

# Sistemas Operacionais

Náthaly Martins De Sá

## **Tópico 01**

1. Uso da chamada de sistema fork.
  - 1.1

```
ufabc@ufabc-OptiPlex-9010: ~/Sistemas/MCTA026_lab1_codigo
(base) ufabcd@ufabc-OptiPlex-9010:~$ ls
anaconda3      Documentos    eclipse-workspace  Música      Sistemas
Arduino        Downloads    Imagens           Público     snap
'Área de Trabalho' eclipse      Modelos           R           Vídeos
(base) ufabcd@ufabc-OptiPlex-9010:~$ cd Sistemas/
(base) ufabcd@ufabc-OptiPlex-9010:~/Sistemas$ ls
MCTA026_lab1_codigo  MCTA026_lab1_codigo.zip
(base) ufabcd@ufabc-OptiPlex-9010:~/Sistemas$ cd MCTA026_lab1_codigo/
(base) ufabcd@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ ls
prog1.c prog3.c prog5.c prog7.c prog9c.c
prog2.c prog4.c prog6.c prog8.c prog9p.c
(base) ufabcd@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ gcc -o prog1 pr
og1.c
prog1.c: In function 'main':
prog1.c:10:15: warning: implicit declaration of function 'fork' [-Wimplicit-fun
ction-declaration]
   10 |         pid = fork();
      |
prog1.c:13:72: warning: implicit declaration of function 'getpid' [-Wimplicit-fu
nction-declaration]
   13 |         printf("Filho (PID %d) vai dormir por 10 segundos.\n", getpid());
      |
prog1.c:14:17: warning: implicit declaration of function 'sleep' [-Wimplicit-fun
ction-declaration]
   14 |         sleep(10);
      |
(base) ufabcd@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ ./prog1
Pai com filho com PID 6040.
Aguardando término do filho!
Filho (PID 6040) vai dormir por 10 segundos.
Filho terminando a execução.
Pai viu filho com PID 6040 terminar.
Filho terminou com status = 0.
Pai terminando a execução.
(base) ufabcd@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$
```

Primeiramente foi utilizado o comando **ls** para listar arquivos ou diretórios no Linux e em outros sistemas operacionais baseados no Unix, a partir disso com o comando **cd** foi acessado a pasta **Sistemas** e houve a compilação do Programa 1 (prog1.c – linguagem c) com o comando **gcc**, automaticamente programa passou a ser chamado de **prog1**, graças ao comando **-o** que

nomeia o arquivo de saída. Logo em seguida executamos ( *./prog1* ) o Programa 1.

## 1.2

```
ufabc@ufabc-OptiPlex-9010: ~/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Sistema... x  ufacb@ufabc-OptiPlex-9010: ~/Sistema... x v
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ kill -SIGINT 6217
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$
```

```
ufabc@ufabc-OptiPlex-9010: ~/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Sistema... x  ufacb@ufabc-OptiPlex-9010: ~/Sistema... x v
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ ./prog1
Pai com filho com PID 6217.
Aguardando término do filho!
Filho (PID 6217) vai dormir por 10 segundos.
Pai viu filho com PID 6217 terminar.
Filho foi morto por sinal 2.
Pai terminando a execução.
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$
```

No caso da utilização do comando **kill -SIGINT** com o PID informado na 1º parte da saída da execução, o comando permite que se encerre um processo utilizando seu ID específico (PID 6217) e com o sinal **SIGINT** o processo se encerra sendo interrompido, gerando assim a 2º parte da saída do Programa 1 com um sinal de erro 2 que equivale ao sinal **SIGINT**.

