OPERATING SYSTEMS:
ASSIGNMENT-03

2)a)Process:

A process is a program in execution. The execution takes place sequentially. This is the basic unit of work to be implemented in the system. When program is loaded into memory it becomes a process. It could be divided into 4 sections: stack, heap, text and data.

b)Process in memory:

All processes require their own private memory space to run. These are divided into 3 sections-text, data and stack.

c)passive versus active process:

When a program is passive it is a part of the process. Passive entity is sequence of instructions written to perform a specific task in the computer. They gets stored in secondary memory. A program is needed for the computer to run basically in its CPU.

Active process is the one which gets executed. It contains program code and its activity. Depending on OS,active process could be the multiple threads which gets executed concurrently.

c)process state:

A process could be in any of the following states:

New- state of being created

Ready-Process has all the resources needed to run,but CPU is not currently working on it.

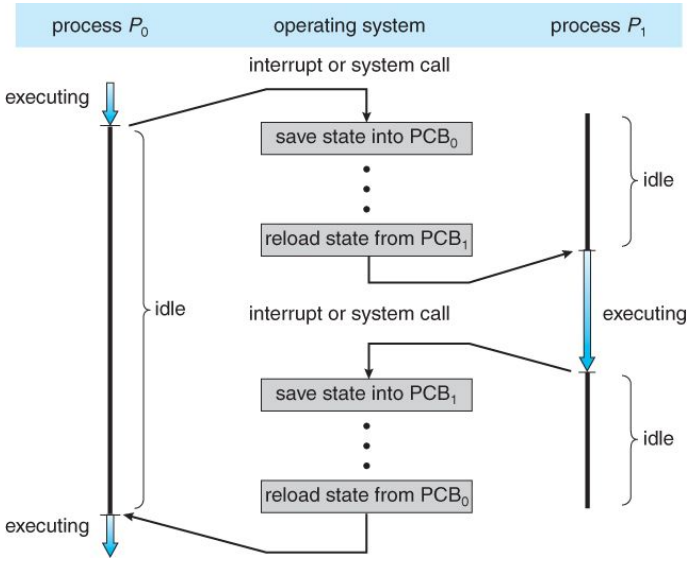
Running-CPU is working on process instructions

Waiting- Process could not run at the moment because it is waiting for some event to occur or some other process was running or waiting for some resources

Terminated-process has completed execution

d)CPU switch from one process to another process:

- When an interrupt occurs CPU saves the current state of the process and switch to kernel mode to handle the interrupt and then do state restore of the interrupted process.
- Similarly context switch occurs when time slice for one process has expired and other process is loaded from the Ready queue. This is instigated by the timer process which cause the current process state to be stored and restores the state of the new process.
- Saving and restoring involves saving and restoring all the program counters and control blocks and registers.
- Context switching can happen frequently but it is just the loss of CPU time and hardwares has special provisions for speeding this process up.



e)Process Control Blocks:

- For each process PCB stores the process-specific information listed above.
- Process state-state of the current process as running,terminated etc
- Process ID and parent process ID
- CPU registers and program counters -they need to be saved and restored when swapping process in and out of CPU
- CPU scheduling information-priority information and pointers to scheduling queues
- Memory management information
- Accounting information(such as user and kernel time consumed etc)
- I/O status information

f)Difference between Thread and Process

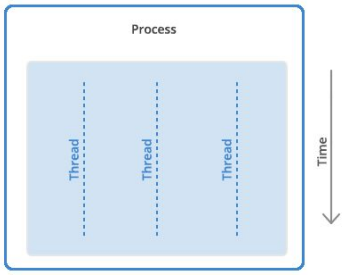
Process:

Process generally provides the resources necessary for the execution of the program. Process has virtual address space, executable code, open handles to system objects, security context,

process identifier, environment variables, priority class, minimum and maximum working set sizes and thread of execution. Primary threads starts each process and additional threads could be added to it.

Thread:

Thread lies within a process and can be scheduled for execution. All threads of a process share virtual address space and system resources. Each thread maintains an exception handler, scheduling priority, local storage, thread identifier and set of structures to save the thread context until it is scheduled for execution.



g)PCB in linux:

In Linux kernel, each process is represented by a task struct in a doubly linked list head of which is init_task(pid0 and not pid1). This is called as process table. In user mode process table is visible to normal users under /proc. This table stores the following data:

Process identification data

Process state data

Process control data

h)eight system calls

1.fork()

This system call is used to create new child process which runs concurrently with the parent process(called system call fork).after the new process is created both of the calls executes the instruction next to fork. Child process uses the same program counters and open files as the parent process. This returns negative,zero or positive values depending on whether the child process is created or no.

