מגיש: שילה גילאור
תז: 302537394

2) בחלק זה תראו שמספר הסיגנלים המתקבלים יכול להיות קטן ממספר הסיגנלים שנשלחים. במילים אחרות, סיגנלים אינם נכנסים לתור. על מנת להראות זאת, כתבו 2 תכניות **client.c ,server.c**. ה-client יכול לשלוח 2 סוגים של סיגנלים: SIGINT,SIGUSR1.
בתוך ה-server ה- handler של SIGINT סופר את מספר הסיגנלים SIGINT שמתקבלים.
ה-handler של SIGUSR1 מדפיס את מספר הסיגנלים SIGINT שהגיעו ל-server.
ה- client מקבל 3 פרמטרים בסדר הבא: **pid** של ה-server, מספר סיגנל
SIGUSR1(10) או SIGINT(2)
ומספר הסיגנלים שישלחו. לדוגמא:

1) client <server pid> 2 1000
2) client <server pid> 10 1

בדוגמה הראשונה client ישלח 1000 סיגנלים SIGINT לשרת עם <server pid>.
בדוגמה השניה השרת עם <server pid> ידפיס את מספר ה- SIGINT שהתקבלו.

אתרים שמהם למדתי:
https://stackoverflow.com/questions/60446063/sending-a-process-id-from-client-to-server
https://en.wikipedia.org/wiki/Signal_(IPC)
https://man7.org/linux/man-pages/man7/signal.7.html

הסבר:

כתבתי קוד ל client שמשתמש ב kill שישלח הודעה לשרת כמספר הפעמים שמתבקש משורת ההרצה.
גם כן כתבתי קוד מצד שרת שמטפל ב signal של SIGINT ויספור אותם בתוך נכלם גלובאלי, גם SIGUSR1 שיידע להדפיס כמה סיגנלים קיבל ואז לאפס את הנעלם.

shilo@shilo-VirtualBox: ~/Desktop/os_final/os_final_handing/task_2/task_22

```
shilo@shilo-VirtualBox:~/Desktop/os_final/os_final_handing/task_2/task_22$ make all
gcc -o client client.c
client.c: In function ‘main’:
client.c:14:22: warning: implicit declaration of function ‘atoi’ [-Wimplicit-function-declaration]
   14 |       for (int i=0;i<atoi(argv[3]);++i)
      |                      ^~~~
client.c:16:18: warning: implicit declaration of function ‘kill’ [-Wimplicit-function-declaration]
   16 |         status = kill(atoi(argv[1]), atoi(argv[2]));
      |                  ^~~~
gcc -o server server.c
shilo@shilo-VirtualBox:~/Desktop/os_final/os_final_handing/task_2/task_22$ ./client 4489 2 1000
shilo@shilo-VirtualBox:~/Desktop/os_final/os_final_handing/task_2/task_22$ ./client 4489 10 1
shilo@shilo-VirtualBox:~/Desktop/os_final/os_final_handing/task_2/task_22$
```

shilo@shilo-VirtualBox: ~/Desktop/os_final/os_final_handing/task_2/task_22

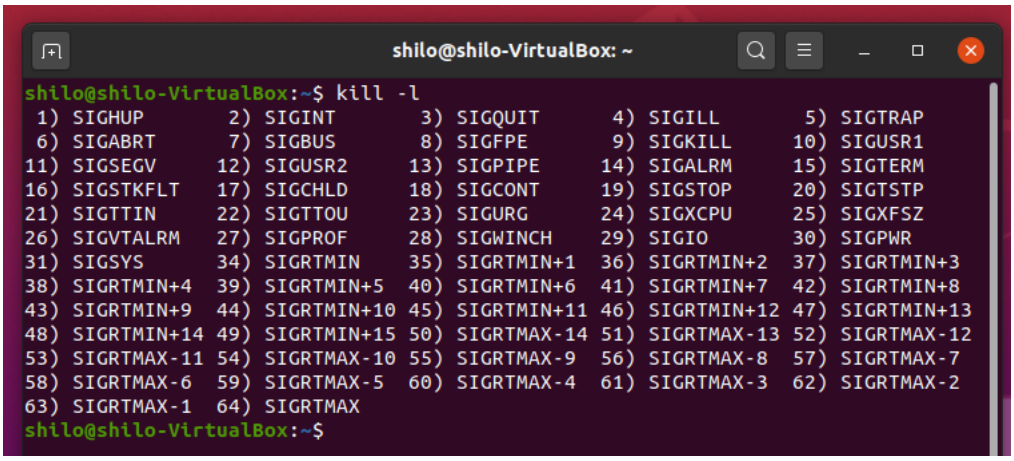
```
shilo@shilo-VirtualBox:~/Desktop/os_final/os_final_handing/task_2/task_22$ ./server
pid: 4489
SIGUSER1 recieved, global_counter = :357 , and restarting the counter
```

2) למדו על real time signals שנצברים לתור (**sigqueue**, **sigaction**, etc). הסבירו את היתרונות והחסרונות של שתי השיטות. הוסיפו את ההסברים לקובץ **q22_<your_id>.pdf**.

אתרים מהם למדתי:

https://stackoverflow.com/questions/231912/what-is-the-difference-between-sigaction-and-signal
https://stackoverflow.com/questions/9696706/some-questions-about-linux-signals
https://man7.org/linux/man-pages/man7/signal-safety.7.html

subject	signal	sigqueue	sigaction	kill
real time signals	both	Only real time	both	1-33 standard 34-64 realtime
Block other signals	Does not		Does block	
Works on other OS	yes		NO!	No (Taskkill)
signal	recieve	send	recieve	send



Comments:

`signal` is actually of the Unix System V behavior. POSIX allows either this behavior or the much more sane BSD behavior, but since you can't be sure which one you'll get, it's still best to use `sigaction`

with sigaction it is recommended to use write and not printf since printf is not async while write is.

Recommended to use sigaction more than signal if possible.

The effects of ***`signal()`*** *in a multi-threaded process are unspecified*

When only use functions defined by Standard C! is an order: use signal and not sigaction.

Whether you're starting from scratch or modifying an old program, sigaction should be the right option.

`sigaction()` is good and well-defined, but is a Linux function and so it works only on Linux. `signal()` is bad and poorly-defined, but is a C standard function and so it works on anything.

sigqueue() have the same semantics with respect to the null signal as *kill*(), and that the same permission checking be used. But because of the difficulty of implementing the "broadcast" semantic of *kill*() (for example, to process groups) and the interaction with resource allocation, this semantic was not adopted. The *sigqueue*()