ASSIGNMENT NO. :02
TITLE : UNIX Process Control

OBJECTIVE :

Process control system calls: The demonstration of fork, execve and wait system calls along with zombie and orphan states.

a) Implement the C program in which main program accepts the integers to be sorted. Main program uses the fork system call to create a new process called a child process. Parent process sorts the integers using merge sort and waits for child process using wait system call to sort the integers using quick sort. Also demonstrate zombie and orphan states.

b) Implement the C program in which main program accepts an integer array. Main program uses the fork system call to create a new process called a child process. Parent process sorts an integer array and passes the sorted array to child process through the command line arguments of execve system call. The child process uses execve system call to load new program that uses this sorted array for performing the binary search to search the particular item in the array.

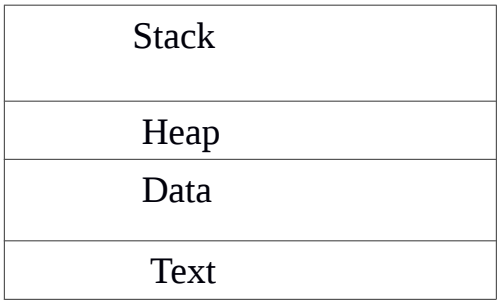
SOFTWARE REQUIREMNTS :

1)LINUX(Ubuntu)
2)GNU C Compiler

THEORY :

Process :

A Process is a program in execution. The execution of a process must progress in a sequentional fahtion. A process is defined as an entity which represents the basic unit of work to be implemented in the system. When a program is loaded into the memory and it become a process, it can be divided into four part or sections :- Stack , Heap , text , Data.



Process Life Cycle:

1. **New:**-This is the initial state when a process is first started/created.

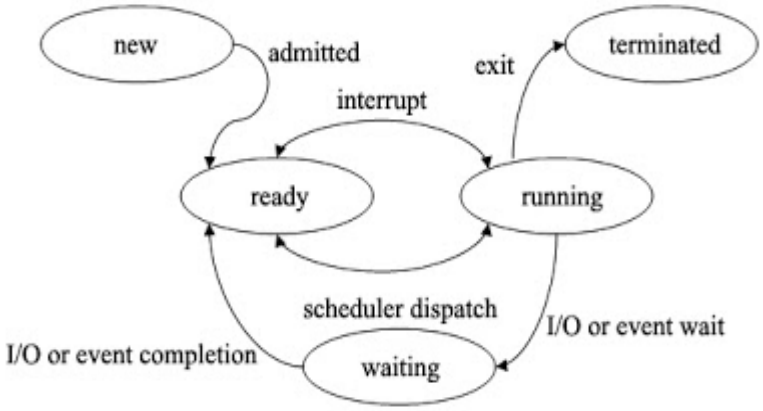
2.Ready:-The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after **Start** state or while running it by but interrupted by the scheduler to assign CPU to some other process.

3.**Running:**-Once the process has been assigned to a processor by the OS scheduler, the process state is

set to running and the processor executes its instructions

4.**Waiting:-**Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

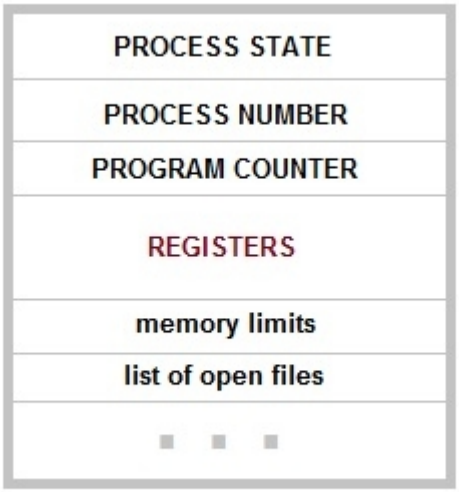
5.**Terminated or Exit:-**Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.



PCB(Process Control Block):-

There is a Process Control Block for each process, enclosing all the information about the process. It is a data structure, which contains the following :

- Process State - It can be running, waiting etc.
- Process ID and parent process ID.
- CPU registers and Program Counter. **Program Counter** holds the address of the next instruction to be executed for that process.
- CPU Scheduling information - Such as priority information and pointers to scheduling queues.
- Memory Management information - Eg. page tables or segment tables.
- Accounting information - user and kernel CPU time consumed, account numbers, limits, etc.
- I/O Status information - Devices allocated, open file tables, etc.



Fork() :

Fork is used to cerate a new process, which becomes the child process of the caller. It takes no argument and returns a process ID.

- If fork() returns a negative value ,the creation of a child process was unsucessful.
- 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.

Unix will make an exact copy of the parent's address space and give it to the child.

Execv() :

execve() executes the program pointed to by filename. Argv is an array of argument string passed to the new program.envp is an array of strings, conventially of the form key=value, which are passed as environment to the new program. Both argv and envp must be terminated by a null pointer.

Wait() System call :

The system call wait() is easy. This function block the calling process until one of its child processes exit or a signal received. For our purpose,we shall ignore signals.Wait() takes the address of an integer variable and return the processes ID of the completed processes. Some flag that indicate the completion status of the child processes are passed back with the integer pointer. One of the main purpose of wait() is to wait for completion of child processes.

The execution of wait() could have two possible situations:

1) If there are at least one child processes running when the call to wait() is made, the caller will be blocked until one of its child processes exit. At that moment,the caller resumes its execution.

2) If there is no child running when the call to wait() is made,then this wait() has no effect at all. That is,it is as if no wait() is there.

Zombie State :

A process that has complete execution but still has an entry in the process table.It is a process terminated state. This occurs for child processes, where the entry is still needed to allow the parent process to read its child's exit status : once the exit status is read via the wait system call,the

zombie's entry is removed from the process table and it is said to be “reaped”.

Orphan Process :

An orphan process is a computer whose parent process has finished or terminated,though it remains running itself. In a unix like operating system any orphaned process will be immediately adopted by the special init system process.

Header files :

1) #include<sys/wait.h> : It is used declarations for waiting.
2)#include<stdio.h> : This ANSI header file relates to "standard" input/output functions. Files,dvices and directories are referenced using pointers to objects of the type FILE . The header file contains declarations for these functions and macros, defines the FILE type, and contains various constants related to files.
3)#include<stdlib.h>:This ANSI header file contains declarations for many standard functions, excluding those declared in other header files discussed in this section.

4)#include<sys/types.h> :This POSIX header file contains declarations for the types used by system-level calls to obtain file status or time information.

Conclusion :

Thus we have implemented merge and quick sort using parent and child process.