This lab is intended to allow you to get your Linux system up and running in preparation for the work ahead in CITX 2183.

Installing the System

1) Insert the **Linux** CD and reboot the computer.
2) Type "**text**" when prompted for the installation type.
3) Follow the prompts and select custom setup
4) Select "**Disk Druid**"
5) Highlight "**ADD**"
6) Create the following partitions. Give the mount point the partition name shown in the table.

Type	Partition	Size
Linux Native	/ boot	32 MB
Swap		256 MB
Linux Native	/	Use remaining space

7) Select "**OK**" and save the changes
8) At the Partitions to Format window, select **OK** ↵
9) At the **LILO** Window, select **OK**.
10) Highlight **install boot loader to MBR** ↵.
11) Accept at the Next screen.
12) Use **Boot / DHCP**.
13) Select **Generic 3 Button Mouse**.
14) Select a time zone.
15) Root PW = **SecretPassWord**
16) Create a user. PW = **SecretPassWord**
17) At the "Authentication Configuration" screen, accept the defaults.

18) At the "Package Group" select :

- X Windows
- Gnome
- KDE
- Multimedia
- Anything else you find interesting.
- Deselect everything Else.

19) Video Card = AT1 Mach64 - See End Notes

X Server = XF86_Mach64 - See End Notes

20) Highlight and Begin Installation

21) After package installation

22) Do **NOT** create a boot disk

23) Select NEC Multisync FE700 Monitor

24) Select 1 MB of video memory: See End Notes

25) No Clockchip Setting

26) Video Mode → 16 bit

800 x 600

27) At this point the installation wizard will test the X-Server settings:

a) You will be prompted as to weather you can see the GUI properly. If you can click **OK.**

b) You will be presented with a dialog box asking if you want Linux to start in graphic or command mode. Choose **text** mode.

28) The system will eject the installation CD and restart.

28) Type **startx** to start Gnome.

Note:

- *Xconfigurator may recognize your video card. In which case, just accept the default settings.*

Kernel Recompilation:

Modified version of the kernel-HOWTO document by:

Al Dev (alavoor@yahoo.com) located at:
http://www.linuxdoc.org/HOWTO/Kernel-HOWTO-2.html

You can find out about the present kernel by using the:

```
bash# uname -a
```

1. Start X-windows with 'startx' :

```
bash# startx
bash# cd /usr/src/linux
bash# make xconfig
```

NOTE: The **"make xconfig"** will bring up a user friendly GUI interface. Within 'make xconfig' you must do these to avoid problems -

- Select proper CPU type - Pentium 3, AMD K6, Cyrix, Pentium 4, Intel 386, DEC Alpha, PowerPC otherwise kernel will not boot!!
- Select SMP support - whether single CPU or multiple CPUs
- Filesystems - Select Windows95 Vfat, MSDOS, NTFS as part of kernel and not as loadable modules.
- Enable the Loadable kernel modules support! With this option you can load/unload the device drivers dynamically on running linux system on the fly. See these man pages

2. Save and Exit "make xconfig". All the options which you selected is now saved into configuration file at */usr/src/linux/.config* (dot config file). And now, do -

```
bash# make dep
bash# make clean
```

3. Now, give the make command:

```
        bash# cd /usr/src/linux
        bash# nohup make bzImage &
        bash# tail -f nohup.out     (.... to monitor the
progress)
This will put the kernel in /usr/src/linux/arch/i386/boot/bzImage
```

4. After bzImage is successful, copy the kernel image to /boot directory. You must copy the new kernel image to /boot directory, otherwise the new kernel **MAY NOT** boot.

```
bash# cp /usr/src/linux/arch/i386/boot/bzImage /boot/bzImage.myker.8oct2001
bash# man lilo
bash# man lilo.conf
```

5. And edit /etc/lilo.conf file and put these lines:

```
        image=/boot/bzImage.myker.26mar2001
        label=myker
        root=/dev/hda1
        read-only
You can check device name for 'root=' with the command -
        bash# df   /
```

6. You must re-run lilo even if entry 'myker' exists, everytime you create a new bzImage.

```
bash# lilo
bash# lilo -q
```

