ICS Homework 4

March 10, 2023

1 System Software

1.1 Signal handler

One TA writes the following code about a user-defined signal handler on a x86-64 Linux machine. Read this C program and answer the questions below.

```
#include <signal.h>
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

void handler(int sig)
{
        printf(”hello\n”);
}

int main(void)
{
        signal(SIGINT, handler);
        kill(getpid(), SIGINT);
        while(1);
        return 0;
}
```

recive

send

← send ctrl-c

1. Does the function 'handler' run in user mode or kernel mode?
   SOLUTION:
   User mode.

2. What's the output of this program? When we type a 'ctrl-c', what will happen? How to use *kill* command to stop this program?
   SOLUTION:
   hello
   When we type a 'ctrl-c', it will print one more 'hello'.
   First find the pid $*pid* of this program. And then use '*kill* − 9 $*pid*'

3. When we run this program in gdb, can we see the same result as Q2? If not, explain why and show how to fix it to get the same result in gdb as

Q2.
SOLUTION:
No.

Because gdb will stop the debugging program immediately whenever an error signal happens. Although SIGINT does not indicate an error in the program, it is normally fatal so it can carry out the purpose of the interrupt: to kill the program. Gdb will stop our program and not pass this signal to our program. (see: https://sourceware.org/gdb/onlinedocs/gdb/Signals.html)
We can change these settings with the handle command like

```
handle SIGINT nostop print pass
```