Team 17
Introduction to Computer Systems
Coding Project 2 (Action 1 & 4 - 25 points)
May 10, 2022

Name	Email
Kevin Zhang	kyz2@scarletmail.rutgers.edu
Justine Catli	jlc562@scarletmail.rutgers.edu
Sumant Pottepalem	ssp213@scarletmail.rutgers.edu
Harsh Patel	hp440@scarletmail.rutgers.edu
Jawad Abdussalam	jja153@scarletmail.rutgers.edu
Felix Shames	fs410@scarletmail.rutgers.edu
Brandon Yuen	by161@scarletmail.rutgers.edu
Taqiya Ehsan	te137@scarletmail.rutgers.edu

All had equal contributions.
Project content begins on the next page.

Action 1

All threads handle the same three signals SIGILL, SIGABRT and SIGSEGV.

How to run the file:

compilation: `gcc -o project2action1and4.1 project2action1and4.1.c -pthread`
running: `./project2action1and4.1`

Action 4

Q1) For SIGILL, SIGABRT and SIGSEGV, the signal numbers are output as was configured within the signal handler:

The other signals are all set to SIG_IGN for the threads so they were all ignored when invoked by the same process.

```
te137@engsoft:~$ kill -SIGFPE 1546495
te137@engsoft:~$ kill -SIGINT 1546495
te137@engsoft:~$ kill -SIGCHLD 1546495
te137@engsoft:~$ kill -SIGHUP 1546509
te137@engsoft:~$ kill -SIGTSTP 1546509
```

```
te137@engsoft:~$ ./p2action4
Process ID: 1546495
Created thread 3343574784
Created thread 3335182080
Created thread 3326789376
Created thread 3318396672
Thread 3343574784 terminated
Thread 3335182080 terminated
Thread 3326789376 terminated
Thread 3318396672 terminated
te137@engsoft:~$ ./p2action4
Process ID: 1546509
Created thread 4257974016
Created thread 4249581312
Created thread 4241188608
Created thread 4232795904
Thread 4257974016 terminated
Thread 4249581312 terminated
Thread 4241188608 terminated
Thread 4232795904 terminated
te137@engsoft:~$
```

Q2) The results depend on the signal that is being sent. For SIGILL, SIGABRT, SIGSEGV, these signals will go through the signal handler, which results in a print statement being displayed. For the other signals, they will be ignored.

Observations:
By running the program, we see that it will run indefinitely (with the included while loop for the signal actions - Image Shown Below). With the program running indefinitely, the threads are never terminated since the while loop never ends. Hence, we need to use CTRL + C to end the program.

```
    while(1){
    sigaction(SIGILL, &new_action, NULL);
    sigaction(SIGABRT, &new_action, NULL);
    sigaction(SIGSEGV, &new_action, NULL);
    }
    signal(SIGILL, sig_handler);
    signal(SIGABRT, sig_handler);
    signal(SIGSEGV, sig_handler);
```

```
jja153@engsoft:~$ ./action
Created thread 95348480
Created thread 86955776
Created thread 78563072
Created thread 70170368
^C
```

As shown in the image above, we were also able to display the threads that were created based on their Thread ID number. Overall, we saw that all threads ran in a similar fashion, and we believe this is due to the fact that they all originate from the same void function.

To run the program:
compilation: `gcc -o action action.c -pthread`
running: `./action`