Linux Questions and Answers

A Linux White Paper

Preface

For someone new (and even not so new) to Linux®, there are hundreds of questions, problems and concerns that arise during the learning process, especially for someone coming from the world of Microsoft® Windows®. Other Linux white papers deal with individual topics, often in lengthy fashion. This paper addresses a number of miscellaneous questions, both frequently- and rarely-asked, grouped by topic. The focus will be on helping Windows users make the transition to Linux, but those converting from other operating systems should find useful information here as well.

Note: Because of the differences between Linux "distributions" from various vendors, some of the information below may apply to one distribution but not another. This will be pointed out where known; however, with all the available distributions it is impossible to identify all such situations here. If the problem is that a command or program described doesn't exist in a given distribution, it is generally possible to download a copy of that program from a Web site, if needed. On the other hand, there may be a functionally similar utility already provided with that distribution. To find out, consult the user's manual or contact the distribution vendor. If all else fails, there is a list of Web sites in the *Miscellaneous* section, below, that may be of assistance.

For general terminology questions, please refer to the white paper entitled *A Brief Linux Glossary for Windows Users*, available from the same sources as this paper.

Special Note: If you are reading this document online with Adobe Acrobat Reader, simply click on the Web addresses highlighted in blue to go to those Web sites via your Web browser.

Contents

Installation

These questions all deal with some aspect of installing the operating system.

Q. During installation Linux creates a swap space partition. Why do I need this and how is it different from a Windows swap file?

Like Windows, Linux uses a certain amount of space for holding programs temporarily, when there is not enough available RAM (random access memory) to hold all the programs that are running concurrently. Generally, the least recently used program (or part of a program) is copied from memory to a file on your hard drive until it is needed again, at which time the current least recently used program is swapped out in its place and the first program is loaded back into memory. (This is an over-simplified explanation; there is much more to it, but this will do for this question.) This file is called a swap file in Windows or OS/2 and "swap space" in Linux, but in either case it is a form of data file that is read from and written to off and on as long as your system is running.

Windows puts the swap file (a hidden system file with different names for different versions of Windows) in the bootable data partition by default. OS/2 does the same, but by changing the CONFIG.SYS file a user can put the swap file in any directory in any partition on any drive they like. Linux, by default, requires a special swap partition in which to store the swap file. (Actually, Linux does allow swap files to be put in data partitions, with caveats—see below for more on this.)

Q. During installation, I have a choice of creating swap space in a Primary or an Extended partition. Which should I choose?

Either will work. There is no technical advantage to doing it one way or the other, but there are only a limited number of primary partitions that can be created, so if you plan on installing more than one operating system you might want to put the swap space in an Extended partition and save a Primary partition for other uses.

Note: If you have Microsoft Windows installed, it may not boot if you have more than one primary partition installed on the bootable disk drive (what Windows sees as C:). In this case, Linux would have to be installed in an Extended partition, or in a Primary partition on a second hard disk drive. For more information, refer to the lilo (LInux LOader) documentation on a Linux system, using the command: `man lilo` (no, ***man*** is *not* a politically incorrect command—it is merely short for manual). If you do not yet have a Linux system set up where you can read the lilo documentation there are a couple of other options available: 1) If you are installing from a CD set, look for a directory or separate CD (as in Red Hat 6.2) containing documentation, possibly in HTML format. 2) Visit Linux Web sites, such as *http://linux.ctyme.com* and *http://www.linuxdoc.org*, for online documentation.

Q. How large a swap partition should I create?

Although it can be smaller, for best results the partition size should be *at least* equal to the amount of memory installed in the system—preferably *twice* the amount of physical memory. In other words, if you have 64MB of RAM installed use double that amount for your swap space partition (64MB * 2 = 128MB).

If you need more than 128MB of swap space, but you are using an older distribution that does *not* support a swap file larger than 128MB, there are ways around the 128MB swap file limitation: 1) Linux allows more than one swap partition on a hard drive, and 2) Linux also allows swap files to be put in *data* partitions (i.e., in the same partitions as your programs and data files).

If you create a swap partition and then later decide that it is not large enough for your needs, you can delete the original one and replace it with a larger swap partition if you have the contiguous space to do so—in other words, if there is no other partition immediately following the original swap partition. If there is insufficient space to create the larger swap partition, you can still create a second partition elsewhere on that drive. They do not have to be back-to-back. (Or you can use a disk partitioning program, such as Partition Magic, to reallocate partition space on the fly between existing partitions, without losing data.) Linux supports up to eight swap partitions.

Alternatively, if you have multiple physical disk drives, splitting your swap space between the disk drives *may* improve system performance, because both drives can be used concurrently for swapping. (This performance increase is more apparent on a system with SCSI drives, because SCSI devices run in parallel—at the same time—while standard IDE devices run in serial—they alternate. Even so, on a system with multiple IDE drives, it can be faster to switch back and forth between physical drives (an electronic operation) than to shuttle the read/write heads of one drive all over the disk, dividing their time between partitions on the same drive.) Also, drives added to the system later may be faster than the original drive that came with the system. Using these newer drives will give you faster disk read/writes when swapping.

Dividing the swap space across multiple drives may also increase system reliability. Like any device, hard drives have a finite life expectancy, so spreading the workload between multiple drives means that the primary drive doesn't have to work as hard, which in turn *may* (no guarantees, though) extend its useful life. (For instructions on how to do this, refer to the question, *How can I create multiple a swap partitions on one or more drives?*, below.)

Q. How can I speed up performance by using multiple swap partitions?

If your system (especially a server) has multiple drives set up as a RAID 0 array (data striping), it would automatically spread the data across all the drives in parallel, greatly increasing the speed of disk writes versus a non-RAID setup, where the swap partition is on only one drive. Of course, not everyone can afford a RAID setup; fortunately, there is an alternative that offers many of the same performance benefits in a multidisk environment, at least as far as swap space performance is concerned.

As it happens, there is a way to "parallelize" swap file read/writes through the use of the priority setting in the ***/etc/fstab*** file. (***Note:*** This works best with multiple SCSI drives and/or controllers. Two IDE drives on the same controller will see little or no performance boost from this technique, but multiple IDE drives on multiple controllers may benefit somewhat.)

Use an appropriate tool (such as *kfstab* [available from kfstab.purespace.de/kfstab], which knows the exact layout of the file that ***fstab*** requires) to open the file **/etc/fstab**. This file lists the partitions used by Linux, including swap space. (***Note:*** It *is* possible to use a simple text editor such as *vi*, but if you do not get the columns lined up just right, or if you use tabs instead of spaces or vice versa, you can corrupt your system; so if you elect to go this route, rather than using a tool specifically designed for editing ***fstab***, such as *kfstab*, be sure to make a backup copy of ***fstab*** first, so that you can recover if your system crashes.)

When you open **fstab**, if you have multiple SCSI drives with a swap partition on each, you might see something like this (without the headings):

<partition name>	***<mount point>***	***<partition type>***	***<mount options>***	***<backup dump>***	***<pass>***
/dev/sda2	none	swap	sw	0	0
/dev/sdb2	none	swap	sw	0	0
/dev/sdc2	none	swap	sw	0	0
/dev/sdd2	none	swap	sw	0	0

Under normal circumstances, Linux would use the swap partition **/dev/sda2** first, then **/dev/sdb2**, and so on, one at a time, until it had enough space to meet its current swapping needs.

On the other hand, if you change the settings to:

```
/dev/sda2     none          swap        sw,pri=3     0     0
/dev/sdb2     none          swap        sw,pri=3     0     0
/dev/sdc2     none          swap        sw,pri=3     0     0
/dev/sdd2     none          swap        sw,pri=1     0     0
```

you will be assigning the first three partitions the *same priority level*—a higher one than the fourth partition. (Swap partitions are used in order from highest priority to lowest—where a priority level of 32767 is the highest and 0 is the lowest.) This has the effect of forcing Linux to write to the first three partitions in parallel, greatly increasing the read/write throughput of the swap space. The fourth partition would be used only if the first three fill up.

In addition to parallelizing the swapping, this technique also allows you to prioritize access to the fastest drives first. Therefore, you can set up an emergency swap partition on an old, slow drive, using low priority, so that it will be used only when all high-speed swap space is exhausted.

Q. How can I create multiple swap partitions on one or more drives?

To create multiple swap partitions:

1. First, use the: **su –** command to enter *superuser* mode. (If you don't do this you will be unable to use the ***fdisk*** command.)
2. Enter the command: `fdisk /dev/hda` if your bootable hard drive is IDE, or use: `fdisk /dev/sda` if your bootable hard drive is SCSI. (If you are not sure which type of disk drives your system uses and you guess wrong, the worst that can happen is that Linux will issue an error message saying *"Unable to open xda."* You can't hurt anything.) This is a text-mode program, so it is not pretty. Where it says, *"The number of cylinders for this disk is set to xxxx"* note the number displayed for *xxxx* (for example, 1024).
3. At the prompt *"Command (m for help)."*
    a. Type: p (and press Enter) to see the current partitions. Check the number shown under the End column for the last cylinder listed. If it is less than the number of cylinders from Step 2, above, there is room for another partition. (It is only necessary to create another partition on the same drive if your Linux distribution doesn't allow swap partitions larger than 128MB and you want additional swap partitions.) The difference between the two numbers determines how much space is available for use.
    b. (If there is already an existing swap partition, but it is not large enough, use the: d (delete) command to remove it; then create a new, larger one.)
    c. Enter: n to create a new partition.
    d. If you are creating a Primary partition:
        1) Enter: p to create the partition. When prompted for the partition number, use the next higher number than the highest existing partition number. (In other words, if the last device description on the left shows: **/dev/hda1**, use **2** as the new partition number.)

2) Type in the number suggested for the first cylinder (which should be one higher than the ending cylinder for the preceding partition, or press **Enter** to take the default.

3) Enter the partition size in megabytes, such as: `+128M`. (Be sure to include both the **+** and the **M** or you will get a *"Value out of range"* error.)

4) Enter: `p` again to verify that the new partition was created. It should show an Id type of *83* and a System type of *Linux*.

5) Enter: `t` to change the partition type, then enter the partition number you used in Step 1), above. Finally, when prompted for the "Hex code," enter the swap partition type code: `82`. If everything worked correctly, you should see a message like *"Changed system type of partition [x] to 82 (Linux swap)."* (where [x] is the partition number from Step 1).

e. If you are creating an Extended partition:

1) Enter: `e` to create the partition. When prompted for the partition number, use the next higher number than the highest existing partition number. (In other words, if the last Device description on the left shows: **/dev/hda1**, use **2** as the new partition number.)

2) Type in the number suggested for the first cylinder (which should be one higher than the ending cylinder for the preceding partition, or press Enter to take the default.

3) Press **Enter** to reserve all the available space.

4) Enter: `p` again to verify that the new Extended partition was created. It should show an Id type of *5* and a System type of *Extended*.

5) Enter: `t` to change the partition type, then enter the partition number you used in Step 1), above. Then, when prompted for the "Hex code", enter the Linux Extended partition type code: `85`. If everything worked correctly, you should see a message like *"Changed system type of partition [x] to 85 (Linux extended)."* (where [x] is the partition number from Step 2). ***Note:*** This step is not strictly necessary. You can create Linux swap partitions in a "regular" extended partition, but changing the partition type to Linux Extended takes only a few seconds and it clarifies the partition structure for future ***fdisk*** users on this system.

6) Enter: `n` to create a new partition. When prompted for the first cylinder number, press Enter. Enter the desired swap partition size in megabytes, such as: `+128M`. Enter: `p` again to verify that a type 83 partition was created with the same starting block as the *Linux extended* partition you created in Step 1).

7) Enter: `t` to change the partition type, then enter the number for the Linux extended partition you just created. When prompted for the "Hex code," enter the swap partition type code: `82`. If everything worked correctly, you should see the message *"Changed system type of partition [x] to 82 (Linux swap).*

8) If you did not use up all of the available partition space in Step 6), create one or more additional partitions to hold data, but this time do *not* change the partition default type of 83.

f. To write your changes to the hard disk and exit the program, type: `w` and press **Enter**.

4. When you are returned to the *Hard Disk Selection* panel, select *No further hard disk changes* and press **Enter**.

5. Repeat steps 2-4 for each additional hard drive that you wish to add a swap partition to, changing the ***fdisk*** command from: `fdisk /dev/hd`**`a`** to: `fdisk /dev/hd`**`b`**; or from:

`fdisk /dev/sd`**a** to: `fdisk /dev/sd`**b**, depending on the type of hard drives you are using**.** When you are done, enter: `q` to exit ***fdisk***. (For more on disk drive naming conventions in Linux, see question *Wait a minute! Are you saying that there are no drive letters in Linux?*)

Note: To see the complete list of partition types available for your Linux distribution, at the *"Command (m for help)"* prompt, enter: `l` (for "list partition types").

For performance reasons, if you have more than one hard disk drive your first choice should be to create multiple swap partitions, rather than multiple partitions on the same drive or one swap partition and a swap file in a data partition. However, if you have a shortage of available Primary partitions (or there is no more unallocated disk space from which to create new partitions), you *can* create swap files in existing data partitions. To do this, see the question *How do I create a swap file in an existing Linux data partition?*, below:

Q. How do I create a swap file in an existing Linux data partition?

If for some reason you can't, or do not wish to, create Linux swap partitions—or if you have no more available disk space to use for a new swap partition—it is possible to create swap files within partitions that currently hold programs and/or data files. However, your *first* choice should be for one or more Linux swap partitions.

Important Note: There are two disadvantages to locating swap files outside of dedicated swap partitions: 1) The performance of swap files that are located in *data* partitions is slower than that of swap files in *swap* partitions (due to file system overhead and noncontiguous data blocks), and 2) It is also possible that if the swap file is ever damaged it could corrupt the filesystem for that partition, resulting in lost data. For these reasons it is *strongly* recommended that you use swap *partitions* whenever possible, leaving swap files in data partitions as a last resort. Swap files might be appropriate if you temporarily need some extra swap space.

You can create as many as eight swap files, if necessary. Linux will apportion the swapped data between these files as appropriate.

To create a swap file in a data partition, use a command similar to the following:

```
dd if=/dev/zero of=<swap path> bs=<size> count=<size>
```

For example:

```
dd if=/dev/zero of=/Swapdir/Swapfile bs=1024 count=65536
```

(***Note:*** If **/Swapdir** does not yet exist, it must be created *before* running the ***dd*** command. For example: `mkdir /Swapdir` or `mkdir /home/Swapdir`. In order to create the directory, you must be running as the root operator.)

The ***dd*** command (sometimes called "data duplicator," although the original meaning of the ***dd*** acronym is lost in the mists of time) is used to create the swap file. The **if=** parameter identifies the **i**nput **f**ile source for creating the swap file (device "zero"), **of=** is where you specify the path to the **o**utput (swap) **f**ile you want to create, **bs=** is the block size to use and **count=** is where you define how large to make the swap file, in increments of the block size. In the preceding example, we are creating a 64MB swap file (65,536 * 1024 [1K] equals 65,536K, or 64MB), called **Swapfile** (use whatever name you prefer), in a directory called **Swapdir**.

Next, to initialize (or make ready for use) the swap file, use the following command:

```
mkswap /Swapdir/Swapfile 65536
```

(Use the same path as in the ***dd*** command and the file size you specified in the **count=** parameter. If you get a "No such file or directory" error, verify that you typed the upper/lower case correctly and did not omit the leading ***/***.)