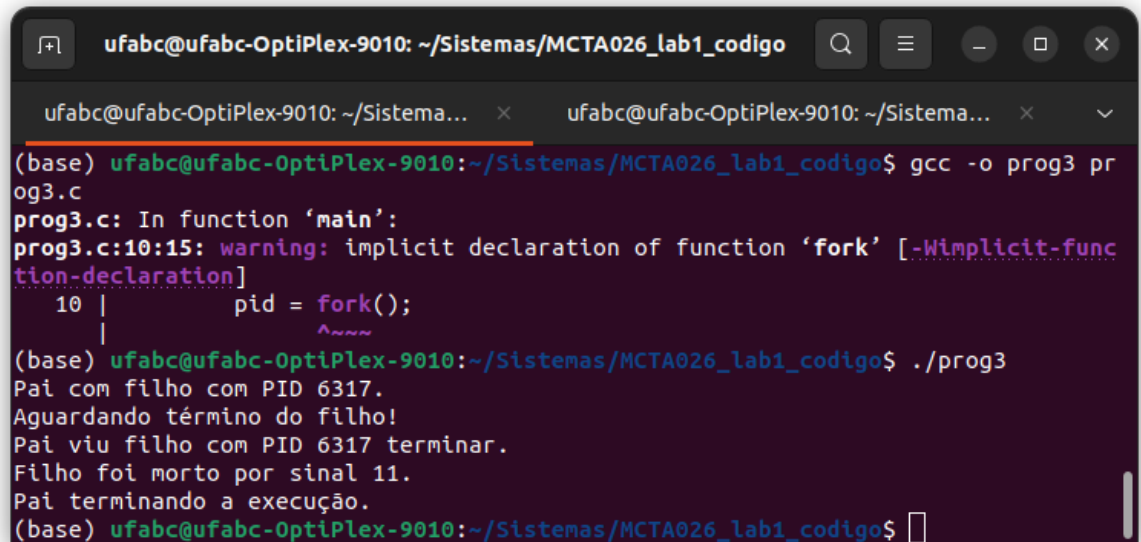
## 2. Sinais como condições de erro.

### 2.1

```
ufabc@ufabc-OptiPlex-9010: ~/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Sistema... x  ufacb@ufabc-OptiPlex-9010: ~/Sistema... x v
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ gcc -o prog2 prog2.c
prog2.c: In function 'main':
prog2.c:10:15: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
   10 |         pid = fork();
      |               ^~~~~
prog2.c:14:26: warning: division by zero [-Wdiv-by-zero]
   14 |         int x = 2/0; //Divisão por zero.
      |                   ^
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ ./prog2
Pai com filho com PID 6372.
Aguardando término do filho!
Pai viu filho com PID 6372 terminar.
Filho foi morto por sinal 8.
Pai terminando a execução.
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$
```

Ao executar o Programa 2 o processo filho é terminado, pois é detectado um erro, sendo assim ao executar o programa a saída envia um sinal (Sinal 8) para informar qual foi o erro naquela execução.

