ಕರ್ನಾಟಕ ರಾಜ್ಯ ಮುಕ್ತ ವಿಶ್ವವಿದ್ಯಾನಿಲಯ

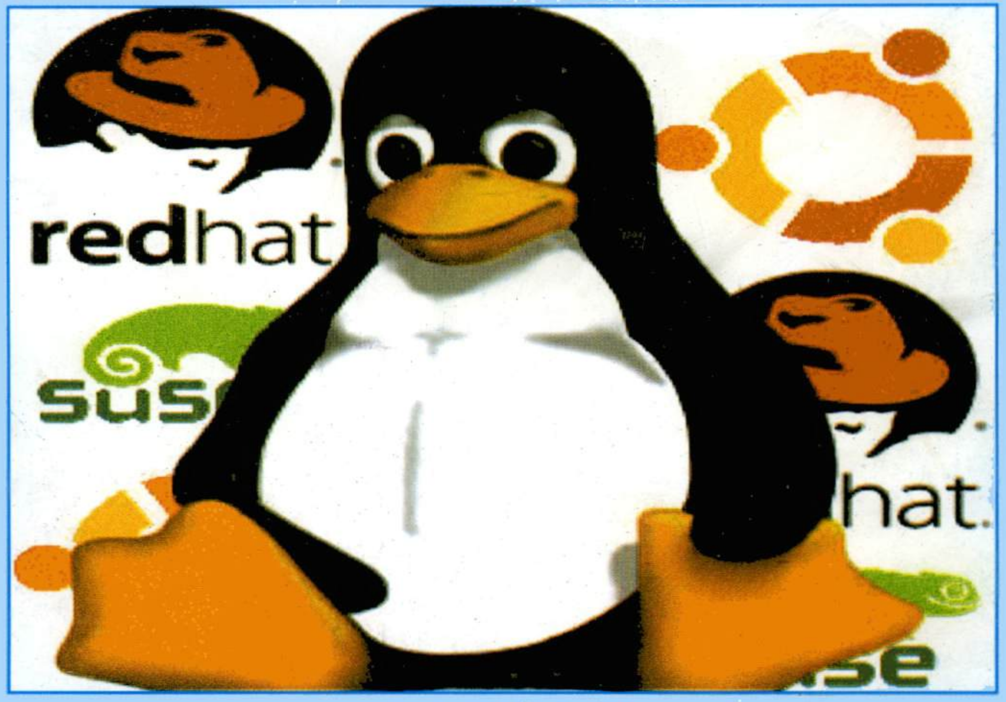
ಮಾನಸಗಂಗೋತ್ರಿ, ಮೈಸೂರು – 570 006

KARNATAKA STATE OPEN UNIVERSITY

Manasagangotri Mysore - 570 006

M.Sc. Computer Science

Second Semester



Course : 12

Practical-4

MSCS-512 LINUX INTERNALS & NETWORK PROGRAMMING

ಉನ್ನತ ಶಿಕ್ಷಣಕ್ಕಾಗಿ ಇರುವ ಅವಕಾಶಗಳನ್ನು ಹೆಚ್ಚಿಸುವುದಕ್ಕೆ ಮತ್ತು ಶಿಕ್ಷಣವನ್ನು ಪ್ರಜಾತಂತ್ರೀಕರಿಸುವುದಕ್ಕೆ ಮುಕ್ತ ವಿಶ್ವವಿದ್ಯಾನಿಲಯ ವ್ಯವಸ್ಥೆಯನ್ನು ಆರಂಭಿಸಲಾಗಿದೆ.

ರಾಷ್ಟ್ರೀಯ ಶಿಕ್ಷಣ ನೀತಿ 1986

The Open University System has been initiated in order to augment opportunities for higher education and as instrument of democrating education.

National Educational Policy 1986

ವಿಶ್ವ ಮಾನವ ಸಂದೇಶ

ಪ್ರತಿಯೊಂದು ಮಗುವು ಹುಟ್ಟುತ್ತಲೇ – ವಿಶ್ವಮಾನವ. ಬೆಳೆಯುತ್ತಾ ನಾವು ಅದನ್ನು 'ಅಲ್ಪ ಮಾನವ'ನನ್ನಾಗಿ ಮಾಡುತ್ತೇವೆ. ಮತ್ತೆ ಅದನ್ನು 'ವಿಶ್ವಮಾನವ'ನನ್ನಾಗಿ ಮಾಡುವುದೇ ವಿದ್ಯೆಯ ಕರ್ತವ್ಯವಾಗಬೇಕು.

ಮನುಜ ಮತ, ವಿಶ್ವ ಪಥ, ಸರ್ವೋದಯ, ಸಮನ್ವಯ, ಪೂರ್ಣದೃಷ್ಟಿ ಈ ಪಂಚಮಂತ್ರ ಇನ್ನು ಮುಂದಿನ ದೃಷ್ಟಿಯಾಗಬೇಕಾಗಿದೆ. ಅಂದರೆ, ನಮಗೆ ಇನ್ನು ಬೇಕಾದುದು ಆ ಮತ ಈ ಮತ ಅಲ್ಲ; ಮನುಜ ಮತ. ಆ ಪಥ ಈ ಪಥ ಅಲ್ಲ ; ವಿಶ್ವ ಪಥ. ಆ ಒಬ್ಬರ ಉದಯ ಮಾತ್ರವಲ್ಲ; ಸರ್ವರ ಸರ್ವಸ್ತರದ ಉದಯ. ಪರಸ್ಪರ ವಿಮುಖವಾಗಿ ಸಿಡಿದು ಹೋಗುವುದಲ್ಲ; ಸಮನ್ವಯಗೊಳ್ಳುವುದು. ಸಂಕುಚಿತ ಮತದ ಆಂಶಿಕ ದೃಷ್ಟಿ ಅಲ್ಲ; ಭೌತಿಕ ಪಾರಮಾರ್ಥಿಕ ಎಂಬ ಭಿನ್ನದೃಷ್ಟಿ ಅಲ್ಲ; ಎಲ್ಲವನ್ನು ಭಗವದ್ ದೃಷ್ಟಿಯಿಂದ ಕಾಣುವ ಪೂರ್ಣ ದೃಷ್ಟಿ.

ಕುವೆಂಪು

Gospel of Universal Man

Every Child, at birth, is the universal man. But, as it grows, we trun it into "a petty man". It should be the function of education to turn it again into the enlightened "universal man".

The Religion of Humanity, the Universal Path, the Welfare of All, Reconciliation, the Integral Vision - these ***five mantras*** should become view of the Future. In other words, what we want henceforth is not this religion or that religion, but the Religion of Humanity; not this path or that path, but the Universal Path; not the well-being of this individual or that individual, but the Welfare of All; not turning away and breaking off from one another, but reconciling and uniting in concord and harmony; and above all, not the partial view of a narrow creed, not the dual outlook of the material and the spiritual, but the Integral Vision of seeing all things with the eye of the Divine.

Kuvempu

Karnataka State Open University

Manasagangothri, Mysore – 570 006

M.Sc in Computer Science

Second Semester

LABORATORY EXERCISES

MSCS-512: Practical-4

Linux Internals and Network Programming

Developed by Academic Section, KSOU, Mysore
Karnataka State Open University, 2010

LABORARTORY MANUAL FOR LINUX INTERNALS

Structure

Objectives

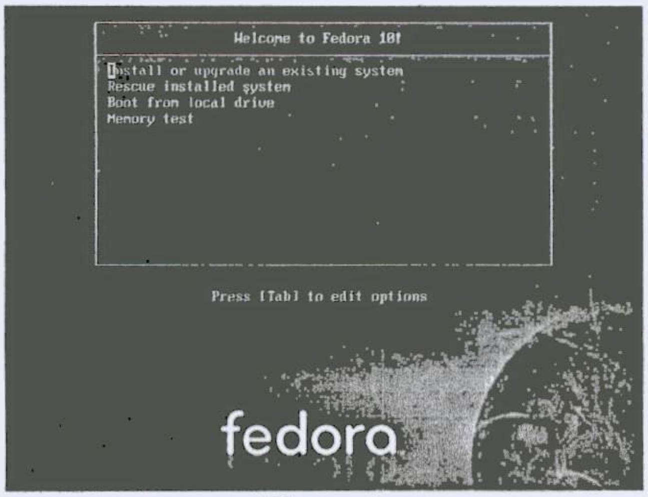
- To know about the configuration of Linux system
- To know about the Linux internals such as Process details, Memory Management
- To learn the basics of process creation and IPC mechanisms
- To learn the basics of user management, network management and administration utilities
- To know about device drivers and configuration of devices

Exercise-1: Installing Fedora Linux (One of the Linux Distributions)

The operations described here are almost similar with any other distributions such as RedHat, SuSE etc.

To install F10 Linux

1. Power on your computer and insert the first installation media
2. The screen below appears (Figure 1). Press <ENTER>
3. Select ***Skip*** and press <ENTER> at the ***Disc Found*** screen (Figure 2).
   (If you are performing a network based installation you will not get this screen)

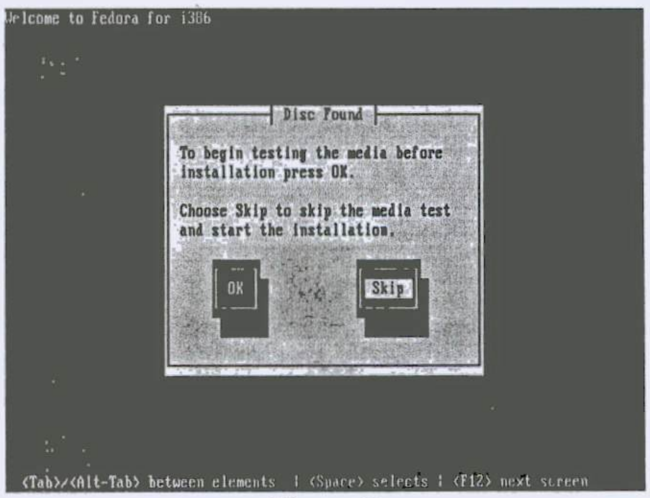


```
Welcome to Fedora 10!

Install or upgrade an existing system
Rescue installed system
Boot from local drive
Memory test

Press [Tab] to edit options
```

Figure -1



```
Welcome to Fedora for i386

Disc Found
To begin testing the media before
installation press OK.

Choose Skip to skip the media test
and start the installation.

<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen
```

Figure 2

4. Press ***Next*** at the following screen (Figure 3)
5. At the ***Language Selection*** Screen, select "English (English)" and press ***Next*** (Figure 4)

