developerWorks®

Learn Linux, 101: Create partitions and filesystems

Divide and conquer your disk space

Ian Shields

December 04, 2012
(First published July 12, 2010)

Learn how to create partitions on a disk drive and how to format them for use on a Linux® system as swap or data space. You can use the material in this article to study for the LPI 101 exam for Linux system administrator certification, or just to learn about partitions and Linux filesystems for your own use.

View more content in this series

Overview

In this article, learn about disk partitions and Linux filesystems. Learn to:

- Create a partition
- Use `mkfs` commands to set up ext2, ext3, ext4, xfs, Reiser v3,and vfat filesystems
- Create and manage swap space

This article helps you prepare for Objective 104.1 in Topic 104 of the Linux Professional Institute's Junior Level Administration (LPIC-1) exam 101. The objective has a weight of 2.

Note: This article includes material for the LPI Exam 101: Objective Changes as of July 2, 2012. We have added basic information on ext4 filesystems. We have also added some basic information on the `gdisk` command and the GUID Partition Table (GPT). The new code listings and figures were all done on a 64-bit Fedora 17 system.

About this series

This series of articles helps you learn Linux system administration tasks. You can also use the material in these articles to prepare for Linux Professional Institute Certification level 1 (LPIC-1) exams.

See our developerWorks roadmap for LPIC-1 for a description of and link to each article in this series. The roadmap is in progress and reflects the latest (April 2009 with minor updates in July 2012) objectives for the LPIC-1 exams: as we complete articles, we add them to the roadmap. In the meantime, though, you can find earlier versions of similar material, supporting previous LPIC-1 objectives prior to April 2009, in our LPI certification exam prep tutorials.

Prerequisites

To get the most from the articles in this series, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this article. Sometimes different versions of a program will format output differently, so your results may not always look exactly like the listings and figures shown here.

You should also be familiar with the material in our article, "Learn Linux 101: Hard disk layout."

Block devices and partitions

Our article, "Learn Linux 101: Learn Linux, 101: Hard disk layout," introduced you to hard drive layouts, partitions, and some basic use of the `fdisk` and `gdisk` commands to view partition information. You learned about the Master Boot Record (MBR), partition tables, partitions, including *primary*, *extended*, and *logical* partitions. You were also introduced to GUID Partition Tables (GPT), a new format used to address the size limitations inherent in the MBR layout. Finally, you learned that a Linux filesystem contains *files* that are arranged on a disk or other *block storage device* in *directories*. As with many other systems, directories on a Linux system may contain other directories called *subdirectories*. That article also discussed the considerations that guide you in making choices about partitioning.

Note: This article focuses on the LPI requirements related to the `fdisk` command and partitioning using MB layouts. It includes some `gdisk` command information in Creating an ext4 filesystem. Refer to the earlier article and its resources for more information on GPT.

We'll start this article with a review of block devices and partitions, and then show you more about the `fdisk` command, which is used to create, modify, or delete partitions on block devices. You will also learn about the various forms of the `mkfs` command (mkfs stands for *make filesystem*); `mkfs` commands are used to format partitions as a particular filesystem type.

Note: In addition to the tools and filesystems required for the LPI exams, you may encounter or need other tools and filesystems. Find a brief summary of some other available tools in Other tools and filesystems.

Block devices

A *block device* is an abstraction layer for any storage device that can be formatted in fixed-size *blocks*; individual blocks may be accessed independently of access to other blocks. Such access is often called *random access*.

The abstraction layer of randomly accessible fixed-size blocks allows programs to use these block devices without worrying about whether the underlying device is a hard drive, floppy, CD, solid-state drive, network drive, or some type of virtual device such as an in-memory file system.

Examples of block devices include the first IDE hard drive on your system (/dev/sda or /dev/hda) or the second SCSI, IDE, or USB drive (/dev/sdb). Use the `ls -l` command to display /dev entries.

The first character on each output line is **b** for a **block** device, such as floppy, CD drive, IDE hard drive, or SCSI hard drive; and **c** for a **character** device, such as a or terminal (tty) or the null device. See the examples in .

Linux block and character devices

```
[ian@echidna ~]$ ls -l /dev/loop1 /dev/null /dev/sd[ab] /dev/sr0 /dev/tty0
brw-rw----. 1 root disk   7,  1 2010-06-14 07:25 /dev/loop1
crw-rw-rw-. 1 root root   1,  3 2010-06-14 07:25 /dev/null
brw-rw----. 1 root disk   8,  0 2010-06-14 07:25 /dev/sda
brw-rw----. 1 root disk   8, 16 2010-06-14 07:25 /dev/sdb
brw-rw----+ 1 root cdrom 11,  0 2010-06-14 07:25 /dev/sr0
crw--w----. 1 root root   4,  0 2010-06-14 07:25 /dev/tty0
```

Partitions

For some block devices, such as floppy disks and CD or DVD discs, it is common to use the whole media as a single filesystem. However, with large hard drives, and even with USB memory keys, it is more common to divide, or partition, the available space into several different *partitions*.

Partitions can be different sizes, and different partitions may have different filesystems on them, so a single disk can be used for many purposes, including sharing it between multiple operating systems. For example, I use test systems with several different Linux distributions and sometimes a Windows® system, all sharing one or two hard drives.