## 2.2

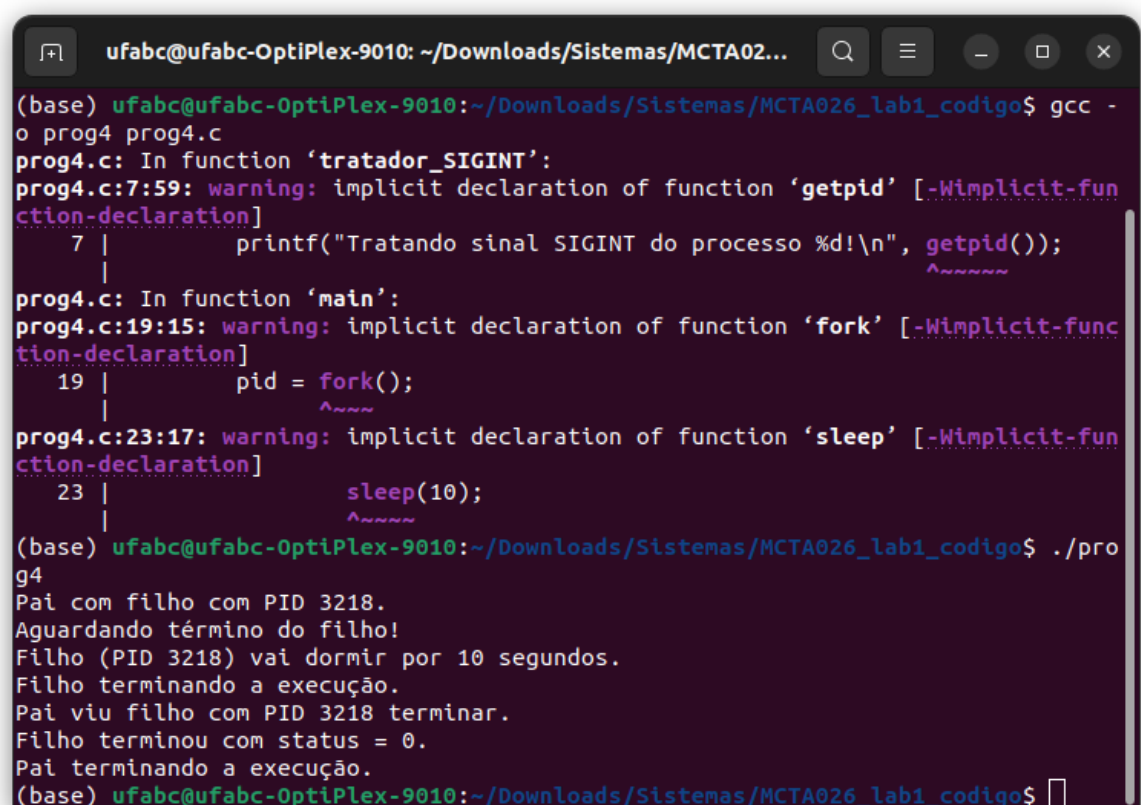


```
ufabc@ufabc-OptiPlex-9010: ~/Sistemas/MCTA026_lab1_codigo
(base) ufabcd@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ gcc -o prog3 prog3.c
prog3.c: In function 'main':
prog3.c:10:15: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
   10 |         pid = fork();
      |               ^~~~~
(base) ufabcd@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ ./prog3
Pai com filho com PID 6317.
Aguardando término do filho!
Pai viu filho com PID 6317 terminar.
Filho foi morto por sinal 11.
Pai terminando a execução.
(base) ufabcd@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$
```

Assim como no Programa anterior, ao executar o Programa 3 o processo filho é terminado, graças a um erro de execução. Neste caso é detectado um erro diferente do anterior, logo também é informado um sinal diferente (Sinal 11), sendo assim cada número equivale a um erro específico.

## 3. Como tratar um sinal?

### 3.1



```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ gcc -o prog4 prog4.c
prog4.c: In function 'tratador_SIGINT':
prog4.c:7:59: warning: implicit declaration of function 'getpid' [-Wimplicit-function-declaration]
    7 |         printf("Tratando sinal SIGINT do processo %d!\n", getpid());
      |                                                           ^~~~~~
prog4.c: In function 'main':
prog4.c:19:15: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
   19 |         pid = fork();
      |               ^~~~~
prog4.c:23:17: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
   23 |         sleep(10);
      |         ^~~~~~
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog4
Pai com filho com PID 3218.
Aguardando término do filho!
Filho (PID 3218) vai dormir por 10 segundos.
Filho terminando a execução.
Pai viu filho com PID 3218 terminar.
Filho terminou com status = 0.
Pai terminando a execução.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

```
ufabc@ufabc-OptiPlex-9010: ~/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Sistema... x  ufacb@ufabc-OptiPlex-9010: ~/Sistema... x
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ kill -SIGINT 6486
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$
```