After initializing the file, "synchronize" it to ensure that it is properly written to disk:

```
sync
```

And, finally, to tell Linux to start using this new swap file:

```
swapon /Swapdir/Swapfile
```

This will activate the swap file temporarily (until the next time you log out or reboot). To make it permanently active, use an appropriate tool (such as *kfstab* [see the earlier question, *How can I speed up performance by using multiple swap partitions?*, for details], which knows the exact layout of the file that ***fstab*** requires) to open the file **/etc/fstab**.

When you open **fstab**, you might see something like (without the headings):

<partition name>	***<mount point>***	***<partition type>***	***<mount options>***	***<backup dump>***	***<pass>***
/dev/hda1	/	ext2	defaults	0	1
/dev/hda2	/usr	ext2	defaults	0	1
/dev/hda3	none	swap	sw	0	0

(There may be other disk partitions listed if you have other swap and/or data partitions.) To activate your new swap file, add a line similar to the following immediately after the last partition listing. Then save and close the **/etc/fstab** file.

/Swapdir/Swapfile	*none*	swap	sw	0	0

Should you later decide that you no longer need the swap file, it is easy enough to remove. First, use the ***swapoff*** command to disable the use of the file:

```
swapoff /Swapdir/Swapfile
```

Then, simply delete the file:

```
rm /Swapdir/Swapfile
```

Q. I already have a large swap file in my Windows partition. Is there a way for Linux to use that swap space instead of creating another file?

Yes, although Linux is rather meticulous about what format the data takes in its swap space. Thus, there is more to it than simply pointing Linux to the path of the Windows swap file.

I have found several descriptions of how to accomplish this (and all of them reportedly work) but I have not had the need to try any myself. They all appear to be terribly complicated, so unless you consider yourself well versed in Linux, it might be best to avoid this topic entirely until someone develops a utility to simplify the process. If you would like to look over one such procedure, to make up your own mind whether or not to try it, read the document at *http://www.linuxdoc.org/HOWTO/mini/Swap-Space-1.html*. It seems rather thorough. (Use the

left and right arrows to move back and forth through the document. Use the up arrow to go to the document table of contents.)

(If you *already* have a Linux swap *partition* before you decide to try this technique to use your Windows swap file instead, you should later delete it using ***fdisk***, to free up the disk space used by that partition. For directions on how to use ***fdisk*** to check your partition tables, and to delete a partition, see the question *How large a swap partition should I create?*, above.)

Another possible way to share swap space is to have a swap *partition* shared by both operating systems. This is not a "no-brainer" either, but if you are interested do an Internet search for "Linux swap space" and you should find several such documents.

Q. If I install Linux on an IBM® ThinkPad® with a TrackPoint® II or III pointer, what kind of mouse should I select?

The installation program should correctly recognize the TrackPoint II/III as a PS/2-type mouse. Most such ThinkPad notebooks have two "mouse" buttons, but some newer systems have a third button. When installing on a ThinkPad with two buttons, select the *Emulate 3 Button*s option on the *Mouse Configuration* screen. If your ThinkPad has three buttons do *not* select this option, or it will cause problems. To use the "third button" on a two-button ThinkPad (or any other system with a two-button mouse), press both buttons together (called "chording"). (These instructions *should* apply equally to other brands of notebook computers using IBM's "pointing stick" technology, identified as a little "pencil eraser"—usually red, green, or black—nestled between the G, H and B keys; but to be sure, contact the notebook vendor.)

Migration from Windows

If you are in the process of moving from Windows to Linux, or are setting up a Linux system to coexist with Windows computers and perhaps share data between the two types of systems, these questions may save you some time and effort.

Q. How do I get Netscape for Linux to recognize my Netscape for Windows bookmark file?

For best results, try to use the same version of Netscape on both systems, or as close as possible (e.g., 4.72 and 4.7, but not 3.0) to minimize the chance of inconsistencies in the bookmark file format.

Use the Linux Kfind, or similar, utility to determine where the default bookmark file is kept. It should be in a directory with a name like **/home/*username*/.netscape** (substitute your account name for *username*). If you simply copy the **bookmark.htm** file from your Windows PC to a floppy and then to the appropriate Linux directory, when you restart Netscape and open the bookmarks you will see the *default* bookmarks, not your customized list. The problem is in the file name. The Linux version is called bookmark**s**.html (note the two extra letters). Therefore, you must rename the **bookmark.htm** file to **bookmarks.html** before Netscape for Linux will recognize it (reverse the process to copy an updated file back to your Windows system).

First, copy the **bookmark.htm** file from the Windows system. If you use a floppy, Zip disk or other removable medium, you can rename the file on the removable disk before copying to the Linux system. (The file should be stored in a directory called something like **\Program Files\Netscape\Communicator\Users*yourusername***, but you can use the Windows *Find* or *Search* function—depending on the Windows product—from the *Start* menu to locate it.) Once the file is copied to the removable disk, use the Windows file manager to rename it **bookmarks.html**.

On the Linux computer, KDE graphical interface users can copy the file to the **/home/*username*/.netscape** directory using the K File Manager (just click on the icon on the taskbar, or Panel, that looks like a house in front of a manila folder). Gnome users can use the Gnome File Manager (click on the Gnome icon--the vaguely G-shaped footprint--on the Panel, select *Programs*, then *File Manager*). If you prefer to work from a command line interface, use the following command:

```
cp /dev/fd0/bookmarks.html /home/yourusername/.netscape
```

(Note the use of *forward* slashes and that **/dev/fd0** is used in place of **a:**. Linux and UNIX® do not use drive letters, unlike DOS, Windows or OS/2.)

Alternatively, you could combine the renaming and copying steps into one, using the following command:

```
mv /dev/fd0/bookmark.htm /home/yourusername/.netscape/bookmarks.html
```

(***Note:*** The diskette drive may already be "mounted" as **/dev/floppy** or **/mnt/floppy** in your Linux distribution. If so, use that mount location instead of **/dev/fd0**.)

If you transfer the file directly to the Linux system (via network connection, for example) or prefer to rename the file once it is on the Linux system go to the **/home/username/.netscape** directory (`cd /home/username/.netscape`) and use a file manager or the move (***mv***) command to change the name: `mv bookmark.htm bookmarks.html`. (If you use the ***mv*** command without a target location, it simply renames the files.)

Note: In addition to the bookmarks file, you might also want to copy the following files from the same Netscape directory on your Windows system, to your Linux system: **liprefs.js**, **prefs.js** and **proxy.cfg**. They contain the user preferences information (font selections, proxies, servers and so on). In addition, if you did not have "cookies" disabled on your Windows system, copy **cookies.txt** as well (some Web sites store information about your user account in this file). This will save you from having to manually reenter your preferences on your Linux system. (You will probably have to overwrite the default files with the same names on your Linux system.)

If Netscape is still running, close and restart it so it will read the new bookmarks file (as well as the other files you copied). If it still doesn't work, verify that you put it in the correct directory and have spelled the file name accurately—it *must* be in all lower-case.

Q. Are there any Linux programs that will read and write Microsoft Office files?

Yes, there are. For a list of several such programs and how compatible they are, see the white paper entitled *What Good is a Linux Client?,* available from the same sources as this paper.

Configuration & Setup

Here are some tips for tailoring Linux or configuring devices.

Q. How do I get my "winmodem" to work with Linux?

Winmodem is actually a trademark of 3Com Corporation, but is often used generically to describe a class of low-cost software-based modems, designed to be inexpensive by replacing some hardware functions with software. This software typically incorporates low-

level Windows functions that can't be duplicated in Linux. Most such modems won't work with Linux, however, some can be made to be compatible. These sometimes are referred to as "Linmodems". Go to *http://www.o2.net/~gromitkc/winmodem.html* for more information on this topic. In most cases, the easiest "fix" is simply to disable the software-based modem and install a traditional modem adapter, or attach en external modem to a serial or USB port. Many 56K modems are quite inexpensive these days. (For information on how to disable the built-in modem, consult the user's guide for the computer, or contact the computer vendor's technical support organization.)

Q. How can I tell how much memory Linux is using?

From a command shell, use the "concatenate" command: `cat /proc/meminfo` for memory usage information. You should see a line starting something like: **Mem: 64655360**, etc. This is the *total* memory Linux thinks it has available to use. If you have *more* than 64MB of RAM installed and see a number close to the one above, Linux is not using the remainder. To see how much memory is currently *unused*, use the `free` command.

Q. If Linux is not using all the memory I have installed, how do I make it use the rest?

To force Linux to recognize and use the full amount of memory you have installed, open a command shell and login temporarily as the "root" operator, with the ***su*** command. (For instructions on how to do this, see the *Root/User Accounts, Groups and Permissions* section, below.) When prompted, type in the root password. At the command prompt, use an editor to open the file **/etc/lilo.conf**, such as: `pico /etc/lilo.conf`. The first line of the file should start: `boot=`, followed by other lines for `map=`, `install=` and so on. Position the cursor in front of the "b" in `boot=`, then insert the following: `append="mem=128M"` and press the **Enter** key to move the `boot=` line down. Finally, close the editor and exit the file (**Ctrl-X**, in the case of the Pico editor; **Q** to quit various other editors). Save the changes, if prompted. From the command prompt, type: `/sbin/lilo` to restart LILO, then type: `exit` to logout as the root operator and then `exit` again to close the shell session. The next time you reboot Linux should be using all the memory installed. (Use the `cat /proc/meminfo` command to verify.)

Warning: It is possible to render Linux unbootable by specifying a memory size larger than that physically installed and available. If your computer reserves some memory, rendering it unavailable to Linux, you must specify the lesser amount. For example, during power up (POST), if your memory count shows something like **130496** (kilobytes), rather than the full **131072** KB for 128MB of RAM (128 * 1024), this means that some of the memory is reserved by the hardware. In this case, do *not* use `mem=128M` in the example above. Instead, use the exact amount of **130496K** (`append="mem=130496K"`), or play it safe and "step down" to 127MB, as in: `append="mem=127M"`. If you somehow manage to leave your system in an unbootable state (i.e., it locks up on boot when it tries to use the reserved memory), the remedy is to edit the **/etc/lilo.conf** to correct the memory size.

Q. I have a printer attached to the parallel port. What is this port called in Linux?

When prompted for the name of a parallel port in Linux, in place of **LPT1**, **LPT2**, or **LPT3** use **/dev/lp0**, **/dev/lp1**, **or /dev/lp2**, respectively.

Q. I have a device attached to a serial port. What is this port called in Linux?

When prompted for the name of a serial port in Linux, in place of **COM1** through **COM8** substitute **/dev/ttyS0** through **/dev/ttyS7**, respectively.

Q. What is my floppy drive called in Linux?

If prompted for the path of a diskette drive in Linux, in place of "**a:**" substitute **/dev/fd0**, **/dev/floppy**, or **/mnt/floppy**, depending on the Linux distribution you are using. Unlike the DOS/Windows world, UNIX/Linux doesn't use drive letters. Devices are assigned "mount points" (or mounting locations). To determine which mount location is assigned to the floppy drive on your system, type: `mount` at a command prompt. You should see something like:

```
/dev/hda1 on /type ext2 (rw)
none on /proc type proc (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/fd0 on /mnt/floppy type vfat (rw,nosuid,nodev,user=username)
```

Look for the part that starts with **/dev/fd0 on**…. This tells you what "soft" mount location to use, instead of **/dev/fd0**: "**/mnt/floppy**", in this particular case.

Q. Wait a minute! Are you saying that there are no drive letters in Linux?

That is correct. Hard drives, floppy drives, CD-ROM drives, Iomega Zip and Jaz drives, and other devices that get drive letters in DOS/Windows are instead assigned mount points by Linux. Depending on the device involved, here are some of the designations used by Linux, following their typical Windows drive letters:

Floppy drives
A: drive (in Windows) = **/dev/fd0** (or **/dev/floppy** or **/mnt/floppy** in Linux, as explained above)
B: drive = **/dev/fd1** (see note, below)

IDE/EIDE disk drives
C: drive (boot drive; first/primary drive on the first disk controller) = **/dev/hda**
D: drive (second/slave drive on the first disk controller) = **/dev/hdb**
E: drive (first/primary physical drive on the second controller) = **/dev/hdc**
F: drive (second/slave physical drive on the second controller) = **/dev/hdd**
(CD-ROM drives will often be attached as the primary drive on the second controller, or **/dev/hdc**. IDE/ATA Zip drives will generally take the next available drive letter after the hard drives and/or CD-ROM drives.)

SCSI disk drives
C: drive (boot drive; lowest-numbered drive on the primary controller) = **/dev/sda**
D: drive (second lowest-numbered drive on the primary controller) = **/dev/sdb**
E: drive (third lowest-numbered drive on the primary controller) = **/dev/sdc**
(And so on. There can be many SCSI drives, on one or more controllers. The mount point designations just keep incrementing up through **/dev/sdz**. If there are more than 26 physical drives, they will be incrementally labeled from **/dev/sdaa** up through **/dev/sdzz**.)

Note: Unlike the primary floppy drive (**/dev/fd0**) Linux doesn't automatically map a second diskette drive (**/dev/fd1**) to something like **/dev/floppy** or **/mnt/floppy**. However, if desired you can set up your own alias, such as **/mnt/floppy2**. The mapping entries that are used for this are located in the **/etc** directory, in a file called **fstab**. To add an alias for **fd1**, use an editor to open **/etc/fstab** and look for the entry for **fd0**, which should look something like:

```
/dev/fd0    /mnt/floppy     auto    noauto,owner      0  0
```

This line maps **/dev/fd0** to the directory **/mnt/floppy**. To create an alias for **fd1**, create a subdirectory in the **/mnt** directory called **floppy2**. Then add a new entry (using whatever alias you wish) to the **fstab** file in order to map the second floppy drive to a mounting point, such as:

```
/dev/fd1    /mnt/floppy2    auto    noauto,owner      0  0
```

Save the file and close the editor. Now, when you use the Mount command it should include something like:

```
/dev/fd1 on /mnt/floppy2 type vfat (rw,nosuid,nodev,user=username)
```

Q. What about disk drive partitions? Don't they have drive letters either?

Nope. Linux just tacks numbers onto the end of the drive identifier. For example, if the first IDE hard drive had three *primary* partitions, they would be named/numbered, **/dev/hda1**, **/dev/hda2** and **/dev/hda3**. If a second hard drive had two primary partitions, they would be called **dev/hdb1** and **/dev/hdb2**. Likewise, a SCSI hard drive with three primary partitions would be labeled **/dev/sda1, /dev/sda2 and /dev/sda3**.

That is just for primary partitions. If you also have *extended* partitions, they begin numbering at **5** and continue from there. If, for instance, you had one IDE drive with two *primary* and two *extended* partitions, the partitions would be called: **/dev/hda1**, **/dev/hda2**, **/dev/hda5** and **/dev/hda6**.

Q. Wow, that's confusing! Isn't there a simpler way to access drives?

Yes. Most Linux distributions provide desktop icons for some devices, such as floppy and CD-ROM drives that can be clicked on for direct access. If you install software for accessing an Iomega Zip drive, it should also provide an icon to use. And, of course, the graphical file managers, such as the K File Manager and Gnome File Manager allow you to access files on various drives without worrying about drive letters. (When you open KFM, look for an icon called *mnt/*. When you single-click on it, KFM will display all the devices that are ready for use. Click on the icon called *cdrom/* to display the contents of the currently inserted CD, or click on the one called *floppy/* to see what is on the inserted diskette.)

Q. Is there an easy way to see what devices, such as drives, have been mounted?

Yes. **Red Hat** users running KDE can click on the KDE menu and select *Red Hat*, then *System* and finally *Disk Management*. This will bring up a panel that shows the devices mounted, and provides *Mount* and/or *Unmount* buttons (so that you can take a device offline, then remount it). The *Mount/Unmount* buttons will toggle back and forth between the two options being used. There are also buttons provided to let you format the drives.

