Escalonamento em Tempo Real

Marcelo Henrique Alacantara, Mauricio Souza Menezes, Ricardo Azevedo, Verena Azevedo

Curso de Bacharelado em Sistemas de Informação – Universidade do estado da Bahia (UNEB) – Campus de Castanhal
413200-000 – Salvador – BA– Brasil

{marcelo.henrik, mauriciosm95,rick.azevedoo7, verena.ral, emailautor4}@gmail.com

Resumo. Este artigo visa explicar parte do processo de escalonamento, será abordado o escalonamento em Sistemas operacionais em tempo real.

1. Introdução

Os sistemas computacionais são utilizados para as mais diversas tarefas executando assim processos com diversas finalidades e graus de prioridade a depender da finalidade da máquina. Um dos aspectos de maior importância, com relação a processos, é o escalonamento. Escalonar é "selecionar qual o próximo processo a obter uso da CPU" (Stuart, p. 93). Segundo Tanenbaum (p.88) o escalonador é a parte do sistema operacional responsável por escolher, segundo um algoritmo de escalonamento, qual processo na fila de prontos poderá utilizar a CPU". Qual o melhor algoritmo de escalonamento? Tanembaum (p.90) responde "para ambientes diferentes, são necessários diferentes algoritmos de escalonamento". Neste artigo discorreremos sobre o algoritmo de escalonamento de tempo real, também conhecido como RTOS (Real-Time Operation System) Sistema Operacional de Tempo Real.

2. A necessidade de um sistema de tempo real

Os sistemas operacionais de tempo real geralmente são feitos para sistemas de comunicações, aeroespaciais e de defesa, médicos e onde é necessário uma resposta confiável e rápida. Considere o sistema computacional que controla a temperatura de uma usina nuclear, refrigerando-o ou aquecendo-o quando necessário. Considere o sistema de controle de centrífugas de enriquecimento de urânio ou um sistema de radar aeroespacial, que recebe informações de posicionamento das aeronaves, para que possíveis colisões sejam detectadas e evitadas. Se o sistema não tratar estas entradas dentro de suas restrições de tempo, poderá causar uma tragédia. Todos esses sistemas possuem uma necessidade em comum. Há a necessidade de serem servidos em um limite determinado de tempo. Caso qualquer um dos sistemas supracitados exceda o tempo limite para cumprir sua função danos catastróficos podem ser causados.

3. A prioridade dos processos de Tempo Real

A prioridade de tempo real pode ser modificada pelo usuário através de *chamadas de sistema*. Da mesma maneira que os processos convencionais, processos na política Round Robin têm sua fatia de tempo calculada baseada na prioridade estática. Os sistemas operacionais geralmente possuem uma biblioteca que os programas normais chamam com o fim de comunicar com o núcleo. Normalmente esta biblioteca é escrita na linguagem C (libc), como a glibc e MS LibC. Esta biblioteca manipula os detalhes de baixo nível relacionados com a passagem de informação para o núcleo e com a chamada da rotina Os sistemas operacionais definem chamadas de sistema para todas as operações envolvendo o acesso a recursos de baixo nível (periféricos, arquivos, alocação de memória, etc) ou abstrações lógicas (criação e finalização de tarefas, por

exemplo).

Geralmente as chamadas de sistema são oferecidas para as aplicações em modo usuário através de uma biblioteca do sistema (system library), que prepara os parâmetros, invoca a interrupção de software e retorna à aplicação os resultados obtidos.