```
ufabc@ufabc-OptiPlex-9010: ~/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Sistema... x  ufacb@ufabc-OptiPlex-9010: ~/Sistema... x
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$ ./prog4
Pai com filho com PID 6486.
Aguardando término do filho!
Filho (PID 6486) vai dormir por 10 segundos.
Tratando sinal SIGINT do processo 6486!
Na verdade, acho que não vou fazer nada! :-)
Filho terminando a execução.
Pai viu filho com PID 6486 terminar.
Filho terminou com status = 0.
Pai terminando a execução.
(base) ufacb@ufabc-OptiPlex-9010:~/Sistemas/MCTA026_lab1_codigo$
```

É possível observar que ao enviar o sinal **SIGINT**, o processo informado (PID) é tratado, tomando assim um rumo diferente seguindo seu programa. No caso do Programa 4, é acrescentada mais uma resposta ao processo filho, incluindo “Na verdade, acho que não vou fazer nada! :-)” na saída diferente da execução sem o sinal **SIGINT**. Sendo assim, para tratar um sinal, basta criar um tratador específico para aquele sinal no próprio programa e em uma segunda execução enviar esse sinal (que equivale ao número que consta na saída da execução, logo o número 11 (Sinal 11) equivale ao sinal **SIGINT**), tratando assim o erro e dando continuidade a execução.

### 3.2

```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$ gcc -o prog4 prog4.c
prog4.c: In function 'tratador_SIGSEGV':
prog4.c:7:60: warning: implicit declaration of function 'getpid' [-Wimplicit-function-declaration]
    7 |         printf("Tratando sinal SIGSEGV do processo %d!\n", getpid());
      |
prog4.c: In function 'main':
prog4.c:19:15: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
    19 |         pid = fork();
      |
prog4.c:23:17: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    23 |         sleep(10);
      |
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog4
Pai com filho com PID 5563.
Aguardando término do filho!
Filho (PID 5563) vai dormir por 10 segundos.
Tratando sinal SIGSEGV do processo 5563!
Na verdade, acho que não vou fazer nada! :-)
Filho terminando a execução.
Pai viu filho com PID 5563 terminar.
Filho terminou com status = 0.
Pai terminando a execução.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$ kill -SIGSEGV 5563
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

Como observado, foi alterado o tratador do Programa 4 (**SIGINT** para **SIGSEGV**) de forma a tratar o mesmo erro apresentado na execução do Programa 3 (Sinal 11) e ao citá-lo de forma correta em uma segunda execução, como apresentado acima, é possível tratar o sinal e dar continuidade à execução normalmente.

4. Como mascarar um sinal?
- 4.1 Aplicando sinal **SIGQUIT**:

```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x  ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x
(base) ufabcd@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$ kill -SIGQUIT 3875
(base) ufabcd@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x  ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x
(base) ufabcd@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$ gcc -o prog5 prog5.c
prog5.c: In function 'tratador_SIGINT':
prog5.c:7:59: warning: implicit declaration of function 'getpid' [-Wimplicit-function-declaration]
    7 |         printf("Tratando sinal SIGINT do processo %d!\n", getpid());
      |                                                 ~~~~~
prog5.c: In function 'main':
prog5.c:25:15: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
    25 |         pid = fork();
      |             ~~~~
prog5.c:29:17: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    29 |         sleep(10);
      |         ~~~~~
(base) ufabcd@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog5
Pai com filho com PID 3875.
Aguardando término do filho!
Filho (PID 3875) vai dormir por 10 segundos.
Filho terminando a execução.
Pai viu filho com PID 3875 terminar.
Filho terminou com status = 0.
Pai terminando a execução.
(base) ufabcd@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

Aplicando sinal **SIGTERM**:

```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x  ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x
(base) ufabcd@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$ kill -SIGTERM 3834
(base) ufabcd@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x  ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x
(base) ufabcd@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog5
Pai com filho com PID 3834.
Aguardando término do filho!
Filho (PID 3834) vai dormir por 10 segundos.
Pai viu filho com PID 3834 terminar.
Filho foi morto por sinal 15.
Pai terminando a execução.
(base) ufabcd@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

