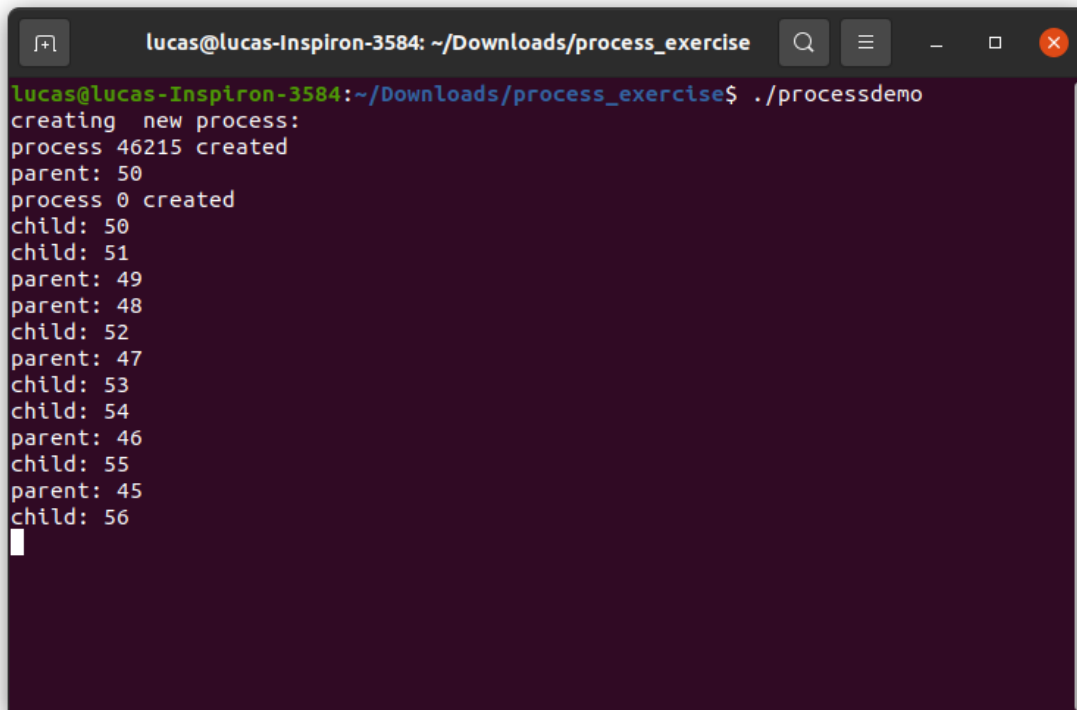


SISTEMAS OPERACIONAIS
PRÁTICA 1

9. Descreva a saída e explique por que ela é dessa forma.

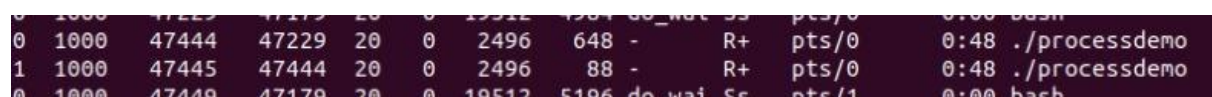


```
lucas@lucas-Inspiron-3584: ~/Downloads/process_exercise
lucas@lucas-Inspiron-3584:~/Downloads/process_exercise$ ./processdemo
creating new process:
process 46215 created
parent: 50
process 0 created
child: 50
child: 51
parent: 49
parent: 48
child: 52
parent: 47
child: 53
child: 54
parent: 46
child: 55
parent: 45
child: 56
```

Como visto na imagem, os processos pai e filho começam com o valor 50, mas o processo pai possui uma contagem decrescente, e o processo filho contém uma contagem crescente. Os valores de cada um são ajustados pela função 'adjustX', que recebe um valor para ser acrescentado ou diminuído no valor inicial. Essa mudança ocorre em uma condição "if" presente no código.

11. Qual o processo pai e qual o processo filho? (Dica, verifique a coluna PID e PPID. Se não souber o que é PID e PPID, procure no Google). Justifique.

