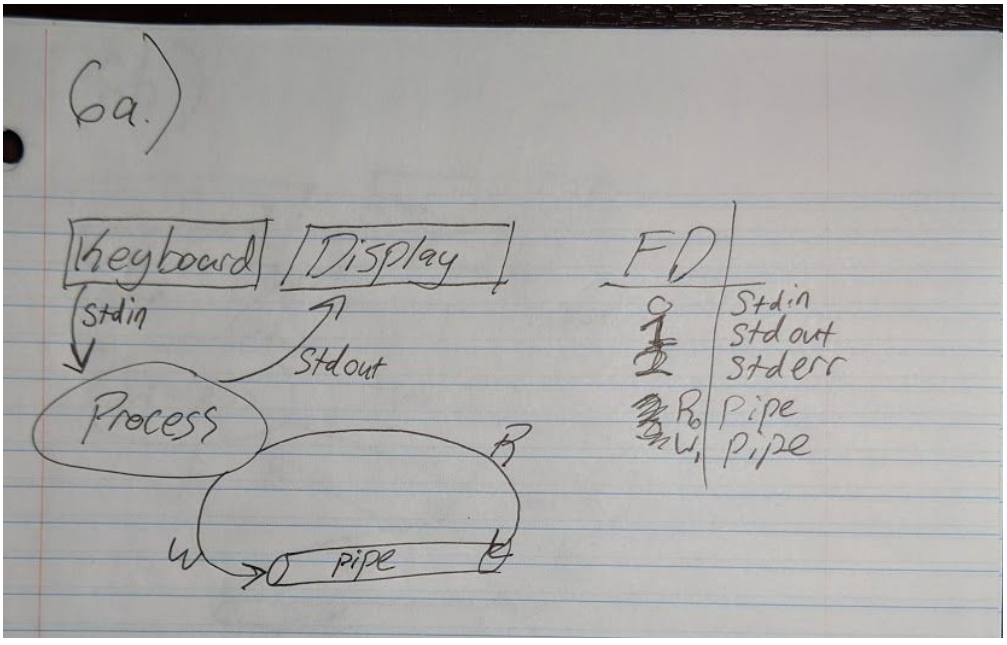
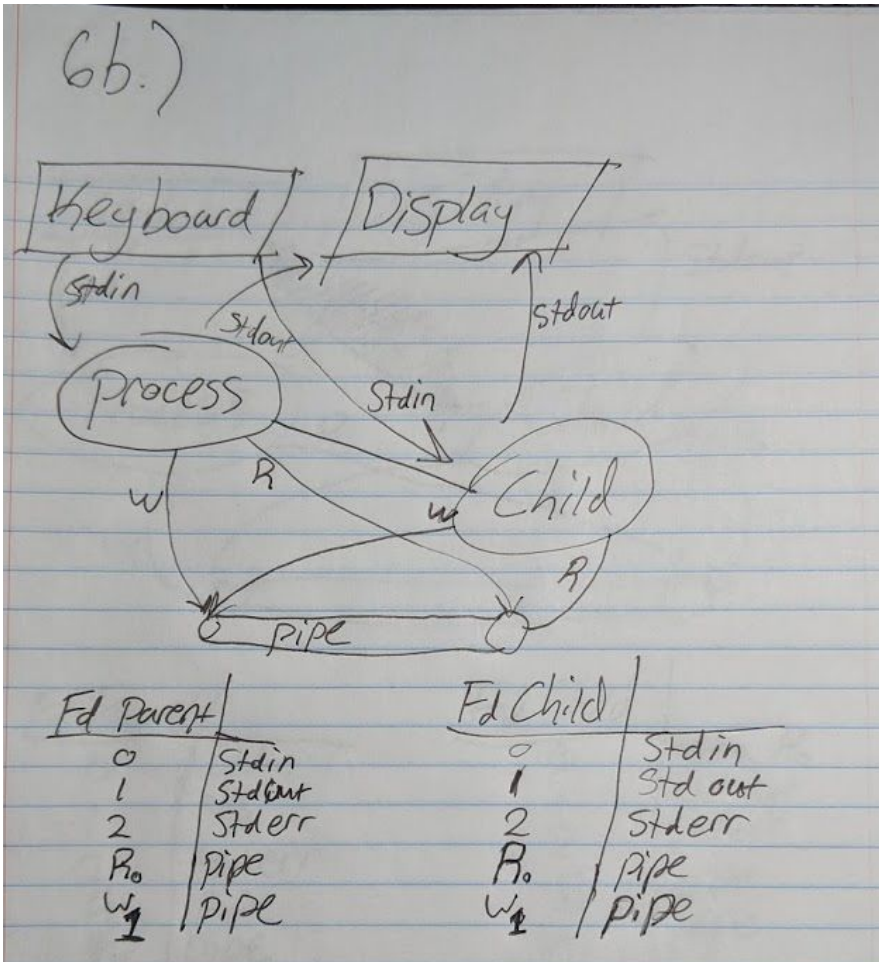
Brendan Cronan