You will recall from the article, "Learn Linux 101: Learn Linux, 101: Hard disk layout," that hard drives have a *geometry*, defined in terms of cylinders, heads, and sectors. Even though modern drives use *logical block addressing* (*LBA*), which renders geometry largely irrelevant, the fundamental allocation unit for partitioning purposes is usually still the cylinder.

Displaying partition information

Partition information is stored in a *partition table* on the disk. The table lists information about the start and end of each partition, information about its *type*, and whether it is marked bootable or not. To create and delete partitions, you edit the partition table using a program specially designed for the job. For the LPI exam, you need to know about the `fdisk` program, so that is what is covered here, although several other tools could be used. We will mention some at the end of this article.

The `fdisk` command with the `-l` option is used to list partitions. Add a device name, such as /dev/sda, if you want to look at the partitions on a particular drive. Note that partitioning tools require root access. shows the partitions on the primary hard drives of two of my systems.

Listing partitions with fdisk

```
[root@attic4 ~]# fdisk -l /dev/sda

Disk /dev/sda: 640.1 GB, 640135028736 bytes
255 heads, 63 sectors/track, 77825 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00064a1a

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1         127     1020096   83  Linux
```

```
/dev/sda2             128        1402    10241437+  82  Linux swap / Solaris
/dev/sda3   *       46340       56538    81920000   83  Linux
/dev/sda4            1403       46339   360956422    5  Extended
/dev/sda5            1403       10420    72437053+  83  Linux
/dev/sda6           10421       19344    71681998+  83  Linux
/dev/sda7           19345       28350    72340663+  83  Linux
/dev/sda8           28351       37354    72324598+  83  Linux
/dev/sda9           37355       46339    72171981   83  Linux

Partition table entries are not in disk order

[root@echidna ~]# fdisk -l /dev/sda

Disk /dev/sda: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000de20f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        9111    73184076    7  HPFS/NTFS
/dev/sda2            9634        9730      779152+  83  Linux
/dev/sda3            9731      116679   859067842+   5  Extended
/dev/sda5            9731       20917    89859546   83  Linux
/dev/sda6           20918       39644   150424596   83  Linux
/dev/sda7           39645       53905   114551451   83  Linux
```

Notes:

1. The header information shows the disk size and geometry. Most large disks using LBA have 255 heads per cylinder and 63 sectors per track, making a total of 16065 sectors, or 8225280 bytes per cylinder.
2. In the second example, the first primary partition (/dev/sda1) is marked *bootable* (or *active*). This enables the standard DOS PC master boot record to boot the partition. This flag has no significance for the LILO or GRUB boot loaders. The first example uses GRUB as the boot loader, and the fact that /dev/sda3 is marked bootable is probably an accident of the history of my use of this drive.
3. The *Start* and *End* columns show the starting and ending cylinder for each partition. These must not overlap and should generally be contiguous, with no intervening space.
4. The *Blocks* column shows the number of 1K (1024 byte) blocks in the partition. For most disks in use at the time of writing, the sector size is 512 bytes, so the maximum number of blocks in a partition is therefore half of the product of the number of cylinders (End + 1 - Start) and the number of sectors per cylinder. A trailing + sign indicates that not all sectors in the partition are used.
5. The *Id* field indicates the intended use of the partition. Type 82 is a Linux swap partition, and type 83 is a Linux data partition. There are approximately 100 different partition types defined. The second disk is shared between several operating systems, including Windows/XP, hence the presence of Windows NTFS (and possibly FAT32) partitions.

Partitioning with fdisk

You have just seen how to display partition information using the `fdisk` command. This command also provides a menu-driven environment for editing the partition table to create or remove partitions.

Warnings

Before you start modifying partitions, there are some important things to remember. You risk **losing your existing data** if you do not follow these guidelines.

1. **Back up important data before you start**, as with any operation that may cause data loss.
2. **Do not change partitions that are in use**. Plan your actions and execute them carefully. Booting a live distribution from CD, DVD, or USB is one good way to ensure that no hard drive partitions are in use.
3. **Know your tool**. The `fdisk` command does not commit any changes to your disk until you tell it to. Other tools, including `parted`, may commit changes as you go.
4. **Stop if you do make a mistake**. Partitioning tools write the partition table. Unless the tool you are using also includes the ability to move, resize, format, or otherwise write to the data area of your disk, your data will not be touched. If you do make a mistake, stop as quickly as possible and seek help. You may still be able to restore your previous partition table definitions and thus recover your partitions and data.

Start fdisk

To start `fdisk` in interactive mode, simply give the name of a disk, such as /dev/hda or /dev/sdb, as a parameter. The following example boots a Knoppix live DVD. You will need root authority, and you will see output similar to .

Starting interactive fdisk

```
knoppix@Microknoppix:~$ su -
root@Microknoppix:~# fdisk /dev/sda

The number of cylinders for this disk is set to 121601.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):
```

Most modern disks have more than 1024 cylinders, so you will usually see the warning shown in . Type `m` to display a list of available one-letter commands as shown in .

