Stephenson de S. L. Galvão

Modelagem do Sistema Operacional de Tempo Real FreeRTOS

Natal - Rn, Brasil

1 de junho de 2009

Stephenson de S. L. Galvão

Modelagem do Sistema Operacional de Tempo Real FreeRTOS

Qualificação de mestrado apresentada ao programa de Pós-gradução em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte, como requisito parcial para a obtenção do grau de Mestre em Ciências da Computação.

Orientador:
Prof. Dr. David Déharbe

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO

Natal - Rn, Brasil

1 de junho de 2009

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Falar dos grandes desafios (SBC) e do desafio do compilador “Verifiging compile”, “Verified Software repository” desafio de Jim Woodcock

1.1 Objetivos

Falar do objetivo da dissertação e não só da qualificação. Items a serem discutidos:

- Abragência da especificação
- Profundidade em aspectos pelo menos da construção do software
- Se necessário tem a possibilidade de estessão até o nível de assemblagem devido aos códigos em assembler que compoem o FreeRTOS.

1.2 Metodologia

Métodologia da dissertação, no contexto do que já foi feito

2 *FreeRTOS*

O FreeRTOS é um sistema operacional de tempo real enxuto, simples e de fácil uso. O seu código fonte, feito em *C* com partes em *assembly*, é aberto e possui pouco mais de 2.200 linhas de código, que são essencialmente distribuídas em quatro arquivos: `task.c`, `queue.c`, `croutine.c` e `list.c`. Uma outra característica marcante desse sistema está na sua portabilidade, sendo o mesmo oficialmente disponível para 17 arquiteturas monoprocessadores diferentes, entre elas a PIC, ARM e Zilog Z80, as quais são amplamente difundidas em produtos comerciais através de sistemas computacionais embutidos.

Como a maioria dos sistemas operacionais de tempo real, o FreeRTOS provê para os desenvolvedores de sistemas concorrentes de tempo-real acesso aos recursos de *hardware*, facilitando com isso o desenvovimento dos mesmo. Assim, FreeRTOS trabalha como na figura 2.1, fornecendo uma camada de abstração localizada entre a aplicação e o hardware, que tem como papel esconder dos desenvolvedores de aplicações os detalhes do hardware que será utilizado.

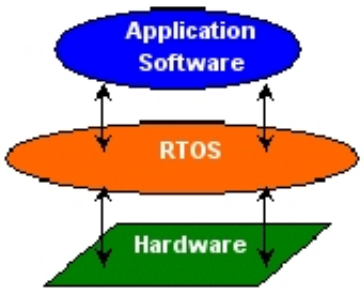


Figura 2.1: Camada abstrata proporcionada pelo FreeRTOS

Para prover tal abstração o FreeRTOS é composto por um conjunto de bibliotecas de tipos e funções que devem ser linkeditadas[1] com o código da aplicação a ser desenvolvida. Juntas, essas bibliotecas fornecem para o desenvolvedor serviços como gerenciamento de tarefa, comunicação e sincronização entre tarefas, gerenciamento de memória e controle dos dispositivos de entrada e saída.

[1]

A criação de uma aplicação utilizando o FreeRTOS pode ser divida em duas partes. Na primeira parte são criadas, de acordo com modelos fornecidos pelo FreeRTOS, as tarefas e demais **estruturas de controle** que serão utilizadas pela a aplicação. Na segunda parte é feito o cadastramento das tarefas utilizadas pelo o sistema assim como a inicialização do mesmo. Por fim, o sistema é **compilado** para arquitetura desejada.

