Partitions

Partitions are like politics - everyone has an opinion. While you can accept the default configuration that the popular distributions give you, you will probably find that sooner or later, you wished you'd made better choices. Here's the lowdown on partitions, how they work with Linux (compared to Windows), some examples of how you might want to set yours up -and why.

When you install a Linux system, one step in the process is usually to define and create partitions. As part of that step, you're asked to mount various directories to those partitions (or is it mount those partitions on various directories?) This is a far cry from Windows, where all of the wheeling and dealing with the storage devices themselves is kept hidden from the user.

And so for a Windows user, this installation business with Linux can become very confusing. Which partitions do you need to create? If you read a few books, they'll offer suggestions to set up partitions for directories like /, /boot, /var, /usr (did they forget the 'e'?), tmp and so on. But every book offers a different set of recommendations. And it seems that workstations need different partitions than servers? How do you know which partitions? And how do you know what size the partitions should be? Partitions must be dangerous - there are a lot of warnings about destroying data. And since we didn't have to do partitions with Windows, this must be mean that Linux is harder.

And what's with these directories with the cryptic names. Where do they come from? What are they used for? With Windows, we didn't have to worry any of this - the installation process created Program Files and My Documents for us.

OK, those are all good questions. Here's what's going on.

Machines and devices

Machines are comprised of devices. One class of devices is storage devices, like floppy disk drives and hard disk drives. (Hard disk drives are also referred to as 'fixed disk drives', because they're fixed in place, as opposed to being removable, although some wags in the early days said it was because they're always in the shop, being fixed..) Other types of storage devices include removable storage media like Iomega's Zip and Jazz drives, and CD-ROM and DVD readable and writable drives.

Machines usually have at least one hard disk that contains software that is used for booting the machine. This disk also contains the rest of the operating system software, some application software, and user data. In order to store software and data on the hard disk, the disk must be partitioned - divided into one or more areas called partitions - much like an area of land must be plotted and divided into one or more plots before building.

So a partition is simply an area on a hard disk. That's all. Partitions are transparent to Windows users because when Windows is installed, it automatically creates one partition for each physical hard disk drive in the machine. Since most machines just have one drive, most users end up with just one partition - named "Drive C" - upon which everything is crammed. Indeed, most users are thus not even aware of the concept of a 'partition'.

Windows handles the partitioning of a hard disk behind the scenes while it's being installed, but since it is doing so automatically, it ends up making bad choices. Windows is used by a wide variety of users, from occasional hobbyists all the way to mission critical business services. But it uses the same default partitioning layout for everyone, and thus must use the lowest common denominator for everyone - one big ol' partition for everything from the boot software to drafts of last year's company picnic memo.

The various distributions of Linux, on the other hand, handle the partitioning of a hard disk in different ways. Some distributions try to be as user-friendly as Windows, resulting in a partition set up that, too, is less than optimal, although they also provide an easy way to manually modify the suggested default. Other distros require involvement of the person doing the installation so that they can physically make specific choices.

Once a hard disk has been partitioned, files are placed on it in a tree-like structure of folders and subfolders. While both Windows and Linux use a similar tree-like file structure, how that file structure corresponds to the partition(s) on a hard disk differ between the two operating systems.

This paper explains what you need to know in order to make good choices when setting up your partitions. It is aimed at a Windows user who is learning Linux and setting up Linux one of the first few times.

Partitions and sectors on a hard disk

A hard disk is a circular piece of material, much like a vinyl record (for those of you who remember vinyl) or compact disk. Data is laid down on a hard disk in segments, one file after another. You can think of files residing on a hard disk like the three files shown in Figure 1a. (In reality, things are a bit more complicated, but this simplistic explanation serves our purpose.)

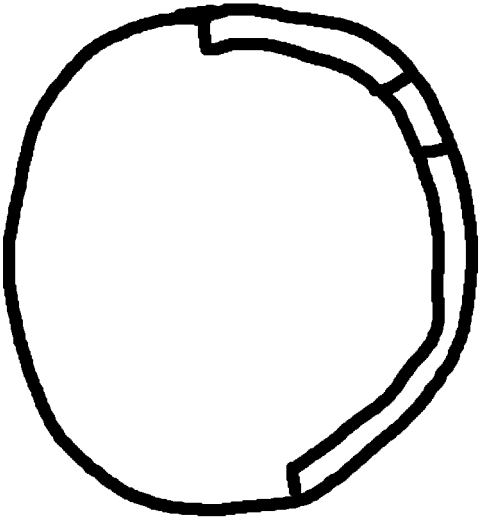

Figure 1. Three files laid out on a disk.

