Michael Smith

CSE 460 – Operating Systems

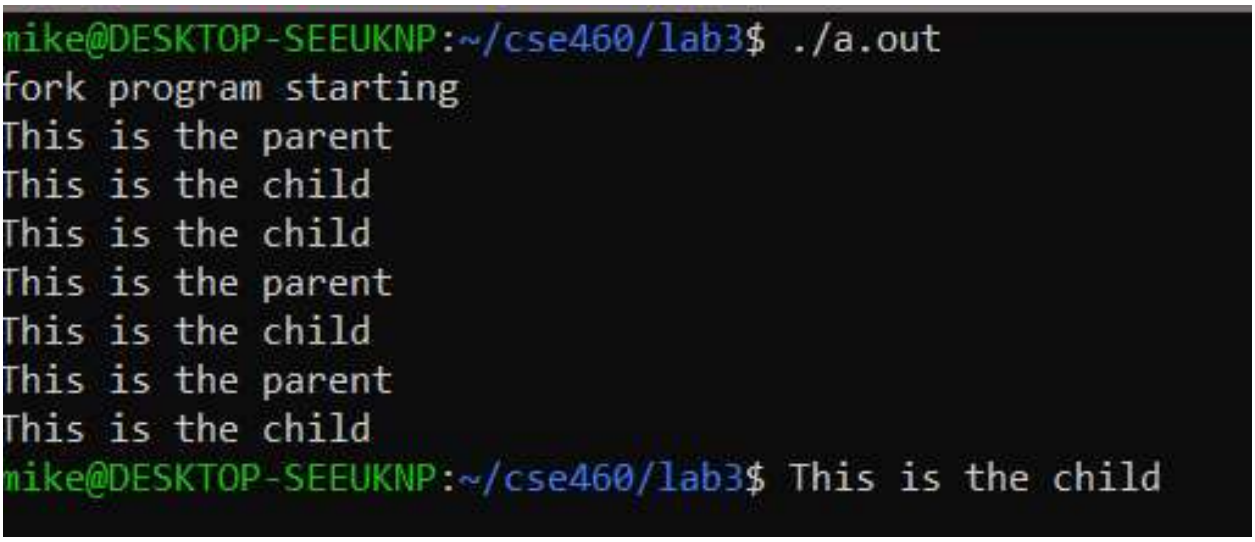
Lab 3 (Worth 20 Points)

1.) Replacing a Process image

```
#include <unistd.h>
#include <iostream>

using namespace std;

int main()
{
        cout << "Running ps with execl\n";
        execl( "ps", "ps", "-ax", (char *)NULL );

        cout << "Done!\n";

        return 0;

}
~
```

```
mike@DESKTOP-SEEUKNP:~/cse460/lab3$ ./a.out
Running ps with execl
Done!
mike@DESKTOP-SEEUKNP:~/cse460/lab3$
```

2.) Duplicating a process image



The parent process is create 3 times while the for loop is running and the child is outputted 5 times. The processes us a turn order system while outputting and share resources.

3.)Waiting for a Process

```cpp
//test_wait.cpp
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>

using namespace std;

int main()
{
  pid_t pid, childPid;          //process id
  char *message;
  int n;
  int exit_code;

  cout << "fork program starting\n";
  pid = fork();
  switch ( pid ) {
  case -1:
    cout << "Fork failure!\n";
    return 1;
  case 0:
    childPid = fork();
    if (childPid == 0) {
        message = (char *)"This is the grandchild\n";
        n = 9;
        exit_code = 21;
    }
    else {
    message = "This is the child\n";
    n = 5;
    exit_code = 9;
    }
    break;
  default:
    message = "This is the parent\n";
    n = 3;
    exit_code = 0;
    break;
  }
  for ( int i = 0; i < n; ++i ) {
    cout << message;
    sleep ( 1 );
  }
  //This is for the grandchild to finish
  if (childPid !=0) {
```

```
  default:
    message = "This is the parent\n";
    n = 3;
    exit_code = 0;
    break;
  }
  for ( int i = 0; i < n; ++i ) {
    cout << message;
    sleep ( 1 );
  }
  //This is for the grandchild to finish
  if (childPid !=0) {
        int stat_val;
        pid_t granchild_pid;

        granchild_pid = wait (&stat_val);
        cout << "Grandchild finished PID = " << granchild_pid << endl;
        cout << "Grandchild finished PPID = " << getpid() << endl;
        cout << "Grandchild finished GPPID = " << getppid() << endl;

        if (WIFEXITED (stat_val))
                cout << "Grandchild exited with code" << WEXITSTATUS (stat_val) << endl;
        else
                cout << "Grandchild extied terminated abnormally." << endl;
  }


  //waiting for child to finish
  if ( pid != 0 ) {             //parent
    int stat_val;
    pid_t child_pid;

    child_pid = wait ( &stat_val );     //wait for child
    cout << "Child finished: PID = " << child_pid << endl;
    if ( WIFEXITED ( stat_val ) )
        cout << "child exited with code " << WEXITSTATUS ( stat_val ) << endl;
    else
        cout << "child terminated abnormally!" << endl;
  }
  exit ( exit_code );
}
```

```
mike@DESKTOP-SEEUKNP:~/cse460/lab3$ ./a.out
fork program starting
This is the parent
This is the child
This is the grandchild
This is the parent
This is the grandchild
This is the child
This is the parent
This is the grandchild
This is the child
This is the grandchild
This is the child
This is the child
This is the grandchild
This is the grandchild
This is the grandchild
This is the grandchild
This is the grandchild
Grandchild finished PID = 5077
Grandchild finished PPID = 5076
Grandchild finished GPPID = 5075
Grandchild exited with code21
Child finished: PID = 5076
child exited with code 9
```

