Instituto Federal de Santa Catarina - Câmpus Garopaba

Tecnologia em Sistemas para Internet

Sistemas Operacionais

Alberto Felipe Friderichs Barros
alberto.barros@ifsc.edu.br

INSTITUTO FEDERAL
Santa Catarina

Escalonamento de Processos

Introdução:

Num sistema de tempo compartilhado, **vários processos compartilham uma mesma CPU**. Periodicamente, o S.O. decide parar de executar um processo e começar a executar outro porque o primeiro já teve a porção que lhe cabe da CPU.

INSTITUTO FEDERAL
Santa Catarina

Quando um processo é suspenso temporariamente da forma acima descrita, ele tem que ser reinicializado mais tarde, exatamente no mesmo estado em que se encontrava quando foi interrompido. Isto significa que toda a informação sobre o processo tem que ser explicitamente salva durante a suspensão deste processo **(Scheduler).**

INSTITUTO FEDERAL
Santa Catarina

Sempre que a CPU se torna ociosa, o sistema operacional deve selecionar um dos processos na fila de prontos para ser executado. O processo de seleção é realizado pelo Scheduler de curto prazo que seleciona um processo dentre os prontos na memória. A fila de prontos nem sempre sera um FIFO, pode ser prioridades, lista encadeada, arvore, etc.

INSTITUTO FEDERAL
Santa Catarina

Critérios de escalonamento:

- **Utilização de CPU:** Manter a CPU tão ocupada quanto possível, variar de 40 a 90%.
- **Throughput:** Número de processos concluídos por unidade de tempo. (hora ou segundos).
- **Tempo de Turnaround:** Quanto tempo ele leva para ser executado.
- **Tempo de espera:** Tempo que o processo fica esperando na fila de prontos até ser executado pela CPU.
- **Tempo de resposta:** Tempo que leva da submissão até a primeira saida.

INSTITUTO FEDERAL
Santa Catarina

Algoritmos de escalonamento:

- **Sistemas Batch (lote):** sistemas em lote que não possuem interação com o usuário.
- **Sistemas Interativos:** são sistemas que tem a interação com o usuário, tempo compartilhado (time sharing).
- **Sistemas de Tempo Real:** tempo é um fator crítico.

Escalonamento em Sistemas Batch

1 - FIRST COME FIRST SERVED (FCFS)

- O primeiro processo a entrar é o primeiro a ser atendido (FIFO)
- Muito antigo, simples de Implementar.
- **Não Preemptivo**
- Problema: Tempo de processamento elevado. Processos curtos, podem demorar muito.

INSTITUTO FEDERAL
Santa Catarina

2 - Shortest Job First (SJF) :

- Job mais curto primeiro;
- Não se conhece o tempo necessário para execução.
- Limite de tempo do processo que é especificado pelo usuário quando submete o job, o usuário é motivado a estimar o job de maneira precisa. OU
- Estimar o próximo pico de CPU com base em picos anteriores. Media exponencial.

Escalonamento em Sistemas Interativos

- Time Sharing (tempo compartilhado).
- Usuários impacientes.
- Fazer um escalonamento de forma para agradar todos os usuários. Compartilhamento justo entre os diversos processos.
- Para isso temos vários algoritmos de escalonamento:

1. Round-Robin;
2. Prioridade;
3. Múltiplas Filas;
4. Shortes Process Next;
5. Garantido;
6. Lottery;
7. Fair-Share.

INSTITUTO FEDERAL
Santa Catarina

Round-Robin:

- Muito antigo, simples e mais utilizado;
- **Preemptivo** (não posso deixar um sistema utilizar todo o processador, tenho vários processos interativos, sendo executado por vários usuário), posso tirar do processador após o término do quantum; Time Out (tempo esgotado).
- Cada processo recebe uma fatia de tempo de execução do processador chamado quantum; Time Slice.
- Ao final desse tempo, o processo é suspenso e outro processo é colocado em execução;
- Também suspenso em caso de interrupção;
- Escalonador mantém uma fila de processos prontos.

