TOSThreads

Pedro Rosanes

30 de março de 2011

TOSThreads permite programação em thread no TinyOS, sem violar violar ou limitar o modelo de concorrência do sistema. O TinyOS executa em uma única thread, a nível de kernel enquanto a aplicação executa em uma ou mais threads, a nível de usuário. Em termos de escalonamento, o kernel tem prioridade máxima, ou seja, a aplicação só executa quando o núcleo do sistema está ocioso. Ele é responsável pelo escalonamento de tarefas e execução das chamadas de sistemas.

A interface entre as threads a nível de usuário e o kernel é feito através de chamadas de sistema bloqueantes. Essas chamadas são implementadas aproveitando os serviços já disponíveis do TinyOS. São responsáveis por manter o estado do serviço *split-phase* que será usado, bloquear a thread que a invocou e acordar a thread do kernel.

Passam a existir três tipos de contextos de execução: tarefas, interrupções e threads. Tarefas e interrupções podem interromper threads de aplicação, mas não o contrário. Threads tem preempção entre elas, de modo que é necessário o uso de primitivas de sincronização. As opções fornecidas são *Mutex*, semáforos, barreiras, variáveis de condição, e contador bloqueante. Esta ultima foi desenvolvida especialmente para o TinyOS. Seu uso se dá de forma que a thread fica bloqueada até o contador atingir um número arbitrário, enquanto outras threads podem incrementar ou decrementar esta variável através de uma interface.

O TinyOS retoma o controle sobre a aplicação de dois modos diferentes. No primeiro, uma aplicação faz uma chamada de sistema que posta uma tarefa para processar o serviço. No segundo modo, um manipulador de interrupção posta uma tarefa. Porém, neste caso o TinyOS só acorda depois de terminada a execução da interrupção.

O escalonador de threads utiliza uma política *Round-Robin* com um tempo de 5 milisegundos. É ele que oferece toda a interface para manipulação de threads, como pausar, criar e destruir. É interessante nota que o escalonador não existe em um contexto de execução específico, seu contexto depende de quem utilizou sua interface.

As threads podem ser estáticas ou dinâmicas. A diferença é o momento de criação da pilha e do bloco de controle da thread. O primeiro tipo em tempo de compilação, o segundo em tempo de execução. O bloco de controle, também chamado de *Thread Control Block* (TCB) contém informações essenciais da thread, como seu identificador, seu estado de execução, o valor dos registradores (para troca de contexto), entre outras.

A troca de contexto é feita por códigos específicos para cada plataforma. É utilizada a linguagem assembly junto com C, para armazenar o valor dos registradores importantes na TCB.

Referências

[1] P. Levis e C. Sharp. TEP 134: The TOSThreads Thread Library. http://www.tinyos.net/tinyos-2.x/doc/html/tep106.html

[2] TOSThreads, TinyOS Tutorial. http://docs.tinyos.net/index.php/Boot_Sequence