7. Reboot the machine and at lilo press tab key and type 'myker' If it boots then you did a good job! Otherwise at lilo select your old kernel, boot and re-try all over again. Your old kernel **is still INTACT and SAFE** at say */boot/vmlinuz-2.0.34-0.6*

Sample lilo.conf File

Given below is a sample /etc/lilo.conf file. You can have many kernel images on the same /boot system. On my machine I have something like:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=firewall

image=/boot/vmlinuz-2.2.14-5.0
        label=ker2214
        read-only
        root=/dev/hda9

image=/boot/vmlinuz-2.2.17-14
        label=ker2217
        read-only
        root=/dev/hda9

#image=/usr/src/linux/arch/i386/boot/bzImage
#       label=myker
#       root=/dev/hda7
#       read-only

image=/boot/bzImage.myker.11feb2001
        label=myker11feb
        root=/dev/hda9
        read-only

image=/boot/bzImage.myker.01jan2001
        label=myker01jan
        root=/dev/hda9
        read-only

image=/boot/bzImage.myker-firewall.16mar2001
        label=firewall
        root=/dev/hda9
        read-only
```

The following is modified from http://www.enteract.com/~lspitz/linux.html

Eliminating Services

1) Use the vi editor to modify the **/etc/inetd.conf** file. Use # at the beginning of the line to comment out everything but the **telnet** and **ftp** daemons.

```
bash>vi /etc/inetd.conf
```

After you exit **vi** type the following to verify that you have indeed eliminated all unwanted services.

```
bash>grep -v "^#" /etc/inetd.conf
```

2) Next are the .rc scripts, these scripts are used to determine what services are started by the init process. You will find these scripts in /etc/rc.d/rc3.d

```
bash>cd /etc/rc.d/rc3.d
```

To prevent a script from starting replace the capital S with a small s. That way you can easily start the script again just by replacing the small s with a capital S. You can use the mv command to rename the script. (See step 4)

3) To see how many services are running on your system before stopping the scripts from running type:

```
bash>ps aux | wc -l
```

4) Use the following list to help decide which services to shut off. Shut off laptop dns, and nfs support if you find it is running. Remember, you need to leave telnet and ftp and the X font server running

S05apmd (You only need this for laptops)
S10xntpd (Network time protocol)
S11portmap (Required if you have any rpc services, such as NIS or NFS)
S15sound (Saves sound cared settings)
S15netfs (This is the nfs client, used for mounting filesystems from a nfs server)
S20rstatd (Try to avoid running any **r** services, they provide too much information to remote users)
S20rusersd
S20rwhod
S20rwalld
S20bootparamd (Used for diskless clients, you probably don't need this vulnerable service**)**
S25squid (Proxy server)
S34yppasswdd (Required if you are a NIS server, this is an extremely vulnerable service)
S35ypserv (Required if you are a NIS server, this is an extremely vulnerable service)
S35dhcpd (Starts dhcp server daemon)
S40atd (Used for the at service, similar to cron, by not required by the system)
S45pcmcia (You only need this script for laptops)
S50snmpd (SNMP daemon, can give remote users detailed information about your system)
S55named (DNS server. If you are setting up DNS, upgrade to the latest version of BIND, http://www.isc.org/bind.html)
S55routed (RIP, don't run this unless you REALLY need it)
S60lpd (Printing services)
S60mars-nwe (Netware file and print server)
S60nfs (Use for NFS server, do not run unless you absolutely have to).
S72amd (AutoMount daemon, used to mount remote file systems)
S75gated (used to run other routing protocols, such as OSPF)
S80sendmail (You can still send email if you turn this script off, you just will not be able to receive or relay)
S85httpd (Apache webserver, I recommend you upgrade to the latest version, http://www.apache.org)
S87ypbind (Required if you are a NIS client)
S90xfs (X font server)
S95innd (News server)
S99linuxconf (Used to remotely configure Linux systems via browser, every black-hat's dream :)

5) To see how many services are running after stopping the scripts from running type:

```
bash>ps aux | wc -l
```

6) Confirm which are left running by executing the following command:

```
bash>netstat -na --ip | less
```

Logging and Tweaking

1) Use vi to add the following line to the /etc/inetd.conf file:

```
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -L -i -o
```

From the command line type:

bash> vi /etc/inetd.conf

```
Insert the above line into the file using the same format as the other
lines in the file.

