To study UNIX Device Driver installation steps.

Theory:

Installing a device driver in the UNIX system involves the following steps

- Compiling the device driver
- Modifying the kernel configuration tables and files
- Linking the device driver with the kernel object files to produce a new kernel
- Creating the necessary entries in the / dev directory
- Rebooting the system with the new kernel

Compiling the driver:

It is similar to compiling a C program. It is compiled to produce an object rather than a linked executable file

```
cc –O –c lp.c
mv lp.o Driver.o
```

This invokes the C compiler to generate 'optimized' code(-O) and an object file(-c) only. *mv* command rename the object file Driver.o so that the idinstall command will be able to locate the file.

Configuring the Kernel:

The next step is modifying the kernel configuration tables to reflect the addition of new driver.

The two configuration files used by the system to generate a new kernel are mdevice and sdevice files in the /etc/conf/cf.d directory (these are placed in files called Master and System, respectively). The mdevice file contains an entry for each device driver that exits on the system and sdevice file contains information for each driver that is to be incorporated into the kernel.

<u>The mdevice file:</u>

mdevice file consists of nine fields. For example(for line printer)

```
lp      Iocrw  Hcio          lp      0      0      1      1      -1
```

1. Device name: This is the internal name of the device driver
2. Function list: This specifies the entry points that are declared within the device driver. Each entry point is represented by a single letter.
3. Driver characteristics: This specifies the characteristics of the driver. The field consists of a list of letters with each letter representing characteristics.
4. Handler prefix: This is the prefix used for each of the driver's entry points. This string may be up to four characters long and must be unique among all drivers installed on the system.
5. Block major number: This is the major device number used to reference the driver. This value is assigned automatically during the installation process and ought to be left as 0 by the driver writer.
6. Character major number: This is the major device number used to reference the driver. This value is assigned automatically during the installation process and ought to be left as 0 by the driver writer.
7. Minimum units: This specifies the minimum number of these devices that can be specified in the sdevice file.
8. Maximum units: This specifies the maximum number of these devices that can be specified in the sdevice file.
9. DMA Channel: This specifies the DMA channel that the device will use. If DMA is not used the field is set to – 1. On some systems different devices may not share the same DMA channel.

<u>The sdevice file</u>

The sdevice file consists of ten fields. For example

```
lp    Y    1    3    1    7    378   37f   0    0
```

1. Device name: This is the internal name of the device driver.
2. Configure: This indicates whether or not this driver ought to be incorporated in the kernel.
3. Unit: This specifies the number of units installed.
4. lpl: This specifies the cpu priority level at which interrupts will be processed. If the driver has no interrupt handler this field is set to 0.
5. Interrupt Type: This specifies the interrupt architecture used by this driver.
6. Vector: This specifies the interrupt vector used by this device.
7. Start I/O Address (SIOA): This specifies the starting address on the I/O bus through which the device communicates.
8. End I/O Address (EIOA): This specifies the ending address on the I/O bus through which the device communicates.
9. Start Controller Memory Address(SCMA): This specifies the starting address of the device's dual ported memory.
10. End Controller Memory Address(ECMA): This specifies the ending address of the device's dual ported memory.

Installing the new driver:

On some UNIX systems utilities are provided to make the process of installing drivers easier. These tools will update the mdevice and sdevice files and perform many other tasks needed.

Building a new kernel:

Once the kernel configuration tables are configured we need to generate the new kernel. The command to build the kernel varies from system to system. On SVR3.2 system utility to run is ***idbuild.***

Creating Entries in /dev:

In order to access driver we need to create the special files in /dev directory. On some systems it must be done manually using ***mknod.*** When we install our driver we may optionally provide a NODE file that defines the entries in the node file.

For example:

```
lp    lp    c    0
```

1. Device Name: Internal name of the device driver.
2. Node Name: Name of the special file to be created in the /dev directory.
3. Node Type: Specifies whether the special file is character or block.
4. Minor Device No: Specifies the Minor device no for this Special file.

Above entry will create a special file with the name /dev/lp

Rebooting the new kernel:

We create a scratch directory (say /tmp/driver) into which we copy the Driver.o, Master, System and Node files. We then move into that directory and run the driver installation program. We build a new kernel by running the idbuild utility:

```
/ etc / conf / bin / idinstall    -a    lp
```