El 49398. Megha Lohar

Sunbeam
Institute of Information Technology

DESD Embedded Linux Device Driver

Question 1

1) Explain types of devices.

There are 3 types of Devices
1) Character Device
2) Block Device
3) Network Device

1) Character Device :-
Often abbreviated cdevs, character devices are generally not addressable, providing access to data only as a stream, generally of character (bytes). These devices are presented as a special files in a /dev directory & support direct reading & writing of any data, byte, like a stream. Character device are accessed via a special file called a character device node. They can also provide additional interface not present in block device node, such as I/O control (IOCTL) commands, memory mapping & device polling.
Ex: character device include keyboard, mice, printers & pseudo-device

2) Block Device :
Often abbreviated blkdevs, block devices are addressable in device-specified chunks called blocks & generally support seeking, the random access of data. Block

E-1_49398_Megha Lohar

devices are characterized by random access to data organized in fixed-size block In most Unix system, a block device can only handle I/O operations that transfe one or more whole blocks, which are usually 512 bytes (or a larger power of two) bytes in length Linux, instead allows the application to read & write a block device like a char device - it permits the transfer of any number of bytes at a time. As a result, block & char devices differ only in the way data is managed internally by the kernel, & thus in the kernel / driver software interface

3) Network Interface Device:

Network device drivers receive & transfer data packets on hardware interface that connect to extern system, & provide a uniform interface that network protocols can access. Breaking Unix's "everything is a file" design principle, network devices are not accessed via device node but with a special interface called the socket API. Network devices provide access to a network (such as the internet) via a physical adapter (such as your laptop's 802.11 card) & a specific protocols (such as IP).

Ex: Ethernet devices are most common type of network devices

Question 2:

Explain diffⁿ betⁿ application programming & module programming

E1-49398 - Megha Lohare

Module programming	Application Programming
1) A module runs in kernel space	1) Application runs in user space
2) kernel modules have a higher execution privilage.	2) Application has lower execution privilage than that of Modules.
3) A kernel module does not execute sequentially. A kernel module registers itself in order to serve future request.	3) An application program typically executes sequentially & perform a single task from begining to end.
4) Kernel modules donot define a main () program.	4) Application program define main () program.

Question 3

Write a short not on debugging techniques.

Kernel code cannot be easily executed under a debugger, nor can it be easily traced, because it is a set of functionalities not related to a speific process kernel code errors can be also be exceedingly hard to reproduce & can be bring down the entire system with them, thus destroying much of the evidence that could be used to track them down.

Following are some of the debugging techniques:

Debbing by Printing :

The most common debbuing technique is

EI-39398- Megha Lahar

monitoring. Which in application programming is done by calling printf at suitable points. When you are debbuging kernel code, you can accomplish some goal with printk

Debbuing by Querying:

The printk method is always not good. Because the log of prink is too big + too frequent then it may block the terminal + even can reduced the system performance. So, in some situation it's better to get kernel log "time to time" always than printk

We can do this day 2 ways:

1) By creating kernel log file manually inside /proc though by using ioct()

This can be done by 3 ways:

a) Creating small log file.

b) Creating large log file "seq-file" connect with "seq operation" structure.

c) Same like b but this time using "file-operation" structure.

2) Using ioctl()

Dubbing by Watching:

Sometimes minor problems can be tracked down by watching the behaviour of an application in user space. Watching programs can also help in building confidence that a driver is working correctly. There are various ways to watch a user-space program working. You can also debbuggen on it to step through its function + print statements, or run the program under strace. The strace command is a powerful tool that shows all the system calls issued by a user-space program. By default

81-39598-Megha Lohar

strace prints tracing information on stderr.

① Debugging system faults & system Hangs.

⇒ Debbuing system faults: Even if you've used all the monitoring & debbuing techniques, sometimes bugs remain in the driver, & the system faults when the driver is executed. Kernel oops are not kernel panics. Instead, it gives us a dump info of CPU register & their values with the location of fault. These faults are also called as CPU traps & all are mentioned in linux source code at arc/<desired architecture>/kernel/traps.c understanding this file understand all kernel oops or kernel dump.

