# RTAES - Observação do escalonamento de processos

António Barros, Cláudio Maia

## Objectivos

Os objectivos deste conjunto de actividades são os seguintes:

1. Utilizar o módulo do kernel `ftrace` para monitorização de eventos do kernel.
2. Instalar a aplicação de registo de eventos do kernel `trace-cmd`.
3. Instalar a aplicação de visualização de registo de eventos `Kernelshark`.
4. Observação do comportamento do kernel no escalonamento de processos.

# 1 Introdução

## 1.1 Monitorização do kernel

Monitorizar a actividade do núcleo de um sistema operativo é uma tarefa mais complexa do que monitorizar um programa que execute no espaço de utilizador (*user space*). Esta complexidade deve-se ao facto de que nenhuma aplicação que execute no espaço de utilizador pode aceder ao espaço do núcleo (*kernel space*). Desta forma, só o próprio núcleo pode observar e registar a sua actividade.

O Linux tem um módulo que permite expor os eventos que ocorrem no núcleo: o `ftrace` (*function tracer*). Esta *framework* foi inicialmente desenvolvida por Steven Rostedt, com o intuito de permitir aos desenvolvedores observar a actividade do núcleo. Entre os vários eventos que o monitoriza, estão os eventos associados ao escalonador.

## 1.2 Registar eventos do kernel

O `trace-cmd` é um utilitário que recolhe informação a partir dos dados expostos pelo `ftrace` e permite filtrar e registar num ficheiro de texto os eventos de interesse. É uma ferramenta bastante útil na análise de módulos do kernel.

Quando o `trace-cmd` regista um período de actividade do sistema, os resultados são guardados num ficheiro de texto chamado `trace.dat`. Neste ficheiro, a sequência de eventos é registada linha-a-linha, por ordem cronológica.

## 1.3 Observação dos traços

A inspecção visual do ficheiro `trace.dat` permite-nos observar a sequência de chamadas de funções do kernel. Porém, o número de eventos pode ser grande, o que dificulta a sua análise. O `Kernelshark` é um utilitário de visualização dos traços registados pelo `trace-cmd`, apresentando o seu conteúdo através de gráficos temporais de actividade e de listas de eventos.

**Nota:** Este utilitário corre em modo gráfico, pelo que é necessário um ambiente gráfico.

# 2 Preparação

## 2.1 Instalação do `trace-cmd` e do `Kernelshark` no Raspberry Pi

A instalação destes utilitários deve ser feita através da seguinte linha de comando:

```
1 $ sudo apt-get install trace-cmd kernelshark
```

O gestor de pacotes `apt` irá procurar e apresentar todos os pacotes necessários para instalar estes dois utilitários. Responda ao `apt-get` que aceita instalar os pacotes indicados.

## 2.2 Sessão remota gráfica

Para poder visualizar o `Kernelshark` é necessário preparar o computador de trabalho e, possivelmente, o Raspberry Pi para ser permitir uma sessão remota gráfica. Com o Raspberry Pi, há duas possibilidades imediatas:

- **X Window System – X11**
  O computador de trabalho tem que ter um servidor X instalado. No Raspberry Pi, o servidor SSH tem incorporado um cliente X, pelo que suporta nativamente este tipo de sessões.
- **Virtual Network Computing – VNC**
  O computador de trabalho tem que ter um cliente VNC instalado. No Raspberry Pi, é necessário activar o servidor VNC.

### 2.2.1 X11

Do lado do Raspberry Pi não é necessária qualquer configuração, pois o servidor SSH já inclui um cliente X. No computador de trabalho, é necessário ter instalado um servidor X, para apresentar as janelas das aplicações gráficas.

**Linux** Os sistemas Linux com configuração gráfica (*desktop*) costumam trazer instalado o gestor de janelas X11 `xorg`. Para verificar que o `xorg` está instalado, execute a seguinte linha de comando:

```
$ dpkg -l | grep xorg
```

Para arrancar com uma sessão SSH com possibilidade de janelas gráficas basta adicionar a opção `-X` ao comando `ssh`:

```
$ ssh -X pi@raspberrypi.local
```

**MacOS** Os sistemas MacOS não trazem um gestor de janelas X11 desde a versão 10.8 (OS X Mountain Lion). A melhor opção é o XQuartz, um projecto *open source* que mantém actualizada a aplicação `X11.app` que acompanhou o sistema operativo Mac OS X, desde a versão OS X 10.4 Tiger até à OS X 10.7 Lion.

O instalador pode ser obtido neste URL: `https://www.xquartz.org`.

Para arrancar com uma sessão SSH com possibilidade de janelas gráficas basta adicionar a opção `-X` ao comando `ssh`:

```
$ ssh -X pi@raspberrypi.local
```

**Windows** A configuração em Windows é ligeiramente mais trabalhosa, necessitando:

- do **Xming X Server for Windows** (`https://sourceforge.net/projects/xming/`), e
- (possivelmente) do cliente SSH **PuTTY** (`https://www.putty.org`).

