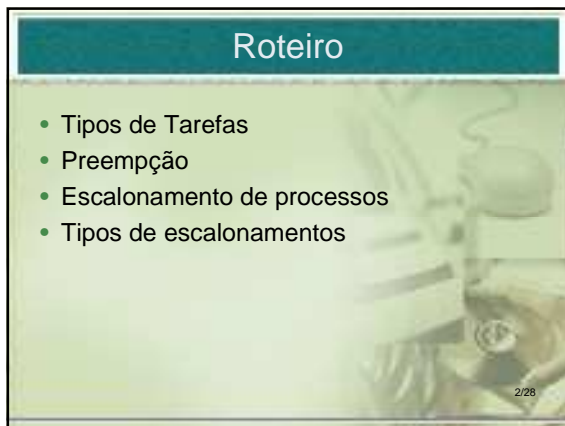




Sistemas Operacionais

Marcos Laureano

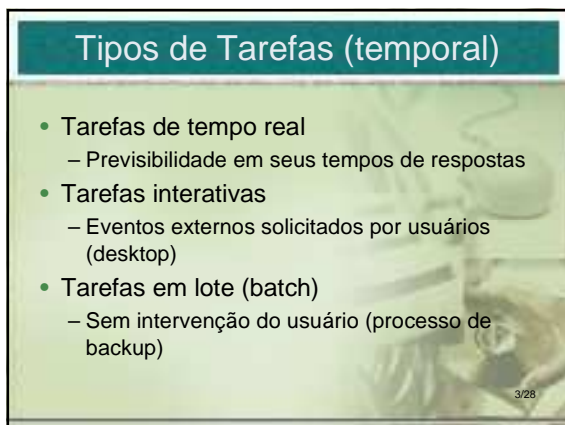
1/28



Roteiro

- Tipos de Tarefas
- Preempção
- Escalonamento de processos
- Tipos de escalonamentos

2/28



Tipos de Tarefas (temporal)

- Tarefas de tempo real
 - Previsibilidade em seus tempos de respostas
- Tarefas interativas
 - Eventos externos solicitados por usuários (desktop)
- Tarefas em lote (batch)
 - Sem intervenção do usuário (processo de backup)

3/28

Tipos de Tarefas (processador)

- Tarefas orientadas a processamento (CPU-bound tasks)
 - Uso intensivo do processador
- Tarefas orientadas a entrada/saída (IO-bound tasks)
 - Passam a maior parte do tempo esperando respostas dos dispositivos de entrada/saída

4/28

Preempção

- Se diz que um algoritmo/sistema operacional é preemptivo quando um processo entra na CPU e o mesmo pode ser retirado da CPU antes do término da execução do mesmo.

5/28

Processo

- Um programa em execução, o qual é constituído por uma seqüência de instruções, um conjunto de dados e um registro descritor.

6/28

Processo

- Se um processo passa a maior parte do tempo esperando por dispositivos de E/S, diz-se que o processo é limitado por E/S (I/O-bound).
- Se, ao contrário, o processo gasta a maior parte do seu tempo usando a CPU ele é dito limitado por computação (compute-bound ou CPU-bound ou ainda UCP-bound).
- Obviamente, processos I/O-bound devem ter prioridade sobre processos CPU-bound.

7/28

Escalonamento de processos

- Quando um ou mais processos estão prontos para serem executados, o sistema operacional deve decidir qual deles vai ser executado primeiro.
- A parte do sistema operacional responsável por essa decisão é chamada escalonador, e o algoritmo usado para tal é chamado de algoritmo de escalonamento.

8/28

Sistemas preemptivos

- Nestes sistemas uma tarefa pode perder o processador caso termine seu quantum de tempo, execute uma chamada de sistema ou caso ocorra uma interrupção que acorde uma tarefa mais prioritária (que estava suspensa aguardando um evento).

9/28

Sistemas não-preemptivos

- A tarefa em execução permanece no processador enquanto tiver condições de executar, só abandonando o mesmo caso termine de executar, solicite uma operação de entrada/saída ou libere explicitamente o processador, voltando à fila de tarefas prontas

10/28

Crítérios de Escalonamento

1. Justiça: fazer com que cada processo ganhe seu tempo justo de CPU;
2. Eficiência: manter a CPU ocupada 100% do tempo (se houver demanda);
3. Tempo de Reposta: minimizar o tempo de resposta para os usuários interativos;
4. Tempo de Turnaround: minimizar o tempo que usuários batch devem esperar pelo resultado;
5. Throughput: maximizar o número de jobs processados por unidade de tempo.

11/28

Problemas

- Uma complicação que os escalonadores devem levar em consideração é que cada processo é único e imprevisível.

12/28

Solução

- Para que um processo não execute tempo demais, praticamente todos os computadores possuem um mecanismo de relógio (clock) que causa uma interrupção periodicamente.

13/28

Escalonamento FCFS ou FIFO

- Processos são despachados de acordo com sua ordem de chegada na fila de processos prontos do sistema.
- Uma vez que um processo ganhe a CPU, ele roda até terminar.
- FIFO é uma disciplina não preemptiva.