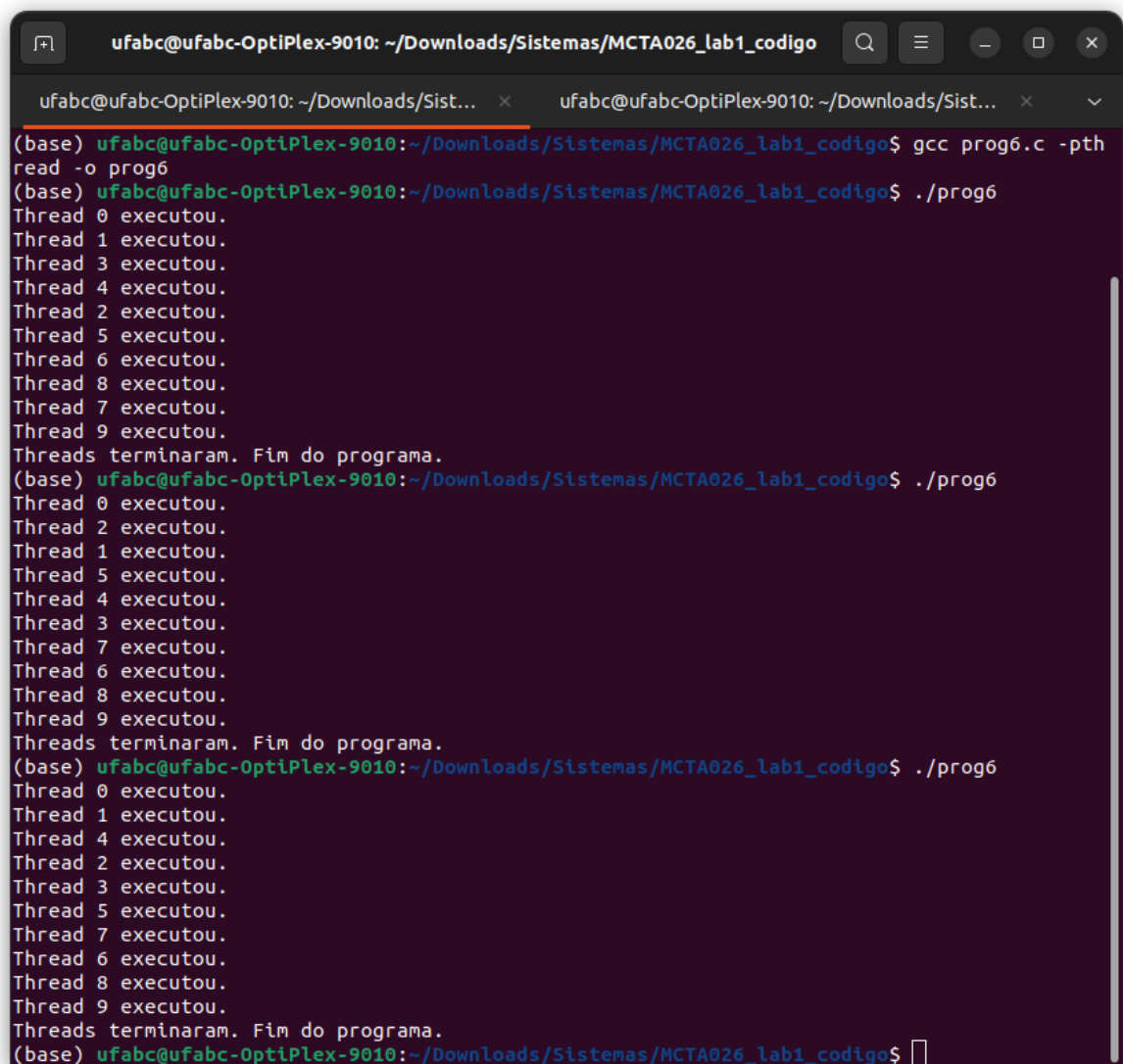


Com sinal **SIGQUIT** há o abandono da execução, enquanto com o sinal **SIGTERM** há a terminação ordenada do processo. Desta forma, na primeira execução não houve sinal de erro, graças a máscara, que se trata do bloqueio de sinais incluídos no programa (no Programa 5 foi instalado máscara para **SIGQUIT**), logo o processo filho bloqueou o sinal enviado em segundo plano e se encerrou com o status 0. Enquanto que na segunda execução, com o sinal **SIGTERM**, houve a obrigatoriedade de término do processo, porém de forma normal seguindo o programa e como o sinal não foi mascarado (bloqueado temporariamente), ele foi exposto na saída da execução.