Help in fdisk

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

Command (m for help):
```

Use the `p` command to display the existing partition on this particular disk; shows the output.

Displaying the existing partition table

```
Command (m for help): p

Disk /dev/sda: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000de20f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        9111    73184076    7  HPFS/NTFS
/dev/sda2            9634        9730      779152+  83  Linux
/dev/sda3            9731      116679   859067842+   5  Extended
/dev/sda5            9731       20917    89859546   83  Linux
/dev/sda6           20918       39644   150424596   83  Linux
/dev/sda7           39645       53905   114551451   83  Linux

Command (m for help):
```

This particular disk is a 1TB disk with a Windows/XP partition of a little under 80GB. It is a primary partition, and it is marked bootable, as is typical for a Windows system.

Adding partitions

Let's now use part of the free space to add some partitions.

1. We will create a swap partition as /dev/sda4. This will be a primary partition, filling the 521 cylinder gap between the end of /dev/sda1 and the start of /dev/sda2. Don't even begin to wonder what crazy things cause this gap to exist; I created the gap deliberately so I could write this article.
2. We will create a 40GB logical partition as /dev/sda8.
3. Finally, we will create a small 2000MB logical partition for sharing data between the Linux and Windows systems. This will eventually be formatted as FAT32 (or vfat). It will be /dev/sda9.

Creating our partitions

Let's start by using the `n` command to create a new partition; see .

Creating our first partition

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
p
Selected partition 4
First cylinder (9112-121601, default 9112):
Using default value 9112
Last cylinder, +cylinders or +size{K,M,G} (9112-9633, default 9633): +521

Command (m for help): p

Disk /dev/sda: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000de20f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        9111    73184076    7  HPFS/NTFS
/dev/sda2            9634        9730      779152+  83  Linux
/dev/sda3            9731      116679   859067842+   5  Extended
/dev/sda4            9112        9633     4192965   83  Linux
/dev/sda5            9731       20917    89859546   83  Linux
/dev/sda6           20918       39644   150424596   83  Linux
/dev/sda7           39645       53905   114551451   83  Linux

Partition table entries are not in disk order

Command (m for help):
```

We took the default for the first cylinder and specified the value of +521 for the number of cylinders. You can see from that our partition is approximately 4GB in size. Since it is a primary partition, it must be numbered from 1 through 4. It is a good idea to assign partition numbers sequentially; some tools complain if this is not done and `fdisk` warns us that our partition table entries are no longer in disk order.

Notice also that our new partition was assigned a type of 83, for a Linux data partition. Think of this as an indicator to the operating system of the intended use of the partition. The eventual use should match this, but at this point we don't even have the partition formatted, let alone have any data on it. We'll create out other partitions first, then look at how to change the partition type.

You may have noticed that when we entered the `n` subcommand to create a new partition, the only choices were 'l' for logical and 'p' for primary. You will only see options for the remaining possible types of partitions. You would see 'e' for extended if the drive did not already have an extended partition. Also note that our extended partition (/dev/sda3) is type 5.

Now let's define the 40GB Linux partition and the 2000MB FAT32 partition. This time we will simply specify sizes of +40G and +2000M, indicating 40GB and 2000MB, respectively. We let `fdisk` calculate the number of cylinders for us. The results are shown in .

Creating our data partitions

```
Command (m for help): n
First cylinder (53906-116679, default 53906):
Using default value 53906
Last cylinder, +cylinders or +size{K,M,G} (53906-116679, default 116679): +40G

Command (m for help): n
First cylinder (59129-116679, default 59129):
Using default value 59129
Last cylinder, +cylinders or +size{K,M,G} (59129-116679, default 116679): +2000M

Command (m for help): p

Disk /dev/sda: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000de20f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        9111    73184076    7  HPFS/NTFS
/dev/sda2            9634        9730      779152+  83  Linux
/dev/sda3            9731      116679   859067842+   5  Extended
/dev/sda4            9112        9633     4192965   83  Linux
/dev/sda5            9731       20917    89859546   83  Linux
/dev/sda6           20918       39644   150424596   83  Linux
/dev/sda7           39645       53905   114551451   83  Linux
/dev/sda8           53906       59128    41953716   83  Linux
/dev/sda9           59129       59384     2056288+  83  Linux

Partition table entries are not in disk order

Command (m for help):
```

Changing partition type

Before we leave `fdisk`, we will change the partition types for the swap and vfat partitions. We do this using the `t` subcommand to set the partition type. We set /dev/sda4 to type 82 (Linux swap) and /dev/sda9 to type 9 (FAT32). If you want to see the full list of supported types, enter `L` as you see in .

Changing partition types

```
Command (m for help): t
Partition number (1-9): 4
Hex code (type L to list codes): 82
Changed system type of partition 4 to 82 (Linux swap / Solaris)

Command (m for help): t
Partition number (1-9): 9
Hex code (type L to list codes): b
Changed system type of partition 9 to b (W95 FAT32)

Command (m for help):
```

Saving our partition table

So far, we have just been doing an in-memory edit of a partition table. We could use the `q` command to quit without saving changes. If something is not how you want it, you can use the `d` command to delete one or more partitions so you can redefine them. If you are happy with your setup, use the `v` command to verify your setup, and then the `w` command to write the new partition table and exit. See . If you run `fdisk -l` again, you will see that Linux now knows about the new partitions. Unlike in some operating systems, it is not always necessary to reboot to see the changes. A reboot may be required if, for example, /dev/hda3 became /dev/hda2 because the original /dev/hda2 was deleted. If a reboot is needed, `fdisk` should tell you to do so.