14/28

Escalonamento Round Robin (RR)

- Cada processo recebe um intervalo de tempo, chamado quantum, durante o qual ele pode executar.
- Se o processo ainda estiver executando ao final do quantum, a CPU é dada a outro processo.
- Se um processo bloqueou ou terminou antes do final do quantum, a troca de CPU para outro processo é obviamente feita assim que o processo bloqueia ou termina.
- É preemptivo.

15/28

Problema das trocas de processos

- Mudar de um processo para outro requer um certo tempo para a administração — salvar e carregar registradores e mapas de memória, atualizar tabelas e listas do SO, etc.
- Isto chama-se troca de contexto.
- Suponha que esta troca dure 5 ms.
- Suponha também que o quantum está ajustado em 20 ms.
- Com esses parâmetros, após fazer 20 ms de trabalho útil, a CPU terá que gastar 5 ms com troca de contexto. Assim, 20% do tempo de CPU é gasto com o overhead administrativo

16/28

Solução ?

- Para melhorar a eficiência da CPU, poderíamos ajustar o quantum para 500 ms.
- Agora o tempo gasto com troca de contexto é menos do que 1 %.
- Considere o que aconteceria se dez usuários apertassem a tecla <ENTER> exatamente ao mesmo tempo, disparando cada um processo.
- Dez processos serão colocados na lista de processo aptos a executar.
- Se a CPU estiver ociosa, o primeiro começará imediatamente, o segundo não começará antes de ½ segundo depois, e assim por diante.
- O azarado do último processo somente começará a executar 5 segundos depois do usuário ter apertado <ENTER>, isto se todos os outros processos tiverem utilizado todo o seu quantum.
- Muitos usuários vão achar que o tempo de resposta de 5 segundos para um comando simples é "muita" coisa.

17/28

Conclusão

- Ajustar um quantum muito pequeno causa muitas trocas de contexto e diminui a eficiência da CPU, mas ajustá-lo para um valor muito alto causa um tempo de resposta inaceitável para pequenas tarefas interativas.
- Um quantum em torno de 100 ms frequentemente é um valor razoável.

18/28

Escalonamento com prioridades

- Cada processo possui uma prioridade associada, e o processo pronto para executar com a maior prioridade é quem ganha o processador.
- Para evitar que processos com alta prioridade executem indefinidamente, o escalonador pode decrementar a prioridade do processo atualmente executando a cada tick de relógio.
- Se esta ação fizer com que a prioridade do processo se torne menor do que a prioridade do processo que possuía a segunda mais alta prioridade, então uma troca de processos ocorre.

19/28

Problemas com prioridades

- Se as prioridades não forem ajustadas de tempos em tempos, os processos nas classes de prioridades mais baixas podem sofrer o fenômeno que chamamos *starvation* (o processo nunca recebe o processador, pois sua vez nunca chega).

20/28

Selfish Round-Robin

- Alternância circular egoísta.
- O processo fica em uma fila até atingir a maturidade suficiente, depois entra na fila Round-Robin tradicional.
- Processos mais antigos são favorecidos.

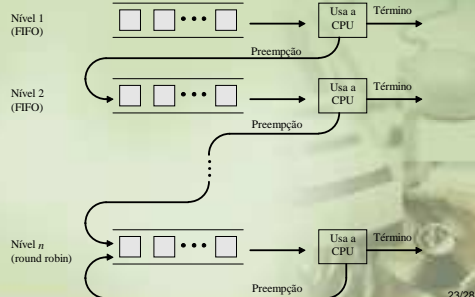
21/28

Multilevel Feedback Queues

- Favorecer pequenos jobs;
 - Favorecer jobs limitados por E/S para atingir uma boa utilização dos dispositivos de E/S; e
 - Determinar a natureza de um job tão rápido quanto possível e escalonar o job de acordo.
- Multilevel feedback queues (filas multi-nível com retorno) fornecem uma estrutura que atinge esses objetivos.

22/28

Multilevel Feedback Queues



23/28

Escalonamento com Prazos

- No escalonamento com prazos certos jobs são escalonados para serem completados até uma certa data ou hora, ou um prazo.
- Esses jobs podem ter alta importância se entregues dentro do tempo, ou podem não ter utilidade alguma se terminarem de ser processados além do tempo previsto no prazo.

24/28

Escalonamento Shortest-Job-First (SJF)

- Shortest-job-first (o menor job primeiro) é um algoritmo não preemptivo no qual o job na fila de espera com o menor tempo total estimado de processamento é executado em seguida.
- SJF reduz o tempo médio de espera sobre o algoritmo FIFO.
- Entretanto, os tempos de espera tem uma variância muito grande (são mais imprevisíveis) do que no algoritmo FIFO, especialmente para grandes jobs.
- SJF favorece jobs pequenos em prejuízo dos jobs maiores.
- Também pode ser encontrado como SPF (shortest-process-first)

25/28

Shortest-Remaining-Time (SRT)

- Menor tempo de execução restante.
- Alternativa preemptiva ao SJF ou SPF.
- Tenta aumentar o rendimento atendendo pequenos processos que chegam.

26/28

Escalonamento por fração justa

- Fair Share Scheduling – FSS.
- Utilizado para grupos de processos
 - Linux
 - Processos de usuários
- Os processos do conjunto são manipulados por outros algoritmos de escalonamento.

27/28