Device Drivers
Quiz 1

Sreepathy Jayanand
CED171038

Part - A

1. i. Floating point units, depending on the bottleneck stage delay usually takes a lot more time compared to integer arithmetic. A kernel is the fundamental program that maintains everything else and hence has to be extremely fast. So we should try to avoid floating point calculation.

ii. Since the kernel is usually that looks at exception flags and other trap bits set by registers after an operation, it cannot easily trap itself. So this has to be taken care by manually saving and restoring floating point registers.

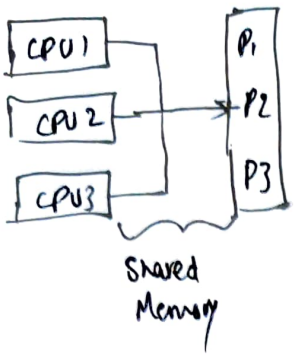
2. Depends.

It depends on the implementation of the kernel. Some support symmetrical multiprocessing, others don't.

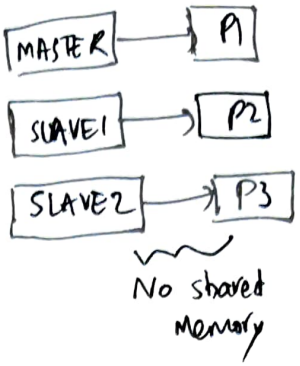
Linux, windows, mac (current versions) all support symmetrical multiprocessing, and hence without proper protection, kernel code executing simultaneously on two or more processors can con-currently access the same resource.

Meanwhile MS-DOS doesn't support symmetrical multiprocessing.

Symmetric Multiprocessing



Asymmetic Multiprocessing



3. i. Process control system calls

Eg: i. fork() : Creates a child process
ii. kill pid : Kills a process with id = $pid.

Process control system calls help to start, stop, pause etc. Helps to control processes.

ii. Information management system call

Eg: i. ls : List files and folders in $PWD.
ii. getpid() : Obtain the process id of a process.

These system calls help to give the user information regarding files, folders, their contents etc.

iii. Protection system call.

Eg: i. chown : Change ownership of a file/folder.
ii. chmod : Change file modes.

These system calls help to change permission like read, write, execute permissions of a user.

4. Processor affinity: Affinity basically means likeness. Processor affinity refers to how much a thread/process likes running on a processor. This enables the binding & unbinding of a thread/process to a range of CPUs so that the process will execute only on the designated CPUs rather than any CPU.

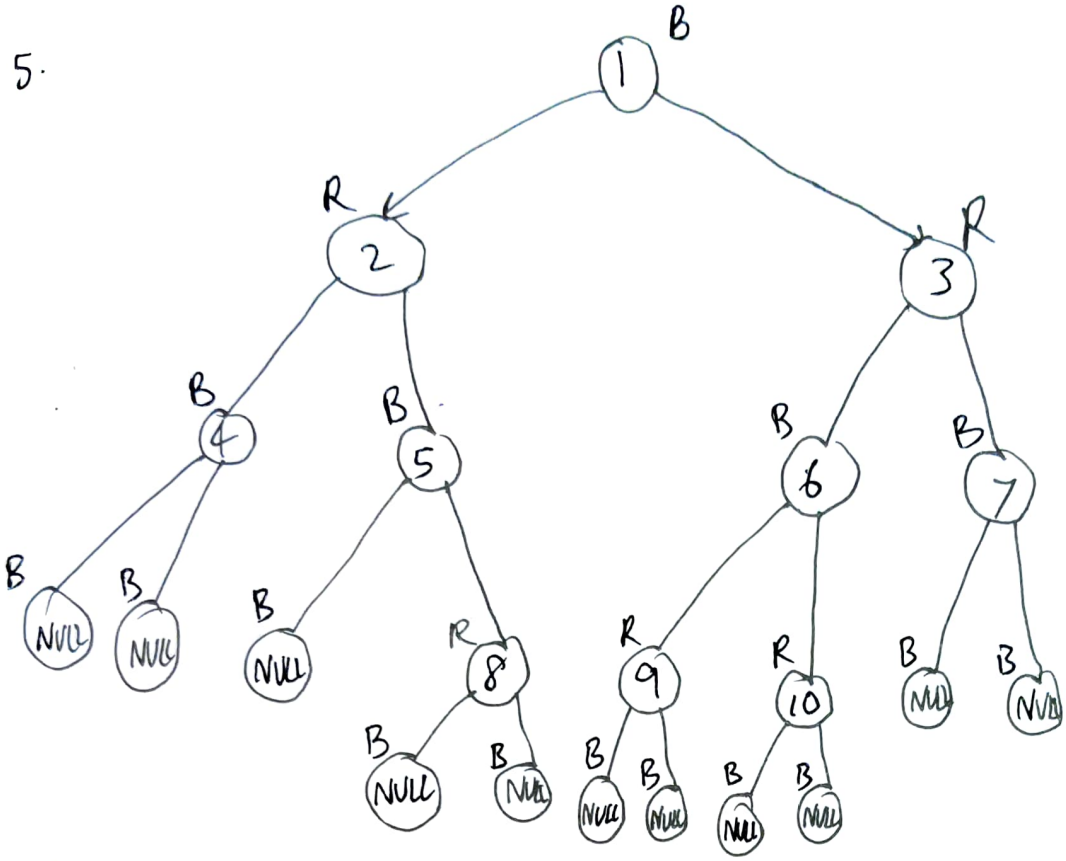
How it is controlled?

