Michael Osei

5/8/2023

COSC 430

Week 13 Assignment

Code:

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

#include <unistd.h>

#include <sys/wait.h>

volatile sig\_atomic\_t signals\_received = 0;

void parent\_sig\_handler(int signum) {

signals\_received++;

printf("Parent: received %d signals\n", signals\_received);

}

void child1\_sig\_handler(int signum) {

signals\_received++;

printf("Child 1: received %d signals\n", signals\_received);

}

void child2\_sig\_handler(int signum) {

signals\_received++;

printf("Child 2: received %d signals\n", signals\_received);

}

void child3\_sig\_handler(int signum) {

signals\_received++;

printf("Child 3: received %d signals\n", signals\_received);

}

int main() {

struct sigaction sa;

pid\_t child1\_pid, child2\_pid, child3\_pid;

int status;

// Set up signal handlers for parent and children

sa.sa\_handler = parent\_sig\_handler;

sigemptyset(&sa.sa\_mask);

sa.sa\_flags = 0;

sigaction(SIGUSR1, &sa, NULL);

sa.sa\_handler = child1\_sig\_handler;

sigemptyset(&sa.sa\_mask);

sa.sa\_flags = 0;

sigaction(SIGUSR2, &sa, NULL);

sa.sa\_handler = child2\_sig\_handler;

sigemptyset(&sa.sa\_mask);

sa.sa\_flags = 0;

sigaction(SIGINT, &sa, NULL);

sa.sa\_handler = child3\_sig\_handler;

sigemptyset(&sa.sa\_mask);

sa.sa\_flags = 0;

sigaction(SIGTERM, &sa, NULL);

// Create child processes

child1\_pid = fork();

if (child1\_pid == 0) {

// Child 1 process

while (1) {

sleep(1);

kill(getppid(), SIGUSR1);

}

}

child2\_pid = fork();

if (child2\_pid == 0) {

// Child 2 process

while (1) {

sleep(1);

kill(getppid(), SIGUSR2);

}

}

child3\_pid = fork();

if (child3\_pid == 0) {

// Child 3 process

while (1) {

sleep(1);

kill(getppid(), SIGINT);

}

}

// Parent process

while (1) {

sleep(1);

kill(child1\_pid, SIGTERM);

kill(child2\_pid, SIGTERM);

kill(child3\_pid, SIGTERM);

wait(&status);

wait(&status);

wait(&status);

printf("Parent: received %d signals\n", signals\_received);

}

return 0;

}

Result:

