Linux Commands – Red Hat Specific

Windows	Linux

General Commands	
dir **dir /ad** **attrib –h** sorting	**ls** **ls** *filename* or *file** **ls** *directory* or *dir** **ls –l** - Directory listing with long filenames, owner, and permissions **ls –ld** - Show only the directories matching the search **ls – R** - Show subdirectories also (just like **dir /s**) **ls –al** **ls –Sharl –** sort by size, all files, including hidden **ls –lart** – sort by date, reverse order (newest @ the bottom of list) **-S** - sorts by size, largest first **-r** - reverses any sort command, i.e. **ls –rS** sorts smallest first **-t** - sorts by modification time, recent first **-u** - sort by last accessed time **-x** - sort by file extension Remember – Linux is case sensitive, so if you're looking for a file beginning with Vol, type **ls Vol*** Colors: Blue = directory Green = script Grey = file
dir filename* /s | more	**find –depth –iname "***filename***" | less** - this will find anything starting with filename, with the –iname switch allowing it to be case insensitive, and will pipe the results to the less screen so you can page through the results using [Page Up] and [Page Down] keys. Pretty cool! Press Q to quit
attrib	**chmod**
md	**mkdir**
rd	**rmdir**
del	**rm –i** (without the –i, rm will not ask for confirmation)
deltree	**rm –rf ***

copy	**cp –i** (without the –i, cp will not prompt for destructive actions like file replacement - if the file exists, it is overwritten)
	
	How to copy all files from a CD and then change the files to read/write
	
	`mount /dev/cdrom /mnt/cdrom`
	`mkdir /root/snortinstall`
	`cp –r -v /mnt/cdrom/* /root/snortinstall`
	`cd /root/snortinstall`
	`chmod –R +wr /root/snortinstall/*`
	`umount /mnt/cdrom`

copy con	**cp /dev/tty** *filename* then CTRL+D to quit
edit	**vi** or **joe**
type	**cat**
move	**mv**
ren	**mv**
cls	**clear**
help	**man** *command*
fdisk	**fdisk** or **cfdisk**
diskcopy	To make a copy of a floppy diskette, insert the source and execute: **dd if=/dev/fd0 of=/tmp/floppy bs=1024 count=1440** …then insert the destination and execute: **dd if=/tmp/floppy of=/dev/floppy bs=1024 count=1440**
ver	**cat /etc/redhat-release** – shows Red Hat version **ls /lib/modules** - this will show the kernel version of all kernels installed (you could have more than one to boot to) **uname –sr** – this will show your current running version **uname –a** – shows all the versioning information, except the distribution (Red Hat)
whoami	**finger** **finger –l** (most information) **w** **users**
set	**set** – displays system environment. Pipe set to the less command in order to read all of it. **set | less**
Help	**man** **info** ***command*—help** HOWTOs – www.linuxdoc.org, howto.tucows.com
Hard drive info	**hdparm –i /dev/hda**
Reboot	**shutdown –r now** or **shutdown –rf now** (stands for right fu**ing now) **reboot** (reboot is not clean, do not use often)
Shutdown	**shutdown –h now** or or **halt** (halt is not clean, can be bad on disks)
Working directory	**pwd** - shows the full path of your current directory
Compile binary	**gcc –o <*binary name*> <*.c file*>**
Make Executable out of script	**chmod +x *filename***
Running Processes, Uptime, etc.	**top**
Find a specific process	**ps –ax | grep *process_name*** ***i.e. ps –ax | grep snort***
Mem	**free –t**

Login as another user	**su *username***
Login as another user AND get their path – awesome!	**su – *username***
See bootup messages after booting	**dmesg**
Display a calendar	**cal *month year***
Startup file	**/etc/rc.d/rc.local**
Disk Space	**df –h**
Directory Usage	**du –h**
Install RPM	**rpm –ivh *rpm_name.rpm*** This installs a given rpm with install, verbose, and hash marks To list all installed rpms: **rpm –qa | sort | more** This will list all installed rpms, sort alphabetically, and display them one screen at a time. You could redirect it to a file to peruse at your convenience or for auditing purposes.
GRUB boot files	**/boot/grub/grub.conf** **/boot/grub/menu.lst**
Path	To see your path, type **echo $PATH**
Net Send	You can send messages to other users that are connected to the same linux box, such as on the console or SSH'd to the box. First, find out what TTY the user is connected to by simply typing `w` Now, issue the `write` command to the username and the TTY they are on, like this: `write root tty1` Now you can simply type away, hitting the [ENTER] key to send the line. You can keep on typing and sending messages until you escape out by using `[CTRL]+c.`

mail	Edit the /etc/mail/submit.cf file to put in your SMTP server:
	
	`vim /etc/mail/submit.cf`
	
	Search for the following line:
	`# "Smart" relay host (may be null)`
	`DS`
	
	Change **DS** to `DSmail.corpone.org`

MD5 CheckSum	`md5sum filename`

Directory Size	**How to find - Size of a directory & Free disk space** This article explains 2 simple commands that most people want to know when they start using Linux. They are finding the size of a directory and finding the amount of free disk space that exists on your machine. The command you would use to find the directory size is ' *du* '. And to find the free disk space you could use ' *df* '. All the information present in this article is available in the man pages for du and df. In case you get bored reading the man pages and you want to get your work done quickly, then this article is for you. - **'du' - Finding the size of a directory** **$ du** Typing the above at the prompt gives you a list of directories that exist in the current directory along with their sizes. The last line of the output gives you the total size of the current directory including its subdirectories. The size given includes the sizes of the files and the directories that exist in the current directory as well as all of its subdirectories. Note that by default the sizes given are in kilobytes. **$ du /home/david** The above command would give you the directory size of the directory /home/david **$ du -h** This command gives you a better output than the default one. The option '-h' stands for *human readable format*. So the sizes of the files / directories are this time suffixed with a 'k' if its kilobytes and 'M' if its Megabytes and 'G' if its Gigabytes. **$ du -ah** This command would display in its output, not only the directories but also all the files that are present in the current directory. Note that 'du' always counts all files and directories while giving the final size in the last line. But the '-a' *displays* the filenames along with the directory names in the output. '-h' is once again human readable format. **$ du -c** This gives you a *grand total* as the last line of the output. So if your directory occupies 30MB the last 2 lines of the output would be 30M . 30M total

The first line would be the default last line of the 'du' output indicating the total size of the directory and another line displaying the same size, followed by the string '*total*'. This is helpful in case you this command along with the grep command to only display the final total size of a directory as shown below.

$ du -ch | grep total
This would have only one line in its output that displays the total size of the current directory including all the subdirectories.

Note : In case you are not familiar with pipes (which makes the above command possible) refer to Article No. 24 . Also grep is one of the most important commands in Unix. Refer to Article No. 25 to know more about grep.

$ du -s
This displays a summary of the directory size. It is the simplest way to know the total size of the current directory.

$ du -S
This would display the size of the current directory excluding the size of the subdirectories that exist within that directory. So it basically shows you the total size of *all the files* that exist in the current directory.

$ du --exculde=mp3
The above command would display the size of the current directory along with all its subdirectories, ***but*** it would exclude all the files having the given pattern present in their filenames. Thus in the above case if there happens to be any mp3 files within the current directory or any of its subdirectories, their size *would not be included* while calculating the total directory size.

-

'df' - finding the disk free space / disk usage

$ df
Typing the above, outputs a table consisting of 6 columns. All the columns are very easy to understand. Remember that the 'Size', 'Used' and 'Avail' columns use kilobytes as the unit. The 'Use%' column shows the usage as a percentage which is also very useful.

$ df -h
Displays the same output as the previous command but the '-h' indicates *human readable format*. Hence instead of kilobytes as the unit the output would have 'M' for Megabytes and 'G' for Gigabytes.

Most of the users don't use the other parameters that can be passed to 'df'. So I shall not be discussing them.

I shall in turn show you an example that I use on my machine. I have

	actually stored this as a script named *'usage*' since I use it often. **Example :** I have my Linux installed on /dev/hda1 and I have mounted my Windows partitions as well (by default every time Linux boots). So 'df' by default shows me the disk usage of my Linux as well as Windows partitions. And I am only interested in the disk usage of the Linux partitions. This is what I use : **$ df -h | grep /dev/hda1 | cut -c 41-43** This command displays the following on my machine **45%** Basically this command makes 'df' display the disk usages of all the partitions and then extracts the lines with /dev/hda1 since I am only interested in that. Then it cuts the characters from the 41st to the 43rd column since they are the columns that display the usage in % , which is what I want. **Note :** In case you are not familiar with pipes (which is used in the above command) then refer to Article No. 24 . 'cut' is another tool available in Unix. The above usage of cut gets the the characters that are present in the specified columns. If you are interested in knowing how to mount you Windows partitions under Linux, please refer to Article No. 3 . There are a few more options that can be used with 'du' and 'df' . You could find them in the man pages.

Boot Loader Information	

Grub	Grub is an acronym for **Gr**and **U**nified **B**ootloader.
	
	An excellent overview of Grub and how to use it is located at http://sdb.suse.de/en/sdb/html/fhassel_grub_overview.html
	
	For now, here is some general information on the configuration file, grub.conf:
	
	Edit the file /etc/grub.conf to add/remove menu items, and adjust the default. The default is indicated by the line **default x**, where x is the number of the **title** you want to boot, beginning with 0 (zero). Get it? Didn't think so.
	
	How about this explanation then?
	
	The following example shows the structure of the menu file **/etc/grub.conf**. In this example, /dev/hda5 is the Linux boot partition, /dev/hda7 is the root partition, and /dev/hda1 contains a Windows operating system.
	
	`gfxmenu (hd0,4)/message`
	`color white/green black/light-gray`
	`default 0`
	`timeout 8`
	
	`title linux`
	`kernel (hd0,4)/vmlinuz root=/dev/hda7 vga=791`
	`initrd (hd0,4)/initrd`
	`title windows`
	`root (hd0,0)`
	`makeactive`
	`chainloader +1`
	`title floppy`
	`root (fd0)`
	`chainloader +1`
	`title failsafe`
	`kernel (hd0,4)/vmlinuz.shipped root=/dev/hda7 ide=nodma apm=off acpi=off vga=normal nosmp maxcpus=0 3`
	`initrd (hd0,4)/initrd.shipped`
	
	The entries have the following meaning:
	
	• As you can easily guess, the first two lines cover the configuration of the splash menu: the background image is located in /dev/hda5 and has the name "`message`". Foreground: white, background: green, selection: black, background of the selection: light gray.
	• The entry "`default 0`" in the third line indicates that the first menu entry ("`title linux`") is the default selection for booting.
	• Line 4: The timeout is 8 seconds.

	Regarding the entries of the operating systems that can be booted: • The first entry ("`title linux`") boots **SuSE Linux**. • The Linux kernel is located in the first logical partition of the first hard disk (`hd0,4`) (the boot partition in this example); the file name is vmlinuz. Kernel parameters (such as the specification of the root partition, vga, etc.) are appended directly. **Attention**: The root partition must be specified as a Linux device name, since it is sent to the Linux kernel as a parameter. • Information on the position of the **initrd**: The initrd is also located in the first logical partition on the first hard disk. • The next section starts **Windows** from the first partition of the hard disk (`hd0,0`). To be on the safe side, the option "makeactive" is set in the following line, as Windows can only be started from a visible partition that is set active. The entry "`chainloader +1`" causes the first sector of the indicated partition to be read and executed. • The subsequent section can be used to start an operating system from a **floppy disk** without performing any changes in the BIOS. • The final section starts Linux in the **failsafe mode**. See the website for a better explanation if this didn't jog your memory.
GUI or Text Boot	To change the boot mode between GUI and Text mode, you need to set the default runlevel in the initab file: - **vi /etc/inittab** - Find the runlevel section and the line that reads: **id:*X*:initdefault** (where X = 3 or 5) - Change the line for what you want as follows: **id:3:initdefault** = text mode **id:5:initdefault** = GUI mode
Exit GUI to Text Mode	Uhoh – your /etc/inittab is set to boot into GUI mode automatically, but your screen resolution doesn't support it. How do you exit the GUI when you can't see anything? Well, good old **[CTRL] + [ALT] + [Backspace]** doesn't do the trick because it will exit GUI, but immediately go right back into the GUI. So what's a geek to do? To exit GUI, use **[CTRL] + [ALT] + [F1]**. This will exit GUI and put you at the console (command line interface).