Lab 3: Interprocess Communication

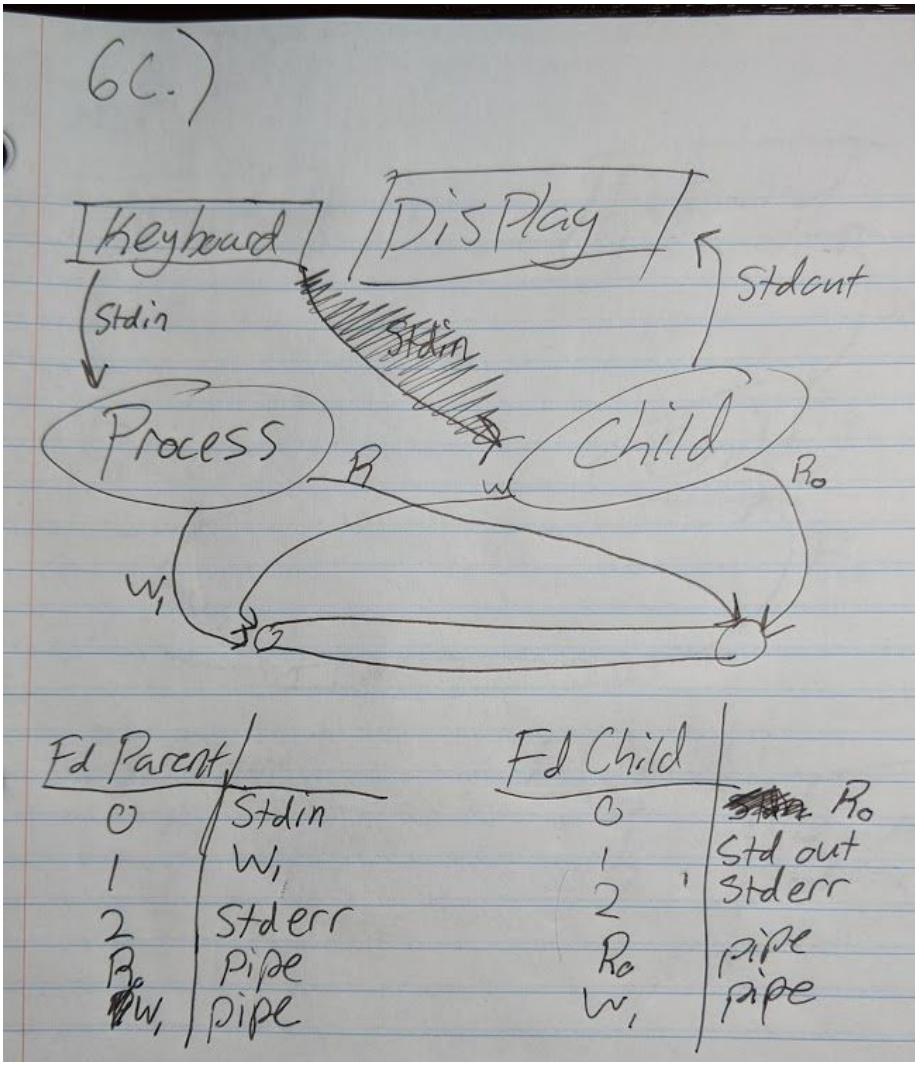
Github Link:https://github.com/brendan-cronan/Labs/tree/master/CIS452/Lab3

