

**IFPB**

**Disciplina: Sistemas Operacionais**

**Docente: Henrique Do Nascimento Cunha**

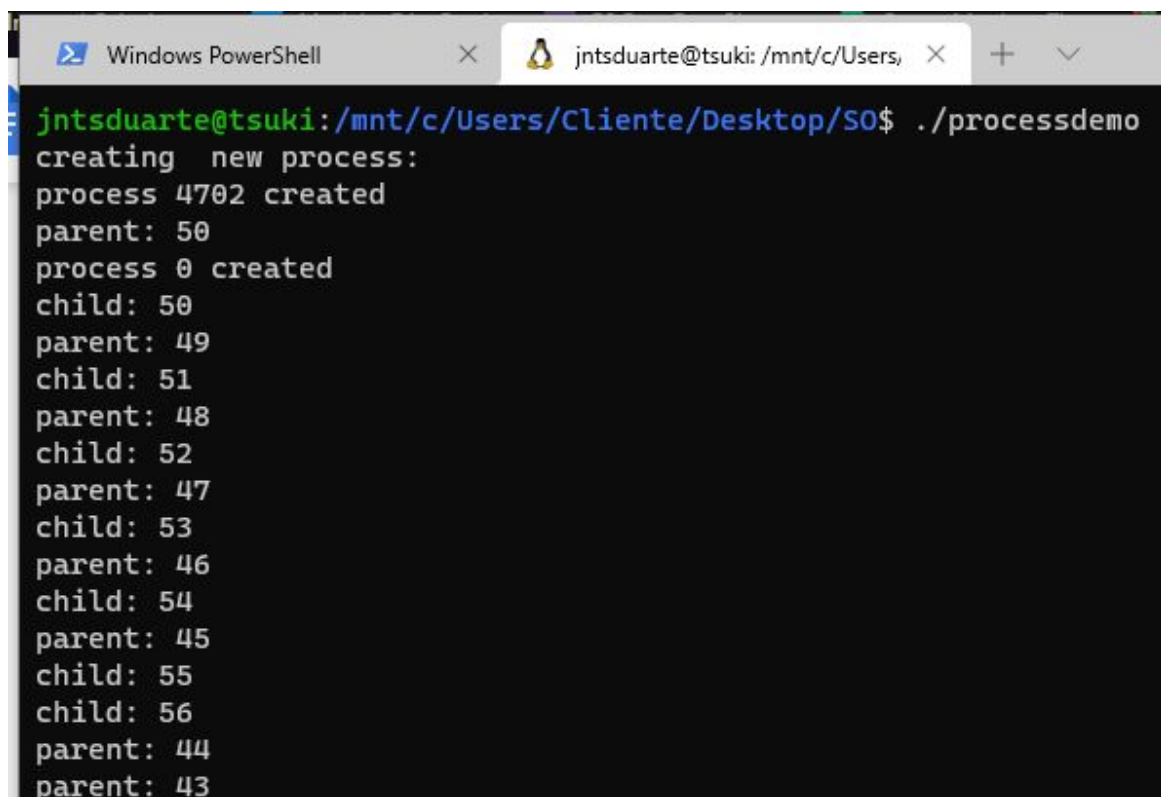
**Discente: Jonatas da Silva Duarte**

## **Roteiro 1 - Processos**

### ***Passo 1 ao 7:***

- Fiz o roteiro usando WSL, para rodar o Ubuntu na minha máquina Windows;
- Tive alguns problemas para compilar o código do arquivo “processdemo.c”, então depois de algumas pesquisas consegui resolver e compilar normalmente, apenas adicionei algumas libs e ‘void’ ao *main* (mas que não alteravam em rodar o programa);
- Para a análise do código e mudanças usei o notepad.

### ***Passo 8 ao 15: Descreva a saída e explique por que ela é dessa forma.***



```
jntsduarte@tsuki:/mnt/c/Users/Cliente/Desktop/SO$ ./processdemo
creating new process:
process 4702 created
parent: 50
process 0 created
child: 50
parent: 49
child: 51
parent: 48
child: 52
parent: 47
child: 53
parent: 46
child: 54
parent: 45
child: 55
child: 56
parent: 44
parent: 43
```

- Assim como vimos no print, os processos, pai e filho, são criados e ambos iniciados com um valor de 50, sendo que o filho possui uma contagem crescente e o pai decrescente;
- Os valores mostrados no terminal estão loop infinito (que é feito pela função 'adjustX');
- Os valores iniciais são iguais graças ao 'Fork' dado no *main*, sendo assim o filho é uma cópia exata (pelo menos inicialmente) do processo pai;
- A mudança dos valores ocorrem graças à condição (if) que existe no código, tendo execuções diferentes caso seja o pai ou o filho (identificados pela variável 'c'):

```
void main(){
    int c;
    srand(time(NULL));
    printf("creating new process:\n");
    c = fork();
    printf("process %i created\n", c);
    if (c==0)
        adjustX("child", 1);    /* child process */
    else
        adjustX("parent", -1); /* parent process */
}
```

- Executando o comando ps xl, temos:

0	1000	4704	8	20	0	2488	716	-	R+	pts/0	1:09	./processdemo
1	1000	4705	4704	20	0	2488	84	-	R+	pts/0	1:09	./processdemo

- PID é o número de identificação de um processo, enquanto o PPID é especificamente o do processo-pai (são mostrados na terceira e quarta coluna, respectivamente) ;
- Como vemos no print, nosso processo pai é representado pela faixa vermelha e o filho pela amarela, este último possui o PPID sendo igual ao PID do processo pai;
- Já o processo pai possui um PPID diferente, já que foi iniciado pelo terminal.