Boot Single-User Mode	This is kind of like safe-mode in Windows – it skips a lot of startup files and services: 1. Power on the box 2. At the boot screen for Grub, select the version you want to boot and type `e` for edit 3. Select the line that starts with `kernel` and type `e` for edit 4. Go to the end of the line and type `single` 5. Hit `[ENTER]` 6. Now type `b` for boot

tcpdump	Dump traffic with a specific host IP address to a file: **`tcpdump –i interface –w dump_file_name host IP_address`** example: **`tcpdump –i eth1 –w dump host 208.159.105.98`** Then read the file that was created: **`tcpdump –r dump`**
tcpflow	Download from http://dries.ulyssis.org/rpm/packages/tcpflow/info.html tcpflow will capture and save to individual files actual data streams Examples: **`tcpflow –s`** **`tcpflow –s port http`** **`tcpflow –s port http or https or smtp or 3128`**

Remove Three-Finger Salute	
Three-Finger Salute	Edit /etc/inittab. Find the line: **ca::ctrlaltdel:/sbin/shutdown -t3 -r now** Remark it out by placing a pound sign (#) in front of it. Next time you boot, CTRL-ALT-DEL should do nothing! Or, point the line to something else, like script that will clear the screen and echo a remark to the user. For example: **ca::ctrlaltdel:/salute.sh** vi salute.sh clear echo echo echo "What are you doing, you IDIOT?!?!" echo echo

VMware Workstation on SUSE 9.1	

Install VMware Workstation v4.5.2 on SUSE Linux 9.1	Download the VMware-<xxx>.rpm file from www.vwmare.com
	
	`su -`
	`rpm -Uvh VMware-<xxx>.rpm`
	`cd /usr/src/linux`
	`make cloneconfig`
	`make prepare`
	`vmware-config.pl`

Launch VMware	`/usr/bin/vmware`
How to Uninstall VMware	rpm –e VmwareWorkstation

Mounting and Zipping	
Mount Floppy	FAT floppy's = **mount –t vfat /dev/fd0 /mnt/floppy** Linux floppy's = **mount /dev/fd0 /mnt/floppy** or mount through linuxconf: type **linuxconf**, navigate to **File Systems**, then **Access local drive**, select the floppy drive, **Enter**, tab to **Mount** and **enter**
Mount CD-ROM	**mount /dev/cdrom /mnt/cdrom**
Read a floppy or CD	If it's a DOS FAT disk, Red Hat includes a set of programs called the mtools. Using mtools, you can access the floppy drive with the following commands, just like their DOS counterparts: **mdir** **mcopy** **mdel** **mdeltree** **mformat** **mmove** **mren** **mtype** If it's not a DOS FAT floppy, you must mount the floppy before reading it. Note also that if you put in a different floppy after mounting the drive, you must unmount the floppy and then mount it again for Linux to recognize the new floppy.
Mount Windows share using "mount"	First, you need to create a mount point (directory). I usually do this in the /mnt directory Example: **mkdir /mnt/share** Then, mount the share: **mount –t smbfs –o username='*domain**username*' //*server/share* /mnt/*directory*** If you need to pass the password in the command line as well, add it as follows (no spaces between username and password, only a comma): **mount –t smbfs –o username='*domain**username*',password=*password* //*server/share* /mnt/*directory*** To unmount, simply type: **umount /mnt/*directory*** Example: **mount –t smbfs –o username='corpone\bwestbro' //ohcont04/data /mnt/data**

Mount Windows share using "smbmount"	First, you need to create a mount point (directory). I usually do this in the /mount directory Example: **mkdir /mnt/share** Then, mount the share: **smbmount *//server/share* /mnt*/directory* –o username=*username*, workgroup=*domain*** Example: **smbmount //ohcont04/data /mnt/data –o username=bwestbro, workgroup=corpone**
unmount	**umount /mnt/floppy**
smblient	**smbclient //*server/share* –W *domain* –U *username%password* -c '*put /root/xxxx xxxxx*'** **Example:** **smbclient //ohcont04/bacup –W corpone –U bwestbro%*password* –c 'put /root/test.txt test.txt'**

Mount NFS Share	`mount server:/share /local_dir`
	
	For instance, to mount the SysOps NFS directory on 192.168.10.11 to your local mount directory of /mnt/nfs (which needs to be created already on your local machine), the command would be:
	
	`mount 192.168.10.11:/SysOps /mnt/nfs`

Secure NFS with IPTables	To secure NFS using IPTables on Red Hat, you must first static the ports that NFS uses. By default, NFS opens dynamic ports for the inbound service requests. To static the ports you have to create a file name nfs in the /etc/syconfig directory, and add the specific ports. In our example, which you can use real-world, we'll static the ports to 4000, 4001 and 4002.
	
	`vim /etc/sysconfig/nfs`
	
	Now add the following lines to this new file:
	`STATD_PORT=4000`
	`MOUNTD_PORT=4001`
	`LOCKD_TCPPORT=4002`
	`LOCKD_UDPPORT=4002`
	
	Save and close the file.
	
	Now restart nfs:
	`service nfs restart`

You can verify what ports RPC (e.g. NFS) is listening on now that you reconfigured it by issuing the command:

```
rpcinfo -p
```

The following information will be shown:

```
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100011    1   udp    660  rquotad
100011    2   udp    660  rquotad
100011    1   tcp    663  rquotad
100011    2   tcp    663  rquotad
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
100003    2   tcp   2049  nfs
100003    3   tcp   2049  nfs
100021    1   udp   4002  nlockmgr
100021    3   udp   4002  nlockmgr
100021    4   udp   4002  nlockmgr
100021    1   tcp   4002  nlockmgr
100021    3   tcp   4002  nlockmgr
100021    4   tcp   4002  nlockmgr
100005    1   udp   4001  mountd
100005    1   tcp   4001  mountd
100005    2   udp   4001  mountd
100005    2   tcp   4001  mountd
100005    3   udp   4001  mountd
100005    3   tcp   4001  mountd
```

Finally, you need to allow portmapper, nfs and your new nfs ports through IPTables. If you're setting up iptables as I usually do and are just using the default INPUT chain, you would configure your iptables something like this:

First, delete your last DROP line:

```
iptables –D INPUT –j DROP
```

Next, add all the necessary TCP and UDP ports for NFS:

```
iptables –A INPUT –p tcp --dport 111 –j ACCEPT
iptables –A INPUT –p udp --dport 111 –j ACCEPT
iptables –A INPUT –p tcp --dport 2049 –j ACCEPT
iptables –A INPUT –p udp --dport 2049 –j ACCEPT
iptables –A INPUT –p tcp --dport 4000:4002 –j ACCEPT
iptables –A INPUT –p udp --dport 4000:4002 –j ACCEPT
```

Add your final DROP line back in:

```
iptables –A INPUT –j DROP
```

Finally, save and restart iptables:

```
service iptables save
service iptables restart
```

format a: (floppy) **as Linux -->** **as DOS -->**	**<u>Linux</u>** **fdformat /dev/fd0** then **mkfs –t ext2 /dev/fd0** **<u>DOS</u>** **fdformat /dev/fd0** then **mkfs –t msdos /dev/fd0**
Make a boot disk	1. Check /etc/lilo.conf (or /etc/grub.conf) and right down the version of the image being loaded at boot. The line will look as follows: image = /boot/vmlinuz-*x.x.x-x.x* 2. Login as root 3. Insert a blank floppy – type **mkbootdisk –device /dev/fd0 *x.x.x-x.x*** 4. Reboot with floppy and see if works
Zipped files	.tar – usually means that the file is not compressed .tar.gz or .tgz – usually means that the file was first combined into a tar file, then compressed into a zip format with **gunzip**. .tar.bz – usually means that the file was again first combined into a tar file, then compressed with **bzip2** Extracting: for a simple .tar file called myfile.tar, type the following – **tar -xvf myfile.tar** For a file that's both tar and zipped, such as myfile.tar.gz or myfile.tgz, type the following – **tar –xvzf myfile.tar.gz** To tar files, do this: **tar –cvf *tar_filename [files/directory]*** To tar and zip files, do this: **tar –zcvf *tar_filename [files/directory]***

Common Linux Device Names	
A: floppy	**/dev/fd0** or **/dev/floppy**
B: floppy	**/dev/fd1**
Master drive on primary IDE channel	**/dev/hda**
First partition on master drive on primary partition	**/dev/hda1**
Third partition on master drive on primary partition	**/dev/hda3**
Slave drive on primary IDE channel	**/dev/hdb**
Master drive on secondary IDE channel	**/dev/hdc**
Slave drive on secondary IDS channel	**/dev/hdd**
SCSI target ID 0	**/dev/sd0**
SCSI target ID 1	**/dev/sd1**
Second partition on the hard disk at SCSI target ID 0	**/dev/sd02**
SCSI tape device	**/dev/st0**
CD-ROM	**/dev/cdrom** or the IDE/SCSI designation if SCSI (**/dev/hdb, /dev/sd5,** etc.**)**
Keyboard	**/dev/tty**
Modem	**/dev/modem**
Sound system	**/dev/sound**

Interesting trick with devices - Since the keyboard is a device with a filename (/dev/tty) you can create a text file from what you type at the keyboard...very similar to **copy con**.

IP Configuration	
ipconfig	**ifconfig**
IP configuration utilities	**ifconfig** **netconfig** (preferred) **linuxconf** (catch all) **netconf** (not preferred, use CTRL+X for pulldowns)
route print	**route –n** (the –n is no resolution and is faster)
route add	**route add –net *network* netmask *subnet_mask* gw *gateway_address***
route add –p (permanent) (option 1)	Add the following information to the **/etc/sysconfig/static-files** file (simply make the file if it does not already exist): - **vi /etc/sysconfig/static-files** - **i** (for insert) - **any net x.x.x.x netmask y.y.y.y gw z.z.z.z** - **:wq** Now reboot to see if the route stuck
route add –p (permanent) (option 2)	You could also simply add a line to the **rc.local** (e.g. autoexec.bat) file that runs at startup: - **vi /etc/rc.d/rc.local** - **i** (for insert) - **route add –net *network* netmask *subnet_mask* gw *gateway_address*** - **:wq**
route delete	**route del –net *network* netmask *subnet_mask***
route add default gateway	**route add default gw *gateway_address***
route add default gateway (permanent)	If you need the default route to survive a reboot (non-DHCP for instance), do it in the **/etc/sysconfig/network** file by adding the following line: **GATEWAY=x.x.x.x**
DNS servers	**/etc/resolv.conf**
Hosts file	**/etc/hosts** - IP address, at least one space, then hostname - You can have more than one hostname on a line for a single IP address, just separate the hostnames by at least one space
hostname	**/etc/sysconfig/network**
Change hostname	- Change name in **/etc/hosts** - Change name in **/etc/sysconfig/network** **HOSTNAME="xxxxx"** - For **Mandrake**, also add the following line to the network file **DOMAINNAME="xxxxx.xxx"**

Manually setup static IP and routing by hand **TEMPORARY SETUP** This will NOT last through a reboot	Example: NIC Information: IP address = 192.168.0.10 Subnet Mask = 255.255.255.0 Gateway = 192.168.0.1 Hostname = mypc.acme.com DNS = 192.168.0.77 1. Configure the NIC and IP address with the ifconfig command: **ifconfig eth0 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.255** (If you're dealing with a PCMCIA NIC, you be able to get a report of the IRQ and IO values by typing **cardctl config**) 2. Create the local route: **route add –net 192.168.0.0 netmask 255.255.255.0 dev eth0** 3. Create the local gateway: **route add default gw 192.168.0.1**
Static IP Address **PERMANENT SETUP** This will last through a reboot	Use the networking file for the specific interface. For eth0, use **/etc/sysconfig/network-scripts/ifcfg-eth0** For DHCP the file looks like: DEVICE=eth0 BOOTPROTO=dhcp ONBOOT=yes For STATIC the file looks like: DEVICE=eth0 IPADDR=*192.168.1.2* NETMASK=*255.255.255.0* NETWORK=*192.168.1.0* BROADCAST=*192.168.1.255* GATEWAY=*192.168.1.1* BOOTPROTO=none ONBOOT=yes Then: 1. Set the hostname: **vi /etc/sysconfig/network** **HOSTNAME="mypc.acme.com"** 2. Enter hosts file information: **vi /etc/hosts** **192.168.0.10 mypc.acme.com mypc** 3. Enter the DNS server information: **vi /etc/resolv.conf** **nameserver 192.168.0.77**

