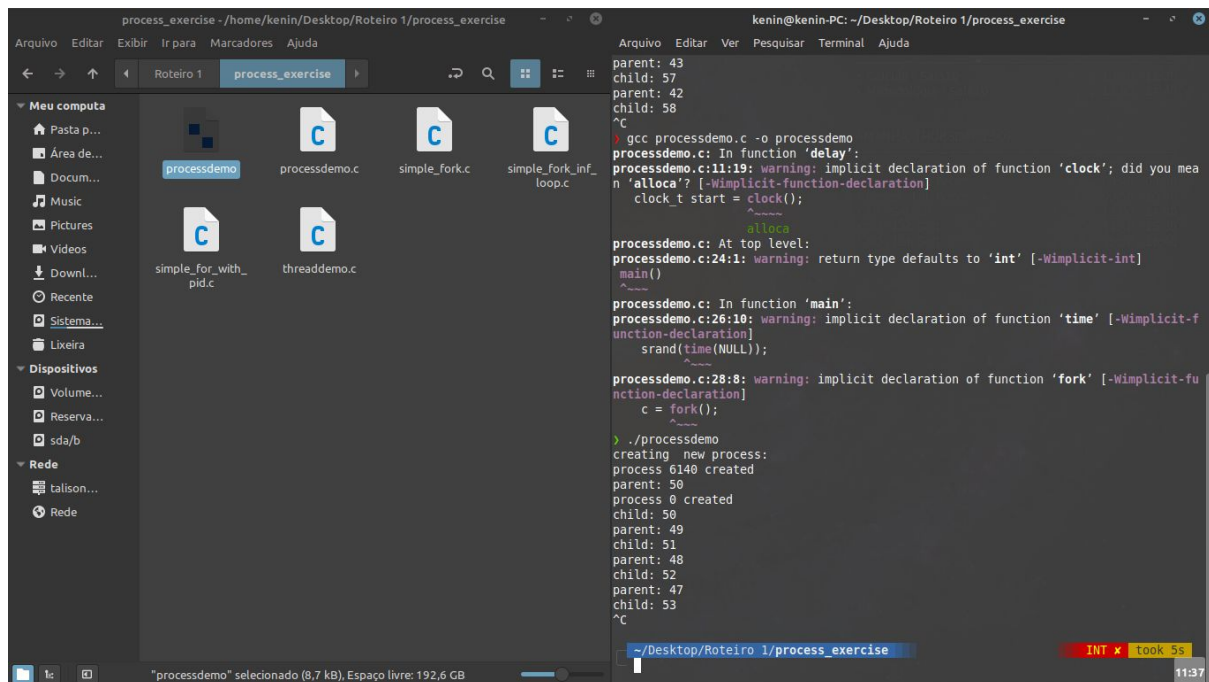


Roteiro I

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

R - O programa cria dois processos, no quais, serão um processo child, que retornará o valor "0", para a variável c e um processo parent, que retorna o PID do processo child, assim, cada um dos processos irá entrar na árvore de condições, onde serão separados por child(variável 'c' igual a '0') e parent(a variável 'c' não é igual a '0'). Assim, processo parent começa a decrementar o valor pré-definido de x e printa "parent: 'x'", e o child vai incrementar o valor e x e o printará "child: 'x'".



```
process_exercise ~/home/kenin/Desktop/Roteiro 1/process_exercise
Arquivo  Editar  Exibir  Ir para  Marcadores  Ajuda
←  →  ↑  Roteiro 1  process_exercise
Meu computa
  Pasta p...
  Área de...
  Docum...
  Music
  Pictures
  Videos
  Downl...
  Recente
  Sistema...
  Lixeira
Dispositivos
  Volume...
  Reserva...
  sda/b
Rede
  tallison...
  Rede
"processdemo" selecionado (8,7 kB), Espaço livre: 192,6 GB

kenin@kenin-PC: ~/Desktop/Roteiro 1/process_exercise
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
parent: 43
child: 57
parent: 42
child: 58
^C
> gcc processdemo.c -o processdemo
processdemo.c: In function 'delay':
processdemo.c:11:19: warning: implicit declaration of function 'clock'; did you mea
n 'alloca'? [-Wimplicit-function-declaration]
   clock_t start = clock();
                     ^~~~~
processdemo.c: At top level:
processdemo.c:24:1: warning: return type defaults to 'int' [-Wimplicit-int]
   main()
   ^~~~~
processdemo.c: In function 'main':
processdemo.c:26:10: warning: implicit declaration of function 'time' [-Wimplicit-f
unction-declaration]
   srand(time(NULL));
           ^~~~~
processdemo.c:28:8: warning: implicit declaration of function 'fork' [-Wimplicit-fu
nction-declaration]
   c = fork();
           ^~~
> ./processdemo
creating new process:
process 6140 created
parent: 50
process 0 created
child: 50
parent: 49
child: 51
parent: 48
child: 52
parent: 47
child: 53
^C
~/Desktop/Roteiro 1/process_exercise  INT x took 5s  11:37
```