Para ver esses dados, usei o comando ps xl:



```
0 1000 47444 47229 20 0 2496 648 - R+ pts/0 0:48 ./processdemo
1 1000 47445 47444 20 0 2496 88 - R+ pts/0 0:48 ./processdemo
0 1000 47449 47179 20 0 19512 5196 do_wai Ss pts/1 0:00 bash
```

PID é o número de identificação de um processo, e o PPID é a identificação do processo-pai (terceira e quarta coluna). Sendo assim, vemos que o processo pai é o processo da primeira linha, visto que o processo de baixo tem o PPID igual ao PID do processo pai.

12. Use o comando "kill -9 PID" para matar o processo filho. O que aconteceu?

```
child: 92
parent: 9
child: 92
parent: 8
child: 93
parent: 7
child: 94
parent: 6
child: 95
parent: 5
parent: 4
parent: 3
parent: 2
parent: 1
parent: 0
parent: -1
parent: -2
parent: -3
parent: -4
parent: -5
```

O processo pai permanece sendo exibido na tela, enquanto o processo filho para.

13. Use o comando "kill -9 PID" para matar o processo pai. O que aconteceu?

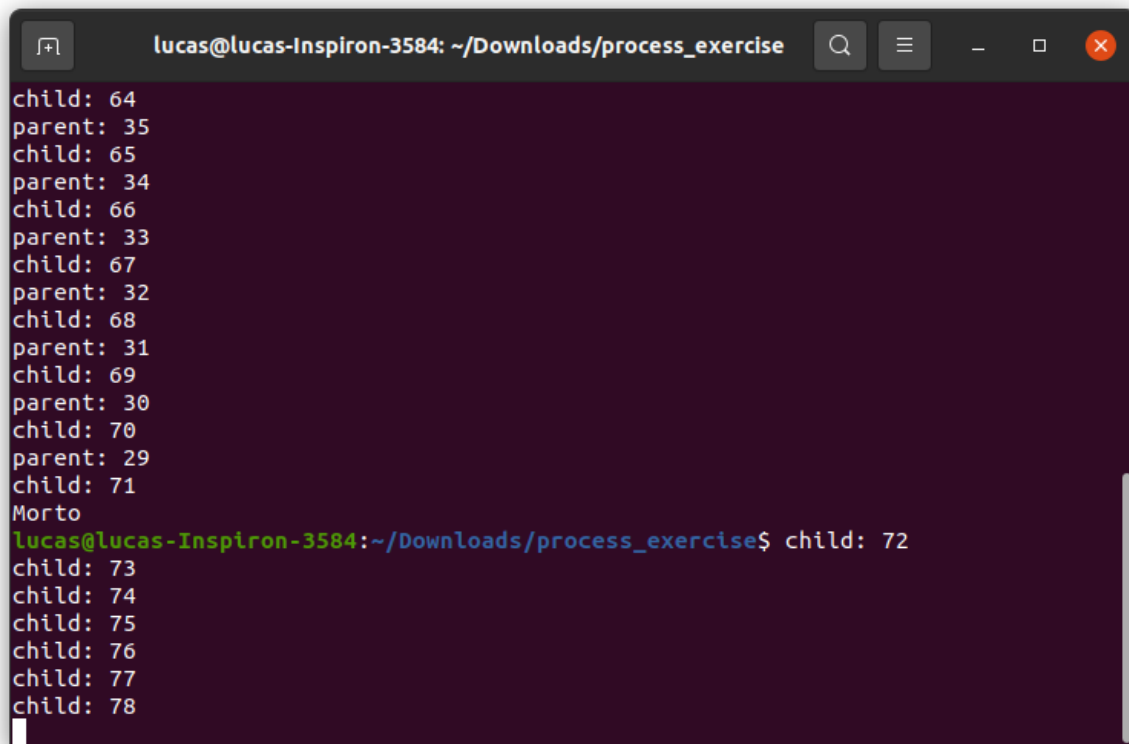
```
lucas@lucas-Inspiron-3584: ~/Downloads/process_exercise
lucas@lucas-Inspiron-3584:~/Downloads/process_exercise$ ./processdeno
creating new process:
process 49646 created
parent: 50
process 0 created
child: 50
child: 51
parent: 49
child: 52
parent: 48
child: 53
parent: 47
child: 54
parent: 46
child: 55
parent: 45
child: 56
parent: 44
child: 57
parent: 43
child: 58
parent: 42
Morto
lucas@lucas-Inspiron-3584:~/Downloads/process_exercise$ child: 59
child: 60
child: 61
child: 62
child: 63
child: 64
child: 65
child: 66
child: 67
child: 68
child: 69
child: 70
child: 71
```