Static the Gateway	Either use the **/etc/sysconfig/network-scripts/eth0** (preferred method) or use the **/etc/rc.d/rc.local** file
DCHP	**dhcpcd *eth0***
Take interface up or down	**ifup *eth0*** **ifdown *eth0***
Restart Network	**/etc/rc.d/init.d/network restart**
Promiscuous mode	**ifconfig –*eth0* -promisc**
Determine IP Status – including whether interface is in promiscuous mode	**ip link**
Listening ports – netstat	**netstat –tpan** will show ports what ports are open on the external interface – nice to use to verify things like Apache or SSH are listening for incoming connections
IP Information	There is no man page, so use `ip help` instead if you need help. `ip a` – display addresses `ip a help` – further help for address `ip r` – display routes (similar to the `route` command) `ip a a 10.10.146.58/30 dev eth0 label eth0:0` – add a virtual interface to dev eth0 `ip l s eth0 mtu 1200` – change the mtu to 1200

Recover a Lost Root Password	
Recover root password	For LILO: 1. Boot linux 2. At the LILO prompt (i.e. Boot:) type **linux single** 3. At the command line#, type **passwd root** 4. Change the password 5. Reboot and try new password For GRUB: 1. Boot linux 2. At the GRUB menu, press **e** (for edit) 3. Add the word "single" at end of line 4. Press "b" to boot

PSCP – Secure Copy from Windows (using Putty tools)	
Pull a File	**pscp *root@source_ip:/full_source_path dest_path*** Example: pscp root@207.169.53.93:/home/projects/corpone/CorpOne-SAi_Test.html ./
Push a File	**pscp *source_path root@dest_ip:/dest_path*** Example: pscp "c:\my documents\whatever.txt" root@207.169.53.93:/home/projects/corpone/CorpOne-SAi_Test.html

SCP – Secure Copy from Linux	
Pull a File	**scp *root@source_ip:/full_source_path dest_path*** Example: scp root@207.169.53.93:/home/projects/corpone/CorpOne-SAi_Test.html ./
Push a File	**scp *source_path root@dest_ip:/dest_path*** Example: scp "/root/docs/whatever.txt" root@207.169.53.93:/home/projects/

User Administration	
Add a user	**useradd**
Modify a user	**usermod**
Delete a user	**userdel**
Change password	**passwd *username*** (if logged on as root, can change anyone's password – if logged on as user, can only change own password, no need to type your username as the switch)
Group administration command	**groupadd** **groupmod** **groupdel** **gpasswd**
Who is logged on	**users**
What group(s) do I belong to?	**groups**
User login activity	**last** **lastb** **who** **ac**
List last logged in users	**Last** – shows the following information: • Type of Login • Process ID of login process • Device name of tty • Init ID or abbreviated ttyname • User Name • Hostname for remote login • Exit Status of a process • Time entry was made • IP address of remote host
Manage user accounts	• chage (1) - change user password expiry information • groups (1) - print the groups a user is in • newusers (8) - update and create new users in batch • passwd (1) - update a user's authentication tokens(s) • nologin (5) - prevent non-root users from log into the system • su (1) - run a shell with substitute user and group IDs • useradd (8) - Create a new user or update default new user information • userdel (8) - Delete a user account and related files • usermod (8) - Modify a user account • chgrp (1) - change the group ownership of files • chown (1) - change the user and group ownership of files • gpasswd (1) - administer the /etc/group file • groupadd (8) - Create a new group

	• groupdel (8) - Delete a group • groupmod (8) - Modify a group • groups (1) - print the groups a user is in • grpck (8) - verify integrity of group files • pwconv (8) - convert to and from shadow passwords • pwunconv (8) - convert to and from shadow passwords • grpconv (8) - convert to and from shadow passwords • grpunconv (8)- convert to and from shadow passwords • vipw (8) - edit the password file • vigr (8) - edit the group file
List user password settings	**chage –l *[account_name]***
Change user password settings	**chage [*account_name]*** To set a non-expiring password, set the Maximum Password Age to **99999**
Linux Security Admin Guide	http://www.nic.com/~dave/SecurityAdminGuide/SecurityAdminGuide-5.html

Permissions Discussion

UNIX/Linux has no per-user permissions/rights/policies. Everything is done on files, making sure who can read/write/exectute the right files.

To check the current permissions of files, run "ls -l". If you run "ls -l /", to list all files in the root directory, you will get output that looks something like this:

Code:

```
drwxr-xr-x    2 root     root         4096 2003-07-15 22:41 bin
drwxr-xr-x    4 root     root         1024 2003-07-16 03:26 boot
drwxr-xr-x   20 root     root       118784 2003-07-16 03:32 dev
...
```

The first column (that looks like drwxr-xr-x) is the mode of the file. The first character indicates what kind of file it is. d, as in this case, means directory. - means regular file, and then there are some other file types of which you need not know more now, like named FIFOs, sockets, devices, etc.

The rwxr-xr-x is the permissions of file. The first rwx means that the owner of the file can read, write and execute the file. For directories, the right to execute it means the right to use it. Just being able to read a directory means that you can read what files are in the directory, but you won't be able to use them without the execute permission on the directory. The first r-x means that those that are in the same group as the file have read and execute permissions to the file. The second r-x means that all those that are neither the owner of the file nor are in the same group as the file have read and execute permissions on it. So r=read, w=write and x=execute, and the first group of three applies to the owner of the file, the second group applies to those in the same group of the file, and the third group applies to all other users.

The second column (2, 4 and 20 in this case) is the number of links that the file has. Don't care about that for now.

The third column is the owner of the file, in this case root.

The fourth column is the group of the file, in this case the root group.

The fifth column is the size of the file.

The sixth is the time the file was last modified

The seventh is naturally the name of the file.

Changing Permissions

there are three command that you need to learn.

chmod /*change the ppermissions of a file*/
chown /*change the owner of the file*/
chgrp /*change the group that the file belongs to*/

example:
`chmod 777 somefile.file` /*this gives exe write and read to all */ not recommended for any file.

for chmod here is a list of the numbers and what they meen

the 100's are for the owner of the file
400 reed
200 write
100 execute

10's are for the group of the file
40 read
20 write
10 execute

1's are for everyone else
4 read
2 write
1 execute

you add the number together to get different permissions

To change multiple files, you can use `chmod –R` for an entire directory, or use wildcards like `chmod 755 *.txt.`

XWindows	
Configure XWindows from command line	**Xconfigurator**
Cycle through video settings	**[CTRL] + [ALT] + [+]**
Leave XWindows NOW – handy when you can't see the screen due to video resolution problems	**[CTRL] + [ALT] + [BACKSPACE]**
Shoot Xwindows back through your SSH session	Launching SSH with the –X switch will forward X-windows back through your SSH tunnel `ssh -X`

MYSQL	
Open mysql command line from same server	**mysql –u*user* –p*password*** ***i.e. mysql –uroot –pwopnam***
Open mysql command line from remote server	**mysql –u*user* –p*password* –h*ip_address database_name*** ***i.e mysql –uroot –pwopnam –h207.169.53.5 snort***
Connect to database	**connect *database_name*** ***i.e. connect snort***
Query table names	**show tables;**
Query columns from a table	**show columns from *table_name;*** ***i.e show columns from acid_event;***
Query the most recent data (field) in a column	**select max(*field*) from *table_name*;** ***i.e. select max(timestamp) from acid_event***
Misc.	**show databases** **use *database_name*** **show tables** **show grants for *user@localhost***

Backup Database	`mysqldump –u root –p --opt database_name > /path/filename.sql`
	
	For example:
	`mysqldump –u root –p --opt snort > /root/snort.sql`

Restore Database	`mysql –u root –p database_name < /path/filename.sql`
	
	For example:
	`mysql –u root –p snort < /root/snort.sql`

MISC Stuff	`revoke all privileges on pec.* from matt@'%';`
	
	`GRANT CREATE, INSERT, SELECT, UPDATE ON pec.* TO change_user@localhost IDENTIFIED BY 'password';`
	
	`flush privileges;`
	
	`mysql -u changes –p`
	
	`USE mysql`
	`SELECT user,host FROM user;`
	`DELETE FROM user WHERE user="";`
	
	`CONNECT pec;`
	`SHOW TABLES;`
	
	`SELECT * FROM table_name;`
	`SHOW COLUMNS FROM table_name;`
	
	`SELECT MAX(date_opened) FROM changes;`
	`SHOW GRANTS FOR matt@'%';`
	`SHOW GRANTS FOR matt@localhost;`
	
	`ALTER TABLE `table_name` DROP `field_name`;`

```
DROP TABLE table_name;

DROP DATABASE db_name;

SHOW COLUMNS FROM table_name;

select * from users;
select * from systems;
select * from changes;

select * from users order by user_first_name;
select * from users order by user_first_name desc;
select * from users order by user_last_name;
select * from users order by user_last_name desc;

select * from users where user_last_name = 'Westbrook';
select * from users where user_last_name != 'Westbrook';
select * from users where user_first_name = 'Matt';
select * from users where user_first_name != 'Matt';

select * from users where user_first_name like 'B%';
select * from users where user_first_name like 'M%';
select * from users where user_first_name like '%r';

UPDATE table_name SET column1_name='new_value' where
column_name='current_value';
UPDATE systems SET system_name='IISMBS' WHERE
system_name='jax-vst1';
```

Change password:

```
set password for change_user@localhost=password('password');
```

Delete specific row(s) in a table:

```
delete from table_name where column_name=criteria;
```

Delete all rows in a table:

```
delete from table_name;
```

Or

```
truncate table_name;
```

File Editing and Scripts	
Vi - File Editor	**vi** *filename* Press **Insert** key to type in inert mode Press **Escape** key to leave insert and enter edit mode To search, type /*string* and Enter Search & replace on a line, type :s/*oldpattern*/*newpattern*/ Global search & replace, :%s/*oldpattern*/*newpattern*/g Add a c at end of line if you want to confirm each change :%s/*oldpattern*/*newpattern*/gc :wq = write file (save) and quit :q! = quit without saving . = repeat last command de = delete to end dd = delete line dw = delete word u = undo last command (can be used multiple times) yy = copy cc = cut p = paste
Batch file	Linux uses scripts rather than batch files. There are three steps to creating a script. 1. Design the script 2. Type it into a text file 3. Set the file's attributes to tell Linux that it's a script file (chmod a+x *filename*) Ok, so let's walk through it. Here's an extremely simple script that just echoes back whatever you type in, called sayit: echo $1 Change to your home directory. Use vi to create the script by typing **vi sayit**. Then hit the **Insert** key and type **echo $1**. Press the **ESC** key. Now type **:x** to save and exit. Type **cat sayit** to type out the file and verify it. Finally, type **chmod a+x sayit** to make Linux see it as a script file. Now type **./sayit "Hello World"** and you should see Hello World echoed back to the screen (we used quotes because there was a space, just like Windows). Note that you have to type the whole path, even if you're sitting in the same directory. Here's another one, using a variable. Change the line **echo $1** to **echo Hello $USERNAME**. Then save and run it. Check it out! Linux has environment variables similar to Windows. While Windows variables are surrouned by % (i.e. %username$) Linux is simply preceded by a $.
echo to email	echo "this is my text to email" | mail bwestbrook@corpone.org

	Secure Default Firewall Ruleset *** Explanation ***	As a brief explanation, the firewall rules for iptables are not really kept in any editable file. That is, the rules, once loaded, exist in memory and will overwrite the file they came from. So how do you configure iptables? And how does it load it's ruleset after a reboot? Well, one way is to make changes to the ruleset in memory, on the fly. You then tell iptables to save the rules in memory to a file. When the box reboots, iptables reads the rules from this saved file. So why can't you just change the actual rules in file? Because it's overwritten any time that you save the rules. And you can't delete rules by simply re-reading the file – the file will append to the rules in memory. Instead, you should create a file of your own with all your firewall rules and comments, run your file to add, delete or modify rules in memory, and then save the iptables memory to the **/etc/sysconfig/iptables** file. Whew! Rather then create a script to do our changes, we will perform the changes on the fly. We'll then save the memory to a file so they get removed permanently on reboot. To do this, we will perform the following: 1. Delete (flush) all of the current rules 2. Define our chains/tables in memory 3. Add our "default" rules in memory 4. Add other rules in memory as needed 5. Save the new iptables from memory to the iptables file 6. Restart iptables to verify our changes
	New Default Firewall Ruleset *** All Distros ***	**** IMPORTANT NOTE:** Making these changes via a remote SSH connection WILL lock you out almost immediately. In order to stay consistent with our local firewall rules, we will remove any rules that came with the distribution and set up our own. First, let's remove anything that may currently exist in the iptables rules by "flushing" everything as follows: **iptables -F** **iptables –F INPUT** **iptables –F OUTPUT** **iptables –F FORWARD** **iptables –F –t mangle** **iptables –F –t nat** **iptables –X** **iptables –Z** Define three chains, INPUT, FORWARD and OUTPUT. These three rules by themselves will drop all incoming packets and all forward packets. All packets initiated by the host will be allowed. **iptables –P INPUT DROP** **iptables –P FORWARD DROP** **iptables –P OUTPUT ACCEPT** As a way of explanation, here is the meaning of some of the most common options for iptables commands:

–i – Interface tells the kernel which interface should be filtered.

