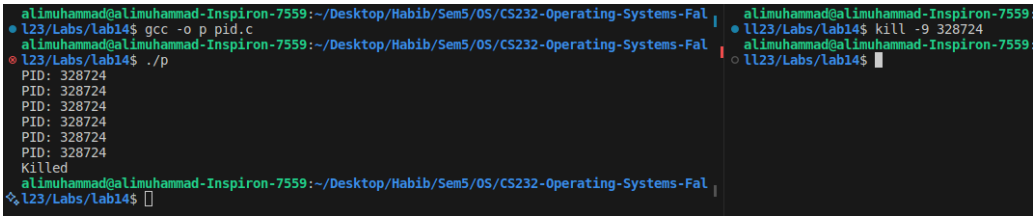
Operating Systems - CS/CE 232L/324L
Lab 14: Signals in Depth

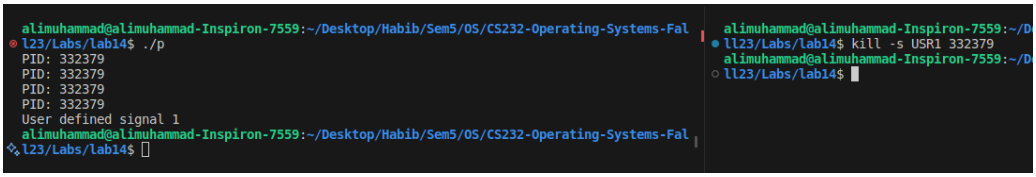
Ali Muhammad Asad - aa07190

Example 1: `kill -9 <pid>`



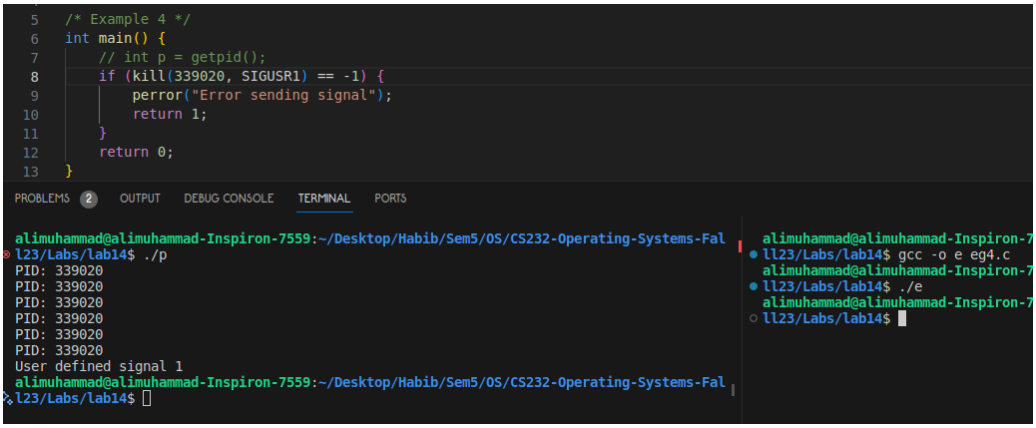
Example 1: Using `kill -9` to kill a process

Example 2: `kill -s USR1 <pid>`



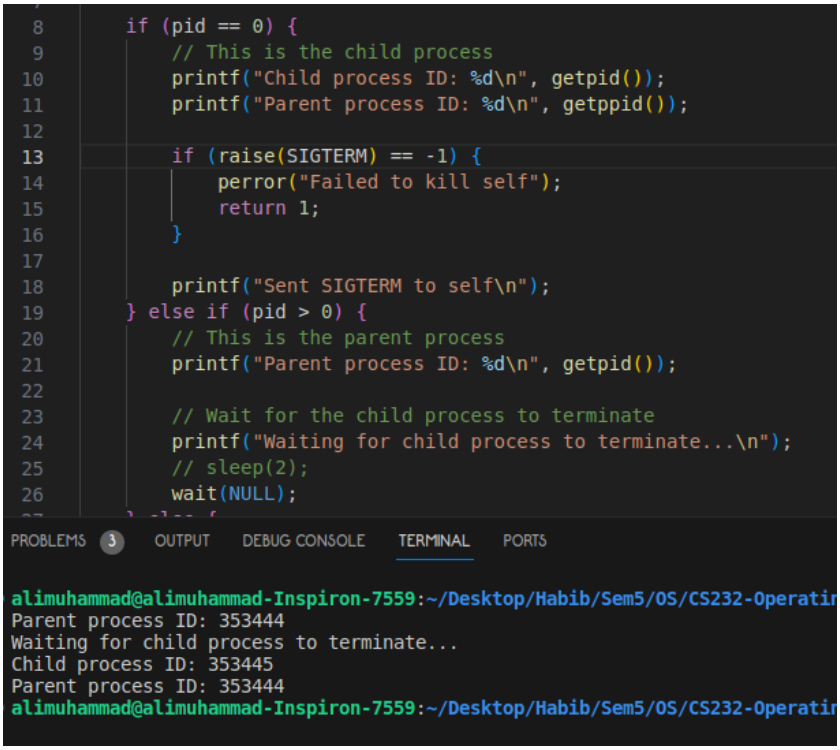
Example 2: Using `kill -s USR1` to send a signal to a process

