# HW2: Programming Assignment v. 2.06.2022 11PM

Working With Parent and Child Processes

The objective of this assignment is to write and test a program that creates three child processes and obtains status information from them. It uses fork(), exec(), wait() and WEXITSTATUS(status) commands.

Due Date: Thursday, February 10,2022, 11:00 pm

Extended Due Date with 20% penalty: Friday, February 11, 2022 ,11:00 pm

## 1. Purpose

Write a C program called ***Generator*** that reads strings from a file, whose name will be provided as a command line argument to the program. Generator forks three child processes for each line, that will run programs *Lucas, HexagonalSeries* and *HarmonicSeries*.

## 2. Description of assignment

You will be creating four programs: *Generator.c, Lucas.c, HexagonalSeries.c and HarmonicSeries.c*.

**Generator.c**: *Generator.c* takes **one** mandatory argument which is the name of the .txt file. ***Generator.c*** will read all the lines from the given **.txt** file and send each line value to the first child process and then the other child processes will use the result obtained from the previous child Process as their input argument.

a. The *Generator* is responsible for executing the fork() functions to create the child processes.
b. Each created child Process runs the exec() function to run the program needed (***Lucas,*** *HarmonicSeries or HexagonalSeries*).The *Generator* should provide the required argument for the first child process to complete its execution. The second child process should use the result returned by the first child process as an argument and the third child process will use the result obtained from the second child process as input.
c. The wait() function is used to wait for the child processes to complete its execution. The Macros WIFEXITED(status) and WEXITSTATUS(status) are used to obtain the result(as an eight-bit integer) from the three child processes.

The *Generator* saves the status(result) obtained from each child process. After the completion of the execution of all the processes the *Generator* will print the outputs from each child processes executed.

Lucas.c, HexagonalSeries.c, HarmonicSeries.c: Each of these programs receives one line as an argument. The programs are used to print the sum of the respective series of numbers using the argument provided and return a result.

**Note:** 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.

*Generator.c* should indicate the process ID of the child process it created and the child processes (*Lucas, HexagonalSeries, HarmonicSeries*) should indicate their own process ids. The example output that is shown below depicts the expected format of the output and must be strictly adhered to.

**Hint:** A good starting point is to implement the functionality for the *Lucas.c, HarmonicSeries.c, and HexagonalSeries.c* programs, and then write the code to manage its execution using the Initiator program.

## 3. Input and Output

For example, the "input*.txt*" file could contain the strings "10" and "3" on two separate lines which would mean to run all three child processes with the first input to Lucas being 10 and then 3. Use fopen() function to read the string from the file.

Notes:
- You can assume that the input numbers to be between 1(inclusive) and 20 (inclusive).
- Note that the end of a line is indicated *differently* in text files for Windows and Linux system. So please test your program in a linux environment e.g. on the CS Machines.

## 4. Task Requirements

1. The *Generator* must read the characters from the .txt file, the name of which will be passed as an argument to it. Then send each line (String), one at a time, to the first child process. The first child process must accept the string as an argument.
2. The *Generator* should spawn up to 3 processes using the fork() function for each line from the input file and must ensure that one full cycle of fork(), exec() and wait() is completed for a given process before it moves on to spawning a new process.
3. Once it has used the fork() function, the *Generator* will print out the process ID of the process that it created. This can be retrieved by checking the return value of the fork() function.
4. Child-specific processing immediately follows. The exec() function loads the *Lucas /HexagonalSeries/HarmonicSeries* program into the newly created process. This exec() function should also pass the value to the *Lucas/HexagonalSeries/HarmonicSeries* program. For this assignment, it is recommended that you use the execlp() function. The "man" page for exec (search for "man exec" on google or if you are on linux you can type that into a shell) gives details on the usage of the exec() family of functions.
5. When the *Lucas/HexagonalSeries/HarmonicSeries* program is executing, it prints out its process ID; this should match the one returned by the fork() function in step 3.
6. *The Lucas/HexagonalSeries/HarmonicSeries* program then prints the respective numbers (refer to 6.d,e,f) and returns the result (refer 6.a,b,c).
   a. *Lucas* should return: (assume n is the argument provided)
      *n if sum of the first n Lucas numbers is greater than 50.*
      *Sum of the first n Lucas numbers if the sum is less than 50.*
   b. *Hexagonal* series should return: (assume n is the argument provided)
      *n if sum of the first n Hexagonal numbers is greater than 100.*
      *Sum of the first n Hexagonal numbers if the sum is less than 100.*
   c. *Harmonic* series should *return* the sum of the first n harmonic numbers.
   d. *Lucas* should *print* the sum of the first n Lucas numbers (assume n is the argument provided) and the nth Lucas number.
   e. *HexagonalSeries* should *print* the sum of the first n Hexagonal numbers and the nth Hexagonal number(assume n is the argument provided).
   f. *HarmonicSeries* should *print* the sum of the first n numbers of the harmonic series (assume n is the argument provided).
7. *Lucas/HexagonalSeries/HarmonicSeries* program should return the respective results after executing. Each result is received and stored by the *Generator*. The stored value is used as the argument for the next child process that is forked. You can use the WEXITSTATUS() macro to determine the exit status code (see man 2 wait).
   **Note:** Please be careful of the data types of the input arguments and the returned results from each child process.
8. Parent-specific processing in the Generator should ensure that the Generator will wait() for each instance of the child-specific processing to complete. Once all the processes are complete output the result as mentioned in 6.a,b,c to the terminal.

