em C e Linux;
- Visualizando Processos;
- Iniciando Novos Processos;
- Substituindo o Código de um Processo;
- Duplicando o Código de um Processo;
- Esperando por um Processo;
- Processos Zumbis;
- Redireção de Entrada e Saída.

Nesta Aula

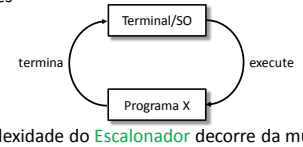
- Ciclos de CPU e E/S;
- Conceitos de Preempção;
- Algoritmos de Escalonamento;
- FCFS – First Come First Served;
- SJF – Shortest Job First;
- Round-Robin;
- Escalonamento por Prioridade;
- Filas Multinível.

Escalonamento

- É uma necessidade, decorrente principalmente devido a multiprogramação;
  - Há a possibilidade de dois ou mais processos estarem no estado PRONTO;
- Módulo do SO → ESCALONADOR
- Algoritmo → Algoritmo de Escalonamento
  - Vários tipos;
  - Depende do que se deseja privilegiar!

Historicamente ...

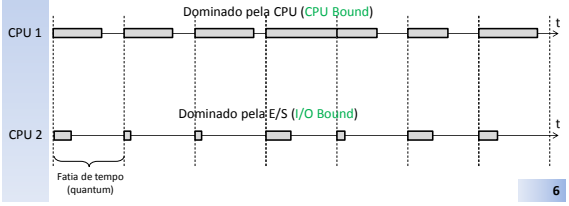
- Sistemas em Lote → Algoritmo Simples
  "Execute o próximo processo na fila"
- Sistemas monotarefa/monousuário (DOS) → Algoritmo Simples



- Complexidade do Escalonador decorre da multitarefa e multiusuário!
- Computadores de Grande Porte
  - Ambos Lote e Multitarefa → Escalonador Complexo!

Comportamento de um Processo

- Processos podem ser classificados quanto ao seu comportamento;
  - Utilizam mais CPU ou mais E/S;
- O Comportamento do processo também pode ser influenciado pelo Sistema Computacional em que ele executa;



Comportamento de um Processo

- CPU é um recurso escasso (depende do contexto);
- Um bom escalonador pode impactar consideravelmente o desempenho do Sistema Computacional;
- Computadores Pessoais → Duas Modificações:
  - Apenas um programa "em primeiro plano";
  - CPUs atualmente são muitíssimo rápidas, deixando assim de ser um recurso escasso;
  - Sistemas iterativos!
  - Escalonador deve levar tais fatos em consideração!
- Servidores → Novamente, Escalonador muitíssimo importante!

O Custo do Chaveamento de Processos

- Chavear processos é muito caro!
  1. Modo Usuário → Modo Kernel;
  2. Salvar o estado do processo no seu PCB;
  3. Executar o algoritmo de Escalonamento;
  4. Carregar o estado do processo selecionado;
  5. Modo Kernel → Modo Usuário.
- Adicionalmente ... (prov. a parte mais cara)
  - Páginas Virtuais Invalidadas
  - Cache invalidada (duas vezes!)

Comportamento de um Processo

- Tendência Geral → Quanto mais rápida a CPU maior a tendência dos processos serem I/O Bound;
- Dois pontos interessantes:
  1. Que processo executar (pai ou filho) após a criação de um novo processo?
  2. Ao término de um processo, o escalonador deve escolher outro processo para ser executado. O que ocorre se não há processos prontos para serem executados?

Tipos de Algoritmos de Escalonamento

- Podem ser organizados como:
  - Preemptivos;
  - Não-Preemptivos;
- Podem ser categorizados como:
  - Algoritmos para sistemas em Lote;
  - Algoritmos para sistemas iterativos;
  - Algoritmos para sistemas de tempo real;
  - Algoritmos híbridos.