**4.2** Não é possível tratar ou mascarar os sinais SIGKILL e SIGSTOP, pois sempre é executada a ação default, logo eles não podem ser capturados e nem ignorados.

## 5. Criação de threads.

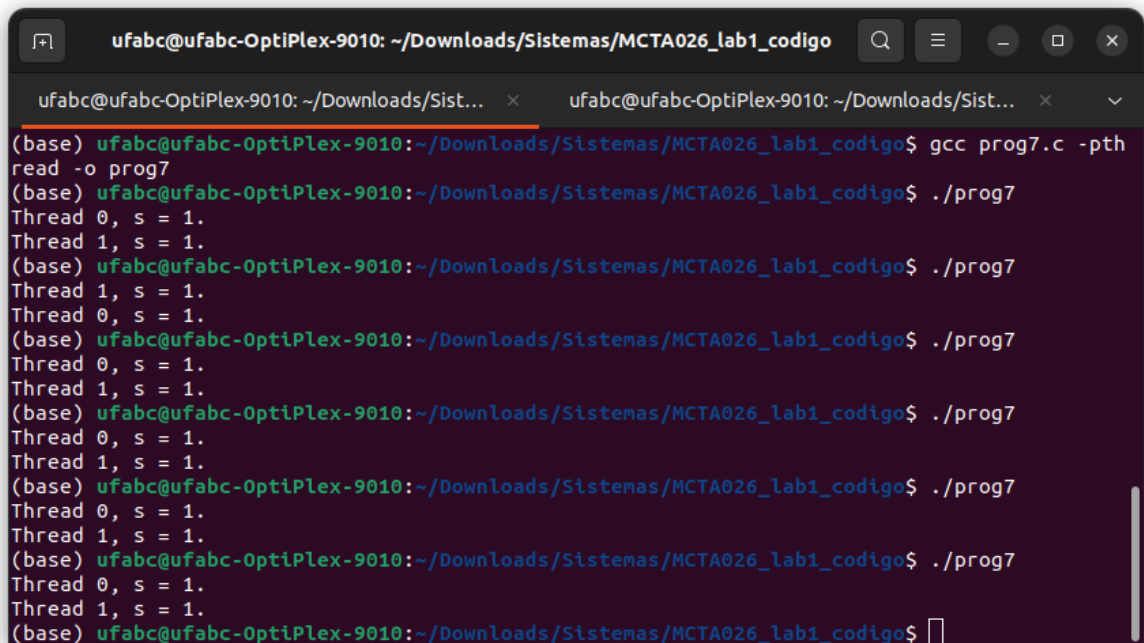
### 5.1



```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x  ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ gcc prog6.c -pth
read -o prog6
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog6
Thread 0 executou.
Thread 1 executou.
Thread 3 executou.
Thread 4 executou.
Thread 2 executou.
Thread 5 executou.
Thread 6 executou.
Thread 8 executou.
Thread 7 executou.
Thread 9 executou.
Threads terminaram. Fim do programa.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog6
Thread 0 executou.
Thread 2 executou.
Thread 1 executou.
Thread 5 executou.
Thread 4 executou.
Thread 3 executou.
Thread 7 executou.
Thread 6 executou.
Thread 8 executou.
Thread 9 executou.
Threads terminaram. Fim do programa.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog6
Thread 0 executou.
Thread 1 executou.
Thread 4 executou.
Thread 2 executou.
Thread 3 executou.
Thread 5 executou.
Thread 7 executou.
Thread 6 executou.
Thread 8 executou.
Thread 9 executou.
Threads terminaram. Fim do programa.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

Execução alternada até finalizar o número de Thread's informado no programa, ou seja a partir de um contador o programa recebe o identificador da Thread, logo após imprime esse identificador criado e finaliza ao completar o número de Thread's informado.

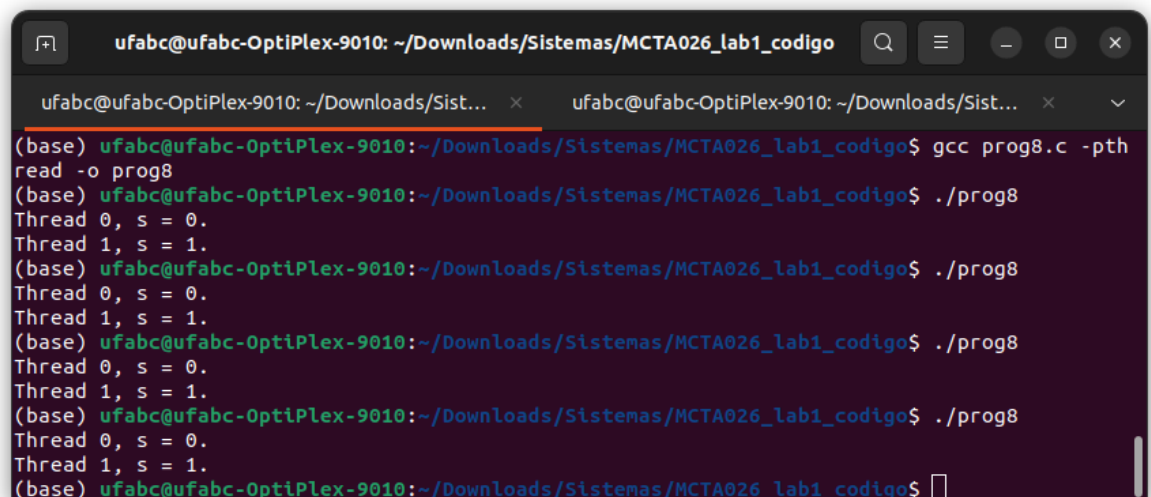
## 5.2