2.exit()

When this system call is called child leaves an exit status returned to the parent. When the child finishes it becomes a zombie. Exit first calls the functions registered by atexit(), in reverse order of registration. Exit function() flushes all the output streams closes all open streams and removes all files created by tempfile()

3.wait()

This blocks the calling process until one of its child process exits or signal is received. After the child process terminates, parent continues its execution after wait system instruction.

4.open()

Used to open the file for read/write operations
Example:

```
#include<sys/types.h>
#includ<sys/stat.h>
#include <fcntl.h>
int open (const char* Path, int flags [, int mode ]);
```

5.read()

This function reads cnt bytes of input into the memory area indicated by buff from the file indicated by file descriptor. This also updates access time for the file
Example:

```
size_t read (int fd, void* buf, size_t cnt);
```

6)write()

Writes cnt bytes from buf to the file or socket associated with file descriptor.cnt bytes should not be greater than Integer_max.
Example:

```
size_t write (int fd, void* buf, size_t cnt);
```

7)close()

This conveys that necessary operation is done and the file descriptor could close the file pointed by file descriptor.
Example:

```
#include <fcntl.h>
int close(int fd);
```

8)chmod()

This system call is used to change file mode bits
Example:

```
int chmod(const char *pathname, mode_t mode);
```

i)Process Scheduler

This component assigns the work to the resources for completing the work on the basis of scheduling algorithm. Types of process scheduler are as follows:
Long Term Scheduler: this selects processes from secondary memory and loads into ready queue in main memory for execution. This controls degree of multi programming. This processes generally takes a long time.
Short term scheduler: this selects process from ready queue and makes them execute instructions. This schedules the short term process.
Medium term scheduler: this is used to swap processes in memory. This occurs in time which is intermediate between short and long term schedulers.

3)

A 64-bit machine should be able to address up to 2^{64} addressable units. It ranges from 2^0 to 2^64 addresses.theoritically it should have 16.8million terabytes of memory.
6)
A process is in new state when it is freshly created and it is not running
When the created process gets admitted by shared resources and kernal it moves onto ready state when it has all the resources and files needed to run the program.
When the process is initiated the thread moves to the running state when the scheduler dispatches the run request after the previous process has completed its execution
When an interrupt occurs during the execution, the process gets stopped and it moves to the ready state.
But when the running process undergoes wait because of input/output or event wait, it goes to waiting state depending on what the input/output says
After wait is completed it moves to the ready state from where execution restores.
After the process is completed, it encounters exit signal and gets terminated.

7)Process Identifier:

pid t_pid;

Process Identifier is used to identify a process uniquely. This is generally assigned when the process gets initiated by the commands and this process identifier helps us to identify and kill all the zombie processes in the system. They could be seen from the process table.
State of the process:

long state

a process could be in any one of the following states:
New- The process is in the creation stage.
Ready - All the resources are ready to start the process but it is waiting for instruction from CPU
Running -CPU works on instructions in program.
Waiting - process could not run at the moment because of various reasons like waiting for resource or waiting for a process to get over. Once the wait is over execution resumes normally.
Terminated - The process has completed its execution.
scheduling information

unsigned int time_slice

-this kind of information includes priority information and pointers to scheduling queues. This scheduling information keeps track of states also. This raises the alarm via the scheduler once the state is over or need to be initialized.
this process's parent-

struct task_struct *parent

This is the process which could have some other processes running from it. This parent process shares its resources with all other child process. Child process have access to all the open files in parent process.
this process's children

struct list_head children

-This process got its way out from other parent processes and this child process could access all the files and resources from the parent process.
list of open files-

struct files_struct *files

certain files are needed when the process gets executed and when this files belongs to the parent process they could be accessed by the child process also. When the system boots it requires the files for the kernel to start its process.
address space of this process-

struct mm_struct *mm

All the process has its own private address space in the memory. When a process gets initialized it is allocated with the address space and they are unique for each and every process.