→ Usando o comando kill -9 PID para matar o processo filho e o processo pai.

Primeiro usando para matar o processo filho:

```
child: 82
parent: 17
child: 83
parent: 16
child: 84
parent: 15
child: 85
parent: 14
parent: 13
parent: 12
parent: 11
parent: 10
parent: 9
```

→ Vemos que quando o filho morre, o processo pai permanece tendo sua saída exibida no terminal, mas o filho não;

→ Reiniciando os processos e matando o pai primeiro:

```
parent: -76
child: 176
parent: -77
child: 177
parent: -78
child: 178
Killed
jntsduarte@tsuki:/mnt/c/Users/Cliente/Desktop/SO$ child: 179
child: 180
child: 181
child: 182
child: 183
child: 184
child: 185
```

→ Quando matamos o pai primeiro, temos uma diferença bem perceptível: o terminal fica livre para usarmos, mas a saída do processo filho permanece sendo exibida!

→ Caso matarmos o pai e depois o filho, temos:

```
child: 92
parent: 8
child: 93
parent: 7
child: 94
parent: 6
Killed
jntsduarte@tsuki: /mnt/c/Users/Cliente/Desktop/S0$ child: 95
child: 96
child: 97
child: 98
child: 99
child: 100
child: 101
child: 102
child: 103
child: 104
```

→ Sendo assim, a diferença entre a ordem de matar o processo pai ou o filho está nesta parte do terminal ficar livre (e continuar exibindo a saída do filho), caso seja morto o pai primeiro. Enquanto que se o filho morre primeiro, o terminal fica ocupado com a saída do pai, mesmo que não tenha mais o filho.

**Passo 16 ao 22:**→ Rodando o programa *'threaddemo.c'*, temos:

[illegible]

- O programa faz uso de *threads* no lugar dos processos, diferentemente do programa *processdemo* (mesmo que sejam muito semelhante nas demais funções);
- O valor de X é compartilhado pelas duas *threads* (assim como vemos no terminal);
- Com relação a velocidade de saída, as *threads* possuem uma vantagem muito significativa, sendo bem mais rápido do que o programa anterior e mais leves também. Isso se deve exatamente ao uso das *threads* com relação aos processos (já que ambos programas possuem mesmas funções);
- Vemos que ele também possui apenas um processo (diferente do programa anterior), está marcado em vermelho:

```
jntsduarte@tsuki:/mnt/c/Users/Cliente$ ps xl
```

F	UID	PID	PPID	PRI	NI	VSZ	RSS	WCHAN	STAT	TTY	TIME	COMMAND
4	1000	4946	4945	20	0	10040	5132	do_wai	Ss	pts/0	0:00	-bash
0	1000	4959	4946	20	0	19032	760	-	RL+	pts/0	0:14	./threaddemo
4	1000	4964	4963	20	0	10040	5140	do_wai	Ss	pts/1	0:00	-bash
0	1000	4977	4964	20	0	10520	3148	-	R+	pts/1	0:00	ps xl

- Quando removemos o loop da função *main*, temos:

```
jntsduarte@tsuki:/mnt/c/Users/Cliente/Desktop/SO$ ./threaddemo
creating threads:
jntsduarte@tsuki:/mnt/c/Users/Cliente/Desktop/SO$ ./threaddemo
```

- Vemos que quando o loop é removido, as *threads*, em alguns casos, nem são executadas. (OBS: dependendo das execuções, o programa pode exibir saídas distintas)
- Então, as vezes inicia as duas *threads* e para (além de outras saídas diferentes):

```
jntsduarte@tsuki:/mnt/c/Users/Cliente/Desktop/SO$ ./threaddemo
creating threads:
adjustment = 1; x = 50
adjustment = -1; x = 51
adjustment = -1; x = 51
jntsduarte@tsuki:/mnt/c/Users/Cliente/Desktop/SO$ ./threaddemo
```

→ Para alterar o código e deixar a saída igual ao do programa anterior (onde uma saída era crescente e outra decrescente), basta colocarmos a variável X, antes global, dentro da função de *adjustX*:

```
void * adjustX(void *n)
{ int i = (int)n;
  int x = 50; /* assim deixamos que cada thread siga um caminho crescente ou decrescente */
  while (1) /* loop forever */
  { printf("adjustment = %2i; x = %i\n", i, x);
    x += i;
    delay(rand() % MAXDELAY);
  }
  return(n);
}
```

→ Assim, temos a saída onde ambas iniciam com o valor 50 e seguem os caminhos (cada thread usa sua própria variável):

```
Windows PowerShell  x  jntsduarte@tsuki: /mnt/c/Users, x  jntsduarte@tsuki: /mnt/c/Users, x  +  v
jntsduarte@tsuki:/mnt/c/Users/Cliente/Desktop/SO$ ./threaddemo
creating threads:
adjustment = 1; x = 50
adjustment = -1; x = 50
adjustment = -1; x = 49
adjustment = -1; x = 48
adjustment = -1; x = 47
adjustment = 1; x = 51
adjustment = 1; x = 52
adjustment = -1; x = 46
adjustment = -1; x = 45
adjustment = 1; x = 53
adjustment = 1; x = 54
adjustment = -1; x = 44
adjustment = -1; x = 43
adjustment = 1; x = 55
adjustment = 1; x = 56
adjustment = -1; x = 42
adjustment = 1; x = 57
adjustment = -1; x = 41
adjustment = 1; x = 58
adjustment = -1; x = 40
adjustment = 1; x = 59
adjustment = 1; x = 60
adjustment = -1; x = 39
adjustment = -1; x = 38
adjustment = 1; x = 61
adjustment = -1; x = 37
adjustment = 1; x = 62
adjustment = -1; x = 36
```