Jesse Oesterblad
CSE 460
Lab 3

Processes and Signals

1. Replacing a Process Image

```
Script started on Thu 26 Jan 2017 11:03:53 AM STD
jesse@JESSE-PC:part1$ ./test_exec
Running ps with execlp
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 /init
    2 ?        Ss     0:00 /bin/bash
   11 ?        S      0:00 script
   12 ?        S      0:00 script
   13 ?        Ss     0:00 bash -i
   16 ?        R      0:00 ps -ax
jesse@JESSE-PC:part1$ ./test_execl
Running ps with execl
Done!
jesse@JESSE-PC:part1$ exit
exit

Script done on Thu 26 Jan 2017 11:04:06 AM STD
```

```cpp
//test_execl.cpp
#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;
}
```

2. Duplicating a Process Image

```
Script started on Thu 26 Jan 2017 11:32:14 AM STD
jesse@JESSE-PC:part2$ ./test_fork
```

```
fork program starting
This is the parent
This is the child
This is the parent
This is the child
This is the parent
This is the child
jesse@JESSE-PC:part2$ This is the child
This is the child
exit
exit

Script done on Thu 26 Jan 2017 11:32:27 AM STD
```

The parent process goes through the for-loop 3 times and the child goes through 5 times. The processes take turns outputting and they appear to share the resources fairly.

---

3. Waiting for a Process

```
Script started on Thu 26 Jan 2017 12:10:28 PM STD
jesse@JESSE-PC:part3$ ./test_wait
fork program starting
This is the parent
This is the child
This is the grandchild
This is the parent
This is the child
This is the grandchild
This is the parent
This is the child
This is the grandchild
This is the child
This is the grandchild
This is the child
This is the grandchild
This is the grandchild
This is the grandchild
Grandchild finished: PID = 165
Grandchild finished: PPID = 164
Grandchild finished: GPPID = 163
```

```
Grandchild exited with code 18
Child finished: PID = 164
child exited with code 9
jesse@JESSE-PC:part3$ exit
exit

Script done on Thu 26 Jan 2017 12:10:41 PM STD
```

```
//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 = 7;
      exit_code = 18;
    } else {
      message = (char *)"This is the child\n";
      n = 5;
      exit_code = 9;
```

```
    }
    break;
  default:
    message = (char *)"This is the parent\n";
    n = 3;
    exit_code = 0;
    break;
  }
  for ( int i = 0; i < n; ++i ) {
    cout << message;
    sleep ( 1 );
  }

  //waiting for grandchild to finish
  if ( childPid != 0 ) {//child
    int stat_val;
    pid_t grandchild_pid;

    grandchild_pid = wait ( &stat_val );//wait for grandchild
    cout << "Grandchild finished: PID = " << grandchild_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 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 );
}
```

---

4. Signals

```
Script started on Thu 26 Jan 2017 12:16:23 PM STD
jesse@JESSE-PC:part4$ ./test_signal
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
^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
jesse@JESSE-PC:part4$ exit
exit

Script done on Thu 26 Jan 2017 12:16:47 PM STD
```

When I hit ^C, the program outputs from the func() function. signal() is catching the SIGINT sent by pressing ^C. The int value of SIGINT is 2 as "described in the original POSIX.1-1990 standard."

```
Script started on Thu 26 Jan 2017 12:29:53 PM STD
jesse@JESSE-PC:part4$ ./test_alarm
Alarm testing!
Waiting for alarm to go off!
```

```
Alarm has gone off
Done!
jesse@JESSE-PC:part4$ exit
exit

Script done on Thu 26 Jan 2017 12:30:04 PM STD
```

I see “Alarm testing! Waiting for alarm to go off!” and then it waits a few seconds until “Alarm has gone off” “Done!”. The parent process says its message, starts listening for a signal and then pauses. This pause will not allow the program to continue until it receives a signal. The child process should still be sleeping. After it is done sleeping, the child process uses kill to send its parent process a signal. Since our parent process is listening for the SIGALRM signal, we send that to ding the alarm. I could also send 14 instead of SIGALRM.

```
Script started on Thu 26 Jan 2017 01:16:09 PM STD
jesse@JESSE-PC:part4$ ./test_sigaction
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
^COops! -- I got a signal 2
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
```

```
jesse@JESSE-PC:part4$ exit
exit

Script done on Thu 26 Jan 2017 01:16:34 PM STD
```

```
Script started on Thu 26 Jan 2017 11:39:53 PM UTC
jesse618:~/workspace/cse460 $ ./test_sigaction
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
^COops! -- I got a signal 2
CSUSB CS 460 lab on signals
^\Oops! -- I got a signal 3
Terminated
jesse618:~/workspace/cse460 $ exit
exit

Script done on Thu 26 Jan 2017 11:40:06 PM UTC
```

```
//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 action;
  sigset_t set;
```

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

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

  while ( 1 ) {
    cout << "CSUSB CS 460 lab on signals" << endl;
    sleep ( 1 );
  }

  return 0;
}
```

---

5. **Report**

I have successfully completed all sections of this lab. I am giving myself full credit: 20/20.

Doesn’t affect the report:

The Windows Subsystem for Linux (WSL) does not support ^\ as SIGQUIT from the keyboard yet. I am not exactly sure why this is the case or if it will be addressed. CTRL + INPUT has been bugged on WSL in the past. I completed the last portion of the lab on a cloud9.io workspace.