cs201 – sum 2020

Assignment 3 – (40 points)

In this assignment, you will create your own code file called **dicegame.c**. Upload this to the d2l assignment folder before the due date. You may want to look at fork4.c example code in week 6 for how to deal with multiple child processes.

Please note the requirements and follow them for full credit.

Dice Game

In this program, the child processes are going to randomly roll two dice and the send back via the return value to the parent process the sum of the two dice. The parent process will reap each child, get the return value, and determine if the child won, loss or did neither (a draw). The child won if the sum is either 7 or 11. The child lost if the sum is either 2, 3 or 12. Any other value is a 'draw'.

Requirements

Need to use **fork()** to create 4 child processes. Each child process will roll two dice randomly using **rand()** function. The parent needs to use **waitpid()** to reap each child and determine a win/loss/draw. Have each child prints out it's pid number and the two rolls of the dice before returning. Have the parent also print out the child pid and the sum so we can verify that the correct values were gotten from the right child. **The parent will print out the sum, determine and print out whether the child had a win/lose/draw, don't print this in the child process's code.**

Using the rand() function in stdlib.h library (include this)

You will need to generate random numbers between 1 and 6 for each dice. Also, you need to make sure that these random numbers are unique for each child process and each time the program is run. To do this, each child process will need to 'seed' it's own random number generator with a different value.

Typically you will use srand() function with a call to the time() function (the time() function is in the time.h library. Make sure you include this). However, since the time between launching child processes is shorter than a second, each child ends up with the same value from the time() function. To make sure the time is different, use the for loop counter to make the time value different. I used this is my code:

```
if(pid == 0) // in child process
{
     srand((unsigned)time(0)*i);
      ...
```

Note that you will need to do this inside the code for each child separately.

Now, you can use the rand() function to get a value from 0 to MAX_INT. This is way too big so to cut it down, use a modulo (%) 6 and add 1 to get a value between 1 and 6:

```
     dice1 = (rand() % 6) + 1;
```

Make sure you get two values randomly and then sum them. Return the sum in a return statement for the parent to pick up. You could get one value between 2 and 12 but the probability distribution tends to be different so it doesn't properly represent the rolling of two dice.

<u>Example Output</u>

Below is some output examples. If you want 'spice it up' and print out a more elaborate messages for win, loss and draw (maybe even some ASCII art) go ahead and do so as long as the message is correct.

NOTE: the child is printing the dice rolls. The parent is printing the sum and whether the child wins/loses/draws.

```
-----------------------
EXAMPLE RUN #1:
-----------------------
Child 10025 rolls a 3 and a 3
child 10025 gets a 6
child 10025 gets a draw. MEH.
Child 10024 rolls a 6 and a 2
child 10024 gets a 8
child 10024 gets a draw. MEH.
Child 10026 rolls a 6 and a 6
child 10026 gets a 12
child 10026 loses! BOOHOO..
Child 10023 rolls a 3 and a 4
child 10023 gets a 7
child 10023 wins! YAY!
-----------------------
EXAMPLE RUN #2:
-----------------------
Child 11399 rolls a 2 and a 3
Child 11398 rolls a 1 and a 6
Child 11397 rolls a 3 and a 5
child 11398 gets a 7
child 11398 wins! YAY!
child 11397 gets a 8
child 11397 gets a draw. MEH.
child 11399 gets a 5
child 11399 gets a draw. MEH.
Child 11400 rolls a 1 and a 2
child 11400 gets a 3
child 11400 loses! BOOHOO..
```