Tyler King

Final Report

**Abstract:**

My project was to simulate a fight between a character (you) and an opponent in the rules of Dungeons and Dragons (DnD). For those who don’t know DnD is a highly number based table top role playing game where a person makes a character and role plays how they see fit. This program is just a simple simulation to point out the basics of the game.

**Introduction:**

This project simulates the battle sequence of Dnd, my objective was for people to come away with the understanding that you have to roll for your attack to hit and for the damage you deal. To hit the opponent you must roll a twenty sided dice and then get a value that is higher than the opponent’s armor class. If you roll under, you miss. The motivation behind this project was the fact that over this semester I met some really great new people and truly got into this game. I figured this project would be a fun way to combine academics and hobby.

**Background:**

In this program I am using pthreads, fifos, and hardware topics that we covered in the class this semester. In Character.cpp my pthread should be running continuously sending the data of name and wins into fifo1 and fifo2 respectively. In proj.cpp I have a while loop that is running indefinitely to read in from the fifos and check if the value of the previous wins is different. If it is, update the value and print it out. I used the two process approach to keep from having too much output clutter on one terminal. Where I could not get my fifos to properly send and read data the implementation is there.

**Implementation:**

As explained above, I used threads, fifos, and hardware. The hardware I used for the project was a raspberry pi2; a neat little piece of hardware that would allow me to generate four threads before seeing diminishing returns. The fifos were used to send the player name and their wins to the other program to be printed to the other terminal and then be updated continuously until the player dies or quits. The pthread was used in Character.cpp to run until the program terminated and to continuously send data to the other program via fifos 1 and 2.

**Experiments/Results:**

Unfortunately the only thing my program did not operate properly was my fifos sending and reading data. I wrote out many different pseudo code algorithms to try and figure out the issue but nothing would work. Though, I am running these two programs in c++ and I was highly unsure of my syntax and implementation of the fifos in the language. I tried to follow the method for fifos in C as much as possible but had to look to other sources to find the syntax specific to C++. The rest of the programs work fine as I have tested to make sure that when I read or write to my fifo that it would tell me so. That was an infinite loop and only for error checking. I ran the two programs together numerous times hoping all the different methods would end up having what I want pop up but after close to 50 different re-workings I elected to choose the best option for submission. The Character.cpp is the file that runs the main simulation, which runs without a hitch. You choose who you want to fight as and then go until you quit or die. The proj.cpp runs in an infinite loop and gathers data from the fifo but the name is whatever is at the memory location and the wins is correct at zero. The issue with that file though is that it is not being constantly updated.

**Discussion/Conclusion:**

To me, my observations did NOT make sense. I followed the same algorithm for fifos from my labs into this project and still the fifos did not come out running properly. What I was expecting is that Character.cpp would run the simulation and just print out the details of the fight. Whereas the other terminal which would be running prioj.cpp would be outputting your name and constantly updating how many wins over the opponent you have achieved. From this I have learned that C++ is horrible for Real Time programming and wish that C had more OOP capabilities and that a raspberry pi will not connect to anything in the labs. Both in terms of video output to a monitor (no hdmi in linux lab) and that the 16 pin connectors will not fit on the raspberry pi 2 and that plugging the pi via Ethernet into the lab will prevent the pi from accessing the nfsroot server with a firewall. Other approaches that I tried were actually the ladder two I just mentioned. I wanted to use UDP networking to send info from my pi to the board in the lab then incorporate FairCom and after I tried to configure my pi to work with the firewall with no avail, I tried to find a 16 pin connector and just use GPIO to send a signal to the ts7250 board to increment a value and then print it and send it to the database. Once those cards (figure of speech) were out I elected to change my project up a bit while keeping the same theme. Instead of using GPIO or UDP I elected for fifos and kept using threads and the different hardware.

**Appendices:**

**Proj.cpp**

#include <iostream>

#include <string>

#include <cstdlib>

#include <pthread.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <ctime>

#include <time.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/time.h>

#include <fcntl.h>

using namespace std;

int main()

{

cout<<"This is the reader program!"<<endl;

//pthread\_t thread1;

//pthread\_create(&thread1, NULL, thread, NULL);

//making the fifos

int fifoIn;

int fifoIn2;

char name[50];

int wins;

int win=-2; //setting wins2 to be different from start of other program

//opening fifos

fifoIn = open("fifo",O\_RDONLY);

fifoIn2 = open("fifo2",O\_RDONLY);

//infinite loop to keep reading in data

while(1)

{

//these are returning >0 just not the right values

read(fifoIn, &name, sizeof(name));

read(fifoIn2,&wins, sizeof(int));

//if read in value updates print

if(win!=wins)

{

win = wins;

cout<<name<<" has won "<<win<<" battles so far!"<<endl;

}

}

return 0;

}

**Character.cpp**

#include <iostream>

#include <string>

#include <cstdlib>

#include <pthread.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <ctime>

#include <time.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/time.h>

#include <fcntl.h>

using namespace std;

pthread\_t thread1;

//struct for data being sent to thread

struct person

{

char name[50];

int wins;

};

//global person

person stuf;

//function to be threaded

void\* thread(void\* data)

{

//making fifos

int fifoOut;

int fifoOutTwo;

mkfifo("fifo",O\_RDWR);

mkfifo("fifo2",O\_RDWR);

//opening fifos

fifoOut = open("fifo",O\_RDWR);

fifoOutTwo = open("fifo2",O\_RDWR);

//loop to write to fifo

while(1)

{

//these two are returning >0

write(fifoOut, &stuf.name, sizeof(string));

write(fifoOutTwo,&stuf.wins,sizeof(int));

}

pthread\_exit(0);

}

//function to get value 1-20

int atkRoll()

