ReTMiK (Real-Time Micro-mouse Kernel) **v2.0**

Bruno Gravato, Bruno Pereira

***Resumo* – O ReTMiK é um kernel de Tempo-Real, criado em 1997 no âmbito do concurso Micro-Rato da UA. Desde 1998 que foi utilizado por várias equipas no referido concurso, sem nunca ter sofrido qualquer alteração. Em finais de 2003, no âmbito da disciplina Sistema de Tempo-Real, opção do 5º ano da LEET, foram adicionadas novas funcionalidades a este kernel por forma a disponibilizar novas possibilidades aos seus utilizadores, bem como para auxiliar o desenvolvimento de um sistema operativo baseado no ReTMiK. Entre as principais novidades desta versão encontram-se novos mecanismos para facilitar a sincronização e gestão de tarefas, novos algoritmos de escalonamento e a adopção de uma licença de software livre.**

I. INTRODUÇÃO

O kernel ReTMiK [1] foi desenvolvido em finais de 1997 para o Concurso Micro-Rato [2] da UA, tendo sido utilizado por alguns participantes a partir da edição de 1998, e.g. [3]. Funciona sobre a plataforma kit188 (também referida como DET188) e foi desenvolvido em linguagem C, usando o compilador TurboC 2.0. Este kernel *multitasking* permite definir tarefas periódicas e com relação de fase, as quais são activadas automaticamente, de forma transparente para o utilizador. O código das tarefas é reentrante e o kernel permite preempção. Utiliza um escalonador de tempo-real que, nesta versão, permite usar dois algoritmos - *Deadline Monotonic* e *Earliest Deadline First*. Está particularmente adaptado a suportar o desenvolvimento de programas de controlo de robôs baseados em comportamentos autónomos, constituindo uma ajuda preciosa ao programador. A sua arquitectura do sistema está ilustrada na Fig. 1.

Em 2003 no âmbito do trabalho final de uma disciplina opcional do 5º ano da Licenciatura em Engª de Electrónica e Telecomunicações, Sistemas de Tempo Real [4], leccionada pelo professor Luís Almeida, foram acrescentadas novas funcionalidades, entre as quais se destacam a sincronização de tarefas no acesso a recursos partilhados (recorrendo à não preempção), novos algoritmos de escalonamento, extensões à gestão de tarefas (singulares ou grupos) e adopção da licença GPL [5].

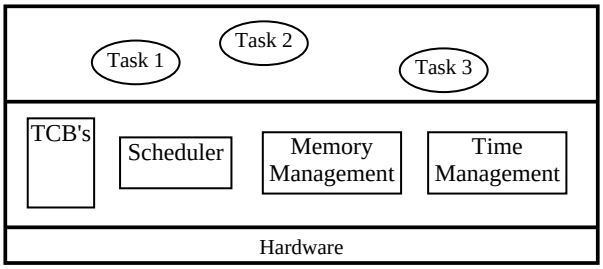

Fig. 1 – Arquitectura do Sistema

II. SINCRONIZAÇÃO DE TAREFAS

Na versão anterior do ReTMiK, a única forma de impedir a preempção de uma tarefa durante o acesso a um recurso partilhado, era a inibição das interrupções. Este mecanismo funciona bem para intervalos de tempo muito curtos mas não é aconselhável para períodos longos pois bloqueia todas as tarefas em execução no sistema, inclusivé a gestão temporal do kernel, perda de *ticks*. Esta versão do ReTMiK permite (des)activar a preempção em qualquer momento, por qualquer tarefa, sem inibir as interrupções. Embora continue a causar bloqueio a todas as tarefas, este mecanismo não perturba a gestão temporal do kernel. Para o efeito são disponibilizadas as macros ***set_preempt()*** e ***set_nopreempt()***, e ainda ***get_preempt()*** que permite verificar o estado de preempção actual.

III. ESCALONAMENTO

O escalonador original do ReTMiK apenas permitia um algoritmo de escalonamento – *Deadline Monotonic (DM)*, mas o respectivo interface de criação de tarefas forçava as *deadlines* a serem iguais aos períodos o que na prática resultava num escalonamento *Rate Monotonic (RM)*. Assim, retirou-se tal limitação, permitindo definir *deadlines* iguais ou inferiores ao período. Acrescentou-se ainda um outro algoritmo – *Earliest Deadline First (EDF)*, bem como a possibilidade de escolha e alteração do algoritmo a usar, quer antes, quer durante a execução das tarefas, com efeito imediato.

A Fig. 2 ilustra uma aplicação com três tarefas, em que se pode observar a diferença de um escalonamento *DM* e *EDF*. De forma breve, a utilização de *EDF* é mais complexa mas resulta num menor número de preempções. As diferenças tornam-se mais nítidas em situações de carga elevada, em que o escalonador *EDF* exibe alguma equidade (*fairness*) na atribuição do CPU às tarefas, ao contrário do escalonador *DM*, cujo comportamento está rigidamente orientado por prioridades fixas.