```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x  ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ gcc prog7.c -pth
read -o prog7
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog7
Thread 0, s = 1.
Thread 1, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog7
Thread 1, s = 1.
Thread 0, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog7
Thread 0, s = 1.
Thread 1, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog7
Thread 0, s = 1.
Thread 1, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog7
Thread 0, s = 1.
Thread 1, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog7
Thread 0, s = 1.
Thread 1, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

## 6. Implementando a exclusão mútua.

### 6.1



```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA026_lab1_codigo
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x  ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sist... x
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ gcc prog8.c -pth
read -o prog8
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog8
Thread 0, s = 0.
Thread 1, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog8
Thread 0, s = 0.
Thread 1, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog8
Thread 0, s = 0.
Thread 1, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prog8
Thread 0, s = 0.
Thread 1, s = 1.
(base) ufabcd@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$
```

## 7. Produtor-consumidor com troca de mensagens.

### 7.1



```
ufabc@ufabc-OptiPlex-9010: ~/Downloads/Sistemas/MCTA02...  
(base) ufacb@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prod  
erro na criacao de mailbox: Success  
o resultado eh: 0  
(base) ufacb@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./cons  
s  
Erro na criacao de mailbox: Success  
o tipo da mensagem eh: 1  
o conteudo da mensagem eh: Esta eh uma mensagem de teste  
(base) ufacb@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./cons  
s  
Erro na criacao de mailbox: Success  
o tipo da mensagem eh: 1  
o conteudo da mensagem eh: Esta eh uma mensagem de teste  
(base) ufacb@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prod  
erro na criacao de mailbox: Success  
o resultado eh: 0  
(base) ufacb@ufabc-OptiPlex-9010:~/Downloads/Sistemas/MCTA026_lab1_codigo$ ./prod  
d cons  
erro na criacao de mailbox: Success  
o resultado eh: 0
```