–p – Protocol defines the protocol that the rule will apply to. Protocols are lised in the **/etc/protocols** file, and you can define rules for any of them.

-s – Source IP address

-d – Destination IP address

–m – Match is a directive for matching. Commonly you can match state, protocol or both.

-j – Jump to what to do if the packet matches the rule

--dport – Destination Port.

--sport – Source Port

--state – defines packet state. There are four states: INVALID, ESTABLISHED, NEW and RELATED. The first three are pretty obvious, while RELATED is a new connection that is associated to an existing connection, such as the FTP data connection being RELATED to the FTP control connection.

Now let's allow some exceptions to our default of dropping all inbound packets. All of the following rules override the global DROP command that we started with.

This rule will accept anything that originates from the local loopback interface and allow it to be used by user applications:
iptables -A INPUT -i lo -j ACCEPT

This rule allows connections that have already been established and are in the connection table maintained by the kernel, such as responses to our HTTP requests (line wrapped):
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT

This rule allows any ICMP packets so we can perform pings and traceroutes as well as respond to pings:
iptables -A INPUT –p ICMP –j ACCEPT

The following rules are used to allow access to various applications. Use them as needed (line wrapped):

SSH
iptables -A INPUT -p tcp -m tcp --dport 22 –j ACCEPT

Webmin
iptables -A INPUT -p tcp -m tcp --dport 10000 –j ACCEPT

HTTPS
iptables -A INPUT -p tcp -m tcp --dport 443 –j ACCEPT

Tomcat (Port 8080)
iptables -A INPUT -p tcp -m tcp --dport 8080 –j ACCEPT

		MySQL **iptables -A INPUT -p tcp -m tcp --dport 3306 –j ACCEPT** The very last rule we will put in is to drop all remaining packets that didn't match any of our rules. This is simply good practice: **iptables –A INPUT –j DROP** Finally, save your revised rules to a file, restart iptables and then verify your rules are all in place: **service iptables save** **service iptables restart** **iptables -L**
	Other Useful iptables Commands	See the full ruleset file: **less /etc/sysconfig/iptables** Print out the current iptables with corresponding **chain numbers** (needed to manipulate and insert lines): **iptables -L --line** NOTE: When you print out the current ruleset you may notice that the first line may appear to allow anything from anywhere. To see what it truly is, you'll want to look at the actual file in **/etc/sysconfig/iptables** To insert a rule into a specific position, use: **iptables –I INPUT *chain-number new-rule*** To replace a specific rule, use: **iptables –R INPUT *chain-number new-rule*** To add a line at the end of the rule set, use: **iptables –A INPUT *new-rule*** For multiple, sequential ports, use the : symbol, such as: **iptables –A INPUT –p tcp –m tcp --dport 11000:11099** For multiple, non-sequential ports, use multiport, such as: **iptables –A INPUT –p tcp –m multiport --dport 80,443,8080** To check what rules are being used, you can view the Byte and Packet counters: **iptables – vL** To clear (zero-out) the counters: **iptables -Z** To conduct port redirection – example is port 80 to port 8080: **iptables –t nat –A PREROUTING –p tcp --dport 80 –j REDIRECT --to-port 8080** To conduct port redirection for local requests on the box – same example: **iptables –t nat –A OUTPUT –d 127.0.0.1 –p tcp --dport 80 –j REDIRECT --to-port 8080**

GPG

√		Description
	Key Creation	Under GPG, there is no GUI. To create a key under, perform the following steps at a command prompt: 1. `gpg --gen-key` 2. Select `1` for **DSA and ElGamal** 3. For **What keysize do you want**, type `2048` 4. For **Key is valid for**, enter `13m` 5. Verify date of expiration is 13 months from today and enter `Y` for **Is this correct?** 6. For **Real Name**, enter a logical name for the particular key you are creating 7. For **Email Address**, enter your email address 8. For **Comment**, enter a brief comment for what the key is to be used for or leave it blank 9. Review your entries, make any changes as necessary, then type `O` for Okay 10. Create a **Passphrase** based on the requirements in the Standards section above. 11. Reenter the passphrase 12. Now begin moving the mouse and typing junk at the keyboard to create random information for the key generator. 13. When it's completed, you will be back at the command prompt 14. Export the public key by typing `gpg --export KeyID > KeyID-pubkey.pgp` where **KeyID** is the **Real Name** you entered when creating the key 15. Export the private key by typing `gpg --export-secret-key KeyID > KeyID-seckey.pgp` where **KeyID** is the **Real Name** you entered when creating the key 16. Copy the two .pgp files you created to a floppy, USB, etc. for backup purposes. 17. Delete the exported files from the system
	GPG Commands	The *keyname* referenced in the commands below can be either the owners's name, the user's email address or the key ID. If you happen to have gpg keys with the same owner's name and email address, you can differentiate between them based on the key ID. What is the key ID? It is the hex ID given after the keysize. For instance, using the command `gpg --list-keys` will show all the keys in your keyring, like this: # gpg --list-keys pub 1024D/**28394F0E** 2006-09-07 [expires: 2006-10-07] uid **MyTestKey** (testkey1) <**testkey@corpone.org**> sub 2048g/020D77F8 2006-09-07 [expires: 2006-10-07] pub 1024D/**2608AE83** 2006-09-07 [expires: 2007-10-02] uid **MyTestKey** (testkey2) <**testkey@corpone.org**> sub 2048g/C25E7EF4 2006-09-07 [expires: 2007-10-02] You can see that there are two keys with the same name (MyTestKey) and email address (teskey@corpone.org). However, the key IDs will always be different. The keysize is shown, followed by the public hex key ID, shown

		highlighed above. Using this example you can therefore use the key ID to delete a key, like this: **gpg --delete-secret-key 2608ae83** **gpg --delete-key 2608ae83** Generate (create) a key: **gpg –-gen-key** Export the public key of the designated name to a file: **gpg --export –a *keyname* > *keyname*-pubkey.key** Export the secret key of the designated name to a file: **gpg --export-secret-key –a *keyname* > *keyname*-pubkey.key** Import a public key: **gpg –-import *filename*** Import a secret key: **gpg –-allow-secret-key-import –-import *filename*** Export the fingerprint of a key to verify a public key: **gpg --fingerprint > *keyname*-fingerprint** List your public key(s): **gpg --list-keys** List your secret key(s): **gpg --list-secret-key** Delete your secret-key by name: **gpg --delete-secret-keys *keyname*** Delete a public key: **gpg --delete-keys *keyname*** Sign and encrypt the given filename with your default key to the receiver: **gpg –se –r *receiver_keyname filename*** Decrypt a filename with your default-key: **gpg –d *filename***
	Trust Other Keys	To trust other keys that you've imported: **gpg --edit-key *"Real_Name_of_Key"*** **Command> trust => 5** for ultimate trust **Command> check** **Command> enable** **Command> save**

Setup NTP for Synching Time

For more information, all of the following have good information:

- http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch24_:_The_NTP_Server
- http://www.brennan.id.au/09-Network_Time_Protocol.html
- http://ntp.isc.org/bin/view/Support/AccessRestrictions
- http://www.eecis.udel.edu/~mills/ntp/html/accopt.html

Note that this configuration is for a protected, internal server, NOT an Internet accessible server. That is, you trust (relatively so) the clients on your internal network.

Configure NTP

√		Description
	Backup	First backup your original ntp.conf file: `cp /etc/ntp.conf /etc/ntp.conf.orig`
	Locate an Internet Time Server	Go to http://ntp.isc.org/bin/view/Servers/WebHome and locate 2 or 3 servers in your timezone, near you geographically. For illustration purposes, we'll suppose you gathered the following three servers: 198.30.92.2 128.10.252.7 130.126.24.44

	Configure the Server	Create a new `/etc/ntp.conf` file and place the following contents in the file:
		
		This first line will ignore all connections by default.:
		`# Ignore all NTP connections by default`
		`restrict default ignore`
		
		Allow full control of our time service from the local host – while the statement seems odd (restrict) it really means allow with the following restrictions – which there will be none:
		`# Allow full access from localhost`
		`restrict 127.0.0.1`
		
		Now enter the IP addresses of the servers you chose to update from:
		`# Servers and their restrictions`
		`server 198.30.92.2`
		`server 128.10.252.7`
		`server 130.126.24.44`
		
		Set the restrictions for our upstream servers
		`restrict 198.30.92.2 mask 255.255.255.255 nomodify`
		`restrict 128.10.252.7 mask 255.255.255.255 nomodify`
		`restrict 130.126.24.44 mask 255.255.255.255 nomodify`

		Now allow whatever hosts/networks you are going to allow to query your server for time – for example:
		`# Allowed clients`
		`restrict 10.0.0.0 mask 255.0.0.0 nomodify`
		`restrict 172.16.0.0 mask 255.255.224.0 nomodify`
		`restrict 192.168.0.0 mask 255.255.0.0 nomodify`
		`restrict 207.169.53.0 mask 255.255.255.0 nomodify`
		
		Set the logfile:
		`#Logfile`
		`logconfig all`
		`logfile /var/log/ntpd`
		
		Set the driftfile. This file will hold the "drift" or time shift due to network latency:
		`# Driftfile`
		`driftfile /etc/ntp/drift`

	Set "Step-Tickers"	The server in the /etc/ntp/step-tickers file will be the ones that your server will sync with immediately when it boots up:
		
		`vim /etc/ntp/step-tickers`
		
		Now simply add the IP addresses of your upstream servers that you're getting time from, one on each line, such as:
		
		`198.30.92.2`
		`128.10.252.7`
		`130.126.24.44`
		
		That's it. Save and close the file.

	Update Your System Clock	Run the ntpdate command against each IP address of your upstream servers 2 or 3 times each. You should see the "jitter" come way down between the first time you run it and the last:
		
		`ntpdate 198.30.92.2`
		`ntpdate 198.30.92.2`
		`ntpdate 128.10.252.7`
		`ntpdate 128.10.252.7`
		`ntpdate 130.126.24.44`
		`ntpdate 130.126.24.44`

	Start your Service	Start your NTP daemon:
		`service ntpd restart`
		
		Then turn it on to run automatically:
		`chkconfig –level 2345 ntpd on`

Now that ntp is running, to determine if it is synchronizing properly, issue the command (wait a couple of minutes after starting the ntpd service before running this command):

```
ntpq –p
```

This command should show output similar to below:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 LOCAL(0)        LOCAL(0)         10 l   1   64    3    0.000    0.000   0.015
*navobs1.oar.net .USNO.            1 u   5   64    1  222.555   68.548   0.015
+darkcity.cerias .GPS.             1 u  60   64    1   44.732  -19.367   0.015
+ntp-2.gw.uiuc.e truechimer.cso.   2 u  61   64    1   22.875  -27.707   0.015
```

A telltale sign that you don't have proper synchronization is when all the remote servers have jitters of 4000 with delay and reach values of zero.

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 LOCAL(0)        LOCAL(0)        10 u    -   64    0    0.000    0.000 4000.00
 navobs1.oar.net 0.0.0.0         16 u    -   64    0    0.000    0.000 4000.00
 darkcity.cerias 0.0.0.0         16 u    -   64    0    0.000    0.000 4000.00
 ntp-2.gw.uiuc.e 0.0.0.0         16 u    -   64    0    0.000    0.000 4000.00
```

This could be caused by:

- The **`restrict default ignore`** statement in the **`/etc/ntp.conf`** file not being commented out;
- A firewall blocking access to your Internet NTP servers over port UDP 123.

