

Xenomai Short Intro

Paulo Pedreiras
pbrp@ua.pt
DETI/University of Aveiro

Sistemas Tempo-Real
Out/2013

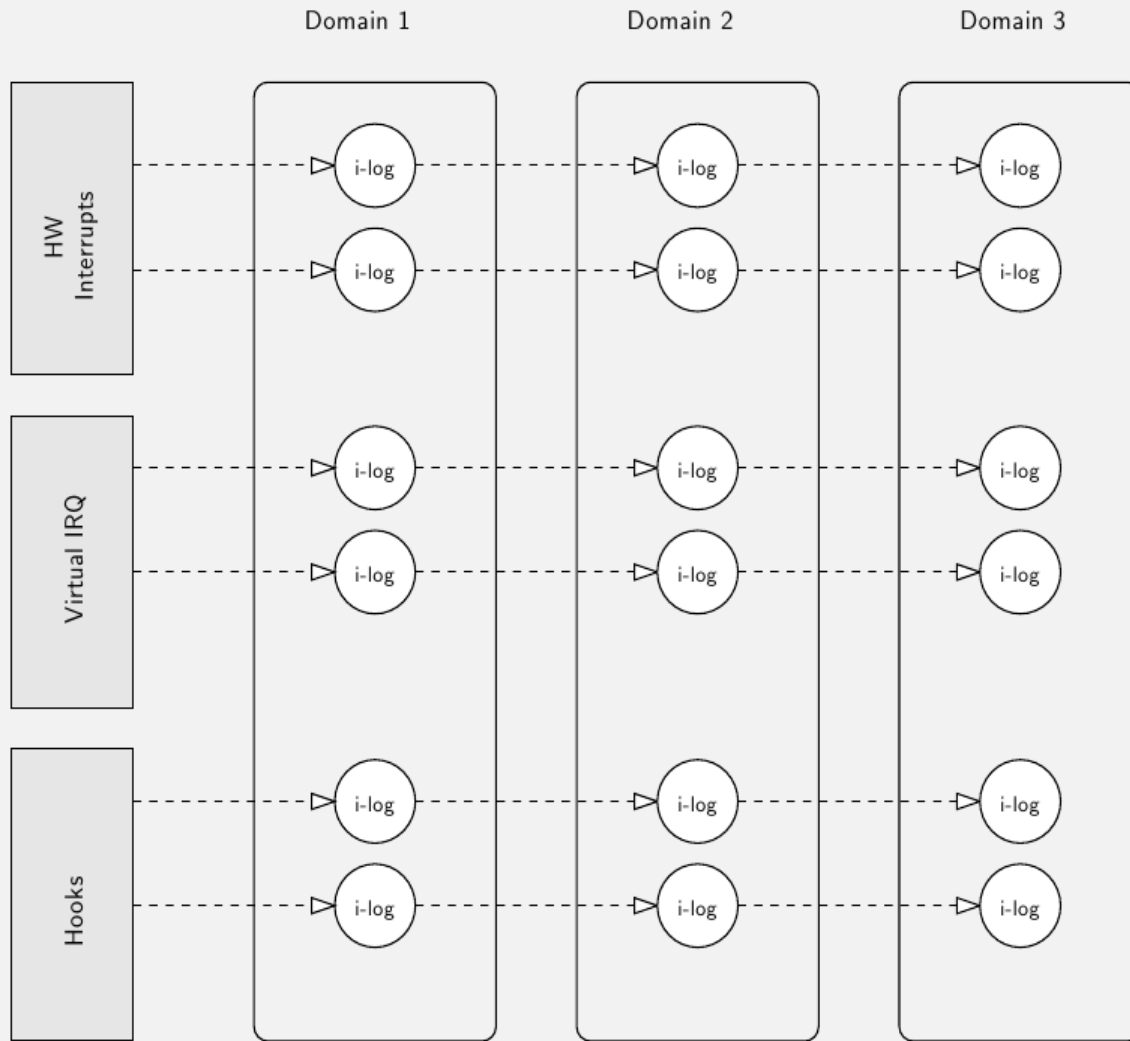
(Rev. 1 - Out/2015)

Agenda

- Adeos
- Xenomai
 - Introdução
 - Estrutura de domínios
 - Interrupções
 - Threads em modo utilizador
 - Interfaces
 - Exemplo de aplicação
 - Sumário

- Adeos (Adaptive Domain Environment for Operating Systems)
 - É uma camada de abstracção de hardware integrada com um nanokernel
 - Também conhecido por “hypervisor”
 - Opera directamente sobre o HW e suporta um ou mais sistemas operativos
 - Cada SO faz parte de um domínio distinto
 - Cada domínio contém uma entidade para gerir interrupções
 - Estas interrupções são definidas em sentido lato e incluem
 - Interrupções propriamente ditas, geradas por HW e SW,
 - *hooks* para mudanças de contexto,
 - etc.