Gnome users (of any distribution) can select *File manager* from the Gnome icon on the Panel (the Linux "taskbar") and then click on the **/mnt** icon to view the various devices.

I have not seen a KDE equivalent for the Red Hat Disk Management tool for other distributions. As an alternative, the Mount command can be used, as described earlier, in the question *What is my floppy drive called in Linux?*. Also, for a less cluttered view of disk drives mounted, with total and available space, try the disk filespace command (use: `df -h` to display measurements in "human-readable" form—gigabytes, megabytes, etc.). It won't show non-disk devices, however. (Read the ***info*** manual pages at: `info df`, for more information.)

Q. Is there any way to change the default language used by the operating system and applications?

The short answer is "Yes." Of course, as with most things Linux, *how* you do it depends on the distribution you are using, the user interface (desktop environment) you are using, your

applications and perhaps other factors. The default language is set during installation, however there are ways to change this once the system is up and running.

If, like many users, you are running the **KDE** environment, the way to change the language is very simple. From the KDE menu:

1. Select *Settings*, then *Desktop,* then *Language.*
2. Specify your first choice from nearly three dozen languages (including Catalan, Slovak and Macedonian). You are allowed to choose up to three languages. (When you start a program, KDE will try to find a translation for the menus, dialog boxes and messages in the first language. If it fails, it will try the second choice and, if necessary, the third. If all three fail (or if you do not select three languages), it will use whatever was the original default language specified during installation.)
3. Press *OK* to save your changes and exit. (*Apply* will save your changes but *not* exit. The *Default* button will return all three language settings to the original "Default language" choice.)
4. Log out and then log in again to see your changes.

Note: This procedure only affects certain system utilities, help files, error messages and the like. Your applications may or may not be affected, depending on what support for other languages are programmed into them. Therefore, you may end up with a mix of two or more languages depending on the various programs you are running at any one time. You may be able to set up language choices within specific applications as well.

If, after logging out/in you discover to your horror that you somehow selected the wrong language and you can't read most icons, menus and other system functions, try to follow the instructions above. If you can manage to return to the language settings panel, but can't read the language names, the little flag icons should allow you to find the language you want, or you can press the *Default* button at the bottom—the second one from the left, if you can't read it. *If you are unable to find your way back to the language settings you may have to reinstall Linux to specify the default language all over again.*

Gnome users can use the same KDE tool, if they wish. The only extra step is to go to the Gnome menu, select the *KDE menus* option, then proceed as directed, above.

If you use another desktop environment besides KDE or Gnome, refer to the manual or tutorial that came with the environment for information on how to change the language.

General Usage

How to perform various tasks using Linux.

Q. How do I format a floppy (diskette)?

If you are using the KDE user interface, the easiest way is to use the KDE Floppy Formatter utility supplied. Simply click on the KDE icon on the Panel, select *Utilities*, then *KFloppy*. The File System option gives you the choice of the *Dos* (FAT) file system (for DOS/Windows or OS/2) or the Linux native *ext2fs* file system.

If you are using Gnome, go to the Gnome icon (the fancy G-shaped footprint) on the Panel and select *Utilities,* then *GFFloppy*.

From the command shell, to format a diskette for the native Linux file system, use the `mke2fs` command. You can use the ***mtools*** utility to access DOS/Windows-formatted diskettes. Use the commands: `mformat`**,** `mcopy` **and** `mdel` without parameters to see a list of command

options. (Using ***mtools***, to change from one directory on the floppy to another, use the following command: `mcd a:subdir`, where *subdir* is the name of the directory you wish to change to.)

Q. Is there online documentation for most programs?

Yes. The official help tool for Linux is called ***man*** (short for manual). To read the manual for various programs (including many commands), type: `man <something>` (where <*something*> is the name of the program you need help with) at a shell prompt. For example, to read the online manual for the ***man*** program itself, use: `man man`. You can also use a viewer called ***less*** to view the manual: `man man | less` (where **|** is the piping symbol—**Shift-Backslash**—located above the **Enter** key on a 101-key keyboard). Using ***less***, press the **Spacebar** to move ahead a page at a time; to back up, use the **B** key. When you are done, press **Q** to quit.

Many newer programs have shifted to the ***info*** tool rather than using ***man***. Like ***man***, you can view ***info*** pages using the command: `info <something>`. Some programs now supply HTML or XML pages, kept in the ***/usr/doc/<programname>*** directory, in place of either ***man*** or ***info*** pages. These documents may be viewed from a Web browser or other HTML/XML-capable program.

Another option is to *redirect* the manual to a text file for browsing with an editor. For example: `man mv | col –b > move.txt` (to redirect the text for the ***mv*** command) will strip out the screen formatting codes that do not translate into print and then copy the output to a file called **move.txt** (***col*** is the name of the formatting utility). Once it is in a text file, it can be read with Emacs, Pico, Vi, or any other text editor. (***Note:*** Substituting **>>** for **>** in the example above would *append* the text to the end of an existing file, whereas using **>** a second time would *overwrite* **man.txt** with new data.)

Likewise, to print a manual you will need to *pipe* (send) the document to a formatting program that will strip out the screen formatting codes before printing. To do this for ***info*** pages, type: `info mv | col –b | lpr`. (***lpr*** is the program that sends the document to the print spooler.)

Another "quick and dirty" source of helpful information is the ***whatis*** command. ***Whatis*** provides a brief synopsis of many commands. For example, try: `whatis man`. You will be shown several "definitions" of what the ***man*** command does. The same goes for: `whatis whereis`, to find out what the ***whereis*** command does. If there is no information available for a particular command you will see a "*nothing appropriate*" message instead. ***Note:*** If the ***whatis*** command doesn't work, it may mean that the ***whatis*** database has not yet been built. To create it, first use the: **su –** command to log in as root, then enter: `/usr/sbin/makewhatis`. This may take several minutes. When the process finishes, try the ***whatis*** command again.

Q. How do I kill a program that has locked up?

If you are not the owner of the program (the user account who installed it), you must log in as root. (For instructions on how to do this, see the *Root/User Accounts, Groups and Permissions* section, below.) Then, from a command line, you can type: `ps` (process status) to see what programs are running. To the right of the process name will be a Process ID (PID) number, such as 506. This is what you will use to kill the program. From the command prompt, type: `kill pid` (for example, `kill 506`). This tells the program to either shape up (respond normally) or ship out (close itself). This will usually solve the problem, but occasionally stronger measures are required. Preceding the PID with `–9` (such as: `kill –9 506`) *will* kill

the program. (This is sometimes called “signal 9,” not to be confused with “Plan 9 from Outer Space”, a cheesy B-movie.)

An alternative for killing a *background process*, or daemon is to use the `–HUP` (hangup) parameter, as in: `kill –HUP 506`. It essentially signals the program to close itself, then restart and reread its configuration file. Again, if the ***kill*** command doesn’t work the first time, you may have to resort to the `–9` parameter rather than `–HUP`.

Q. Is there a way to increase the priority of a program to make it run faster?

Yes. If you have a program that normally takes quite a while to run and you really, really need it to complete faster, you can boost its job priority level so that it uses more processor cycles. Of course, this means that all other currently executing programs will run slower. But if this is what you want, the ***nice*** command will do the job for you.

Linux supports priority levels from 19 (lowest) to –20 (highest), with the default being 0. To change the priority to a negative number (increasing the priority), you must first log in as root. Then, to *start* a program called Myprog with a high priority-level of `–10`, use the command: `nice –n –10 myprog`. If you wish to increase the priority of an *already running* program, use the ***renice*** command, identifying the program by its PID: `renice –10 506` (the `–n` argument is not used).

Conversely, if you have a long-running program that is a low-priority job and it is slowing the entire system down while the program is running, you can lower its priority using the same techniques by providing positive numbers—such as 8—rather than negative numbers. By default, ***nice*** will *reduce* the priority by 10. In other words, the command: `nice myprog` is the same as: `nice –n 10 myprog`.

It may take some experimenting for you to find the right priority levels to assign to specific programs. Too high a priority may bring the rest of the system to its knees; too low and the job may take forever to complete. *As a rule of thumb, unless you have a reason to be changing priorities, leave them alone.* The default setting is best most of the time.

Q. Is there a spelling checker that I can use outside of a word processor?

Yes. Linux includes a program called ***ispell***, which will read any text file, highlight questionable words, offer alternatives and let you add the words to the ***ispell*** dictionary if desired. It runs from the command line, with a text-mode (i.e., non-graphical) interface, but is fairly easy to use. There are a number of parameters that you can use (type: `ispell --?` for a list of them or `info ispell` for more detailed explanations). But the simplest usage to check a file called **memo.txt** would be: `ispell memo.txt`. It will even check a series of files in order, such as: `ispell memo.txt mydata.txt` or even `ispell *.txt` (to check all files in the current directory with a .txt extension).

When the program runs it will present a “questionable” word within the context of the line of text containing the word and offer suggested replacements in a numbered list. To select one of the suggestions from the list to replace the original word, simply type in the number of the word you prefer. There are also several menu options displayed along the bottom of the program. To **R**eplace the word with something you type in, type **R** (or **r**, ***ispell*** options are not case-sensitive) and then whatever word you want to replace the original with. (You also can type in the number of a replacement word from the list. This is the same as just typing in the number directly, but is there in case you forget you can do it that way. Use **A** to **A**ccept the existing word as is (i.e., ignore it) for the rest of the spell-checking session. (Using the **spacebar** tells ***ispell*** to accept/ignore the word this time only.) **I** lets you **I**nsert (or add) the

existing word into the ***ispell*** dictionary. Entering an **L** lets you **L**ook up other words in the dictionary. (Perhaps you would like to find another word not on the suggestion list.) If the word is correct but capitalized inappropriately, choosing **U** will **U**ncapitalize it. Either **Q** or **X** will end the program. If you have specified a list of files to spell-check, **X** will stop checking the current file, save it and begin on the next one. This is essentially "save and exit." On the other hand, **Q** will ask whether you really want to **Q**uit immediately without saving your changes. (Type **Y** or **N**, depending on your preference.)

Be sure to check out the associated online help for that program by entering **?**. In addition to explaining the above options, there are others listed that you might not know about or remember, including Redraw screen (**^L**).

Warning: ispell is not discriminating as to what files it checks. It is entirely possible to spell-check a word processing document or a binary program file. Saving changes to a document created by a word processor *may* (or may not) corrupt it, but the word processor probably has its own built-in spelling-checker anyway, so why bother with ***ispell***? On the other hand, you *will* corrupt a *program* if you change anything and then save the changes with the **X** option. If you accidentally open a binary file (easily identifiable by the "gibberish" displayed that ***ispell*** is trying to correct), be sure to **Q**uit immediately with the **Q** option!

GUI/Desktop

The following questions involve window manager/desktop and other graphical user interface questions.

Q. My copy of Linux came with KDE and Gnome. Why do I need both?

Technically speaking, you probably do not *need* both. However, both desktop environments are popular, so this gives you the option to decide which you like better. Also, some programs will work only with one or the other, allowing you to use both types of programs, switching between the two desktops as needed.

Q. I have both KDE and Gnome installed. How do I switch between them?

From the graphical interface, simply log out. Then at the *Log in* screen, type your login ID and password and choose which desktop you wish to load (*Session Type*). This choice will remain your default until you change it to something else.

Q. Can I add more desktop themes to use with Gnome and KDE?

Yes. One source for more Gnome themes is *http://gtk.themes.org*. To install a new theme in Gnome, after downloading it, start the Gnome Control Center by clicking on the "toolbox" icon on the Panel. Then click on *Theme Selector* (under *Desktop*) in the panel on the left. Press the *Install new theme* button (in the center of the right-hand panel), and when the dialog box opens, scroll down to the name of the file you downloaded (called something like **new_theme.tar.gz** and most likely in your login directory). Click on the name of the theme you downloaded and press the *OK* button. When the theme appears in the *Available Themes* box you can press the *Preview* button (if *Auto Preview* isn't already selected) to see how it looks before applying it. If you think you might like it, press the *Try* button to conditionally apply it to your desktop. If you approve of the results, press *OK* to save the changes. If you decide you do not like it after all, use the *Revert* button to back out the theme and return to your previous theme.

KDE themes can be found at *http://kde.themes.org*. To install a new KDE theme, start the KDE Control Center from the "home" icon (it looks like a little house in front of a manila folder) on the Panel. Then, click on *Theme Manager* (under *Desktop*) in the panel on the left. Press

the *Add* button (in the upper-right part of the right-hand panel). When the dialog box opens, find the name of the theme file you downloaded. Double-click on the name to install it. Then click on the theme name in the *Installer* box. The window in the middle shows how it will look. If you like the appearance, press the *Apply* button at the bottom of the panel to change your desktop theme. If you do not like the appearance, press *Cancel* to exit the KDE Control Center.

Other sources for themes include many Web sites that offer software downloads. For a list of some of these sites, refer to the *What Good is a Linux Client?* white paper, available from the same sources as this Q&A paper.

Q. How can I add programs to the Panel?

KDE users can click on the KDE icon, select *Panel*, then *Add Application* and then choose the desired program from the various menus shown. Once the application has been added to the panel, you can rearrange the order of the icons by right-clicking on an icon and selecting *Move,* then dragging the icon to wherever you want it.

Gnome users must first log in as root, then click on the Gnome menu icon, then *Programs* and *Settings*, then select *Menu editor*. When the popup menu appears you can create a submenu or add to an existing menu.

Q. What is a virtual desktop?

When running a number of programs concurrently, the screen sometimes becomes cluttered with several windows open at once, covering each other up. One solution, of course, is to minimize some of the windows until they are needed. However, this entails minimizing and restoring those windows repeatedly, as different windows are accessed. This can be tedious over the course of a day.

An alternative is to use a program that creates “virtual desktops.” Each desktop is a clean slate where you can open one or more programs. Rather than minimizing/restoring all those programs as needed, you can simply shuffle between virtual desktops with programs intact in each one. (If this doesn’t seem like much of an improvement, consider that with eight open programs, a user might have to minimize and restore each of those eight programs multiple times in turn. With eight virtual desktops, each program could stay open at all times, needing only one click to change to the next, instead of two.) Virtual desktops can help reduce desktop clutter.

There are several programs of this type available for Microsoft Windows users to buy or download, but Linux users do not even have to look for such a program. Both KDE and Gnome include virtual desktop support by default and users can specify how many desktops they want to use—or disable them entirely. To switch between desktops, click on the buttons on the Panel (for One, Two, Three, etc.), start one or more programs, then select another virtual desktop, start more programs and so on. To cycle through all the desktops in turn, use *Ctrl-Tab*.

Q. You said I could have eight virtual desktops, but I see only four. Where are the rest?

The default setting is four, but you can easily change the number in increments of two (i.e., 2, 4, 6 or 8). To do this in KDE, click on the KDE icon on the Panel, then select *Panel* and *Configure*. This will display the *KPanel Configuration* window. Use the *Visible* slider to change the number of virtual desktops. (The *Width* slider lets you control how big the icons are on the Panel.)

If you are using the Gnome environment, from the Gnome icon on the Panel go to *Program* and select *Setting*, then *Gnome Control Center.* When the Gnome Control Center panel appears, click on *Desktop*, then *Windows Manager* and finally *Workspaces*. Select the desired number of virtual columns or desktops and press *OK* to close the panel.