Useful troubleshooting command (must have the **`ntpd`** daemon service stopped to use this):

- **`ntptrace –d –v <time_server>`**

Test NTP

Want to test the whole thing? Set your system and hardware clock to a wacked date, like follows:

`date 0603008501999` (this sets the current time on the software clock to June 3, 1999, 8:50am

`hwclock --systohc` (this sets the hardware clock to match the software clock)

Now, reboot the server and see if both of the clocks re-synched properly by issuing the following two commands to check the software and hardware clocks, respectively:

```
date
hwclock
```

If you see some errors such as **`set_rtc_mmss: can't update from x to x`** (where x = a number), then you need to manually set the hardware clock. This error stems from a known kernel issue (as of October 27, 2003) when the hardware clock is off from the software clock by more than some amount (there seems to be some debate as to what that amount is). To fix this, set the clock manually and reboot to see if the error(s) go away:

```
hwclock –systohc
reboot
```

Help File Library: **Bash Scripting Cheat Sheet**

Written By: ph34r
A quick cheat sheet for programmers who want to do shell scripting. This is not intended to teach programming, etc. but it is intended for a someone who knows one programming language to begin learning about bash scripting.

Basics

All bash scripts must tell the o/s what to use as the interpreter. The first line of any script should be:
#!/bin/bash

You must make bash scripts executable.
chmod +x filename

Variables

Create a variable - just assign value. Variables are non-datatyped (a variable can hold strings, numbers, etc. with out being defined as such).
varname=value

Access a variable by putting $ on the front of the name
echo $varname

Values passed in from the command line as arguments are accessed as $# where #= the index of the variable in the array of values being passed in. This array is base 1 not base 0.
command var1 var2 var3 .... varX
$1 contains whatever var1 was, $2 contains whatever var2 was, etc.

Built in variables:

Variable	Use
$1-$N	Stores the arguments (variables) that were passed to the shell program from the command line.
$?	Stores the exit value of the last command that was executed.
$0	Stores the first word of the entered command (the name of the shell program).
$*	Stores all the arguments that were entered on the command line ($1 $2 ...).
"$@"	Stores all the arguments that were entered on the command line, individually quoted ("$1" "$2" ...).

Quote Marks

Regular double quotes ("like these") make the shell ignore whitespace and count it all as one argument being passed or string to use. Special characters inside are still noticed/obeyed. Single quotes 'like this' make the interpreting shell ignore all special characters in whatever string is being passed.

The back single quote marks (`command`) perform a different function. They are used when you want to use the results of a command in another command. For example, if you wanted to set the value of the variable contents equal to the list of files in the current directory, you would type the following command: contents=`ls`, the results of the ls program are put in the variable contents.

Logic and comparisons

A command called test is used to evaluate conditional expressions, such as a if-then statement that checks the entrance/exit criteria for a loop.

test expression
Or
[expression]

Numeric Comparisons

int1 -eq int2	Returns True if int1 is equal to int2.
int1 -ge int2	Returns True if int1 is greater than or equal to int2.
int1 -gt int2	Returns True if int1 is greater than int2.
int1 -le int2	Returns True if int1 is less than or equal to int2
int1 -lt int2	Returns True if int1 is less than int2
int1 -ne int2	Returns True if int1 is not equal to int2

String Comparisons

str1 = str2	Returns True if str1 is identical to str2.
str1 != str2	Returns True if str1 is not identical to str2.
str	Returns True if str is not null.
-n str	Returns True if the length of str is greater than zero.
-z str	Returns True if the length of str is equal to zero. (zero is different than null)

File Comparisons

-d filename	Returns True if file, filename is a directory.
-f filename	Returns True if file, filename is an ordinary file.

-r filename	Returns True if file, filename can be read by the process.
-s filename	Returns True if file, filename has a nonzero length.
-w filename	Returns True if file, filename can be written by the process.
-x filename	Returns True if file, filename is executable.

Expression Comparisons

!expression	Returns true if expression is not true
expr1 -a expr2	Returns True if expr1 and expr2 are true. (&& , and)
expr1 -o expr2	Returns True if expr1 or expr2 is true. (||, or)

If Statements

If...then

```
if [ expression ]
        then
                commands
fi
```

If..then...else

```
if [ expression ]
        then
                commands
        else
                commands
fi
```

If..then...else If...else

```
if [ expression ]
        then
                commands
elif [ expression2 ]
        then
                commands
else
                commands
fi
```

Case select

```
case string1 in
        str1)
                commands;;
        str2)
                commands;;
        *)
                commands;;
esac
```

string1 is compared to str1 and str2. If one of these strings matches string1, the commands up until the double semicolon (; ;) are executed. If neither str1 nor str2 matches string1, the commands associated with the asterisk are executed. This is the default case condition because the asterisk matches all strings.

Iteration (Loops)

```
for var1 in list
do
        commands
done
```

This executes once for each item in the list. This list can be a variable that contains several words separated by spaces (such as output from ls or cat), or it can be a list of values that is typed directly into the statement. Each time through the loop, the variable var1 is assigned the current item in the list, until the last one is reached.

```
while [ expression ]
do
        commands
done

until [ expression ]
do
        commands
done
```

Functions

Create a function:

```
fname(){
        commands
}
```

Call it by using the following syntax: fname

Or, create a function that accepts arguments:

```
fname2 (arg1,arg2...argN){
        commands
}
```

And call it with: fname2 arg1 arg2 ... argN

Debugging

The shell has a number of flags that make debugging easier:

`sh -n command`

Read the shell script but don't execute the commands. IE. check syntax.

`sh –x command`

Display commands and arguments as they're executed. In a lot of my shell scripts you'll see

```
# Uncomment the next line for testing
# set -x
```

See also:

http://www.linux.org/docs/ldp/howto/Bash-Prog-Intro-HOWTO.html

Test your Linux Skills

1. Create a new directory in your home directory
2. Make a file from scratch in this directory
3. Mount the floppy drive
4. Type out the file you created
5. Clear the screen
6. Edit the file, make some changes to it and save it
7. Copy the file to the floppy
8. Delete the file
9. Copy the file back from the floppy
10. Read a CD-ROM directory
11. Perform a search for the file linux.conf
12. Print the routing table
13. Check your IP address
14. Check your DNS servers
15. Rename your machine
16. Reboot and make sure your new machine name stuck
17. Add a route to the routing table
18. Delete the route you just added
19. Create a new default gateway
20. Put the default gateway back to how it was
21. Show your path

More Resources

Understanding Linux Configuration Files

http://www-106.ibm.com/developerworks/linux/library/l-config.html

Linux Shortcuts and Commands:

Linux Newbie Administrator Guide
by Stan and Peter Klimas

This is a practical selection of the commands we use most often. Press `<Tab>` to see the listing of all available command (on your PATH). On my small home system, it says there are 2595 executables on my PATH. Many of these "commands" can be accessed from your favourite GUI front-end (probably KDE or Gnome) by clicking on the right menu or button. They can all be run from the command line. Programs that require GUI have to be run from a terminal opened under a GUI.

Legend:
<> = single special or function key on the keyboard. For example <Ctrl> indicates the "control" key.
italic = name of the file or variable you probably want to substitute with your own.
`fixed width` = in-line Linux commands and filenames.

Notes for the UNIX Clueless:
1. LINUX IS CASE-SENSITIVE. For example: Netscape, NETSCAPE and nEtscape are three different commands. Also my_filE, my_file, and my_FILE are three different files. Your user login name and password are also case sensitive. (This goes with the tradition of UNIX and the "c" programming language being case sensitive.)
2. Filenames can be up to 256 characters long and can contain letters, numbers, "." (dot), "_" (underscore), "-" (dash), plus some other not recommended characters.
3. Files with names starting with "." are normally not shown by the `ls` (list) or dir commands. Think of these files as "hidden". Use `ls -a` (list with the option "all") to see these files.
4. "/" is an equivalent to DOS "\" (root directory, meaning the parent of all other directories).
5. Under Linux, all directories appear under a single directory tree (there are no DOS-style drive letters).
6. In a configuration file, a line starting with # is a comment.

7.1 Linux essential shortcuts and sanity commands

`<Ctrl><Alt><F1>`
Switch to the first text terminal. Under Linux you can have several (6 in standard setup) terminals opened at the same time.
`<Ctrl><Alt><Fn>` (n=1..6)
Switch to the nth text terminal.
`tty`
Print the name of the terminal in which you are typing this command.
`<Ctrl><Alt><F7>`
Switch to the first GUI terminal (if X-windows is running on this terminal).
`<Ctrl><Alt><Fn>` (n=7..12)
Switch to the nth GUI terminal (if a GUI terminal is running on screen n-1). On default, nothing is running on terminals
8 to 12, but you can run another server there.
`<Tab>`
(In a text terminal) Autocomplete the command if there is only one option, or else show all the

available options.
THIS SHORTCUT IS GREAT! It even works at LILO prompt!

`<ArrowUp>`

Scroll and edit the command history. Press <Enter> to execute.

`<Shift><PgUp>`

Scroll terminal output up. Work also at the login prompt, so you can scroll through your bootup messages.

`<Shift><PgDown>`

Scroll terminal output down.

`<Ctrl><Alt><+>`

(in X-windows) Change to the next X-server resolution (if you set up the X-server to more than one resolution). For multiple resolutions on my standard SVGA card/monitor, I have the following line in the file `/etc/X11/XF86Config` (the first resolution starts on default, the largest determines the size of the "virtual screen"):

```
Modes "1024x768" "800x600" "640x480" "512x384" "480x300" "400x300" "1152x864"
```

`<Ctrl><Alt><->`

(in X-windows) Change to the previous X-server resolution.

`<Ctrl><Alt><BkSpc>`

(in X-windows) Kill the current X-windows server. Use if the X-windows server crushes and cannot be exited normally.

`<Ctrl><Alt><Del>`

Shut down the system and reboot. This is the normal shutdown command for a user at the text-mode console. Don't just press the "reset" button for shutdown!

`<Ctrl>c`

Kill the current process (mostly in the text mode for small applications).

`<Ctrl>d`

Log out from the current terminal. See also the next command.

`<Ctrl>d`

Send [End-of-File] to the current process. Don't press it twice else you also log out (see the previous command).

`<Ctrl>s`

Stop the transfer to the terminal.

`<Ctrl>q`

Resume the transfer to the terminal. Try if your terminal mysteriously stops responding.

`<Ctrl>z`

Send the current process to the background.

`exit`

Logout. I can also use `logout` for the same effect. (If you have started a second shell, e.g., using `bash` the second shell will be exited and you will be back in the first shell, not logged out.)

`reset`

Restore a screwed-up terminal (a terminal showing funny characters) to default setting. Use if you tried to "cat" a binary file. You may not be able to see the command as you type it.

`<MiddleMouseButton>`

Paste the text which is currently highlighted somewhere else. This is the normal "copy-paste" operation in Linux. (It doesn't work with Netscape and WordPerfect which use the MS Windows-style "copy-paste". It does work in the text terminal if you enabled "gpm" service using "setup".) Best used with a Linux-ready 3-button mouse (Logitech or similar) or else set "3-mouse button emulation").

`~`

(tilde) My home directory (normally the directory `/home/my_login_name`). For example, the

command `cd ~/my_dir` will change my working directory to the subdirectory "*my_dir*" under my home directory. Typing just "cd" alone is an equivalent of the command "cd ~".

`.`

(dot) Current directory. For example, `./my_program` will attempt to execute the file "my_program" located in your current working directory.

`..`

(two dots) Directory parent to the current one. For example, the command `cd ..` will change my current working directory one one level up.

7.2 Common Linux commands--system info

`pwd`

Print working directory, i.e., display the name of my current directory on the screen.

`hostname`

Print the name of the local host (the machine on which you are working). Use `netconf` (as root) to change the name of the machine.

`whoami`

Print my login name.

`id` *`username`*

Print user id (uid) and his/her group id (gid), effective id (if different than the real id) and the supplementary groups.

`date`

Print or change the operating system date and time. E.g., I could change the date and time to 2000-12-31 23:57 using this command:

```
date 123123572000
```

To set the hardware (BIOS) clock from the system (Linux) clock, use the command (as root)

`setclock`

`time`

Determine the amount of time that it takes for a process to complete + other info. Don't confuse it with the `date` command. E.g. I can find out how long it takes to display a directory content using:

```
time ls
```

`who`

Determine the users logged on the machine.

```
rwho -a
```

(=remote who) Determine all users logged on your network. The rwho service must be enabled for this command to run. If it isn't, run setup as root to enable "rwho".