A seguir serão detalhadas os principais serviços providos pelo o FreeRTOS junto com a biblioteca que disponibiliza tão serviço. Após isso será também demonstrado como é criada uma aplicação utilizando o FreeRTOS

2.1 Gerenciamento de Tarefas e CoRotinas

2.1.1 Tarefa

Para entender como fuciona o gerenciamento de tarefas do FreeRTOS é necessário primeiramente entender-se o conceito de tarefa. Uma tarefa é uma unidade básica de execução que compõem as aplicações, as quais geralmente são multitarefas. Para o FreeRTOS uma tarefa é composta por :

- Um estado que demonstra a atual situação da tarefa
- Uma prioridade que varia de zero até uma constante máxima definida pelo o usuário
- Uma pilha naqual é armazenada o ambiente de execução (estado dos restradores) da tarefa quando está é interrompida

Os possível estados que uma tarefa pode assumir são :

- **Em execução**: Indica que a tarefa esta sendo executada pelo processado
- **Pronta**: Indica que a tarefa está pronta para entrar em execução mas não está sendo executada
- **Bloqueada**: Indica que a tarefa esta esperando por algum evento para continuar a sua execução
- **Suspensa**:Indica que a tarefa foi suspensa pelo kernel através da chamada de uma funcionalidade usada para controlar as tarefas

A permutação que ocorre entre os estados de uma tarefa funciona como demonstra a figura 2.2. Nela uma tarefa com o estado “em execução” pode ir para o estado pronta, bloqueado ou suspenso, uma tarefa com o estado pronto pode ser suspensa ou entrar em execução e as tarefa com o estado bloqueada ou suspensa só podem ir para o estado pronto.

Entranto, vale enfatizar que por tratar-se de um SOTR para arquiteturas monoprocessadores o FreeRTOS não permite que mais de uma tarefa seja executada no mesmo momento. Assim no FreeRTOS apenas uma tarefa pode assumir o estado pronto em um determinado instante, restando as demais os outros estado. Com isso, para decidir qual tarefa deve ser executada existe um mecanismo no sistema operacional denominado escalonador, o qual será detalhado na sessão 2.2.

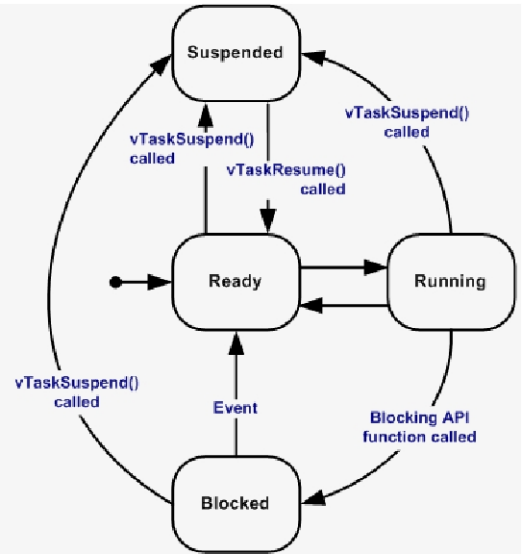


Figura 2.2: Grafo de estados de uma tarefa

Tarefa Ociosa

No FreeRTOS existe também uma tarefa denominada de tarefa ociosa, a qual é executada quando nenhuma tarefa está em execução. A tarefa ociosa tem como principal finalidade excluir da memória tarefas que não serão mais usadas pelo sistema. Assim quando uma aplicação informa para o sistema que uma tarefa não será mais utilizada essa tarefa só será excluida quando a tarefa ociosa entrar em execução. A tarefa ociosa possui a menor prioridade dentre as tarefas que compoem um sistema.

2.2 Escalonador de Tarefas

O escalonador é a parte mais importante de um sistema. É ele quem decide qual tarefa deve entrar em execução e realiza entre a tarefa que está no processador, ou seja em execução, com a nova tarefa que ira ocupar o processador, a tarefa que irá entrar em execução. No FreeRTOS o escalonador pode funcionar de três modos diferentes :

- **Preemptivo**: Quando o escalonador interrompe a tarefa em execução mudando o seu estado e ocupa o processador com outra tarefa
- **Cooperativo**: Quando o escalonador não tem permissões de interromper a tarefa em execução, tendo que esperar a mesma interromper a sua execução para que ele possa decidir qual será a próxima tarefa que irá entrar em execução e realizar a troca das mesmas.
- **Híbrido**: Quando o escalonador pode comporta-se tanto como preemptivo como cooperativo.