Saving the partition table

```
Command (m for help): v
999521580 unallocated 512-byte sectors

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
root@Microknoppix:~# fdisk -l /dev/sda

Disk /dev/sda: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000de20f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        9111    73184076    7  HPFS/NTFS
/dev/sda2            9634        9730      779152+  83  Linux
/dev/sda3            9731      116679   859067842+   5  Extended
/dev/sda4            9112        9633     4192965   82  Linux swap / Solaris
/dev/sda5            9731       20917    89859546   83  Linux
/dev/sda6           20918       39644   150424596   83  Linux
/dev/sda7           39645       53905   114551451   83  Linux
/dev/sda8           53906       59128    41953716   83  Linux
/dev/sda9           59129       59384     2056288+   b  W95 FAT32

Partition table entries are not in disk order
```

More on fdisk

You may notice that we did not change the bootable flag on any partition. As our disk stands now, it still has the Windows Master Boot Record (MBR) and will therefore boot the first primary partition that is marked bootable (the NTFS partition in our example).

Neither LILO nor GRUB uses the bootable flag. If either of these is installed in the MBR, then it can boot the Windows/XP partition. You could also install LILO or GRUB into your /boot partition (/dev/hda2) and mark that partition bootable and remove the bootable flag from /dev/hda1. Leaving the original MBR can be useful if the machine is later returned to being a Windows-only machine.

You can also use `fdisk` to fix the partition order in the partition table if you need to. This will usually change the partition numbers, so you may have other work to do to restore your system to a working system. To make this change, use the `f` subcommand to switch to expert mode and then the `f` subcommand to fix the partition order, as shown in . If you just want to see what the new partition order would be without changing it, you can use the `q` subcommand to quit as we have done in this example, rather than writing the updated partition table to disk.

Fixing the partition table order.

```
Command (m for help): p

Disk /dev/sda: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000de20f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        9111    73184076    7  HPFS/NTFS
/dev/sda2            9634        9730      779152+  83  Linux
/dev/sda3            9731      116679   859067842+   5  Extended
/dev/sda4            9112        9633     4192965   82  Linux swap / Solaris
/dev/sda5            9731       20917    89859546   83  Linux
/dev/sda6           20918       39644   150424596   83  Linux
/dev/sda7           39645       53905   114551451   83  Linux
/dev/sda8           53906       59128    41953716   83  Linux
/dev/sda9           59129       59384     2056288+   b  W95 FAT32

Partition table entries are not in disk order

Command (m for help): x

Expert command (m for help): f
Done.

Expert command (m for help): r

Command (m for help): p

Disk /dev/sda: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000de20f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        9111    73184076    7  HPFS/NTFS
/dev/sda2            9112        9633     4192965   82  Linux swap / Solaris
/dev/sda3            9634        9730      779152+  83  Linux
/dev/sda4            9731      116679   859067842+   5  Extended
/dev/sda5            9731       20917    89859546   83  Linux
/dev/sda6           20918       39644   150424596   83  Linux
/dev/sda7           39645       53905   114551451   83  Linux
/dev/sda8           53906       59128    41953716   83  Linux
/dev/sda9           59129       59384     2056288+   b  W95 FAT32

Command (m for help): q
```

You have now seen one way to add partitions to a Linux workstation. Other choices you might make are covered in the article, "Learn Linux, 101: Find and place system files."

Filesystem types

Linux supports several different filesystems. Each has strengths and weaknesses and its own set of performance characteristics. One important attribute of a filesystem is *journaling*, which allows for much faster recovery after a system crash. Generally, a journaling filesystem is preferred over a non-journaling one when you have a choice. You may also want to consider whether your chosen filesystem supports *Security Enhanced Linux* (or SELinux). Following is a brief summary of the types you need to know about for the LPI exam. See Resources for additional background information.

The ext2 filesystem

The ext2 filesystem (also known as the *second extended filesystem*) was developed to address shortcomings in the Minix filesystem used in early versions of Linux. It has been used extensively on Linux for many years. There is no journaling in ext2, and it has largely been replaced by ext3 and more recently ext4.

The ext3 filesystem

The ext3 filesystem adds journaling capability to a standard ext2 filesystem and is therefore an evolutionary growth of a very stable filesystem. It offers reasonable performance under most conditions and is still being improved. Because it adds journaling on top of the proven ext2 filesystem, it is possible to convert an existing ext2 filesystem to ext3 and even convert back again if required.

The ext4 filesystem started as extensions to ext3 to address the demands of ever larger file systems by increasing storage limits and improving performance. To preserve the stability of ext3, it was decided in June 2006 to fork the extensions into a new filesystem, ext4. The ext4 filesystem, was released in December 2008 and included in the 2.6.28 kernel. Some of the changes from ext3 are:

- File systems up to 1 exabyte with files up to 16 terabytes
- The use of extents to replace block mapping as a means of improving performance
- Journal checksums to improve reliability
- Faster file system checking because unallocated blocks can be skipped during checks.
- Delayed allocation and multiblock allocators to improve performance and reduce file fragmentation.
- Timestamp changes that provide finer granularity, a new creation date (which will require eventual updates to many other libraries and utilities to become fully useful). A further timestamp change addresses the year 2038 problem which is caused by storing timestamps as signed 32-bit integers.

The ReiserFS filesystem

ReiserFS is a B-tree-based filesystem that has very good overall performance, particularly for large numbers of small files. ReiserFS also scales well and has journaling. It is no longer in active development, does not support SELinux and has largely been superseded by Reiser4.