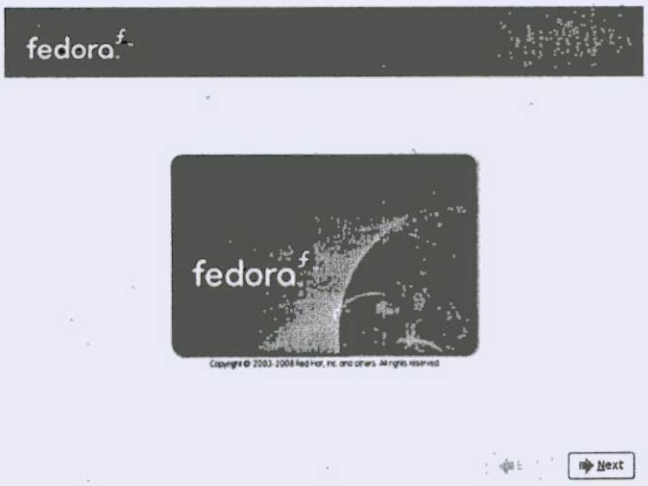
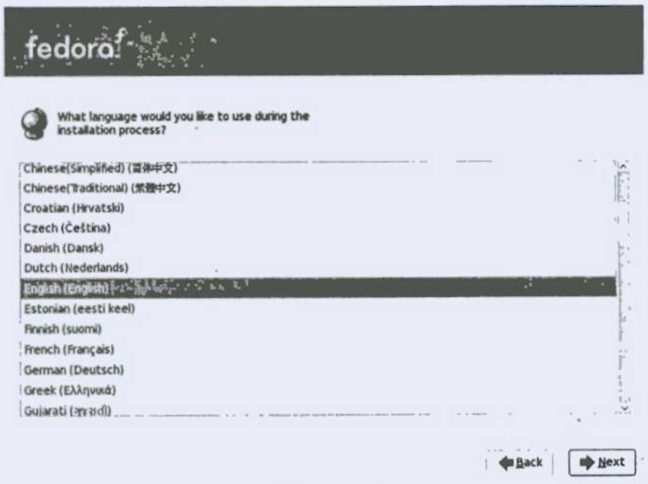

fedora
fedora
Copyright © 2003-2008 Red Hat, Inc. and others. All rights reserved.
Next

Figure -3

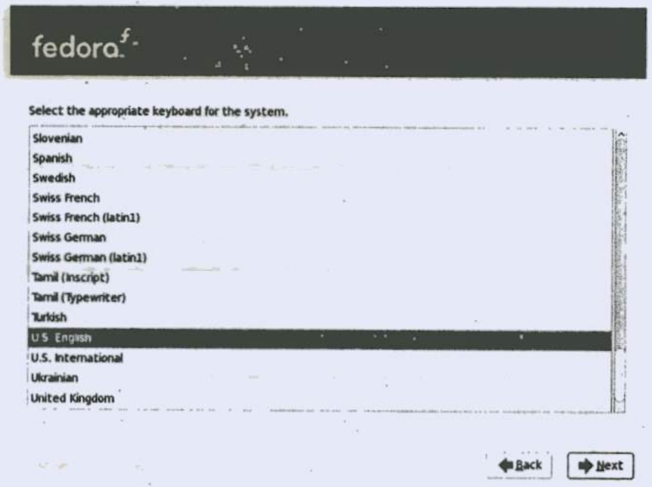


What language would you like to use during the installation process?

```
Chinese(Simplified) (简体中文)
Chinese(Traditional) (繁體中文)
Croatian (Hrvatski)
Czech (Čeština)
Danish (Dansk)
Dutch (Nederlands)
English (English)
Estonian (eesti keel)
Finnish (suomi)
French (Français)
German (Deutsch)
Greek (Ελληνικά)
Gujarati (ગુજરાતી)
```

Figure 4

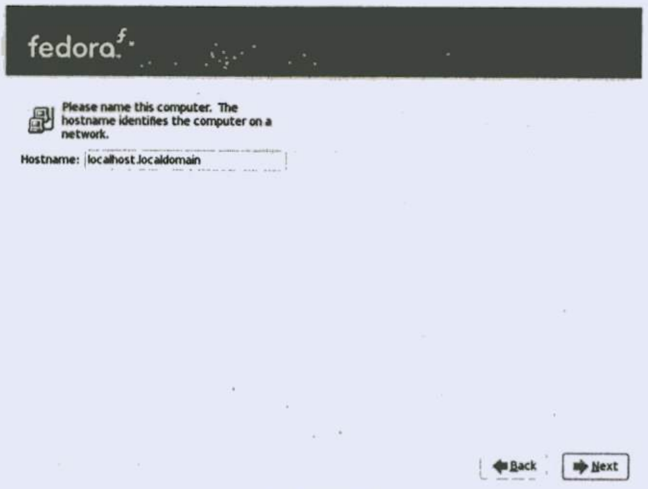
6. Choose your keyboard type and press ***Next*** .
7. Accept the defaults in the ***Hostname Configuration*** screen (Figure 6) and click ***Next***



Select the appropriate keyboard for the system.

Slovenian
Spanish
Swedish
Swiss French
Swiss French (latin1)
Swiss German
Swiss German (latin1)
Tamil (Inscript)
Tamil (Typewriter)
Turkish
U.S. English
U.S. International
Ukrainian
United Kingdom

Figure -5

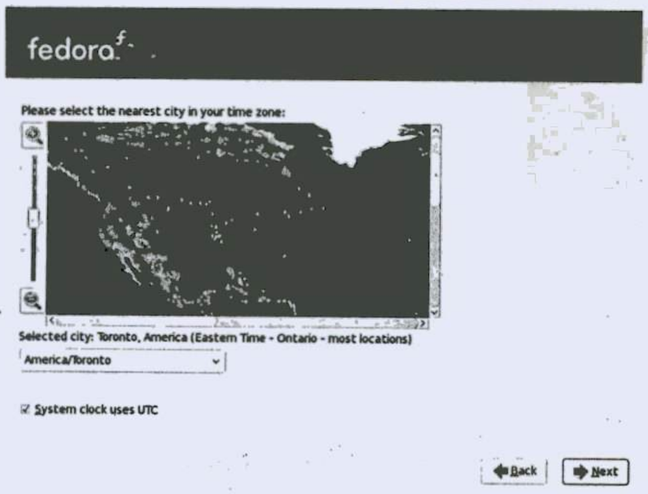


Please name this computer. The hostname identifies the computer on a network.

Hostname: localhost.localdomain

Figure 6

8. Using the map, select the city closest to your timezone. And then click "Next".
9. Type in "password" (in both fields) as your root password and click *Next* (Figure 8). Click "Yes" if you getting a "Weak Password" warning.



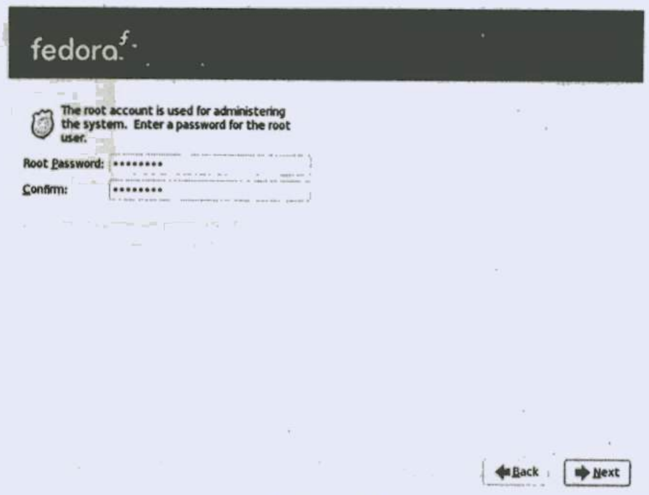
Please select the nearest city in your time zone:

Selected city: Toronto, America (Eastern Time - Ontario - most locations)

America/Toronto

☑ System clock uses UTC

Figure -7



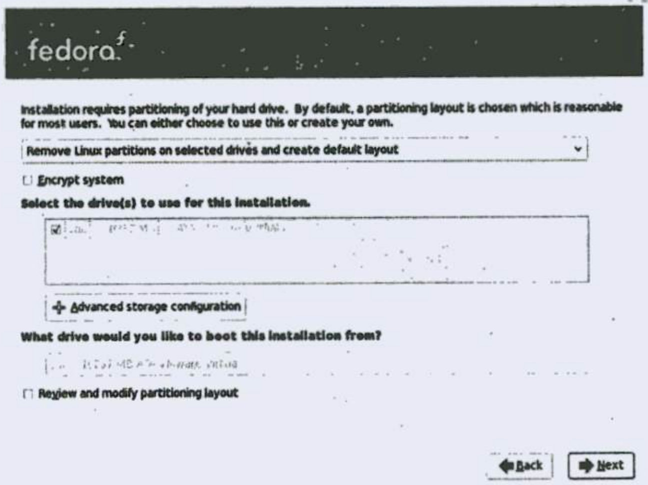
The root account is used for administering the system. Enter a password for the root user.

Root Password: ••••••••

Confirm: ••••••••

Figure 8

10. In the disk partitioning screen, accept the defaults (Remove Linux partitions on selected drives and create default layout) and click "Next" (Figure 9). Click "Write changes to disk" if you get a "Writing partitioning to disk" dialog box.
11. In the ***Package Group Selection*** screen, UN-CHECK the "Office and Productivity" option and click "Next".


Installation requires partitioning of your hard drive. By default, a partitioning layout is chosen which is reasonable for most users. You can either choose to use this or create your own.

Remove Linux partitions on selected drives and create default layout

☐ Encrypt system

Select the drive(s) to use for this installation.

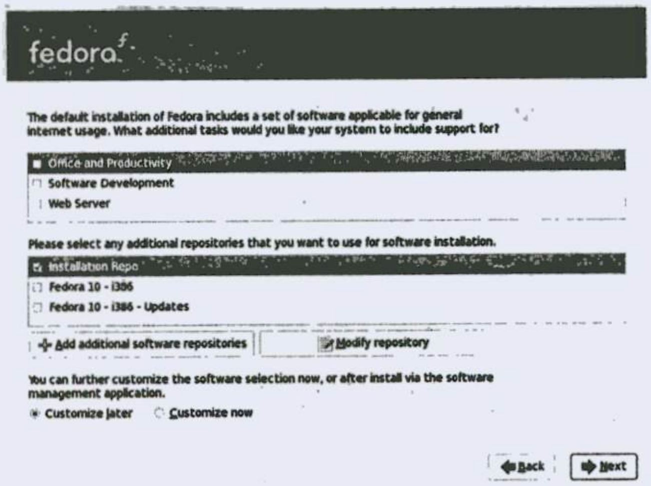
☑ [illegible]

What drive would you like to boot this installation from?

[illegible]

☐ Review and modify partitioning layout

Figure -9


The default installation of Fedora includes a set of software applicable for general internet usage. What additional tasks would you like your system to include support for?

■ Office and Productivity
☐ Software Development
Web Server

Please select any additional repositories that you want to use for software installation.

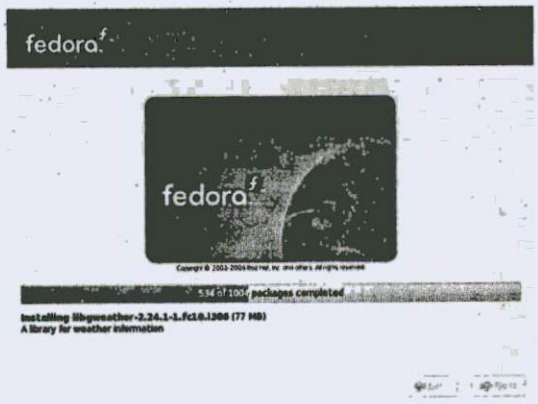
☑ Installation Repo
☐ Fedora 10 - i386
☐ Fedora 10 - i386 - Updates

You can further customize the software selection now, or after install via the software management application.

● Customize later ○ Customize now

Figure 10

12. The installation will begin (Figure 11- shown below).


Copyright © 2003-2008 Red Hat, Inc. and others. All rights reserved.

534 of 1004 packages completed

Installing libgweather-2.24.1-1.fc10.i386 (77 MB)
A library for weather information

13. Once the installer runs to completion, you will be prompted to reboot the system.

14. After rebooting, you will be walked through some quick customization steps. Click Forward at the first screen.

15. Memorize the entire license governing the software you have just installed - because whatever you agree to in the license can and will be used against you in a court of law. Just kidding ....Click forward again at the License Information screen.