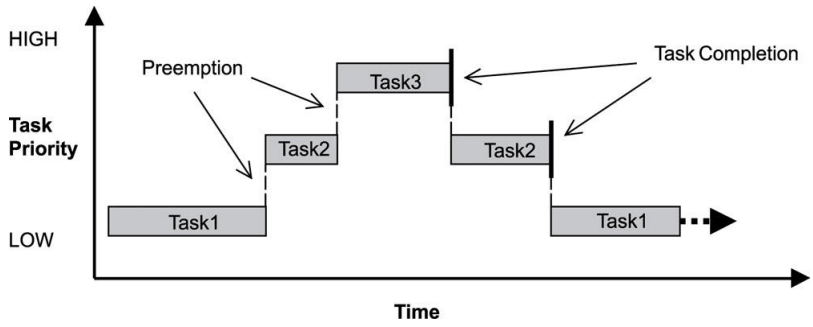
Para as tarefas o escalonador funciona de forma preemptiva, sendo que a decisão de qual tarefa deve entrar em execução e baseada na prioridade e segue a seguinte política: a tarefa em execução deve ter prioridade maior ou igual a tarefa de maior prioridade com o estado "pronta". Assim sempre que uma tarefa, com prioridade maior que a tarefa em execução, entrar no estado pronto, ele deve imediatamente entrar "em execução". Um exemplo claro da política preemptiva de prioridade discutida acima pode de visto na figura 2.3, naqual três tarefas, em ordem crescente de prioridade, disputam a execução do processador.

2.3 Corotinas

Outro conceito importante suportado pelo FreeRTOS é o de Corotina. Como as tarefas corotinas são unidades de execução independentes que formam uma aplicação. Por isso assim como as tarefas, uma corotina é formada por uma prioridade um estado, sendo a principal diferença entre uma corotina e uma tarefa a falta de uma pilha para armazenar o contexto de execução, a qual está presente nas tarefas e nas corotinas não.

Os estados que uma corotina pode assumir são:

- **Em execução**: Quando a corotina está sendo executada
- **Pronta**: Quando a corotina está pronta para ser executada mas não está em execução



1. Tarefa 1 entra no estado pronto, como não há nenhuma tarefa em execução esta assume controle do processador entrando em execução
2. Tarefa 2 entra no estado pronto, como está tem prioridade maior do que a tarefa 1 ela entra em exução passando a tarefa 1 para o estado pronto
3. Tarefa 3 entra no estado pronto, como está tem prioridade maior do que a tarefa 2 ela entra em exução passando a tarefa 2 para o estado pronto
4. Tarefa 3 encerra a sua execução, sendo a tarefa 2 escolhida para entrar em execução por ser a tarefa de maior prioridade no estado pronto
5. Tarefa 2 encerra a sua execução e o funcionamento do escalonador é passado para a tarefa 1

Figura 2.3: Funcionamento de um escalonador preemptivo baseado na prioridade

- **Bloqueada**: Quando a corotina esta bloqueada esperando por algum evento para continuar a sua execução.

As transições entre os estados de uma corotina ocorre como demonstra a figura 2.4. Nela uma corotina em execução pode ir tanto para o estado bloqueado como para o estado suspenso, uma corotina de estado bloqueado só pode ir para o estado pronto e uma corotina de estado pronto só pode ir para o estado “em execução”.

