ANÁLISE SOBRE O SISTEMA OPERACIONAL QNX

Antônio Walerio da Silva Dutra [1]
Helber Cardoso Lemos [1]
James Clébio Silva de Assis [1]
Matheus Alves dos Santos [1]
Tiago José Passos [1]

RESUMO

O QNX é um Sistema Operacional de Tempo Real (RTOS) baseado em UNIX amplamente utilizado no controle de sistemas embarcados e industriais. É utilizado em processos altamente críticos como controle de vôo, controle de esteiras de fábricas, sinais de trânsito e usinas nucleares. Destaca-se no mercado há mais de 20 anos pela sua confiabilidade e robustez, sendo encontrado em inúmeras aplicações aeronáuticas e aeroespaciais - situações onde qualquer erro do sistema simplesmente não pode acontecer.

Aracaju – Novembro de 2010

[1] Todos os autores são graduandos no curso Tecnólogo de Sistemas para a Internet, na Faculdade de Administração e Negócios de Sergipe - FANESE

1. INTRODUÇÃO

Inicialmente chamado de QUNIX, o QNX é um Sistema Operacional de Tempo Real (RTOS) baseado em UNIX amplamente utilizado no controle de sistemas embarcados e industriais. É utilizado em processos altamente críticos como controle de vôo, controle de esteiras de fábricas, sinais de trânsito e usinas nucleares. Mesmo sendo um sistema de arquitetura fechada e paga, o QNX disponibiliza uma versão gratuita para uso não-comercial.

O QNX se destaca no mercado há mais de 20 anos pela sua confiabilidade e robustez, sendo encontrado em inúmeras aplicações aeronáuticas e aeroespaciais - situações onde qualquer erro do sistema simplesmente não pode acontecer.

Sua versão mais recente é a QNX Neutrino RTOS 6.4.1, lançado em maio de 2009.

2. PRINCIPAIS CARACTERÍSTICAS DO SISTEMA

O QNX é um sistema operacional comercial de Tempo Real muito usado em operações embarcadas (Sistemas com requisitos específicos e tarefas pré- definidas), utiliza arquitetura de microkernel, é um sistema seguro e estável baseado na plataforma UNIX, o QNX é considerado o principal sistema operacional de tempo real no que diz respeito a família de processadores “x86”.

O kernel QNX contém apenas escalonador de CPU, comunicação inter-processos, redirecionamento de interrupção e temporizadores.

Todos os processos são executados em espaço de Usuário.

O Microkernel é preemptivo com escalonador baseado em prioridades.

Tem Suporte a clusters de tolerância a falhas e é otimizado para sistemas distribuídos (coleção de computadores autônomos interligados através de uma rede de computadores e equipados com software que permita o compartilhamento dos recursos do sistema: hardware, software e dados).

3. ARQUITETURA DO NÚCLEO (KERNEL)

O Kernel do QNX é de um tipo chamado microkernel. Muitos fabricantes dizem que seus sistemas operacionais possuem microkernel - ou até nanokernel. Muitas vezes não passa de marketing.

Um microkernel não é simplesmente um kernel pequeno. Ao contrário dos kernels tradicionais, a idéia é oferecer o mínimo de serviços necessários para que o sistema funcione. Os outros serviços são providos através de outros processos opcionais. No QNX o kernel chamado de "procnto".

A idéia não é simplesmente fazer um kernel pequeno, mas sim dar modularidade a ele, o que consequentemente diminuirá o tamanho. Ele é utilizado para juntar os módulos do sistema operacional. Muitos desenvolvedores acreditam que - devido à alta performance - ele foi desenvolvido em Assembly, quando na verdade foi desenvolvido na linguagem C. O alta performance está relacionada às sucessivas redefinições de algoritmos e estruturas de dados. Desde sistemas embarcados com memória extremamente limitada à sistemas com gigabytes de memória rodam o QNX.