1. Order and content of the program output:
    a. The program prints wating… until the user interrupts the program.
    b. Then, the terminal prints “^C”
    c. The program prints “received an interrupt.”
    d. And then, “outta here.”
2. The answer above is a result of the following:
    a. Part a. is self evident.
    b. Pause() causes the program to sleep until an interrupt is received.
    c. Since signal() was called with a signal handler function, the signal unpauses the process and the handler function is invoked.
    d. It then prints that an interrupt has been received.
    e. Then, it waits for one second and exits the process.
3. The standard output of the child process will be directed toward the file temp.
4. The standard output of the child process will be directed to the standard output as per usual.
5. The program creates a pipe, forks a child process, the parent then reads the user’s input from stdin and then sends the string down the pipe to the child process and the child process prints it out.
6. Diagrams for each point in the program:
    a. 

b.

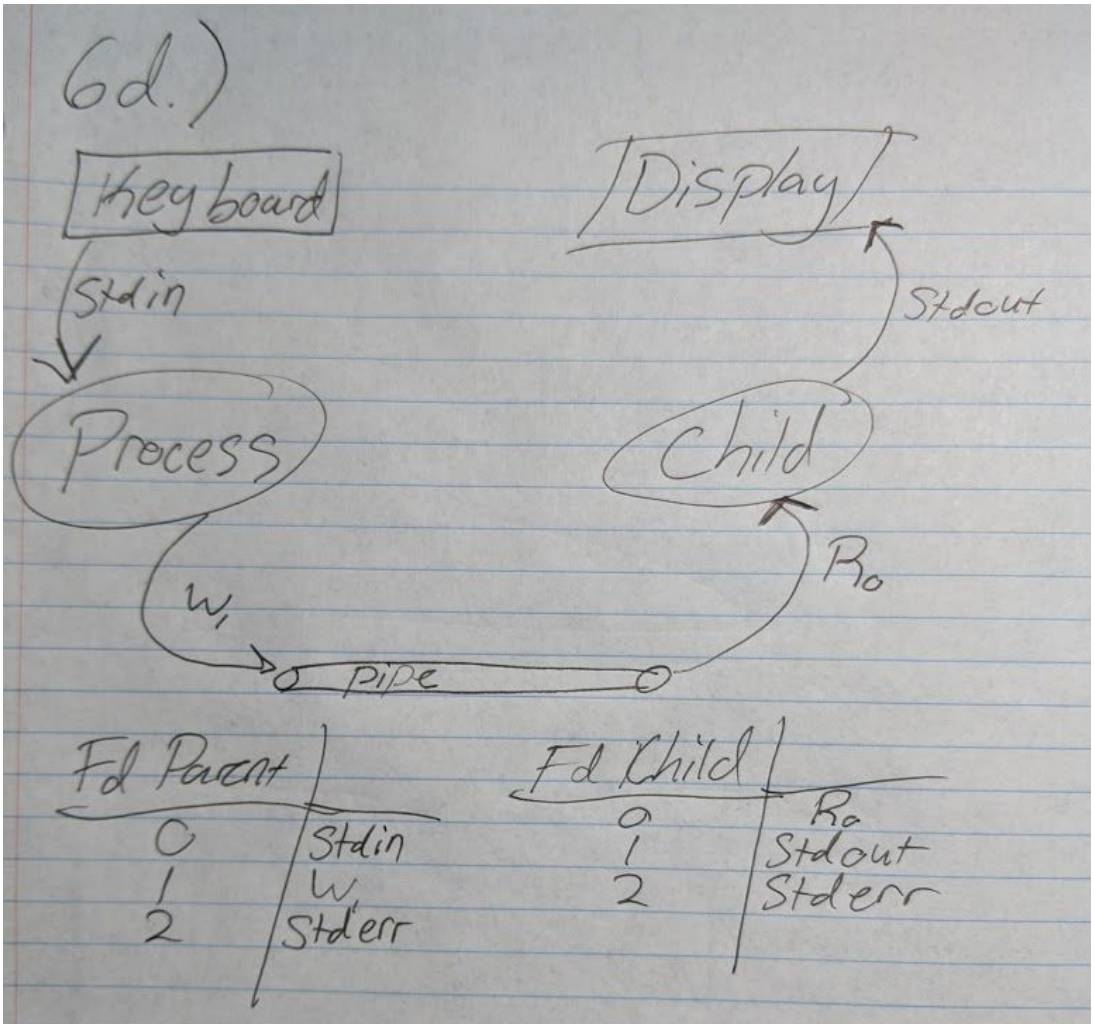

6b.)
Keyboard
Display
stdin
stdout
stdout
Process
Stdin
R
W
W
Child
R
Pipe
Fd Parent
0 Stdin
1 Stdout
2 Stderr
R0 Pipe
W1 Pipe
Fd Child
0 Stdin
1 Std out
2 Stderr
R0 Pipe
W1 Pipe

c.



i. Note, Please ignore the scribbled out sections, I did not want to re draw them and they were not intended to be there.

d.



7. Programming Assignment:
    a. See Appendix For Code
    b. As well as sample output.

APPENDIX

CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>

void sigHandler (int);

int main()
{

    pid_t pid, parent_pid;
    parent_pid= getpid();

    if ((pid = fork()) < 0) {