INSTITUTO FEDERAL
Santa Catarina

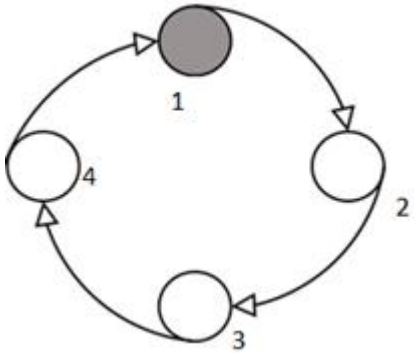
Round-Robin:

- Os processos são colocados em uma fila circular (de pontos) e executados um a um.
- Quando seu tempo acaba, o processo é suspenso e volta para o final da fila.
- Outro processo (primeiro da fila) é então colocado em execução.
- Quando um processo solicita E/S, vai para a fila de bloqueados e ao terminar a operação, volta para o final da fila de pontos.

INSTITUTO FEDERAL
Santa Catarina

Round-Robin:

- Pressupõe a igualdade de prioridades e tempo de execução.



INSTITUTO FEDERAL
Santa Catarina

Round-Robin:

- Problema:
- Tempo de chaveamento de processos (processo de troca). Desvantagem.

- Quantum:
- Se for muito pequeno, ocorrem muitas trocas diminuindo assim a eficiência da CPU. Se for muito longo, o tempo de resposta é comprometido.

INSTITUTO FEDERAL
Santa Catarina

Round-Robin:

- t = 4mseg (quantum)
- Context Switch = 1mseg (tempo de chaveamento), então 25% de tempo do CPU é perdido na troca de processos, menos eficiência.

- t = 100mseg
- Context Switch = 1mseg (tempo de chaveamento), então 1% de tempo do CPU é perdido.

- **Quantum razoável: 20/50 mseg.**

INSTITUTO FEDERAL
Santa Catarina

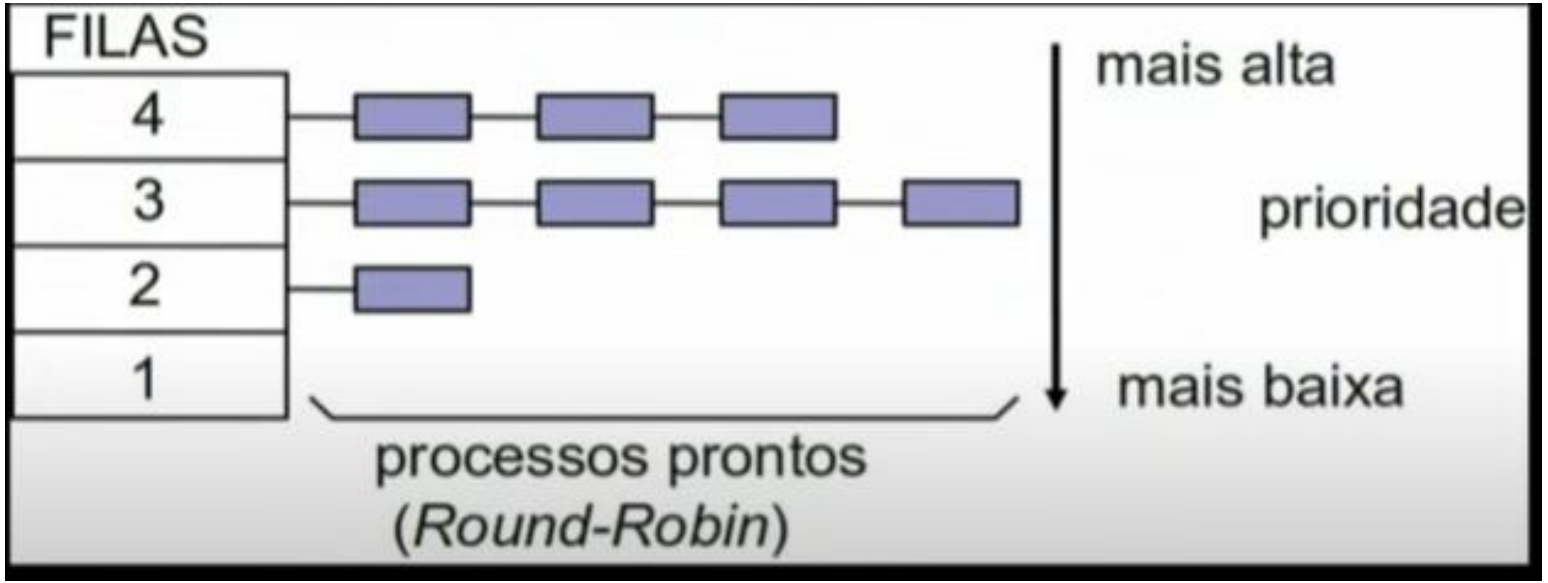
Algoritmos com Prioridades:

- Cada processo possui uma prioridade; Fila de prioridades.
- Quando todos os processos da fila de prioridades terminarem sua execução, entra os processos da fila convencional.
- Os processos prontos com maior prioridade são executados primeiro;
- Prioridades são atribuídas dinamicamente pelo sistema ou estatisticamente;

Curiosidade: Dizem que quando um IBM 7094 foi desligado no MIT em 1973, foi achado um processo de baixa prioridade que tinha sido submetido em 1967 e ainda não tinha sido executado. **(inanição)**

INSTITUTO FEDERAL
Santa Catarina

Algoritmos com Prioridades:



INSTITUTO FEDERAL
Santa Catarina

Algoritmos com Prioridades:

- Enquanto houver processos na classe maior: rode cada um de seus processos usando Round-Robin (quantum fixo de tempo).
- Se essa classe não tiver mais processos: passe a de menor prioridade.
- Não esqueça de ajustar as prioridades de alguma forma. Do contrário, ao menos prioritárias podem nunca rodar (inanição).
- Inanição (starvation) significa fome, fraqueza por falta de alimento ou seja, um processo pode nunca vir ser executado, vai “morrer da fome”. Solução: Envelhecimento aumento de 1 pt na prioridade a cada 15 min por exempo.

INSTITUTO FEDERAL
Santa Catarina

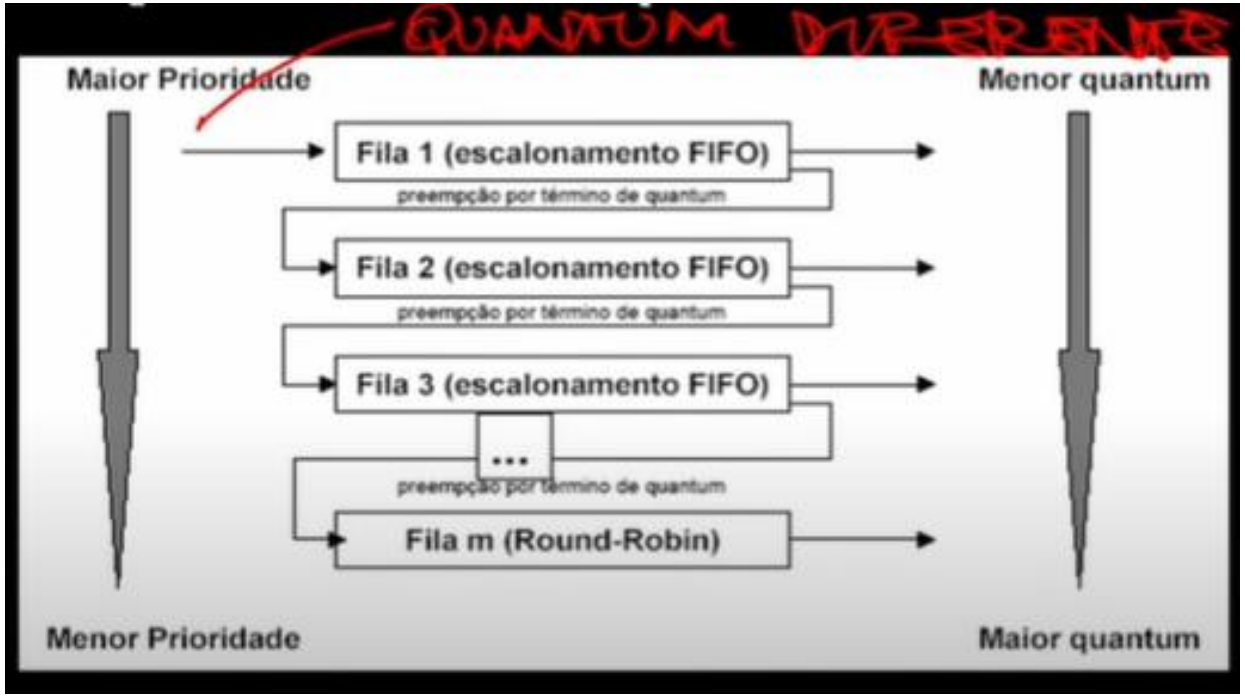
Múltiplas Filas:

- Cada vez que um processo é executado e suspenso, ele recebe mais tempo para execução;
- Inicialmente recebe 1 quantum e é suspenso;
- Então muda de classe e recebe 2, sendo suspenso;
- Reduz o número de trocas de processo;
- Os mais curtos terminam logo;
- Aos mais longos é dado mais tempo, progressivamente;
- Preemptivo.

INSTITUTO FEDERAL
Santa Catarina

Multiplas Filas:

- Linux e Windows



INSTITUTO FEDERAL
Santa Catarina

Multiplas Filas:

- Um processo precisa de 100 quanta (unidade, cada quanta pode situar de 20-50 mseg) para ser executado;
- Inicialmente, ele recebe um quantum para execução.
- Das próximas vezes ele recebe, respectivamente 2, 4, 8, 16, 32 e 64 quanta para execução.
- Quanto mais próximo de ser finalizado, menos frequente é o processo na CPU;
- Ele desce na fila de prioridade;
- Eficiência – os pequenos ainda tem vez.

INSTITUTO FEDERAL
Santa Catarina

Shortest Process Next :

- Processo mais curto próximo;
- Não se conhece o tempo necessário para execução.
- Como empregar esse algoritmo: estimativa de tempo.
- Com base em execuções antigas da mesma tarefa;
- Verificar o comportamento passado do processo e estimar o tempo.

INSTITUTO FEDERAL
Santa Catarina

Algoritmo Garantido:

- Aloco um tempo garantido por cada usuário;
- Garantir um tempo dedicado e compartilhado entre os n usuários que utilizam os processos.
- N usuários (ou processos em sistema monousuário): 1/n do tempo de CPU para cada usuário.
- Exemplo: se eu tenho 10 usuários (1/10), 10% do tempo alocado para cada usuário, desses processos.

INSTITUTO FEDERAL
Santa Catarina

Algoritmo da Loteria:

- Cada processo recebe “bilhetes” que lhe dão direito a recursos do sistema (inclusive processador);
- Fatia de processamento iguais por bilhete;
- Quando um escalonamento deve ser feito, escolhe-se aleatoriamente um bilhete;
- Processos mais importantes podem receber mais bilhetes;
- Processos podem doar bilhetes para colaboração com outros.

INSTITUTO FEDERAL
Santa Catarina

Algoritmo da Loteria:

- Aleatoriedade pode causar o problema de **inanição**.
- Precisa garantir que todos terão sua vez de rodar.
- Um modo é manter duas filas:
  - Bilhetes já sorteados.
  - Bilhetes ainda não sorteados. (Quando não tiver mais, ele volta para a fila dos sorteados).

- Quando a lista de não sorteados se esvazia, os bilhetes da lista de sorteados são transferidos a ela, reiniciando o processo.

INSTITUTO FEDERAL
Santa Catarina

Algoritmo Fair-Share:

- O dono do processo é levado em conta;
- Se um usuário A possui 9 processos e um usuário B apenas 2, o usuário A ganha 90% do uso da CPU, com round-robin (injusto);
- Compartilhamento justo: se um usuário foi prometida uma certa fatia de tempo, ele receberá, independentemente do número de processos (Ex: 50% para A e 50% para B).

INSTITUTO FEDERAL
Santa Catarina

Algoritmo Fair-Share:

- Usuário 1 -> A, B, C, D
- Usuário 2 -> E
- Foi prometido 50% da CPU a cada um e foi usado Round-Robin.
- A, E, B, E, C, E, D, E, A, E, ...

- Se 2/3 devem ir ao usuário 1:
- A, B, E, C, D, E, A, B, E, ...
- 66% para usuário 1 e 33% para usuário 2.