Para arrancar com uma sessão SSH com possibilidade de janelas gráficas, deve seguir estes passos:

1. Arrancar o Xming.
2. Na "Linha de comandos"do Windows, experimente as seguintes linhas de comandos:

```
set DISPLAY=localhost:0.0
ssh -X pi@raspberrypi.local
```

3. Se o passo anterior não funcionar, então descarregue e arranque com o PuTTY.
   (a) Abrir "*Connection settings*"na barra da esquerda,
   (b) abrir "*SSH*",
   (c) abrir o menu de configurações "*X11*",
   (d) activar a opção "*Enable X11 forwarding*".
4. Escolher "*Session*"na barra da esquerda e efectuar a ligação SSH ao Raspberry Pi.

### 2.2.2 VNC

Do lado do Raspberry Pi, é necessário activar o servidor VNC, realizando os seguintes passos:

1. Entre na aplicação de configuração, através do comando

```
$ sudo raspi-config
```

2. Seleccione a opção "`3 Interface Options`".
3. Seleccione a opção "`P3 VNC`".
4. Seleccione a opção "`<Yes>`"para activar o servidor.
5. Saia da aplicação de configuração.

Agora, para se ligar remotamente ao Raspberry Pi, precisa de instalar um cliente VNC. Uma solução sem custos e disponível para Linux, Mac OS e Windows é o **RealVNC Viewer**, que pode ser obtido neste URL: `https://www.realvnc.com/en/connect/download/viewer/`. Para iniciar uma sessão remota, tem que inserir o nome ou IP do Raspberry Pi. Será pedido o nome de utilizador e senha para autenticação, devendo inserir as mesmas credenciais que utiliza para uma sessão SSH.

# 3 Obter um traço de execução

Implemente o seguinte programa que recebe um argumento da linha de comando que representa o número de iterações que um ciclo terá que realizar.

```c
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
  unsigned long int i, n;

  if(argc > 2) {
    printf("Usage: %s number-of-iterations\n\n", argv[0]);
    exit(EXIT_FAILURE);
  }
  else if(argc == 1) {
    n = 1000000;
    printf("Default number of iterations: 1000000\n");
  }
  else if(sscanf(argv[1], "%lu", &n)!= 1) {
    printf("Usage: %s number-of-iterations\n\n", argv[0]);
                exit(EXIT_FAILURE);
  }


  printf("[%d] Starting loop (%lu iterations)...\n", getpid(), n);
  for(i = 0; i < n; i++) {
    getpid();
  }
  printf("[%d] Finished loop (%lu iterations)!\n", getpid(), n);

  return 0;
}
```

Grave-o no ficheiro `ciclo.c` e compile-o sem qualquer optimização do compilador. Em seguida execute o programa, fornecendo-lhe alguns valores para o número de iterações do ciclo.

```
$ gcc ciclo.c -o ciclo
$ ./ciclo
Default number of iterations: 1000000
[2406] Starting loop (1000000 iterations)...
[2406] Finished loop (1000000 iterations)!
$ ./ciclo 5000
[2407] Starting loop (5000 iterations)...
[2407] Finished loop (5000 iterations)!
```

Agora vai correr o mesmo programa, mas com o `trace-cmd` a registar a actividade do kernel. O evento do kernel de interesse é o `sched_switch`, que ocorre quando há uma mudança de contexto. O comando `trace-cmd` tem que ser executado com permissões de administrador (`sudo`). A opção `-e` especifica qual o evento a monitorizar. Por fim, é indicado ao comando `trace-cmd` a linha de comando a ser observada: neste caso, executar o programa `ciclo` com 100000 iterações.

```
$ sudo trace-cmd record -e sched_switch ./ciclo 100000
```

No final da execução, terá sido criado o ficheiro de texto `trace.dat`. Este ficheiro tem toda a informação relevante para a sua interpretação, mas é muito longo para a nossa análise. Espreite o seu conteúdo com o comando

```
$ more trace.dat
```

Pode ver uma síntese do ficheiro com o comando

```
$ trace-cmd report
```

Esta síntese já nos permite compreender melhor a sequência de eventos.

Vamos agora ver o gráfico com a representação da sequência de eventos, com o `Kernelshark`. Arranque este utilitário com o comando

```
$ kernelshark trace.dat
```

O `Kernershark` executa dentro de uma janela gráfica, apresentando quatro gráficos temporais, um para cada núcleo de execução do Raspberry Pi. Com o rato pode fazer *zoom in*, clicando o rato e deslocando-o para a direita, e *zoom out* clicando com o rato e deslocando-o para a esquerda.

Também pode seleccionar processos específicos para apresentar as suas linhas temporais. Para tal, escolha no menu "`Plots`"a opção "`Tasks`"e seleccione os processos relevantes.

# 4 Visualizar concorrência entre processos

Experimente obter um traço de execução do programa `exercicio3` do guião anterior, para cada uma das seguintes situações:

1. atribuindo os dois processos ao mesmo núcleo de processamento, por exemplo

   ```
   $ sudo trace-cmd record -e sched_switch ./exercicio3 1 1
   100000000
   ```

2. atribuindo os dois processos a núcleos de processamento distintos, por exemplo

   ```
   $ sudo trace-cmd record -e sched_switch ./exercicio3 2 3
   100000000
   ```

Analise os resultados e relacione o que os gráficos revelam com os resultados da aula anterior, quando executou as mesmas linhas sob o comando `time`.