The XFS filesystem

XFS is a filesystem with journaling. It comes with robust features and is optimized for scalability. XFS aggressively caches in-transit data in RAM, so an uninterruptible power supply is recommended if you use XFS.

The swap filesystem

Swap space must be formatted for use as swap space, but it is not generally considered a filesystem.

The vfat filesystem

This filesystem (also known as *FAT32*) is not journaled and lacks many features required for a full Linux filesystem implementation. It is useful for exchanging data between Windows and Linux systems as it can be read by both Windows and Linux. Do **not** use this filesystem for Linux, except for sharing data between Windows and Linux. If you unzip or untar a Linux archive on a vfat disk, you will lose permissions, such as execute permission, and you will lose any symbolic links that may have been stored in the archive.

The ext3 filesystem is mature and was used as the default filesystem on many distributions. The ext4 filesystem is replacing it as the default filesystem on several distributions, including Red Hat enterprise Linux 6. Fedora 17 and Ubuntu 12.10. The ReiserFS filesystem was used for many years as the default on some distributions, including SUSE, but is less used today.

Creating filesystems

Linux uses the `mkfs` command to create filesystems and `mkswap`command to make swap space. The `mkfs` command is actually a front end to several filesystem-specific commands such as `mkfs.ext3` for ext3, `mkfs.ext4` for ext4 and `mkfs.reiserfs` for ReiserFS.

What filesystem support is already installed on your system? Use the `ls /sbin/mk*` command to find out. An example is shown in .

Filesystem creation commands

```
[ian@echidna ~]$ ls /sbin/mk*
/sbin/mkdosfs      /sbin/mkfs.ext2     /sbin/mkfs.ntfs
/sbin/mke2fs       /sbin/mkfs.ext3     /sbin/mkfs.vfat
/sbin/mkfs         /sbin/mkfs.ext4     /sbin/mkfs.xfs
/sbin/mkfs.btrfs   /sbin/mkfs.ext4dev  /sbin/mkhomedir_helper
/sbin/mkfs.cramfs  /sbin/mkfs.msdos    /sbin/mkswap
```

You will notice various forms of some commands. For example, you will usually find that the files mke2fs, mkfs.ext2, and mkfs.ext3 are identical, as are mkreiserfs and mkfs.reiserfs. Filesystems that may be needed to boot the system will usually use hard links to provide the different names for the same file. Filesystems that cannot be used for the / filesystem in Linux, such as vfat or msdos, may use symbolic links instead. The article "Learn Linux, 101: Create and change hard and symbolic links" will help you learn about these different kinds of links.

There are a few common options for all `mkfs` commands. Options that are specific to the type of filesystem being created are passed to the appropriate creation command, based on the type of filesystem specified in the `-type` option. Our examples use `mkfs -type`, but you may use the other forms directly with equal effect. For example, you may use `mkfs -type ext2`, `mk2fs`, or `mkfs.ext2`.

For the manual pages for a specific filesystem, use the appropriate `mkfs` command as the name, for example, `man mkfs.ext3`. Many of the values displayed in the output examples below can be controlled by options to `mkfs`.

Now that we have created all our partitions, we will reboot the Fedora 12 system and format the filesystems using that rather than the somewhat slower live Knoppix DVD. Of course, you could continue to use the Knoppix system if you wished. Remember that you need root authority to create filesystems.

Creating an ext3 filesystem

Let's format the /dev/sda8 partition as ext3 using the `mkfs` command as shown in .

Creating an ext3 filesystem

```
[root@echidna ~]# mkfs -t ext3 /dev/sda8
mke2fs 1.41.9 (22-Aug-2009)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
2624496 inodes, 10488429 blocks
524421 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
321 block groups
32768 blocks per group, 32768 fragments per group
8176 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 20 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Note that a journal is created with ext3. If you wish to add a journal to an existing ext2 system, use the `tune2fs` command with the `-j` option.

A useful option for ext2, ext3, and ext4 filesystems is the `-L` option with a name, which assigns a label to the partition. This can be used instead of the device name when mounting filesystems; it provides some level of insulation against changes that may need to be reflected in various control files. To display or set a label for an existing ext2, ext3, or ext3 filesystem, use the `e2label` command. Labels are limited to a maximum size of 16 characters.

A more recent development is to use a *Universally Unique Identifier*, or *UUID*, rather than a label. A UUID is a 128-bit identifier usually displayed as 32 hexadecimal digits and four hyphens. Most Linux filesystems generate an UUID automatically when the filesystem is formatted. Use the `blkid` command (which does not need root authority) as shown in to see the UUID for the partition we just formatted. UUIDs are more likely to be unique than labels and are especially useful for hot-plugged devices such as USB drives.

Displaying a UUID using blkid

```
[ian@echidna ~]$ blkid /dev/sda8
/dev/sda8: UUID="87040def-920e-4525-9c81-c585ddc46384" SEC_TYPE="ext2" TYPE="ext3"
```

Creating an ext4 filesystem

Step outside the traditional MBR layout for a moment and format a USB flash drive using the GUID Partition Table (GPT) layout to learn how it works. Then create a Linux partition on the drive with the `gdisk` command. Finally, create an ext4 filesystem on the new partition with the `mkfs` command.

With `gdisk`, look at a typical USB flash drive (also known as USB thumb drives or USB keys). They are commonly shipped with a FAT32 filesystem spanning the whole drive and they usually are not partitioned. shows how `gdisk` displays information for a drive that is not already in GPT layout.

Using gdisk to display information on a non-GPT disk

