it take the resource manager

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

The kernal implements a fair-share scheduling algorithm that gives process a share of CPU time based on priorities assigned to them, depending on the nature of the task. High priority process get scheduled more often and receive more cpu time but the process can exercise crude control of its scheduling by using the system call nice() as follow.

Process priority is the function of this nice value.

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

this algorithm gives algorithm group A twice the slot of group B, their times that of C and four time that of D, where user processes are group by priority.

Justify the following. process 0 and process 1 exisit through the lifetime of the system

True.
The PID 0 is reserved for the swapper process and 1 for the init process. the startup fun^n for the kernal establishes memory management, detects the type of CPU and any additional functionality such as floating point capabilities & then switches to non architecture specific Linux kernal functionality via call to start_kernal() init is the father of the process.

Justify the following. At the kernal level support for the protected process is two fold.

True.
At the kernal level support for protected Process is two fold first, the bulk of process creation occours in kernal mode to avoid injection attacks. second Protected process have special bit set in their EPROCESS structure that modifies the behaviour of security related routines in the process manager to deny certain access rights that would normally be granted to administrators.

Justify the following. In Linux the file is usually accessed via file names.

→ the actually are not directly associated with such names,
Instead, a file is referenced by an innode which is assigned a uniaue nummerical value. this value is called inode number or ino.

5. explain the behaviour of following program

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

the fork system call in ~~Linux~~ unix creates a new process. The New process 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 that is the 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"

2 shells are running.

9. write a C program to open a file in write append mode. the size of the file is n bytes. At the (n+100th) byte the same file write the string unix

→

```
#include<stdio.h>
 void main()
 {
  int fd;
 char *buf = "UNIX";
 fd = open("file.c", 'a');
 if (fd == -1)
 perror ("error");
 lseek (fd, 0, SEEK_END);
  lseek (fd, 100, SEEK_CUR);
  write (fd, buf, 5);
  close (fd);
 }
```