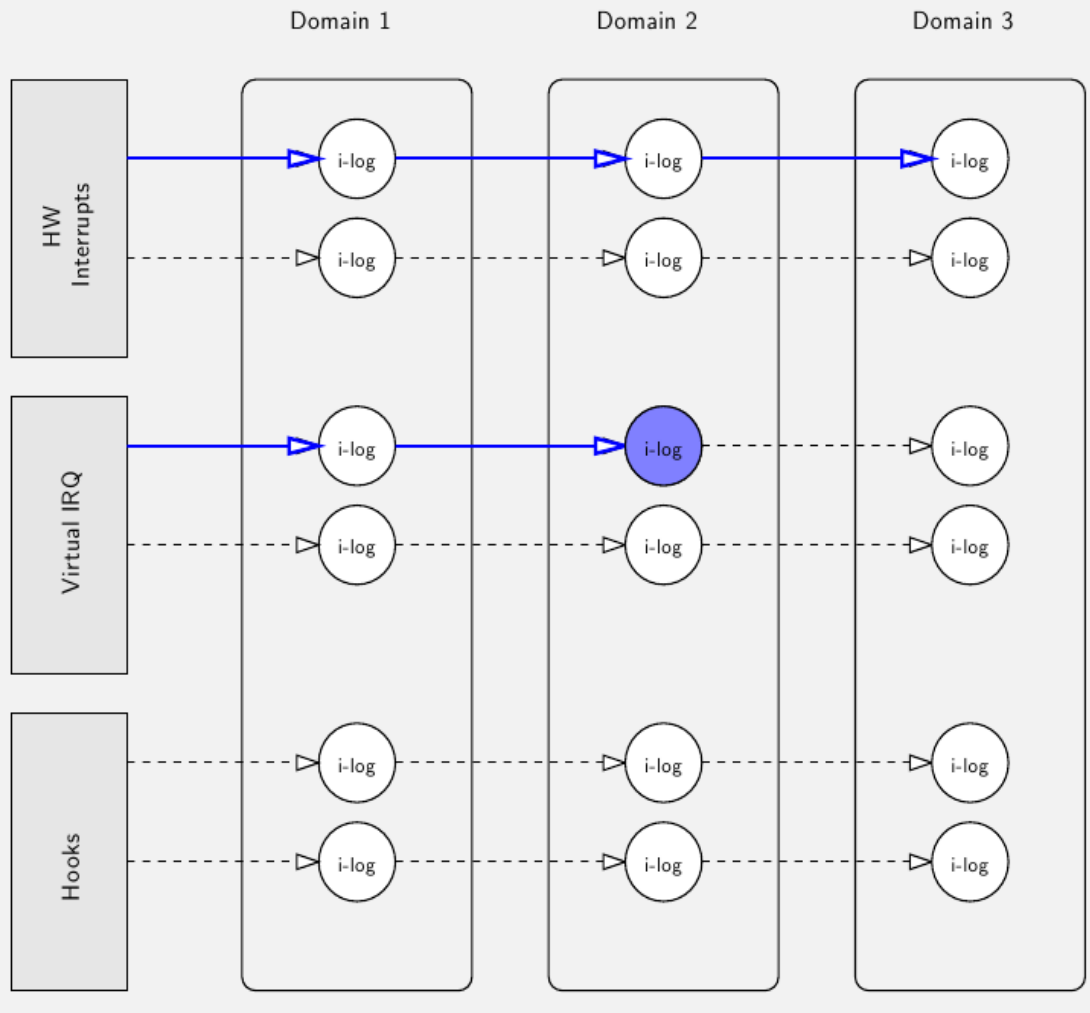
Adeos



- Domínios são identificados por um número único
- E estão organizados por prioridades
 - Menor número -> maior prioridade a tratar os eventos
- Eventos são propagados de acordo com uma pipeline

