RELATÓRIO DE LABORATÓRIO 2

SISTEMAS OPERACIONAIS

VICTOR MONEGO

ENGENHARIA ELETRÔNICA – UTFPR

29 DE MARÇO DE 2024

1.Introdução Geral

O seguinte relatório diz respeito ao Laboratório 02 de Sistemas Operacionais, focado na lógica de criação de processos em Unix.

Para a realização das análises a seguir, foi utilizada a plataforma WSL(Ubuntu) no Windows 11, e os códigos foram comentados e modificados usando o programa Notepad++.

2. Exercício 01: “fork.c”

O exerpto da figura 01 abaixo apresenta o código “fork.c” utilizado no exercício 1. Note que o código não é de autoria própria e as únicas alterações feitas foram os comentários a respeito dele.

```c
/*
Criação de processos em UNIX.
Compilar com gcc -Wall fork.c -o fork
Carlos Maziero, DINF/UFPR 2020
*/
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
int main ()
{
  int retval ;
  printf ("Ola, sou o processo %5d\n", getpid()) ;
  retval = fork () ;
  printf ("[retval: %5d] sou %5d, filho de %5d\n", retval, getpid(), getppid()) ;
  if ( retval < 0 )
  {
    perror ("Erro") ;
    exit (1) ;
  }
  else
    if ( retval > 0 )
      wait (0) ;
    else
      sleep (5) ;
  printf ("Tchau de %5d!\n", getpid()) ;
  exit (0) ;
}
```

Figura 1: Código fork.c

Comentários:

- 13: declarando a variável retval, que receberá o retorno do fork()
- 14: o comando getpid() é utilizado para apresentar o Process ID do processo pai
- 15: a variável retval recebe o retorno do fork()
- 16: requisitado que o processo se identifique, declarando o retorno do fork, apresentando seu próprio pid e também o pid do seu processo pai. na prática, retval indicará qual o id do próximo processo a se apresentar e ,nessa situação, como ambos os processos irão responder ao comando o processo pai se apresenta, e logo após o processo filho também se apresenta. Podemos perceber que o processo filho está se apresentando, pois ele irá apresentar um retval de zero. Caso contrário, seu retval sendo maior que 0 irá apresentar o pid do processo filho que irá se apresentar posteriormente.
- 17: erro no fork()
- 19: nesse caso, imprimir a mensagem de erro, e retornar 1 para a função main, indicando interrupção.
- 23: este trecho está sendo executado apenas pelo processo pai
- 24: nesse caso, não esperar e imediatamente executar o comando final, após o processo filho.
- 25: por último, este trecho será executado apenas pelo processo filho.
- 26: o processo filho dorme por 5 segundos.
- 27: o processo irá se "despedir". Seu processo filho irá se despedir antes e, devido ao tempo de sleep, não será imediatamente.
- 28: a função retorna 0 indicando término com êxito

A imagem 02 abaixo indica o diagrama de tempo do programa acima.

