Universidade Estadual Paulista

“Julio de Mesquita Filho”

Sistemas Operacionais de Tempo-Real (RTOS)

Breno Luiz B. Auricchio, Daniel Cálice Martin e Danilo Rodrigues Pereira

Rio Claro
2009

Sumário

Sistemas Operacionais de Tempo-Real – Características

Resumo. Este artigo descreve características de sistemas operacionais construídos para lidar com aplicações de tempo real. Apresenta os conceitos utilizados na alocação de memória, escalonamento de processos e outras partes do sistema. Mostra também alguns casos de uso e as implementações mais conhecidas deste tipo de sistema operacional bem como suas particularidades.

Palavras - chave: Sistema Operacional, Tempo Real

1. Introdução

Podemos considerar sistema operacional como um conjunto de rotinas a serem executadas pelo processador, assim como programas feitos por usuários, porém são executadas de forma não linear como na maioria das outras aplicações, mas sim de acordo a necessidade que possa ocorrer, tendo como principal função o gerenciamento do processador, memória e dispositivos de entrada e saída, bem como a interação entre máquina e usuário, tornando a utilização mais simples, rápida e segura.

Portanto, conforme estudos a ser realizado a seguir, podem dizer que os sistemas operacionais de tempo real, diferenciam-se pelo fato do tempo de resposta exigido ser rigoroso no processamento das informações, pois caso não tenham seus critérios executados no tempo estipulado, pode sofrer conseqüências irreparáveis

2. Sistemas de Tempo-Real

Atualmente nos deparamos com uma quantidade crescente de aplicações, de importância na sociedade, que apresentam comportamentos definidos segundo restrições temporais. Alguns exemplos dessas aplicações se encontram no controle de plantas industriais, de tráfego aéreo ou ferroviário, de aquisição de dados, nas telecomunicações, na eletrônica embarcada em carros e aviões, na robótica, equipamentos médicos, em sistemas de multimídia, etc. Aplicações que apresentam tal característica, de estarem sujeitas a restrições temporais, são comumente identificadas como Sistemas de Tempo Real.

As necessidades de segurança num número cada vez maior de aplicações com restrições temporais colocam à prova as metodologias e ferramentas convencionais, sob pena de perdas em termos financeiros, ambiental ou humano. Neste sentido surge toda uma demanda por algoritmos de suporte computacional e por metodologias para o desenvolvimento de tais aplicações.

3. Sistemas e Aplicações de Tempo Real

Um sistema computacional de tempo real é um sistema cuja correção depende não somente do resultado lógico dos processamentos, mas também do instante em que os resultados são produzidos. Os sistemas de tempo real, portanto, devem ter a habilidade de executar um comando ou instrução e disponibilizarem a resposta em um tempo relativamente previsível, ou seja, que seja possível estimar o tempo de processamento necessário.

4. Requisitos das Aplicações de Tempo Real

As aplicações de tempo real diferenciam-se em grau de criticidade. Algumas possuem restrições temporais rígidas (*hard real-time*); outras possuem restrições temporais que não são tão rígidas (*soft real-time*). Embora haja outras classificações, neste trabalho serão abordadas apenas as duas classes a seguir:

4.1 HARD Real-Time

Hard real-time significa que o sistema, isto é, aplicação, sistema operacional, *hardware*, devem ser projetados, implementados e testados de modo a garantir que as respostas ocorram dentro de um

intervalo de tempo bem definido, para todas as situações em que o sistema possa se encontrar. Uma falha no atendimento dos requisitos temporais poderá causar grandes prejuízos ou, em casos extremos, até perda de vidas. Exemplos de aplicações de sistemas *hard real-time* podem ser encontrados em controles industriais, militares, equipamentos médicos, etc.

4.2 SOFT Real-Time

Soft real-time representa os sistemas que possuem respostas para uma instrução, comando, etc em um tempo não necessariamente exato, mas dentro de um tempo médio de resposta. Nestes sistemas o tempo de resposta é importante, mas os prejuízos não são tão graves se ocasionalmente um prazo for perdido.

5. Características

Segundo Monfret e Linimon (2000), algumas características são desejáveis para que um sistema seja considerado de tempo real. Dentre elas:

- *Multi-threading*: O sistema deve possuir suporte a execução de várias *threads* simultaneamente e cada thread deve ter sua prioridade;
- Preempção: o escalonador deve interromper um processo escalonado (em execução) por outro que tenha prioridade maior no momento em que altera seu *status* para apto;
- Um número suficiente de níveis de interrupção, com suporte a agrupamento de interrupções;
- O tempo da troca de contexto entre processos deve ser pequeno.
- A latência de interrupção (tempo entre a interrupção e a execução de tarefa) deve ser compatível com os requerimentos da aplicação e deve ser previsível. Este valor depende do número de interrupções pendentes simultâneas;
- O tempo que cada chamada de sistema utiliza ao ser executada deve ser conhecido. Deve ser previsível e não depender do número de objetos no sistema;
- Deve vir acompanhado de uma boa documentação, ferramentas de desenvolvimento e teste de aplicações;
- Suportar mais de um dispositivo.