4.) Signals

```cpp
#include <signal.h>
#include <unistd.h>
#include <iostream>

using namespace std;

void func ( int sig )
{
  cout << "Oops! -- I got a signal " << sig << endl;
}

int main()
{
  (void) signal ( SIGINT, func );       //catch terminal interrupts

  for ( int i = 0; i < 20; ++i ) {
    cout << "CSUSB CS 460 lab on signals" << endl;
    sleep ( 1 );
  }
  return 0;
}
~
```

```
mike@DESKTOP-SEEUKNP:~/cse460/lab3$ ./a.out
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
^COops! -- I got a signal 2
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
^COops! -- I got a signal 2
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
^COops! -- I got a signal 2
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
CSUSB CS 460 lab on signals
mike@DESKTOP-SEEUKNP:~/cse460/lab3$
```

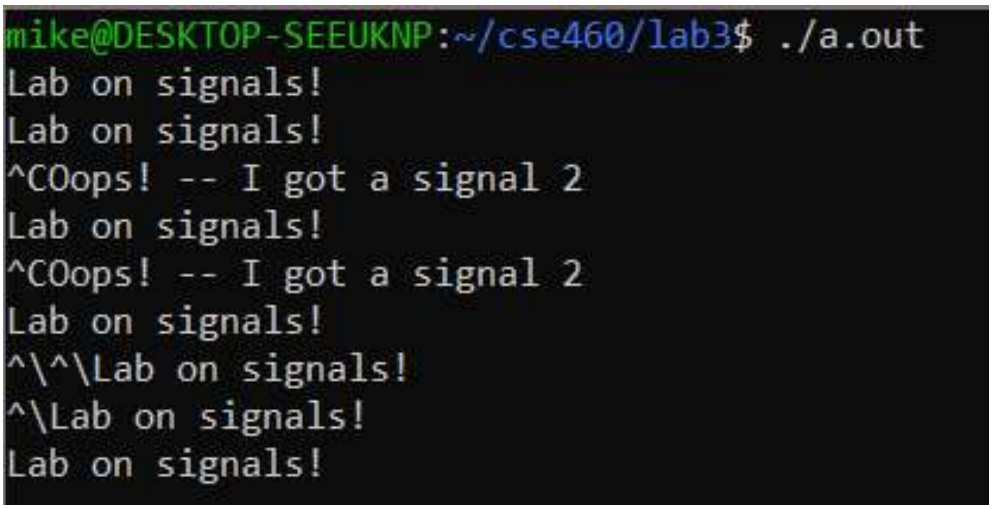
When pressing ^C, the output comes from the func(), while signal() catches the SIGNIT sent by the ^C. The SIGNIT value then becomes 2.