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.

R - O parent é o processo com PID '6373', e o child é o processo com PID '6374'. Pois o valor do PID, ou Process ID, é o ID determinado pelo SO para o determinado programa, e o PPID, ou parent Process ID, é o PID do processo pai ao processo que tem esse PPID.

```
kenin@kenin-PC: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
0 1000 1808 1353 20 0 1254168 67092 poll s Sl ? 0:01 nemo-desktop child: 74  
0 1000 1840 1676 20 0 523652 98208 poll s Sl ? 0:01 /usr/share/dis parent: 26  
1 1000 1875 1732 20 0 1397188 215508 futex Sl ? 0:34 /usr/share/dis child: 75  
0 1000 2160 1 20 0 896428 63976 poll s Sl ? 0:00 mintUpdate parent: 25  
0 1000 2195 1353 20 0 1522080 96860 poll s Sl ? 0:06 /usr/bin/pytho child: 76  
0 1000 2280 1430 20 0 377888 9528 poll s Sl ? 0:00 /usr/lib/gvfs/ parent: 24  
0 1000 2305 1430 20 0 374472 7744 poll s Sl ? 0:00 /usr/lib/gvfs/ child: 77  
0 1000 2311 1 20 0 659376 94956 poll s Sl ? 0:17 mintreport-tra parent: 23  
4 1000 4102 1 20 0 1164180 266348 poll s Sll ? 1:20 /opt/google/ch child: 78  
0 1000 4107 4102 20 0 7612 764 pipe w S ? 0:00 cat parent: 22  
0 1000 4108 4102 20 0 7612 764 pipe w S ? 0:00 cat child: 79  
4 1000 4111 4102 20 0 448020 60108 wait S ? 0:00 /opt/google/ch parent: 21  
4 1000 4112 4111 20 0 27220 3996 skb wa S ? 0:00 /opt/google/ch child: 80  
5 1000 4115 4111 20 0 448020 15092 poll s S ? 0:00 /opt/google/ch parent: 20  
0 1000 4134 4102 20 0 741900 265896 poll s Sl ? 11:38 /opt/google/ch child: 81  
0 1000 4139 4102 20 0 542136 84896 futex Sll ? 0:50 /opt/google/ch parent: 19  
1 1000 4148 4134 20 0 474816 23812 skb wa S ? 0:00 /opt/google/ch child: 82  
1 1000 4164 4115 20 0 5388128 615316 futex Sl ? 52:21 /opt/google/ch parent: 18  
1 1000 4165 4115 20 0 9160596 176508 futex Sl ? 0:25 /opt/google/ch child: 83  
1 1000 4212 4115 20 0 4850860 172428 futex Sl ? 0:00 /opt/google/ch parent: 17  
1 1000 4237 4115 20 0 4771908 93468 futex Sl ? 0:00 /opt/google/ch child: 84  
0 1000 4318 4102 20 0 1026784 56796 futex Sl ? 0:45 /opt/google/ch parent: 16  
1 1000 5065 4115 20 0 4769432 94600 futex Sl ? 0:00 /opt/google/ch child: 85  
1 1000 5080 4115 20 0 4904660 222632 futex Sl ? 0:23 /opt/google/ch parent: 15  
0 1000 5120 1 20 0 1065992 75936 poll s Sl ? 0:00 nemo child: 86  
0 1000 5306 5120 20 0 763460 134044 poll s Sl ? 0:18 xed /home/keni parent: 14  
1 1000 5470 4115 20 0 4827952 135036 futex Sl ? 0:00 /opt/google/ch child: 87  
1 1000 5485 4115 20 0 4743748 54808 futex Sl ? 0:00 /opt/google/ch parent: 13  
0 1000 5990 1332 20 0 636188 41072 poll s Ssl ? 0:01 /usr/lib/gnome child: 88  
0 1000 5997 5990 20 0 70136 8428 sigsus Ss pts/0 0:00 zsh parent: 12  
0 1000 6032 1332 20 0 26892 3624 sigsus Ss ? 0:00 /bin/zsh -dfxc child: 89  
0 1000 6035 6032 20 0 74544 3392 poll s Sl ? 0:00 /home/kenin/.o parent: 11  
0 1000 6251 5990 20 0 67392 7740 sigsus Ss pts/1 0:00 zsh child: 90  
0 1000 6286 1332 20 0 26892 3564 sigsus Ss ? 0:00 /bin/zsh -dfxc child: 91  
0 1000 6288 6286 20 0 74544 3364 poll s Sl ? 0:00 /home/kenin/.o parent: 10  
0 1000 6372 5997 20 0 4504 756 - R+ pts/0 0:00 ./processdemo parent: 9  
1 1000 6374 6373 20 0 4504 80 - R+ pts/0 0:00 ./processdemo child: 92  
4 1000 6377 6251 20 0 29004 1532 - R+ pts/1 0:00 ps xl parent: 8  
child: 93  
parent: 7
```