6. Escalonador de Processos

A maioria das tarefas neste tipo de sistema está bloqueada a maior parte do tempo, e a lista de processos aptos é geralmente pequena, contendo dois ou três na maior parte do tempo. Somente uma tarefa por *CPU* é executada (WIKIPEDIA 2005).

É importante que a manutenção na lista de aptos não interrompa as tarefas do sistema por muito tempo. Por exemplo, incluir um processo na lista pode ser feito com eficiência ao mesmo tempo em que uma busca é efetuada nesta mesma lista se a interrupção de inserção durar pouco tempo.

Como a lista contém poucos processos, a busca por processos é feita seqüencialmente visto que o tempo para iniciar a busca (*set-up time*) é praticamente nulo (WIKIPEDIA 2005).

A lista geralmente é ordenada pela prioridade de cada processo.

Fugindo do comum, existem algoritmos de escalonadores capazes de trabalhar com prioridades dinâmicas. Isso é útil principalmente em sistemas que se adaptam a situações externas e as tarefas que necessitam receber maior ou menor atenção em determinados momentos. Por exemplo, um robô ao subir escadas deve ter maior prioridade em processos que mantém seu equilíbrio. Já ao trafegar pelo salão deveria estar atento para não colidir com objetos do ambiente, cumprimentar pessoas, etc.

Outros algoritmos focam em tomada de decisões quanto à escolha de processos em sistemas muito carregados e que compartilham um só processador. É o caso do algoritmo DASA que além de escalonar processos que possuem dependências faz gerência de recursos e se comporta muito bem em casos onde é impossível completar todas as tarefas do sistema (*overload*) (CLARK 1990).

7. Alocação de Memória

Alocação de memória é uma das partes mais críticas em um sistema de tempo real. Dois pontos principais devem ser observados atentamente: a velocidade da alocação e a fragmentação do espaço disponível (WIKIPEDIA 2005).

O tempo gasto na alocação deve ser constante. Não pode em hipótese alguma depender do estado do sistema em determinado momento, (quantos processos em execução, por exemplo) ou de quanto

tempo o hardware está em operação. Isso é necessário porque é muito comum um sistema de tempo real não ser reiniciado durante anos ou até mesmo durante toda sua vida útil.

A lista ligada de blocos livres de memória, esquema utilizado na maioria dos sistemas operacionais, não pode ser utilizada em sistemas de tempo real. Neste esquema de alocação a lista não tem um tamanho fixo, portanto não é possível determinar um tempo constante de alocação para uma requisição e, além disso, o sistema teria que percorrê-la cada vez que algum processo solicitasse memória até encontrar um fragmento com a quantidade suficiente de memória (WIKIPEDIA 2005).

Outro detalhe é que quanto mais fragmentada a memória, maior a lista e maior o tempo de procura. Não é aceitável, por exemplo, que após um mês ou um ano de execução o sistema de controle de tráfego aéreo tenha que ser reiniciado e milhares de aeronaves no piloto automático fiquem desgovernadas no espaço até que o sistema volte a operar.

Para resolver este problema é comum utilizar alocação de blocos fixos, ou seja, tem-se uma lista ligada com tamanho pré-definido onde cada nó representa um bloco de memória de tamanho pré-definido. Uma alocação demora no máximo o tempo de percorrer a lista de blocos. Apesar de ser um esquema bastante limitado, é bastante utilizado em sistemas embarcados simples (*embedded systems*). Outra forma utilizada é o esquema de alocação denominado *Buddy block* (“Bloco Companheiro”) (WIKIPEDIA 2005).

Escolher o melhor algoritmo pode ser também uma questão de como o dispositivo em que o sistema operacional vai residir é operado. Em um celular por exemplo, que pode ficar ligado por muitos dias seguidos não é aconselhável que ocorra fragmentação de memória ao preço de o usuário ser interrompido no momento de uma ligação que valeria milhões para sua empresa. Já em outro dispositivo que fique ligado por um tempo pequeno, que seja reiniciado freqüentemente ou que tenha processos que não necessitem de muita memória, a alocação não é um problema tão sério. Um relógio com cronômetros de alta precisão só que em quantidade fixa de marcadores é um bom exemplo.

8. RT-Linux (Real-Time Linux)

O RT-Linux é uma extensão do Linux proposto para dar suporte a tarefas com restrições temporais críticas. O escalonador do RT-Linux trata o sistema operacional Linux como uma tarefa comum. O *kernel* do Linux é executado apenas quando não existe nenhuma tarefa de tempo real para ser executada e o *microkernel* de tempo real está inativo. O Linux nunca pode bloquear interrupções ou proteger-se de requisições de interrupções emitidas pelo escalonador do RT-Linux.

