* fork() system call is used to create child processes in a C program. fork() is used where parallel processing is required in your application.
* The fork() system function is defined in the headers **sys/types.h** and **unistd.h**.
* wait() system call is used to wait in the parent process for the child process to finish.
* To finish a child process, the exit() system call is used in the child process.
* The wait() function is defined in the header **sys/wait.h** and the exit() function is defined in the header **stdlib.h**.
* System call fork() is used to create processes. It takes no arguments and returns a process ID.
* The purpose of fork() is to create a new process, which becomes the child process of the caller.
* After a new child process is created, both processes will execute the next instruction following the fork() system call.
* Therefore, we have to distinguish the parent from the child. This can be done by testing the returned value of fork():
* If fork() returns a negative value, the creation of a child process was unsuccessful.
* fork() returns a zero to the newly created child process.
* fork() returns a positive value, the process ID of the child process, to the parent. The returned process ID is of type pid\_t defined in sys/types.h.
* Normally, the process ID is an integer. Moreover, a process can use function getpid() to retrieve the process ID assigned to this process.
* pid\_t fork(void);
* The fork() system function does not accept any argument. It returns an integer of the type **pid\_t**.
* On success, fork() returns the PID of the child process which is greater than 0. Inside the child process, the return value is 0. If fork() fails, then it returns -1.
* **C System Functions in Exec Family:**
* The exec function families are defined in the header unistd.h
* The exec system call is used to execute a file which is residing in an active process. When exec is called the previous executable file is replaced and new file is executed.
* using exec system call will replace the old file or program from the process with a new file or program. The entire content of the process is replaced with a new program.
* The user data segment which executes the exec() system call is replaced with the data file whose name is provided in the argument while calling exec().
* The new program is loaded into the same process space. The current process is just turned into a new process and hence the process id PID is not changed, this is because we are not creating a new process we are just replacing a process with another process in exec.
* The exec() call replaces the entire current contents of the process with a new program. It loads the program into the current process space and runs it from the entry point.
* **Stat System Call:**
* Stat system call is a system call in Linux to check the status of a file such as to check when the file was accessed. The stat() system call actually returns file attributes. The file attributes of an inode are basically returned by Stat() function. An inode contains the metadata of the file.
* An inode contains: the type of the file, the size of the file, when the file was accessed (modified, deleted) that is time stamps, and the path of the file, the user ID and the group ID, links of the file, and physical address of file content.
* **Syntax of C Stat system call:**
* To use the stat system call in C programming language, you have to include the following header file:
* #include <sys/stat.h>
* Stat is used to get the status of a file.
* The return type of the function in int, if the function is executed successfully, 0 is returned if there are any errors, -1 will be returned.
* st\_dev: It is the ID of device in which we have our file residing currently.
* st\_rdev: This field describes that a particular file represents a particular device.
* st\_ino: It is the inode number or the serial number of the file. As it is an index number so it should be unique for all files
* st\_size: st\_size is the size of the file in bytes.
* st\_atime: It is the last time or the recent time at which the file was accessed.
* st\_ctime: It is the recent time at which the status or the permissions of the file was changed.
* st\_mtime: It is the recent time at which the file was modified.