# MultiProcessing Assignments - 1

1. Write a multiprocessing application to perform the following.
2. Application to receive an input file name as command line argument.
3. Store the filename in a buffer.
4. Create a child using fork().
5. Display the process id of both parent and child
6. Let child prompt and read 2 lines from user and write to input file and exit.
7. Let parent after fork() do the following

* wait for child to complete.
* open input file name, read and display the contents

1. Add a function below and call it from parent after wait()

/\*Using WIFEXITED macro check if the child has exited and in such case extract exit code and display,

Else, use WIFSIGNALLED macro to check if the child was terminated using signal (say using SIGINT) then the retrieve and display the signal number

which caused the exit \*/

process\_display\_exit\_code(int exitstatus)

[Hint: to test above function you may send a kill signal to child process]

1. Refer the code snippet below and answer the queries.

int g\_value =10; /\* global variable \*/

int main()

{

int pid;

int l\_value =5;

printf(“Writing a sample code\n”);

pid = fork(); /\* fork() returns 0 to child process and process id of child to parent process\*/

if(0 == pid)

{

printf(“For child Local variable value=%d\n and global variable value=%d\n”,l\_value,g\_value);

exit(0);

}

else

{

printf(“For parent Local variable value=%d\n and global variable value=%d\n”,l\_value,g\_value);

}

printf(“Code common for both parent process and child process\n);

return 0;

}

1. What will be the output of parent process and child processes?
2. Find out whether the value of local variable and global variable value will be same for both parent process and child process
3. Will the order of execution be same always or could be different? Will it impact the output?
4. Why the first printf() statement will be executed only by parent process and not by child process?

Optional:

1. Extend the solution to Q1 to
2. add separate cleanup handlers for parent and child

[Hint : register handlers using atexit()]

1. In cleanup handlers, display pid and parent pid of the exiting process
2. Implement a function below to use system call sendfile() to copy from input file descriptor to output file descriptor

size\_t sendfile\_copy(<destination fd>, <source fd>, <offset position in input buffer source>, <number of bytes to copy>)

1. Calculate the execution time for the above step b)
2. Implement a function below to use system call read() and write() to copy from input file descriptor to output file descriptor

size\_t read\_write\_copy(<buffer containing data to be written>, <input buffer source>, <offset position in input buffer source>, <number of bytes to copy>)

1. Calculate the execution time for the above step d)
2. Compare execution time in c) and e). Which one is better?
3. Run program at console, verify if the “fout.txt” is same as “fin.txt”
4. Run command below and capture and view the internal system calls

Strace <prgname>

1. Try the above program using a non-existing input file name. Capture the errors thrown and display using strerror(), perror()