```
[root@attic4 ~]# gdisk -l /dev/sdc
GPT fdisk (gdisk) version 0.8.4

Partition table scan:
  MBR: MBR only
  BSD: not present
  APM: not present
  GPT: not present


***************************************************************
Found invalid GPT and valid MBR; converting MBR to GPT format.
***************************************************************

Warning! Main partition table overlaps the first partition by 2 blocks!
Try reducing the partition table size by 8 entries.
(Use the 's' item on the experts' menu.)

Warning! Secondary partition table overlaps the last partition by
33 blocks!
You will need to delete this partition or resize it in another utility.
Disk /dev/sdc: 62530624 sectors, 29.8 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): D7900F13-839B-49B2-9C43-4367528DC381
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 62530590
Partitions will be aligned on 32-sector boundaries
Total free space is 0 sectors (0 bytes)

Number  Start (sector)    End (sector)  Size       Code  Name
   1              32        62530623   29.8 GiB    0700  Microsoft basic data
```

First note that you must run `gdisk` as root or use `sudo`, depending on how your system is set up. Secondly note that `gdisk` wants to convert the existing format to GPT and warns you about this. Because you used the `-l` option to simply list the partitions, no action is performed.

Now run `gdisk` in interactive mode. List the available commands, then the known partition types before you create a new empty GUID partition table and finally a Linux partition. You'll display the new information and then quit `gdisk`. shows the interaction.

Using gdisk to create a GUID Partition Table (GPT) and a partition

```
[root@attic4 ~]# gdisk /dev/sdc
GPT fdisk (gdisk) version 0.8.4

Partition table scan:
  MBR: MBR only
  BSD: not present
  APM: not present
  GPT: not present


***************************************************************
Found invalid GPT and valid MBR; converting MBR to GPT format.
THIS OPERATION IS POTENTIALLY DESTRUCTIVE! Exit by typing 'q' if
you don't want to convert your MBR partitions to GPT format!
***************************************************************

Warning! Main partition table overlaps the first partition by 2 blocks!
Try reducing the partition table size by 8 entries.
(Use the 's' item on the experts' menu.)

Warning! Secondary partition table overlaps the last partition by
33 blocks!
You will need to delete this partition or resize it in another utility.

Command (? for help): ?
b back up GPT data to a file
c change a partition's name
d delete a partition
i show detailed information on a partition
l list known partition types
n add a new partition
o create a new empty GUID partition table (GPT)
p print the partition table
q quit without saving changes
r recovery and transformation options (experts only)
s sort partitions
t change a partition's type code
v verify disk
w write table to disk and exit
x extra functionality (experts only)
? print this menu

Command (? for help): l
0700 Microsoft basic data  0c01 Microsoft reserved    2700 Windows RE
4200 Windows LDM data      4201 Windows LDM metadata  7501 IBM GPFS
7f00 ChromeOS kernel       7f01 ChromeOS root         7f02 ChromeOS reserved
8200 Linux swap            8300 Linux filesystem      8301 Linux reserved
8e00 Linux LVM             a500 FreeBSD disklabel     a501 FreeBSD boot
a502 FreeBSD swap          a503 FreeBSD UFS           a504 FreeBSD ZFS
a505 FreeBSD Vinum/RAID    a800 Apple UFS             a901 NetBSD swap
a902 NetBSD FFS            a903 NetBSD LFS            a904 NetBSD concatenated
a905 NetBSD encrypted      a906 NetBSD RAID           ab00 Apple boot
af00 Apple HFS/HFS+        af01 Apple RAID            af02 Apple RAID offline
af03 Apple label           af04 AppleTV recovery      af05 Apple Core Storage
be00 Solaris boot          bf00 Solaris root          bf01 Solaris /usr & Mac Z
bf02 Solaris swap          bf03 Solaris backup        bf04 Solaris /var
bf05 Solaris /home         bf06 Solaris alternate se  bf07 Solaris Reserved 1
bf08 Solaris Reserved 2    bf09 Solaris Reserved 3    bf0a Solaris Reserved 4
bf0b Solaris Reserved 5    c001 HP-UX data            c002 HP-UX service
ef00 EFI System            ef01 MBR partition scheme  ef02 BIOS boot partition
fd00 Linux RAID

Command (? for help): o
This option deletes all partitions and creates a new protective MBR.
Proceed? (Y/N): y
```

```
Command (? for help): n
Partition number (1-128, default 1):
First sector (34-62530590, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-62530590, default = 62530590) or {+-}size{KMGTP}:
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help): p
Disk /dev/sdc: 62530624 sectors, 29.8 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): B321C1C5-795A-4836-9E16-D7D45DD8F3F2
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 62530590
Partitions will be aligned on 2048-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048        62530590   29.8 GiB    8300  Linux filesystem

Command (? for help): q
```

Your final task is to create an ext4 filesystem on your new partition, /ev/sdc1. With the `-L` option of `mkfs`, label the partition and also show the GUID using the `blkid` command as you did for the ext3 partition above. illustrates this.

Creating an ext4 partition