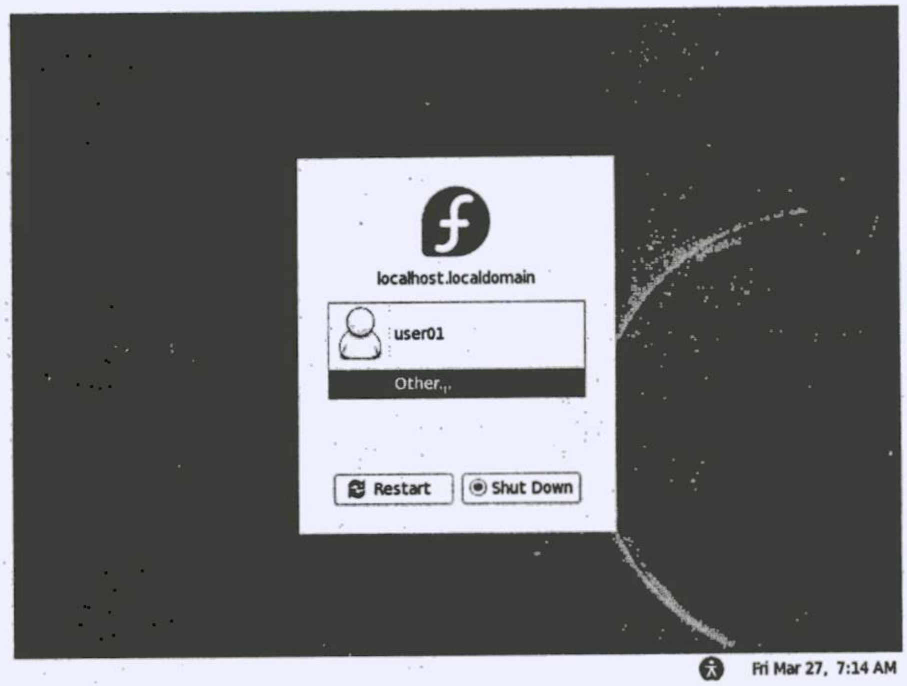
16. The next screen will allow you to create a new user account. We will create a new user named "user01". Complete the fields in the screen with the information below:
Click Forward when done.

```
Username: user01
Full Name: user01
Password: 01user01
Confirm Password: 01user01
```

17. Make sure that the current date and time in the "Date and Time" screen is correct and then click ***Forward.***

18. In the Hardware Profile screen, you may choose or refuse to contribute information to the Fedora project using Smolt. Make your choice
and then click Finish.

19. You will be presented with the Fedora login screen (Figure 12). From this point, you can logon to the system as the user "user01" with the password ***01user01*** that
was created earlier.



20. Done.

Exercise-2: To create a new partition of memory

In this exercise, you will create the partitions on the free space you left on the drive during the initial installation. Partitioning a disk allows the disk to be regarded as a group of independent storage areas. Partitions also make backups easier and help to restrict and confine potential problem areas.

Hard disk space is not infinite and one of your duties as an administrator is to manage the little space you have. For instance a simple way to restrict the total storage area on a disk that users can store their personal files is to create a separate partition for the user's home directory (Of course quotas can also be used).

You will be using the *fdisk* utility

1. While logged on as root, display the current structure of your disk. Type:

```
    root@serverXY root# fdisk -l
```

```
Disk /dev/hda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/hda1 * 1 25 200781 83          1 25 200781 83     Linux
/dev/hda2 26 752 5839627+ 8e Linux LVM
```

2. Display the current disk usage statistics. Type:

```
    root@serverXY root# df -h
```

```
Filesystem               Size    Used          Avail   Use%   Mounted on
/dev/mapper/VolGroup00           -Log                         Vol01
4.9G                     4.6G    0             100% /
/dev/hda1                190M    9.9M          171M    6%     /boot
/dev/shm                 125M    0             125M    0%     /dev/shm
```

From the sample output above under the *Use%* column, you can see that the primary partition (/dev/hda1) on which our root (/) directory is mounted on is completely (100%) filled up. Of course your output might be different if you have a different sized disk or if you did not follow the partitioning scheme used during the OS install.

3. You will use the *fdisk* program to create a new partition.

The *fdisk* program is interactive. Type:

```
    root@serverXY root# fdisk /dev/had
```

Type *m* for help
Command (m for help): m
Command action

- a toggle a bootable flag
- b edit bsd disklabel

- c toggle the dos compatibility flag
- d delete a partition
- l list known partition types
- m print this menu
- n add a new partition
- o create a new empty DOS partition table
- p print the partition table
- q quit without saving changes
- s create a new empty Sun disklabel
- t change a partition's system id
- u change display/entry units
- v verify the partition table
- w write table to disk and exit
- x extra functionality (experts only)

Type ***n*** to add a new partition

Command (m for help): ***n***

Type ***p*** to create a primary partition

- Command action
- e extended
- p primary partition (1-4)
- p

Type ***3*** as the partition number (i.e. /dev/hda3)

Partition number (1-4): ***3***

Press <ENTER> to accept the default first cylinder ***(753)***

First cylinder (753-1305, default 753): <ENTER>

Per our original stated goal we will create a partition that is the size of the current root filesystem. The sample system we are using has a 10 GB hard drive and the size of slash (/ = root filesystem) being approximately equal to 5000MB. Therefore the new partition will be approximately 3750 MB in size.

You need to make the necessary adjustments on your system to reflect the correct size of your own hard drive. e.g. if you are using a 4 GB hard drive and your root filesystem is 2GB in size; then you want to create a new partition that is roughly 1500MB (1.5GB) in size.

To create a partition that is 3750 MB (1 GB) in size. Type ***+3750M*** below

Last cylinder or +size or +sizeM or +sizeK (753-1305, default 1305): ***+3750M***

Type ***p*** below to view (print) your changes.

Disk /dev/hda: 10.7 GB, 10737418240 bytes

```
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/hda1 * 1 25 200781 83          1 25 200781 83     Linux
/dev/hda2 26 752 5839627+ 8e Linux LVM
/dev/hda3 753 1209 3670852+ 83 Linux
```

The new partition you created is the one on ***/dev/hda3*** above.
You will notice that the partition type is ***83***.

Next we will change the partition type from 0x83 (i.e. ext2) to 0x8e (LVM).

Type ***t*** to change the partition type:

Command (m for help): ***t***

We want to change the type for 3rd partition that we just created (i.e. /dev/hda3), so type ***3*** as the partition number.

Partition number (1-4): ***3***

The Hex code for the Linux LVM type partition is 8e, so type 8e when prompted for the Hex Code.

Hex code (type L to list codes): ***8e***

Changed system type of partition 3 to 8e (Linux LVM)

Type ***p*** below to view (print) your changes.

Command (m for help): ***p***

```
Disk /dev/hda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/hda1 * 1 25 200781 83          1 25 200781 83     Linux
/dev/hda2 26 752 5839627+ 8e Linux LVM
/dev/hda3 753 1209 3670852+ 8e Linux LVM
```

Type ***w*** to write the partition table to disk and exit

Command (m for help): ***w***

The partition table has been altered!

Calling ***ioctl()*** to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.

The kernel still uses the old table.

The new table will be used at the next reboot.

4. As per the warning message you may have gotten after writing the partition table to disk in the previous step, you may need to reboot your system. Type:

```
root@serverXY root# shutdown -r 2
```

After rebooting the system......

When making critical changes to the hard disk structure, it is good practice to do this in single user mode, which prevents other users from continually trying to access the disk while you make the changes.

Your X window system may also have problems coming up because of lack of free space, so you should perform the following steps while in single user mode.

1. Log back into the system as the superuser. Switch to single user mode. Type:

```
root@serverXY root# init s
```

2. Run the disk free ***df*** command again to see how much free usable space you have. Type:

```
root@serverXY root# df -h
```

```
Filesystem              Size        Used        Avail       Use%   Mounted on

/dev/mapper/VolGroup    00          -Log                           Vol01
4.6G                    4.6G        11M         100% /
/dev/hda1               190M        9.9M        171M        6%      /boot
/dev/shm                125M        0           125M        0%     /dev/shm
```

Exercise-3: To create a Physical Volume

First we will create a physical volume and assign the free space in /dev/hda3 to it.

1. Use the ***pvdisplay*** command to view the physical volumes currently defined on the system. Type:

```
root@serverXY ~# pvdisplay
```

```
--- Physical volume ---
PV Name: /dev/hda2
VG Name: VolGroup00?
PV Size: 5.56 GB / not usable 0
Allocatable: yes
PE Size (KByte): 32768
Total PE: 178
Free PE: 2
Allocated PE: 176
PV UUID D2oFKM-iOfB-DeHd-6Uyo-8jLO-kmXO-CI7f43
```

2. Initialize the /dev/hda3 partition as a physical volume. Type:

```
root@serverXY ~# pvcreate /dev/hda3
```

```
Physical volume "/dev/hda3" successfully created
```

To assign a Physical volume to a Volume Group

We next need to assign the physical volume (PV) that was created above to a Volume Group (VG)

1. Use the ***vgdisplay*** command to view the currently configured volume groups. Type:

```
root@serverXY ~# vgdisplay
```

```
--- Volume group ---

VG Name VolGroup00?

System ID

Format lvm2

VG Size 5.56 GB

PE Size 32.00 MB

Total PE 178

Alloc PE / Size 176 / 5.50 GB

Free PE / Size 2 / 64.00 MB

VG UUID UnJx7C-Bn8Z-GPcD-4Omt-8Gcd-XGxV-l70T06?
```

NOTES: From the output above, it is found that the volume group name is VolGroup00, the size of the VG is 5.56 GB and there are only 2 physical extents (PE) that are free in the VG, which is equivalent to approximately 64 MB of space.

2. Assign the PV to the ***VolGroup00*** volume group. Use the ***vgextend*** command, type:

```
root@serverXY ~# vgextend VolGroup00? /dev/hda3
```

Volume group "VolGroup00" successfully extended

3. Run the vgdisplay command again to view your changes. Type:

```
root@serverXY ~# vgdisplay
```

```
VG Name VolGroup00?
System ID
Format lvm2
Cur PV 2
Act PV 2
VG Size 9.06 GB
PE Size 32.00 MB
Total PE 290
Alloc PE / Size 176 / 5.50 GB
Free PE / Size 114 / 3.56 GB
VG UUID UnJx7C-Bn8Z-GPcD-4Omt-8Gcd-XGxV-170T06?
```

NOTES: From the output above, it is found that the VG size has increased to 9.06 GB and the free physical extents has also increased to 3.56 GB from 64 MB.

Exercise-4: To create a Logical Volume

Since we have been able to create some free space in the volume group, we can now go ahead to create the Logical Volume (LV) that will house the actual /home file system.