⇒ Dubbing System hangs: Many times we become helpless when our system hangs but linux with customi kernel with debugging allow us to come out us smoothly without harming the hard disk & data corruption. That's why this function called magic sysReq where sysReq for system request. To enable this feature, we need to echo 1 on /proc/sys/kernel/sysreq now if sys hangs

Question 4:

Explain I/O architecture. Also explain three major hardware components of I/O architecture

The computer system I/O architecture is its interface to the outside world. This architecutre is designed to provide a systematic means of controllin interaction with the outside world & to provide the

EI-39398 Megha lahar

operating system with the information it needs to manage I/O activity effectively. Data transfer betⁿ central unit & device can be handled in generally three types of mo

1) Programmed I/O :

Programmed I/O instructions are the result of I/O instruction written in computer program. Each data item transfer is initiated by the instruction in the program. Usually, the program controls data transfer to & from CPU & peripheral. Transfering data under programmed I/O require constant of the peripherals by the CPU.

2) Interrupt intiated I/O :

In the programmed I/O method the CPU stays in the program loop untill the I/O unit indicates that it is ready for data transfer. This is time consuming process because it keeps the processor busy endlessly.

3) Direct Memory Access :

Removing the CPU from the path & letting the peripheral device manage the memory buses directly would improve the speed of transfer.

Question 5.

Which low level functions are used to access the devic

The low level I/O system in C provides functions that can be used to access files & devices.

• Open() : The open funⁿ can be used to open & existing

B1 39398 Megha Lohar

file or to create a new file.

    int open (char * filename, int flags, int perms);

2) Close(): The close() function closes the file that was opend using the open function. It takes the file descriptor as a parameter to close the file.

    int close (int filedesc);

3) Read(): The low level I/O system defines the read() function for reading data from a file

    int read (int filedes, char *buffer, int size);

4) Write(): The write() function enables you to write contents to a file.

    int write (int fileds, char *buffer, int size);

Question 6

Explain linux Device model (LDM/LDDM) in brief

Linux Device model is the fundamental part of linux device system although it is complexi it's knowledge is must for every writer or linux kernel programmer LDM is also called as uniform device model or UDev, the word "uniform" is important over the here because every linux system is up, supported uniformaly by the model

Question 7

Explain arguments & return value of read() & write system calls.

E1-39398 Megha Lohar

The read () function syntax:

Int read (int filedes, char * buffer, int size)

1) The first arguments is a file descriptor, that is obtained by opening the file
2) The second argument is a coracter array, where the data will be stored during the read operation
3) The third argument is the number of bytes to be Read

The write () function syntax:

Int write (int fileds, char *buffer, int size);

1) The first argument is a file descriptor, that is obtained by opening the file.
2) The second argument is a charater array, where tha data will be stored during the write operation
3) The third argument is the number of bytes to be Writter

Question 8

What does struct cdev c-dev represents in kernel space

struct cdev is one of the element of the inode structure. As you probuby may know already, an inode structure is used by the kernel inteanally to represents file The struct cdev is the kernel's internal structure that represents char devices

Question 9.

How does struct cdev c-dev

B1 39398 Megha lohar

How will you load & unload the module in kernel?

For loading a Module in kernel insmod is used & for Unloading a Module rmmod is used load Module syntax: sudo insmod module.ko
Unload Module syndax: sudo rmmod module.ko

Question 10:

What is IOCTL, explain in detail?

IOCTL is refered to as a input & output contool, which is used to talking to device drivers. This system call, available in most driver categories. The major use o this is in case of handling some specific operations of a device for which the kernel does not have a system call by default.

```
int ioctl (int fd, unsigned long cmd.....);
```

The ioctl driver method has a prototype that differs somewhat from the user-space version.

```
int (*ioct) (strut inode *inode, Strucl file, *filp unsigned i
    cmd, unsigned long arg);
```

Question 11

What is recommended in device driver coating?

* User programmes are self-executable & independent process. Is created for each program at runtime.

* Standard linker is not used to link kernel moduels.

E1-59398 Megha Lahar

3) Typical user programs have single entry point i.e. main ().