O processo pai para, e o processo filho continua sendo exibido. Porém, o terminal fica livre pra ser usado durante a execução do código

14. Rode o programa novamente. Identifique e mate o processo pai primeiro em seguida o filho. O que aconteceu?

Ao matar o processo pai, ele para de ser exibido, e o processo filho continua a ser exibido com o terminal livre pra ser usado durante a execução do código.

Ao matar o processo filho, ele também deixa de ser exibido.



```
lucas@lucas-Inspiron-3584: ~/Downloads/process_exercise
child: 64
parent: 35
child: 65
parent: 34
child: 66
parent: 33
child: 67
parent: 32
child: 68
parent: 31
child: 69
parent: 30
child: 70
parent: 29
child: 71
Morto
lucas@lucas-Inspiron-3584:~/Downloads/process_exercise$ child: 72
child: 73
child: 74
child: 75
child: 76
child: 77
child: 78
```

15. Faz diferença matar o pai ou o filho antes?

Sim. Matando o processo pai primeiro, o processo filho continua sendo exibido e o terminal fica livre pra ser usado. Matando o processo filho primeiro, o terminal fica ocupado com a saída do pai.

18. Rode o programa. O que ele faz? Qual a diferença dele para o programa processdemo.c?

```
lucas@lucas-Inspiron-3584: ~/Downloads/process_exercise
adjustment = -1; x = 49
adjustment = 1; x = 48
adjustment = -1; x = 49
adjustment = -1; x = 48
adjustment = 1; x = 47
adjustment = -1; x = 48
adjustment = 1; x = 47
adjustment = 1; x = 48
adjustment = -1; x = 49
adjustment = -1; x = 48
adjustment = 1; x = 47
adjustment = 1; x = 48
adjustment = -1; x = 49
adjustment = 1; x = 48
adjustment = -1; x = 49
adjustment = 1; x = 48
adjustment = 1; x = 49
adjustment = -1; x = 50
adjustment = 1; x = 49
adjustment = 1; x = 50
adjustment = -1; x = 51
adjustment = -1; x = 50
adjustment = -1; x = 49
```

O programa usa threads no lugar dos processos, e o valor de X é compartilhado pelas duas threads

19. Qual a diferença de velocidade de saída (medido em linhas por segundo) comparado a processdemo? Quem é mais rápido? Você tem uma ideia do porquê?

O programa threaddemo é muito mais rápido que o processdemo devido ao uso de threads, e o threaddemo é muito mais leve que o processdemo também, pois ele possui apenas um processo rodando:

```
1 1000 4576 3293 20 0 25531228 50280 r0tex Sl ? 0:00 /bin/gd
0 1000 4949 3956 20 0 19040 816 - Rl+ pts/0 0:27 ./threa
0 1000 4954 3949 20 0 19512 5224 do_wai Ss pts/1 0:00 bash
```

21. Investigue o efeito de remover o loop infinito no fim do main(). O que acontece? Por que?


```
lucas@lucas-Inspiron-3584: ~/Downloads/process_exercise
adjustment = 1; x = 50
adjustment = -1; x = 50
adjustment = 1; x = 51
adjustment = -1; x = 49
adjustment = -1; x = 48
adjustment = -1; x = 47
adjustment = 1; x = 52
adjustment = 1; x = 53
adjustment = 1; x = 54
adjustment = -1; x = 46
adjustment = 1; x = 55
adjustment = -1; x = 45
adjustment = -1; x = 44
adjustment = 1; x = 56
adjustment = -1; x = 43
adjustment = 1; x = 57
adjustment = -1; x = 42
adjustment = 1; x = 58
adjustment = 1; x = 59
adjustment = -1; x = 41
adjustment = -1; x = 40
adjustment = 1; x = 60
```