1. Use the ***lvdisplay*** command to view the currently configured logical volumes. Type:

```
root@serverXY ~# lvdisplay
```

```
LV Name /dev/VolGroup00/LogVol01
VG Name VolGroup00?
LV Status available
# open 1
LV Size 5.00 GB
Current LE 160
Segments 1
Allocation inherit
Read ahead sectors 0
Block device 253:0
```

```
--- Logical volume ---
    LV Name /dev/VolGroup00/LogVol00
    VG Name VolGroup00?
    LV Status available
    # open 1
    LV Size 512.00 MB
    Current LE 16
    Segments 1
    Allocation inherit
    Read ahead sectors 0
    Block device 253:1
```

*NOTES: From the output above, it is observed that: There are currently 2 logical volumes defined. The first one is **/dev/VolGroup00/LogVol00**. There is the SWAP filesystem. The second one is **/dev/VolGroup00/LogVol01**. This is the root (/) filesystem.*

2. Create a new logical volume called □LogVol02□ using the lvcreate command. We will use up the entire free space (i.e 3.56 GB) on the volume group for the logical volume (as shown in the vgdisplay command). Type:

```
root@serverXY ~# lvcreate --size 3.56G --name LogVol02? VolGroup00?
```

```
Rounding up size to full physical extent 3.56 GB
Logical volume "LogVol02" created
```

3. Use the ***lvdisplay*** command again to view the new LV. Type:

```
root@serverXY ~# lvdisplay /dev/VolGroup00/LogVol02
```

```
--- Logical volume ---

        LV Name /dev/VolGroup00/LogVol02
        VG Name VolGroup00?
        LV UUID SINQr0-oSrx-mbba-kEWq-J0vx-T6mI-s2Nqc7
        LV Size 3.56 GB
        Current LE 114
        Segments 2
        Allocation inherit
        Read ahead sectors 0
        Block device 253:2
```

Exercise-5: To create a filesystem

To make the volume that was created earlier usable by the Operating system, you need to create a filesystem on it. Here you will create an ext3 journaling file system, using the mke2fs program.

1. Use the ***mkfs.ext3*** utility to create an ext3 type filesystem on the /dev/VolGroup00/LogVol02 volume. Type:

```
root@serverXY ~# mkfs.ext3 /dev/VolGroup00/LogVol02
```

```
mke2fs 1.37 (21-Mar-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
467712 inodes, 933888 blocks
46694 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=956301312
29 block groups
32768 blocks per group, 32768 fragments per group
16128 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912, 819200, 884736
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

This filesystem will be automatically checked every 36 mounts or 180 days, whichever comes first. Use tune2fs -c or -i to override.

TIP:

You may need to temporarily disable the SELINUX on your system for the above command to work. To do this, issue the command ***setenforce 0*** as the super-user.

(a) To use various filesystem utilities

Here we will walk through the use of some common filesystem utilities that can be used in maintaining the filesystem, fixing filesystem problems, debugging filesystem issues etc.

1. Find out the value of the current ***maximal mount count*** on the /dev/VolGroup00/LogVol02 volume. Type:

```
root@serverXY ~# dumpe2fs /dev/VolGroup00/LogVol02 | grep -i "maximum mount count"
```

```
dumpe2fs 1.37 (21-Mar-2005)

Maximum mount count: 36
```

2. Adjust/set the maximal mount count value to zero between filesystem checks on the /dev/VolGroup00/LogVol02 volume. Use the tune2fs command. Type:

```
root@serverXY ~# tune2fs -c 0 /dev/VolGroup00/LogVol02

tune2fs 1.37 (21-Mar-2005)

Setting maximal mount count to -1
```

3. Use the *fsck* command to check the file system you just created above. Type:

```
root@serverXY root# fsck -Cfp /dev/VolGroup00/LogVol02

fsck 1.37 (21-Mar-2005)

/dev/VolGroup00/LogVol02: |=================================== \ 99.9%

/dev/VolGroup00/LogVol02: 11/467712 files (9.1% non-contiguous), 24714/933888 blocks
```

Exercise-6: To relocate /home to the new filesystem

You have almost saved the day, but you need to move the contents of the /home directory to the new separate filesystem that was just created for it.

1. First you will create a new directory (***/temphome***) to temporarily mount the new partition on. Type:

```
root@serverXY root# mkdir /temphome
```

2. Mount the new partition. Type:

```
root@serverXY ~# mount /dev/VolGroup00/LogVol02 /temphome
```

3. Display the amount of free space on the system now. Type:

```
root@serverXY ~# df -h
```

4. Copy the contents of the /home directory to the /temphome directory

```
root@serverXY root# cp -var /home/* /temphome
```

*NOTE: You may run out of space on the destination device (/dev/VolGroup00/LogVol02) while making the copy above. This may be due to disparity in size between the space currently used by /home and the space available on **/temphome**. This is not a serious problem in our example here. Simply find one of the LARGE-USELESS-FILE.tar files that are unreasonable downloaded and delete it to enable you continue with the lab. Then wipe the **/temphome** directory clean and retry the copy command above again.*

5. Delete the entire contents of the □/home□ directory. Type:

```
root@serverXY ~# rm -rf /home/*
```

6. Re-mount the file system on the ***/dev/VolGroup00/LogVol02*** partition at the ***/home*** directory. Type:

```
root@serverXY root# mount /dev/VolGroup00/LogVol02 /home
```

7. Un-mount the ***/temphome*** directory. And display your disk usage again.

```
root@serverXY root# umount /temphome

root@serverXY root# df -h
```

How much free space do you have on the root partition now?

8. You should create a volume label for your new partition using the ***tune2fs*** program. But first ***unmount*** the file system you want to label.

```
root@serverXY root# umount /home && tune2fs -L /home /dev/VolGroup00/LogVol02

        tune2fs 1.37 (21-Mar-2005)
```

9. You need to make your changes (mount points etc) permanent between reboots. Use a pager to view your ***/etc/fstab*** file. Type:

```
root@serverXY ~# more /etc/fstab
```

```
This file is edited by fstab-sync - see 'man fstab-sync' for details

/dev/VolGroup00/LogVol01 / ext3 defaults 1 1
LABEL=/boot /boot ext3 defaults 1 2
/dev/devpts /dev/pts devpts gid=5,mode=620 0 0
/dev/shm /dev/shm tmpfs defaults 0 0
/dev/proc /proc proc defaults 0 0
/dev/sys /sys sysfs defaults 0 0
/dev/VolGroup00/LogVol00 swap swap defaults 0 0
/dev/fd0 /media/floppy auto pamconsole,exec,noauto,managed 0 0
/dev/hdc /media/cdrom auto pamconsole,exec,noauto,managed 0 0
```

You will use the cat command to append a new entry for the ***/home*** mount point to the ***/etc/fstab*** file. Type:

```
root@serverXY root# cat >> /etc/fstab << over

>
> LABEL=/home /home ext3 defaults 1 0
> over
```

10. Restore the security contexts for the /home directory, using the *restorecon* command. Type:

```
root@serverXY ~# restorecon -R /home
```

11. Reboot the system and make sure everything works fine and that the */home* partition got mounted automatically. Make sure your users can still log in.

Exercise-7: Starting and shutting down the Linux system.
It uses - *shutdown, reboot and init (change in the runlevel)*

7 (a) Starting Up and changing runlevels

In this exercise you will reboot and shutdown the computer properly

To use the shutdown and reboot commands

1. At the Login prompt, type ***root*** and press [ENTER]
2. At the password prompt type ***password*** and press [ENTER]
3. While logged into the system as the superuser (root), reboot the system after 2 minutes delay: Type

```
      [root@localhost root]# shutdown r 2
```

Broadcast message from root (pts/0) Thu Apr 25 19:13:25 2002...
The system is going DOWN for reboot in 2 minutes !!

The above message will be seen on the terminals of all logged on users
4. After the system reboots, log into the system and try the ***reboot*** command. Type:

```
      [root@localhost root]# reboot
```

7(b) To use the init command

After the system reboots, you will change the current runlevel to reboot the system again

1. Log in as root
2. Type in the password - "password"
3. Find out the current runlevel. Type:

```
      [root@localhost root]# runlevel
```

   Output: N 3

The above shows that you are currently in runlevel 3. (i.e. Command line multi-user mode)
4. Change to runlevel 6 (runlevel 6 is interpreted by most versions of Linux to mean reboot)

```
      [root@localhost root]# init 6
```

The system will go down for an immediate reboot without any warning message.

7(c) To shutdown the system without rebooting

Next you will shutdown the system without rebooting.

1. While logged in as root, first send a customized test warning message to the console of all logged on users without actually bring down the system. The message you will send is ***I am going out for lunch break***. Type:

```
[root@localhost root]# shutdown -k now "I am going out for lunch"
```

Broadcast message from root (pts/4) (Tue Jan 7 17:00:12 1943):

```
I am going out for lunch
The system is going down to maintenance mode NOW!
```

Do you know what the ***now*** in the command above does?

2. Now do an actual shutdown of the system after 2 minutes. Type:

```
      [root@localhost root]# shutdown -h 2
```

```
Broadcast message from root (pts/1) (Tue Jan 7 17:13:32 1943):
The system is going DOWN for system halt in 2 minutes!
```

7(d) To poweroff the system

1. If your system is not powered on, turn it on and log in as the superuser.
2. Use the poweroff command to bring down the system. Type:

```
      [root@localhost root]# poweroff
```

Exercise-8: Common System Utilities

8(a) To use cd

cd stands for change directory. You will start these labs by changing to other directories on the file system.

1. Log in to the computer as root

2. Change from your current directory to the /etc directory.

```
      [root@localhost root]# cd /etc
```

3 . Note that your prompt has changed from `[root@localhost root]#` to : `[root@localhost etc]#`

4. Change to the ***/usr/local/*** directory

```
      [root@localhost etc]# cd /usr/local
      [root@localhost local]#
```

What has changed about your prompt?

5. Change back to root's home directory

```
      [root@localhost local]# cd /root
```

6. Change to the /usr/local/ directory again. Type

```
      [root@localhost root]# cd /usr/local
```

7. To change to the parent directory of the local directory type ***cd ..***

```
      [root@localhost local]# cd ..
```

What is the parent directory of the /usr/local/ directory?

8. To quickly change back to roots home directory type ***cd*** without any argument.

```
      [root@localhost usr]# cd
      [root@localhost root]#
```

8(b) To use pwd

pwd stands for present working directory. It shows the location you are in on the file system.

1. To find out your current working directory type

```
      [root@localhost root]# pwd
      output: /root
```

2. Change your directory to the /usr/local/ directory using the ***cd*** command

```
      [root@localhost root]# cd /usr/local
```

3. Use pwd to find your present working directory

```
      [root@localhost local]# pwd
      output: /usr/local
```

4. Return to root's home directory.