When you partition a disk, you create separate areas on it, much like shown in Figure 2, where there are three partitions, A, B, and C.

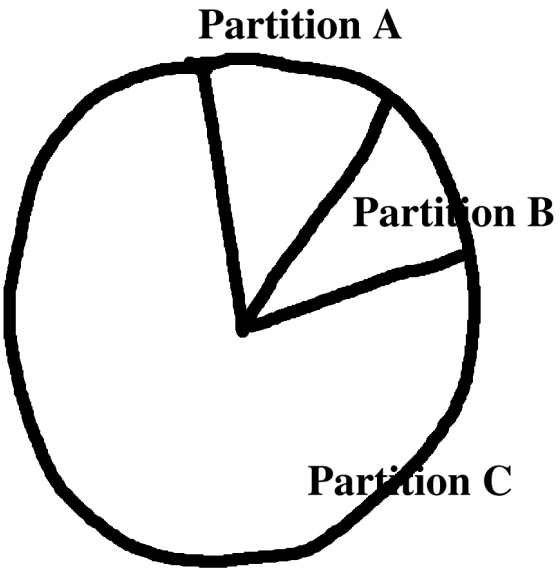

Figure 2. A hard disk divided into three partitions.

Files are then wholly contained within one of the partitions, as shown in Figure 3. Partition A contains three files (the second file wraps to the second row), Partition B contains one file, and Partition C contains four files, the fourth of which wraps around to the second row.

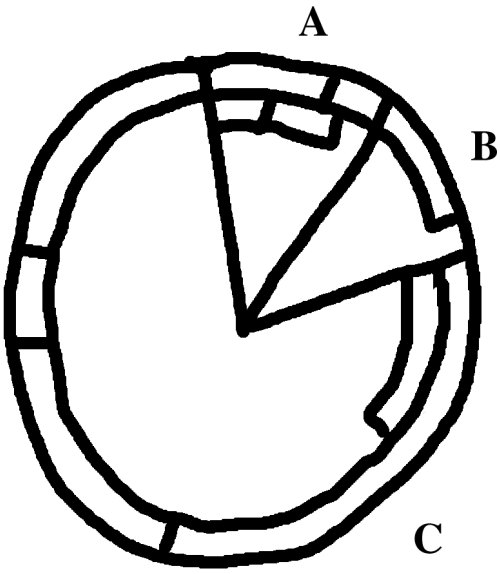

Figure 3. Files must reside entirely in a single partition.

After enough files are created and deleted, gaps of unused space start showing up on the disk. In Figure 4, the outside row in Partition B has two files, the second row has three, and the start of the fourth that takes up the rest of the partition. When a file is created, if it's too big, it may not be able to be placed in a single contiguous area as it could have been when the disk was nearly empty. Instead, the operating system will divide the file into pieces and fits those pieces in where it can.

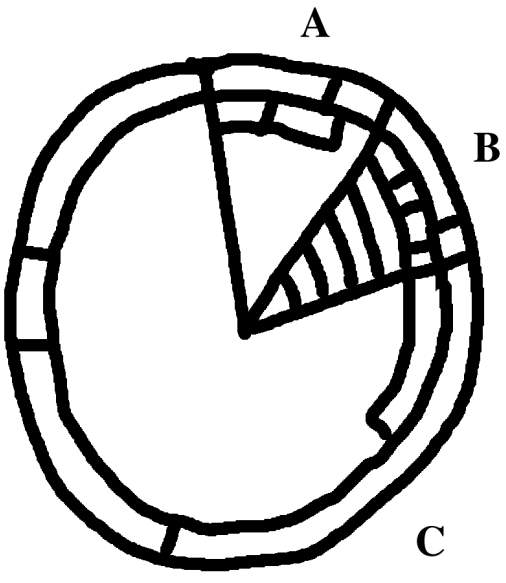

Figure 4. Partition B is completely full of files.

In Figure 5, the second, third and fifth files were deleted, leaving gaps. If you want to save another file that's larger than any of the open spaces, the new file will have to be saved in pieces.

