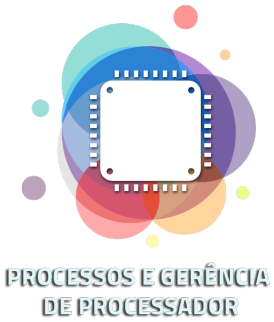

Tratamento de sinais em Linux

No final deste documento, você encontrará um pequeno programa que exemplifica a utilização da chamada de sistema signal(), que faz a instalação de uma rotina para capturar sinais e modificar o comportamento padrão destes sinais.

A ação padrão dos sinais SIGINT e SIGTERM é finalizar a execução do processo. Neste programa, vamos capturar esses sinais e executar nossa própria ação, evitando que o processo seja terminado quando receber os sinais

A linha 7 define uma função de nome captura() que será utilizada como função para tratamento dos sinais. Ela verifica o tipo de sinal recebido e coloca essa informação na saída padrão. Antes de terminar, ela mostra a mensagem “Pretendo continuar executando!!!” e o processo continua executando normalmente.

A linha 18 faz a instalação da função rotina para o tratamento dos sinais SIGINT e SIGTERM por intermédio da chamada de sistema signal(). A partir deste ponto, sempre que chegar qualquer um destes sinais, a execução do processo será desviada para a função captura().

Ao final, o processo entra em um loop infinito no qual, a cada segundo, ele emite uma mensagem afirmando que está executando normalmente.

Ao ser colocado em execução, o processo começa a exibir uma série de mensagens informando que está vivo.

Para testar o sinal SIGINT, utilizaremos o próprio terminal no qual o processo está executando. Ao pressionar simultaneamente as teclas <Ctrl>+<C>, o sinal SIGINT é enviado ao processo, que faz com que ele seja encerrado. Como nosso programa faz a captura deste sinal, no lugar de ser encerrado, a execução do processo será desviada para a rotina captura, que verifica o tipo do sinal, mostra essa informação e informa que continuará executando.

O mesmo teste pode ser realizado para verificar a captura do sinal SIGTERM, mas, para enviar esse sinal, utilizaremos o comando kill. Esse comando é utilizado para enviar sinais a processos, informando como parâmetro o sinal que deve ser enviado. Se não for informado o tipo de sinal, por padrão é enviado o sinal SIGTERM.

Primeiramente, precisamos descobrir o PID do processo que queremos terminar. Para isso, é necessário abrir outro terminal e executar o comando:

```
ps -a
```

Verificamos o número do processo (PID) e enviamos o sinal para ele. Se o PID for, por exemplo, 8053, enviamos o sinal SIGTERM por intermédio do comando:

```
kill 8053
```

De forma semelhante ao explicado para o SIGINT, o sinal SIGTERM será capturado e uma mensagem será exibida na tela.

Não importa o tipo de sinal enviado (SIGINT ou SIGTERM), o processo mostrará uma mensagem informando o sinal recebido e continuará executando e informando a cada segundo que está vivo.

Uma forma de terminar a execução do processo é por intermédio do sinal SIGKILL, que não pode ser capturado. Para enviar o sinal SIGKILL, podemos utilizar o comando “kill -s SIGKILL <PID>” ou o comando “kill -9 <PID>”, onde <PID> será o PID do processo a ser encerrado. Para o exemplo anterior, podemos encerrar o processo por intermédio do comando:

```
Kill -9 8053
```

EXEMPLO PARA TRATAMENTO DE SINAIS

```
1   #include <signal.h>
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <unistd.h>
5
6   // Rotina para tratamento do sinal capturado
7   static void captura(int sinal) {
8     if (sinal == SIGINT)
9       printf("Recebido o sinal SIGINT.\n");
10    if (sinal == SIGTERM)
11      printf("Recebido o sinal SIGTERM.\n");
12    printf("Pretendo continuar executando!!!\n");
13  }
14
15  int main(){
16    int i;
17    // Captura os sinais SIGINT e SIGTERM
18    if ((signal(SIGINT, captura) == SIG_ERR) || (signal(SIGTERM, captura) == SIG_ERR)) {
19      printf("Erro ao instalar o tratador do sinais.\n");
20      exit(1);
21    }
22    i = 1;
```


```
23      while (1) { // Loop infinito
24        printf("Estou vivo [%d].\n", i++);
25        usleep(1000000);
26      }
27    }
```