```
      [root@localhost root]# cd
```

8(c) To use mkdir

The mkdir command is used to create directories. You will create two directories called *folder1* and *folder2*

1. Type
```
     [root@localhost root]# mkdir folder1
```

2. Create a second directory called folder2
```
     [root@localhost root]# mkdir folder2
```

3. Now change your working directory to the *folder1* directory you created above.
```
     [root@localhost root]# cd folder1
```

4. Display your current working directory.
```
     [root@localhost folder1]# pwd
     /root/folder1
```

8(d) To use ls

The ***ls*** command stands for list. It lists the contents of a directory.

1. Type ***ls*** in root's home directory
```
     [root@localhost root]# ls
     List the contents of the directory:
```
2. Change to the folder1 directory

3. List the contents of *folder1* directory. Type ***ls***
```
     [root@localhost folder1]# ls
     file11 file12
```

4. Change to the folder2 directory and list its contents here:

5. Change back to your home directory and list all the hidden files and folders.
```
     [root@localhost folder2]# cd
     [root@localhost root]# ls
```
.. .bash_history .bash_logout .bash_profile .bashrc folder1 folder2 .gtkrc .kde screenrc

6. To obtain a long or detailed list of all the files and folders in your home directory type:
```
     [root@localhost root]# ls al
     total 44
     drwx------ 5 root root 4096 May 8 10:15 .
     drwxr-xr-x 8 root root 4096 May 8 09:44 ..
     -rw------- 1 root root 43 May 8 09:48 .bash_history
     -rw-r--r-- 1 root root 24 May 8 09:44 .bash_logout
     -rw-r--r-- 1 root root 191 May 8 09:44 .bash_profile
     -rw-r--r-- 1 root root 124 May 8 09:44 .bashrc
     drwxrwxr-x 2 root root 4096 May 8 10:17 folder1
```

Exercise-9: Creation of User

9(a) To create the user manually

A.) Editing the /etc/passwd file

1. Launch your editor of choice and open up the file */etc/passwd*
2. Add the text below to the bottom or end of the file :

```
me:x:500:500:me mao:/home/me:/bin/bash
```

3. Save your changes and close the *passwd* file.

B.) Editing the /etc/shadow file

5. Launch your editor and open up the */etc/shadow* file.
6. Add a new entry like the one below to the bottom of the file - put an asterisk (*) in the

```
password field. Type:
me:*:11898:0:99999:7:::
```

7. Save your changes and close the shadow file.

C.) Editing the /etc/group file.

8. Launch your editor and open up the */etc/group* file.
9. At the bottom of the file add a new entry like:

```
me:x:500:me
```

10. Save your changes and close the group file.

D.) Creating the home directory

11. Copy the entire contents of the */etc/skel* directory into /home directory, renaming the new directory to the name of the user i.e. */home/me*. Type :

```
[root@localhost root]# cp   r   /etc/skel    /home/me
```

12. The root user owns the directory you just created, because she created it. In order for the user me mao to be able to use the directory you will change the permissions/ownership of the folder. Type:

```
[root@localhost root]# chown  r me:me  /home/me
```

E.) Creating a password for the user.

13. You will use the *passwd* utility. Type *passwd* and follow the prompts (the new password is *a1b2c3*):

```
[root@localhost root]# passwd   me
```

Changing password for user *me*.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.

14. Logout of the system when you are done.

9(b)To create the user automatically

There are loads of utilities available to simplify all the tasks/steps that we manually performed in the previous exercise. We only walked through the manual process of creating a user, so that you can see what actually goes on in the background.

In this exercise we will use some common utilities to manage and simplify the process. You will create another user account for the user *Ying Yang* the login name will be *ying*.

And the password for ying will be *y1i2n3*.

You will also create a group called *common* and add the user me and *ying* to the group.

To automatically create a new account

1. Login to the system as root.
2. You will create the user ying using all the defaults of the *useradd* command. Type:

```
     root@localhost root# useradd -c "Ying Yang" ying
```

3. Use the tail command to examine the addition you just made to the /etc/passwd file. Type:

```
     root@localhost root# tail -n 4 /etc/passwd
     pcap:x:77:77::/var/arpwatch:/sbin/nologin
     me:x:500:500:me mao:/home/me:/bin/bash
     ying:x:501:501: Ying Yang:/home/ying:/bin/bash
```

List the new entry here?
4. The user ying will not be able to login to the system until you create a password for the user. Type:

```
     root@localhost root# passwd ying
```

Changing password for user ying.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.

5. Use the ***id*** utility to quickly view information about the new users you just created. Type:

```
     root@localhost root# id me
     uid=500(me) gid=500(me) groups=500(me)
```

and then type:

```
     root@localhost root# id ying
     uid=501(ying) gid=501(ying) groups=501(ying)
```

To automatically create a new group
1. Use the groupadd program to create the new group *common.*

```
     root@localhost root# groupadd common
```

2. Examine the tail end of the /etc/group file to see the new addition. What is the

command to do this?

3. Unfortunately there is not standard program to add existing users to a group. (except you do it while creating the user. You will do it manually by editing the /etc/group file. Run the tail command on the group file.

```
        root@localhost root# tail -n 5 /etc/group
        me:x:500:me
        ying:x:501:
        common:x:502:
```

4. Edit the last entry for the common group you created by appending *me,ying* after the last semi-colon.
5. Launch your text editor and change the last line in the group file from:

```
        common:x:502:
        Change To:
        common:x:502:me,ying
```

6. Save your changes and exit the group file.

7. Run the ***id*** command again on user ying. What has changed?

To modify a user account

1. Use the usermod command to change the comment field for the user *me*. The new comment you
will add will be *first last*. Type:

```
root@localhost root# usermod -c "first last" me
```

Use the tail command to examine your changes to the /etc/passwd file. Write the changed line below?

2. What is the user *me*s login shell ?

3. Use the ***usermod*** command again to change me's login shell to the csh shell. Type:

```
root@localhost root# usermod -s /bin/csh me
```

4. Finally use the ***usermod*** command to undo all the changes you made to the user *me* above. Return the values (login shell etc..) to their original values.

Exercise-10: Installing, Querying and Uninstalling Packages

You will learn how to use the Red Hat package management system and you will also install the Openoffice.org word processing application on your system.

You will use the Red Hat package manager (RPM) to perform the install because the Red Hat package manager has become the standard package installation tool for major distributions such as Fedora, SuSE, Mandrake and so on.

As with most packages, there are issues of dependencies to deal with. This results mostly from issues of shared libraries.
When trying to install a package, the package manager (RPM) may not always tell you what other rpm packages you need to satisfy these dependencies in-which case it will usually give you clues as to the exact files (libraries) you need.

PREAMBLE:

Before proceeding, with the following exercises, download the following files from your software distribution location (ftp server, web server, DVD-ROM or CD-ROM media).

1. openoffice.org-writer-*.rpm - wordprocessor application of openoffice.org
2. openoffice.org-core-*.rpm - core libraries and support files for openoffice.org
3. libwpd-0*.rpm - Library that handles Word Perfect documents.
4. curl-7*.rpm - A utility for getting files from remote servers
5. libidn-0*.rpm - Internationalized Domain Name support library

Assuming that you want to install the openoffice.org software package from the install Fedora Core 4 DVD-ROM media and that the DVD-ROM is mounted at the /media/dvd mount point you will find all the files referenced in the list above under the /media/dvd/Fedora/RPMS directory.

You can also manually download the files needed directly from the any of the Fedora Core 4 mirror sites. For example:

http://mirrors.kernel.org/fedora/core/4/i386/os/Fedora/RPMS/
or
http://download.fedora.redhat.com/pub/fedora/linux/core/4/i386/os/Fedora/RPMS/

NOTES:

The operating system (OS) installation exercise performed in Lab 1 recommended a specific group/list of applications to select during the initial OS install. The *Office/Productivity* group was NOT on the list. One of the reasons for this was because of future exercises like the one in this section.

If you selected the *Office/Productivity* group of packages during the initial operating system installation, then you already have the *openoffice.org-writer** word processing application installed on your system. This is because the openoffice.org suite is grouped under the *Office/Productivity* package group. As a result, some of the steps in the following section will not be very relevant to you since you already have the software installed.

10(a)To query packages for information

1. To see a list of all the packages currently install on your local system type:

```
root@localhost root# rpm qa
libgcc-4.0.0-8
basesystem-8.0-5
glibc-2.3.5-10
device-mapper-1.01.02-1.0
ethtool-3-1
```

You should get a very long list.

2. You will use one of the numerous switches of the rpm command to obtain information about the openoffice.org-writer- *.rpm package.

Mke you sure are in the directory that contains the *openoffice.org-writer- *.rpm* package that you downloaded earlier, then type (The steps below are running the commands from the /media/dvd/Fedora/RPMS/ directory on our sample system with the FC4 DVD-ROM mounted):

```
root@localhost RPMS# rpm qip openoffice.org-writer-*rpm

Warning: openoffice.org-writer-*.rpm: Header V3 DSA signature: NOKEY, key ID 4f2a6fd2

Name : openoffice.org-writer Relocations: (not relocatable)

Version : 1.9.104 Vendor: Red Hat, Inc.

Release : 2 Build Date: Tue 24 May 2005 04:59:17 AM PDT

Install Date: (not installed) Build Host: tweety.build.redhat.com

Group : Applications/Productivity Source RPM: openoffice.org-1.9.104-2.src.rpm

Size : 5933133 License: LGPL/SISSL

Signature : DSA/SHA1, Tue 24 May 2005 02:21:10 PM PDT, Key ID b44269d04f2a6fd2

Packager : Red Hat, Inc. http://bugzilla.redhat.com/bugzilla

URL : http://www.openoffice.org/

Summary : writer module for openoffice.org

Description : Word-processor application of openoffice.org
```

3. From your output in the previous step, what is the summary of description for the *openoffice.org-writer- *.rpm* package ?

4. If you are interested in the particular files that are contained in the openoffice.org-writer- *.rpm package, you could list all the files it provides by typing:

```
root@localhost RPMS# rpm qlp openoffice.org-writer-*rpm
warning: openoffice.org-writer-*rpm: Header V3 DSA signature: NOKEY, key ID 4f2a6fd2
```

```
/usr/bin/oowriter
/usr/lib/openoffice.org1.9.104
/usr/lib/openoffice.org1.9.104/help
```

5. It is possible to download a corrupted or tainted file. Verify the integrity of the package you - downloaded. Type:

```
root@localhost RPMS# rpm -K openoffice.org-writer-*rpm
openoffice.org-writer-*.rpm: (SHA1) DSA sha1 md5 (GPG) NOT OK (MISSING KEYS:
GPG#4f2a6fd2)
```

The above output normally might mean there is something wrong with the package. But before we come to that conclusion, we will first import/install the packager's (Fedora's) public key into the system.

To import a public key via rpm
1. There are several location from which you can obtain a vendors public key – vendors ftp or http site, distribution media, local source etc.. Import Fedora's public key from your local system. Type:

```
root@localhost RPMS# rpm --import /usr/share/rhn/RPM-GPG-KEY-fedora
```

2. After importing the public key, try verifying the package again. Type:

```
root@localhost RPMS# rpm -K openoffice.org-writer-*rpm
openoffice.org-writer-*.rpm: (sha1) dsa sha1 md5 gpg OK
```

Note that most of the output is in lower case now- This indicates success.

10(b)To install packages

Installing packages via rpm can be very easy sometimes and can be a pain at other times. In this example you will try to install the openoffice.org-writer package (openoffice.org-writer-*rpm).

While trying to install software on your system, you might stumble on issues of *failed dependencies.* For example if you try to install package *abc.rpm* the RPM installer might complain about some failed dependencies. It might tell you that package *abc.rpm* requires another package *xyz.rpm* to be installed first.

Dependencies can be regarded as prerequisites.

The output of the error you get, might help direct you to the package you need to install first, in order to satisfy the requirements of the original software.

In the exercise below your primary objective will be to try to install the *openoffice.org-writer-*rpm* package, but when trying to install it might complain about failed dependencies.

The *openoffice.org-writer* package that you will be installing below is one of the more tricky ones to install because it depends on a couple of other packages. Most other packages are easier to install.

1.cd to the location/directory that you downloaded the packages into.

2. Run a dry test to see what will happen if you try to install the openoffice.org-writer- *.rpm package. Type:

```
root@localhost RPMS# rpm -Uvh --test openoffice.org-writer-*rpm
error: Failed dependencies:
openoffice.org-core = 1:1.9.104-2 is needed by openoffice.org-writer-1.9.104-2.i386
libwpd >= 0.8.0 is needed by openoffice.org-writer-1.9.104-2.i386
libcomphelp4gcc3.so is needed by openoffice.org-writer-1.9.104-2.i386
libsb680li.so is needed by openoffice.org-writer-1.9.104-2.i386
```

NOTE: According to the output of the test above, the openoffice.org-writer package needs/requires that the openoffice.org-core and the libwpd* packages be installed. i.e. those two packages are a prerequisite to installing openoffice.org-writer. You can forcefully install the package using the* ***nodeps*** *option, if you absolutely know what you are doing. But this is generally a BAD practice.*

3. Query the openoffice.org-core package for information. Type:

```
root@localhost RPMS# rpm -qip openoffice.org-core-*.rpm
```

4. From your output in the previous step, what is the summary of description for the *openoffice.org-core* package and what version is it?

5. Run a test install of the *openoffice.org-core* package. Type:

```
root@localhost RPMS# rpm -Uvh --test openoffice.org-core-*.rpm
error: Failed dependencies:
curl is needed by openoffice.org-core-1.9.104-2.i386
libcurl.so.3 is needed by openoffice.org-core-1.9.104-2.i386
libwpd-0.8.so.8 is needed by openoffice.org-core-1.9.104-2.i386
```

More dependency issues: The first line of the sample output above suggests that a package called curl is needed by openoffice.org-core.

6. Run the dry-run test of an install of the suggested curl package. Type:

```
root@localhost RPMS# rpm -Uvh --test curl-7*.rpm
error: Failed dependencies:
libidn.so.11 is needed by curl-7.13.1-3.i386
```

The previous output recommends that a library named ***libidn.so**** is needed by curl. With any luck, the libidn*.rpm package will provide this library.

7. Install the Install the libidn-*.rpm package. We would not bother testing this time. We will just hope. Type:

```
root@localhost RPMS# rpm -Uvh libidn-0*.rpm
Preparing... ########################################### 100%
1:libidn ########################################### 100%
```

The output shows the ***libidn*** package was successfully installed and we have satisfied the prerequisite of the curl package.

8. Try to install the curl package. Type:

```
root@localhost RPMS# rpm -Uvh curl-7*.rpm
Preparing... ########################################### 100%
1:curl ########################################### 100%
```

curl is now installed.

9. Let us do a test run of the openoffice.org-core-*.rpm package again. Type:

```
root@localhost RPMS# rpm -Uvh — test openoffice.org-core-*.rpm
error: Failed dependencies:
libwpd-0.8.so.8 is needed by openoffice.org-core-1.9.104-2.i386
```

The previous error shows that we still need something that provides the library named ***libwpd***

10. Install the libwpd-0.*.rpm package. Type:

```
root@localhost RPMS# rpm -Uvh libwpd-0.*.rpm
Preparing... ############################################### 100%
1:libwpd ############################################### 100%
```

Good we are making some headway.

11. You can now install the openoffice.org-core package. Type:

```
root@localhost RPMS# rpm -Uvh openoffice.org-core-*.rpm

Preparing... ######################################### 100%
1:openoffice.org-core ######################################### 100%
```

12. Finally we can now achieve our primary objective - which was to install the openoffice.org-writer. Type:

```
root@localhost RPMS# rpm -ivh openoffice.org-writer-*.rpm
Preparing... ####################################### 100%
1:openoffice.org-writer ####################################### 100%
```

NOTES: RPM supports transactions. In the previous exercises we could have performed a single rpm transaction that included the original package we wanted to install as well as all the packages and libraries (i.e. openoffice.org-core, libwpd, curl etc) it depends on. This would have been possible since we already have already downloaded the relevant packages and we know the path to where the files are. A single command such as the one below would have sufficed:

```
root@localhost RPMS# rpm -Uvh openoffice.org-writer-*.rpm openoffice.org-core-*.rpm \ > libwpd-0*.rpm curl-7*.rpm libidn-0*.rpm
```

13. You may have noticed that when you installed the openoffice.org-writer package, you used the command - ***rpm -ivh*** (step 12) but when you installed the other packages package you used the command: ***rpm -Uvh*** (step 10, 11 etc).

Consult the man page for rpm and write down the difference between the -i and -U option when used with the rpm command.

TIPS: If you are in a GUI environment with X window running, you can use the ***system-config-packages*** program to quickly install and uninstall programs (or groups of programs) from your system. Type:

```
      root@localhost root# system-config-packages
```

10(c) To un-install packages

Un-installing packages is almost just as easy as installing, with Red Hat□s package manager (RPM). In this exercise you will try to un-install the libidn package you installed earlier.

1. Un-install the ***libidn*** package from your system. Type:

```
      root@localhost root# rpm -e libidn
error: Failed dependencies:
libidn.so.11 is needed by (installed) curl-7.13.1-3.i386
```

2. If you want to break the package that relies on libidn and forcefully remove the package from your system. Type:

```
root@localhost root# rpm -e --nodeps libidn
```

NOTES:

1. *The **nodeps** option means No dependencies. i.e. ignore all dependencies.*
2. *The above is just to show you how to forcefully remove a package from your system. There may be times when you need to this, but it is generally not a good practice.*
3. *Forcefully removing a package xyz that another installed package abc relies on effectively makes package abc unusable or at the very least somewhat broken.*
4. *Re-install the **libidn** package you just un-installed to keep everything else happy.*

Exercise-11: Determination of Device types

In this exercise you will determine if a device is a block device or a character device. To determine a device type

1. Log on to your machine as root

2. cd to the /dev directory and list all its contents.

```
root@localhost /dev# ls -l

brw-rw---- 1 root disk 3, 0 Apr 11 07:25 had
brw-rw---- 1 root disk 3, 1 Apr 11 07:25 hda1
brw-rw---- 1 root disk 3, 10 Apr 11 07:25 hda10
brw-rw---- 1 root disk 3, 11 Apr 11 07:25 hda11
brw-rw---- 1 root disk 3, 12 Apr 11 07:25 hda12
```

3. Take note of the entries in the far left of the first column.

4. To see the device type of the first IDE hard disk on the primary master. Type :

```
root@localhost dev# ls -l had
brw-rw---- 1 root disk 3, 0 Apr 11 07:25 had
```

The output of "ls -l hda" shows a letter "b" on the far left of the first column, which means that your hard disk is a block device
The major device number for /dev/hda is shown in the 5th column. It is 3.
The minor device number for /dev/hda is shown in the 6th column. It is 0.

5. To see the device type for the first partition on /dev/hda. Type :

```
root@localhost dev# ls l /dev/hda1
brw-rw---- 1 root disk 3, 1 Apr 11 07:25 /dev/hda1
```

From your output, what is the major number and minor number for /dev/hda1?

6. To see the device type for your mouse .Type :

```
root@localhost dev# ls -l /dev/input/mice
crw------- 1 root root 10, 1 Jun 18 16:03 /dev/input/mice
```

The output of "ls -l /dev/input/mice" shows a letter "c" on the far left of the first column, which means that your mouse is a character device.
From your output, what is the major number and minor number for /dev/input/mice?

7. You can also use the ***file*** command to find out the device type, the major and minor numbers of a device.

```
root@localhost pub# file /dev/input/mice
/dev/input/mice: character special (13/63)
```

Device files allow access to hardware. To demonstrate this using your mouse, you will direct the input of ***/dev/input/mice*** (i.e. the mouse device) to your screen using the cat ***/dev/input/mice*** command.

Moving the mouse after running this command will cause the mouse protocol commands to be written directly to your screen (it will look like garbage). This is an easy way to see if your mouse is working properly.

Exercise-12: Creation of Devices

Devices are usually located in the ***/dev*** directory, but they can also be created anywhere on the file system. You will be using the ***mknod*** command and the ***MAKEDEV*** script. The syntax for the mknod command is:

Usage: mknod **OPTION**... NAME TYPE **MAJOR MINOR**

To create a new CDROM device.

1. Create a new directory called mydevices in root's home directory ***/root/mydevices***.

```
      root@localhost root# mkdir mydevices
```

2. Change directory to *mydevices* directory

3. First find out the major and minor number for your current CDROM drive. It is attached to the file /dev/cdrom.

```
      root@localhost root# file /dev/cdrom
/dev/cdrom: symbolic link to /dev/hdc
```

The above output shows the CDROM drive is actually attached to the file /dev/hdc.

4. Find out the device type, major and minor numbers for the */dev/hdc/* file. What are they?

5. Create a new CDROM device file called *mycdrom* using the major and minor numbers your found out from above. Type:

```
      root@localhost mydevices# mknod -m 0600 mycdrom b 22 0
```

6. Find out the file type for the device you have just created: Type:

```
      root@localhost mydevices# file mycdrom
mycdrom: block special (22/0)
```

You have created a CDROM type of device.

Exercise-13: Mounting Devices

In order to be able to use some devices (mostly block type devices) in Linux, they need to be mounted. In this exercise you will mount the *mycdrom* device you created earlier.

To mount the mycdrom device

1. Type the ***mount*** command without any arguments, to list the devices currently mounted:

```
      [root@localhost root]# mount
```

```
/dev/hda1 on / type ext3 (rw)
none on /proc type proc (rw)
/dev/hda3 on /home type ext3 (rw)
none on /dev/shm type tmpfs (rw)
```

The above output shows that there is currently no folder mounted at */dev/hdc* or */dev/cdrom*

2. Make sure you are in the *~/mydevices* directory.

3. You will now mount the *mycdrom* device at the /mnt folder, using the iso9660 file system.(you should of course place an actual cdrom media in the drive tray). Type:

```
      [root@localhost mydevices]# mount
```

```
 t  iso9660   mycdrom   /mnt
```

mount: block device *mycdrom* is write-protected, mounting read-only

Exercise-14: Backup and Restore

tar stands for tape archive. You will first perform a full backup and then a differential backup of some specific files and folders on your system.

To Use *tar* to perform a full backup

1. Log on to the computer as root

2. Change your directory to the folder1 directory:

3. Create another directory named sample in the present working directory. And cd to the sample directory.

```
        root@localhost folder1# mkdir sample && cd sample
```

4. Create five files .

```
        root@localhost sample# touch file1 file2 file3 file4 file
```

5. Change back to the folder1 directory

```
        root@localhost sample# cd ..
```

6. Perform a full backup of the sample directory using tar.

```
        root@localhost folder1# tar -cvf sample.tar sample
```

```
sample/
sample/file1
sample/file2
sample/file3
sample/file4
sample/file5
```

7. Delete the sample directory and then restore it from the tar archive.

```
        root@localhost folder1# rm rf sample
```

6. Now restore the sample directory from the tar archive.

```
        root@localhost folder1# tar xvf sample.tar && cd sample
```

7. List the contents of the sample directory

Exercise-15: Printing Process-ID and detail using *ps*

Each process in a Linux system is identified by its unique *process ID*, sometimes referred to as *pid*. Process IDs are 16-bit numbers that are assigned sequentially by Linux as new processes are created.