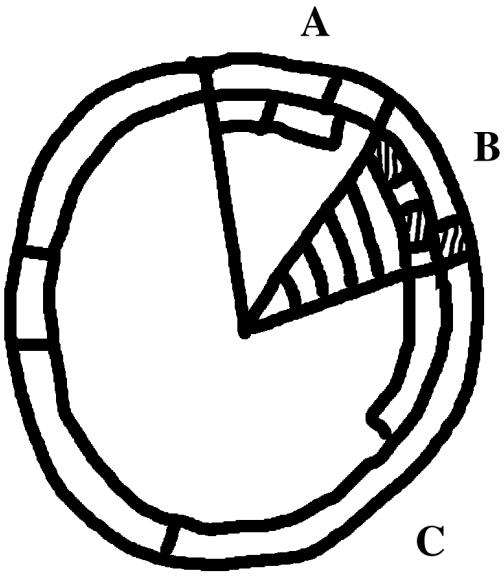

Figure 5. Fragmented disk space in Partition B.

This is called fragmenting, and it's inefficient. In most cases, the flexibility of being able to put files anywhere there is free space outweighs the slight performance hit, but in a few cases, it's unacceptable.

How Windows disks, partitions and file structures work

Windows computers typically have one hard disk that is divided into one one partition. Windows, it's predecessors having been essentially a layer on top of DOS, has inherited a number of attributes that constrain it in some ways. DOS was designed to control personal computers that only had one (or, sometimes, two) floppy disk drives - hard disk drives were unheard of. Storage devices, then, were referred to as "Drive A" and "Drive B". When hard disk drives (and removable disk drives, such as Bernoulli boxes) came upon the scene, the nomenclature was extended so that the first hard disk drive was referred to as Drive C, a second hard disk (or a Bernoulli box) was Drive D, and so on.

In those days, dividing a hard disk into more than partition was unnecessary, because there was no need to create a hard disk with more than one partition. (Why? The capacities were small enough that a single partition could cover an entire hard disk, and networks - with their attendant security risks that would require separation of programs and data on separate partitions - were still in the future.)

Eventually, the limits that Windows artificially placed on partition sizes became a problem. Hard disk capacities grew to the point that more than one partition was required to

take advantage of all of the disk space available on a hard disk. Although these partitions were both on the same physical hard disk drive, Windows referred to them by different drive letters - if a disk drive was divided into three partitions, those partitions would be referred to as Drive C, Drive D, and Drive E. Sometimes, depending on the constraints of the disk controllers inside the machine, another storage device, like a CD-ROM, would be inserted into the drive letter sequence, so that Drive C referred to the first partition on the hard disk, Drive D referred to the CD-ROM, and then Drives E and F referred to the remaining two partitions on the hard disk.

The dual usage of the term 'drive' to refer to a physical disk drive as well as a partition on a physical disk drive generates a fair amount of confusion; as a result, the nomenclature of "physical disk drive" to refer to the physical device, and "logical disk drive" to refer to a partition on a physical disk drive has come into play.

Each logical drive in Windows has its own directory structure. The topmost folder, referred to as the root folder, is denoted by a backslash in Windows (\). The drive letter and backslash (e.g. C:\) uniquely identify the root folder for each logical drive. Underneath each of these root folders are a series of subdirectories and files.

Windows is always placed on Drive C of a machine, regardless of whether Drive C refers to an entire physical disk drive or simply the logical disk drive in a partition on a physical drive. The operating system consists of three types of files - the boot software that controls how the machine is started, the operating system files that control how the machine works, and the configuration files, that provide user-determinable data to customize the machine.

In addition to the operating system, the C drive is also the location where application software, application configuration files, user configuration files and user data are all stored by default. A typical Windows installation produces a file structure that includes a number of standard directories. For instance, you'll usually find C:\WINNT (or C:\WINDOWS), C:\Program Files, C:\Documents and Settings, C:\My Documents, and C:\TEMP.

If there is more than one partition on the hard disk(s) of a Windows machine, the rest of the partitions are formatted to be ready to accept data and/or programs, but they will not have the operating system on them. And programs and data will by default still be directed to the C drive.

As mentioned, each partition has its own tree of folders and subfolders, and they are not related to each other in any way. Each partition can have an identical tree structure, distinguishable only by which drive they're on. Thus, in order to refer to a file, you need to include the name of the logical drive as well as the path and the file name.

This doesn't provide a very coherent view of the entire machine, so at some point, Windows introduced the concept of "My Computer" that acted as a parent to all of the storage devices on a machine. That ended up not being enough, so My Computer became just one of many entities contained in "My Desktop", others being network connections and non-storage devices, such as Recycle Bin and third party add-ins like the PowerDesk FTP site, as shown in Figure 6.