Example 4: `if (kill(3423, SIGUR1) == -1)`



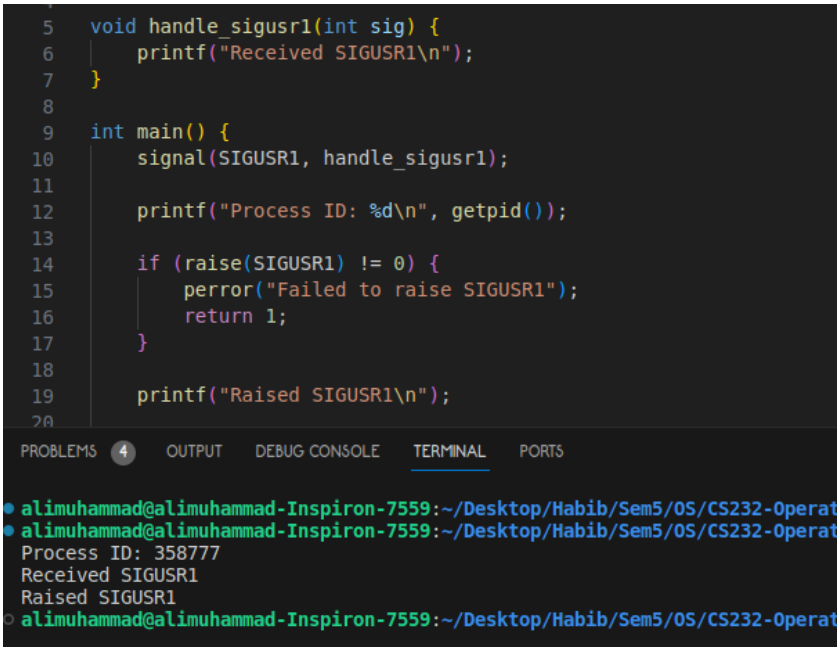
Example 4: Using `kill()` to send a signal to a process

Example 5: Child murders its parent



Example 5: Child murders its parent

Example 6: Process commits suicide; sends signal to kill itself

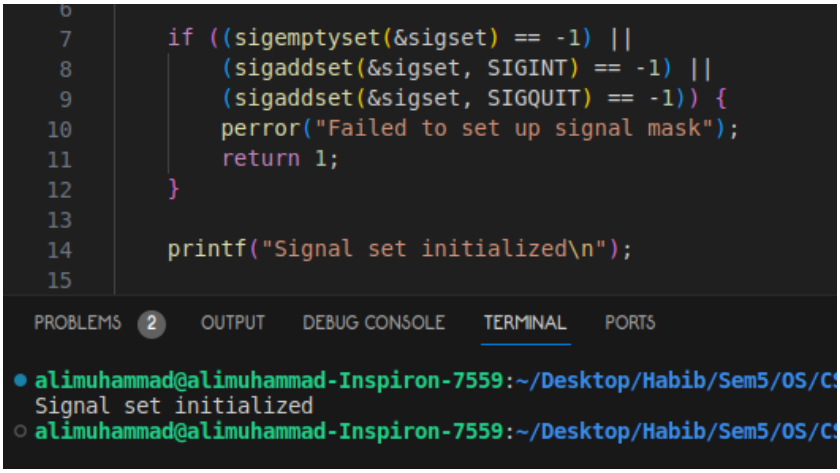


Example 6: Oh no process suicided - reason? This lab

Example 7: `simplealarm.c`

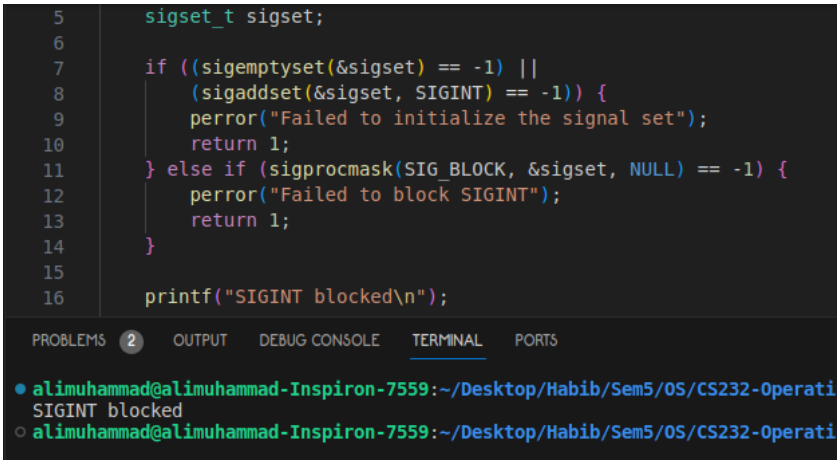
Example 7: Alarm

Example 8: `SIGINT` and `SIGQUIT`



Example 8

Example 9: `sigset_t newsigset` blocking signals



Example 9

Question 1: Is it possible that after a call to 'makepair' 'pipe1' exists but 'pipe2' does not?