A maioria dos serviços de tempo real e threads são implementados diretamente no microkernel, o que dispensa módulos adicionais para o acesso a esses serviços. Além disso, vários outros serviços são suportados, como semáforos e mutexes. Operações que exigem mais recursos, são carregadas em processos externos. Esse modelo derruba o mito de que sistemas monolíticos possuem um tempo de resposta mais rápido que sistemas que utilizam microkernel.

4. CHAMADAS AO SISTEMA (SYSTEM CALLS)

O QNX, por se tratar de um sistema microkernel, implementa um sistema de mensageria, o IPC, Comunicação entre Processos. Estas mensagens são pacotes de bytes passados de um processo para o outro, que permitem transmitir dados e meios de sincronização da execução de vários processos.

O IPC supervisiona o roteamento de mensagens e também gerencia as mensagens de proxies e sinais.

- Mensagens: Fornece comunicação síncrona entre processos cooperativos, onde o remetente exige uma prova de recebimento e uma confirmação para mensagem.
- Proxies: São especialmente adequados para notificação de eventos onde o remetente não precisa interagir com destinatário
- Sinais: Usados para suportar comunicação assíncrona entre processos.

4.1. MENSAGENS

Como o IPC de mensagens espera uma confirmação de recebimento, no seu conteúdo ela utiliza funções da linguagem C. Quando é realizada a sincronização entre processos, são utilizadas as funções SEND(), RECEIVE() e REPLY. A partir do momento que uma mensagem de sincronização é disparada, o processo remetente fica em estado de bloqueio, sendo liberado apenas ao receber o retorno do processo destinatário.

4.2. PROXIES

É uma forma de mensagem não bloqueante adequada especialmente para notificação de eventos onde o realmente não precisa interagir com o destinatário. A única função de um proxy é enviar uma mensagem fixa a um processo especifico que possui o proxy, onde é possível enviar a mensagem para um processo, sem que o remetente fique bloqueado ou esperando um retorno.

4.3. SINAIS

São métodos tradicionais de comunicação assíncrona. Um sinal é considerado entregue a um processo quando a ação deferida por ele é realizada pelo destinátario. Um

processo pode estabelecer um sinal sem nenhuma intervenção. Nenhum dado é transmitido com sinal. Entre o tempo que o sinal é gerado e o tempo que ele é entregue, ele é considerado pendente. Os sinais são entregues a um processo quando o processo esta pronto para pronto para executar no escalonador.

Se o processo não tomar nenhuma ação especial para manipular sinais, ação default é executada – finalizar o processo. Se o processo ignorar o sinal, não ocorre nenhum efeito no processo quando ele é entregue.

5. GERÊNCIA DE PROCESSOS E THREADS

"De forma Simplificada um thread pode ser definido como uma sub-rotina de um programa que pode ser executada de forma assíncrona , ou seja executada paralelamente ao programa chamador. O programa deve especificar os threads associando-os às sub rotinas assíncronas. Desta forma multithreads possibilita a execução de concorrente de sub-rotinas dentro de um mesmo processo." (MACHADO; MAIA, 2007)

Em ambientes multithread existe um ganho de desempenho, pois são diversas instruções em um mesmo processo, ou seja, é como se diversos processos estivessem sendo executados de forma concorrente, porém fazendo uso de apenas um contexto de hardware, dividindo assim o mesmo endereçamento, com isso a comunicação é bem mais rápida, pois não envolve mecanismos lentos. Também não existe a criação, troca e eliminação de processos, diminuindo assim o overhead. As threads são escalonadas assim como os processos.

5.1. THREADS

O QNX é um sistema operacional multithread. No QNX o responsável por decidir qual thread deve estar sendo executada em um dado instante é o kernel. Ele também é o responsável por escalonar, efetuar as trocas de contexto e salvar as informações da thread que esta saindo da CPU nos registradores de controle. Para garantir que duas threads não acessem simultaneamente um recurso compartilhado (seção critica), o QNX implementa a técnica de Exclusão Mutua Com sincronização condicional (semáforos).