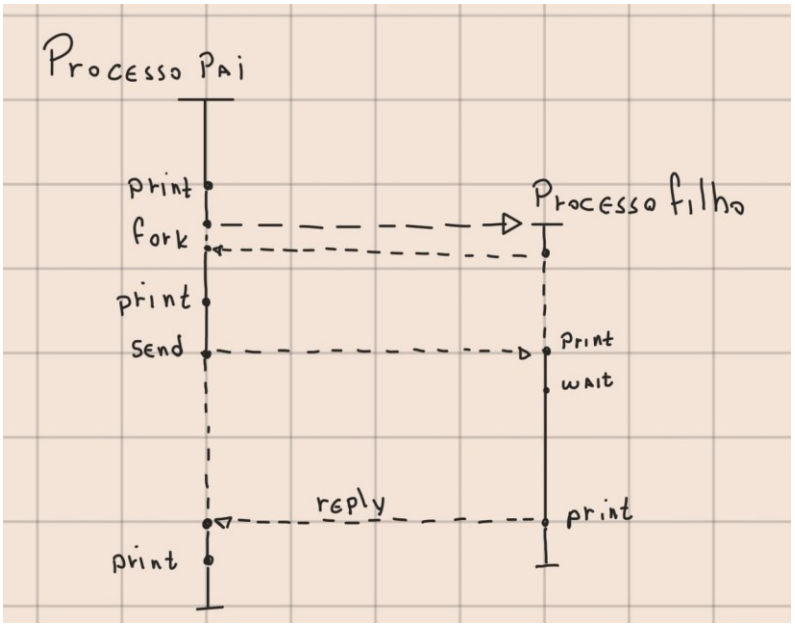

Figura 2: Diagrama de Tempo fork.c

E por fim, a imagem 03 indica como funciona o programa quando executado no prompt da WSL.

```
linuxtor@victornote:/mnt/c/Users/victo/OneDrive/UTFPR/2024.1/Sistemas_Operacionais/Práticas/prática_2$ ./fork
Ola, sou o processo  1414
[retval:  1415] sou  1414, filho de   367
[retval:     0] sou  1415, filho de  1414
Tchau de  1415!
Tchau de  1414!
```

Figura 3: Execução fork.c

3. Exercício 02: “fork-execve.c”

O exerpto da figura 04 abaixo apresenta o código “fork-execve.c” utilizado no exercício 2. Note que o código não é de autoria própria. As alterações feitas foram os comentários e o argumento da função execve().

```c
/*
Criação de processos em UNIX, com execução de outro binário

Compilar com gcc -Wall fork-execve.c -o fork-execve

Carlos Maziero, DINF/UFPR 2020
*/

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

int main (int argc, char *argv[], char *envp[])
{
  int retval ;

  printf ("Ola, sou o processo %5d\n", getpid()) ;
  retval = fork () ;
  printf ("[retval: %5d] sou %5d, filho de %5d\n", retval, getpid(), getppid()) ;

  if ( retval < 0 )
  {
    perror ("Erro: ") ;
    exit (1) ;
  }
  else
    if ( retval > 0 )
      wait (0) ;
    else
    {
      execve ("pratica1_executable", argv, envp) ;
      perror ("Erro") ;
    }

  printf ("Tchau de %5d!\n", getpid()) ;
  exit (0) ;
}
```

Figura 4: fork-execve.c

Comentários:

- 17: declaramos a variável de retorno do fork()
- 19: o processo pai imprime a mensagem
- 20: criação do processo filho
- 21: processo filho se identifica
- 23: erro no fork ()
- 25: imprime mensagem de erro no prompt
- 26: retorna com 1 indicando interrupção
- 29: sou o processo pai
- 30: aguarda para responder imediatamente após o filho retornar a sua função
- 31: sou o processo filho
- 33: executa um programa
- 34: imprime mensagem de erro no prompt caso o execve() retorne, o que significaria um erro.
- 37: se despede. Logo após a despedida do processo filho, o processo pai se despede
- 38: retorna 0, fim do programa

A imagem 05 abaixo indica o diagrama de tempo do programa acima.