Algoritmos Não-Preemptivos

- Escolhe um processo e o deixa executar até que uma das seguintes três condições ocorram:
  - Requeira E/S e deva ser bloqueado;
  - Seja bloqueado pois está a espera de outro processo;
  - Voluntariamente, libera a CPU para o escalonador;
- Implementação relativamente simples;
- Uma boa opção para sistemas de grande porte que devem implementar tanto o sistema em lotes quanto a multitarefa/multiusuário;

Algoritmos Preemptivos

- Escolhe um processo e o deixa em execução por um tempo máximo especificado;
- Ao final de um período de tempo pré-estabelecido (quanta), se o processo ainda estiver em execução, ele será substituído por outro dos processos que estão prontos esperando para executar;
- Requer que uma interrupção de relógio seja gerada em intervalos regulares;
- Definir qual o intervalo (quanta) ótimo para um sistema é uma tarefa complexa;
  - Quanta curto → muita troca de contexto;
  - Quanta longo → diminui a capacidade de resposta (iteratividade) do sistema

Objetivos dos Algoritmos de Escalonamento

- **Todos os Sistemas:**
  - **Justiça** → cada processo recebe uma porção justa da CPU;
  - **Política** → verificar se a política estabelecida é cumprida;
  - **Equilíbrio** → manter ocupada, na medida do possível, todas as partes do sistema;
- **Sistemas em Lote:**
  - **Vazão (throughput)** → maximizar o número de tarefas por unidade de tempo;
  - **Tempo de Retorno** → minimizar o tempo entre submissão e término de processos;
  - **Utilização da CPU** → manter a CPU ocupada o tempo todo;

13

Objetivos dos Algoritmos de Escalonamento

- **Sistemas Iterativos:**
  - **Tempo de Resposta** → responder rapidamente às requisições;
  - **Proporcionalidade** → satisfazer as expectativas dos usuários;
- **Sistemas de Tempo Real:**
  - **Cumprimento de Prazos** → evitar a perda de dados;
  - **Previsibilidade** → evitar a degradação da qualidade em sistemas multimídia;

14

FCFS – First Come First Served

- Sistemas em Lote;
- Primeiro a chegar é o primeiro a ser servido;
- Provavelmente o primeiro e mais simples algoritmo de escalonamento;
- CPU é atribuída aos processos na ordem em que eles a requisitam;
- Processo monopoliza a CPU pelo tempo que ele requerer ou até que o programa requisite uma operação de E/S;
- Basicamente uma fila única de processos prontos;
- Requisições de E/S colocam o processo em estado bloqueado. Processos quando são desbloqueados voltam para o final da fila;

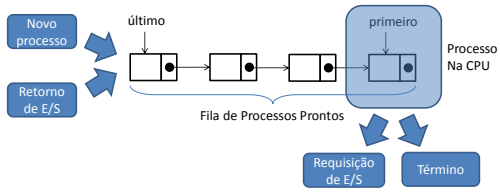
15

FCFS – First Come First Served

- Fácil de entender e programar;
- Algoritmo justo;
- Desvantagens:
  - Não é adequado para processos I/O Bound;
  - Não é adequado para sistemas iterativos;

16

FCFS – First Come First Served



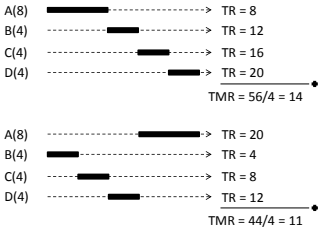
17

SJF – Shortest Job First

- Tarefa mais curta primeiro;
- Sistemas em Lote;
- Utiliza a mesma lista encadeada do FCFS;
- Requer uma rotina que percorra toda a lista de processos prontos a procura daquele com o menor **tempo estimado de processamento**;
- Em alguns casos o tempo que um processo demorará é conhecido ou pode ser estimado;
- Só é aplicável quando todos os processos estão disponíveis simultaneamente (lista de prontos);