Q. Is it possible to share a program across multiple virtual desktops?

Yes. In the upper left-hand corner of a program window there is an icon that looks like a pushpin. Pressing this button will "pin" that application in place, making it appear in all virtual desktops, in the same position onscreen. To remove it from other virtual desktops, press the button again and it will remain on only the virtual desktop from which it was "unpinned."

Q. How can I change my desktop background color?

KDE users can click on the KDE icon on the Panel and select *Settings*, then *Desktop* and finally *Background*. From the *Display Settings* dialog box you can pick which desktop (from among the virtual desktops) to modify, then choose one or two colors and/or wallpaper to use. To test out your choice, press the *Apply* button. When you are finished, press *OK* to save your changes, or *Default* to go back to the original system default colors. (*Cancel* will *not* abort the changes once you press *Apply*; it will only close the panel.)

Q. Is there a faster way to copy/cut & paste text than to use the pull-down menu?

Yes. There is a quick way to *copy* text with the mouse. If you are using a three-button mouse, first position the cursor where you wish to place the text, then select (highlight) the text as you normally would with the mouse. Finally, click the middle mouse button. (If you have a two-button mouse set up to emulate a three-button mouse, press both the left and right mouse buttons simultaneously to achieve the same effect.) The highlighted text will be copied to the cursor position. This should work in most text editors and command line sessions, as well as in word processors and other graphical programs. Alternatively, you can paste the text with **Shift-Insert** (hold the **Shift** key while pressing the **Insert** key), rather than using the third mouse button.

For a keyboard method, you can highlight the text (using the mouse or holding down the **Shift** key while using the **left and right arrow keys** to "paint" the text) then use **Ctrl-C** to copy (or **Ctrl-X** to cut) the text, then **Shift-Insert** (or the third mouse button) to paste; similarly to the Ctrl-C/Ctrl-X/Ctrl-V trio used in Windows.)

Note: Because of all the different user interfaces, editors and applications used in Linux, not all of these techniques will work in every program, so you may need to experiment to see what works where.

Command Shell

The next series of questions are specific to using commands and command shells.

Q. If I want to issue a command, how do I open a command prompt?

In Linux parlance, it is generally referred to as a command shell, or simply a shell. Each Linux distribution includes at least one shell. The most commonly used one today is bash (short for Bourne Again SHell), an enhanced version of the older Bourne shell. To open the default shell for your distribution, KDE users can click the *Kvt* icon on the Panel (the icon that looks like a computer terminal), or press **Ctrl-Alt-F1**. This will provide a command line interface (CLI) from which you can run commands as needed.

Gnome users can go to the Gnome icon on the Panel, select *System* and open either a regular or color Xterm (terminal) session, or command prompt window, by clicking on the icons that look like a computer terminal with a large red or multicolored X in front of it, respectively. Alternatively, you can right-click anywhere and open a new terminal by selecting *New,* then *Terminal*.

If you need only a "quickie" command line, just long enough to execute one command and then close itself, press the **Alt-F2** keys. When the pop-up panel appears, type the command and press **Enter**. Instead of a command, you can enter a data file name (with the path necessary to find it) and Linux will launch the appropriate program, based on the file association. (For example, typing: `/home/username/.netscape/bookmarks.html` will cause the default browser to launch.) If you use a path to a *directory* instead of a file, Linux will open the directory using the default file manager, such as KFM.

Q. What are the Linux equivalents to DOS/Windows commands?

There are a number of commands in Linux that are similar or identical to either the name or function of DOS commands. Here are some common DOS commands with their Linux counterparts:

Command Purpose	DOS Command	Linux Command
Change to the parent directory	**cd..** or **cd ..** (short for change directory)	**cd ..**
Change to the root directory	**cd** or **cd **	**cd /** (note the *forward* slash)
Clear the screen	**cls**	**clear**
Close a command prompt window	**exit**	**exit**
Compare the contents of two files	**fc** (file compare)	**diff** (difference)
Copy a file	**copy**	**cp** (copy)
Create a directory	**mkdir** or **md** (make directory)	**mkdir**
Delete files	**del** or **erase**	**rm** (remove)
Delete directories	**rmdir** or **rd** (remove dir)	**rmdir**
Display file contents, a screen at a time	**more**	**more** or **less**
Display the amount of available RAM	**mem**	**free**
Display/change the date	**date**	**date**
Display/change the time	**time**	**date**
Echo output to the screen	**echo**	**echo**
Find a file by filename	**dir** or **attrib**	**locate**
Find a file by date or other file attributes	**dir** (not by date)	**find**
Find a text string in a file	**find**	**grep**
Format a floppy	**format**	**mke2fs** (or **mformat**, for a DOS/Windows floppy)
List files in directory	**dir** (directory)	**ls** (list) or **dir**
Move files	**move**	**mv** (move)
Rename a file	**ren** (short for rename)	**mv**
Show the current filepath	**cd**	**pwd** (print working directory)

Even though some of the commands are the same, the parameters for these commands may differ between DOS/Windows and Linux. For example, let us look at the differences between changing the date and time in Linux versus Windows.

In Windows, at a command prompt you would type: **`date`** `12-31-2000` to change the date to December 31, 2000.

Then, to change the time to 2:00 p.m., you would use: **`time`** `14:00:00` (or simply **`time`** `14:00`).

In Linux, on the other hand, first you would have to be logged in as the Superuser (or use the command: **`su –`** to temporarily grant yourself that authority), then you would use the command: `date 1231`**`1400`**`2000`. Notice that the 14:00 is embedded in the date string, between the day and the year. This is the way Linux wants it. You can't specify seconds using this command—just hours and minutes.

However, if you want to change only the *time*—but not the date—use the following command instead:

```
date –s 1400
```

or, if you want to specify seconds:

```
date –s 14:00:35
```

(The colons are *required* when specifying seconds.) In both examples, the **–s** parameter tells Linux that you wish to **s**et the time.

For more on the ***su*** command, see the *Root/User Accounts, Groups and Permissions* section, below.

Q. How do I get help for command parameters?

There are several options available. First, many common commands will have short "manual" pages built into Linux. You can access them by typing: `man`, followed by the command name, such as: `man mv` to see the parameters for the ***mv*** (move) command. Some utilities provide more detailed (and sometimes newer) command help using the ***info*** tool; for example: `info mv`. If these tools do not produce results, try following the command with `–help` or `–h`, as the only parameter (as in `mount -h` or `mv --help`). Some commands, such as: `mformat`, will display the list of available arguments whenever the command is used without parameters, or with invalid parameters. When in doubt, try them all until you find the one that works.

To learn how to use the ***info*** command, type: `info info` (the tool uses itself to display the tutorial). To learn more about ***man***, use: `man man` and/or `info man`. (The tutorials are different, so it is not a bad idea to consult both.) When you are finished reading the help displayed by ***man*** or ***info***, press the **Q** key to quit and return to the command shell.

Q. How do I repeat a command?

If you tend to use the same commands over and over, or recently typed in a rather lengthy set of parameters you wish to use again, there are a number of ways you can repeat a command without having to retype it. Linux stores the last 500 commands in a file (called **.bash_history**) for quick retrieval. The simplest way to find a recent command is to use the up and down arrow keys to scroll back and forth through the list until you find the one you want. If you wish to change a command you used recently, use the **up/down arrow** keys until you find the command you want, then use the **left/right arrow** keys to go to the part you wish to change. Type in anything you need to add and use the *Del* key to delete unneeded text.

Alternatively, typing: `!!` (pronounced "bang, bang") and pressing **Enter** will execute the last command entered.

If the command is too far back up the list for scrolling to be convenient and you know the name of the command, you may prefer to jump to it directly, using a shortcut: `! string` (where string represents the command). For example: `! mcopy` might find the previously used command: `mcopy –tnm something.txt c:/mydir` and execute it directly.

If you are not sure of the command name or have used several variations of the command and are not sure which one you want, you can view the entire history file using the command: `history` (without parameters). The commands will be numbered in the order they were executed. When you find the command you want, you can execute it simply by typing: `! 452` (or whatever the correct number is). If you do not want to view the entire 500 line history, you can elect to see just the last x number of commands, such as: `history 50` (to see the last 50).

If you need to search the entire 500 item list but do not want to read every line looking for a command, you can use the ***grep*** command to search the contents. For example, to find that ***mcopy*** command from earlier, type: `history | grep mcopy` (where **|**, the piping symbol, is the **Shift-Backslash** character above the **Enter** key, on a standard 101-key keyboard). You can even use partial-word searches with ***grep***, such as: `history | grep m` (to find the ***mformat***, ***mdel*** and other commands starting with "m" that were executed previously), or even wildcard searches (`history | grep m*y`). If none of these methods finds the command you are looking for, then most likely it was executed more than 500 commands ago and has "fallen out" of the **.bash_history** file.

Q. What can I do if I don't remember the name of a command I need?

You can page through a manual looking for the command you want, but a quicker way is with the ***apropos*** command. It allows you to search a database of available commands, using keywords. For example, if you want to operate a floppy drive but do not remember what command you used the last time (or perhaps this is your first time and you do not know where to start), type: `apropos floppy`. You should see a result like:

```
fd          (4)  - floppy disk device
fdformat    (8)  - Low-level formats a floppy disk
mbadblocks  (1)  - tests a floppy disk and marks the bad blocks
in the FAT
mformat     (1)  - add an MSDOS filesystem to a low-level
formatted floppy disk
mkbootdisk  (8)  - creates a stand-alone boot floppy for the
running system
setfdprm    (8)  - sets user-provided floppy disk parameters
```

(The numbers in parentheses indicate the section in the ***man*** pages where this information can be found, such as: `man mformat` to look at section 1 (the default) in the ***man*** pages for the ***mformat*** command, or: `man 8 mkbootdisk` to go directly to section 8 to see the ***rmt*** command.)

If you were to use the command: `apropos diskette` instead, Linux might return a "*nothing appropriate*" message. This indicates that ***apropos*** doesn't recognize the term "diskette." If this happens on a search, simply try another term to look for, such as "floppy." It is also possible that ***apropos*** won't find every possible related command. For instance, if you search for information on tape drives using: `apropos tape`, it would list a number of commands related to operating the tape drive, including ***mt*** and ***rmt***, but it would not include ***tar***, which can be used for tape backup, so you may need to be creative to find what you are looking for.

Note: If the ***apropos*** command doesn't work, it may mean that the ***whatis*** database (which is accessed by the ***apropos*** command) has not yet been built. To create the database, first use the: **su –** command (and type in your password when prompted) to log in as root, then enter: `/usr/sbin/makewhatis`. This process may take several minutes. When it finishes, try the ***apropos*** command again. For more on ***whatis***, refer to the question, *Is there online documentation for most programs?*

Q. I just used a command and got completely different results from the last time I used it. Why?

Remember that Linux is case-sensitive. Verify the case of the parameters you used for the command. For example, if you type: `tar –xvf filename.tar` one time, and: `TAR –XVF FILENAME.TAR` on another occasion you *will* get different results, because x and X, v and V, f and F, are all valid, yet different, parameters for the ***tar*** command**.** In fact, because commands are case sensitive as well, you could have two completely different commands, called ***tar*** and ***TAR***—with different syntax and parameters—on the same system.

Q. Long pathnames are a pain to type in. Is there any sort of shortcut I can use?

Yes, in some cases. If you are using the Bash shell, or other shell with similar capabilities, you can take advantage of a feature called filename expansion, by using the Tab key. You can type the first part of a directory name, then press **Tab** and it will complete the name for you (as long as there are no other names starting with the same letters.

For example, if you want to list the **/home/johnson/projects/2000** directory, you could type: `ls /ho[Tab]/jo[Tab]/pr[Tab]/2` and the shell will expand the pathname for you. But, what happens if you have two directories under **/home/johnson**, called **/projects** and **/previews**? If you know this ahead of time you could just change **/ho/jo/pr/2** to **/ho/jo/pro/2**, to eliminate the conflict. Alternatively, if pressing the **Tab** key after **/pr** doesn't provide the pathname expansion, pressing **Tab** a second time will bring up a list of all names that start with "*pr*". (In other words: `ls /ho[Tab]/jo[Tab]/pr`**`[Tab][Tab]`**.) Then simply use the keyboard up-arrow key to repeat the previous command and change "*pr*" to "*pro*", or whatever else is needed to create a unique match; then finish the command with the **/2** at the end.

Q. Is there a way to use one command to start more than one program at a time?

Yes. The command shell allows you to "stack" commands on one command line, to be executed in order. For example, if you type: `ls –l; cd ..; ls –a` Linux will list the current directory contents in "long form" (`ls –l`), then change directory to the parent of the current directory (`cd ..`), and finally list all files in that directory (`ls –a`). The "semicolon-space" separating the individual commands is how Linux knows where one command ends and the next begins. If you wanted to copy a file from one directory to another and then to a floppy (while changing its name), and then display the contents of the floppy, you could type:

```
cp /dir1/myfile.dat /dir2; cp /dir1/myfile.dat /dev/fd0/myfile2.dat; ls –a /dev/fd0
```

Stacking commands can be an efficient way of starting several programs that must run sequentially, without having to stand around waiting for each to finish in turn. Simply type in the stacked commands and walk away until the last one is done. Of course, if you need to perform several tasks end to end on a recurring basis, it makes more sense to create a "script" file (like a batch file, for you DOS/Windows users) containing those same statements. Then you just have to type in the name of the script file, instead of a long multi-part command.

Warning: Like many powerful tools, command stacking can be a double-edged sword. It can hurt you as easily as help, if you are not careful. As an illustration of what can happen if you are careless, years ago I was using the OS/2 operating system to write a book about OS/2, which also supports command stacking (although without requiring the semi-colon between commands). I decided to delete the existing files from an old diskette so that I could back up my book chapters (one file per chapter) to it. I *intended* to type in the following command: `del a:\*.*` which would have deleted all files from the root directory of the a: drive. Unfortunately, what I *actually* typed was `del a: *.*` (a space where the backslash should have been). OS/2 interpreted that as "delete all files from a: (`del a:`) and then delete all files from the current directory (`del *.*`)"—all my original files! I realized what was happening about halfway through the files on the hard drive and was able to cancel the job before the rest were deleted. Fortunately, I had a backup from the night before, so I lost only a few hours work, but it could have been much more serious. The moral of this story is be *very* careful with command stacking (or script files), because a lot can happen when you run programs unattended. If you were to issue a command to first copy a file to a floppy and then delete the original, and the floppy copy was unsuccessful (the disk was full, or not formatted, or defective), it is possible for the file to be deleted without having been backed up first.

Q. Is there a way to stack commands and have them execute concurrently in other command sessions?

Yes. Running the stacked commands in the examples from the previous question will cause them all to execute one after the other in the same session window, rendering it unavailable until all the commands have finished running. If you would rather the commands launch another command session and run the commands from there, simply append an ampersand (**&**) symbol to the end of the first command, such as:

```
cp /dir1/myfile.dat /dir2&; cp /dir1/myfile.dat /dev/fd0/myfile2.dat; ls –a /dev/fd0
```

In this case, Linux would launch a new session for the first command, then continue to execute the remaining commands in that session when the first one is finished. Meanwhile, the original session is available for your use.

If for some reason you want *each* command to run in its own session, append the ampersand to *all* of them, as follows:

```
cp /dir1/myfile.dat /dir2&; cp /dir1/myfile.dat /dev/fd0/myfile2.dat&; ls –a /dev/fd0&
```

Of course, if you are doing this just to free up the original session so you can keep working while the other sessions are busy running those commands, a simpler solution might be to just manually launch a second command session to work from while the first one is busy.