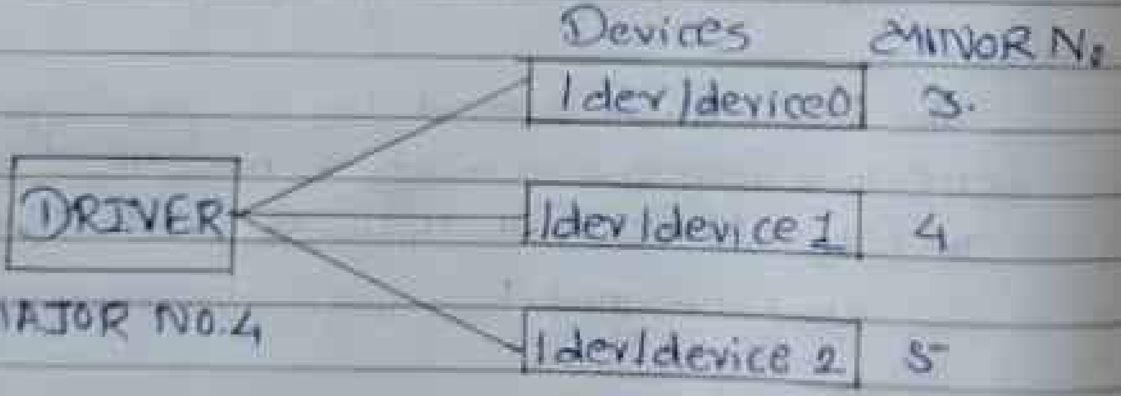
4) User space applications are not multi-threaded & hence rarely concurrency aware.

5) If resources like memory, file or network connection are not released by user space application, they are automatically relased when process terminates

6) User space application may use FPU heavily Resetting FPU for each operation dosen't hamper whole system performance

Question 12.

Explain features of Major / Minor number?



The major no. is to identify the corresponding driver. Many devices may use the same major number.
In other words, The device driver uses the minor number <minor> to distinguish individual physical or logical devices.

EI - 39398, Megha lohar

Question 13

Which functions / API is used to copy from kernel space & vice-a-versa?

Copy to user : Copies a block of data from the kernel to user space

Copy from user : Copies a block data from user space to the kernel.

Question 14:

Explain Interrupt handling in linux. Which are different types of buttom halves & where they are used?

Interrupt handling depends on the type of interrupt. For our purpose, we'll distiguish three main classes of interrupts

1 I/O interrupts : An I/O device require attention; the corresponding interrupt handler must query the device to deetermine the proper cource of action.

2) Timer interrupts : Some timer, either a local APIC timer or an external timer, has issued an interrupt;

3) Interprocessor interrupt : A CPU issued as interrupt to another CPU of a multiprocessor system

a) Tasklets :

b) Workquess

c) SoftIrqs.

El-59598 Megha lohar

Question 15

What is significance of /proc?

Proc file system (Procfs) is virtual file system created on fly system boots & is dissolved at time of system shut down.

It contains the useful information about the processes that are currently running, it is regarded as control & information center for kernel

ls -l /proc | grep "^d"

Question 16

How time is managed in kernel space?

B1-39398 Megha lohar

Question 17.

What is diffⁿ betⁿ get_free_pages (), kmalloc () & vmalloc () ? Explain mechanism behind them

1) kmalloc () : kmalloc is the normal method of allocating for objects smaller than page size in the kernel.

2) get_free_pages : To allocate (& free) entire pages (or multiple pages) at once, one can use

3) vmalloc () : vmalloc () allocates a contiguous memory region in the virtual address space.

Question 18.

What is spinlock? How it differs from semaphore & mutex?

SPINLOCK	SEMAPHORE
1) Spinlock can be used only mutual exclcusion	1) Semaphores can be used either for mutual exculusion or as a conunting semaphore
2) Spinlock can be wastefull if they are hold for a long time duration	2) Semaphore can be allow more than process at any given time to casses the critic section.
3) It is busy wait process	3) It is sleep wait process

El-39398- Megha Lohar

4) In spinlock it is recommeded to disable the interrupts while holding a spinlock	4) Semaphore can be locked with interrupt enabled
5) Spinlocks are valid for only one process	5) Semaphores can be used to synchonize betn diffn process

Question 19 :

What is URB? Explain kernel functions for data transfer using without URBs

Created by USB device driver :

- Assigned to a specific endpoint of a specific USB device
- Submitted to the USB core, by the USB device driver
- Submitted to the specific USB host controlled driver that makes a USB transfer to the device
- When the urb is completed, the USB controlled device notifies the USB device driver

Functions Using URB:
URBs are allocated by calling : usb-alloc-urb();
To free an URB : usb-free-urb();

Functions without Using URB;
USB for Registering : usb-register();
USB Diregistering : usb-deregister();