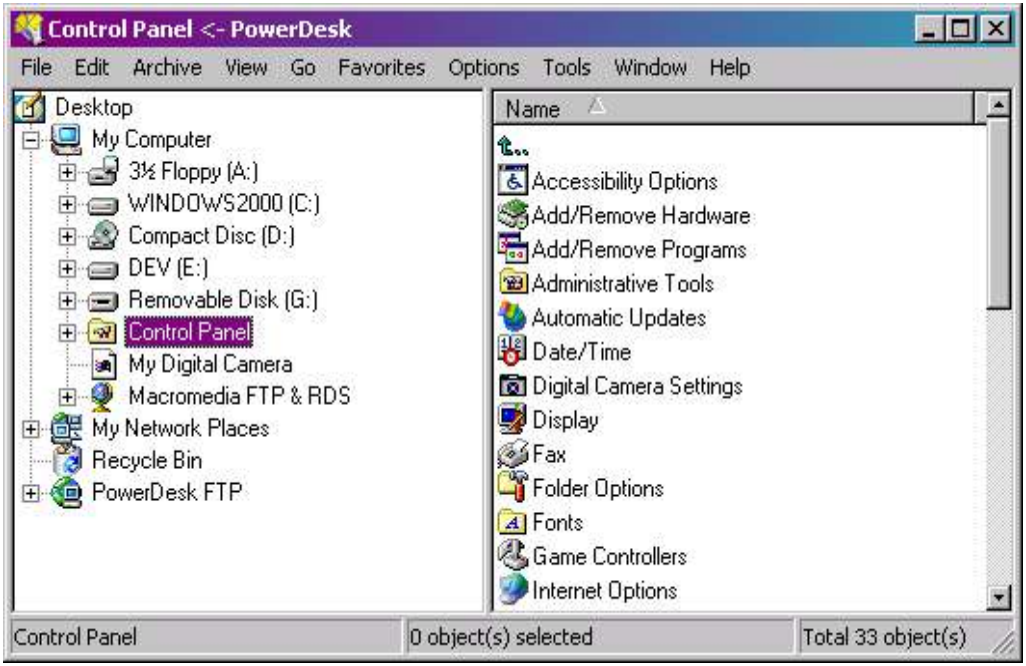

Figure 6. The device hierarchy of a Windows computer.

These artificial wrappers for the various devices are one weakness of the Windows architecture; as new devices and requirements evolve, yet more wrappers or artificial layers will likely need to be introduced into this already unwieldy structure.

The other inherent weakness of the Windows architecture is that, as we saw, all of the programs and user data on a machine are by default contained in one partition.

There is no native separation of of the various types of the software on the machine - it' s all dumped in one place. Files used to boot the machine, core operating system files, system configuration files, system utilities, application software, user configuration files, user data, and everything else is all contained in one contiguous set of directories that is, for the most part, accessible to anyone who sits down to the machine.

This can be changed to some extent by a knowledgeable user, by placing some programs and some data on a different partition, but many of the core files, including the entire operating system, some application software and configuration files, and most (if not all) user configuration files all still need to be placed on Drive C.

This architecture constitutes a huge security risk, as operating system files and user configuration files and data are not segregated from the rest of the file structure on the computer, and can be accessed by any user. Only recent versions of Windows have made any attempt to segregate critical operating system files from user data, but those mechanisms aren' t turned on by default and their limitations can be easily circumvented.

How Linux partitions, etc work

Linux, on the other hand, has always treated all devices as components in a coherent hierarchy. This has two ramifications. The first is that all devices - and, accordingly, all files and folders - fit into a single unified tree structure - there aren't any artificial distinctions like drive letters for various physical components. While disks and partitions are subject to the same basic requirements as with Windows systems - at least one partition per physical hard disk, and partitions can't be spread across disks - this unified tree means that the directory structure can (and nearly always does) span partitions. In other words, Windows uses a separate directory structure for each partition while Linux uses a single directory structure to represent all partitions. When drilling down through the directory tree to get to a specific file on Linux, it's transparent which disk drive and partition you're working with.

The second is that Linux by default separates files according to function, creating separate partitions on the disk for each type of file - and these partitions aren't allowed to communicate with each other except under very specific and controlled circumstances. Furthermore, those partitions are configured differently - for example, the partitions of the hard disk that hold boot and core OS files are usually configured to be read-only, while, in Windows, once you get access to a directory that contains OS files, you can do anything you want to them - and so can malevolent users.

Let's explore how this works, by discussing what the Linux directory structure looks like, and then showing how that file tree is laid out onto disks and partitions.