INSTITUTO FEDERAL
Santa Catarina

Escalonamento em Sistema Tempo Real

- Tempo é um fator crítico. Um erro pode levar a uma perda de uma vida humana ou pode causar um dano irreparável.
- Tipicamente um ou mais aparatos externos ao computador geram estímulos.
- O computador deve reagir apropriadamente dentro de um intervalo fixo de tempo.
- Sistemas Críticos:
  - Piloto automático de aviões.
  - Monitoramento de pacientes em hospitais;
  - Controle de automação em fábricas.

- **Hard Real Time:** atrasos não são tolerados; Aviões, usinas nucleares.
- **Soft Real Time:** atrasos ocasionais são tolerados; Bancos e multimídia (perda de uma parte da voz, pode não ser tão prejudicial).
- Eventos causam a execução de processos;
- Quando um evento externo (sensor) é detectado, cabe ao escalonador arrumar os processos de modo que todos os prazos sejam cumpridos.
- Eventos podem ser classificados como:
  - Periódicos: ocorrem em intervalor regulares de tempo.
  - Aperiódicos: ocorrem em intervalores irregulares de tempo.

Escalonamento em Sistema Tempo Real

Podem ser estáticos ou dinâmicos.

- **Estáticos:** decisões de escalonamento são realizados antes do sistema começar a rodar; Informação disponível previamente sobre tarefas e prazos.

- **Dinâmicos:** decisões de escalonamento são executados em tempo de execução.

Comunicação entre processos

- **Interprocess Communication (IPC).**

- Frequentemente processos precisam se comunicar.

  cat arq1 arq2 | grep alunos

- Um processo tem que se comunicar com o outro, para trabalhar colaborativamente. Alcançar um resultado objetivo.
- A comunicação entre processo é mais eficiente se for estruturada e não utilizar interrupções.

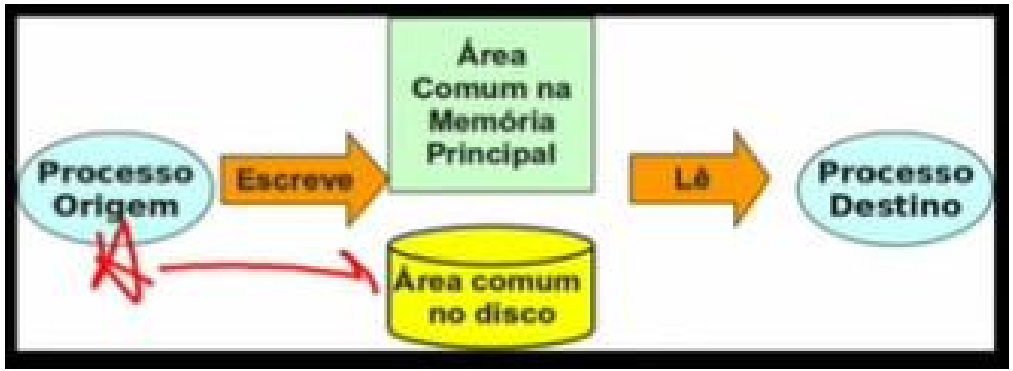
INSTITUTO FEDERAL
Santa Catarina

Comunicação entre processos

- Como um processo passa informação para outro processo?
- Como garantir que processos não invadam espaços uns dos outros, nem entre em conflito?
- Qualquer comunicação gera conflito. Ex: se um processo não está pronto para se comunicar, o outro processo tem que abortar até que esteja pronto.
- Fazer uma chamada de sistema para parar esse processo, colocar esse processo para dormir, para o próximo ser acordado.
- Qual a sequencia adequada quando existe dependência entre processos?

Condição de corrida

- Existe uma sequencia/sincronia adequada para que tudo isso possa acontecer.
- Os processos se comunicam através de alguma área de armazenamento comum.
- Essa área pode estar na memória principal (ex: variáveis em comum a 2 threads) ou pode ser um arquivo compartilhado.



- Condição de corrida: situação onde dois ou mais processos acessam recursos compartilhados concorrentemente (competitiva).
- Há uma corrida pelo recurso.
- Ex: quando estão lendo ou escrevendo algum dado compartilhado e o resultado depende de quem processa no momento propício.
- Dois procedimentos:
- A = d + c (‘a’ região crítica).
- X = a+ y (a variável ‘a’ é compartilhada/race condition).

