Sreepathy. Jayanand
CED17I038

Device Drivers Theory

Part - A

1. True/False. We can have same major number for 2 device

ANS: True. Major number defines the class of the device and minor number identifies which device within the major number.

So the pair <major number, minor number> is unique, but not just major number.

2. How to create a zombie?

~~ANS: Zombie process is a process that has been completed, entry it entry~~

ANS: A child that terminates, but has not been waited for becomes a zombie.

Code to create a zombie process

```
pid = fork ()
if (pid = = 0) {
  exit (0);
}
else {
  sleep( 50);
  ~~pid = wait(& status);~~
}
return 0;
```

3. What is the use of PID ?

ANS: PID → Process identifies is a number used by most operating systems, like Linux, Mac OS, Windows to uniquely identify an active process.

~~PID~~

4. Explain 4 pins of a USB port.

ANS:

Pin	Name	Color	Function
1	Vcc	Red	+5V supply voltage.
2.	D-	White	Data - signal line
3.	D+	Green	Data + signal line
4.	GND	Black	Ground

Pin "1" is used to give the 5V power.
Pin "4" is used to give 0V / ground.
Pins "2" & "3" are used for the differential data transmission.

5. The kernel has the ______zero.

ANS:

6. True / False. Device driver uses 2 different buffers, one for input and other for output.

ANS:

7. ~~What~~ What is "awakening of a call". Give example.

ANS. When ~~a~~ a driver ~~a~~ which is in a blocked state, waiting for some data, it is awakened when the data arrives. Usually the hardware issues an interrupt to signal such an event.

Example: The SCULL is a character device driver that acts on a memory area. When the ~~SC~~ SCULL is in a blocked state and has requested for data, when the data ~~arrives~~ arrives wake_up_interruptibe(...) is called to awaken any writers for the given

8. Explain alloc_region()

~~ANS~~: ANS:

9: What is a file descriptor?

ANS. A file descriptor is a unique non-negative number that uniquely identifies an open file in a computer's operating system. It describes a data resource, and how that resource may be accessed.

When a file descriptor is created the kernel creates an entry of that in the global file table.

10. What is a URB?

ANS: An ~~URB~~ URB consist of all relevant information to execute any USB transaction and deliver the data & status back.

It is an asynchronous operation & usb_submit_urb() after it has queued the request, and can be cancelled using usb_unlink_urb()

11. True/False. PID gets changed when a process state is changed from "sleep" to "running".

ANS. False. PID is process identifier an is unique to that process. So unless the process dies and is restarted its pid will be the same.

12. Explain the difference between ABIs & APIs

ANS. An API or application programming interface defines the interface by which one piece of software communicates with another at the source code level. There will be direct function calls which take of the communication.

An ABI defines low-level binary interface between 2 or more softwares on a particular architecture. It defines how an application interacts, It is more detailed & ~~we have to~~ takes care of lower higher details.

13. Explain "hard link" with example.

ANS. A hard link is a directory entry that associates a name with a file on a file system. All directory based file systems must have at least one hard link giving the original name for each file. The term hard link is usually used in systems that allow more than one hard link.

Example: If we hard link a .c source code and then delete the file, we can still access using hard link, ~~but~~ but not with soft link.

14. True /False. A USB device can never start sending data without first asked by a host computer

ANS. True. Topologically a USB sub-system is not laid out as a bus, but like a tree built out of several point-to-point links. There are 4 wire cables. The USB host controller is in charge of asking every USB device if it has any data to send. Hence without the host controller giving permission it cannot send.

15. Explain log level with respect to printk.

ANS. A log level is a piece of information telling how important a given log message is. It is a powerful way of distinguishing log messages.

Different log levels that can be specified in printk() are KERN_INFO, KERN_ALERT, etc.

16. Every peripheral ____ is controlled by ____ and its ____

ANS. Every peripheral device is controlled by writing and reading its registers.

17. True /False. ~~Does~~ kmalloc() ~~clea~~ doesn't clear the memory it obtains, The allocated region holds its previous value.

ANS. True. kmalloc() is a fast allocating function and doesn't clear the memory it obtains and holds the previous content.

18. Explain how different is the action between writing operation on I/O registers is from writing to RAM.

ANS:. The main difference between writing to I/O registers & RAM is that I/O operations have side effects, while memory operations have none. The only effect of a memory write is storing a value to a location, and a memory read returns the last value written.

19. Data types used by kernel are ———, ——— & ———

ANS. 1. Standard C types (like int, char)
2. Explicit sized types (~~u32~~ like u32)
3. Kernel object types (like pid_t)

20. Explain and compare the following 2 commands.

1. dd bs=1 count=2097152 if=/dev/zero of=pirate
2. dd bs=1024 count=2048 if=/dev/zero of=pirate.

ANS. In case of 1, ~~Because~~ bs=1, this command will copy 2 MB from the device zero to file pirate in 2097152 Byte chunks.

In case of 2, this command copies ~~1024~~ with block size of 1024 Bytes and writes only ~~1024~~ 2048 times.

So 2 is 1024 times faster than 1.

Part - B

3. Explain Kernel synchronisation,

ANS. The two main aspects of synchronization are
i. Provide synchronization constructs for the user space processes & ~~[illegible]~~ treads
ii. Meet synchronization requirements of the operating system itself.

If a processor is a uni-processor and if its non-preemptive there is no need ~~for explicit~~ to take care of kernel synchronisation as there will never be a chance of racing.

In case it is a uni-processor setup but a pre-emptive kernel there is a possibility of racing occuring.

Example : int a ;
a ++ ;

Here a++ is not an atomic operation. When it is converted to machine code it becomes

load a ;
increment a ;
write a ;.