```
( print-pid.c) Printing the Process ID

#include <stdio.h>
#include <unistd.h>
int main ()
{
printf ("The process ID is %d\n", (int) getpid ());
printf ("The parent process ID is %d\n", (int) getppid ());
return 0;
}
```

Compile the above program using gcc print-pid.c and see the result by executing ./a.out in the bash command prompt.

Usage of ps command:

```
% ps
PID TTY TIME CMD
21693 pts/8 00:00:00 bash
21694 pts/8 00:00:00 ps
```

Exercise-16: Usage of *system* call

This program invokes the ls command to display the contents of the root directory, as if you typed ls -l / into a shell.

Using the *system* Call

```
#include <stdlib.h>
int main ()
{
int return_value;
return_value = system ("ls -l /");
return return_value;
}
```

Exercise-17: Using *fork* to Duplicate a Program's Process

The following program is an example of using *fork* to duplicate a program's process. Note that the first block of the *if* statement is executed only in the parent process, while the *else* clause is executed in the child process.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main ()
{
pid_t child_pid;
printf ("the main program process ID is %d\n", (int) getpid ());
child_pid = fork ();
if (child_pid != 0) {
printf ("this is the parent process, with id %d\n", (int) getpid
());
printf ("the child's process ID is %d\n", (int) child_pid);
}
else
printf ("this is the child process, with id %d\n", (int) getpid ());
return 0;
}
```

Exercise-18: Using *fork and exec* to create Process

The program lists the contents of the root directory using the ls command. Unlike the previous example, though, it invokes the ls command directly, passing it the command-line arguments -l and / rather than invoking it through a shell.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
/* Spawn a child process running a new program. PROGRAM is the name
of the program to run; the path will be searched for this program.
ARG_LIST is a NULL-terminated list of character strings to be
passed as the program's argument list. Returns the process ID of
the spawned process. */
int spawn (char* program, char** arg_list)
{
pid_t child_pid;
/* Duplicate this process. */
child_pid = fork ();
if (child_pid != 0)
/* This is the parent process. */
return child_pid;
else {
/* Now execute PROGRAM, searching for it in the path. */
execvp (program, arg_list);
/* The execvp function returns only if an error occurs. */
fprintf (stderr, "an error occurred in execvp\n");
abort ();
}
}
int main ()
{
/* The argument list to pass to the "ls" command. */
char* arg_list[] = {
"ls", /* argv[0], the name of the program. */
"-l",
"/",
NULL /* The argument list must end with a NULL. */
};
/* Spawn a child process running the "ls" command. Ignore the
returned child process ID. */
spawn ("ls", arg_list);
printf ("done with main program\n");
return 0;
}
```

Exercise-19: Using *signal* function

This program skeleton uses a signal-handler function to count the number of times that the program receives SIGUSR1, one of the signals reserved for application use.

```
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
sig_atomic_t sigusr1_count = 0;
void handler (int signal_number)
{
++sigusr1_count;
}
int main ()
{
struct sigaction sa;
memset (&sa, 0, sizeof (sa));
sa.sa_handler = &handler;
sigaction (SIGUSR1, &sa, NULL);
/* Do some lengthy stuff here. */
/* ... */
printf ("SIGUSR1 was raised %d times\n", sigusr1_count);
return 0;
}
```

Exercise-20: Using *wait* system call

Here is the main function from the *fork* and *exec* example again. This time, the parent process calls wait to wait until the child process, in which the *ls* command executes, is finished.

```
int main ()
{
int child_status;
/* The argument list to pass to the "ls" command. */
char* arg_list[] = {
"ls", /* argv[0], the name of the program. */
"-l",
"/",
NULL /* The argument list must end with a NULL. */
};
/* Spawn a child process running the "ls" command. Ignore the
returned child process ID. */
spawn ("ls", arg_list);
/* Wait for the child process to complete. */
wait (&child_status);
if (WIFEXITED (child_status))
printf ("the child process exited normally, with exit code %d\n",
WEXITSTATUS (child_status));
else
printf ("the child process exited abnormally\n");
return 0;
}
```

Exercise-21: Creation of thread

The program creates a thread that prints x's continuously to standard error. After calling pthread_create, the main thread prints o's continuously to standard error.

```
#include <pthread.h>
#include <stdio.h>
/* Prints x's to stderr. The parameter is unused. Does not return.
*/
void* print_xs (void* unused)
{
while (1)
fputc ('x', stderr);
return NULL;
}
/* The main program. */
int main ()
{
pthread_t thread_id;
/* Create a new thread. The new thread will run the print_xs
function. */
pthread_create (&thread_id, NULL, &print_xs, NULL);
/* Print o's continuously to stderr. */
while (1)
fputc ('o', stderr);
return 0;
}
```

Compile and link this program using the following code:

```
% cc -o thread-create thread-create.c -lpthread
```

Exercise-22: Shared Memory concept

- The program illustrates the use of shared memory.

```
#include <stdio.h>
#include <sys/shm.h>
#include <sys/stat.h>
int main ()
{
int segment_id;
char* shared_memory;
struct shmid_ds shmbuffer;
int segment_size;
const int shared_segment_size = 0x6400;
/* Allocate a shared memory segment. */
segment_id = shmget (IPC_PRIVATE, shared_segment_size,
IPC_CREAT | IPC_EXCL | S_IRUSR | S_IWUSR);
/* Attach the shared memory segment. */
shared_memory = (char*) shmat (segment_id, 0, 0);
printf ("shared memory attached at address %p\n", shared_memory);
/* Determine the segment's size. */
shmctl (segment_id, IPC_STAT, &shmbuffer);
segment_size = shmbuffer.shm_segsz;
printf ("segment size: %d\n", segment_size);
/* Write a string to the shared memory segment. */
sprintf (shared_memory, "Hello, world.");
/* Detach the shared memory segment. */
shmdt (shared_memory);
/* Reattach the shared memory segment, at a different address. */
shared_memory = (char*) shmat (segment_id, (void*) 0x5000000, 0);
printf ("shared memory reattached at address %p\n", shared_memory);
/* Print out the string from shared memory. */
printf ("%s\n", shared_memory);
/* Detach the shared memory segment. */
shmdt (shared_memory);
/* Deallocate the shared memory segment. */
shmctl (segment_id, IPC_RMID, 0);
return 0;
}
```

Exercise-23: Allocation and de-allocation of binary Semaphores

This program presents functions to allocate and deallocate a binary semaphore.

```
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/types.h>
/* We must define union semun ourselves. */
union semun {
int val;
struct semid_ds *buf;
unsigned short int *array;
struct seminfo *__buf;
};
/* Obtain a binary semaphore's ID, allocating if necessary. */
int binary_semaphore_allocation (key_t key, int sem_flags)
{
return semget (key, 1, sem_flags);
}
/* Deallocate a binary semaphore. All users must have finished their
use. Returns -1 on failure. */
int binary_semaphore_deallocate (int semid)
{
union semun ignored_argument;
return semctl (semid, 1, IPC_RMID, ignored_argument);
}
```

Exercise-24: Initialization of binary semaphore

This program presents a function that initializes a binary semaphore.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
/* We must define union semun ourselves. */
union semun {
int val;
struct semid_ds *buf;
unsigned short int *array;
struct seminfo *__buf;
};
/* Initialize a binary semaphore with a value of 1. */
int binary_semaphore_initialize (int semid)
{
union semun argument;
unsigned short values[1];
values[0] = 1;
argument.array = values;
return semctl (semid, 0, SETALL, argument);
}
```

Exercise-25: Pipes for Inter-process communication

In this program, a fork spawns a child process. The child inherits the pipe file descriptors. The parent writes a string to the pipe, and the child reads it out. The sample program converts these file descriptors into FILE* streams using fdopen. Because we use streams rather than file descriptors, we can use the higher-level standard C library I/O functions such as printf and fgets.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
/* Write COUNT copies of MESSAGE to STREAM, pausing for a second
between each. */
void writer (const char* message, int count, FILE* stream)
{
for (; count > 0; --count) {
/* Write the message to the stream, and send it off immediately. */
fprintf (stream, "%s\n", message);
fflush (stream);
/* Snooze a while. */
sleep (1);
}
}
/* Read random strings from the stream as long as possible. */
void reader (FILE* stream)
{
char buffer[1024];
/* Read until we hit the end of the stream. fgets reads until
either a newline or the end-of-file. */
while (!feof (stream)
&& !ferror (stream)
&& fgets (buffer, sizeof (buffer), stream) != NULL)
fputs (buffer, stdout);
}
int main ()
{
int fds[2];
pid_t pid;
/* Create a pipe. File descriptors for the two ends of the pipe are
placed in fds. */
pipe (fds);
/* Fork a child process. */
pid = fork ();
if (pid == (pid_t) 0) {
FILE* stream;
/* This is the child process. Close our copy of the write end of
the file descriptor. */
close (fds[1]);
/* Convert the read file descriptor to a FILE object, and read
from it. */
stream = fdopen (fds[0], "r");
reader (stream);
close (fds[0]);
}
else {
/* This is the parent process. */
FILE* stream;
/* Close our copy of the read end of the file descriptor. */
close (fds[0]);
/* Convert the write file descriptor to a FILE object, and write
to it. */
stream = fdopen (fds[1], "w");
writer ("Hello, world.", 5, stream);
close (fds[1]);
}
return 0;
}
```

Exercise-26: Construction of virtual file system

To construct a virtual file system and mount it with a loopback device, follow these steps:

1. Create an empty file to hold the virtual file system. The size of the file will be the apparent size of the loopback device after it is mounted. One convenient way to construct a file of a fixed size is with the dd command. This command copies blocks (by default, 512 bytes each) from one file to another. The /dev/zero file is a convenient source of bytes to copy from.

To construct a 10MB file named disk-image, invoke the following:

```
     % dd if=/dev/zero of=/tmp/disk-image count=20480
20480+0 records in
20480+0 records out
     % ls -l /tmp/disk-image
-rw-rw---- 1 root r oot 1 0485760 Mar 8 01:56 /tmp/disk-image
```

2. The file that you've just created is filled with 0 bytes. Before you mount it, you must construct a file system. This sets up the various control structures needed to organize and store files, and builds the root directory. You can build any type of file system you like in your disk image. To construct an ext2 file system (the type most commonly used for Linux disks), use the mke2fs command. Because it's usually run on a block device, not an ordinary file, it asks for confirmation:

```
     % mke2fs -q /tmp/disk-image
mke2fs 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
disk-image is not a block special device.
Proceed anyway? (y,n) y
```

The -q option suppresses summary information about the newly created file system. Leave this option out if you're curious about it. Now disk-image contains a brand-new file system, as if it were a freshly initialized 10MB disk drive.

3. Mount the file system using a loopback device. To do this, use the mount command, specifying the disk image file as the mount device. Also specify loop=loopback-device as a mount option, using the -o option to mount to tell mount which loopback device to use. For example, to mount our disk-image file system, invoke these commands. Remember, only the superuser may use a loopback device. The first command creates a directory, /tmp/virtual-fs, to use as the mount point for the virtual file system.

```
     % mkdir /tmp/virtual-fs
     % mount -o loop=/dev/loop0 /tmp/disk-image /tmp/virtual-
     fs
```

Now your disk image is mounted as if it were an ordinary 10MB disk drive.

```
     % df -h /tmp/virtual-fs
Filesystem Size Used Avail Use% Mounted on
     /tmp/disk-image 9.7M 13k 9.2M 0% /tmp/virtual-fs
```

You can use it like any other disk:

```
      % cd /tmp/virtual-fs
      % echo 'Hello, world!' > test.txt
      % ls -l
total 13
drwxr-xr-x 2 root root 12288 Mar 8 02:00 lost+found
-rw-rw---- 1 root root 14 Mar 8 02:12 test.txt
      % cat test.txt
Hello, world!
```

When you have done, un-mount the virtual file system.

```
      % cd /tmp
      % umount /tmp/virtual-fs
```

You can delete disk-image if you like, or you can mount it later to access the files on the virtual file system. You can also copy it to another computer and mount it there—the whole file system that you created on it will be intact.

Exercise-27: Usage of *ioctl* system call

The *ioctl* system call is an all-purpose interface for controlling hardware devices. The first argument to *ioctl* is a file descriptor, which should be opened to the device that you want to control. The second argument is a request code that indicates the operation that you want to perform. Various request codes are available for different devices. Depending on the request code, there may be additional arguments supplying data to *ioctl.*

Many of the available requests codes for various devices are listed in the *ioctl_list* man page. Using *ioctl* generally requires a detailed understanding of the device driver corresponding to the hardware device that you want to control. Most of these are quite specialized and are beyond the scope of this book. However, we will present one example to give you a taste of how *ioctl* is used.