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

R - o child foi encerrado, mas o parent continua a ser executado.

```
kenin@kenin-PC: ~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
1 1000 1875 1732 20 0 1397700 218088 futex Sl ? 0:37 /usr/share/dis child: 65  
0 1000 2160 1 20 0 896428 63976 poll s Sl ? 0:00 mintUpdate parent: 34  
0 1000 2195 1353 20 0 1522080 96860 poll s Sl ? 0:06 /usr/bin/pytho child: 66  
0 1000 2280 1430 20 0 377888 9528 poll s Sl ? 0:00 /usr/lib/gvfs/ parent: 33  
0 1000 2305 1430 20 0 374472 7744 poll s Sl ? 0:00 /usr/lib/gvfs/ child: 67  
0 1000 2311 1 20 0 659376 94956 poll s Sl ? 0:17 mintreport-tra parent: 32  
4 1000 4102 1 20 0 1191832 270848 poll s Sll ? 1:40 /opt/google/ch child: 68  
0 1000 4107 4102 20 0 7612 764 pipe w S ? 0:00 cat parent: 31  
0 1000 4108 4102 20 0 7612 764 pipe w S ? 0:00 cat child: 69  
4 1000 4111 4102 20 0 448020 60108 wait S ? 0:00 /opt/google/ch parent: 30  
4 1000 4112 4111 20 0 27220 3996 skb wa S ? 0:00 /opt/google/ch child: 70  
5 1000 4115 4111 20 0 448020 15092 poll s S ? 0:00 /opt/google/ch parent: 29  
0 1000 4134 4102 20 0 736816 262116 poll s Sl ? 11:52 /opt/google/ch child: 71  
0 1000 4139 4102 20 0 541624 84916 futex Sll ? 0:55 /opt/google/ch parent: 28  
1 1000 4148 4134 20 0 474816 23812 skb wa S ? 0:00 /opt/google/ch child: 72  
1 1000 4164 4115 20 0 5410956 642128 futex Sl ? 53:23 /opt/google/ch parent: 27  
1 1000 4165 4115 20 0 9160596 176588 futex Sl ? 0:27 /opt/google/ch parent: 26  
1 1000 4212 4115 20 0 4850860 172896 futex Sl ? 0:10 /opt/google/ch child: 73  
1 1000 4237 4115 20 0 4771908 93468 futex Sl ? 0:00 /opt/google/ch child: 74  
0 1000 4318 4102 20 0 1026784 56028 futex Sl ? 0:51 /opt/google/ch parent: 25  
1 1000 5065 4115 20 0 4769432 94600 futex Sl ? 0:00 /opt/google/ch child: 75  
1 1000 5080 4115 20 0 4904148 225352 futex Sl ? 0:28 /opt/google/ch parent: 24  
0 1000 5120 1 20 0 1065992 75908 poll s Sl ? 0:00 nemo child: 76  
0 1000 5306 5120 20 0 763460 134044 poll s Sl ? 0:22 xed /home/keni parent: 23  
1 1000 5470 4115 20 0 4828976 135440 futex Sl ? 0:12 /opt/google/ch child: 77  
0 1000 5990 1332 20 0 636708 41424 poll s Ssl ? 0:03 /usr/lib/gnome parent: 22  
0 1000 5997 5990 20 0 70136 8428 sigsus Ss pts/0 0:00 zsh parent: 21  
0 1000 6032 1332 20 0 26892 3624 sigsus Ss ? 0:00 /bin/zsh -dfxc parent: 20  
0 1000 6035 6032 20 0 74544 3392 poll s Sl ? 0:00 /home/kenin/.o parent: 19  
0 1000 6251 5990 20 0 67392 7756 sigsus Ss pts/1 0:00 zsh parent: 18  
0 1000 6286 1332 20 0 26892 3564 sigsus Ss ? 0:00 /bin/zsh -dfxc parent: 17  
0 1000 6288 6286 20 0 74544 3364 poll s Sl ? 0:00 /home/kenin/.o parent: 16  
1 1000 6521 4115 20 0 4885896 122720 futex Sl ? 0:03 /opt/google/ch parent: 15  
1 1000 6680 4115 20 0 4743748 54672 futex Sl ? 0:00 /opt/google/ch parent: 14  
0 1000 6717 5997 20 0 4504 756 - R+ pts/0 0:05 ./processdemo parent: 13  
1 1000 6718 6717 20 0 4504 80 - R+ pts/0 0:05 ./processdemo parent: 12  
4 1000 6721 6251 20 0 29004 1456 - R+ pts/1 0:00 ps xl parent: 11  
parent: 10  
parent: 9  
parent: 8  
parent: 7  
parent: 6
```