Test_alarm.cpp output only since the code was provided and unmodified.

```
mike@DESKTOP-SEEUKNP:~/cse460/lab3$ ./a.out
Alarm testing!
Waiting for alarm to go off!
Alarm has gone off
Done!
mike@DESKTOP-SEEUKNP:~/cse460/lab3$
```

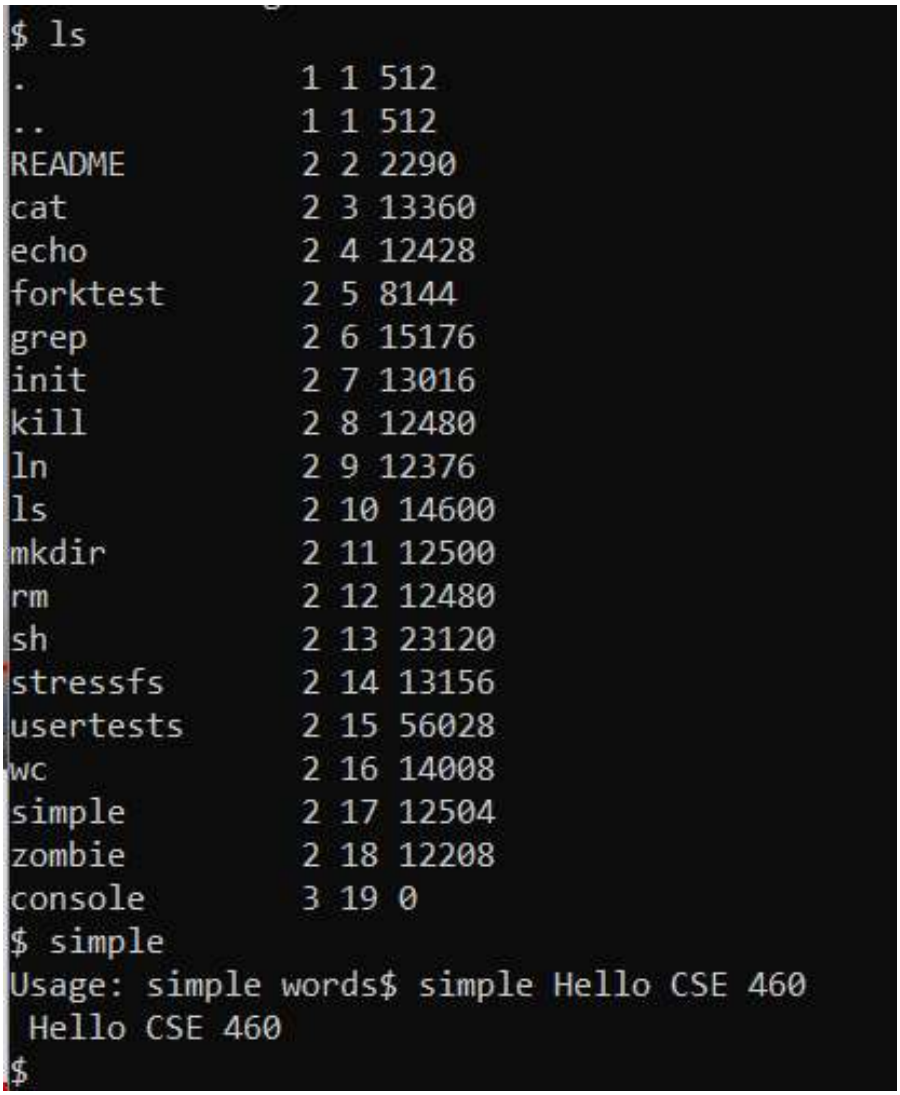
This program runs initially with the "Alarm testing! Waiting for alarm to go off!" Then after a few seconds the second two lines follow up. The parent process says the message then pauses while listening to the signal. Then the child process sleeps. Next it uses kill to send its signal back to the parents. Since the parents are listening, the alarm is sounded.

```cpp
//Test_sigaction.cpp

#include <signal.h>
#include <unistd.h>
#include <iostream>
#include <stdio.h>

using namespace std;

void func ( int sig )
{
  cout << "Oops! -- I got a signal " << sig << endl;
  if (sig == SIGQUIT)
  {
        raise(SIGTERM);
  }
}

int main()
{
        struct sigaction act;
        sigset_t set;

        sigemptyset(&set);
        sigaddset(&set, SIGINT);
        sigaddset(&set, SIGQUIT);
        act.sa_handler = func;
        act.sa_mask = set;
        act.sa_flags = 0;

        sigaction(SIGINT, &act, NULL);
        sigaction(SIGQUIT, &act, NULL);

        while(1) {
                cout << "Lab on signals!" << endl;
                sleep(1);
        }

        return 0;
}
~
~
~
```



Unfortunately the Windows Subsystem does not support the ^\ command. So due to a bug in the subsystem with SIGQUIT, the code would continue to run.

5.) XV6



Conclusion: I was able to complete every question on the lab, all the code was functional. Even was able to create the simple file in the XV6 OS. Unfortunately due to a bug that was not in my control the ^\ command is not working. But that as I said is out of my control, so I don't believe I should lose points for it.

Total: 20/20