```
[root@attic4 ~]# mkfs -t ext4 -L IAN-USB32 /dev/sdc1
mke2fs 1.42.3 (14-May-2012)
Filesystem label=IAN-USB32
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
1957888 inodes, 7816324 blocks
390816 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
239 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
 32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
 4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[root@attic4 ~]# blkid /dev/sdc1
/dev/sdc1: UUID="b3a3c2bc-ea8d-40c6-967e-927006f55416" TYPE="ext4" LABEL="IAN-USB32"
```

To see all the parameters that you can specify when creating an ext4 partition, consult the man pages for `mkfs.ext4`.

Creating an XFS filesystem

Let's now reformat the partition we just formatted as ext3 using the XFS filesystem. Our Fedora 12 system uses SELinux (Security Enhanced Linux), so we should specify larger inodes than the default of 256 using the `-i` parameter. Recommended value is 512. Notice that the XFS formatter notifies you if it finds a recognized filesystem already on the partition. Note also that the UUID was reassigned by the XFS format.

Creating an XFS filesystem

```
[root@echidna ~]# mkfs -t xfs -i size=512 /dev/sda8
mkfs.xfs: /dev/sda8 appears to contain an existing filesystem (xfs).
mkfs.xfs: Use the -f option to force overwrite.
[root@echidna ~]# mkfs -t xfs -f -i size=512 /dev/sda8
meta-data=/dev/sda8              isize=512    agcount=4, agsize=2622108 blks
         =                       sectsz=512   attr=2
data     =                       bsize=4096   blocks=10488429, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =internal log           bsize=4096   blocks=5121, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[root@echidna ~]# blkid /dev/sda8
/dev/sda8: UUID="1b6798f2-f07f-4d5e-af06-2470df37ddb3" TYPE="xfs"
```

You can label an XFS system using the `-L` option with a name. You can use the `xfs_admin` command with the `-L` option to add a label to an existing XFS filesystem. Use the `-l` option of `xfs_admin` to display a label. Unlike ext2, ext3, and ReiserFS, XFS labels have a maximum length of 12 characters.

Creating a ReiserFS filesystem

You create a ReiserFS filesystem using the `mkfs` command with the `-t reiserfs` option or the `mkreiserfs` command. ReiserFS does not support SELinux and is being replaced by Resier4

You can label a ReiserFS system using the `-l` (or `--label`option with a name). You can use the `reiserfstune` command to add a label or display the label on an existing ReiserFS filesystem. Labels are limited to a maximum length of 16 characters.

You may need to install the ReiserFS package on your system to use ReiserFS as it may not be included in a default install. See the man or info pages for more details.

Creating a vfat filesystem

We'll now create the FAT32 (vfat) filesystem on /dev/sda9.

Creating a vfat filesystem

```
[root@echidna ~]# mkfs -t vfat /dev/sda9
mkfs.vfat 3.0.9 (31 Jan 2010)
[root@echidna ~]# blkid /dev/sda9
/dev/sda9: LABEL="" UUID="CF72-99A8" TYPE="vfat"
```

UUIDs for vfat filesystems are shorter than normal UUIDs and therefore somewhat less likely to be unique. If you want a label instead, use the `dosfslabel` command. Labels for DOS partitions are limited to 11 characters in length.

Creating swap space

Now let's create some swap space on the /dev/sda4 partition using the `mkswap` command as shown in .

Creating swap space

```
[root@echidna ~]# mkswap /dev/sda4
Setting up swapspace version 1, size = 4192960 KiB
no label, UUID=8f5a3a05-73ef-4c78-bc56-0e9b1bcc7fdb
```

Note that recent versions of `mkswap` show you the generated UUID.

Unlike regular filesystems, swap partitions aren't mounted. Instead, they are enabled using the `swapon` command. Your Linux system's startup scripts will take care of automatically enabling your swap partitions.

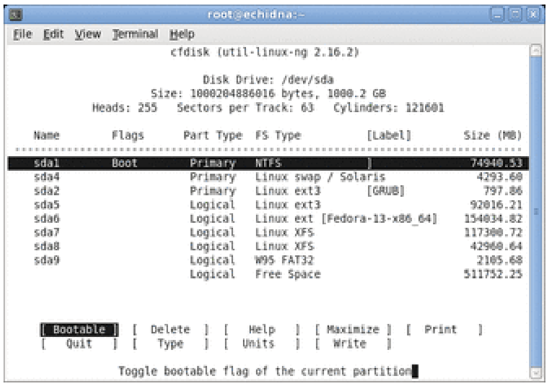
Other tools and filesystems

The following tools and filesystems are not part of the LPI objectives for this exam. This very brief overview touches on some of the tools and filesystems that you may encounter.

Partitioning tools

Many Linux distributions include the `cfdisk` or `sfdisk` commands. The `cfdisk` command provides a more graphical interface than fdisk, using the ncurses library functions as shown in . The `sfdisk` command is intended for programmer use and can be scripted. Use it if you know what you are doing.

Using cfdisk



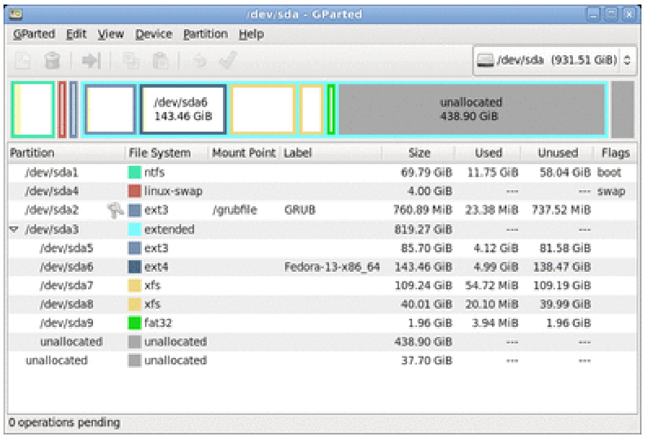
```
                      cfdisk (util-linux-ng 2.16.2)