`finger` *`user_name`*

System info about a user. Try: `finger root`

`last`

Show listing of users last logged-in on your system.

```
history | more
```

Show the last (1000 or so) commands executed from the command line on the current account. The "| more" causes the display to stop after each screenful.

`uptime`

Show the amount of time since the last reboot.

`ps`

(=print status) List the processes currently run by the current user.

```
ps axu | more
```

List all the processes currently running, even those without the controlling terminal, together with the name of the user that owns each process.

`top`

Keep listing the currently running processes, sorted by cpu usage (top users first). In KDE, you

can get GUI-based Ktop from "K"menu under "System"-"Task Manager" (or by executing "ktop" in an X-terminal).

`uname -a`

(= Unix name with option "all") Info on your (local) server. I can also use `guname` (in X-window terminal) to display the info more nicely.

`free`

Memory info (in kilobytes).

`df -h`

(=disk free) Print disk info about all the filesystems (in human-readable form)

`du / -bh | more`

(=disk usage) Print detailed disk usage for each subdirectory starting at the "/" (root) directory (in human legible form).

`cat /proc/cpuinfo`

Cpu info--it show the content of the file `cpuinfo`. Note that the files in the `/proc` directory are not real files--they are hooks to look at information available to the kernel.

`cat /proc/interrupts`

List the interrupts in use.

`cat /proc/version`

Linux version and other info

`cat /proc/filesystems`

Show the types of filesystems currently in use.

`cat /etc/printcap`

Show the setup of printers.

`lsmod`

(As root. Use `/sbin/lsmod` to execute this command when you are a non-root user.) Show the kernel modules currently loaded.

`set|more`

Show the current user environment.

`echo $PATH`

Show the content of the environment variable "PATH". This command can be used to show other environment variables as well. Use "set" to see the full environment.

`dmesg | less`

Print kernel messages (the content of the so-called kernel ring buffer). Press "q" to quit "less". Use `less /var/log/dmesg` to see what "dmesg" dumped into this file right after the last system bootup.

7.3 Basic operations

`any_command` `--help |more`

Display a brief help on a command (works with most commands). "--help" works similar to DOS "/h" switch. The "more" pipe is needed if the output is longer than one screen.

`man` *`topic`*

Display the contents of the system manual pages (help) on the topic. Try `man man` first. Press "q" to quit the viewer. The command `info` *`topic`* works similar and may contain more up-to-date information. Manual pages can be hard to read. Try *`any_command`* `--help` for short, easy to digest help on a command. If more info needed, have a look to the directory `/usr/doc`. To display manual page from a specific section, I may use something like in this example: `man 3 exit` (this displays an info on the command `exit` from section 3 of the manual pages).

`apropos` *`topic`*

Give me the list of the commands that have something to to do with my topic.

`help command`

Display brief info on a bash (shell) build-in command.

`ls`
List the content of the current directory. Under Linux, the command "dir" is an alias to ls. Many users have "ls" to be an alias to "ls --color".
`ls -al |more`
List the content of the current directory, all files (also those starting with a dot), and in a long form. Pipe the output through the "more" command, so that the display pauses after each screenful.
`cd` *`directory`*
Change directory. Using "cd" without the directory name will take you to your home directory. "cd -" will take you to your previous directory and is a convenient way to toggle between two directories. "cd .." will take you one directory up.
`cp` *`source destination`*
Copy files. E.g., `cp /home/stan/existing_file_name .` will copy a file to my current working directory. Use the "-r" option (for recursive) to copy the contents of whole directories, e.g. , `cp -r my_existing/dir/ ~` will copy a subdirectory under my current working directory to my home directory.
`mcopy` *`source destination`*
Copy a file from/to a DOS filesystem (no mounting necessary). E.g., `mcopy a:\autoexec.bat ~/junk` . See `man mtools` for related commands: mdir, mcd, mren, mmove, mdel, mmd, mrd, mformat ....
`mv` *`source destination`*
Move or rename files. The same command is used for moving and renaming files and directories.
`ln` *`source destination`*
Create a hard link called *destination* to the file called *source*. The link appears as a copy of the original files, but in reality only one copy of the file is kept, just two (or more) directory entries point to it. Any changes the file are automatically visible throughout. When one directory entry is removed, the other(s) stay(s) intact. The limitation of the hard links are: the files have to be on the same filesystem, hard links to directories or special files are impossible.
`ln -s` *`source destination`*
Create a symbolic (soft) link called "destination" to the file called "source". The symbolic link just specifies a path where to look for the file. In contradistinction to hard links, the source and destination don't not have to tbe on the same filesystem. In comparison to hard links, the drawback of symbolic links are: if the original file is removed, the link is "broken", symbolic links can also create circular references (like circular references in spreadsheets or databases, e.g., "a" points to "b" and "b" points back to "a").
`rm` *`files`*
Remove (delete) files. You must own the file in order to be able to remove it. On many systems, you will be asked or confirmation of deleation, if you don't want this, use the "-f" (=force) option, e.g., `rm -f *` will remove all files in my current working directory, no questions asked.
`mkdir` *`directory`*
Make a new directory.
`rmdir` *`directory`*
Remove an empty directory.
`rm -r` *`files`*
(recursive remove) Remove files, directories, and their subdirectories. Careful with this command as root--you can easily remove all files on the system with such a command executed on the top of your directory tree, and there is no undelete in Linux (yet). But if you really wanted to do it (reconsider), here is how (as root): `rm -rf /*`
`cat` *`filename`* `| more`
View the content of a text file called "filename", one page a time. The "|" is the "pipe" symbol

(on many American keyboards it shares the key with "\") The pipe makes the output stop after each screenful. For long files, it is sometimes convenient to use the commands head and tail that display just the beginning and the end of the file. If you happened to use "cat" a binary file and your terminal displays funny characters afterwards, you can restore it with the command "reset".

`less` *`filename`*

Scroll through a content of a text file. Press q when done. "Less" is roughly equivalent to "more" , the command you know from DOS, although very often "less" is more convenient than "more".

`pico` *`filename`*

Edit a text file using the simple and standard text editor called `pico`.

`pico -w` *`filename`*

Edit a text file, while disabling the long line wrap. Handy for editing configuration files, e.g. `/etc/fstab`.

`find / -name "`*`filename`*`"`

Find the file called "filename" on your filesystem starting the search from the root directory "/". The "filename" may contain wildcards (*,?).

`locate` *`filename`*

Find the file name of which contains the string "filename". Easier and faster than the previous command but depends on a database that normally rebuilds at night.

`./program_name`

Run an executable in the current directory, which is not on your PATH.

`touch` *`filename`*

Change the date/time stamp of the file *`filename`* to the current time. Create an empty file if the file does not exist.

`xinit`

Start a barebone X-windows server (without a windows manager).

`startx`

Start an X-windows server and the default windows manager. Works like typing "win" under DOS with Win3.1

`startx -- :1`

Start another X-windows session on the display 1 (the default is opened on display 0). You can have several GUI terminals running concurrently. Switch between them using <Ctrl><Alt><F7>, <Ctrl><Alt><F8>, etc.

`xterm`

(in X terminal) Run a simple X-windows terminal. Typing `exit` will close it. There are other, more advanced "virtual" terminals for X-windows. I like the popular ones: `konsole` and `kvt` (both come with kde) and `gnome-terminal` (comes with gnome). If you need something really fancy-looking, try `Eterm`.

`xboing`

(in X terminal). Very nice, old-fashioned game. Many small games/programs are probably installed on your system. I also like `xboard` (chess).

`shutdown -h now`

(as root) Shut down the system to a halt. Mostly used for a remote shutdown. Use <Ctrl><Alt><Del> for a shutdown at the console (which can be done by any user).

`halt`
`reboot`

(as root, two commands) Halt or reboot the machine. Used for remote shutdown, simpler to type than the previous command.

Network apps

`netscape`

(in X terminal) Run netscape (requires a separate Netscape installation). The current versions of

Netscape (4.x) are known to be big and buggy. They occasionally crash by vanishing (no other harm done). Also, when not connected to the network , Netscape likes to refuse to do anything (looks like it hanged)-it revives when you connect.

`netscape -display` *`host`*`:0.0`

(in X terminal) Run netscape on the current machine and direct the output to machine named "host" display 0 screen 0. Your current machine must have a permission to display on the machine "host" (typically given by executing the command `xhost` *`current_machine_name`* in the xterminal of the machine host. Other X-windows program can be run remotely the same way.

`lynx` *`file.html`*

View an html file or browse the net from the text mode.

`pine`

A good text-mode mail reader. Another good and standard one is `elm`. Your Netscape mail will read the mail from your Internet account. `pine` will let you read the "local" mail, e.g. the mail your son or a cron process sends to you from a computer on your home network. The command `mail` could also be used for reading/composing mail, but it would be inconvenient--it is meant to be used in scripts for automation.

`elm`

A good tex-mode mail reader. See the previous command.

`mutt`

A really basic but extremally useful and fast mail reader.

`mail`

A basic operating system tool for e-mail. Look at the previous commands for a better e-mail reader. `mail` is good if you wanted to send an e-mail from a shell script.

`licq`

(in X term) An icq "instant messaging" client. Another good one is `kxicq`. Older distributions don't have an icq client installed, you have to do download one and install it.

`talk` *`username1`*

Talk to another user currently logged on your machine (or use "`talk` *`username1@machinename`*" to talk to a user on a different computer) . To accept the invitation to the conversation, type the command "`talk` *`username2`*". If somebody is trying to talk to you and it disrupts your work, your may use the command "`mesg n`" to refuse accepting messages. You may want to use "`who`" or "`rwho`" to determine the users who are currently logged-in.

`mc`

Launch the "Midnight Commander" file manager (looks like "Norton Commander" for Linux).

`telnet` *`server`*

Connect to another machine using the TELNET protocol. Use a remote machine name or IP address. You will be prompted for your login name and password--you must have an account on the remote machine to login. Telnet will connect you to another machine and let you operate on it as if you were sitting at its keyboard (almost). Telnet is not very secure--everything you type goes in open text, even your password!

`rlogin` *`server`*

(=remote login) Connect to another machine. The login name/password from your current session is used; if it fails you are prompted for a password.

`rsh` *`server`*

(=remote shell) Yet another way to connect to a remote machine. The login name/password from your current session is used; if it fails you are prompted for a password.

`ftp` *`server`*

Ftp another machine. (There is also `ncftp` which adds extra features and `gftp` for GUI .) Ftp is good for copying files to/from a remote machine. Try user "anonymous" if you don't have an account on the remote server. After connection, use "?" to see the list of available ftp

commands. The essential ftp command are: `ls` (see the files on the remote system), `ASCII`, `binary` (set the file transfer mode to either text or binary, important that you select the proper one), `get` (copy a file from the remote system to the local system), `mget` (get many files at once), `put` (copy a file from the local system to the remote system), `mput` (put many files at once), `bye` (disconnect). For automation in a script, you may want to use `ncftpput` and `ncftpget`, for example:

```
ncftpput -u my_user_name -p my_password -a remote.host.domain remote_dir
*local.html
```

`minicom`

Minicom program (looks like "Procomm for Linux").

File (de)compression

`tar -zxvf` *`filename.tar.gz`*

(=tape archiver) Untar a tarred and compressed tarball (*.tar.gz or *.tgz) that you downloaded from the Internet.

`tar -xvf` *`filename.tar`*

Untar a tarred but uncompressed tarball (*.tar).

`gunzip` *`filename.gz`*

Decompress a zipped file (*.gz" or *.z). Use gzip (also `zip` or `compress`) if you wanted to compress files to this file format.

`bunzip2` *`filename.bz2`*

(=big unzip) Decompress a file (*.bz2) zipped with bzip2 compression utility. Used for big files.

`unzip` *`filename.zip`*

Decompress a file (*.zip) zipped with a compression utility compatible with PKZIP for DOS.

`unarj e` *`filename.arj`*

Extract the content of an *.arj archive.

`uudecode -o` *`outputfile filename`*

Decode a file encoded with `uuencode`. uu-encoded files are typically used for transfer of non-text files in e-mail (uuencode transforms any file into an ASCII file).