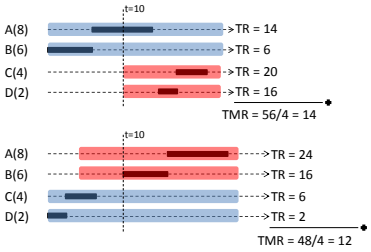
18

SJF – Shortest Job First



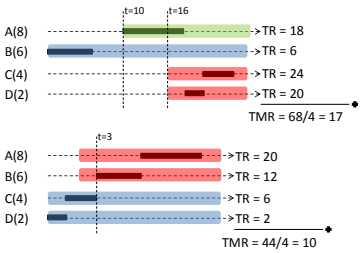
SJF – Shortest Job First

- O que acontece quando processos são inseridos na lista de prontos em tempos distintos?



SJF – Shortest Job First

- O que acontece quando processos são inseridos na lista de prontos em tempos distintos?

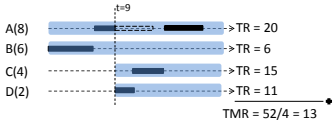


Shortest Remaining Time Next - SRTN

- Próximo Menor Tempo Restante
- Variação preemptiva do escalonamento por prioridade;
- Requer que o tempo de execução seja conhecido a priori;
- A preempção acontece apenas quando um processo pronto tem tempo restante menor que o processo atual;

Shortest Remaining Time Next - SRTN

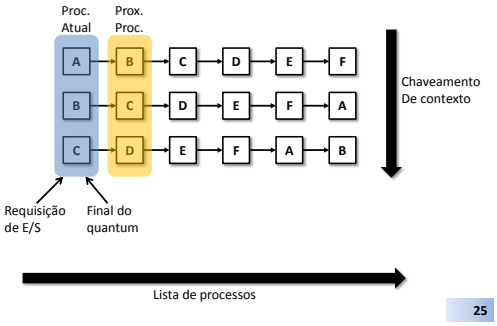
- O que acontece quando processos são inseridos na lista de prontos em tempos distintos?



Round-Robin

- Escalonamento preemptivo;
- Um dos algoritmos mais antigos, simples e justos;
- Amplamente utilizado;
- Implementação simples, lista circular;
- A cada processo é atribuído uma "fatia de tempo" → **quantum**;
- Um processo executa até que:
  - Tenha executado por um tempo igual ao quantum;
  - Requeira E/S ou outro tipo de interrupção do processador;

Round-Robin



Round-Robin

- Em sua realização mais simples todos os processos recebem o mesmo quantum;
- Definir a duração do quantum pode ser problemático;
  - Quantum muito longo → diminui a resposta do sistema;
  - Quantum muito curto → implica em alto custo de chaveamento entre processos;
- Na prática, valores de quantum entre 20 e 50ms são utilizados;

Escalonamento por Prioridades

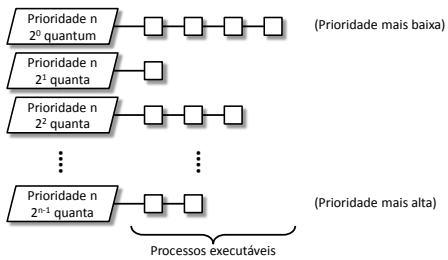
- Escalonamento preemptivo;
- Desdobramento do escalonamento circular;
- Baseado na ideia de que processos não são igualmente importantes para o sistema e/ou usuário;
- O próximo processo a ser escolhido respeita a ordem de prioridades;
- Pode ser implementado via uma fila de prioridades;
- Atribui um número (**prioridade**) a cada processo;
  - Prioridade pode ser alterada estática ou dinâmicamente;