- Propagação de eventos
 - Os eventos propagam-se do primeiro domínio até ao último
 - Um domínio pode encaminhar o evento para o domínio subsequente ou pará-lo
 - Um domínio pode também suspender eventos
 - Equivalente a desligar as interrupções para os domínios subsequentes
 - Os eventos podem ser retomados posteriormente
 - A suspensão de eventos pode ser efectuada selectivamente

Xenomai no Adeos

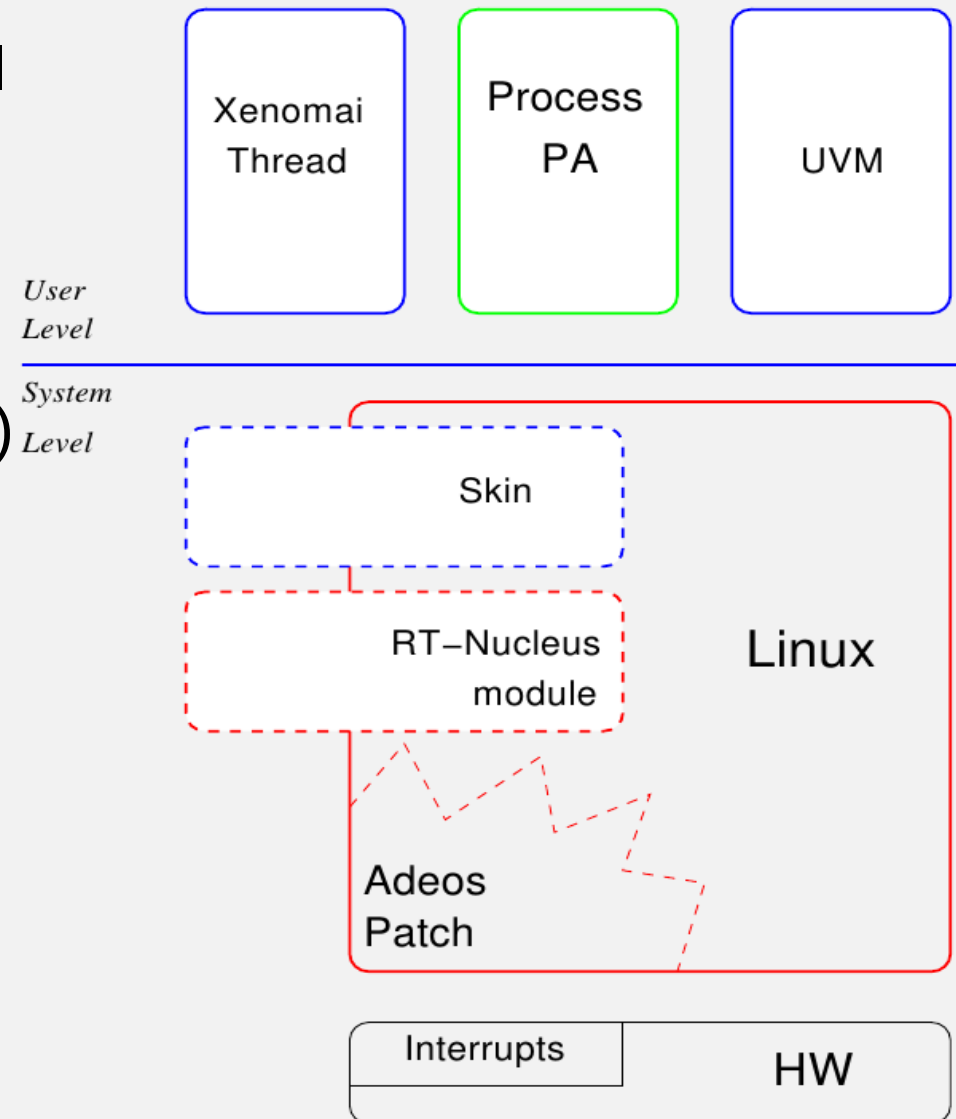


- Ideia básica

- Linux no domínio 3
- RTOS no domínio 1
- Domínio 2 usado para suspender selectivamente eventos
- Operações Linux para fazer enable/disable de interrupções são modificadas para *stall* e *unstall*