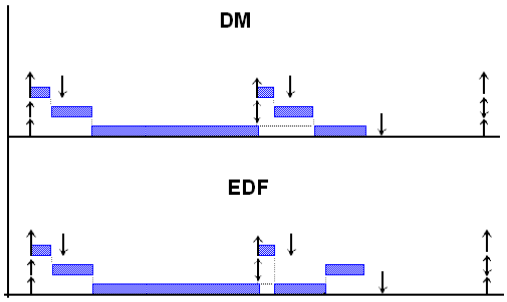

Fig. 2 – Gráficos de Gantt de uma aplicação com 3 tarefas escalonadas por DM e por EDF

IV. Gestão de Tarefas

O desenvolvimento de um sistema operativo baseado no ReTMiK realizado por outro grupo evidenciou a necessidade de complementar os mecanismos de gestão de tarefas existentes no *kernel*. A versão 2.0 do ReTMiK fornece um conjunto de mecanismos que permite activar ou parar tarefas individualmente ou em grupo, bem como destruir tarefas, libertando os respectivos recursos. A Fig. 3 ilustra o novo diagrama de estados do kernel.

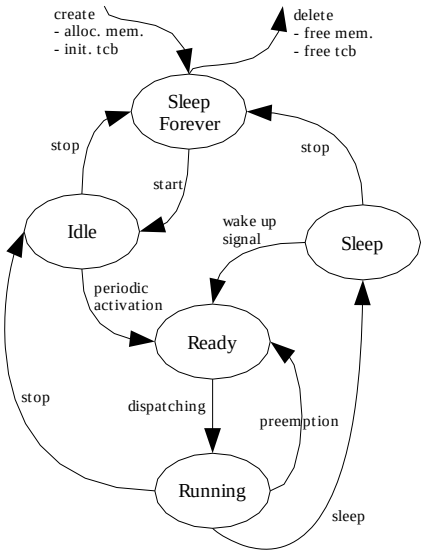

Fig. 3 – Diagrama de estados

A. *Tarefas Singulares*

Além do iniciar síncrono da execução de todas as tarefas com a primitiva ***start_all()***, é agora possível iniciar e parar a execução individual de tarefas, recorrendo às primitivas ***start_task()*** e ***stop_task()***. Este mecanismo pressupõe que essas tarefas existam já em memória, após serem criadas com a função ***create_task()***. Para permitir uma gestão de tarefas mais completa, esta versão inclui agora a função ***delete_task()***, que permite destruir uma tarefa libertando os respectivos recursos (*stack* e *TCB-Task Control Block*).

Foi ainda acrescentada a primitiva ***get_status()*** que permite conhecer o estado de execução de uma tarefa**.**

B. *Grupos de Tarefas*

Por vezes é necessário iniciar ou parar a execução de conjuntos de tarefas sincronamente. De forma a colmatar esta necessidade o ReTMiK permite definir grupos de tarefas (campo *group* no *task descriptor*). O início ou paragem sincronos da execução destes grupos é possível graças às novas primitivas ***start_group()*** e ***stop_group()***. Por questões de velocidade de implementação, estas primitivas recorre actualmente à inibição das interrupções, o que causa bloqueios ao *kernel* às tarefas. Contudo, esta limitação poderá ser resolvida mais tarde, integrando o código relativo à alteração do estado e *flags* das tarefas dentro do ***tick_handler()***.

V. Licença GPL

Questões como a propriedade intelectual relativamente a software, têm sido bastante discutidas a nível mundial.

Tratando-se de um projecto académico e que está disponível publicamente, achou-se que seria benéfico adoptar uma filosofia de software livre, que permitisse a qualquer pessoa utilizar e alterar o ReTMiK, preservando, ao mesmo tempo, os direitos de autor. Como tal, adoptou-se a Licença Pública Geral GNU (*GNU General Public License*, *GNU GPL*, ou simplesmente *GPL*), por conceder e proteger as 4 liberdades para o Software Livre [6], segundo Richard M. Stallman [7] (fundador da *Free Software Foundation* [8] e do projecto *GNU* [9]).

VI. Conclusão

Esta nova versão do kernel ReTMiK acrescenta importantes funcionalidades anteriormente inexistentes, como a sincronização de tarefas no acesso a recursos partilhados, recorrendo à não preempção sem inibição de interrupções; novos algoritmos de escalonamento, nomeadamente *DM* e *EDF*, com possibilidade de alteração dinâmica; mecanismos de gestão dinâmica de tarefas singulares e de grupos de tarefas; incentivo ao desenvolvimento e promoção de software livre [8], adoptando a Licença Pública Geral GNU (*GNU GPL*) [5].

Uma descrição mais detalhada do funcionamento e primitivas disponíveis do kernel ReTMiK encontra-se disponível no manual de utilização [1] do mesmo.

Referências

[1] http://sweet.ua.pt/~lda/retmik/retmik.html
[2] http://microrato.ua.pt/
[3] http://www.bulldozer.4mg.com/index.htm
[4] http://sweet.ua.pt/~lda/str/str.htm
[5] http://www.gnu.org/licenses/gpl.html
[6] http://www.ansol.org/filosofia/softwarelivre.pt.html
[7] http://www.stallman.org/
[8] http://www.fsf.org/
[9] http://www.gnu.org