

Operating Systems - CS/CE 232L/324L

Lab 14: Signals in Depth

Ali Muhammad Asad - aa07190

Example 1: `kill -9 <pid>`

```
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fall123/Labs/Lab14$ gcc -o p pid.c
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fall123/Labs/Lab14$ ./p
PID: 328724
PID: 328724
PID: 328724
PID: 328724
PID: 328724
Killed
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fall123/Labs/Lab14$
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fall123/Labs/Lab14$ kill -9 328724
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fall123/Labs/Lab14$
```

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

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

```
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fall123/Labs/Lab14$ ./p
PID: 332379
PID: 332379
PID: 332379
PID: 332379
User defined signal 1
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fall123/Labs/Lab14$
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fall123/Labs/Lab14$ kill -s USR1 332379
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fall123/Labs/Lab14$
```

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

Example 4: `if (kill(3423, SIGUSR1) == -1)`

```
5  /* Example 4 */
6  int main() {
7      // int p = getpid();
8      if (kill(339020, SIGUSR1) == -1) {
9          perror("Error sending signal");
10         return 1;
11     }
12     return 0;
13 }
```

alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fal
l23/Labs/Lab14\$./p
PID: 339020
PID: 339020
PID: 339020
PID: 339020
PID: 339020
PID: 339020
User defined signal 1
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-Systems-Fal
l23/Labs/Lab14\$

alimuhammad@alimuhammad-Inspiron-7
l23/Labs/Lab14\$ gcc -o e eg4.c
alimuhammad@alimuhammad-Inspiron-7
l23/Labs/Lab14\$./e
alimuhammad@alimuhammad-Inspiron-7
l23/Labs/Lab14\$

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

Example 5: Child murders its parent

```
8      if (pid == 0) {
9          // This is the child process
10         printf("Child process ID: %d\n", getpid());
11         printf("Parent process ID: %d\n", getppid());
12
13         if (raise(SIGTERM) == -1) {
14             perror("Failed to kill self");
15             return 1;
16         }
17
18         printf("Sent SIGTERM to self\n");
19     } else if (pid > 0) {
20         // This is the parent process
21         printf("Parent process ID: %d\n", getpid());
22
23         // Wait for the child process to terminate
24         printf("Waiting for child process to terminate...\n");
25         // sleep(2);
26         wait(NULL);
27     }
```

alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operati
Parent process ID: 353444
Waiting for child process to terminate...
Child process ID: 353445
Parent process ID: 353444
alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operati

Example 5: Child murders its parent

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

```
5 void handle_sigusr1(int sig) {
6     printf("Received SIGUSR1\n");
7 }
8
9 int main() {
10     signal(SIGUSR1, handle_sigusr1);
11
12     printf("Process ID: %d\n", getpid());
13
14     if (raise(SIGUSR1) != 0) {
15         perror("Failed to raise SIGUSR1");
16         return 1;
17     }
18
19     printf("Raised SIGUSR1\n");
20 }
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

● alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operat
● alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operat
Process ID: 358777
Received SIGUSR1
Raised SIGUSR1
○ alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operat

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

Example 7: `simplealarm.c`

```
5 void handle_sigalrm(int sig) {
6     printf("Received SIGALRM, exiting\n");
7     _exit(0);
8 }
9
10 int main(void) {
11     signal(SIGALRM, handle_sigalrm);
12     alarm(2);
13     for ( ; ; ) ;
14 }
15
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

● alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operat
● alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operat
● alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operat
Received SIGALRM, exiting
○ alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operat

Example 7: Alarm

Example 8: `SIGINT` and `SIGQUIT`

```
6
7     if ((sigemptyset(&sigset) == -1) ||
8         (sigaddset(&sigset, SIGINT) == -1) ||
9         (sigaddset(&sigset, SIGQUIT) == -1)) {
10        perror("Failed to set up signal mask");
11        return 1;
12    }
13
14    printf("Signal set initialized\n");
15
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

- alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-System:~\$./signal_set.c
Signal set initialized
- alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-System:~\$

Example 8

Example 9: `sigset_t` `newsigset` blocking signals

```
5     sigset_t sigset;
6
7     if ((sigemptyset(&sigset) == -1) ||
8         (sigaddset(&sigset, SIGINT) == -1)) {
9         perror("Failed to initialize the signal set");
10        return 1;
11    } else if (sigprocmask(SIG_BLOCK, &sigset, NULL) == -1) {
12        perror("Failed to block SIGINT");
13        return 1;
14    }
15
16    printf("SIGINT blocked\n");

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

- alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-System:~\$./sigset_t_newsigset.c
SIGINT blocked
- alimuhammad@alimuhammad-Inspiron-7559:~/Desktop/Habib/Sem5/OS/CS232-Operating-System:~\$

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

o 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`

[illegible]

Example 11

Example 12: `catchctrlc(int signo)`

```

5 void catchesigt(int signo){
6     char *handmsg = "I got Ctrl-C\n";
7     size_t msglen = sizeof(handmsg);
8
9     write(STDERR_FILENO, handmsg, msglen);
10 }
11
12 int main(void) {
13     printf("PID is %d\n", getpid());
14     struct sigaction act;
15     act.sa_handler = catchesigt;
16     sigemptyset(&act.sa_mask);
17     act.sa_flags = 0;
18
19     if (sigaction(SIGINT, &act, NULL) == -1) {
20         perror("Failed to set up SIGINT handler");
21         return 1;
22     }
23 }

```

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
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 descriptors 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 number
15     check */
16     printf("PID is %d\n", getpid());
17     if (testignored(signo)) {
18         printf("Signal %d is ignored\n", signo);
19     } else {
20         printf("Signal %d is not ignored\n", signo);
21     }
22 }
```

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