As we saw, Windows has a number of standard directories, and the same is true for Linux. A typical Linux file structure includes (but is not limited to) /bin, /boot, /etc, /home, /opt, /sbin, /tmp, /usr, and /var. See Figure 7 for a tree view of the top level directories in a standard Red Hat 8.0 installation.

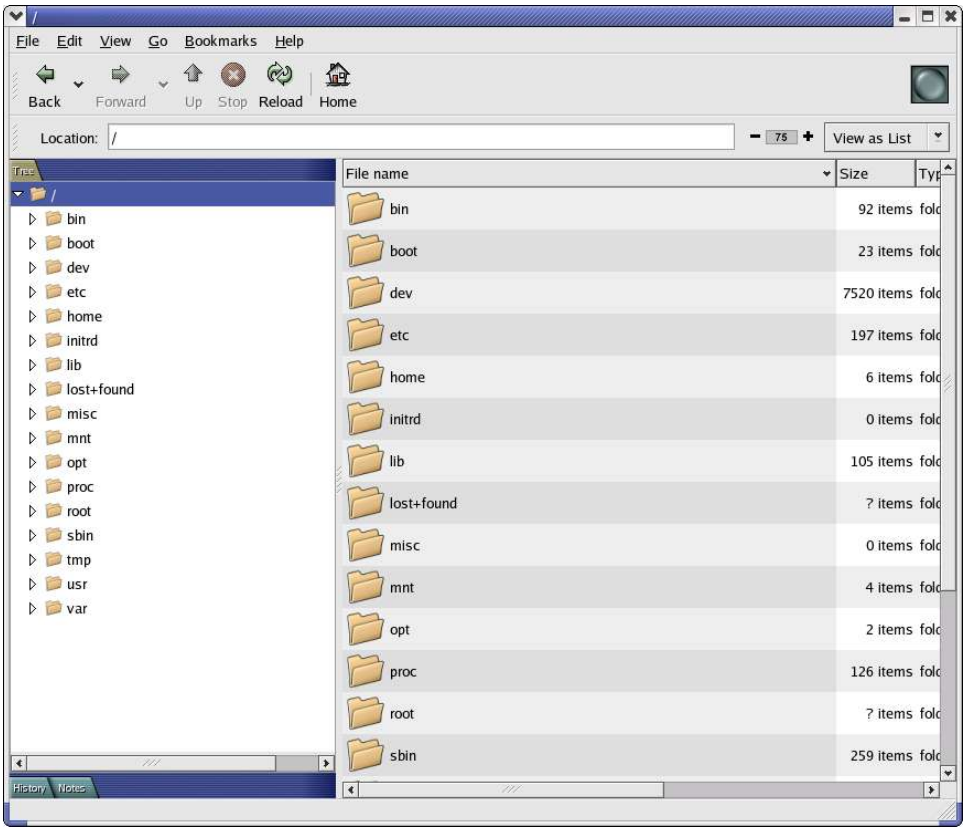

Figure 7. A typical directory structure in Linux.

This file structure actually varies slightly across distributions, and what files are placed in what directories also varies to some extent, but for the most part, these are common across the various flavors of Linux.

In any case, all of these directories are found under the top level directory, /, or what is known as the 'root' directory (not to be confused with the /root directory, which has a specific purpose.) You'll see immediately that there isn't a drive letter associated with the / directory; nor are there drive letters or hooks for other ancillary devices like networked drives, cameras, removable media, and so on. All of those items are still accessible through the directory structure, but the directory structure is designed to provide spots for them out of the box. For example, CD-ROMs and network drives are often found under the /mnt directory.

Since there are no drive letters in a Linux directory structure, where do the partitions go? In Windows it's easy to see the relationship between drives and partitions - there's a one to one mapping. With Linux, just by looking at the directory structure, you can't tell how many drives or partitions are involved. Whether there is more than one drive (and, thus, partition) on a physical hard disk is transparent to the user.

Let's suppose we have one hard disk with one partition, and the entire directory structure and all of the files are on it. This would conceivably work, although in the real world it isn't done. With Linux, there's always a second partition on a machine - and this partition contains the swap file. The swap file, just like in Windows, is a dedicated vehicle used as an extension of RAM - when a particular operation needs more RAM than is physically available on the machine, the operating system will place a portion of what's in RAM (ideally, data that is not needed for the current operation) into the swap file, thus freeing up RAM for its operation. When finished with that operation, the data in the swap file is then (usually) moved back into RAM. In this way, the machine realizes the benefit of being able to access additional RAM without actually physically having it. The swapping of data to and from the swap file does take some time, of course, which means that this technique isn't as fast as simply having more RAM, but it optimizes the use of what actually is in the machine.