**IMPORTANT:** Your program *must* fork(), exec(), wait() the programs in this order for each input provided: *Lucas-HexagonalSeries-HarmonicSeries.*

## 5. Example Outputs

1. This is the output when analyzing the file input.txt which contains the strings on two separate lines:

```
3
10
```

(Note: your process IDs are likely to be different)

```
[Generator] [1266583]: Waiting for the child process 1266584
[Lucas] [1266584]: The sum of the first 3 numbers of the Lucas series is 6
[Lucas] [1266584]: The nth number in the lucas series is 3
[Generator] [1266583]: The child process 1266584 returned 6
[Generator] [1266583]: Waiting for the child process 1266585
[HexagonalSeries] [1266585]: The sum of first 6 hexagonal numbers is:161
[HexagonalSeries] [1266585]: The nth number in the Hexagonal series is 66
[Generator] [1266583]: The child process 1266585 returned 6
[Generator] [1266583]: Waiting for the child process 1266586
[HarmonicSeries] [1266586]: The sum of the first 6 numbers of the harmonic
series is 2.450000
[Generator] [1266583]: The child process 1266586 returned 2
[Generator] [1266583]: The lucas child process returned 6
[Generator] [1266583]: The Hexagonal child process is 6
[Generator] [1266583]: The sum of the first n Harmonic numbers is 2
[Generator] [1266583]: Waiting for the child process 1266587
[Lucas] [1266587]: The sum of the first 10 numbers of the Lucas series is 198
[Lucas] [1266587]: The nth number in the lucas series is 76
[Generator] [1266583]: The child process 1266587 returned 10
[Generator] [1266583]: Waiting for the child process 1266588
[HexagonalSeries] [1266588]: The sum of first 10 hexagonal numbers is:715
[HexagonalSeries] [1266588]: The nth number in the Hexagonal series is 190
[Generator] [1266583]: The child process 1266588 returned 10
[Generator] [1266583]: Waiting for the child process 1266589
[HarmonicSeries] [1266589]: The sum of the first 10 numbers of the harmonic
series is 2.928968
[Generator] [1266583]: The child process 1266589 returned 2
[Generator] [1266583]: The lucas child process returned 10
[Generator] [1266583]: The Hexagonal child process is 10
[Generator] [1266583]: The sum of the first n Harmonic numbers is 2
```

## 6. What to Submit

Use the CS370 *Canvas* to submit a single .zip or .tar file that contains:

- All .c and .h files listed below and descriptive comments within,
  - *Generator.c*
  - *Lucas.c*
  - *HexagonalSeries.c*
  - *HarmonicSeries.c*
- A Makefile that performs both a *make build* as well as a *make clean.* Note that you will have four executables.
- A README.txt file containing a description of each file and any information you feel the grader needs to grade your program, and answers for the 5 questions

For this and all subsequent assignments, you need to ensure that you have submitted a valid .zip/.tar file. After submitting your file, you can download it and examine to make sure it is indeed a valid zip/tar file, by trying to extract it.

**Filename Convention:** The archive file must be named as: <FirstName>-<LastName>-HW2.<tar/zip>. E.g. if you are John Doe and submitting for assignment 2, then the tar file should be named John-Doe-HW2.tar

## 7. Grading

The assignments must compile and function correctly on machines in the CSB-120 Lab. Assignments that work on your laptop on your flavor of Linux/Mac OS X/Windows, but not on the Lab machines are considered unacceptable.

The grading will be done on a 100 point scale. The points are broken up as follows:

| Objective | Points |
|---|---|
| Correctly performing Tasks 1-8 (10 points each) | 80 points |
| Descriptive comments | 5 points |
| Compilation without warnings | 3 points |
| Questions in the README file | 7 points |
| Providing a working Make file | 5 points |

**A readme file should be created by you it should have the following:**

- Output as shown above in Example outputs.
- Answers to the questions provided below

**Questions:** (To be answered in README file. Each question worth 1 point unless mentioned)

1. What is the return value for the fork() in a child process?
2. Give a reason for the fork() to fail?
3. How many of the least significant bits of the status does WEXITSTATUS return?
4. Which header file must be included to use the WEXITSTATUS.

5. In this program, are we doing a sequential processing or a concurrent processing with respect to the child processes? Sequential processing is when only one of the child processes is running at one time, and concurrent processing is when more than one child process can be running at the same time.
6. Is it possible to have any anomalies in the output from child process and the output from parent process. Provide a reason for that possible situation. (2 points)

## 8. Late Policy

Click here for the class policy on submitting late assignments.

**Notes:**

1. You are required to work alone on this assignment.
2. Late Policy: There is a late penalty of 20%. The late period is 24 hours.
3. Note that although WEXITSTATUS(status) is primarily intended for returning the status to the parent, here we are exploiting this capability to transmit the result from the child programs to the parent program.

**Revisions**: Any revisions in the assignment will be noted below.

2/6/2022: Task Requirements corrected.

6b corrected:

n if sum of the first n Hexagonal numbers is greater than 100.

Sum of the first n Hexagonal numbers if the sum is less than 100.

6f corrected: delete some words.

HarmonicSeries should print the sum of the first n numbers of the harmonic series ~~and the nth harmonic number~~.( assume n is the argument provided).