- **Exemplo:**

- Quando um processo deseja imprimir um arquivo, ele coloca o nome do arquivo em uma lista (diretório) de impressão.
- Um processo chamada “printer daemon”, verifica a lista periodicamente para ver se existe algum arquivo para ser impresso e se existir, ele o imprime e remove seu nome da lista.
- Suponha que dois processos, A e B, irão enviar seus documentos para impressão.

INSTITUTO FEDERAL
Santa Catarina

- **Exemplo 2 vaga de avião.**

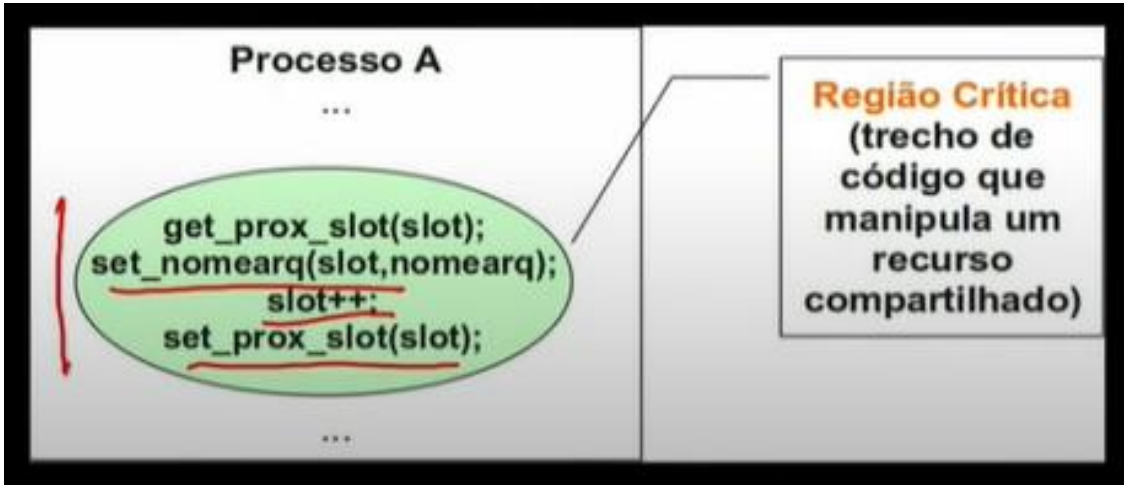
- **Problema de sincronização:**
- Operador OP1 (no Brasil) lê Poltrona1 vaga.
- Operador OP2 (no Japão) lê Poltrona1 vaga.
- Operador OP1 compra Poltrona1.
- Operador OP2 compra Poltrona1.
- Time Slice.