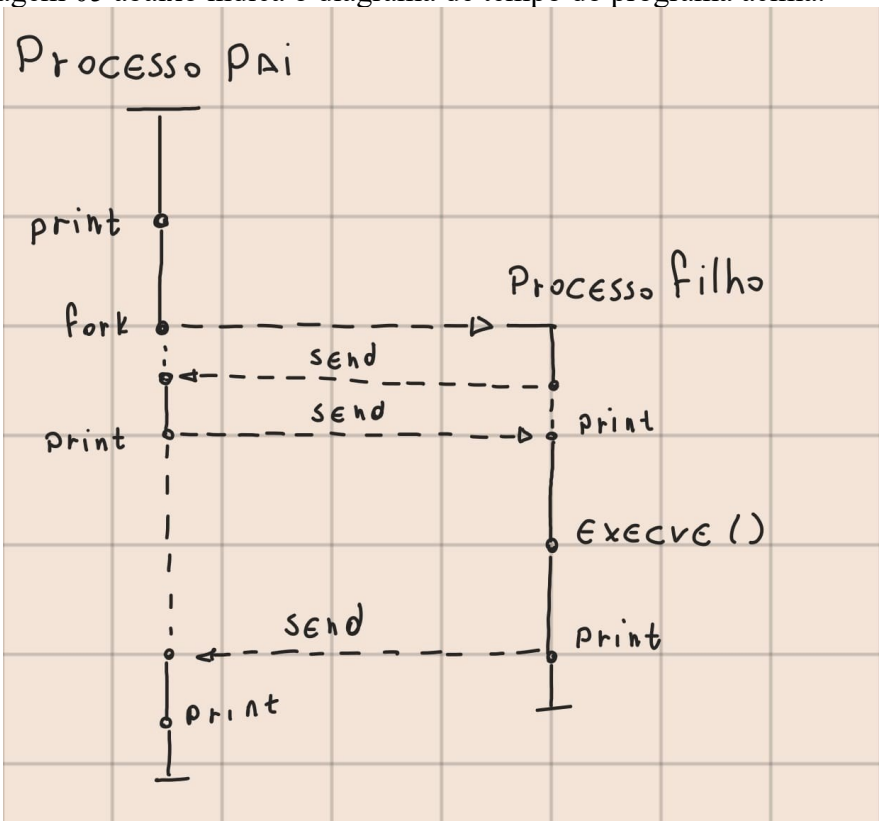

Figura 5: Diagrama de tempo fork-execve.c

E por fim, a imagem 06 indica como funciona o programa quando executado no prompt da WSL.

```
linuxtor@victornote:/mnt/c/Users/victo/OneDrive/UTFPR/2024.1/Sistemas_Operacionais/Práticas/prática_2$ ./fork-execve
Ola, sou o processo   783
[retval:   784] sou   783, filho de   367
[retval:     0] sou   784, filho de   783

Digite o valor float a ser considerado...
-2003.456
O valor absoluto do seu input é: 2003.456000
Tchau de   783!
linuxtor@victornote:/mnt/c/Users/victo/OneDrive/UTFPR/2024.1/Sistemas_Operacionais/Práticas/prática_2$
```

Figura 6: Execução fork-execve.c

Caso o executável do programa requisitado não existir, não estiver no mesmo diretório que o código, ou o caminho especificado, estiver errado, o prompt retornará o erro ilustrado na figura 07.

```
linuxtor@victornote:/mnt/c/Users/victo/OneDrive/UTFPR/2024.1/Sistemas_Operacionais/Práticas/prática_2$ gcc -Wall fork-ex
ecve.c -o fork-execve
linuxtor@victornote:/mnt/c/Users/victo/OneDrive/UTFPR/2024.1/Sistemas_Operacionais/Práticas/prática_2$ ./fork-execve
Ola, sou o processo  1159
[retval:  1160] sou  1159, filho de   367
[retval:     0] sou  1160, filho de  1159
Erro: No such file or directory
Tchau de  1160!
Tchau de  1159!
```

Figura 7:Erro de diretório no fork-execve.c

4. Exercício 03: “fork-print.c”

O exerpto da figura 08 abaixo apresenta o código “fork-print.c” utilizado no exercício 3. Note que o código não é de autoria própria. As alterações feitas foram os comentários e o incremento de uma variável local.

```c
/*
Criação de processos em UNIX, com impressão de valores de variável.

Compilar com gcc -Wall fork-print.c -o fork-print

Carlos Maziero, DINF/UFPR 2020
*/

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

int main ()
{
  int retval, x ;

  x = 0 ;
  retval = fork () ;
  printf ("No processo %5d x vale %d\n", getpid(), x) ;
  x++ ;

  if ( retval < 0 )
  {
    perror ("Erro") ;
    exit (1) ;
  }
  else
    if ( retval > 0 )
    {
      x = 0 ;
      wait (0) ;
    }
    else
    {
      x++ ;
      sleep (5) ;
    }

  printf ("No processo %5d x vale %d\n", getpid(), x) ;
  exit (0) ;
}
```

Figura 8: Código fork-print.c

Comentários:

- 17: declara a variável que retorna o valor do fork()
- 19: declara a variavel x como 0
- 20: fork()
- 21: o processo pai e filho se identificam e apresentam o valor de x atual
- 22: x se incrementa em 1
- 24: erro no fork()
- 26: imprime mensagem de erro
- 27: retorna 1, interrupção
- 30: sou o processo pai
- 32: reinicia x para 0
- 33: o pai segue a próxima instrução imediatamente após o filho seguir
- 35: sou o processo filho
- 37: x se incrementa em 1
- 38: o filho aguarda 5 segundos antes de seguir a proxima instrução
- 39: pai e o filho apresentam os seus respectivos valores de x, sendo que o pai retorna 0 e o filho retorna 2

A imagem 09 abaixo indica o diagrama de tempo do programa acima.

