RTAES - Threads e prioridades

António Barros, Cláudio Maia

Objectivos

Os objectivos deste conjunto de actividades são os seguintes:

1. Implementação de programas multi-thread, com a definição de tarefas periódicas.
2. Atribuição de prioridades específicas a cada thread.
3. Observação da execução concorrente das threads de um processo, sob as políticas de escalonamento SCHED_FIFO e SCHED_RR.

1 Introdução

Por definição, todos os programas implementados na linguagem de programação C começam a sua execução na função principal `int main()`. Essa *linha* (*thread*) de execução é a primeira e, muitas vezes, a única de um programa. Mas, a norma POSIX permite a existência de linhas de execução paralelas e autónomas num processo. Cada linha segue o seu percurso, de forma autónoma das outras linhas – com o seu *program counter* e *stack* específicos – até, eventualmente, terminar.

Uma thread termina quando a função que define retorna (i.e. "chega ao fim"), ou quando é chamada explicitamente a função `pthread_exit()`. Pode consultar a página do manual

```
$ man 3 pthread_exit
```

Deve ter em atenção que um processo termina em uma de duas situações:

1. a função principal chega ao fim e retorna normalmente, ou
2. a última thread viva do processo termina.

No caso 1., o retorno da função principal causa o término de todas as threads que possam estar a executar. A melhor forma de evitar esta situação (se for desejável), é terminar a thread principal com a chamada da função `pthread_exit()`, em vez de permitir um retorno normal.

```
int main()
{
  /* (...) */
  pthread_exit(NULL);
  return 0;
}
```

1.1 Criar uma thread

Com excepção da primeira thread, cada thread adicional tem que ser criada explicitamente com a função `pthread_create()`. A explicação detalhada desta função pode ser encontrada na página do manual

```
$ man 3 pthread_create
```

1.2 Definir a prioridade de uma thread

A prioridade de uma thread é definida através da função `pthread_setschedparam()`. Deve consultar a página do manual para mais detalhes sobre esta função

```
$ man 3 pthread_setschedparam
```

1.3 Definir a afinidade da thread a núcleos de processamento específicos

A afinidade de uma thread é definida através da função `pthread_setaffinity_np()`. Deve consultar a página do manual para mais detalhes sobre esta função

```
$ man 3 pthread_setaffinity_np
```

1.4 Suspender uma thread até ao próximo instante de activação

Para se implementar uma thread periódica, é necessário estabelecer um mecanismo para activar a thread no instante de activação seguinte. A função `clock_nanosleep()` define um temporizador que irá disparar no instante indicado na sua chamada. Para obter detalhes sobre esta função, consulte a página de manual

```
$ man 2 clock_nanosleep
```

2 Implementar programas multi-thread

1. Crie dois programas diferentes, um usando a política de escalonamento SCHED_FIFO e o outro usando SCHED_RR. Cada programa deve criar três tarefas periódicas (usando threads), cada qual com a respectiva prioridade e periodo, diferentes entre si. Para o WCET, faça um teste de modo a estabelecer uma relação entre um determinado número de iterações de um ciclo e o seu tempo de execução aproximado. Assuma que o número de iterações do ciclo representa o trabalho realizado por um *job* da tarefa de tempo real em questão e que cada tarefa tem um número de *jobs* potencialmente infinito.
2. Execute o programa com o `trace-cmd` a registar os eventos de escalonamento, e observe os eventos com o `kernelshark`.
3. Verifique se o traço de execução das threads está de acordo com o previsto, dadas as prioridades e periodos atribuídos.
4. Observe se existem alguns momentos em que as threads são interrompidas (qualquer que seja a sua prioridade) para permitir ao kernel executar alguma operação (processo `kworker`).
5. Verifique que nos momentos em que nenhuma tarefa tem trabalho para executar, o kernel escalona o processo `swapper` que coloca o processador em inactividade (*idle*).
6. Varie os atributos das threads, para observar as possíveis alterações do traço resultante.
7. Compare o traço de execução dos diferentes programas.

Nota: Compile o programa acima, indicando que pretende incluir a biblioteca `pthread`. Exemplo:

```
$ gcc periodic.c -lpthread -o periodic
```