INSTITUTO FEDERAL
Santa Catarina

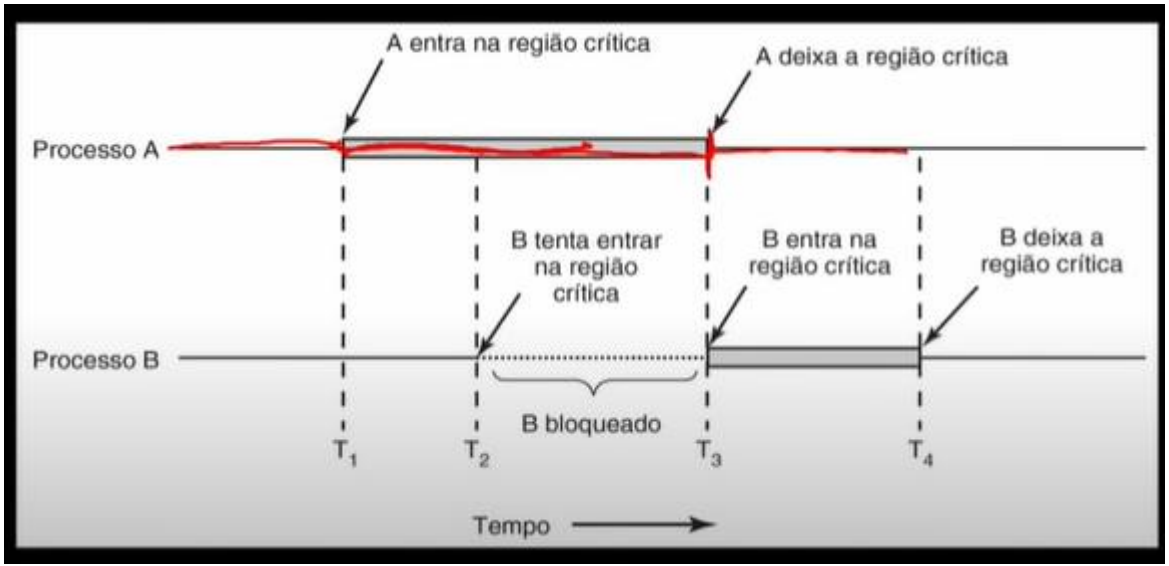
Exclusão Mútua

- Uma solução para as condições de corrida é impedir que mais de um processo leia e escreva em uma variável compartilhada ao mesmo tempo.
- Essa restrição é conhecida como exclusão mútua.
- Excluir mutuamente um processo quando entra em uma região crítica.
- Seções do programa onde são efetuados acessos a recursos partilhados por dois ou mais processos são denominadas regiões criticas (R.C).

- Região crítica do spool de impressão.
- Quando o processo entra na RC, ele só deveria mudar de contexto quando sair da RC.
- Quando o processo entra na região critica, ele somente será tirado quando terminar, não será interrompido.



- Exclusão mutua é então garantir que um processo não terá acesso a uma região crítica quando outro processo estiver utilizando essa região.
- Pode gerar uma fila de clientes para acessar a RC.
- Processo de exclusão mútua.



INSTITUTO FEDERAL
Santa Catarina

- Assegura-se a exclusão mútua recorrente aos mecanismos de sincronização fornecidos pelo SO.
- Presentes também na JVM (Máquina Virtual Java).
- Essas afirmações são válidas também para as threads (compartilham o mesmo espaço de endereçamento).
- Caso que é ainda mais critico, pois todas as thrads dentro do mesmo processo partilham os mesmo recursos.

Regras para programação concorrente (para uma boa solução)

- Exclusão mutua para ser uma solução ideal tem que atender as regras:
- 1 – Dois processos nunca podem estar simultaneamente dentro de suas regiões críticas. Um por vez.
- 2 – Não se pode fazer suposições em relação à velocidade e ao número de CPUs. Funcionar para todos os tipos.
- 3 – Um processo fora da região crítica não deve causar bloqueio a outro processo.
- 4 – Um processo não pode esperar eternamente para entrar em sua região crítica.

INSTITUTO FEDERAL
Santa Catarina

Soluções de exclusão mútua

- Espera Ocupada (Busy Waiting).
- Sleep/Wakeup (dormir/acordar).
- Semáforos.
- Monitores.

INSTITUTO FEDERAL
Santa Catarina

- Espera Ocupada (Busy Waiting).

Enquanto um processo está ocupado atualizando memória compartilhada em sua seção crítica, nenhum outro processo vai entrar na sua região crítica e causar problemas.

INSTITUTO FEDERAL
Santa Catarina

- Sleep/Wakeup.

Quando um processo invoca uma requisição de E/S, ele se auto-bloqueia para esperar o término da operação de E/S. Alguns outros processos devem “acordar” o processo bloqueado. Tal interação é um exemplo de um protocolo bloqueia/acorda (block/wakeup).

INSTITUTO FEDERAL
Santa Catarina

- # Semáforos

Um semáforo é uma variável protegida cujo valor somente pode ser acessado e alterado pelas operações P e V, e por uma operação de inicialização que chamaremos inicializa_semáforo.

INSTITUTO FEDERAL
Santa Catarina

Monitores

Um monitor é uma coleção de procedimentos, variáveis, e estruturas de dados que são todos agrupados em um tipo especial de módulo ou pacote. Monitores possuem uma importante propriedade que os torna uteis para atingir exclusão mútua: somente um processo pode estar ativo em um monitor em qualquer momento. Monitores são uma construção da própria linguagem de programação