Linux scheduler enforces hard processor affinity. This means although it provides natural affinity by attempting to keep processes on the same processor, it also has the feature for a user to say a particular task must be run on a particular subset of CPUs no matter what.

This is stored as a bitmask (a sequence of 1's & 0's). 1 bit per possible processor. By default all bits are 1, which implies the program can run on any processor.

User can use sched_setaffinity() to change the bits to his/her liking.

5.



Properties:

i. All nodes are either red or black

As we can see from the drawing all nodes are labelled either red or black.

ii. Leaf nodes are black.

As we can see from the drawing all leaf (NULL) nodes are labelled back.

iii. Leaf nodes do not contain date.

As we can see from the drawing, all leaf nodes contain NULL as their entry and hence doesn't contain any data.

iv. All non-leaf nodes have 2 children.

Non leaf nodes are: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

We can see all of these have 2 children.

By.

v. If a node is red, both children are black.

Red nodes: 2, 3, 8, 9, 10

(2) has children 4 & 5, both are black.

(3) has children 6 & 7, both are black.

(8) has children NULL & NULL, both are black (leaves are always black)

(9) has children NULL & NULL, both are black

(10) has children NULL & NULL, both are black

vi. Path from a node to leaf contains same number of black nodes as the shortest path to any of its leaves.

(1): We can see that all paths to leafs from (1) has 3 black nodes. (Including (1) and leaf)

(2): We can see that all paths to leafs from (2) has 2 black nodes.

(3): We can see that all paths to leafs from (3) has 2 black nodes

(4): We can see that all paths to leafs from (4) has 2 black nodes.

Similarly (5) has 2

(6) has 2

(7) has 2

(8), (9) & (10) have 1 black node each from itself to any leaf.

And all leafs have 1 black node from itself to itself.

All conditions of red-black tree is satisfied and hence it is a red-black tree.

Part B

1. Explain & differentiate each of the following.

(1) wait()
(2) waitpid()
(3) wait3()
(4) wait4()

1) The wait() system calls suspends execution of the current process until one of its children terminates

Syntax: pid_t wait(int *status);

2) waitpid() system call suspends execution of current process until a child specified by pid argument has changed state. By default, waitpid() waits only for terminated children but its behaviour can be modified via options.

Syntax: pid_t waitpid(pid_t, pid, int *status, int options);

3) wait3(): This function delays its caller until a signal is received or one of its child process terminates or stops due to tracing. If any child has died and has not yet been reported return is immediate, returning process ID and status of one of its children.
If child process is dead, it is discarded.
If there are no children, -1 is returned.
If there are only running or stopped but reported children, the calling process is blocked.

4. wait4(): This is an extended interface with a pid argument of 0, it is equivalent to wait3(). If pid has non zero value, then wait4() returns status only for the process with pid, and not for any other child process.

Syntax: pid_t wait4(pid_t pid, int *status, int options, struct rusage * rusage);

2. Firstly the source code is written in C by including all the necessary headers/includes. Inside which we write one __init() & __exit() macros that help in identifying starting and ending statements which have to be executed.

Then we have to make object file Kbuild exist in the drivers/char/makefile so that whenever it compiles modules it goes into this location and obtain executables.

Now we use make command which helps maintain a group of programs together and specify the location also.

Now after becoming a root/admin we insert the module using insmod, which asks the kernel to load the module. If we want we can check the modules inserted using lsmod.

Whatever message printed using printk() will be put in /var/log/message. We can read that if necessary. And after we have finished our use of the module we can remove it using rmmod.