7.4 Process control

`ps`

(=print status) Display the list of currently running processes with their process IDs (PID) numbers. Use `ps axu` to see all processes currently running on your system (also those of other users or without a controlling terminal), each with the name of the owner. Use "top" to keep listing the processes currently running.

`fg` *`PID`*

Bring a background or stopped process to the foreground.

`bg` *`PID`*

Send the process to the background. Opposite to fg. The same can be accomplished with <Ctrl>z. If you have stopped jobs, you have to type `exit` twice in row to log out.

`any_command``&`

Run any command in the background (the symbol "&" means "run the proceeding command in the background").

`batch` *`any_command`*

Run any command (usually one that is going to take more time) when the system load is low. I can logout, and the process will keep running.

`at 17:00`

Execute a command at a specified time. You will be prompted for the command(s) to run, until you press <Ctrl>d.

`kill` *`PID`*

Force a process shutdown. First determine the PID of the process to kill using ps.

`killall` *`program_name`*
Kill program(s) by name.
`xkill`
(in an xwindow terminal) Kill a GUI-based program with mouse. (Point with your mouse cursor at the window of the process you want to kill and click.)
`lpc`
(as root) Check and control the printer(s). Type "?" to see the list of available commands.
`lpq`
Show the content of the printer queue. Under KDE (X-Windows), you may use GUI-based "Printer Queue" available from "K"menu-Utilities.
`lprm` *`job_number`*
Remove a printing job "job_number" from the queue.
`nice` *`program_name`*
Run *program_name* adjusting its priority. Since the priority is not specified in this example, it will be adjusted by 10 (the process will run slower), from the default value (usually 0). The lower the number (of "niceness" to other users on the system), the higher the priority. The priority value may be in the range -20 to 19. Only root may specify negative values. Use "top" to display the priorities of the running processes.
`renice -1` *`PID`*
(as root) Change the priority of a running process to -1. Normal users can only adjust processes they own, and only up from the current value (make them run slower).
`<Ctrl>c`, `<Ctrl>z`, `<Ctrl>s`, and `<Ctrl>q` also belong to this chapter but they were described previously. In short they mean: stop the current command, send the current command to the background, stop the data transfer, resume the data transfer.

7.5 Basic administration commands

`printtool`
(as root in X-terminal) Configuration tool for your printer(s). Settings go to the file `/etc/printcap`.
`setup`
(as root) Configure mouse, soundcard, keyboard, X-windows, system services. There are many distibution-specific configuration utilities, `setup` is the default on RedHat. Mandrake 7.0 offers very nice `DrakConf` .
`linuxconfig`
(as root, either in text or graphical mode). You can access and change hundreds of setting from it. Very powerful--don't change too many things at the same time, and be careful with changing entries you don't understand.
`xvidtune`
(in X-terminal). Adjust the settings of the graphical display for all resolutions so as to eliminate black bands, shift the display right/left/up/down, etc. (First use the knobs on your monitor to fit your text mode correctly on the screen.) To make the changes permanent, display the frequencies on the screen and transfer them to the setup file `/etc/X11/XF86Config`.
`alias ls="ls --color=tty"`
Create an alias for the command "ls" to enhance its format with color. In this example, the alias is also called "ls" and the "color" option is only envoke when the output is done to a terminal (not to files). Put the alias into the file `/etc/bashrc` if you would like the alias to be always accessible to all users on the system. Type "`alias`" alone to see the list of aliases on your system.

`adduser user_name`
Create a new account (you must be root). E.g., `adduser barbara` Don't forget to set up the password for the new user in the next step. The user home directory is `/home/user_name`.
`useradd user_name`
The same as the command " `adduser user_name` ".
`userdel user_name`
Remove an account (you must be a root). The user's home directory and the undelivered mail must be dealt with separately (manually because you have to decide what to do with the files).
`groupadd group_name`
Create a new group on your system. Non-essential but can be handy even on a home machine with a small number of users.
`passwd`
Change the password on your current account. If you are root, you can change the password for any user using: `passwd user_name`
`chmod perm filename`
(=change mode) Change the file access permission for the files you own (unless you are root in which case you can change any file). You can make a file accessible in three modes: read (r), write (w), execute (x) to three classes of users: owner (u), members of the same group as the owner (g), others on the system (o). Check the current access permissions using:
`ls -l filename`
If the file is accessible to all users in all modes it will show:
`rwxrwxrwx`
The first triplet shows the file permission for the owner of the file, the second for his/her group, the third for others. A "no" permission is shown as "-".
E.g., this command will **add** the permission to read the file "junk" to all (=user+group+others):
`chmod a+r junk`
This command will remove the permission to execute the file junk from others:
`chmod o-x junk`
Also try here for more info.
You can set the default file permissions for the news files that you create using the command `umask` (see `man umask`).
`chown new_ownername filename`
`chgrp new_groupname filename`
Change the file owner and group. You should use these two commands after you copy a file for use by somebody else.
`su`
(=substitute user id) Assume the superuser (=root) identity (you will be prompted for the password). Type "exit" to return you to your previous login. Don't habitually work on your machine as root. The root account is for administration and the su command is to ease your access to the administration account when you require it. You can also use "su" to assume any other user identity, e.g. `su barbara` will make me "barbara" (password required unless I am a superuser).
`kernelcfg`
(as root in X terminal). GUI to to add/remove kernel modules. You can do the same from the command line using the command "`insmod`", but "`insmode`" is less "newbie-friendly".
`lsmod`
List currently loaded kernel modules. A module is like a device driver--it provides operating system kernel support for a particular piece of hardware or feature.
`modprobe -l |more`
List all the modules available for your kernel. The available modules are determined by how your Linux kernel was compliled. Every possible module/feature can be compiled on linux as

either "hard wired" (fast, non-removable), "module" (maybe slower, but loaded/removable on demand), or "no" (no support for this feature at all).

```
insmod parport
insmod ppa
```

(as root) Insert modules into the kernel (a module is roughly an equivalent of a DOS device driver). This example shows how to insert the modules for support of the external parallel port zip drive (it appears to be a problem to get the external zip drive to work in any other way under RH6.0).

`rmmod` *`module_name`*

(as root, not essential). Remove the module *module_name* from the kernel.

```
setserial /dev/cua0 port 0x03f8 irq 4
```

(as root) Set a serial port to a non-standard setting. The example here shows the standard setting for the first serial port (cua0 or ttyS0). The standard PC settings for the second serial port (cua1or ttyS1) are: address of i/o port 0x02f8, irq 3. The third serial port (cua2 or ttyS2): 0x03e8, irq 4. The forth serial port (cua3 or ttyS3): 0x02e8, irq 3. Add your setting to `/etc/rc.d/rc.local` if you want it to be set at the boot time. See man setserial for good a overview.

```
fdisk
```

(as root) Linux hard drive partitioning utility (DOS has a utility with the same name).

```
cd /usr/src/linux-2.0.36
make xconfig
```

(as root in X terminal). Nice GUI front-end for configuration of the kernel options in preparation for compilation of your customized kernel. (The directory name contains the version of your Linux kernel so you may need to modify the directory name if your Linux kernel version is different than 2.0.36 used in this example. You also need the "Tk" interpreter and the kernel source code installed.) The alternatives to "make xconfig" are: "make config" (runs a scripts that asks you questions in the text mode) and "make menuconfig" (runs a text-based menu-driven configuration utility). Try: `less /usr/doc/HOWTO/Kernel-HOWTO` for more information.

After the configuration, you may choose to proceed with kernel compilation of the new kernel by issuing the following commands:

```
make dep
make zImage
```

The last command will take some time to complete (maybe 0.5 h, depending on your hardware). It produces the file "zImage", which is your new Linux kernel. Next:

```
make modules
make modules_install
```

Read: /usr/doc/HOWTO/Kernel-HOWTO for information on how to install the new kernel. You will probably also find it useful to read "man depmode". Configuration, compilation and installation of a new kernel is not difficult but it CAN lead to problems if you don't know what you are doing.

Compilation of a kernel is a good way to test your hardware, because it involves a massive amount of computing. If your hardware is "flaky", you will most likely receive the "signal 11" error (read the beatiful `/usr/doc/FAQ/txt/GCC-SIG11-FAQ`). **See this for details on kernel upgrade.**

```
depmod -a
```

(as root) Build the module dependency table for the kernel. This can, for example, be useful after installing and booting a new kernel. Use "`modprobe -a`" to load the modules.

```
ldconfig
```

(as root) Re-create the bindings and the cache for the loader of dynamic libraries ("ld"). You may want to run ldconfig after an installation of new dynamically linked libraries on your system. (It

is also re-run every time you boot the computer, so if you reboot you don't have to run it manually.)

```
mknod /dev/fd0 b 2 0
```

(=make node, as root) Create a device file. This example shows how to create a device file associated with your first floppy drive and could be useful if you happened to accidentally erase it. The options are: b=block mode device (c=character mode device, p=FIFO device, u=unbuffered character mode device). The two integers specify the major and the minor device number.

```
fdformat /dev/fd0H1440
mkfs -c -t ext2
```

(=floppy disk format, two commands, as root) Perform a low-level formatting of a floppy in the first floppy drive (/dev/fd0), high density (1440 kB). Then make a Linux filesystem (-t ext2), checking/marking bad blocks (-c). Making the files system is an equivalent to the high-level format.

```
badblocks /dev/fd01440 1440
```

(as root) Check a high-density floppy for bad blocks and display the results on the screen. The parameter "1440" specifies that 1440 blocks are to be checked. This command does not modify the floppy.

```
fsck -t ext2 /dev/hda2
```

(=file system check, as root) Check and repair a filesystem. The example uses the partition hda2, filesystem type ext2.

```
dd if=/dev/fd0H1440 of=floppy_image
dd if=floppy_image of=/dev/fd0H1440
```

(two commands, dd="data duplicator") Create an image of a floppy to the file called "floppy_image" in the current directory. Then copy `floppy_image` (file) to another floppy disk. Works like DOS "DISKCOPY".

Program installation

`rpm -ivh` *`filename.rpm`*

(=RedhatPackageManager, install, verbose, hashes displayed to show progress, as root.) Install a content of RedHat rpm package(s) and print info on what happened. Keep reading if you prefer a GUI installation.

`rpm -qpi` *`filename.rpm`*

(=RedhatPackageManager, query, package, list.) Read the info on the content of a yet uninstalled package *filename.rpm*.

`rpm -qpl` *`filename.rpm`*

(=RedhatPackageManager, query, package, information.) List the files contained in a yet uninstalled package *filename.rpm*.

`rpm -qf` *`filename`*

(=RedhatPackageManager, query, file.) Find out the name of the *.rpm package to which the file *filename* (on your hardrive) belongs.

`rpm -e` *`packagename`*

(=RedhatPackageManager, erase=uninstall.) Uninstall a package *pagckagename*. *Packagname* is the same as the beginning of the *.rpm package file but without the dash and version number.

```
kpackage
gnorpm
glint
```

(in X terminal, as root if you want to be able to install packages) GUI fronts to the Red Hat Package Manager (rpm). "glint" comes with RH5.2, "gnorpm" with RH6.0, "kpackage" comes with RH6.1 or must be installed separately but is the best of the three. Use any of them to view which software packages are installed on your system and the what not-yet-installed packages

are available on your RedHat CD, display the info about the packages, and install them if you want (installation must be done as root).

Accessing drives/partitions

`mount`

See here for details on mounting drives. Examples are shown in the next commands.

`mount -t auto /dev/fd0 /mnt/floppy`

(as root) Mount the floppy. The directory `/mnt/floppy` must exist, be empty and NOT be your current directory.

`mount -t auto /dev/cdrom /mnt/cdrom`

(as root) Mount the CD. You may need to create/modify the `/dev/cdrom` file depending where your CDROM is. The directory `/mnt/cdrom` must exist, be empty and NOT be your current directory.

`mount /mnt/floppy`

(as user or root) Mount a floppy as user. The file `/etc/fstab` must be set up to do this. The directory `/mnt/floppy` must not be your current directory.

`mount /mnt/cdrom`

(as user or root) Mount a CD as user. The file `/etc/fstab` must be set up to do this. The directory `/mnt/cdrom` must not be your current directory.

`umount /mnt/floppy`

Unmount the floppy. The directory `/mnt/floppy` must not be your (or anybody else's) current working directory. Depending on your setup, you might not be able to unmount a drive that you didn't mount.

