**Objectives**

* To generate signals and handling them using a signal handler function
* To block specific signals during program execution and catching the signals after all other signals have been handled

**Pre-Lab Theory**

Void sig\_handler(int signo){

If (signo == SIGINT)

Printf(“SIGINT is received);

Else if (signo == SIGPIPE)

Printf(“SIGPIPE is received);

}

Int main(){

Struct sigaction act;

act.sa\_handler = sig\_handler;

act.sa\_flags = 0;

sigaction(SIGINT, &act, NULL);

sigaction(SIGPIPE, &act, NULL);

raise(SIGINT);

raise(SIGPIPE);

}

Task1: write the signal handler for receiving a set of signals

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

void signal\_handler(int signo) {

switch (signo) {

case SIGINT:

printf("Caught SIGINT (Interrupt from keyboard).\n");

break;

case SIGPIPE:

printf("Caught SIGPIPE (Broken pipe).\n");

break;

case SIGQUIT:

printf("Caught SIGQUIT (Quit from keyboard).\n");

break;

case SIGALRM:

printf("Caught SIGALRM (alarm clock termination) .\n");

break;

default:

printf("Caught unexpected signal: %d\n", signo);

break;

}}

int main() {

struct sigaction act;

act.sa\_handler = signal\_handler;

act.sa\_flags = 0;

int signals[4] = {SIGINT, SIGPIPE, SIGQUIT, SIGALRM};

for( int i = 0; i < 4; i++){

sigaction(signals[i], &act, NULL);}

printf("Raising SIGINT...\n");

raise(SIGINT);

printf("Raising SIGPIPE...\n");

raise(SIGPIPE);

printf("Raising SIGQUIT...\n");

raise(SIGQUIT);

printf("Raising SIGALRM..\n");

raise(SIGALRM);

printf("All signals raised and handled.\n");

return 0; }

Task2: in main function prepare a set of signals to be blocked

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

void signal\_handler(int signo) {

switch (signo) {

case SIGINT:

printf("Caught SIGINT (Interrupt from keyboard).\n");

break;

case SIGPIPE:

printf("Caught SIGPIPE (Broken pipe).\n");

break;

case SIGQUIT:

printf("Caught SIGQUIT (Quit from keyboard).\n");

break;

case SIGALRM:

printf("Caught SIGALRM (ALarm clock termination).\n");

break;

default:

printf("Caught unexpected signal: %d\n", signo);

break;

}}

int main() {

struct sigaction act;

act.sa\_handler = signal\_handler;

act.sa\_flags = 0;

int signals[4] = {SIGINT, SIGPIPE, SIGQUIT, SIGALRM};

for (int i = 0; i < 4; i++) {

sigaction(signals[i], &act, NULL);

}

sigset\_t block\_set;

sigemptyset(&block\_set); // Initialization of an empty set

sigaddset(&block\_set, SIGPIPE);

sigaddset(&block\_set, SIGALRM);

sigprocmask(SIG\_BLOCK, &block\_set, NULL);

printf("SIGPIPE and SIGUSR1 are now blocked.\n");

printf("Raising SIGINT...\n");

raise(SIGINT);

printf("Raising SIGPIPE...\n");

raise(SIGPIPE);

printf("Raising SIGQUIT...\n");

raise(SIGQUIT);

printf("Raising SIGALRM...\n");

raise(SIGALRM);

printf("Unblocking signals...\n");

sigprocmask(SIG\_UNBLOCK, &block\_set, NULL);

printf("SIGPIPE and ALRM are now unblocked. Handlers should execute.\n");

return 0;}