UNIX FORK COMMAND

In computing, the fork () command is the go-to method of creating processes in Unix-Like operating systems. The command creates a new process by duplicating the calling process. The new process is called the **child process**. The calling process is referred to as the **parent process**. Both the parent and child process run in separate memory spaces. When the parent process closes or crashes for a given reason, the child process is also killed. After the fork () command is called, the operating system assigns a unique ID in order to keep track of all the processes.

The fork () command does not take any parameter and return an integer value as following: 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.

The function below elaborates the explanation above:

#include <unistd.h>

pid_t fork (void);

Returns: o in child, process ID of child parent, -1 on error.

Fork returns O in the child since child has only one parent, while it returns 1 in parent process since parent can have any number of children. Ones the fork command is called, it will

undergo various states as follows: **Ready/New** – indicating the process is ready to run. The **Run** state is initiated where process is chosen by CPU for execution. The process then enters theBlocked/wait state when it requests access to I/O from critical region. **Terminated** is the next state which implies that the process is killed or deleting the PCB.