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

R - O parent foi encerrado, e o programa em si foi encerrado também.

The screenshot shows a terminal window with two panes. The left pane displays the output of the `ps -ef` command, listing various processes including `parent` and `child`. The right pane shows the execution of the `kill -9 6717` command, which successfully kills the process. A status bar at the bottom indicates that the process was killed and the terminal took 2m 10s to execute the command.

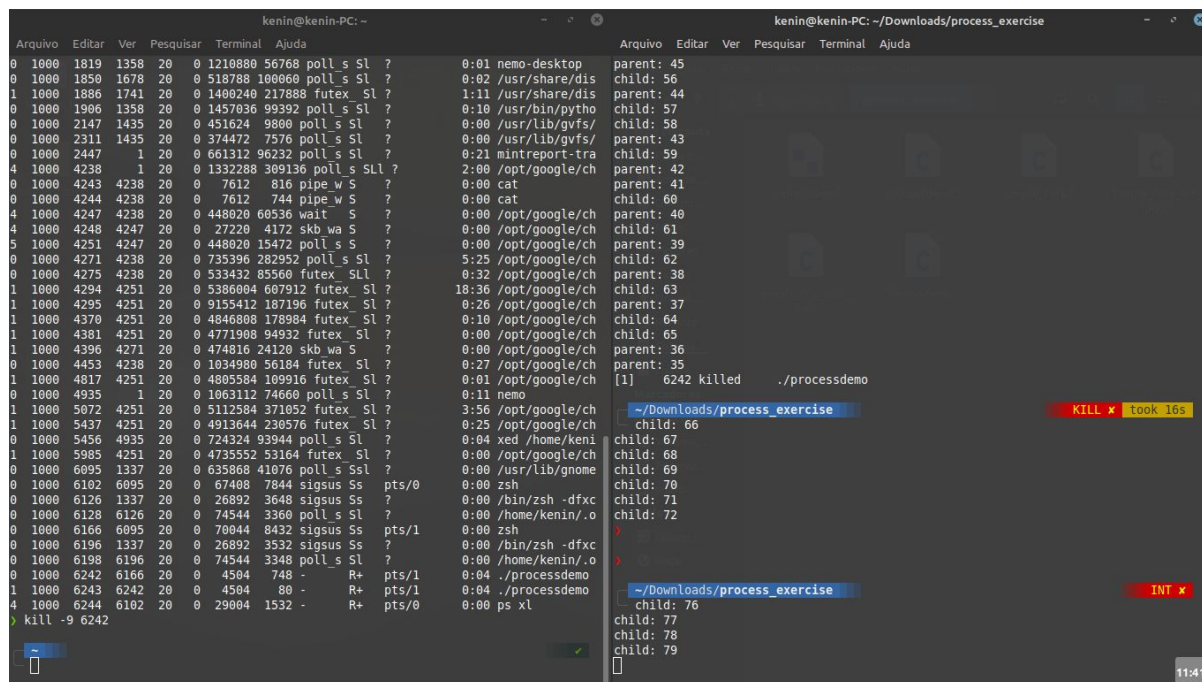
14. Rode o programa novamente. Identifique e mate o processo pai primeiro em seguida o filho. O que aconteceu?