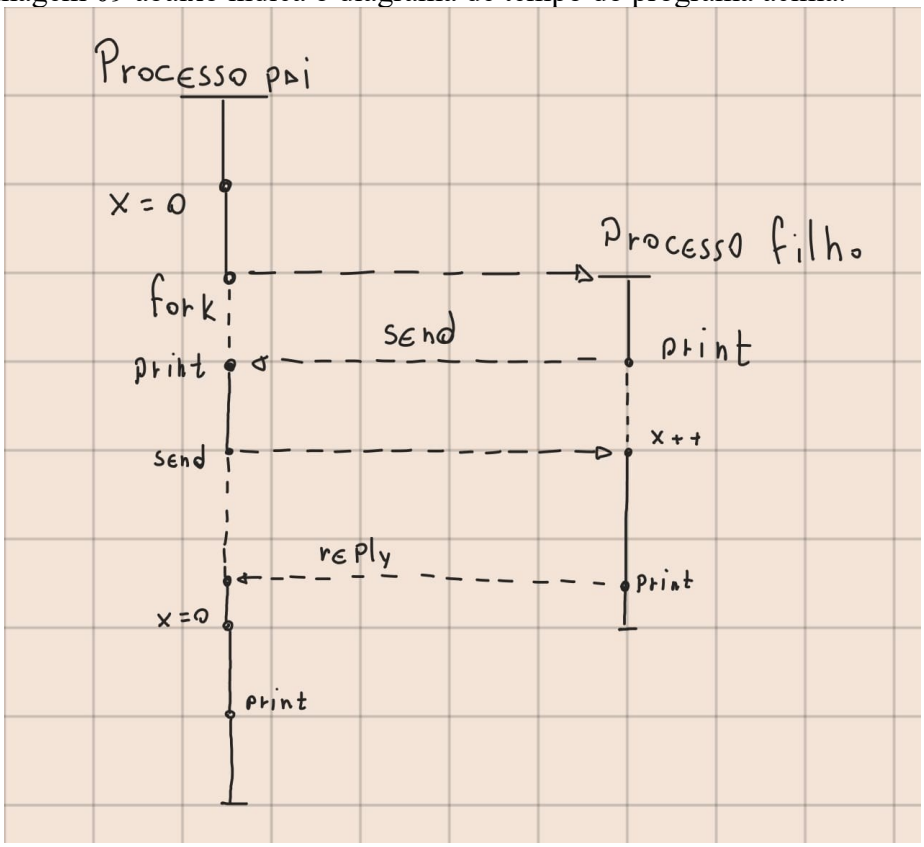

Figura 9: Diagrama de tempo fork-print.c

E por fim, a imagem 10 indica como funciona o programa quando executado no prompt da WSL.

```
linuxtor@victornote:/mnt/c/Users/victo/OneDrive/UTFPR/2024.1/Sistemas_Operacionais/Práticas/prática_2$ gcc -Wall fork-pr
int.c -o fork-print
linuxtor@victornote:/mnt/c/Users/victo/OneDrive/UTFPR/2024.1/Sistemas_Operacionais/Práticas/prática_2$ ./fork-print
No processo   763 x vale 0
No processo   764 x vale 0
No processo   764 x vale 2
No processo   763 x vale 0
linuxtor@victornote:/mnt/c/Users/victo/OneDrive/UTFPR/2024.1/Sistemas_Operacionais/Práticas/prática_2$
```

Figura 10: Execução fork-print.c

Observações: Os códigos apresentados no relatório não são autorais, e são de autoria do Prof. Carlos Maziero, vide a seção “Referências”.

Referências:

- MAZIERO, C. **Criação de Processos em Unix**. Disponível em: https://wiki.inf.ufpr.br/maziero/doku.php?id=so:criacao_de_processos. Acesso em: Março de 2024