**CS 370 Fall 2016**

Programming Assignment 2: One parent with three child processes draft v9/24/16 11:30 AM

Deadline Sept 30, 5 PM. *Late deadline with penalty Oct 2, 5 PM*

**Purpose:** Write program Initiator that reads a text file with a list of numbers, and forks three children that will run programs Mean, Median and Mode that compute mean, median and mode respectively, and return +1, 0 or -1 depend on whether the result is positive, negative or zero. The returned values serve as status codes that are used by the Initiator to report the outcome.

**Description of assignment**:

You will be creating four programs: Initiator, Mean, Median, Mode.

**Initiator**: The Initiator will read a text file *Nums.txt* and sends it to the child process for further processing. The Initiator is responsible for using the

(1) fork() command to launch another process.

(2) exec() command to replace the program driving this process, while also supplying the arguments that this new program (Mean, Median or Mode) needs to complete its execution.

(3) wait() command to wait for the completion of the execution of the process, and WEXITSTATUS(status) to obtain the status from the three child programs.

The Initiator then prints out the message: "Mean is ____, Median is ____ and the Mode is ___." where the blank is positive/negative/zero.

**Mean, Median and Mode**: Each of these receive the "file with list of numbers" as argument and compute mean, median and mode respectively.

Following are the algorithms to find out the mean, median and mode:

Mean:

Step 1: Read numbers from file.

Step 2: Count the numbers.

Step 3: Add the numbers.

Step 4: Divide the sum by the count.

Median:

Step 1: Read numbers from file into an array.

You may write your own code for the steps below, or use the code provided here

Step 2: Sort the array in ascending order.

Step 3: If the number of elements is odd, the element in the middle is the median. If the number is even, the median is the average of the two middle elements.

Mode:

Step1: Read numbers from file.

Step2: Determine number with highest frequency, which will be your mode.

If there are multiple values of Mode, pick the one encountered first.

All print statements must indicate the program that is responsible for generating them. To do this, please prefix your print statements with the program name. The example section below depicts these sample outputs.

A good starting point is to implement the functionality for the Mean, Median and Mode programs, and then write the code to manage its execution using the Initiator program.

**Requirements of Task:**

1. The Initiator must take the text file Nums.txt as input and send it to the child processes. Each of the other three programs must accept the file as an argument.

2. The Initiator should spawn 3 processes using the fork() command and must ensure that it completes one full cycle of fork(), exec() and wait() for a given process before it moves on to spawning a new process.
3. Once it has used the fork() command, the Initiator will print out the process ID of the process that it created. This can be retrieved by checking the return value of the fork() command.
4. Child-specific processing immediately following the fork() command then loads the Mean/Median/Mode program into the newly created process using the exec() command. This ensures that the forked process is no longer a copy of the Mean/Median/Mode. This exec() command should also pass 1 argument to the Mean/Median/Mode program. For this assignment, it is recommended that you use the execlp() command. The man page for exec (man 2 exec) will give details on the usage of the exec() family of functions.
5. When the Mean/Median/Mode is executing it prints out its process ID; this should match the one returned by the fork()command in step 3.
6. The Mean/Median/Mode then determines whether the result is positive, negative or zero.
7. Mean/Median/Mode should return an exit code +1 for positive result, -1 for negative result and 0 if the result is zero. These correspond to the standard UNIX exit codes. Each exit code received by the Initiator should be printed. You can use the WEXITSTATUS() macro to determine the exit status code (see man 2 wait).
8. Parent-specific processing in the Initiator should ensure that the Initiator will wait() for each instance of the child-specific processing to complete.

**Submission**: Use Canvas to submit a single .tar file named PA2.tar that contains:

- All .c and .h files related to the assignment (please document your code),
- a Makefile that performs both a make clean as well as a make all,
- a README.txt file containing a description of each file and any information you feel the grader may need.

**Grading:** The grading will also be done on a 100 point scale. The points are as follows: 10 points each for each of the tasks i.e. Task 1-8 (80 points), 20 point for getting the rest of the things right.

**Example Output**: see here.

Notes:

1. You are required to work alone on this assignment.
2. Late Policy: There is a late penalty of 10% per-day for up to a maximum of 2 days.
3. You may assume that text file Nums.txt contains 10 integers, one in each line, in older to keep the assignment simple.