7.6 Network administration tools

`netconf`

(as root) A very good menu-driven setup of your network.

`ping`*`machine_name`*

Check if you can contact another machine (give the machine's name or IP), press <Ctrl>C when done (it keeps going).

`route -n`

Show the kernel routing table.

`nslookup` *`host_to_find`*

Query your default domain name server (DNS) for an Internet name (or IP number) *host_to_find*. This way you can check if your DNS works. You can also find out the name of the host of which you only know the IP number.

`traceroute` *`host_to_trace`*

Have a look how you messages trave to *`host_to_trace`* (which is either a host name or IP number).

`ipfwadm -F -p m`

(for RH5.2, seen next command for RH6.0) Set up the firewall IP forwarding policy to masquerading. (Not very secure but simple.) Purpose: all computers from your home network will appear to the outside world as one very busy machine and, for example, you will be allowed to browse the Internet from all computers at once.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ipfwadm-wrapper -F -p deny
ipfwadm-wrapper -F -a m -S xxx.xxx.xxx.0/24 -D 0.0.0.0/0
```

(three commands, RH6.0). Does the same as the previous command. Substitute the "x"s with digits of your class "C" IP address that you assigned to your home network. See here for more details. In RH6.1, masquarading seems broken to me--I think I will install Mandrake Linux:).

`ifconfig`
(as root) Display info on the network interfaces currently active (ethernet, ppp, etc). Your first ethernet should show up as eth0, second as eth1, etc, first ppp over modem as ppp0, second as ppp1, etc. The "lo" is the "loopback only" interface which should be always active. Use the options (see `ifconfig --help`) to configure the interfaces.
`ifup` *`interface_name`*
(`/sbin/ifup` to it run as a user) Startup a network interface. E.g.:
```
ifup eth0
ifup ppp0
```
Users can start up or shutdown the ppp interface only when the right permission was checked during the ppp setup (using `netconf`). To start a ppp interface (dial-up connection), I normally use kppp available under kde menu "internet".
`ifdown` *`interface_name`*
(`/sbin/ifdown` to run it as a user). Shut down the network interface. E.g.: `ifdown ppp0` Also, see the previous command.
`netstat | more`
Displays a lot (too much?) information on the status of your network.

Music-related commands

`cdplay play 1`
Play the first track from a audio CD.
`eject`
Get a free coffee cup holder :))). (Eject the CD ROM tray).
`play` *`my_file.wav`*
Play a wave file.
`mpg123` *`my_file.mp3`*
Play an mp3 file.
`mpg123 -w` *`my_file.wav my_file.mp3`*
Create a wave audio file from an mp3 audio file.
`knapster`
(in X terminal) Start the program to downolad mp3 files that other users of napster have displayed for downloading. Really cool!
`cdparanoia -B "1-"`
(CD ripper) Read the contents of an audio CD and save it into wavefiles in the current directories, one track per wavefile. The "1-"
means "from track 1 to the last". -B forces putting each track into a separate file.
`playmidi` *`my_file.mid`*
Play a midi file. `playmidi -r` *`my_file.mid`* will display text mode effects on the screen.
`sox`
(argument not given here) Convert from almost any audio file format to another (but not mp3s). See `man sox`.

Graphics-related commands

`kghostview` *`my_file.ps`*
Display a postscript file on screen. I can also use the older-looking `ghostview` or `gv` for the same end effect.
`ps2pdf my_file.ps my_file.pdf`
Make a pdf (Adobe portable document format) file from a postscript file.
`gimp`
(in X terminal) A humble looking but very powerful image processor. Takes some learning to use, but it is great for artists, there is almost nothing you can't do with gimp. Use your mouse

right button to get local menus, and learn how to use layers. Save your file in the native gimp file format *.xcf (to preserve layers) and only then flatten it and save as png (or whatever). There is a large user manual /usr/

`gphoto`

(in X terminal) Powerful photo editor.

`giftopnm` *`my_file.giff > my_file.pnm`*

`pnmtopng` *`my_file.pnm > my_file.png`*

Convert the propriatory giff graphics into a raw, portable pnm file. Then convert the pnm into a png file, which is a newer and better standard for Internet pictures (better technically plus there is no danger of being sued by the owner of giff patents).

Services in Fedora Core 5

The following is a brief explanation on usage and recommendations for some of the different services packaged with Fedora Core 5.
(from http://www.mjmwired.net/resources/mjm-services-fc5.html)

Understanding Services

Please read the guide on managing services in Fedora. This includes an explanation for services/daemons, runlevels and various tools available to manage your services.

To control services either use `chkconfig` or `ntsysv` if you are using the command line, or use `system-config-services` in the GUI. Gnome users: *System* > *Administration* > *Server Settings* > *Services*.

Individual Services

The following is a brief explanation on usage and recommendations for services packaged with Fedora Core 5. This is not an *exhaustive* list. **Be careful, do not disable things that you're not sure if need or if you do not understand or know what they are.**

DO NOT DISABLE THE FOLLOWING (unless you know what you are doing).
`acpid`, `haldaemon`, `messagebus`, `klogd`, `network`, `syslogd`
Make sure to apply your changes to runlevel 5 *AND* 3.

NetworkManager, NetworkManagerDispatcher

NetworkManager is daemon meant to automate switching between network connections. Many laptop users who switch between *Wireless WiFi* connections and *Ethernet* connections may find this useful. Most stationary computers should have this **disabled**. Some DHCP users *may require* this.

acpid

Advanced Configuration and Power Interface daemon which controls and allows interfacing to power management and certain input devices. It is recommended to be **enabled** for all laptops, and most desktops. Some servers may not require acpi. Common things supported are the *"Power Switch", "Battery Monitor", "Laptop Lid Switch", "Laptop Display Brightness", "Hibernate", "Suspend"*, etc.

anacron, atd, cron

These are schedulers with each having slightly different purposes. It is recommended you keep the general purpose scheduler `cron` **enabled**, especially if you keep your computer running for long periods of time. If you are *running a server* look into which schedulers you require. Most likely `atd` and `anacron` should be **disabled** for desktops/laptops. Please note that some sheduled tasks such as cleaning `/tmp` or `/var` may require `anacron`.

apmd

Is used by some laptops and older hardware. If your computer supports `acpi`, then `apmd` should probably be **disabled**.

auditd

This saves audit records generated by the kernel. Not entirely sure how this information is used, however it is useful for diagnosing issues with `SELinux`. For now I have this **enabled**. This is *optional*, however it may useful for servers or machines with multiple users and *highly recommended* for `SELinux` users.

autofs

This mounts removable disks (such as USB harddrives) on demand. It is recommended to keep this **enabled** if you use removable media.

avahi-daemon, avahi-dnsconfd

Avahi is an implementation of zeroconf and is useful for detecting devices and services on local network without a DNS server. This is also the same as *mDNS*. Most likely this is unnecessary unless you have compatible devices/services. I have this **disabled**.

bluetooth, hcid, hidd, sdpd

Bluetooth is for portable local wireless devices (*NOT* wifi,802.11). Some laptops come with bluetooth support. There are bluetooth mice, headsets and cell phone accessories. Most people do not have bluetooth support or devices, and should **disable** this. Other services with bluetooth: hcid manages all devices, hidd provides support for input devices (keyboard, mouse).

cpuspeed

This throttles your CPU runtime frequency to save power. Many modern laptop CPU's support this feature and now some desktops also support this. Most people should **enable only if** they are users of *Pentium-M, Centrino, AMD PowerNow, Transmetta, Intel SpeedStep, Athlon-64* hardware.

cron

See above.

cupsd, cups-config-daemon

Used for printing. These should be **enabled only if** you have CUPS compatible printer that works in Fedora.

dc_client, dc_server

Distcache is for *distributed session caching*. It is primarily for SSL/TLS servers. Apache can use this. Most desktop users should have these **disabled**.

dhcdbd

This basically an interface for the DBUS system to control DHCP on your computer. It can be left to the default **disabled** state.

diskdump, netdump

Diskdump is a mechanism to help debug kernel crashes. It save a "dump" which can be later analyzed. Netdump does something similar over the network. Unless you are diagnosing a problem, these should be left as **disabled**.

firstboot

This service is specific to Fedora's installation process meant to perform certain tasks that should only be executed once upon booting after installation. Even though it verifies it has been run before, it can be **disabled**.

gpm

This is the console mouse pointer (no graphics). If you do not use the text console (CTRL-ALT-F1,F2..) then **disable** this. However I leave this enabled for runlevel 3 and disabled for runlevel 5.

hidd

See bluetooth.

hplip, hpiod, hpssd

HPLIP is a service to support HP printers in Linux, including *Inkjet, DeskJet, OfficeJet, Photosmart, Business Inkjet and some LaserJet printers*. This supported by HP through HP Linux Printing Project. HPLIP should be **enabled only if** you have a supported compatible printer.

iptables

This is the standard Linux software firewall. This is **required** if you are *directly connected to internet (cable, DSL, T1)*. It is not required if you use a hardware firewall (D-Link, Netgear, Linksys, etc) but it is **highly recommended**.

irqbalance

This service is to increase performance across processors on a multiprocessor system. Since most people do not have multiple processors, it should be **disabled**. However I do not know how it affects *multi-core CPU's* or *hyperthreaded CPU's* (?). There should be no problems on single CPU systems that do not use this.

isdn

This is another form of internet connect service/hardware. Unless you have an ISDN modem, **disable** this.

kudzu

This runs the hardware probe, and optionally configures changed hardware. If you swap hardware or need to detect/re-detect hardware this can be left **enabled**. However most desktop or servers can disable this and run it only when necessary.

lm_sensors

This monitors motherboard sensor values or specific hardware (commonly used with laptops). It is useful for watching realtime values for PC health, etc. This is also popular with GKrellM users. More information on lm_sensors homepage. It is recommended to **disable** this unless you have a need.

mdmonitor

Is useful for monitoring Software RAID or LVM information. It is not a critical service and be **disabled**.

messagebus
: This is an IPC (Interprocess Communication) service for Linux. Specifically this communicates with `dbus`, a critical component. It is highly recommended to leave this **enabled**.

netdump
: See diskdump.

netplugd
: Netplugd can monitor network interfaces and executes commands when their state changes. This can be left to default **disabled**.

netfs
: This is used for automatic mounting of any shared network file space such as NFS, Samba, etc on bootup. Useful if you connect to another server or filesharing on your *local* network. Most single desktop/laptop users should have this **disabled**.

nfs, nfslock
: This the standard network file sharing for Unix/Linux/BSD style operating systems. Unless you require to share data in this manner, **disable** this.

ntpd
: This automatically updates the system time from the internet. Mentioned in the installation process. If you have an active ("always-on") internet connection it is recommended you **enable** this, but it is *not required*.

portmap
: This is complementary service to NFS (file sharing) and/or NIS (authentication). Unless you use those services you should **disable** this.

readahead, readahead_early
: This services is to improve startup performance by preloading certain applications into memory. If you wish to startup faster leave this **enabled**.

rpcgssd, rpcidmapd, rpcsvcgssd
: Used for NFS v4. Unless you require or use NFS v4, these should be **disabled**.

sendmail
: Unless you run a server or you like to transfer or support a locally shared `IMAP` or `POP3` service, most people do NOT need a mail transport agent. If you check your mail on the web (hotmail/yahoo/gmail) or you use a mail program such as Thunderbird, Kmail, Evolution, etc. then you should **disable** this.

smartd
: The SMART Disk Monitoring Daemon can be used to monitor and predict disk failure or problems on hard disk that support this. Most desktop users may not need this unless there is possible problems, but is it recommend to be left **enabled** (especially for servers).

smb
: The SAMBA daemon is *required to share files from Linux to Windows*. This should be **enabled only if** you have windows computers that require file access to Linux. There is information on configuring Samba for FC5.

sshd

SSH allows other users to log into or run applications on your computer from another computer on your network or remotely. *This is a potential security issue.* This is not needed if you have no other computers or no need to login from a remote location (work, school, etc.). Most likely this should be **disabled**.

xinetd

This is a special service. It can launch multiple services based on a request to a specific port. For example: `telnet` is typically connected to port 23. If there is a request for telnet access that `xinetd` detects on port 23, then only will the telnet daemon be executed. For convenience this can be left to **enabled**. Run `system-config-services` and go to *On Demand Services* -or- run `chkconfig --list` and look for the `xinetd` output to show which services are connected to `xinetd`.