R - o parent foi encerrado, o child continuou a ser executado e o seu PPID foi transferido para o system.

The screenshot shows a terminal window with two panes. The left pane displays the output of the `ps -ef` command, listing various processes including `parent` and `child`. The right pane shows the execution of the `kill -9 6899` command, which successfully kills the process. A status bar at the bottom indicates that the process was killed and the terminal took 20s to execute the command.

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

R - Sim. pois o processo filho vai precisar de um processo Parent com um PID válido, e já que o Parent foi morto antes, o PPID do filho acaba sendo transferido para a instância do usuário. Assim, o shell utilizado para inicializar os processos, acaba perdendo a permissão de encerrar o processo child.

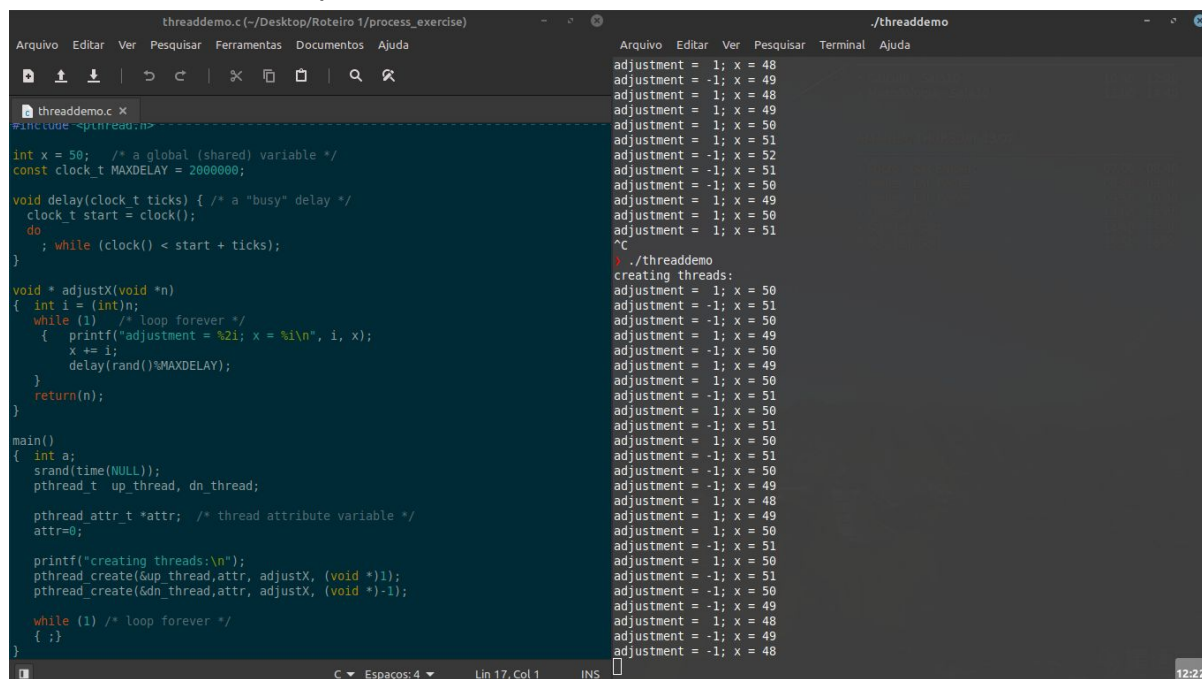


The screenshot shows a terminal window with two panes. The left pane displays the output of the 'ps' command, showing a list of processes with their PIDs, PPIDs, and other details. The right pane shows the output of the 'kill -9 6242' command, which successfully kills the process with PID 6242. A red arrow points to the 'kill -9 6242' command in the left pane, indicating the process being killed.