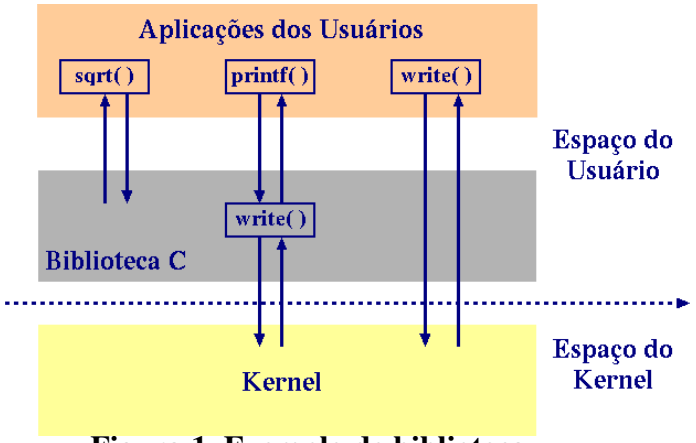

Figura 1. Exemplo de biblioteca

3.1. Exemplos de system call

Nos sistemas *POSIX* e similares, as chamadas de sistema mais usadas são close, execve, fork, wait, kill, open, read, write, clone, nice, exit, send, receive, signal e ioctl. Os sistemas operacionais atuais têm centenas de chamadas de sistema. Por exemplo, o Linux tem quase 400 chamadas de sistema diferentes.

4. Escalonamento em Tempo Real

Com a necessidade de existência de sistemas de tempo real em paralelo a necessidade de seu funcionamento, está a necessidade de políticas de escalonamento, deste tipo de necessidade nascem os algoritmos de escalonamento de tempo real, que tem por objetivo atender aos processos dentro dos requisitos de limite de tempo.

O escalonador é componente básico para garantir o bom funcionamento em sistemas computacionais pois ele é responsável pela garantia na realização da tarefa, em sistemas de tempo real a importância do escalonador é evidente e fica ainda mais claro quando observamos que está diretamente ligado a correção do sistema tendo em vista que a tarefa do escalonador é cumprir restrições temporais.

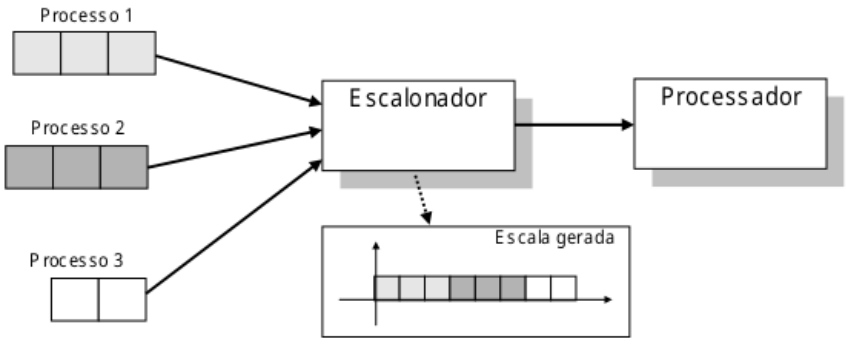

Figura 2. Ilustração escalonador

Cada tarefa realizada pode ser considerado parâmetro importante no escalonamento e são utilizados pelo escalonador, Os atributos mais frequentes encontrados são:

- Tempo de Lançamento: instante de tempo no qual a instância de tarefa se torna disponível para execução. Se as condições de dependência de dados e controle estiverem satisfeitas uma tarefa pode executar a partir do seu tempo de lançamento.

- Tempo Máximo de Execução: Tempo máximo necessário para execução de uma instância de tarefa.

- Deadline ou Deadline Absoluto: Instante de tempo no qual a execução de uma instância de uma tarefa deve estar completa. Diz-se que as instâncias de tarefa não têm deadline se seu valor é infinito.

- Deadline Relativo: Tempo máximo de resposta disponível a partir do tempo de lançamento. A soma do tempo de lançamento com o deadline relativo é igual ao deadline absoluto.

- Período: Periodicidade com que uma nova instância de tarefa será lançada.