                          Disk Drive: /dev/sda
                Size: 1000204886016 bytes, 1000.2 GB
        Heads: 255   Sectors per Track: 63   Cylinders: 121601

    Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
    sda1        Boot        Primary   NTFS             ]               74940.53
    sda4                    Primary   Linux swap / Solaris              4293.60
    sda2                    Primary   Linux ext3       [GRUB]            797.86
    sda5                    Logical   Linux ext3                       92016.21
    sda6                    Logical   Linux ext [Fedora-13-x86_64]    154034.82
    sda7                    Logical   Linux XFS                       117300.72
    sda8                    Logical   Linux XFS                        42960.64
    sda9                    Logical   W95 FAT32                         2105.68
                            Logical   Free Space                      511752.25


     [ Bootable ]  [  Delete  ]  [   Help   ]  [ Maximize ]  [  Print   ]
     [   Quit   ]  [   Type   ]  [  Units   ]  [  Write   ]

              Toggle bootable flag of the current partition
```

Another popular tool for working with the partition table is `parted`, which can resize and format many partition types as well as create and destroy them. While `parted` cannot resize NTFS partitions, `ntfsresize` can. The `qtparted` tool uses the Qt toolkit to provide a graphical interface. It includes the `parted` functions as well as `ntfsresize` functions.

The `gparted` tool is another graphical partitioning tool, designed for the GNOME desktop. It uses the GTK+GUI library, and is shown in . (See Resources for links to both `qtparted` and `gparted`.)

You may have to install the above packages in order to use them as they may not be part of your default install.

Using gparted



Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
/dev/sda1	ntfs			69.79 GiB	11.75 GiB	58.04 GiB	boot
/dev/sda4	linux-swap			4.00 GiB	---	---	swap
/dev/sda2	ext3	/grubfile	GRUB	760.89 MiB	23.38 MiB	737.52 MiB	
/dev/sda3	extended			819.27 GiB	---	---	
/dev/sda5	ext3			85.70 GiB	4.12 GiB	81.58 GiB	
/dev/sda6	ext4		Fedora-13-x86_64	143.46 GiB	4.99 GiB	138.47 GiB	
/dev/sda7	xfs			109.24 GiB	54.72 MiB	109.19 GiB	
/dev/sda8	xfs			40.01 GiB	20.10 MiB	39.99 GiB	
/dev/sda9	fat32			1.96 GiB	3.94 MiB	1.96 GiB	
unallocated	unallocated			438.90 GiB	---	---	
unallocated	unallocated			37.70 GiB	---	---	

0 operations pending

Earlier you saw the `gdisk` command which is quite similar in operation to `fdisk`. Use the `gdisk` command for GPT disks and the `fdisk` command for the more traditional MBR layout. Both `parted` and `gparted` can handle GPT layout as well as MBR layout, so you might find it more convenient to use one of these tools to handle both types.

Many distributions allow you to partition your disk, and sometimes shrink an existing Windows NTFS or FAT32 partition, as part of the installation process. Consult the installation documentation for your distribution.

Logical volume manager

The logical volume manager (or LVM) for Linux allows you to combine multiple physical storage devices into a single *volume group*. For example, you might add a partition to an existing volume group, rather than having to carve out contiguous space large enough for your desired filesystem. The article "Learn Linux 101: Hard disk layout" has more information and an example of creating filesystems with LVM.

RAID

RAID (Redundant Array of Independent Disks) is a technology for providing a reliable data storage using low-cost disks that are much less expensive than the disks found on high-end systems. There are several different types of RAID, and RAID may be implemented in hardware or software. Linux supports both hardware and software RAID.

More filesystems

Connect with Ian

Ian is one of our most popular and prolific authors. Browse all of Ian's articles on developerWorks. Check out Ian's profile and connect with him, other authors, and fellow readers.

You will probably encounter filesystems besides those discussed above.

IBM's *Journaled File System (JFS)*, currently used in IBM enterprise servers, is designed for high-throughput server environments. It is available for Linux and is included in several distributions. To create JFS filesystems, use the `mkfs.jfs` command.

The *btrfs* (B-Tree file system) was initially developed by Oracle and is available under the GPL license. It is a new copy-on-write filesystem for Linux aimed at implementing advanced features while focusing on fault tolerance, repair, and easy administration. This may not be installed by default, so you will probably have to install a package, such as btrfs-progs to enable btrfs support.

There are other filesystems too, such as the cramfs filesystem often used on embedded devices.

Related topics

- A roadmap for LPIC-1
- Linux Professional Institute (LPI) exam prep
- Filesystem Hierarchy Standard