2) To make sure your system is using shadow passwords type the
following as root:
```

```
bash>pwconv
```

3) Remove most of the default system accounts in the **/etc/passwd** file. Use **vi** to remove all but the following accounts:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
mail:x:8:12:mail:/var/spool/mail:
uucp:x:10:14:uucp:/var/spool/uucp:
nobody:x:99:99:Nobody:/:
```

4) Any account listed in the **/etc/ftpusers** file cannot ftp to the system. Check that **root** can not ftp to the system by typing:

bash>less /etc/ftpusers

If root is not listed edit the file using **vi** so as to include root and/or the following:

```
root
bin
daemon
adm
lp
mail
```

```
uucp
nobody
```

Use the **less** command to check the **/etc/securetty** file. This file lists what ttys root can connect to and you should only allow **tty1**, **tty2**, etc in this file.

Use **vi** to create the file **/etc/issue** containing the following text:

```
#
#
#  WARNING:  You must have specific authorization to access
#            this machine.  Unauthorized users will be logged,
#            monitored, and then shot on sight!
#
#
```

Secure Shell:

From: http://www.scrounge.org/linux/ssh.html

Using SSH to connect securely

What is SSH and why should I use it?
(From the SSH FAQ.)

The short answer: because evil crackers can (and do) use "sniffing" techniques to intercept your login information, including passwords. SSH is securely encrypted, so the crackers can't do this. You use SSH instead of telnet (and, hopefully, FTP.)

Perform the following steps:

Configuring your Linux machine to use SSH

For Red Hat or Mandrake, you need to download **ssh-1.2.27-1us.i386.rpm, ssh-extras-1.2.27-1us.i386.rpm, ssh-clients-1.2.27-1us.i386.rpm, and ssh-server-1.2.27-1us.i386.rpm**. (Or whatever are the newest versions.)

They are available from ftp://utopia.hacktic.nl/pub/crypto/linux/redhat/i386/

Also note that SSH has both SSH1 and SSH2 versions. We are sticking with the SSH1 version, because our Windows client only supports SSH1 and also because SSH1 has less restrictive licensing. SSH1 is indicated by files that are numbered like 1.2.27. SSH2, like

2.0.13. If you are using a SSH client that supports SSH2, then feel free to use a SSH2 version. To install them, just FTP them to **/usr/src/ssh** and (as root) follow these steps:

```
rpm -Uvh ssh*.rpm
```

Then start the SSH server by typing `/etc/rc.d/init.d/sshd start`

Configuring your Windows machine to use SSH

- Install the Tera Term telnet client
- Then install the TTSSH extensions to add SSH to Tera Term.
    - Download the latest version of TTSSH.
    - Follow the installation instructions in the TTSSH page.
    - And maybe glance at the Documentation for TTSSH.

If you have installed SSH on your Linux machine(s) and Tera Term and TTSSH on your Windows machine, you should be able to start TTSSH on your Windows machine, choose SSH as a protocol and log into your Linux machine. Answer "yes" when it asks you to add this machine to your "ssh_known_hosts" file. Use the same user name you normally use to log in. Use your normal password for the TTSSH "passphrase."

Make sure that TTSSH uses port 22 for SSH connections. 22 is for SSH, 23 is for regular telnet. Port 23 won't work for SSH.

When you connect it should act just like a normal telnet session *except* that everything (especially including your logon password) is very securely encrypted, so that nobody can use sniffing techniques to intercept anything from your session.

Tightening down your Linux machine

If SSH works, then you should disable the telnet service on your Linux machine by editing `/etc/inetd.conf`, commenting out the "telnet" line, saving `/etc/inetd.conf`, and then typing `/etc/rc.d/init.d/inet restart` to re-read `/etc/inetd.conf`.

You also might want to comment out "linuxconf" because the web administered version of linuxconf also sends everything, including your root password, unencrypted. In general, try to disable all inet services, unless you absolutely need to use them. *Now* try to telnet (regular old telnet, not TTSSH) into your Linux machine. You should get a "connection refused" error. Good. So will all the crackers! But you can connect using the secure and encrypted SSH protocol. This is a good thing.