Launching several programs in different sessions offers possible advantages. If, for example one program is doing heavy database updating while a second is performing spreadsheet recalculations and a third is displaying complex graphics onscreen, each is primarily stressing a different part of your computer's hardware. (The database update is using the hard drive heavily, the spreadsheet is grabbing a lot of processor cycles to calculate numbers and the graphics program is mainly using the video controller.) Running them all at once could greatly improve the performance of your computer by using system resources that would otherwise sit idle until the current program finishes.

Bear in mind, however, that how useful this is will depend to a great degree on the kinds of programs you are running and the type of hardware your computer has installed. For example, if you run two programs at once that both make heavy use of your hard drives, people with multiple IDE drives (or one SCSI drive) will likely see little or no performance improvement by doing this. This is because the time spent jumping back and forth between hard drives, or

even between areas on the same hard drive, offsets much of the time savings of running the programs together. IDE devices are "dumb," relying on the processor on the IDE controller to handle all the disk drive instructions, but the controller can access only one drive at a time, so it has to stop working on one drive, switch to the other, work for a while, then stop and return to the first. This is very disruptive. On the other hand, for those computers with SCSI drives, the situation is very different. SCSI devices are intelligent (which is why they typically cost so much more than IDE drives) and the SCSI controllers can multiprocess, allowing all SCSI devices to execute concurrently and independently of one another. So if you have two programs accessing data on two different SCSI drives, they can both run full speed all the time. Of course, regardless of disk type, if you do not have enough memory to run both of these programs concurrently, Linux will have to alternate between them in memory, swapping all or part of one program out to the hard disk temporarily. This extra disk activity will reduce the amount of time that can be spent reading/writing the data files, again hurting performance.

Similarly, if one program is making heavy use of the processor and the other is primarily graphics-oriented, there still may not be as much an advantage as you might think to running them concurrently. If you have a high-end video card with lots of memory, it may be able to process all of the graphics commands using the video controller's onboard processor and memory. However, many computers—especially those at the low end of the price range—use inexpensive video controllers with little or no video memory (some share the computer's memory instead, using something called Uniform Memory Architecture, or UMA). This means that the computer's processor may have to process all of the spreadsheet calculations *and* the graphics calculations at once, reducing the performance of both programs. Moreover, using the main computer memory for both may again result in swapping, unless you have lots of system RAM.

Still, it can't hurt to try executing your stacked commands concurrently. If they all complete faster than if run sequentially, terrific. If not, you can always go back to running them the "old fashioned" way.

Logging In/Logging Out/Shutting Down

This section contains questions or problems involving shutting down Linux, or logging out of an account.

Q. I can't seem to log in, even though I'm using the correct password. What's wrong?

Linux is case-sensitive, meaning that a password of **a1b2c3** is *not* the same as **A1B2C3**. Be sure that the Caps Lock key is turned off before typing your password (unless, of course, your password is all caps).

Q. Why do I need to use Shutdown? Can't I just turn the power off?

Just as with Windows, using Shutdown allows the system to save open files, flush the system cache and do other necessary system maintenance. Turning the power off abruptly circumvents this housekeeping and can result in lost data and other problems. Also, because the file system is left in a "dirty" state, it must be cleaned up the next time the system is booted, slowing the reboot considerably.

Q. I can find a Logout option, but where is Shutdown?

Using the KDE user interface, click on the KDE icon (the large stylized K) on the Panel and select *Logout*. When the Login panel pops up, click on Shutdown and then one of the three options: *Shutdown*, *Shutdown and restart* or *Restart X Server*. *Shutdown* shuts down the

operating system and turns the system power off; *Shutdown and restart* does a restart, and *Restart X Server* restarts only the X Server process without restarting the system.

From the Gnome desktop, click on the Gnome icon (the large stylized G that looks like a funny footprint) on the Panel and select *Log out*. When the Log Out Confirmation panel pops up, select one of the three choices: *Logout*, *Halt*, or *Reboot and* press the *Yes* pushbutton. *Logout* logs you out of your account and brings up the Login panel; *Halt* logs you out and turns off the system; *Reboot* logs you out and restarts the system. (*Halt* and *Reboot* will require your account password to proceed.) Pressing the *No* pushbutton will return you to your Gnome session.

From a command line, use one of the following commands: `shutdown –r now` (to reboot immediately), or `shutdown –h now` (to halt, or stop, the system). To set an interval before shutting down, replace "now" with "+*number*", such as: `shutdown –r +5` (for a five minute delay)**.** You will be prompted for your account password before the system shuts down.

Q. Is there a faster way to "warm-boot" the system than using Shutdown?

Yes. As with Windows, you can use the **Ctrl-Alt-Del** key combination to restart the system, but first you must open a full-screen virtual console session. A simple way to do this is by pressing **Ctrl-Alt-F1**. Once the text mode screen appears you can press **Ctrl-Alt-Del**. You will *not* be asked whether you are sure; reboot is immediate. Be sure that you do not have any files open before doing this, as you can lose data otherwise. If, after opening a virtual console you change your mind, press **Ctrl-Alt-F7** to close the console and return to the graphical desktop.

Q. Do I have to use Shutdown if I am just turning over the system to another user?

No. In this situation, the preference would be to simply use Logout, allowing the next user to login without the time and effort of rebooting the system.

Q. I clicked on Logout and now the session is locked up. How do I log out?

Just press **Ctrl-Alt-Backspace** to kill the session and return to the Login screen.

Q. How can I enable Linux to automatically restart applications that are running when I use Shutdown or Logout?

On both the KDE *Logout* panel and the Gnome *Log Out Confirmation* panel there is an option to *Save current setup*. Make sure these options are selected before logging out of these environments.

Files and Directories

The following questions relate to working with files and directories.

Q. What is the root directory?

The roots of a tree form the foundation for the entire tree. The trunk of the tree has a few large branches; these branches lead to smaller branches, which in turn have smaller and smaller branches that eventually lead to leaves.

Using this analogy, the root directory is the base (roots and trunk) from which all directories (main branches) and subdirectories (smaller ones) branch-off. Inside these directories are data files and programs (the leaves). This sort of arrangement (used by UNIX/Linux, DOS/Windows, OS/2 and other operating systems) is often referred to as "tree-structured."

Q. Can you explain more how the directory structure of Linux works?

The main difference between how this works in Linux versus Windows, is that Windows uses backslashes (****) instead of slashes (***/***) in the directory pathnames. To illustrate, here is the hierarchy of a series of directories branching off the root:

```
/
/home
/home/mark
/home/mark/.netscape
/home/mark/.netscape/cache
/home/mark/StarOffice51
/home/mark/StarOffice51/database
/home/mark/StarOffice51/gallery
/home/mark/StarOffice51/gallery/clipart
/home/susan
/home/susan/.netscape
/home/susan/.netscape/cache
/usr
/ust/etc
```

To show it another way, this is how it looks lining up those same directories by equivalent levels of "depth":

```
/
  /home
       /mark
             /.netscape
                          /cache
             /StarOffice51
                          /database
                          /gallery
                                   /clipart
       /susan
             /.netscape
                          /cache
  /usr
       /etc
```

The **/** indicates the root, or base, directory. **/home** is a directory off the root and **/home/mark** and **/home/susan** are directories branching off from **/home**. (These subdirectories hold account-related files for user IDs "*mark*" and "*susan*".) Within **/home/mark** there are directories called **.netscape** and **StarOffice51**, which in turn have directories branching off from them. Susan has a similar directory structure for her Netscape data files. Outside of the **/home** directory structure, there is also a directory called **/usr** branching directly off the root. To change analogies for a moment, if **/mark** (and its subdirectories) and **/susan** (and its subdirectories) are each children (and grandchildren) of the **/home** directory, then **/usr** is a sibling (or peer) directory of **/home**. Both are direct offshoots of the root. (This whole naming convention is referred to as the "file namespace".)

From the root directory, to delve deeper into directories and subdirectories, use the ***cd*** (change directory) command. For example, `cd /mark/StarOffice51` would immediately

move you down the directory *path* from the root to the contents of the subdirectory **StarOffice51**. (Using the command: `ls -a` at that point would show you the files and subdirectories stored in **StarOffice51**.) If you then wanted to change to the **cache** directory inside of **StarOffice 51**, you have two ways of doing so. You could type the "absolute" path of: `cd /home/mark/.netscape/cache` (which says, "starting from the root, travel down these directories") to get there, or you could simply use the shorter "relative" path of: `cd cache` to get to the same place. A relative path means, "starting from where we are now, go to the next level directory, called cache." (Had there also been a directory below **cache**, called **etc**, we could get there from the **StarOffice51** directory with the command: `cd cache/etc`).

Remember, preceding the pathname with a forward slash (***/***) tells Linux to start from the root directory. Therefore a command of: `cd /cache/etc` would fail in the preceding example (unless you also happen to have a **cache** directory branching off from the root, with its *own* **etc** subdirectory). Why? Because it is looking for the cache directory starting from the root rather than from the **StarOffice51** directory.

Reversing our direction, if you use the `cd ..` command from within a directory it will step you back up a directory level (such as from **/home/mark/StarOffice51** to **/home/mark**). You can repeat this command as often as necessary to return to the root directory (***/***). Alternatively, to jump directly to the root, use the `cd /` command. (This is equivalent to `cd\` or `cd \` in DOS/Windows.)

Q. There are a number of directories I don't recognize in the root directory. What are they?

These directories store essential system files and programs. For example, the **/etc** (et cetera) directory contains configuration files, where system and application settings are stored. Beneath that, the **/etc/skel** directory contains shell, or "skeleton" files used to create user accounts. The **/usr** (user) directory stores other important files, including program libraries (shareable system program modules) in **/usr/lib**, manual pages for the ***man*** utility in **/usr/man** and other documentation in the **/usr/doc** and **/usr/info** directories. Other such system directories include **/lib** (libraries: security files, system modules and libraries containing software libraries and informational databases), **/opt** (optional: pre-loaded software packages are often stored here) and **/var** (variables: spooling, logs and all messaging files). Unless you have a specific reason, and know what you are doing, it is a good idea to leave those directories alone.

Q. Is there a command line shortcut to the login directory?

Yes. From any shell prompt simply type: `cd` and press the **Enter** key.

Q. Some filenames are preceded by a dot. What does this mean?

A filename that starts with a dot—such as **.bashrc** or **.netscape**—is a hidden file. They are often used to hold user preference settings, and other configuration information. These files are hidden to make it less likely that they will be accidentally deleted. Hiding files also makes a file listing shorter and less cluttered with files the user will rarely be interested in. There are different ways to see these files. For example, the `ls` (list) command with no parameters won't display these files, but `ls -a` (list all) will.

On the other hand, if you are using the K File Manager (KFM) under KDE, simply select *View*, then *Show Hidden Files* from the pull-down menu to see these files. (To start KFM, just click on the Panel icon that looks like a house in front of a manila folder.)

Gnome users can click on the Gnome icon (the funny G-shaped footprint) on the Panel, select *Programs*, then *File Manager*. From the Gnome File Manager, choose *Settings*, then *Preferences*; finally, click on *Show hidden files*.

Q. When I look at a list of files and directories, the names are followed by something like -rw-rw-r-- or lrwxrwxrwx. What does this mean?

These cryptic notations actually describe the "permissions," or access levels, assigned to three groups: the file owner, groups and other users. The *first* position identifies what type of object the item is. Possible types include a physical (actual) **D**irectory/folder (**d**), a physical file (**-**), a mere **L**ink/alias (pointer) to a file or folder (**l**)—similar to a Windows shortcut, a **S**ocket (**s**), a pipe (**=**), or a device (**b** is for "**B**lock" devices, such as disk drives, that handle data a block at a time; and **c** is for "**C**haracter" devices, like serial and parallel ports, that deal with data one byte, or character, at a time). The remaining nine positions are composed of three groups of three characters each.

The first three apply to the file owner (usually its creator). The next trio describes the permissions granted to any authorized groups of users and the final three characters denote any access available to unspecified other users. Each triad of positions reveals whether the user has Read (**r**), Write (**w**)—i.e., create/modify/delete, or Execute/run (**x**) permission, or no permission to perform one of those actions (**-**).

Using the first example from the question, **-rw-rw-r--** tells us that the object is a file (because the first character is **-**). The first **rw-** indicates that the owner has read/write access to the file, but not execute (so it can't be a program, otherwise the owner would be able to run it). Likewise, any groups that have been granted access to the file can Read and Write to it (the second **rw-**). However, any "other" users (those not expressly added to the authorized group) have only Read access to the file, as evidenced by the **r--** at the end. If the permissions were **drw-rw-r--** instead, the only difference would be that the object is a folder/directory rather than a file.

In the other example from the question, **lrwxrwxrwx** means that the object is a link to a file or folder and all three groups have read, write and execute permission to access that object. Execute permission means that if it is a program, script or other executable file, an authorized user (in this case, anyone) can run it, and if it is a folder then the user can read the contents.

Q. The `ls` command doesn't show everything in the directory. What's wrong?

The ***ls*** (lowercase L, not capital i) utility has many parameters to let you view the directory contents in different ways. For example, `ls -a` shows all files and `ls -al` shows all files in "long" form (with additional details), while `ls -a --color` adds a splash of color to differentiate the various types of files. (For the list of parameters, use: `ls --help` or type: `man ls` for a full description.)

Q. Can I use wildcard searches with the `ls` command, as I can with Dir in DOS/Windows?

Sure. You can search for all files ending with the **.txt** extension using: `ls *.txt`, or all files starting with "mc" and with any extension, using: `ls mc*.*`. If you are looking for a file among a long list of files with similar names, you might limit the wildcard to just one character, using the ? symbol. For example: `ls hello?5.c` would find files called **hello05.c**, **hello95.c**, **helloj5.c** and so on, but not **hello105.c** or **hellor7q5.c**. You can even combine wildcard characters for very specific searches, such as: `ls hello?5.*`. (Wildcards can also be used with other commands, including ***cp*** (copy), ***mv*** (move), ***rm*** (remove), etc.)

*Q. I want to do a wildcard search on files with ? or * in the names. How do I do that?*

Unlike DOS/Windows, UNIX/Linux does allow those characters, as well as **)** and **(** and **|** and other special characters in file and directory names. To distinguish between a wildcard search using **?** or ***** and a search for those particular characters, you merely need to insert a backslash (****) before the wildcard character. For example: `ls hello\?.txt` or `ls myfile\*.c` or `ls mc\)127.h` would find the files **hello?.txt**, **myfile*.c** and **mc)127.h**, respectively. You can also combine these "regular expressions" with wildcards in searches, such as `ls hello\?.t?t` (which would list **hello?.txt** and **hello?.tst**), or `ls mc\)127.*` (which would list **mc)127.h** and **mc)127.c**, for example).

Q. How do I create or rename a file with special characters in the name?

As with the previous question, you merely need to insert a backslash before the special character. For example: `touch hello\?.txt` would produce an empty file called **hello?.txt**; `touch myfile\*.c` would result in **myfile*.c**; and `touch mc\)127.h` would create **mc)127.h**. Similarly, `mv hello.txt hello\?.txt` would add a **?** to the filename, immediately before the dot. To use multiple special characters, simply insert a **** before each one; for instance: `touch myfile\(\*\*\).txt` would produce **myfile(**).txt**. The procedure is the same for any other special character. If you are not sure which characters require the **** treatment, simply try the command without it first. If you receive a *"sh: syntax error near unexpected token 'yourfilename.'"* message (where 'yourfilename' is the name you tried to create), then you need the backslash.