        perror ("fork failed");
        exit(1);
    }

   else if (pid > 0 ) {// if pid is NOT zero.  AKA: Parent.
     printf("PARENT PROCESS PID = %d\n",getpid());

     signal (SIGUSR1, sigHandler);
     signal (SIGUSR2, sigHandler);
     signal (SIGINT, sigHandler);

     while(1){

       printf ("\nwaiting...\n");
       pause();

     }



    }
    else{
      //CHILD PROCESS
```

```
      sleep(1);
      printf("CHILD PROCESS PID = %d\n",getpid());
      while(1){
        int rnd = random()%2;
        int signum;
        if(rnd == 0)
          signum = SIGUSR1;
        else
          signum = SIGUSR2;

        int wait_time = (random()%5)+1;
        //printf("waiting %d\n",wait_time);
        sleep(wait_time);
        kill(parent_pid,signum);

      }
    }


    return 0;
}

void sigHandler (int sigNum)
{
    //printf (" received an interrupt with signal number: %d\n",sigNum);
    if (sigNum == SIGUSR1){
      printf("received a SIGUSR1 signal(%d)\n",sigNum);
    }
    else if (sigNum == SIGUSR2){
      printf("received a SIGUSR2 signal(%d)\n",sigNum);
    }
    else if (sigNum == SIGINT){
      printf(" received an Interrupt signal.(%d)\nGuess I get to see what the process afterlife is
like...\n",sigNum);

      sleep (1);
      exit(0);
    }
}
```

SAMPLE OUTPUT

```
PARENT PROCESS PID = 64365

waiting...
CHILD PROCESS PID = 64366
received a SIGUSR2 signal(31)

waiting...
received a SIGUSR2 signal(31)

waiting...
received a SIGUSR2 signal(31)

waiting...
received a SIGUSR1 signal(30)

waiting...
received a SIGUSR2 signal(31)

waiting...
received a SIGUSR1 signal(30)

waiting...
received a SIGUSR1 signal(30)

waiting...
received a SIGUSR2 signal(31)

waiting...
received a SIGUSR1 signal(30)
```