{

return rand()%20+1;

}

//function to get d to d\*v value. Ex a 1d6 can go from 1 to 6

int damDie(int d,int v)

{

int tot=0;

for(int i=0;i<d;i++)

{

tot += rand()%v+1;

}

return tot;

}

//class to hold damage values to be passed into damDie()

class Weapon

{

public:

string name;

int num;

int val;

//empty constructor

Weapon(){}

//set function

void set(string na,int n,int v)

{

name=na;

num=n;

val=v;

}

};

//class that is the character the person chooses. Holds stats that would be

//compared during runtime

class Player

{

public:

string name;

string job;

int hitMax;

int AC;

int str;

int strMod;

int dex;

int dexMod;

int con;

int conMod;

int intel;

int intelMod;

int wis;

int wisMod;

int cha;

int chaMod;

Weapon wep;

//constructor to make a Player

Player(string n,string j,int hit,int ac,int s,int d,int c,int i,int w,int ch,string na,int di,int v)

{

name=n;

job=j;

hitMax=hit;

AC = ac;

str = s;

dex = d;

con = c;

intel = i;

wis = w;

cha = ch;

strMod = (str-10)/2;

dexMod = (dex-10)/2;

conMod = (con-10)/2;

intelMod = (intel-10)/2;

wisMod = (wis-10)/2;

chaMod = (cha-10)/2;

wep.set(na,di,v);

}

};

//Same as player except does not have a class (job)

class Monster

{

public:

string name;

int hitMax;

int AC;

int str;

int strMod;

int dex;

int dexMod;

int con;

int conMod;

int intel;

int intelMod;

int wis;

int wisMod;

int cha;

int chaMod;

Weapon wep;

//constructor for monster

Monster(string n,int hit,int ac,int s,int d,int c,int i,int w,int ch,string na,int di,int v)

{

name=n;

hitMax=hit;

AC = ac;

str = s;

dex = d;

con = c;

intel = i;

wis = w;

cha = ch;

strMod = (str-10)/2;

dexMod = (dex-10)/2;

conMod = (con-10)/2;

intelMod = (intel-10)/2;

wisMod = (wis-10)/2;

chaMod = (cha-10)/2;

wep.set(na,di,v);

}

};

int main()

{

//seed

srand(time(0));

//gather first data for fifo

cout<<"What is your name? ";

cin>>stuf.name;

//make thread to send data continuously

pthread\_create(&thread1,NULL,thread,(void \*)NULL);

Player \*play;

//make monster

Monster mon("Aarakocra",13,12,10,14,10,11,12,11,"Javelin",1,6);

cout<<"Welcome to the Dungeons and Dragons training sim!"<<endl;

cout<<"Choose a character to try!"<<endl;

cout<<"a: Human Fighter, a deadly swordsman wielding a greatsword!"<<endl;

cout<<"b: Human Wizard, a powerful made who throws missles of magic!"<<endl;

char a;

bool go = false;

//loop until player chooses a character

while(!go)

{

cin>>a;

if(a=='a'||a=='A')

{

Player humFig("Human","Fighter",13,16,15,12,14,13,10,8,"Greatsword",2,6);

play = &humFig;

go = true;

}

else if(a=='b'||a=='B')

{

Player humWiz("Human","Wizard",8,13,8,14,13,15,12,10,"Magic Missle",4,4);

play = &humWiz;

go = true;

}

else

cout<<"That was not an option, try again: ";

}

cout<<"Now you are ready to do battle!"<<endl;

cout<<"Your oponents will be the "<<mon.name<<endl;

int myHp = play->hitMax;

int badHp = mon.hitMax;

int count=-1;

//loops till you die or quit

while(myHp>0)

{

count+=1;

char dec;

bool make = false;

myHp = play->hitMax;

badHp = mon.hitMax;

cout<<"An oponent arrives!"<<endl;

cout<<"What will you do?"<<endl;

//loop fight till there is a victor

while(badHp>0 && myHp>0)

{

make = false;

cout<<"a) Fight\tor\tb) Flee"<<endl;

while(!make)

{

cin>>dec;

if(dec=='a'||dec=='A')

{

make = true;

int atR = atkRoll();

if(atR>mon.AC)

{

cout<<"You rolled a "<<atR<<" which beat your enemie's AC!"<<endl;

int dmg = damDie(play->wep.num,play->wep.val);

if(atR==20)

{

cout<<"You delt a critical hit!!!"<<endl;

dmg\*=2;

}

cout<<"You deal "<<dmg<<"damage to the enemy!"<<endl;

badHp -= dmg;

}

else

cout<<"Your roll did not beat the enemy's AC, you miss."<<endl;

}

else if(dec=='b'||dec=='B')

{

myHp = 0;

cout<<"You died running off a cliff. Bummer. Game over."<<endl;

make = true;

}

else

cout<<"That was not an option, try again: ";

if(badHp>0 && dec!='b'&& dec!='B')

{

int atR = atkRoll();

if(atR>play->AC)

{

cout<<"Enemy rolled a "<<atR<<" which beat your AC!"<<endl;

int dmg = damDie(mon.wep.num,mon.wep.val);

if(atR==20)

{

cout<<"They delt a critical hit!!!"<<endl;

dmg\*=2;

}

cout<<"Enemy delt "<<dmg<<" damage to you!"<<endl;

myHp -= dmg;

}

else

cout<<"Enemy missed."<<endl;

}

}

}

if(myHp>badHp)

cout<<"You won the fight!"<<endl;

else

cout<<"You are so dead"<<endl;

}

//update into for fifo out

stuf.wins = count;

//print because the other program wasnt lining up

cout<<"You won "<<stuf.wins<<" fight(s)."<<endl;

return 0;

}