Podemos chamar uma tarefa *periódica* quando cada uma de suas instâncias é lançada em intervalos de tempo regulares. Uma tarefa pode ser chamada de esporádica quando conhece apenas o intervalo mínimo de tempo entre lançamentos e instâncias consecutivas. Tarefas podem ser diferentes de *periódicas* e *esporádicas*, podem ser *aperiódicas*, neste caso não se conhece nada sobre a frequência de ativação de suas instâncias. O atributo período é geralmente usado para especificar tarefas *periódicas* ou *esporádicas*. O atributo *deadline* está associado à criticidade das tarefas. De forma geral, há tarefas criticas e não-críticas. O sistema operacional, através de seu escalonador, deve garantir que nenhum dos deadlines das instâncias das tarefas críticas seja perdido. Exemplos de tais tarefas podem ser encontrados em controle industrial, de avião ou automotivo. Perder o deadline da verificação de altitude de um avião, por exemplo, pode provocar catástrofes caso o avião esteja aterrissando. Para tarefas não-críticas, por outro lado, o não cumprimento ocasional de deadlines é permitido, apesar de não ser desejável. Por exemplo, num sistema multimídia, deixar de exibir algum quadro de imagem vez por outra é tolerável.

4.1. Escalonamento de Tarefas Periódicas

Algoritmos de escalonamento para sistemas de tempo real podem basear-se em diversas abordagens conforme lidem com a escolha das prioridades. Este trabalho foca na abordagem orientada a prioridade, na qual o escalonamento é realizado em tempo de execução. Decisões de escalonamento são tomadas quando ocorrem eventos como lançamentos e encerramentos de instâncias de tarefas e estas possuem uma prioridade associada. As tarefas são colocadas numa fila de prontas, ordenada pela prioridade. No momento de selecionar uma tarefa, o escalonador atualiza a fila de prontas e então escalona a instância da inficionada fila. Diferentes algoritmos definem as prioridades segundo diferentes critérios.

Os algoritmos para escalonamento de tarefas periódicas baseados na abordagem orientada a prioridade podem ser classificados em dois tipos: prioridade fixa e prioridade variável. Um algoritmo de prioridade fixa define prioridades em tempo de projeto, as prioridades das instâncias de uma mesma tarefa são iguais, assim, a prioridade de cada tarefa é fixa em relação as outras tarefas. Já um algoritmo de prioridade variável atribui prioridades em tempo de execução, instâncias de uma mesma tarefa podem receber prioridades diferentes, o que faz com que uma tarefa tenha sua prioridade modificada durante a execução do sistema.

4.2. Escalonamento de Tarefas Aperiódicas ou Esporádicas

Os algoritmos vistos anteriormente contemplam apenas o escalonamento de tarefas periódicas. Quando tarefas periódicas ou esporádicas coexistem com tarefas aperiódicas é necessário prover uma maneira para que se possa escalonar as últimas sem comprometimento das primeiras. Para tarefas aperiódicas não é possível fazer nenhuma previsão quanto intervalos de lançamento de suas instâncias ou sobre o tempo necessário para sua execução. Já no caso de tarefas periódicas e esporádicas, conhece-se alguma informação sobre o intervalo de tempo entre lançamentos de instâncias consecutivas. Cabe aos algoritmos de escalonamento que consideram também as tarefas aperiódicas e esporádicas determinar em que momento se deve executá-las. Este trabalho foca nos algoritmos que suportam escalonamento de tarefas aperiódicas. Nestes algoritmos o objetivo é escalonar as tarefas aperiódicas assim que possível sem causar perdas de deadline das tarefas periódicas e das tarefas esporádicas aceitas no sistema.

5. Algorítmos de escalonamento

Um bom algoritmo de escalonamento proporciona às tarefas *aperiódicas* o menor tempo de reposta possível e escalona tarefas *esporádicas* sem prejuízo as tarefas periódicas e as tarefas esporádicas aceitas. Há diversas técnicas para se tratar o escalonamento de tarefas aperiódicas.

Existem propiedades relacionadas ao tempo e devem consideradas em tarefas de tempo real, são elas:

- Release Time (Tempo de liberação): Tempo em que o processo está pronto para ser processado;
- Deadline (Tempo limite): Tempo máximo permitido para que um processo complete sua execução;

- Worst case execution time (Tempo de Execução no Pior Caso): Tempo máximo, estabelecido em tempo de projeto, para a completa execução de um processo;

- Período: É o intervalo em que um determinado processo deverá se repetir;

- Prioridade: Urgência relativa de um processo para ser executado.