```
#include <fcntl.h>
#include <linux/cdrom.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int main (int argc, char* argv[])
{
/* Open a file descriptor to the device specified on the command
line. */
int fd = open (argv[1], O_RDONLY);
/* Eject the CD-ROM. */
ioctl (fd, CDROMEJECT);
/* Close the file descriptor. */
close (fd);
return 0;
}
```

Exercise-28: Usage of/proc file system

The various entries in the /proc file system are described extensively in the proc man page (Section 5).To view it, invoke this command:

```
% man 5 proc
```

Most of the entries in /proc provide information formatted to be readable by humans, but the formats are simple enough to be easily parsed. For example, /proc/cpuinfo contains information about the system CPU (or CPUs, for a multiprocessor machine). The output is a table of values, one per line, with a description of the value and a colon preceding each value. For example, the output might look like this:

```
% cat /proc/cpuinfo

processor            : 0
vendor_id            : GenuineIntel
cpu family           : 6
model                : 5
model name           : Pentium II (Deschutes)
stepping             : 2
cpu MHz              : 400.913520
cache size           : 512 KB
fdiv_bug             : no
hlt_bug              : no
sep_bug              : no
f00f_bug             : no
coma_bug             : no
fpu                  : yes
fpu_exception        : yes
cpuid level          : 2
wp                   : yes
flags                : fpu vme de pse tsc msr pae mce cx8 apic sep
mtrr pge mca cmov pat pse36 mmx fxsr
bogomips             : 399.77
```

28(a) Extract CPU Clock Speed from */proc/cpuinfo:*

A simple way to extract a value from this output is to read the file into a buffer and parse it in memory using sscanf. The following program includes the function get_cpu_clock_speed that reads from /proc/cpuinfo into memory and extracts the first CPU's clock speed.

```
#include <stdio.h>
#include <string.h>
/* Returns the clock speed of the system's CPU in MHz, as reported
by /proc/cpuinfo. On a multiprocessor machine, returns the speed of
the first CPU. On error returns zero. */
float get_cpu_clock_speed ()
{
FILE* fp;
char buffer[1024];
size_t bytes_read;
char* match;
float clock_speed;
/* Read the entire contents of /proc/cpuinfo into the buffer. */
fp = fopen ("/proc/cpuinfo", "r");
bytes_read = fread (buffer, 1, sizeof (buffer), fp);
fclose (fp);
/* Bail if read failed or if buffer isn't big enough. */
if (bytes_read == 0 || bytes_read == sizeof (buffer))
return 0;
/* NUL-terminate the text. */
buffer[bytes_read] = '\0';
/* Locate the line that starts with "cpu MHz". */
match = strstr (buffer, "cpu MHz");
if (match == NULL)
return 0;
/* Parse the line to extract the clock speed. */
sscanf (match, "cpu MHz : %f", &clock_speed);
return clock_speed;
}
int main ()
{
printf ("CPU clock speed: %4.0f MHz\n", get_cpu_clock_speed ());
return 0;
}
```

28(b) Print the argument list of a running process:

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
/* Prints the argument list, one argument to a line, of the process
given by PID. */
void print_process_arg_list (pid_t pid)
{
int fd;
char filename[24];
char arg_list[1024];
size_t length;
char* next_arg;
/* Generate the name of the cmdline file for the process. */
snprintf (filename, sizeof (filename), "/proc/%d/cmdline", (int)
pid);
/* Read the contents of the file. */
fd = open (filename, O_RDONLY);
length = read (fd, arg_list, sizeof (arg_list));
close (fd);
/* read does not NUL-terminate the buffer, so do it here. */
arg_list[length] = '\0';
/* Loop over arguments. Arguments are separated by NULs. */
next_arg = arg_list;
while (next_arg < arg_list + length) {
/* Print the argument. Each is NUL-terminated, so just treat it
like an ordinary string. */
printf ("%s\n", next_arg);
/* Advance to the next argument. Since each argument is
NUL-terminated, strlen counts the length of the next argument,
not the entire argument list. */
next_arg += strlen (next_arg) + 1;
}
}
int main (int argc, char* argv[])
{
pid_t pid = (pid_t) atoi (argv[1]);
print_process_arg_list (pid);
return 0;
}
```

Exercise-29: Obtaining System Information

29(a) *sysinfo*: Obtaining System Statistics

The *sysinfo* system call fills a structure with system statistics. The following program prints some statistics about the current system.

```
#include <linux/kernel.h>
#include <linux/sys.h>
#include <stdio.h>
#include <sys/sysinfo.h>
int main ()
{
/* Conversion constants. */
const long minute = 60;
const long hour = minute * 60;
const long day = hour * 24;
const double megabyte = 1024 * 1024;
/* Obtain system statistics. */
struct sysinfo si;
sysinfo (&si);
/* Summarize interesting values. */
printf ("system uptime : %ld days, %ld:%02ld:%02ld\n",
si.uptime / day, (si.uptime % day) / hour,
(si.uptime % hour) / minute, si.uptime % minute);
printf ("total RAM : %5.1f MB\n", si.totalram / megabyte);
printf ("free RAM : %5.1f MB\n", si.freeram / megabyte);
printf ("process count : %d\n", si.procs);
return 0;
}
```

29(b) *uname*: Obtaining System information

The *uname* system call fills a structure, with various system information, including the computers network name and domain name and the operating system version it's running.

```
#include <stdio.h>
#include <sys/utsname.h>
int main ()
{
struct utsname u;
uname (&u);
printf ("%s release %s (version %s) on %s\n", u.sysname, u.release,
u.version, u.machine);
return 0;

}
```

Exercise-30: Determine File Owner's Write Permission

The following program shows an example of using stat to obtain file permissions.

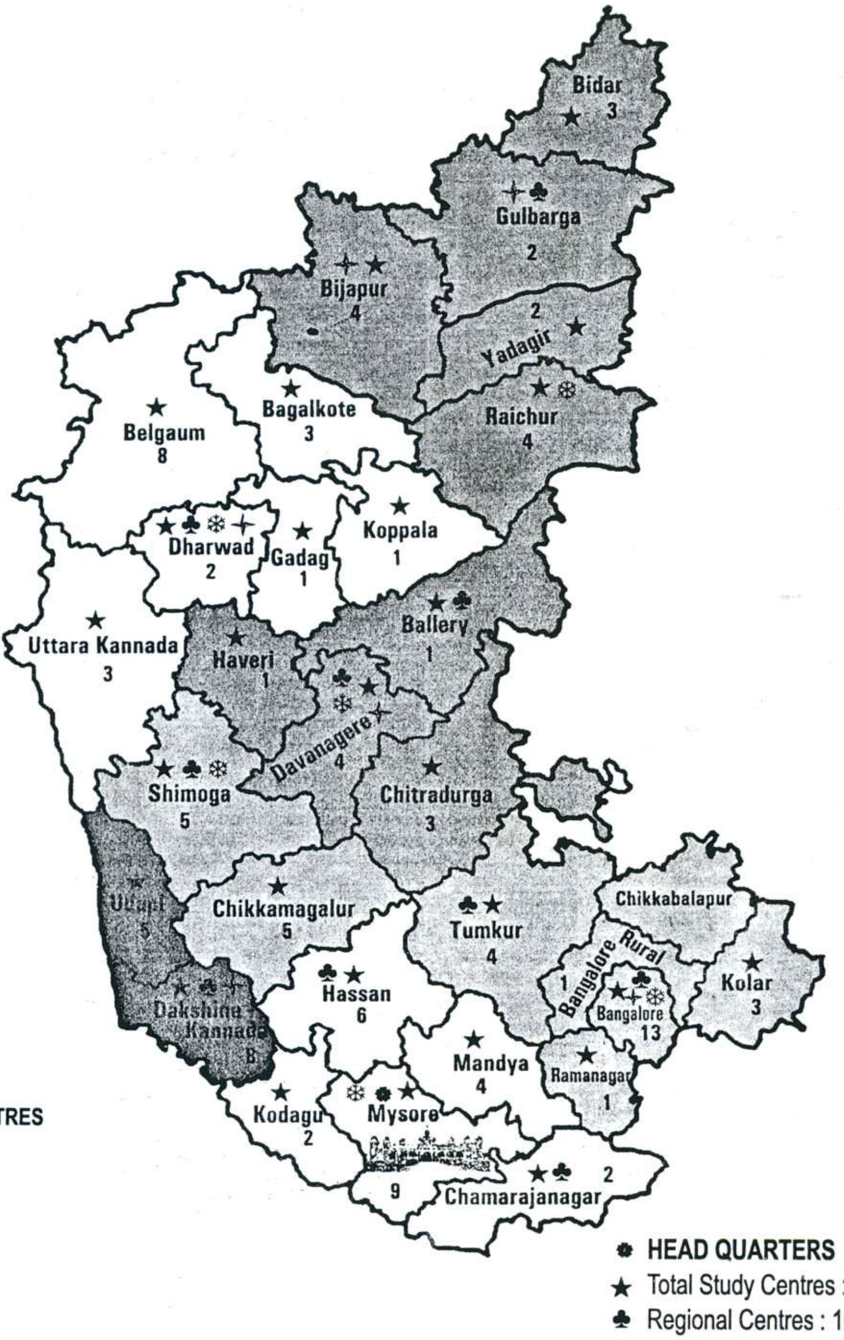
```
#include <stdio.h>
#include <sys/stat.h>
int main (int argc, char* argv[])
{
      const char* const filename = argv[1];
      struct stat buf;
      /* Get file information. */
      stat (filename, &buf);
      /* If the permissions are set such that the file's owner can
      write to it, print a message. */
      if (buf.st_mode & S_IWUSR)
            printf ("Owning user can write `%s'.\n", filename);
      return 0;
}
```

NOTES

ಆದೇಶ ಸಂಖ್ಯೆ : ಕರಾಮುವಿ/ಅಸಾವಿ/2–1464/2011–2012 ದಿನಾಂಕ : 8/8/2011
ಒಳಪುಟ : 60 GSM ಮಾಪ್ಲಿತೊ ಮತ್ತು ರಕ್ಷಾ ಪುಟ : 170 GSM 'ಆರ್ಟ್ ಕಾರ್ಡ್'
ಮುದ್ರಕರು : ಶ್ರೀ ಸುಬ್ರಮಣ್ಯೇಶ್ವರ ಬುಕ್ ಡಿಪೋ, ಬೆಂಗಳೂರು – 560002.

Karnataka State Open University

Manasagangotri Mysore - 570 006



♣ REGIONAL CENTRES
- Bangalore
- Davanagere
- Gulbarga
- Dharwad
- Shimoga
- Mangalore
- Tumkur
- Hassan
- Chamarajanagar
- Bellary

✹ HEAD QUARTERS
- ★ Total Study Centres : 111
- ♣ Regional Centres : 10
- ❄ B.Ed Study Centres : 10
- ✦ M.Ed Study Centres : 08

Feedback : improvement, revision and launching of new programmes
Unsystematic feedback
Systematic feedback
Multimedia learning material
Self-instructional print material
Course materials
Assignments
Programme guide
Audio & video cassettes at study centres
Broadcasts-radio, television & teleconferencing
Students affairs cell
Regional Director & academic staff
Optional
Academic and other support
Laboratory work/practicals
Extended contact programme
Tutor marked assignments
Continuous evaluation
Obligatory
Project
Examination results
Monitoring of assignment
Evaluation - methods and modes
STUDENT
INSTR ONAL SYSTEM