(a seta vermelha na horizontal, significa “ctrl+c”).

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

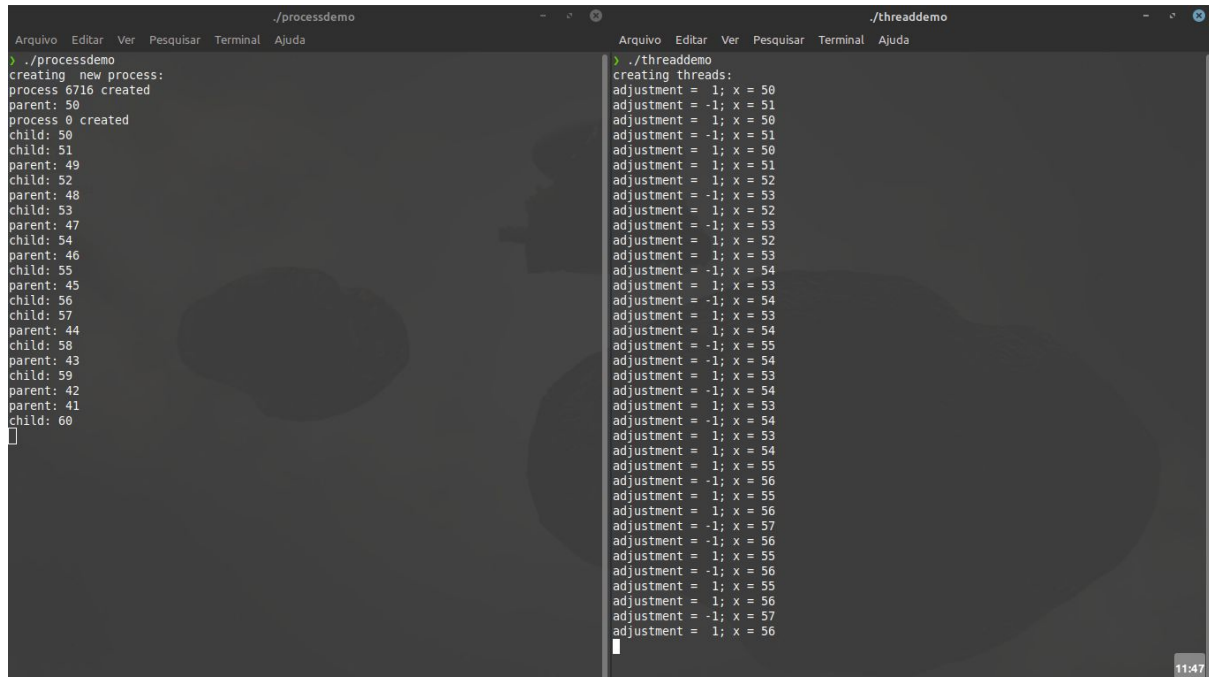
R - Ele segue a ideia do programa com processos, mas ele não tem o delay entre processamentos, assim as threads podem soltar uma quantidade aleatoria de saidas ao mesmo tempo. E elas estão utilizando a mesma variavel.



The screenshot shows a code editor with two panes. The left pane displays the source code of 'threaddemo.c', which includes a global variable 'x', a 'delay' function, and a 'main' function that creates two threads. The right pane shows the output of the program, which displays a series of 'adjustment' values and 'x' values, demonstrating the concurrent execution of the threads.

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

R - quase o dobro da velocidade. o threaddemo. O processamento das threads são mais eficientes que carrega um processo inteiro.