This operation works best if there is contiguous space on the hard disk that is dedicated for the swap file, so that the operating system doesn't have to waste time looking for some free space, and then jumping around from one free spot to another to temporarily store all of the data. Thus, having a single partition dedicated to for the swap file is virtually always done in Linux. (In Windows, space is set aside for a separate swap file, but that file is by default created and stored on the same partition as the operating system.) The rule of thumb is to have a swap partition that is twice as big as the amount of RAM in the machine. So a machine with 256 MB of RAM would have a 512 MB swap partition.

So our Linux machine's hard disk now looks like Figure 8 with two partitions - one for swap and another for 'everything else' - /.

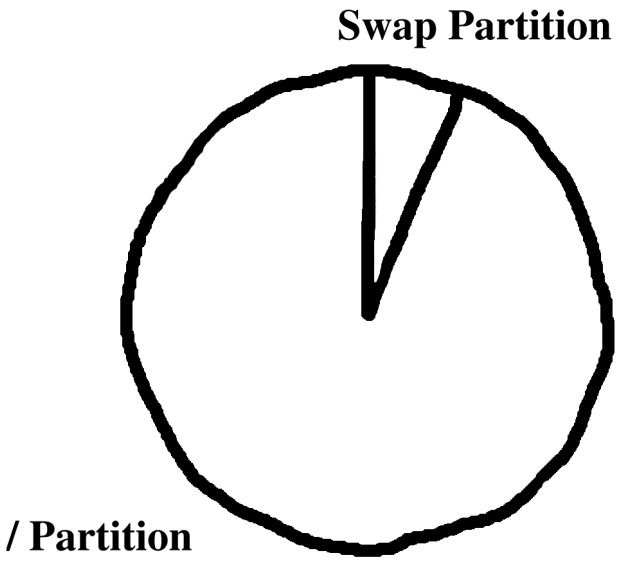

Figure 8. A Linux hard disk with two partitions.

If you were to look back at the directory tree in Figure 6, you'll see that the swap partition doesn't show up in the normal directory structure. So while there's a swap partition, it doesn't visibly map to anything in the directory structure.

The next thing that Linux machines virtually always have is a separate partition that contains all of the files required to boot the machine. These files are all contained in a directory called /boot (unlike Windows, which has various files involved in the boot process scattered in several different directories all over the hard disk.) These files are kept in a separate partition so that they can be locked down and kept secure from the rest of the world - after all, if malicious or incompetent outsiders have access to the boot files, the entire machine's security is suspect. See Figure 9.

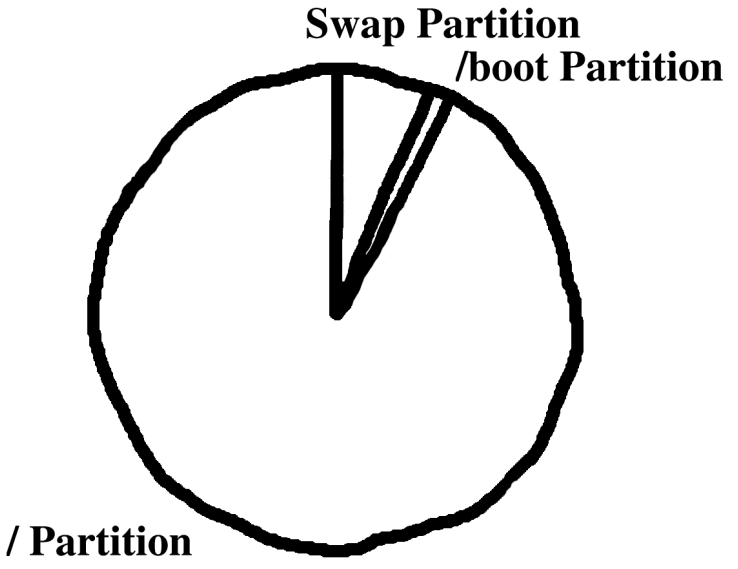

Figure 9. A Linux hard disk with three partitions.

The recommended partition size for /boot varies between 50 and 100 MB, depending on who is doing the recommending, although I've seen partitions being as small as 25 MB on desktop machines without ill effects. Part of the size requirements depend on what type of Linux distribution you're installing. The 50-100 MB recommendation is for typical desktop or server configurations; an embedded configuration would typically need considerably less.

As you can see, the all of the files in the directory tree are now contained in three different partitions (if you include the swap file.) The big conceptual hurdle for Windows users to leap here is the idea that a subdirectory and the files in it, such as /boot, can be contained in a separate partition.