Assim como nas tarafes, a decisão de qual corotina irá entrar em execução é feita pelo o escalonador que para as corotinas funciona de forma cooperativa e besada na prioridade. Assim, quando uma corotina estiver em execução ela não será interrompida pelo o escalonador se uma tarefa de maior prioridade entrar no estado pronto. Com isso, o escalonador deve esperar que a corotina em execução interrompa [2] ou termine a sua execução para que ele possa escolher a próxima tarefa que ira assumir o controle do escalonador.

2.4 Bibliotecas

Bibliotecas que implementam os conceitos acima

[2]

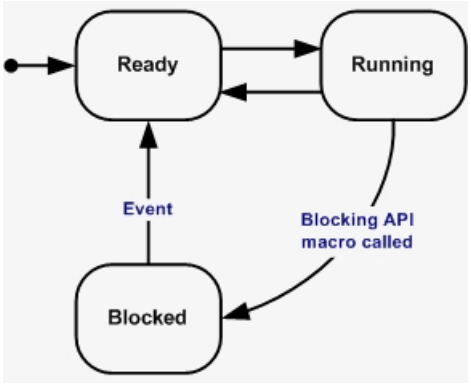


Figura 2.4: Grafo de estados de uma corotina

- Características e conceitos do FreeRTOS
- Apresentar a API
- Exemplo de como é feita uma aplicação de tempo real no FreeRTOS

- seção do artigo do SEMISH 2009 que fala do FreeRTOS
- seção do relatório da modelagem que fala do FreeRTOS
- site do projeto FreeRTOS
- outras fontes que falam sobre sistemas operacionais de tempo real

3 *Método B*

O método B trata-se de uma abordagem formal para especificar e construir sistema computacionais. Nele são encontradas várias qualidades presentes nos demais método formais que sugiram nos últimos trinta anos. Entre elas estão as pré e pos condições, condições necessárias para a execução de um método e alcançadas após a execução do mesmo, mudularização, abstração e refinamento, estratégia de construção/especificação de sistemas através de vários níveis de abstração.

A base do método B está na notação de máquina abstrada(em inglês: *Abstract Machine Notation* - AMN) a qual disponibiliza um framework comum para a especificação e contrução de sistemas, permitindo também a verificação formal do mesmo. Mais especificamente, a AMN trata-se de uma liguagem de especificação de sistemas formada por blocos básicos de construção chamados de Máquina Abstrata ou simplemente Máquina, nos quais são colocados conceitos (informações) bem definidas de parte do sistema. Assim para especificar grandes sistemas o Notação de Maquina Abstrata utiliza-se do paradigma composicional, sendo um sistema complexo formado pela a composição de várias Máquinas Abstratas o que gera também outra Máquina Abstrata

Máquina Abstrata

Para especificar as informações de um sistema uma máquina abstrata é dividida em vários cabeçalhos, nos quais as características do sistemas são semânticamente organizadas e listada. Os principais cabeçalhos presentes em uma máquina abstrata são Maquina, Operações, Variáveis, Invariante, Inicialização e Constantes. A seguir tem-se a explicação “semântica"do que trata cada um desses pedeços de uma especificação.

- Explicar o que é o método B
- Explicar a base teórica de B (AMN e as substituições)
- Explicar como é especificado um sistema em B (como é criado um módulo)

- Falar das obrigações de prova
- Falar dos mecânismo de composição e refinamento
- Dizer que o refinamento pode chegar em um nível concreto que pode ser sintetizado para algumas linguagens de programação.
- Falar do uso de ferramentas
- Falar do projeto B2ASM

4 Revisão Literária

- Enumerar Projetos
- Desafio de software verificado

5 *Proposta*

- Como será feita a modelagem do FreeRTOS
- Falar do estudo do FreeRTOS e identificação dos seus principais conceitos e funcionalidades
- O desenvolvimento progressivo acrescentando novas funcionalidades a cada refinamento
- Ligar a abordagem do compilador verificável ao FreeRTOS
- Dizer como será ou deve feita a união do FreeRTOS para o compilador verificável

6 Atividades e Etapas

Referências Bibliográficas