Important Note: Although you can include many special characters in this manner, *using* some *special characters, including the asterisk (*****), question mark (**?**), hyphen (**-**), vertical bar/piping symbol (**|**), circumflex* or *caret (**^**), apostrophe (**'**) and ampersand (**&**, in a file name is generally not a good idea*, because they may be confused with wildcard, parameter ("argument") and other symbols by various commands. In addition, using a space in a filename (such as **my file.txt**) may cause some commands to treat the name as two filenames, or a parameter and a filename. (To avoid this problem, enclose the name in quotes, as in **"my file.txt"**.) Any of these conditions will cause unintended results. At the very least, these special symbols can make filenames more difficult to read. Unless you have a good reason for doing so, it is probably best not to use them.

Q. How long can a filename be?

A filename can be up to 255 characters long, including numbers, letters and special symbols, as well as a combination of upper and lower case (which are treated as different characters by Linux/Unix). However, for the sake of readability and to make it easier to type from the command line, much shorter names are preferable. This 255-character limit does *not* include the full path. In other words, if a file called **ThisIsMyVeryLongFilename.txt** is in the **/user/mark/garbage/files** directory, the *filename alone* can be up to 255 characters, excluding the characters in the pathname. To illustrate, out of the following path:

`/user/mark/garbage/files/`**`ThisIsMyVeryLongFilename.txt`**

only the 28 characters (bolded) following the final **/** comprise the actual filename (including the dot before "txt").

Q. The directory listing is too long and scrolls off the page. Is there a way to view a page at a time?

Yes. There are programs, called "pagers" to do this. Just as in DOS/Windows, where you could use: `dir c:\ | more` to list the contents of the root directory, in Linux you can type:

```
ls –a / | more
```

Or, using an alternative utility:

```
ls –a / | less
```

Using ***less***, press the **Spacebar** to move ahead a page at a time; to back up, use the **B** key. When you are done, press **Q** to quit.

Q. Is there a way to find files from a command shell?

Just as Windows has a *Find* feature, Linux has the ***locate*** command. It will find data files, programs, directories and other objects whose names match your search argument. For example, `locate myfile` will find **myfile.txt**, **/myfile** and other matches. (For more on ***locate***, including search parameters, use: `man locate`.) ***Locate*** searches a database (called **slocate.db**) on your hard drive used for this purpose, which is updated overnight. If you are searching for a new file that ***locate*** is unable to find (or you regularly shut the system down overnight, so the database doesn't get updated often), you can force a manual database update. To do this, use the: `su` command to temporarily login as root, then from the shell prompt type: `updatedb`. Wait a few minutes for the database update to complete and try the ***locate*** command again.

Note: Linux does also have a ***find*** command, but for performance reasons ***locate*** is better for searching on filenames. If you have a need to search by file date or other attributes or would like to automatically execute a program using the found file as a parameter, or any of a large number of other interesting uses, ***find*** is the command to use. For more on ***find***, use: `info find`.

As an alternative, if you are merely looking for the binary, source and manual pages for a command, you might try the ***whereis*** command. To illustrate the difference between ***whereis*** and ***locate*** try the following two commands: `whereis passwd` and `locate passwd`. As you can see, the ***whereis*** command found only a few files, while the ***locate*** command listed dozens. That is because ***whereis*** was looking for a few specific files (all called **passwd.***), while ***locate*** was busy seeking any file or directory name containing "*passwd*" in the name (including **gpasswd.1.gz**, **autopasswd**, **passwd-1.html**, etc.) For another example, try: `whereis *.doc` and `locate *.doc`. As before the ***locate*** command came up with many files sporting a **.doc** extension. The ***whereis*** command, however, found nothing. Why? Because there is no Linux command called "**.doc"* or even *"doc"*. Remember, ***whereis*** works *only* on files associated with commands, not just any old data file or program.

Q. Is there a way to browse through a text file from the command line?

Yes. Perhaps the best way is to open the file with a text editor, such as Pico, Emacs, Vi or others, and scroll or page through the file. However, if you are trying to do the browsing from the command line without opening the file first, you can use the ***head*** and ***tail*** utilities, or the ***more*** or ***less*** commands. For example, `head myfile.txt` would display the first 10 lines of the file onscreen, by default. The command: `head –40 myfile.txt` would show the first 40 lines. (You can substitute larger or smaller numbers for "40".) Similarly, ***tail*** displays the last 10 lines of a file, as in: `tail myfile.txt`. (If the file is short, but you are not sure how many lines long, you could use the ***cat*** command (short for concatenate) to display the entire file on the screen, such as: `cat myfile.txt`.)

Obviously, this is fine if you know approximately how close to the beginning or end of a file the text string is, but not if it is somewhere in the middle of a large file. In that situation ***more*** or ***less*** would be more productive. (See the question, *The directory listing is too long and scrolls off the page. Is there a way to view a page at a time?* in this section, for information on those commands.)

Q. Is there a command to search text files for a specific character string?

Yes. Naturally, you can open a file with a text editor and use its search function. However, if you prefer to search from the command line, you can use the ***grep*** utility. It is a very powerful text file search tool. You can perform wildcard searches on a specific file, but more importantly, *you can search many files at once*, also with wildcards. For example, say you have a directory full of text files and you know that one of them contains the phrase, "Mary had a little lamb", but you do not know which file it is. Using the command: `grep "Mary had a little lamb" *.txt` might result in the following output:

```
myfile.txt: Mary had a little lamb. Its fleece was white as
snow. And everywhere that Mary
```

(Note the use of quotation marks around the phrase to identify the search string when multiple words are involved.) The ***grep*** command shows the entire line of text containing the search string to make it easier for you to determine whether it is the correct text (because there may be multiple matches).

Note: The ***grep*** utility is by default case-sensitive, so searching for "Mary" would *not* find mary or MARY. If you wish to perform a case-insensitive search, add the `–i` parameter in the preceding example. Then it will find Mary, mary, MARY, mARy, or any other mixture of case.

For example, a case-insensitive wildcard search such as: `grep –i mar* *.txt` might yield this result instead:

```
myfile.txt: Mary had a little lamb. Its fleece was white as
snow. And everywhere that Mary
personnel.txt: Candidates for a pay increase this year include
Johnson, Martin and Littlefield.
personnel.txt: Due to Smith's impending marriage, she will be
taking a two-week honeymoon in
```

Not only were there "hits" in two different files, but there were two separate occurrences of "mar" in the **personnel.txt** file as well.

The results of a search can even be copied to another text file or printed, using the piping symbol (**|**). (See the question, *The directory listing is too long and scrolls off the page. Is there a way to view a page at a time?* in this section, for information on piping.) Search results can also be redirected to a file using the redirection and append symbols (**>** and **>>**). (See the question, *Is there online documentation for most programs?* in the *General Usage* section, for information on redirection.)

There is much more that can be done with the ***grep*** command. For example, the **–c** parameter tells ***grep*** to display only the *count* of the number of times a string appears in a file, rather than the *text* of the matches (such as `personnel.txt:2`). For more information, read the manual page (command: `man grep`).

Q. What are "symbolic" and hard links, and how can I identify them?

Symbolic (soft) links work much like "shortcuts" in Windows, as pointers to programs, data files or directories located elsewhere. Links allow you to start a program from more than one folder (or the Desktop) without having to duplicate the files in multiple places. Links are independent files—only a few bytes in size—that can be renamed or moved without affecting the original programs or files they point to. Conversely, if the original file is moved, deleted or renamed it *will* break the link, because it no longer matches the filename and location pointer stored in the link. (If a link is broken this way, it should be deleted—as it is no longer useful—and a new link created, if needed, for the new filename and/or location. On the other hand, if you rename the original file back to what the link is expecting, or return it to its original location, the link will "magically" work again.)

Symbolic links can be identified, using the K File Manager (kfm) or other graphical file manager, by a small arrow in the lower right corner of an icon, pointing at the icon. From the command line, one command you can use is: `ls -F` (capital F, not lowercase). When the filenames are listed, the symbolic links will be followed by an @ symbol. For example, if you were to list the files in the **/bin** directory, you might see:

```
arch*
ash*
ash.static*
aumix-minimal*
awk@
basename*
bash*
bash2*
bsh@
```

and so on, where **awk**@ and **bsh**@ are the symbolic links to files in other directories. (The * following the other names indicates that they are "original" files.) Alternatively, you could use the: `ls -il` command (see below).

Link files specific to the KDE user interface will have **.kdelnk** filename extensions, such as **Autorun.kdelnk**.

Another type of link is the "hard" link. This is similar in function to a symbolic, or soft, link except that renaming, deleting or moving the original file won't break the link (because it actually links to the physical file on disk, not just the file pathname in the disk directory structure). Hard links apply only to data files or programs, though, not to directories.

All files have at least one hard link (the actual filename in the directory listing). Adding another hard link increments the hard link count, which you can display using the: `ls -il` command. For example:

```
619880  -rw-rw-rw-   2 ownerid  groupid    428 Apr 14 19:42  myfile.txt
145389  -rw-rw-r--   1 ownerid  groupid   7285 Sep 30 10:17  coffee.txt
268394  -rw-rw-rw-   1 ownerid  groupid   8912 Nov  4 19:42  tea.txt
619880  -rw-rw-rw-   2 ownerid  groupid    428 Apr 14 19:42  person.txt
```

(The **–i** parameter says to display the "inode" number—explained below, and the **l** argument tells Linux to show the directory list in long form, including the permissions, owner and group Ids, file sizes, and creation date and times.) There are four different filenames listed, but notice some key similarities between two of them: the files **myfile.txt** and **person.txt** have exactly the same date, time, file size and properties. In addition, they both show "2" for the number of hard links, but all of these similarities *could* just be a coincidence (after all, two unrelated files of the same size and type could have been created within a minute of one

another). The real clincher, however is in the set of numbers preceding the file permissions listings. (For an explanation of file/directory permissions, see the question, *How do I display the permissions for a file or directory?*) The "619880" number is the inode, or physical disk address of the file. Only one file can occupy a given block of disk storage (just as only one car can occupy a particular parking space at any one time). Therefore, these "two" files must in fact be only one physical file with two hard links (filenames) to the file.

Again, unlike soft (symbolic) links, deleting either of these two files doesn't affect the other. Another difference between the two types of links is that hard links can't span disk drives. In other words, a hard link can't be created on **/dev/hdb** to link to a file on **/dev/hda**.

Q. How do I create links (symbolic or hard)?

The ***ln*** (link) command can be used to create both types of links. For example, to create a hard link called **myfile2.txt** to a file called **myfile.txt**, type: `ln myfile.txt myfile2.txt`. This will create the link in the same directory as the original file. To create a symbolic link called **myfile3.txt** to the same original file, use: `ln –s myfile.txt myfile3.txt` (the **–s** parameter is what tells Linux to make the link symbolic).

If you use the: `ls -il` command to list the directory contents, you should see something like the following:

```
619880  -rw-rw-rw-   2 ownerid  groupid    428 Apr 14 19:42  myfile.txt
619880  -rw-rw-rw-   2 ownerid  groupid    428 Oct  2 10:18  myfile2.txt
224516  lrwxrwxrwx   1 ownerid  groupid      6 Oct  2 10:18  myfile3.txt -> myfile
```

Note that all details of the hard link (**myfile2.txt**), including the inode address (619880), are identical to that of the original file (**myfile.txt**) except for the creation date and time. It is treated as a physical file (as verified by the file size and the permission type of **-**), but at the same location as the original file (same inode address) with the same read/write permissions. The **2** following the permissions reflects the number of hard links to the physical file (the directory listing for the original file being the first hard link). Conversely, virtually every aspect of the symbolic link (**myfile2.txt**) is different. The inode address differs because there is a physical file created elsewhere on the disk that contains a pointer to the original file. Likewise the permissions differ, with the symbolic link showing a file type of link (the **l** leading off the permissions) and including e**x**ecute permissions for all user types. The **1** following the permissions demonstrates that this is not a hard link. Finally, the filename is followed by "**-> myfile**", which reveals the name of the original file to which it points.

Note that creating a hard link for this file in the same directory probably would not be the normal usage of a hard link. More likely, you would create it in another directory so that you could access the file from either place (or several places, if you make multiple links). For example: `ln myfile.txt /user/mark/myfile2.txt` would put the new link in the **/user/mark** directory.

Linux itself makes use of hard links. In many cases Linux continues to use programs and files that were first used in earlier versions of Linux, which can sometimes cause problems. For example, if the X Window System was first provided in a directory called **/usr/X11R4**, Linux software which used X Windows would look for those files in that directory. But when a newer version of Linux comes out that uses a later version of X Windows, the files might be stored in **/usr/X11R5** or **/usr/X11R7** or **/usr/X11R10**. So how to keep from breaking all those programs that are expecting the files to be stored in **/usr/X11R4**? Simple. Put hard links from one directory into the other. This way, whichever directory the application software looks in for those files, it will find what it needs.

As an alternative to the command line interface, some graphical file managers (such as GNU Midnight Commander), offer options to create either or both types of links by right-clicking on a file icon.

Q. Can you summarize the differences between symbolic and hard links and differentiate between using links and simply duplicating a file?

Sure. If you merely copy a file from one directory to another, you have doubled the disk space used by the file, and editing one file doesn't affect the other (because they are in no way connected to one another). This is fine if it is what you intended (so that you can keep a pristine original while modifying another copy, or if you made a backup copy for safekeeping.) If you copy a directory instead of a file, you also copy the entire directory's contents, which can result in a lot of disk space used. A file or directory copy can be put anywhere on any disk, either directly attached to the system or on a network or Internet/Intranet drive somewhere else entirely. Because the original and the copy are unconnected, deleting one has no effect on the other.

On the other hand, if you create a symbolic link it takes only a few bytes of space and can be located on any mounted device that you can type in a path for. If you delete, rename or move the original, you break the link (although recreating the original in the same place will restore the link). Using the link to access the file or directory, any changes you make to the file or to the directory contents will affect the original files/directories, because those are what you are actually touching.

Alternatively, if you create a hard link instead, no additional disk space is used, however the link can only be created on the same physical hard disk (but multiple partitions on that disk), not on other local or remote devices). Moving or renaming the original file won't hurt the link; it will reflect the change. When you "delete" the original file, you are actually deleting only the original hard link (the directory name). So, as long as at least one other hard link exists, the file is still accessible. Hard links can be created for files but not for directories. Because the permissions indicate that a hard link is a physical file, any software that tries to access these files will think it is accessing the original files (which, in fact, it is—through the link).

Q. I recognize a number of file types, such as .JPG, .GIF, .WAV, .TXT, .HTM and .ZIP from Windows, but I see many other file types I don't know. What are they for?

There are a number of file extensions that are common in Linux/UNIX but not typically seen in the Windows world. Here are some of them:

.a – Archive file	**.png** – Image file (like .jpg/.gif)
.au – Audio file (like .wav)	**.ps** – PostScript printer-formatted file
.bzip2, .bz2 – Archive file compressed with ***bzip2***	**.rpm** – RPM Package Manager file
.c – C programming language source file	**.so** – Program library file
.conf – Configuration file	**.tar** – Archive file
.cpp – C++ language source file	**.tar.bzip2, .tar.bz2** – Archive file compressed with ***bzip2***
.gz, .gzip – File compressed with ***gzip***	**.tar.gz, .tar.Z** – Archive file compressed with ***gzip***
.h – C or C++ language header file	**.tga** – Image file
.html – HTML file (same as .htm)	**.tgz** – Archive file compressed with ***gzip***
.o – Program object file	**.xpm** – Image file
.pl – Perl script file	**.z, .Z** – Older ***gzip*** file

If you encounter a file type you do not recognize (such as one with no file extension), you can use the ***file*** command to find out more about it; for example: `file smith.xyz` or `file smith` (if there is no file extension).