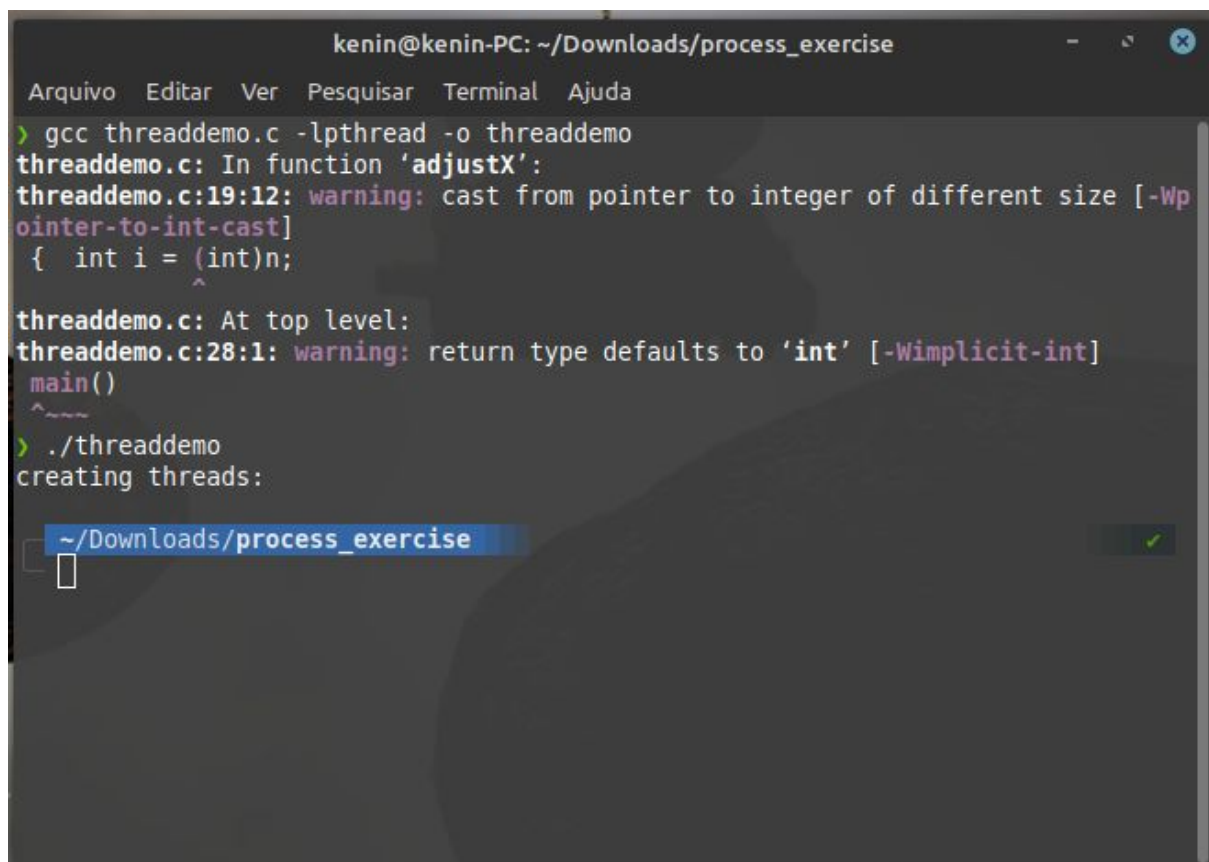


```
./processdemo
> ./processdemo
creating new process:
process 6716 created
parent: 50
process 0 created
child: 50
child: 51
parent: 49
child: 52
parent: 48
child: 53
parent: 47
child: 54
parent: 46
child: 55
parent: 45
child: 56
child: 57
parent: 44
child: 58
parent: 43
child: 59
parent: 42
parent: 41
child: 60
[

./threaddemo
> ./threaddemo
creating threads:
adjustment = 1; x = 50
adjustment = -1; x = 51
adjustment = 1; x = 50
adjustment = -1; x = 51
adjustment = 1; x = 50
adjustment = 1; x = 51
adjustment = 1; x = 52
adjustment = -1; x = 53
adjustment = -1; x = 52
adjustment = -1; x = 53
adjustment = 1; x = 52
adjustment = 1; x = 53
adjustment = -1; x = 54
adjustment = 1; x = 53
adjustment = -1; x = 54
adjustment = 1; x = 53
adjustment = -1; x = 54
adjustment = 1; x = 55
adjustment = -1; x = 54
adjustment = 1; x = 53
adjustment = -1; x = 54
adjustment = 1; x = 53
adjustment = -1; x = 54
adjustment = 1; x = 54
adjustment = -1; x = 54
adjustment = 1; x = 53
adjustment = -1; x = 54
adjustment = 1; x = 55
adjustment = -1; x = 56
adjustment = 1; x = 55
adjustment = 1; x = 56
adjustment = -1; x = 57
adjustment = -1; x = 56
adjustment = 1; x = 55
adjustment = -1; x = 56
adjustment = 1; x = 55
adjustment = 1; x = 56
adjustment = -1; x = 57
adjustment = 1; x = 56
```