Suppose 2 programs are performing increment on the same "a".

```
[ Program 1
     load a      (a = 4) ]
```

↳ Preemption to 2

```
[ Program 2
   load a      (a = 4)
   increment a (a = 5)
   write a     (a = 5) ]
```

↳ Preemption to 1

```
[ Program 1
   increment a  (a = 5)
   write a      (a = 5) ]
```

Here after program 1 & 2 finish executing, actual value of a should be 6 but we only get 5.

Hence a synchronisation technique should be used, even in uni-processor setup.

The different causes of concurrency that can occur are

1. Interrupts
2. Soft irq, & tasklets
3. Kernel pre-emption
4. Sleeping & synchronization with user space.
5. Symmetrical Multiprocessing.

Any interrupt induces unexpected execution of new process. This new process may get into RACE with already running process.

Similarly the kernel preempts, the process may leave a lot of resources unaccounted which may be accessed by new-processes.

Also, too much delay and sleeping in beginning induces race.

Also, high priority processes running in different CPUs can cause racing to occur.

Also, softirqs or high priority routines of kernel can also cause racing to occur. Eg: Timer interrupts.

The kernel has to not just ensure racing doesn't happen by locking resources at the right times. It also has to ensure that no deadlock is occuring while it is locking the resources.

8. Explain (a) MCA bus (b) EISA bus (c) VLB bus (d) SBUS (e) NUBUS

ANS. (a) MCA bus: Micro channel architecture. It is an expansion bus created by IBM that was used in the company's desktop computers. An expansion bus allows additional cards to be connected to the computer's motherboard, expanding the number of I/O ports. The MCA bus architecture was an improvement in both size and speed over AT and ISA.

(b) EISA bus: Extended Industry standard architecture is a bus standard for IBM PC compatible computers. It is designed as an alternative to MCA bus. These buses can be used for bandwidth intensive tasks such as disk access and marketing.

(c) VLB bus: It stands for VESA local bus. The VLB is a hardware interface on the computer's motherboard that is attached to an expansion slot. By connecting a video expansion card to VLB, you can add extra graphic capabilities to your computer,

(d) SBUS: SBUS is a computer bus system that was used in most SPARC-based computers. It was developed as a high speed bus counterpart to their high speed SPARC processor. In the beginning SBUS was used as both a system bus and a peripheral interconnect that allowed input and output devices relatively low latency access for memory.

(e) NuBUS: NuBUS is a 32 bit parallel computer bus. It is no longer used and is replaced by the peripheral component interconnect (PCI) and other parallel buses.

7. Explain loop back interface and SNULL

ANS. The loop back interface is a special virtual network interface that your computer uses to communicate with itself. It is used mainly for diagnostics and trouble shooting and to connect to servers running on the local machine.

When a network interface is disconnected - for example an ethernet is unplugged or wifi is not working, no communication via that interface is possible. Here the loop back interface becomes important. Existing applications can always connect to servers in the same machine and hence we can use this feature to our benefit.

For IPv4, the loopback interface is assigned to all IPs in the 127.0.0.0/8 address block, i.e 127.0.0.1 [illegible] to 127.255.255.254. This IP has the hostname of local host mapped to it.

SNULL or simple network utility for loading localities works by creating 2 interfaces. The message transmitted in one interface comes back via a different interface. It simulates the conversation with real remote hosts inorder to better demonstrate the task of writing a network driver.

~~To~~ SNULL modifies the packet while it leaves the ~~[illegible]~~ computer in such a way that it changes the network id. It then sends it and creates another ~~an~~ interface to accept that packet.

This is done by changing / toggling last bit of the 3rd octet of the ip.

Example : 192.168.5.5 becomes 192.168.4.5 as it leaves the computer.

This makes the user to experience a packet delivery from another network.

Q. Explain interrupt handlers. How h/w interrupt are handled compared to software interrupts.

ANS: Interrupt handler or interrupt service routine is a special block of code associated with a specific interrupt condition. These handlers are initiated by hardware interrupts, software interrupts, exceptions etc.

Hardware interrups arise from electrical conditions implemented using digital logic and are asynchronous in nature. These are then converted software level interrupts for the operating system to handle.
Software interrupt are implemented at the operating system level as a form of a callback function.

Hardware interrupts occurs all the time. A mouse click, keyboard click etc.

Often, Software interrupts is used to perform an input/output request, or communicating with disk controller for reading & writing data.
Software interrupts are called in the user space; it is called in response to the invocation of a system call. Software interrupt numbers are defined by the Operating system. ~~The current~~ When ~~idea~~ a software interrupt is called the current state ~~is~~ of the program is saved so that after the handler finishes executing it can resume.

6. How VFS works?

ANS: Virtual file system is a program that forms an interface between an operating system's kernel and a ~~[illegible]~~ concrete file system. The VFS serves as an abstraction layer that gives applications access to different types of file system, and local and network storage devices. It also manages the data storage and retrieval between OS and the storage subsystem.
The VFS maintains a cache or ~~[illegible]~~ directory lookups to enable easy location of frequently accessed directories.

The VFS describes the system's files in terms of super blocks and inodes, in case of unix. The inodes describe the files and directories within the system, the contents and also the topology.

As a file system is initialized, it registers itself with VFS.
File systems are loaded as and when the system needs them.

Example: The superblock representing a mounted EXT2 ~~for~~ file system contains a pointer to the EXT2 specific inode reading routine.
This EXT2 inode routine, like all of the file system specific inode reads routine, fills out the fields in a VFS inode.
Each VFS superblock contains a pointer to the first first VFS inode on the file system. For the root file system, this inode '/' represents it.