# Assignment No. - 1

1) Justify the following: Process can exercise crude control of their scheduling priority by using nice() system call function.

→ The kernel implements a fair-share scheduling algorithm that gives processes a share of CPU time based on priorities assigned to them, depending on the nature of the task. Higher-priority processes get scheduled more often and receive more CPU time, but a process can exercise crude control of it's scheduling by using the system call nice() as follows:

```
nice (value);
```

Process priority is a function of this nice value.

Process priority = recent CPU usage / a constant + base priority + nice value.

This algorithm gives user group A twice the slot for group B, three times that of C & four times that of D, where user processes are grouped by priority. This method isn't suitable for real-time processing, where the process can't afford to wait on mission-critical tasks; such a process gains instant CPU usage by making system calls and sending high-priority interrupt signals.

The fork system call in Unix creates a new process. The new process. The inherits various properties from its parent that is Environmental variables, File descriptors etc.

After a successful fork call, two copies of the original code will be running. In the original process the return value of fork will be running the process ID of the child. In the new child process the return value of fork will be 0.
When we type 'date' on the unix command line, the command line interpreter, i.e 'shell' forks so that momentally 2 shells are running, then the code in the child process is replaced by the code of the 'date' program by using one of the family of exec system calls.

2) Justify the following: 'Process 0 and process 1 exists through the lifetime of a system'

→ True.

The PID 0 is reserved for the swapper process, and 1 for the init process. The startup function for the kernel (also called the swapper or process 0) establishes memory management (paging tables & memory paging), detects the type of CPU and any additional functionality such as floating point capabilities, and then switches to non-architecture specific Linux kernel functionality via a call to start_kernel().
Init is the father of all processes. Its primary role is to create processes from a script stored in the file /etc/inittab. This file usually has entries which cause init to spawn gettys on each time line

that users can log in. It also controls autonomous processes required by any particular system. A run level is a software configuration of the system which allows only a selected group of processes to exist

3] Justify the following: At the kernel level, support for protected process is two fold.

→ True.

At the kernel level, support for protected processes is twofold: first the bulk of process creation occurs in kernel mode to avoid injection attacks. Second, protected processes have special bit set in their EPROCESS structure that modifies the behavior of security related routines in the process manager to deny certain access rights that would normally be generated to administrators

4] Justify the following: In linux the files are usually accessed via file names.

→ In linux files are usually accessed via filenames, they actually are not directly associated with such names.

Instead a file is referenced by an inode which is assigned a unique numerical value. is called inode numer or ino (This value)

5] Explain the behaviour of following C program.

```
main()
{
```

```
int status;
if(fork()==0)
execl("/bin/date", "date", 0);
wait(&status);
}
```

The fork system call in Unix creates a new process. The new process inherits various properties from its parent that is Enviromental variables, File descriptors, etc.

After a successful fork call, two copies of the original code will be running. In the original process that is parent, the return value of fork will be the process ID of the child. In the new child process the return value of fork will be 0.

When we type "date" on the unix command line, the command line interpreter i.e. "shell" forks so that momentarily 2 shells are running, then the code in the child process is replaced by the code of the "date" program by using one of the family of exec system calls.

6) Write a C program to open a file write append mode. Suppose the size of the file is n bytes. At the $(n+100)^{th}$ byte the same file, write the string "UNIX".

→
```
#include <stdio.h>
void main()
{
```

```
int fd;
char *buf = "UNIX";
fd = open("filefd.c", 'a');
if(fd==1)
    perror("error");
lseek(fd,0,SEEK_END);
lseek(fd,100,SEEK_CUR);
write(fd,buf,5);
close(fd);
}
```