When you think about it for a minute, there's no real reason why this can't be done - it's just that it is never done in the Windows world, and thus Windows users are generally not used to it. But technically, all of the data on a hard disk is laid out in sectors that wrap around the disk one after another, and there's no reason that some of that data - data in a specific directory - can't be placed in a specific partition.

Various Linux partition configurations

So far, we've shown that a minimal installation would have three partitions - one for the swap file, one for /boot, and one for everything else - /. But that's just the beginning.

Put /home on its own partition

At a bare minimum, then, these three make up what the partition scheme for a typical Linux machine should be, but for most installations, there really ought to be at least one more - the partition for user data, called /home. Each user on a Linux machine has their own directory under the /home directory, so that Al's directory would be /home/al and Barb's directory would be /home/barb. (The root user's home directory isn't part of /home - it's got a separate directory called /root, because you don't want root's files intermingled with all of the other users.) User configuration files, user-specific applications, and user data are all examples of files that go into these user directories.

Even if you're the only user on the machine, it's a good idea to keep the /home directory on its own partition - so your directory will be /home/yourname.

How much room should be allocated for the /home partition? The answer is.... it depends. The remaining disk space has to be divided (roughly) into space for applications and other software and for /home's file.

It depends on how much total disk space there is on the physical disk drive. Typical desktop and server installations require a minimum of between 1 and 4 GB of disk space,, depending on how much extra stuff (applications, tools, utilities) is installed, and and may grow to 6 to 10 GB, depending on how many applications users may want installed. Given those requirements, and the total amount of disk space available on the machine, you may choose to allocate from "all but 4 GB" or "all but 10 GB" to /home.

After creating a /home partition, the machine now looks like Figure 10.

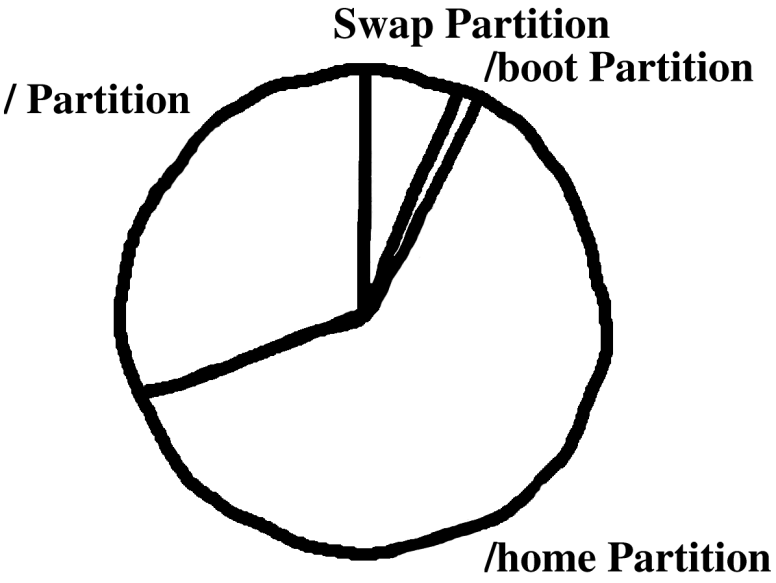

Figure 10. A Linux hard disk with a /home partition.

A few paragraphs ago, I said that there are several reasons why /home should be on its own partition - even for people who are the only user on their desktop machine. Let's look at these reasons in more detail.

The first scenario that comes to mind an upgrade of the operating system, say, from Fedora Core 1 to Fedora Core 2. You've very likely got data in your home directory that you don't want to lose just because you're upgrading the software. If you have /home on a separate partition, you can tell the installation routine to leave the existing /home directory alone. Then when you boot your machine with the new OS, it'll automatically find the existing home partition.

Another scenario that would require a separate /home partition is a file server. Each user with data on the file server will have their own directory under /home. It is possible that an unthinking, inconsiderate, or malicious user will attempt to save oodles of data in their directory. In an extreme situation, if /home isn't on its own partition, it is possible for one of these users to use up all available disk space ("I what? I only saved three files in my directory. How much space do the DVDs for those three Matrix movies take, anyway?) the entire disk By partitioning /home, you can make sure that no matter what a user does, the worst that can happen is that /home gets filled up, but the server keeps on running. Additionally, you can create directory quotas for each user so that they don't hogart too much space themselves.

Why more than four partitions?