5.1. Rate Monotonic (RM)

Escalonamento de taxa monotônica foi desenvolvido por Liu & Layland, onde a ideia principal é dar maior prioridade às tarefas de menor período, ele produz escalas em tempo de execução através de escalonadores preemptivos dirigidos a prioridades é um esquema de prioridade fixa, ou seja, tarefas sempre possuem a mesma prioridade. O algoritmo RM funciona sobre um modelo de tarefas simples que obedece às principais premissas:

- As tarefas são periódicas e independentes;

- O deadline de cada tarefa coincide com seu período;

- O tempo de computação (C) de cada tarefa é conhecido e constante (Worst computation time);

- O tempo de chaveamento entre as tarefas é considerado nulo.

5.2. Earliest Deadline First (EDF)

Escalonamento que foi desenvolvido por Liu & Layland, o EDF produz escalas em tempo de execução através de escalonadores preemptivos dirigidos a prioridades, onde a principal ideia é atribuição dinâmica *de* prioridades de acordo com os deadlines de cada tarefa, ele é um esquema de prioridade dinâmica onde trabalha sobre um modelo de tarefas simples que obedece às premissas:

- Deadline de cada tarefa coincide com seu período;

- As tarefas são periódicas e independentes;

- O tempo de computação de cada tarefa é conhecido e constante(Worst computation time);

- O tempo de chaveamento entre as tarefas é considerado nulo.

5.3. Deadline Monotonic (DM)

Desenvolvido por Leung & Whitehead este modelo estende o modelo de escalonamento de taxa monotônico, onde suas prioridades são sempre fixas ou estáticas. Suas premissas são:

- As tarefas são periódicas e independentes.
- O “deadline” de cada tarefa pode não coincidir com o seu período;

- O tempo de computação de cada tarefa é conhecido e constante (Worst Case Computation Time);
- O tempo de chaveamento entre tarefas é assumido nulo.

6. Referências

DAVE, Marshall. **Programming in C UNIX System Calls and Subroutines using C..** Disponível em: <http://users.cs.cf.ac.uk/Dave.Marshall/C/>. Acesso em: 10 dez. 2017.

LINXOS, LinxOS. **Real-Time Operating System**. Disponível em: <http://www.lynx.com/products/real-time-operating-systems/>. Acesso em: 10 dez. 2017.

WIKIPÉDIA, Wikipédia. **Chamada de sistema**. Disponível em: <http://www.lynx.com/products/real-time-operating-systems/>. Acesso em: 10 dez. 2017.

LINUX, Linux. **Linux Systemcall reference**. Disponível em: <https://syscalls.kernelgrok.com/>. Acesso em: 10 dez. 2017.

UNIRIOTEC, Uniriotec. **Chamadas de sistemas bibliotecas c**. Disponível em: <http://www.uniriotec.br/~morganna/guia/chamadas.html>. Acesso em: 10 dez. 2017.

PUHLMANN, Henrique Puhlmann. **Sistemas Operacionais de Tempo Real - Introdução**. Disponível em: <https://www.embarcados.com.br/sistemas-operacionais-de-tempo-real-rtos/>. Acesso em: 10 dez. 2017.

PRADO, Sergio . **Sistemas de Tempo Real** . Disponível em: <https://sergioprado.org/sistemas-de-tempo-real-parte-2-3/>. Acesso em: 10 jul. 2017.

MORAES, Caio. **Implementando elementos de RTOS no Arduino** . Disponível em: <https://www.embarcados.com.br/elementos-de-rtos-no-arduino/>. Acesso em: 10 jul. 2017.

MENESES SILVA, Andre Luis. **Algoritmo para escalonamento Tempo Real**. Disponível em: <http://slideplayer.com.br/slide/398550/>. Acesso em: 10 jul. 2017.

Y, RTOS. **Sn**. Disponível em: <https://rtos.com/solutions/>. Acesso em: 10 jul. 2017.

Storch, M. F. (1997). A framework for the simulation of complex real-time systems.

Tanenbaum, A. S. (1999). Sistemas Operacionais Modernos. Prentice-Hall, Inc