6. ESCALONAMENTO (SCHEDULING)

As decisões de escalonamento são realizadas pelo micro-kernel nas seguintes situações:

- Quando o processo está bloqueado.
- O timeslice de um processo em execução esgotou.
- Processo em execução sofre preempção.

Todo processo criado no QNX recebe um prioridade, a qual o escalonador utiliza para selecionar o próximo processo com maior prioridade que deve estar em no estado ready (pronto), ou seja, podendo utilizar os recursos da CPU.

Os métodos de escalonamento podem ser de três tipos: escalonamento FIFO, escalonamento Round Robin, e escalonamento Adaptativo. Cada processo no sistema operacional pode utilizar qualquer um desses métodos, porém somente é aplicado os métodos em situações com dois ou mais processos com mesma prioridade e com o estado ready, ou seja , pronto para utilizar os recursos da CPU. Caso um processo com maior prioridade torne-se ready , então causa a preempção dos processos de baixa prioridade.

No escalonamento FIFO a condição que o método seja executado é que o processo renuncie o controle de forma voluntaria, não fazendo chamadas ao método ou através de preempção por um processo com maior prioridade.

No escalonamento Round-Robin, a condição para que o método continue executando é o mesmo procedimento do escalonamento FIFO , porém acrescido de mais um condição onde caso seja esgotado o timeslice do processo então há a preempção e o próximo processo no estado Ready terá o controle .

No escalonamento adaptativo, caso o timeslice do processo termine então a prioridade do processo é reduzida em uma unidade, procedimento conhecido como deterioração de prioridade. Neste caso o processo não é bloqueado. A prioridade do processo não cairá mais que uma unidade, mesmo que o timeslice esgote novamente sem bloquear. Caso o processo seja bloqueado então retorna para sua prioridade original. O escalonamento adaptativo é o método default do criado pelo Shell.

7. CONCLUSÃO

Atualmente, a QNX Software Systems é líder mundial em tempo real, tecnologia OS embarcada.

QNX é multiusuário, multitarefa, trabalha com rede e possui uma boa interface. A semelhança visual do QNX com o Linux é explicada pelo fato de ambos os projetos fazerem uso da interface gráfica PHOTON. Versões mais recentes do QNX possuem diversas aplicações nativas, dentre as quais se destaca o seu navegador de internet, o Voyager, que renderiza praticamente todo tipo de conteúdo (streaming de áudio e vídeo, flash, etc.) usado atualmente na web.

Por se basear em UNIX, o QNX é confiável e estável, podendo ser ideal para profissionais da área gráfica (3D, edição de imagem e vídeo e similares).

Líderes mundiais como a Cisco, General Eletric e Siemens dependem da tecnologia QNX para roteadores de rede, instrumentos médicos, unidades telemáticas de veículos, sistemas de segurança e de defesa, robótica industrial e outras aplicações de missões críticas.

Em 2010, a Research In Motion (RIM), fabricante do BlackBerry, adquiriu o QNX e o implementou em seu primeiro tablet PC, o PlayBook, concorrente direto do iPad, da Apple.

O fato de Gordon Bell e Dan Dodge, fundadores da QNX Software Systems, ainda representarem um papel ativo no desenvolvimento e codificação do QNX até hoje dá um respaldo maior à comunidade QNX no mundo.

8. REFERÊNCIAS

MACHADO, Francis; MAIA, Luiz. Arquitetura de Sistemas Operacionais. 4 ed. Rio de Janeiro: LTC Editora. Rio de Janeiro, 2007.

DO PRADO, Rodrigo; VICENTE, William. Sistema Operacional QNX. Monografia apresentada para banca na Universidade de Campinas: 1998.

TENENBAUM, Andrew S. Sistemas Operacionais Modernos. 2 ed. São Paulo: Pearson Education do Brasil, 1994.

KRTEN, Rob. QNX Neutrino RTOS. Ottawa: QNX Software Systems GmbH & Co. KG., 2009.