Todas as interrupções são inicialmente tratadas pelo *microkernel* de tempo real, e são passadas para a tarefa Linux somente quando não existem tarefas de tempo real para serem executadas. Para minimizar mudanças no kernel convencional, o *hardware* que controla interrupções é emulado utilizando-se macros. Assim, quando o *kernel* convencional desabilita interrupções, este evento é registrado pelo *software* que emula o controlador de interrupções, mas estas não são realmente desabilitadas. Desta forma, o *kernel* convencional não possui controle direto sobre as interrupções, sendo todas tratadas pelo *microkernel* de tempo real.

RT-Linux foi projetado para que nunca espere a liberação de recursos alocados pelo Linux. O *microkernel* de tempo real não requisita *spinlocks* compartilhados, ou estruturas de dados sincronizadas, exceto em situações muito controladas. Por exemplo, tarefas de tempo real e tarefas convencionais podem comunicar-se através de filas sem bloqueio e memória compartilhada. As filas, chamadas de RT-FIFO, são *buffers* utilizados para a troca de mensagens, projetadas de tal forma que tarefas de tempo real nunca são bloqueadas.

Os serviços oferecidos pelo *microkernel* de tempo real são mínimos: tarefas com escalonamento baseado em prioridades fixas e alocação estática de memória. A comunicação entre tarefas de tempo real utiliza memória compartilhada e a sincronização pode ser realizada desabilitando as interrupções de *hardware*. Existem módulos de *kernel* opcionais que implementam outros serviços, tais como um escalonador EDF e implementação de semáforos.

O mecanismo de módulos de kernel do Linux permite que novos serviços sejam disponibilizados para as tarefas de tempo real. Entretanto, quanto mais complexos estes serviços mais difícil será prever o comportamento das tarefas de tempo real.

Atualmente o RT-Linux possui licença não *free*. Desta maneira desestimulando sua aplicação para o desenvolvimento de sistemas de tempo real, baseado em ferramentas de *software* livre, como é o caso neste projeto.

9. QNX

É um dos melhores, senão o melhor, sistema operacional de tempo real disponível. É construído com a filosofia *microkernel*, com foco em computação embutida, distribuído comercialmente pela QNX Software System.

Sua funcionalidade é baseada em serviços e isso o torna flexível a ponto de o usuário poder escolher que parte do sistema deve rodar. É tão pequeno que pode ser instalado até em um diskete de 1.44 polegadas. É considerado um sistema muito rápido e bastante completo.

Pode ser utilizado na maioria das CPU's modernas.

Existe uma versão gratuita para download no site da companhia para uso em aplicações não comerciais.

10. FreeRTOS

FreeRTOS é um mini-*kernel* de sistema operacional de tempo real desenvolvido com foco em microcontroladores. Seu código fonte é livre e pode ser usado em aplicações comerciais. Possui implementação para várias arquiteturas de micro-controladores, dentre elas ARM7, PIC, ATMega, HCS12 (Motorola).

Seu código fonte consiste em apenas três arquivos que disponibilizam toda a estrutura base para executar tarefas. Neles estão presentes as funcionalidades de *Multitasking*, escalonamento de processos, mudança de contexto, etc. Cada implementação possui um ou mais arquivos que fazem uso destes três e especificam as particularidades de cada micro-controlador, bem como as tarefas a serem realizadas (FreeRTOS 2005).

Devido a ênfase em micro-controladores, as tarefas são compiladas juntamente com o kernel do sistema, ou seja, o sistema como um todo é monolítico.

11. Aplicações de RTOS

Muitos sistemas de computadores possuem requisitos de tempo real, pelo menos de *soft real-time*. Isto significa que existem restrições temporais associadas à operação de tais sistemas. Alguns exemplos de aplicações que possuem requisitos de tempo real são:

- Controle automobilístico;
- Equipamentos médicos;
- Aplicações militares, controle aéreo e espacial;
- Jogos;
- Automação industrial;
- Aquisição de dados, robótica;
- Servidores Web;
- Telefone celular, VOIP, TV digital, etc.

12. Considerações Finais

Sistemas de tempo real estão presentes em diversas áreas. Nesse sentido é de fundamental importância o entendimento dos conceitos associados a aplicações tempo real e suas características. Também se deve dar importância aos sistemas operacionais de tempo real que, fornecendo serviços, disponibilizam os recursos necessários ao desenvolvimento de tais sistemas. Neste artigo foram apresentados os principais conceitos e características dos sistemas de tempo real, além de abordar-se sua relação com sistemas operacionais.

13. Referências Bibliográficas

[1] TANENBAUM, A. S. Sistemas Operacionais Modernos. 2ª ed. São Paulo, Pearson Prentice Hall, 2003.

[2] http://www.decom.ufop.br/prof/luizm/cic180_4.doc

[3] http://www.qnx.com

[4] http://www.freertos.org

[5] http://www.rtlinuxfree.com