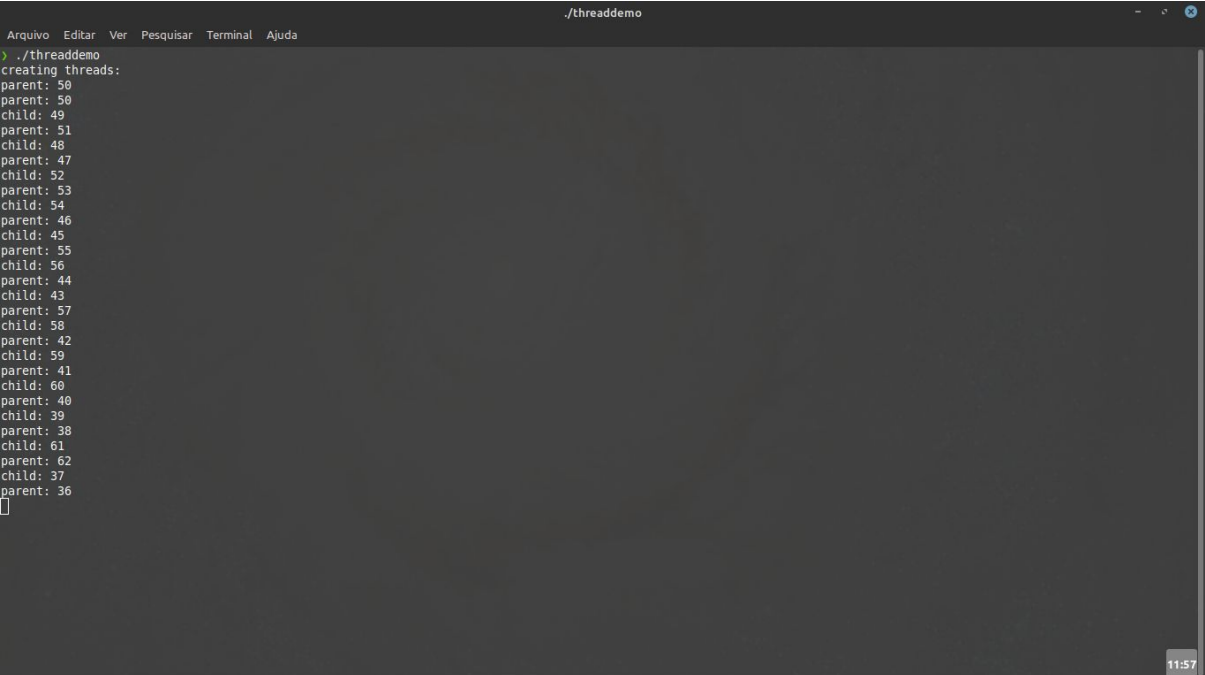
21. Investigue o efeito de remover o *loop* infinito no fim do main(). O que acontece? Por que?

R - nenhuma thread rodou.



```
kenin@kenin-PC: ~/Downloads/process_exercise
Arquivo Editar Ver Pesquisar Terminal Ajuda
> gcc threaddemo.c -lpthread -o threaddemo
threaddemo.c: In function 'adjustX':
threaddemo.c:19:12: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
{ int i = (int)n;
               ^
threaddemo.c: At top level:
threaddemo.c:28:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^
> ./threaddemo
creating threads:
~/Downloads/process_exercise
```

22. Modifique o programa threaddemo.c para ele fazer a mesma coisa que processdemo.c



```
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
./threaddemo
> ./threaddemo
creating threads:
parent: 50
parent: 50
child: 49
parent: 51
child: 48
parent: 47
child: 52
parent: 53
child: 54
parent: 46
child: 45
parent: 55
child: 56
parent: 44
child: 43
parent: 57
child: 58
parent: 42
child: 59
parent: 41
child: 60
parent: 40
child: 39
parent: 38
child: 61
parent: 62
child: 37
parent: 36

```

(arquivo = threaddemo)