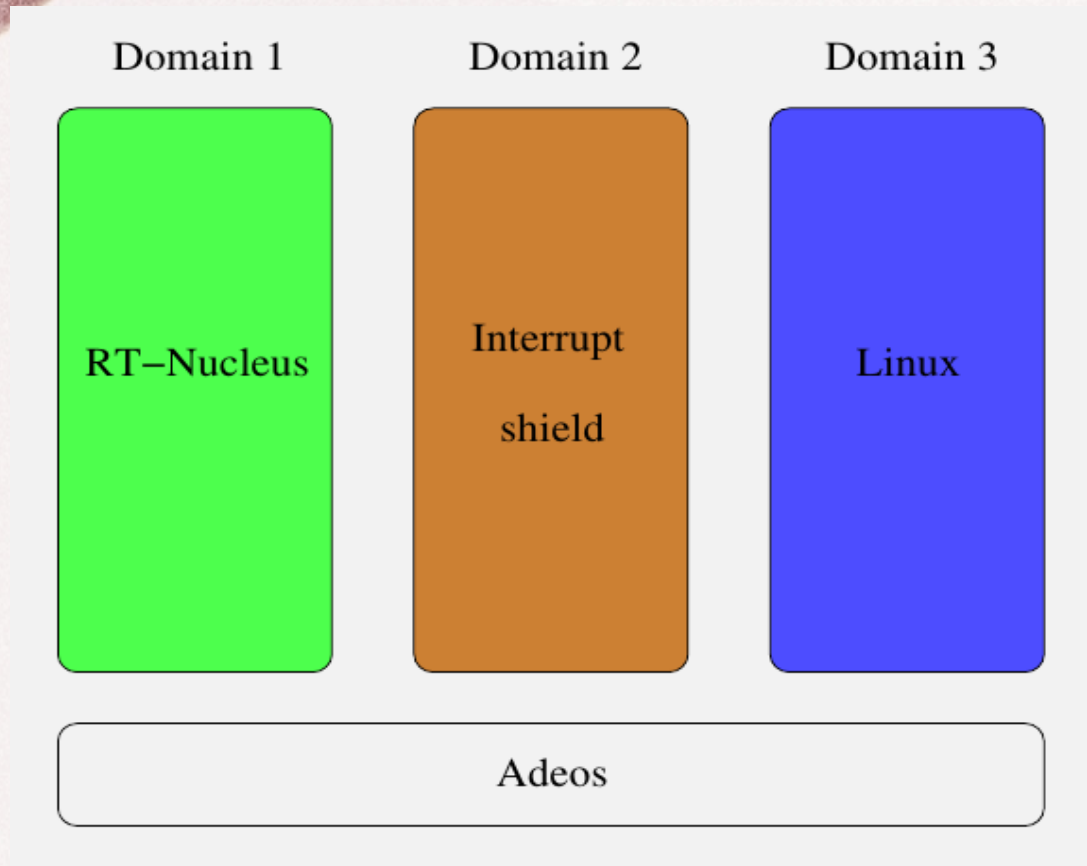
Xenomai

- Surgiu como um ramo do RTAI
- Documentação é bastante extensa e completa
- Possui um conjunto de primitivas nativas (RT-Nucleos) e um conjunto de skins que emulam outros RTOS
 - Estes incluem VXWorks, u-Ittron, ...



Estrutura de domínios

- Estrutura de domínios em Xenomai



- Domínio principal: Xenomai
- Domínio secundário Linux
- Domínio intermédio: interrupt shield

Xenomai threads

- Uma thread tempo-real pode
 - Executar sempre no domínio primário
 - Espaço de endereçamento é “kernel space”
 - Tempos de resposta muito curtos
 - Semelhante a RTLinux e RTAI
 - Executar nos domínios primário e secundário
 - São designadas “Xenomai Threads”
 - Espaço de endereçamento é “user space”
 - Espaço de endereçamento é privado
 - Não interferem com a memória de outros processos nem do kernel
 - Facilita debug
 - As threads Xenomai começam em domínio primário e mudam automaticamente para secundário quando invocam uma “system call” não RT