After having put /home on its own partition, we now have four partitions - three more than a typical Windows machine. Why would a Linux machine ever need more than that? It depends on what type of machine you're setting up. If you're setting up a desktop machine that will be used by a single person, these four partitions are probably going to be plenty. (Although if you have more than one physical hard disk, you may want to read further to see some ideas on how to partition multiple hard disks and then assign directories to those partitions.)

Before discussing what additional partitions might be used for, it's important to mention that you can't just go putting any old directory on its own partition. For example, in order for the machine to boot properly, you need to have /bin, /etc, and /sbin all on the same partition. That said, there are a few directories that would do well to be on their own partition. For example, the /var partition is used for 'variable data files', such as spool directories (print spoolers), administration and logging data, and other transient data files. The /tmp directory is used just as it sounds - for temporary files.

After time, the amount of space taken up by files in these directories can grow, particularly if there are a lot of users on the machine. Spool files may end up orphaned after a trashed print run, log files may get to be large, and since you can't always count on a program to clean up after itself, after a while there may be a fair amount of 'stuff' accumulated. You don't want overruns in these directories to use up all of the available disk space, so keeping them on their own partition makes it impossible for a runaway process or an overgrowth of files to fill up a disk.

There are security reasons for doing so as well. By definition, the /var and /tmp directories are wide open to be written to, and thus malicious users have access to those areas. Keeping them on their own partition provides another layer of protection between the innards of the operating system and the rest of the computer.

In general, anything that has data that you want to keep safe and/or separate. If an event corrupts or otherwise messes with one partition, generally only that partition has to be restored from backup. For example, if you are running a database application with large databases, you would want to keep the databases on a separate partition. Then, if the OS gets messed up, it has to be restored or reinstalled, but the database is safe.

Configurations for a single hard disk

Now that we've been through the theory, let's look at a couple of possible configurations for a machine with a single hard disk.

Sample configuration 1

For a desktop machine with an 8 GB hard drive and 256 MB of RAM:

```
/swap        512 MB
/boot        100 MB
/home       3500 MB
/           4000 MB
```

Sample configuration 2

For a desktop machine with a 50 GB hard disk and 512 MB of RAM:

```
/swap       1 GB
/boot     100 MB
/home      40 GB
/           9 GB
```

Configurations for a multiple hard disk system

A fair number of machines have more than one hard disk these days, either because the additional device was needed to meet space requirements, or simply due to an upgrade.

Sample configuration 1

For a desktop machine with one 8 GB hard disk, one 40 GB hard disk, and 512 MB of RAM:

```
/swap     1 GB on drive 1
/boot   100 MB on drive 1
/home    40 GB on drive 2
/         7 GB on drive 1
```

Sample configuration 2

For a desktop machine with two 50 GB hard disks and 512 MB of RAM:

```
/swap          1 GB on drive 1
/boot        100 MB on drive 1
/home         40 GB on drive 1
/              9 GB on drive 1
/home/music   50 GB on drive 2
```

Sample configuration 3

For a server machine with two 120 GB hard disks and 1 GB of RAM:

```
/swap           2 GB on drive 1
/boot         100 MB on drive 1
/              20 GB on drive 1
/var            5 GB on drive 1
/tmp            5 GB on drive 1
/home/shared   88 GB on drive 1
/home        1200 GB on drive 2
```

Setting partition parameters

Earlier I mentioned that one of the advantages of creating multiple partitions is that each partition can be set up with different permissions parameters - one being read-only, another being read-write, and so on. Indeed, many discussions mention this fact, and then don' get around to saying HOW to do it.

First, it's useful to know how to determine how a partition is set up to begin with. The file /etc/mtab contains information about what filesystems are currently mounted, where they are mounted, and with what options. For example,

```
/dev/hda2   /          ext3      rw
/none       /proc      proc      rw   0   0
none        /dev/pts   devpts    rw,gid=5,mode=620 0 0
dev/hda1    /boot      ext3      rw   0   0
```

This looks much like the contents of fstab although the parameters at the end of each line are somewhat different. You can see how each of the four filesystems are mounted as read/write.

Changing read/write permissions on partitions: *\\\ TK

Directory tree standards

There are currently two directory tree standards in widespread use. (Standards are good things. That's why we have so many of them.)

The first, FHS, has been used by Red Hat for years. You can find more info at www.pathname.com/fhs/. The second standard is handled by LSB, at http://www.linuxbase.org/

Thanks to Jesse Kipp and Tom Francis for feedback on this whitepaper.

For updates to this whitepaper as well as other HOWTO whitepapers, please visit www.hentzenwerke.com.