Solution: Yes its possible. The function attempts to create two named pipes (FIFOs) using mkfifo for pipe1 and pipe2. If pipe1 gets created successfully but pipe2 fails (due to lack of permissions or space etc, that could set errno to something other than EEXIST), the function will not remove pipe1 before it returns. In this scenario, pipe1 would exist, but pipe2 would not.

Question 2: Does a 'makepair' return value of 0 guarantee that FIFOs corresponding to 'pipe1' and 'pipe2' are available on return?

Solution: A return value of 0 from makepair does indeed guarantee that both FIFOs corresponding to pipe1 and pipe2 have been successfully created and are available for use. The function only returns 0 if both mkfifo calls succeed (ignoring the case where either FIFO already exists, as indicated by errno == EEXIST). If any step of the FIFO creation process or the signal mask restoration with sigprocmask fails, the function cleans up by unlinking any FIFOs that might have been created during this call and returns -1.

Example 10: `mysighand`

```
11      newact.sa_handler = my_sighand; /* set the new handler */
12      newact.sa_flags = 0; /* no special options */
13
14      if (sigemptyset(&newact.sa_mask) == -1 ||
15          sigaction(SIGINT, &newact, NULL) == -1) {
16          perror("Failed to install SIGINT signal handler");
17          return 1;
18      }
19
20      printf("SIGINT handler installed\n");
21
22      /* Loop forever, waiting for signals */
23      for ( ; ; ) ;
```

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operat
SIGINT handler installed
^CReceived SIGINT
^CReceived SIGINT
^CReceived SIGINT
```

Example 10

Example 11: `struct sigaction act`

```
    struct sigaction act;
    printf("PID: %d\n", getpid());
    if (sigaction(SIGINT, NULL, &act) == -1) { /* Find current
    handler */
        perror("Failed to get old handler for SIGINT");
        return 1;
    } else if (act.sa_handler == SIG_DFL) { /* If SIGINT handl
    default */
        act.sa_handler = SIG_IGN; /* Set new SIGINT handler to

        if (sigaction(SIGINT, &act, NULL) == -1) {
            perror("Failed to set new handler for SIGINT");
            return 1;
        }
    }

    printf("SIGINT handler set to ignore\n");
```

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Oper
PID: 390242
SIGINT handler set to ignore
^C^C^C^C^C^C^C^C^CKilled
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Oper
```

Example 11

Example 12: `catchctrlc(int signo)`

```
void catchsigt(int signo) {
    char *handmsg = "I got Ctrl-C\n";
    size_t msglen = sizeof(handmsg);

    write(STDERR_FILENO, handmsg, msglen);
}

int main(void) {
    printf("PID is %d\n", getpid());
    struct sigaction act;
    act.sa_handler = catchsigt;
    sigemptyset(&act.sa_mask);
    act.sa_flags = 0;

    if (sigaction(SIGINT, &act, NULL) == -1) {
        perror("Failed to set up SIGINT handler");
        return 1;
```

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-
PID is 394418
SIGINT handler set up
^CI got Ct^CI got Ct^CI got Ct^CI got Ct^CI got Ct^CI got Ct^CI got
I got CtKilled
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-
```

Example 12

Question 3: Why didn't Example 12 use fprintf or strlen in the signal handler?

Solution: In the context of signal handlers, it is important to use functions that are async-signal-safe. When a signal handler is invoked, it interrupts the normal flow of the program execution. If such functions are called again from a signal handler, it could lead to data corruption or undefined behaviour. fprintf and strlen are not async-signal-safe functions, and could interact with buffered I/O or cause unpredictable results if a signal occurs. write is used instead since it is async-signal-safe that directly interacts with file desriptors without buffering.

Example 13: `testignored`

```
5   int testignored(int signo) {
6       struct sigaction act;
7       if ((sigaction(signo, NULL, &act) == -1) ||
8           (act.sa_handler == SIG_IGN))
9           return 1;
10      return 0;
11  }
12
13  int main(void) {
14      int signo = SIGINT; /* Change this to the signal numb
        check */
15      printf("PID is %d\n", getpid());
16      if (testignored(signo)) {
17          printf("Signal %d is ignored\n", signo);
18      } else {
19          printf("Signal %d is not ignored\n", signo);
20      }
```

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS23
PID is 398874
Signal 2 is not ignored
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS23
```

Example 13