# MultiProcessingAssignments - 3

**ExecXX() calls**

1. Refer the codesnippet below and answer the queries

/\* creates a child process which is executing “ls” executable

int create(char \*pgm, char \*argv[])

{

int pid;

pid = fork();

If (0 == pid)

{

execvp(pgm,argv);/\* executes pgm searching for the path in “PATH” environment variable \*/

perror(“execvp”); /\* the execvp function returns only if an error occurs \*/

exit(0);

}

else

{

wait(NULL);

printf(“Inside parent process\n”);

}

}

int main()

{

char \*argv[]={“ls”,”-l”,NULL}; /\*List of NULL terminated arguments \*/

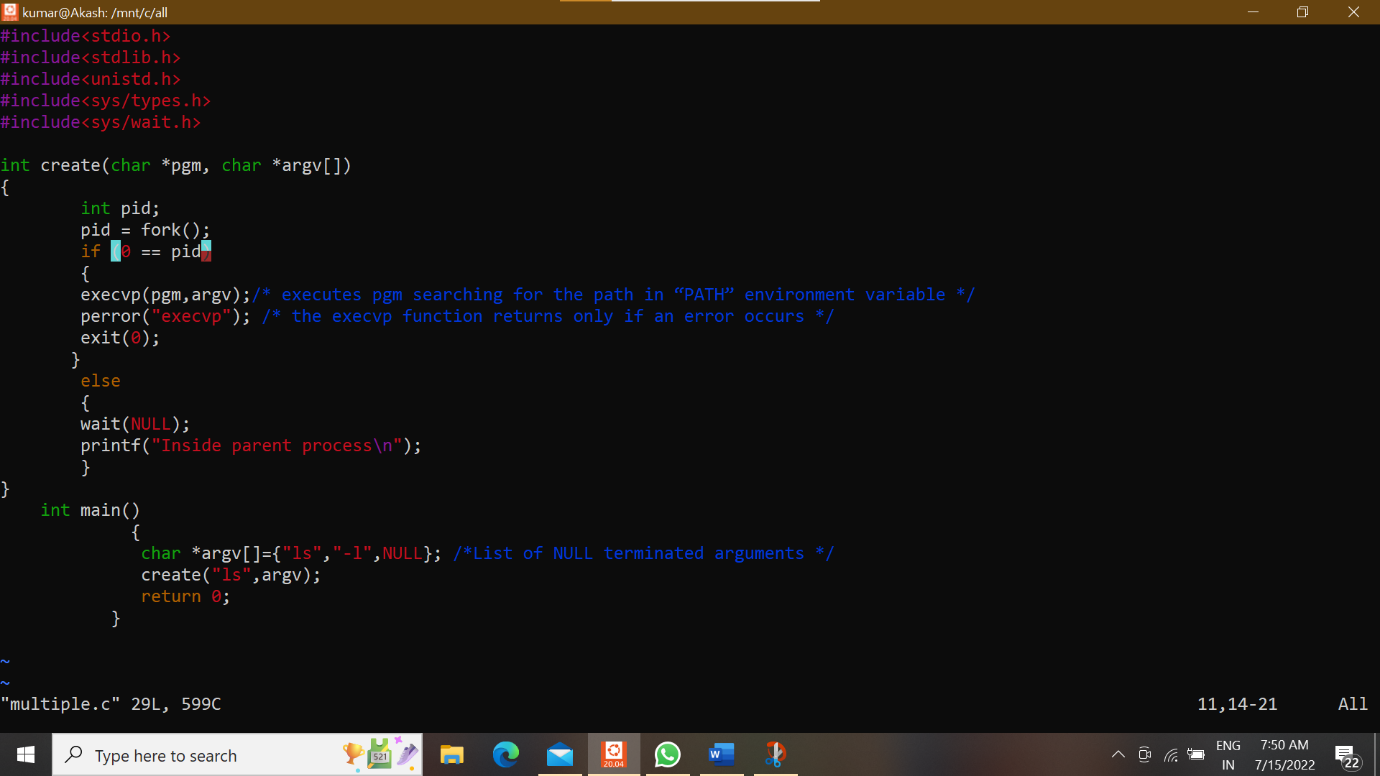
create(“ls”,argv);

return 0;

}

Queries:

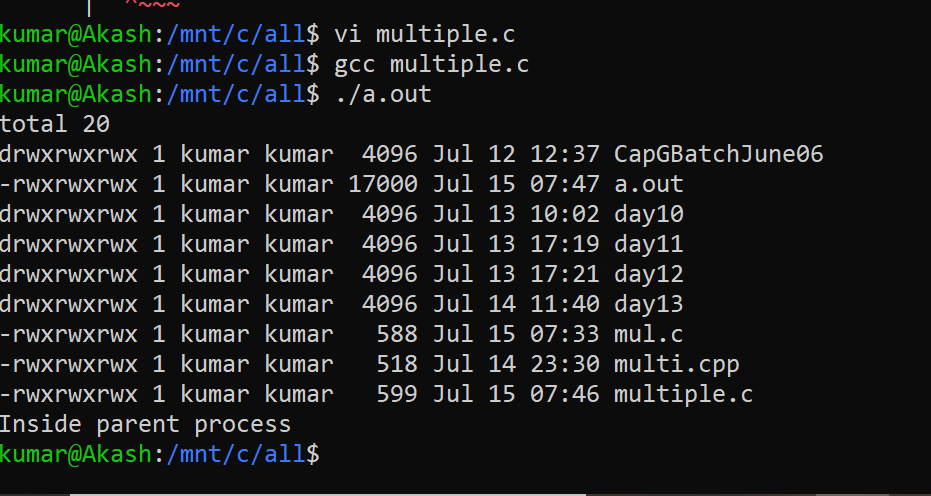
1. What is the output of parent process and child process?



1. How does execvp call work?

The execvp function is most commonly used to **overlay a process image that has been created by a call to the fork function**.

Identifies the location of the new process image within the hierarchical file system



1. What would be the output if execvp() call is invoked just after *pid=fork()* statement.
2. How many processes would be created and what would be the output if execvp() is invoked before *pid =fork*?
3. When would you use execXX() ?

The exec family of functions replaces the current running process with a new process.

It can be used **to run a C program by using another C program**. It comes under the header file unistd.

1. Write a program that received a program name and uses an exec() system call to execute a program.

For example if the name of your executable is myexecutable, then if you invoke as follows:

***myexecutable mycmd a b***

then, mycmd will be executed with command line parameters a & b by the program myexecutable

(Note: mycmd is a user defined program)

1. 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 stepd)
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()