Wayne State University

CSC 4421 - Winter 2017
Computer Operating Systems Labs
Lab 8 - Signals

Instructor 001: David Warnke
Instructor 003: Rui Chen

Points Possible: 100

Due: March 27, 2017, by 11:59pm

Redo lab 5, but now use only the signal mechanism system calls. **The kill() system call does not necessarily terminate a process, it is the system call used to send a signal.** Don't use wait() or pipe(). You must use kill() and pause() system calls. You must create only one child, and allow the processes to run back and forth. The parent can print the iteration since it runs first. You can still read/write from/to a file. A sample of the screen output is listed below.

```
x = 19530

ITERATION 1
Parent: x = 19525
Child: x = 3905

ITERATION 2
Parent: x = 3900
Child: x = 780

ITERATION 3
Parent: x = 775
Child: x = 155

ITERATION 4
Parent: x = 150
Child: x = 30

ITERATION 5
Parent: x = 25
Child: x = 5
```

output.txt

You can learn from this code below on how to do the parent-child communication.

```
//From 60-256 System Programming: Signals II by Dr. B. Boufama, http://kobti.myweb.cs.
    uwindsor.ca/60-256/c6-2.pdf
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
void action(int);
void action(int dummy){
  sleep(1);
  printf("Switching\n");
}

int main(int argc, char *argv[]){
  pid_t pid;
  if((pid=fork())>0){//parent
  sleep(1);
  while(1){
  printf("Parent is running\n");
  kill(pid, SIGUSR1);
  signal(SIGUSR1, action);
  pause();
  }
}
else
  while(1){//child
  signal(SIGUSR1, action);
  pause();
  printf("Child is running\n");
  kill(getppid(), SIGUSR1);
  }
}
```

sample.c