Q. Should I back up my entire system?

That is not as easy a question to answer as it may sound. A fully configured system with operating system, many large applications, data files and perhaps a library of multimedia files (music, video, photographs, etc.) can amount to many gigabytes of information. Unless you have a high-capacity, high-speed backup option (such as a 4mm, 8mm or DLT tape drive, or an Iomega Jaz removable disk cartridge, for example), it just is not practical to back up everything. On the other hand, because you probably installed all your software from a few CDs, it is not absolutely necessary to back up the operating system and applications. If you back up only your data files, setup files, account files and others that have been created or modified since installation, you can always reinstall the software from CD, then restore the backed up user files.

Of course, the disadvantage to this, in the event of catastrophic disk failure, is the time and effort needed to install all that software individually, rather than just starting a restore program, inserting a tape or Jaz cartridge and walking away. But if you do not have a high-capacity backup solution at your fingertips, a partial backup is the most effective method for all but a complete system rebuild. Each application will have its own set of configuration files that should be backed up, as well as data files, but you will have to find out what they are from the software documentation or the vendors.

From a Linux standpoint, the directories you should be sure to back up include:

- **/etc**
- **/home**
- **/opt**
- **/usr/lib**
- **/usr/local**
- **/usr/src/linux** (if you have modified any of the kernel programming)
- **/usr/X11R6/lib/X11** (if you are using X Windows)—Case is important in this folder name!
- **/var/spool/mail**

Of course, you should also back up any directories containing important user data files. If you lack large-capacity storage, you can always back up each application's files to its own set of floppies using an archiving/compression utility, such as ***tar***, ***gzip***, ***bzip2*** or ***zip***. These and other tools can span more than one disk or tape to store all the data in one related archive. This method may not be as quick or efficient as a tape drive, but at least you will have a copy of your essential files. It is not necessary to back up the program files themselves, as you can always just reinstall the applications from scratch, but you might want to back up the files that contain the user setup information (preferences), such as default margins and tabs for a word processor. (One clue as to which files these are is to look at the file dates. Generally, the user setup files will have much newer dates than the files created when the program was first installed. If in doubt about what to back up, contact the software vendor for the filenames.)

Q. How often should I back up my files?

There is no one right answer to this question either. Every user is different. One way to answer the question for yourself is to use the "ouch" test. Consider how much work you could

afford to lose before it hurts. If you are a home user, perhaps a once-a-week backup would be adequate (right after you pay bills, for example). In a high-performance office environment, once or twice a day—even hourly—may be required.

It is also a good idea to use a "grandparent-parent-child" strategy of rotating backup media. (In other words, use three *or more* sets of tapes or cartridges and alternate among them.) This way, if for some reason your most recent backup is corrupted, at least you have the next older one to fall back on. Plus, if you discover that you accidentally deleted a file a few days ago—or it was damaged by a virus—you can restore the file from a backup made before the deletion.

Another option is to back up only what has changed since the last backup (some archival programs track this information from backup to backup). The advantage is that often very little has changed since the last time, making for very quick backups. The disadvantage is that you must keep copies of all the intervening backups since the last complete archive. Losing even one partial backup along the way can result in the loss of important information. Thus, even with a policy of incremental backups, it is a good idea to do full backups periodically.

Q. What backup software should I use?

There are a large number of commercial and free programs available for use, some primarily for tape drives and others for more general usage (magnetic and optical disks, network drives, etc). Many of these have graphical interfaces to make them easier to use. It is a good idea to try out a few different programs to see what you like best.

There are too many available tools to try to describe them all here. But there are a few common commands and utilities—likely to ship with your Linux distribution—that you can start with:

tar — ***tar*** was the earliest and most common "packaging" tool for Unix/Linux (it does the *un*packing of files as well—there is no "untar" command). The ***tar*** command creates a file that contains all the other files and directories, optionally in a compressed state (using ***gzip*** or ***bzip2***), to save space. If you have used ***zip/unzip*** or PKzip with DOS/Windows, then you are already familiar with the concept. Files packaged with ***tar*** typically end with a **.tar** extension. If also compressed with ***gzip***, the extension may be **.tar.gz**, **.tar.Z** or **.tgz**. ***Tar*** files compressed with ***bzip2*** generally have a **.tar.bz2**, **.tar.bz, .tbz2 or .tbz** extension. The ***gzip*** and ***bzip2*** compression utilities can also be used independently of the ***tar*** command to compress and decompress files (see below). There are many different parameters available for the ***tar*** command (which does the unpacking of files as well), so it is best to read the manual page for ***tar*** using the command: `info tar`. ***Note:*** It is *not* advisable to compress ***tar*** files on a tape backup, because ***tar*** would not be able to recover from tape errors.

gzip/gunzip — ***gzip*** (GNU zip) is the tool traditionally used for compressing tarred files and its companion ***gunzip*** is used to decompress those files. More recent tools, such as PKunzip, can generally uncompress files created by ***gzip*** as well as those compressed by other tools, such as PKzip and ***zip***. Due to this flexibility and because PKzip and ***zip*** are also available on other operating system platforms, they have become more popular than ***gzip***. Files compressed with ***gzip*** usually end with **.gz**, or in the case of some older files, **.z** or **.Z**. You can read more about the ***gzip***/***gunzip*** commands using: `info gzip`.

bzip2/bunzip2 — Think of ***bzip2*** as a newer, faster version of ***gzip***. It has a very similar command structure (including many of the same parameters), but it tends to be faster at compressing and decompressing than ***gzip***/***gunzip***. Bzipped files end with a **.bz2** or **.bz** extension. Learn more about ***bzip2***/***bunzip2*** from: `info bzip2`.

PKzip/PKunzip — PKzip is a popular shareware/commercial packaging and compression tool, primarily in the DOS/Windows world. PKzip is very fast and offers varying degrees of

compression vs. speed. Newer versions include a graphical front user interface. PKunzip is compatible with files created by PKzip, ***zip***, ***gzip*** and ***tar***, as well as the UUencode, XXencode, MIME and BinHex file formats. PKzip files end with a **.zip** extension. For more information, read the documentation that comes with PKzip.

zip/unzip — ***zip*** and ***unzip*** are a pair of command-line tools that will package and compress (and decompress) files and directories. In addition to being very fast, they are free and available for other platforms, including DOS/Windows, OS/2 and other UNIX OSes, including SCO. The ***unzip*** command can also unpack files created by PKzip. Like PKzip, ***zip***-compressed files end with a **.zip** extension. The ***zip*** and ***unzip*** utilities may be included with your Linux distribution; if so, read their manuals using the following commands: `info zip` and `info unzip`.

Other utilities you can obtain separately include **Arkeia**, **Backup Edge**, **BRU2000**, **Catchup**, **CTAR**, **Drive Image**, **Kbackup**, **KDat**, **PerfectBACKUP+**, **Sitback**, **Taper** and many more.

For a lot more on this topic, see the related white paper entitled, *Installing Linux Applications for the First Time*, available from the same sources as this paper.

Root/User Accounts, Groups and Permissions

These are questions revolving around administering accounts, passwords and permissions.

Q. Sometimes instructions say that I must "be root." What does this mean?

It means that you must be logged in as the root user, or *superuser*. (Sorry, but you do not get a cape with a big S on it!) There are two ways to do this. If you will be working as root extensively, you should logout as yourself and login as root. When you are done, reverse the process and log back in as yourself.

On the other hand, if you need to be root only long enough to execute a command, you can open a command shell and issue the command: `su` (substitute user) to temporarily login as another user, while still inside "your" login shell. You will be prompted for the root password. Once that is provided you will "be root" and can issue commands as needed until you end the root session by typing: `exit` and pressing **Enter** or by closing the shell.

Some commands won't work unless you have logged in as root (i.e., `su` won't work, because Linux "knows" it is still you in charge). An alternative is to use the ***su*** **–** command (`su` followed by a hyphen, with a space separating them). This makes you root with the root's login shell, as if you had logged off and logged back in as root.

Q. What is the root account and how does it differ from a user account?

When you installed Linux, you were asked to provide a root password. Later you were prompted for a user account name and password. The root account is equivalent to the system-level access you are given under Windows, with the ability to delete, create and format partitions, create/delete directories and files, or do anything else your heart desires. Linux operates like a network client/server system. The root account is like a system administrator, able to control what access authority the various clients (users) have. There can be many different user accounts set up on the same Linux system, each with its own set of permissions, configuration options, choice of user interface and so on.

This is very useful in a business environment where multiple people share the same system, as each user account can be tailored for each individual user. On the other hand, it can be a

nuisance when there is only one user, because there are some system functions that can be performed only by the root account, necessitating that the user log out of the user account and log in to the root account long enough to perform the needed operation, then log out of the root account and back into the user account. (Fortunately there is a shortcut you can use in some situations as an alternative. See the description of the ***su -*** command in the earlier question *Sometimes instructions say that I must 'be root'. What does this mean?*.)

As compensation, however, doing things this way makes it much less likely that the user will accidentally delete or corrupt important system files that they have no business playing with in the first place.

Important Note: Most, if not all, Linux distributions require that you create a user account in addition to the root account when you install Linux. While it is tempting to log on as root and use the root account for everything, this is a ***V**ery **B**ad **I**dea*. There are too many ways in which you can accidentally (or someone else can maliciously) corrupt up a system from the root account. By running mostly from the user account, your system is protected to a great degree from most disasters. (It won't prevent a disk crash, of course, but at least you can't accidentally delete your **/usr/lib** directory, for example.) If the Linux distribution you installed required you to create a user account, *use it*. If you did not create one during installation, create one now—and *use it!*

Q. Why would I want to add other user accounts to my system?

If you are the only one who uses your system, there may be no need to set up other user accounts (besides your own). On the other hand, if you share your computer with other people (second and third shift workers, perhaps; or you share a common work area; or other members of your family use your computer), or even if you wish to make your printer or other devices available to others on the same network, you will need to set up *individual* user accounts and possibly *group* accounts.

Q. How do I add users?

There are a number of tools available for doing this. For instance, **Red Hat** Linux 6.2 includes the graphical Red Hat Disk Management tool (***linuxconf*****)**. For occasions where you are not running a graphical interface, or prefer a command line tool, a couple of text-mode programs that are often included with Linux are ***adduser*** and ***useradd***. Despite the similarity of their names and purpose, ***adduser*** is the more functional of the two.

To use ***linuxconf***:

1. Run: `linuxconf` from a command line under either KDE or Gnome. (***Linuxconf*** is *not* located on a menu anywhere.)

   In order to use Linuxconf and similar system-level tools, you will need to first be logged in as root. To do this, use the ***su –*** command and enter the root password when prompted.

2. Once the ***linuxconf*** panel opens, scroll down to the heading called *Users accounts*. Immediately below, there is a subheading called *Normal*, and beneath it *User accounts*.

3. Double-click on *User accounts* to open a panel that lists all existing accounts.

4. To create a new account, click on the *Add* button at the bottom of the panel. You will see a subpanel with four tabs: *Base info*, *Params*, *Mail settings* and *Privileges*.

5. When you are done filling in the necessary information, press the *Accept* button to save your changes (or *Cancel* to reject the changes and not create a new account).S

Users of other distributions besides Red Hat have other graphical user administration tools at their disposal. For instance, **Caldera OpenLinux** 2.3 users have *Webmin* and *COAS*. Although neither individually can do everything ***linuxconf*** can do, together they offer similar functionality.

To add users with the *COAS* tool:

1. From the KDE menu, select *Settings*, then *COAS*, then *System* and finally *Accounts.* A panel will appear.
2. On the tool bar, select *User* and then *Create user.*
3. When the menu appears, begin customizing as many users as you need.

If your distribution doesn't offer a graphical tool, or if you prefer to use the command line interface, you can use ***adduser***. As with ***linuxconf***, before using ***adduser*** you must login as root. Then type: `adduser` *`newusername`* (where *newusername* is the user name you wish to create).

The program will assign the next available ID for both the user and default group, then remind you to set a password for this new account. (To add the password, see below.) This will supply the minimum information necessary to create a user. To add personal information to the account record, such as the user's name, address and phone number, use the `chfn` (Change Finger Name) command (see below).

Q. How do I delete users?

This is a multistep process:

1. Log in as root.
2. Use the ***userdel*** command with the name of the user you wish to delete, as in: `userdel mary`. Note, however, that while this deletes the user account name from the **/etc/passwd** file, it doesn't delete all associated files.
3. Follow up with the ***groupdel*** command to remove the user's UPG entry (the default group), such as: `groupdel` *`groupname`*.
4. Remove any other group affiliations for the user (besides the UPG), using: `usermod –G` *`username username`*. (Note that the `–G` parameter is capitalized.)
5. Use a text editor to remove the login ID from the **/etc/group** file.
6. Delete the user's home directory and mailbox using: `rm –R /home/`*`username`* `/usr/spool/mail/`*`username`*. (Note that `–R` is capitalized.)

Q. Is there any way to change a user ID?

Yes. To change a user ID (login name), use the ***usermod*** command. For example, to change an account name from "*mary*" to "*Mary_Jones*", use: `usermod –l Mary_Jones mary`. If you change a login name, you should also change the user home directory, where all the user account information is stored. To do this, use the command: `usermod –d /home/Mary_Jones –m mary` to move the user account information to the directory **/home/Mary_Jones** from **/home/mary**. (The **–l** parameter tells ***usermod*** that you are changing the login name; **–d** identifies the directory you are changing; and **–m** means to move the contents to the specified directory.)

Q. Is there a way to disable a user account?

Yes. If you are using "shadow" passwords (see the question *What is a shadow password?,* in this section, for an explanation), the simplest way to disable an account is via the ***usermod*** command. Merely assign an expiration date that has already passed, such as: `usermod -e 01/01/92 mary`. (See below for an explanation of shadow passwords.)

As an alternative, *experienced* users can edit the **/etc/passwd** file. To disable the account, find the line for the appropriate user and insert an asterisk (*****) in front of the encrypted password. The password is located between the first and second colon separators, following the user name. For example:

```
mary:*Ms.2jbeTWmUYi:500:Mary Jones:/home/mary:/bin/bash
```

Warning: This alternative approach should be used only as a last resort, by a proficient Linux user, because manually editing this file can cause major problems if not done correctly!

Q. How do I add or change personal information in an account?

If you wish to add name, address and telephone number information to your account or change information already there, there is the ***chfn*** (Change Finger Name) command. You will be prompted for your password, then shown the data fields for the information above.

To change *other* user accounts, login as root. Then specify which user you wish to change the information for, such as: `chfn fred`, and make the additions or changes.

If you prefer a graphical tool, **Red Hat** Linux 6.2 users running KDE can click on the KDE menu (the big K on the Panel) and select the *Red Hat* option, then *System* and finally *About Myself*. This will bring up a panel with entry fields for all the information accepted by the ***chfn*** command. I am not aware of any such graphical tools for desktop environments other than KDE, so if you are not using KDE you will have to use the ***chfn*** command line option.

Q. How do I add groups?

Default groups are automatically created when you add users. Should you need to add groups manually you can do so with the ***groupadd*** command. Normally, group IDs (GIDs) are automatically supplied (by adding 1 to the last number used) when a group is created. However, because you are creating a group manually you will need to specify a group ID yourself. The maximum GID number is 65536, so you might want to specify something very large (say, 50000 or higher) to keep from possibly running into one already assigned by the system. To create a group called "*mygroup*" with a GID of 50000, type: `groupadd -g 50000 mygroup`.

Q. How do I add/delete users in a group?

To add users, use the ***passwd*** program with the **–a** (add) parameter. For example, to add user "*mary*" to group "*mygroup*" type: `gpasswd -a mary mygroup`.

To delete users, simply replace the **–a** parameter with the **–d** (delete) parameter, such as: `gpasswd -d mary mygroup`.

Q. How do I change a password?

To change your own account password, **Red Hat** 6.2 users running KDE can click on the KDE menu and select *Red Hat*, then *System* and finally *Change Password*. This will bring up a panel prompting you for the current password (for the logged in user). If the correct password

is supplied, you will be prompted for a new password to use. (If it is too short or invalid for some other reason, the prompt panel will reappear.) I am not aware of any graphical password management tools for desktop environments other than KDE, so if you are not using KDE you will have to do it from the command line.

From the command line, simply use the ***passwd*** command. It will prompt you for your current password and then the new password twice, for verification.

To change another password, you must be logged in as root. Then tell the program which user you wish to change the password for, such as: `passwd fred`. You won't be required to know the current password; simply enter the new password *twice*.

Q. What is a shadow password?

Shadow passwords are used to protect system passwords (for user accounts) by making the file containing those passwords (**/etc/shadow**) readable by the root operator only. When shadow passwords are used they replace the encrypted password in the **/etc/passwd** file with asterisks. (Moving the passwords to **/etc/shadow** makes it less likely that the encrypted password can be decrypted, because only the root operator has access to the file.) To see if you have shadow passwords enabled on your system, use the command: `ls /etc/shadow`. If you receive a message like: `ls /etc/shadow: No such file or directory`, the files have not yet been installed. For more information on shadow passwords, read the manual page: `man 5 shadow`.

Q. How do I display the permissions for a file or directory?

From a command line, use the ***ls*** command with the **–l** parameter. For example, `ls –a –l *.txt` would display all .txt files in the current directory, with their permissions, such as:

```
-rw-rw-rw-   1 ownerid  groupid     1428 Apr  2 19:42  myfile.txt
-rw-rw-r--   1 ownerid  groupid      914 Sep 30 10:17  personnel.txt
```

The first 10 characters (or "switches") on each line contain the permissions for that particular file, directory, folder or other object. For an explanation of what the characters mean, see the question, *When I look at a list of files and directories, the names are followed by something like -rw-rw-r-- or lrwxrwxrwx. What does this mean?*

If you are using the K File Manager (KFM) under KDE (just click on the Panel icon that looks like a house in front of a manila folder), simply select *View*, then *Long View*, from the pull-down menu to display all objects with their permissions. Gnome users will have to use the command line method.

Q. How do I change permissions?

If you are the system administrator (root), or the owner of a file, directory or other object, you have the authority to grant or remove permissions for other users and groups, using the ***chmod*** command. The operative symbols to remember are **u** for **U**ser (owner), **g** for **G**roup, **o** for **O**ther, **a** for **A**ll, **r** for **R**ead, **w** for **W**rite and **x** for e**X**ecute. Precede the **r**/**w**/**x** characters with a plus sign (**+**) to *add* permission or a minus sign(**-**) to *deny* permission.

For example, to *grant* **R**ead/**W**rite access for the file **myfile.txt** to **G**roups and **O**thers, use the command: `chmod go+rw myfile.txt`. To *remove* **W**rite and e**X**ecute access from all accounts, use: `chmod ugo-wx myfile.txt` or `chmod a-wx myfile.txt`. (Note that **a**, for ***A****ll*, is a convenient shortcut for **ugo**.)

Communications

These questions relate to configuring or using communications features of Linux.

Q. How do I set up my system for dial-up Internet access?

This is a multipart process. Before you begin you should contact your Internet Service Provider (ISP). There is some information you will need to provide to Linux that only your ISP can provide. (A technician who is conversant with Linux can probably step you through the entire process, but because you may not find a Linux expert at your ISP, here are the steps necessary to do it yourself.)

- If you are using KDE, use the Kppp utility to start the configuration process.
  - Log in as the root operator, then use your favorite text editor to open the **/etc/ppp/options** file. If you see a statement with the option "lock" you will need to "comment it out" to disable it. (Kppp handles locking the connection and the hardware, so Lock is redundant.) To disable the lock option, insert a # symbol in front of it, such as: `#lock`. Save the file and exit the editor. Log out as root and log back in as yourself.
  - To start Kppp, go to the *KDE Application Starter* menu (the big K icon on the Panel) and select *Internet*, then *Kppp*.
  - When the *Kppp Configuration* panel appears, press the *Setup* button to create a new ISP connection record.
  - From the *Accounts* tab, press the *New* pushbutton.
    - ✓ When the *Edit Account* panel appears, you will be in the *Dial* tab. Enter whatever you want to use as the ISP name where it says *Connection Name*. Enter the dial-up connection *phone number* in the next field (including area code, or anything needed to dial out of the switchboard, such as **9,**). The default *authentication type* is *PAP*. Use that unless your ISP tells you otherwise. (The remaining fields are for experienced users. Unless you have a reason to enter information in those fields, leave them blank.)
    - ✓ Click on the *IP* tab. When that panel appears either choose *Dynamic IP Address*, or enter a static address. Your ISP will have to tell you which option to use, but generally the ISP will assign a temporary IP address each time you connect, so most likely the ISP will tell you to select Dynamic IP Address. If you are setting up a connection inside a firewall, you will probably be using a static address. The administrator in charge of the firewall will assign you a static address and tell you the subnet mask to enter in the appropriate fields. Use the "Auto-configure hostname from this IP" checkbox if you are connecting to a DHCP server or a Bootp server. (Your network administrator will tell you if you need this.)
    - ✓ If you are connecting from home, you can skip the *DNA* and *Gateway* tabs. If you are connecting in an office environment, your network administrator will tell you if you need to enter anything in those tabs. Unless told otherwise, you should not need to enter anything in the *Login Script* tab either. If you live outside the United States or Canada and your ISP charges by bytes transferred, use the *Accounting* tab to select the country, the company and whether you are charged by incoming data, outgoing, or both; otherwise ignore it. Click the *OK* button to close the *Edit Account* panel and return to the *Kppp Configuration* panel.
  - Click on the *Device* tab to set up your modem connection. If the modem was set up correctly during installation, you should have to select only the port to use. COM1 through COM4 are mapped respectively to **/dev/ttyS0** through **/dev/ttyS3**. Use the one that corresponds to the port to which your modem is connected. An internal

modem may be mounted as **/dev/modem**. For *Flow Control*, use *CRTSCTS* to enable hardware flow control. For *Line Termination*, choose *CR/LF*. For a 14.4Kbps modem, choose a connection speed of *57,600bps*. For faster modems, select *115,200bps*. If you would like to prevent a second program from disconnecting one already using the modem, check the *Use Lock File box*.

- In the *Modem* tab, you will configure the modem. Use the *Busy Wait* box to specify how long to wait for a busy modem or dirty phone line before disconnecting. The *Modem volume* slider lets you decide how loud to make the connection tones. Use the *Modem Commands* button to set the default control codes for certain functions. The *Query Modem* button can be used to test the modem, and the *Terminal* pushbutton pops up a mini-terminal for entering control codes manually.
- Click the *OK* button at the bottom of the panel to close the *KPPP Configuration* panel and return to the *KPPP* panel.
- In the *Connect to* field, select the ISP for which you just created a record. Then enter the *Logon ID* and *Password* you were assigned. Make sure the *Show Log Window* box is checked. This will let you see the progress of the logon process, which is helpful in case there are problems logging on. Then press the *Connect* button to dial the ISP. If you are unable to connect, verify all of the information you entered in previous steps. If you do not find anything wrong, contact the ISP or network administrator for further assistance.

- If you are using Gnome rather than KDE, there is nothing packaged with the standard Linux distributions, but there is a downloadable tool you might try called Gnome Dialup. As of this writing, it is still under development but worth a try if you are looking for a graphical dial up access setup tool for Gnome. Gnome Dialup can be found at *http://tor-pw1.netcom.ca/~phantom/gnome_dialup/index.html*. Other such tools may exist as well.
- From the command line, you can use the ***pppd*** command to start the PPP daemon. This approach is not for the faint of heart, however. There are many parameters that need to be set via this command, so unless you are quite knowledgeable about Point to Point Protocol setup or know someone who is, this probably is not the best approach for you. For more on the ***pppd*** command, read the instructions at: `info pppd`.

If you need help getting connected to the Internet via your office network, contact your help desk. If you are trying to set up Linux for DSL or cable modem (non-dialup) access at home, contact your DSL or cable modem provider for assistance (and hope someone there has ever heard of Linux).

Miscellaneous

This section contains questions or problems that did not fit into any of the other categories.

Q. What is the difference between Linux and UNIX?

UNIX began as a proprietary operating system developed by Bell Laboratories in the 1960s. It eventually spawned a number of mutually incompatible commercial versions from such companies as Apple (Mac OS X), Digital (Digital UNIX), Hewlett-Packard (HP-UX), IBM (AIX®), NeXT (NeXTSTEP) and others.

Linux is an attempt to create a nonproprietary operating system for the masses. It is a free, open source, UNIX-like operating system, originally begun by Linus Torvalds in 1991. “Linux” really refers to only the operating system kernel, or core. More than 200 people have contributed to the development of the Linux kernel. The rest of a Linux “distribution” package

consists of various utilities, device drivers, applications, a user interface and other tools that generally can be compiled and run on other UNIX-based operating systems as well.

Q. Why does Linux seem so haphazard?

You may have heard the old saw that a camel is a horse designed by committee, illustrating the difficulty of getting a number of people to agree on the design of anything, and the results of compromise. If anything, Linux illustrates a horse designed by complete anarchy, with a healthy helping of Darwinian natural selection thrown in for good measure. Thousands of programmers over the years have written applications, utilities, games, device drivers, user interfaces and other software for Linux. In the absence of a central authority to dictate a mandatory "look and feel" each programmer had his or her idea of what was the best design for software and developed the programs accordingly. Over time certain software became more popular than others, which in turn spawned "work-alikes" and "near-clones," taking features of the previous products and adding to them, or offering slightly different variations.

Eventually, companies such as Red Hat, Caldera, TurboLinux and SuSE took the Linux kernel and added what they considered the best available utilities, user interfaces, etc., to form a "distribution," or packaged deliverable. Because each distribution chose from among this programming chaos, it was inevitable that the various programs selected for a deliverable looked and worked differently from one another, and that each deliverable looked and, to some extent, worked differently from other deliverables. This is the reason for the inconsistencies between programs in a deliverable. The flip side is that because the distribution companies were free to choose from among all available programs, not just those that slavishly conformed to an arbitrary style, the deliverables were able to include the very best programs—even the ugly or nonconformist ones.

Q. Where can I download or buy Linux software?

There are quite a few sources for free or commercial Linux software if your local software store doesn't carry what you are looking for. Here are just a few of them:

For free software, try:

- DLR Fresh Archive (*http://www.go.dlr.de/fresh/linux/src*)
- Freshmeat (*http://freshmeat.net;* love the name!)
- Linux Apps (*http://www.linuxapps.com*)
- Linux Archives (*http://home.linuxarchives.com/software.html*)
- Linux Game Tome (*http://www.happypenguin.org*)
- Linux Games (*http://www.linuxgames.com*)
- Linux Software Encyclopedia (*http://stommel.tamu.edu/~baum/linuxlist/linuxlist/linuxlist.html*)
- Linuxberg/Tucows Linux (*http://linux.tucows.com*)
- Slashdot (*http://slashdot.org*)

If you see some commercial software that you would like to buy, there are several Web sites in the United States with extensive Linux inventories:

- Indelible Blue (*http://www.indelibleblue.com*)
- Linux Mall (*http://www.linuxmall.com*)
- The Linux Store (*http://www.thelinuxstore.com*)

In Australia, Linux users can go to:

- Everything Linux (*http://www.everythinglinux.com.au*)

Beyond merely software, these sites also carry Linux books, and hardware devices known to be Linux-friendly, should you need to add to or upgrade your system. Some of them even offer such things as caps, mugs, neckties and other Linux paraphernalia. If you are looking for an extensive list of Linux books, try *O'Reilly & Associates* (*http://linux.oreilly.com*).

Q. Where can I go for assistance on Linux?

There are a great number of Internet sources for Linux assistance, including vendor Web sites, independent Web sites and Usenet newsgroups. Some informational Web sites include:

- Caldera Linux support page (*http://www.caldera.com/support*).
- Debian Linux support page (*http://www.valinux.com/services/support/?session_hash=7d7ccc8f28c0f42cddc98da037d7875a*).
- Gnome Web site (*http://www.gnome.org*) — Home of the Gnome desktop environment.
- KDE Web site *(http://www.kde.org*) — Home of the KDE desktop environment.
- Linux at IBM (*http://www-4.ibm.com/software/is/mp/linux*) — The IBM Linux home page, containing news, software downloads and support options.
- Linux Documentation Project (*http://www.linuxdoc.org*) — A collection of Frequently Asked Questions (FAQs), how-to documents, manuals and online magazines.
- Linux Gazette (*http://www.linuxgazette.com*) — It includes articles, columns and even comic strips.
- Linux Guide (*http://www.firstlinux.com/guide*) — A glossary of Linux and communications terms.
- Linux Journal (*http://www.linuxjournal.com*) — It contains links to newsgroups, chat rooms, online manuals, FAQs, vendor support sites, local Linux user groups and more.
- Linux Man Pages (*http://linux.ctyme.com*) — A collection of the **man** (manual) instructions for various Linux commands and utilities.
- Linux Newbies (*http://www.linuxnewbies.org*) — Help files for "newbies" (those new to Linux).
- Red Hat Linux support page (*http://www.redhat.com/apps/support*).
- Slick Penguin (*http://www.slickpenguin.com*) — White papers, case studies and other Linux business-related implementation success stories.
- SuSE Linux support page (*http://www.suse.com/us/support/index.html*).
- TurboLinux support page (*http://www.turbolinux.com/support*).

The following newsgroups can provide a significant amount of user-to-user help:

- *comp.os.linux.announce* — Announcements of new products, updates, bug fixes, etc.
- *comp.os.linux.hardware* — Support on hardware issues, including compatibility, configuration, device drivers and product evaluations.
- *comp.os.linux.misc* — A catch-all for whatever doesn't fit in one of the other groups.
- *comp.os.linux.networking* — For questions about networking hardware, software and configurations.

- *comp.os.linux.setup* — How to install and configure Linux and add-on products.
- *comp.os.linux.x* — Installing, configuring and using the X Window System under Linux.

There are other Linux newsgroups provided for other topics, such as vendor-specific questions (*alt.os.linux.caldera, linux.debian.user* or *redhat.rpm.general*, for example) and those for programmers.

For current news about Linux and article archives, visit:

- CNET Linux Center (*http://linux.cnet.com/?tag=st.ne.ni.refer.1491268*) — A Linux-specific news site. It not only has news, but also links to popular Linux downloads, product reviews, Linux events, Linux company stock quotes and other resources.
- ICE News (*http://www.nikos.com/icenews/linux.html*).
- Linux Planet (*http://www.linuxplanet.com/linuxplanet*).
- Linux Today (*http://linuxtoday.com/index.html*).
- Linux Weekly News (*http://www.lwn.net*).
- Linux World (*http://www.linuxworld.com*).
- Zdnet Linux Homepage (*http://www.zdnet.com/enterprise/filters/resources/0,10227,2186824,00.html*).

Mark T. Chapman
IBM Server Group
July 7, 2000

IBM Server Group
3039 Cornwallis Road
Dept. LO6A
Research Triangle Park, NC 27709

Produced in the USA
12-00