Escalonamento por Prioridades

- Várias formas de ajustar dinâmicamente a prioridade:
  a) Utilização de quatum – especialização do Round-Robin;
  b) A cada interrupção do relógio a prioridade é decrementada; Verifica-se se há algum processo na lista de prontos com prioridade menor que o processo atual;
  c) Decrementa-se a prioridade de um processo quando ele executa durante todo o quantum e incrementa-se a prioridade de processos que bloqueiam antes de terminar o quantum (I/O Bound);

Filas Multinível

- Chaveamento de processos é caro!
- Uma forma de aliviar o custo associado a preempção é executar diferentes processos por diferentes quantidades de tempo;
- No escalonamento por filas multinível, um processo pode ser agendado para execução em diferentes filas de execução:
  - Um processo é escolhido para execução sempre do nível mais alto;
  - Caso não hajam processos no nível mais alto, o nível subsequente é utilizado e assim sucessivamente;
  - Quando um processo executa durante todos os quanta a ele destinados (por sua altura de prioridade nas filas) ele é movido para o nível imediatamente menos prioritário;
  - Se um processo bloqueia antes de terminar o seu quanta ele é movido para o nível imediatamente mais prioritário;



Shortest Process Next

- Shortest Job First sempre resulta no menor tempo médio;
- Como adaptá-lo para preempção?
- O problema é sabem o quanto demora cada programa;
- Solução: Estimar o tempo de execução com base no histórico;
- $E = aT_0+(1-a)T_1$;
  - Qual o efeito de um "a" grande ou pequeno?
  - Qual a faixa de valores válidos para "a"?
- Ex: a = ½
  - T_0; $T_0/2 + T_1/2$; $T_0/4 + T_1/4+T_2/2$; $T_0/8 + T_1/8+T_2/4+T_3/2$ ...

Escalonamento Garantido

- Geralmente utilizado para dividir a atenção da CPU em sistemas multiusuário;
- Uma garantia simples de cumprir seria por exemplo:
  - Caso hajam N usuários conectados no sistema, a CPU será dividida entre os N usuários iqualitáriamente resultado em 1/N - OH tempo de CPU para os processos de cada usuário;
  - OH refere-se ao custo de chaveamento, tempo transcorrido executando o escalonador de processos pelo sistema operacional;

Escalonamento Garantido

- Para fazer valer esta garantia o escalonador tem que manter um controle de quanto tempo já foi gasto pela CPU nos processos de um dado usuário;
- Difícil de implementar;
- Deve manter um controle detalhado não apenas do número de usuários e número de processos por usuário como também do tempo decorrido em cada processo;
- A criação e/ou destruição de um processo assim como a autenticação/saída de um usuário demanda que toda a estrutura seja reformulada;

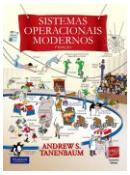
Escalonamento por Loteria

- "Biblites" são atribuídos a cada processo;
- Cada bilhete dá direito ao acesso a um tipo de recurso do computador;
- Randomicamente um bilhete é sorteado;
- O processo detentor do biblhete recebe acesso ao recurso em questão;
- 50 sorteios por segundo significa preempção a cada 20 ms;
- Caso o mesmo processo seja sorteado duas vezes não há necessidade de troca de contexto;
- Prioridade pode ser implementada conferindo mais bilhetes ao mesmo processo

Escalonamento por Fração Justa

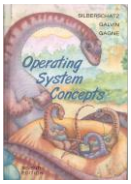
- Se considerarmos apenas os processos e não os usuários uma situação um "injusta" pode ocorrer;
  - usuário A → 10 processos; usuário B → 1 processo;
  - usuário A → 90% do tempo de CPU, usuário B → 10%;
- A cada usuário é dado uma parcela do tempo da CPU;
- Varia com o número de usuários autenticados;
- Um processo só é escalonado caso ele pertença a um usuário ainda com tempo de CPU para utilizar;

Bibliografia - Básica



- 3ª Edição
- Páginas 87-97

Bibliografia - Básica



- 7ª edição
- Páginas 85-90

Bibliografia - Adicional

- Páginas 258-275