Xenomai threads

- As prioridades usadas no domínio principal (RT-Nucleos) são compatíveis com as usadas no secundário (Linux)
- Quando uma thread Xenomai migra de domínio preserva a sua prioridade
- As threads Xenomai iniciam-se no modo primário
- Quando no domínio primário uma thread
 - É removida da ready queue do Linux
 - É escalonada pelo escalonador do RT-Nucleos
 - Tem sempre precedência sobre qualquer tarefa Linux, qualquer que seja a sua prioridade
 - As tarefas Linux são escalonadas em background em relação às tarefas em domínio primário

Xenomai threads

- Uma thread Xenomai mantém-se em modo primário até que invoque uma primitiva não TR
 - E.g. escrever para um ficheiro
- Quando tal acontece é mudada transparentemente para o domínio secundário
- Quando é movida para o domínio secundário, uma tarefa Xenomai:
 - É inserida na ready queue do Linux
 - O RT-Nucleos invoca o scheduler do Linux
 - As tarefas Xenomai migradas comportam-se como se fossem tarefas nativas Linux, nomeadamente:
 - Podem fazer preempção sobre tarefas Linux
 - Podem sofrer preempção de tarefas Linux
 - Perda de propriedades tempo-real!

Interrupts

- Para reduzir a latência
 - Os interrupts não são imediatamente encaminhados para o Linux se uma Xenomai Thread está em execução
 - Este mecanismo funciona da seguinte forma:
 - Quando uma Xenomai Thread está em execução o “Interrupt shield domain” é activado
 - Todos os interrupts Linux são suspensos
 - Esta suspensão termina quando a Xenomai Thread termina
 - E assim os interrupts pendentes são servidos
 - Desta forma a latência das Xenomai threads é reduzida, mesmo quando executam em domínio secundário.

Interfaces

- O Xenomai disponibiliza diversas API ao utilizador
 - A API interna (core) é a interface usada pelo RT-Nucleos.
 - Não deve ser usada directamente
 - Há diversas skins
 - Uma é a Native Xenomai Interface
 - Pode ser usada em novas aplicações
 - Conjunto de primitivas bem organizado
 - As skins são módulos que podem ou não ser usados
 - Exemplos
 - POSIX
 - RTAI
 - U-Ittron
 - VxWorks
 - A ideia é facilitar o porting de aplicações já existentes

Exemplo

- Corpo da tarefa

```
void task_a(void *cookie) {
    RTIME to=0,ta=0;
    unsigned long overruns;
    int err;

    rt_printf("Task a init\n");

    /* Set task as periodic */
    err=rt_task_set_periodic(NULL, TM_NOW, TASK_A_PERIOD_NS);
    // Ciclo infinito ("job")
    for(;;) {
        err=rt_task_wait_period(&overruns); // Suspende até novo período
        ta=rt_timer_read();
        if(err) {
            rt_printf("Task a overrun!!!\n");
            break;
        }
        rt_printf("Task a activation\n");
        if(to!=0)
            rt_printf("Measured period (ns)= %lu\n",ta-to);
        to=ta;

        /* Task "load" */
        simulate_load(TASK_A_LOAD_NS);
    }
    return;
}
```

Exemplo

- Criação de tarefas

.....

```
/* Create RT task */
/* Args: descriptor, name, stack size, priority [0..99] and mode (flags for CPU, FPU,
joinable ...) */
err=rt_task_create(&task_a_desc, "Task a", TASK_STKSZ, TASK_A_PRI0, TASK_MODE);
if(err) {
    rt_printf("Error creating task a (error code = %d)\n",err);
    return err;
} else
    rt_printf("Task a created successfully\n");
```

```
/* Start RT task */
/* Args: task descriptor, address of function/implementation and argument*/
rt_task_start(&task_a_desc, &task_a, 0);
```

.....

Sumário

- Xenomai
 - Suporta modo kernel/tempo-real com alto desempenho
 - Permite execução em user-space, logo
 - Protecção de memória
 - Facilidade de debug
 - Prioridades são portadas entre domínios
 - Latência melhorada em user-space por meio de interrupt shielding
 - Skins para diversos RTOS
 - Facilita porting de aplicações

Bibliografia e links

- Building a RTOS over Adeos (<http://home.gna.org/adeos/>)
- Xenomai Programming Interfaces (http://www.xenomai.org/index.php/Included_documentation_summary#Programmer.27s_reference_manuals)
- Karim Yaghmour, Jon Masters, "Building Embedded Linux Systems, 2nd Edition", O'Reilly Media, Inc, 2008 ISBN: 0596529686