Use the Reiser4 file system under Linux$_{\text{rev.2}}$

An alternative advanced file system designed for daredevils

Check out the ext2 file system (second extended file system), the ext3 (third extended file system), and Reiser4 and learn how to create your own Reiser4 file system. Ext2 was not only the most widely used file system, but also the traditional UNIX® style file system that is ill-suited for today's hard drive sizes. Ext3 file system added journaling, but little else. If you want something really advanced, you may want to try the modern Reiser4 file system.

Introduction

I have always been fascinated by file systems and hard drives - in the early 1990s I spent a lot of money on a huge (at that time) 80-megabyte hard drive for my computer. There is something magical in the way large data chunks are thrown back and forth in the bus, read and write thousands of files, and run benchmarks.

Maybe you do not share my passion for hard drives and software that monitors files and directories, but it is likely that you are interested in data security, efficient use of the hard drive, and squeezing the maximum performance out of the frail I/O subsystem of your computer.

Yes, it is 'frail'. Unless you spend a lot of money on exotic equipment, hard disk I/O subsystem is not much more advanced than the processor, RAM, and video card. The famous Moore's Law does not apply here, only Micromagnetism and advanced manufacturing technology.

As the most widely used operating system, Linux® has the most extensive support for different file systems. In this respect, Linux is different from other UNIX®-systems, which traditionally support their own file systems and ISO-9660 file system used with standard CD-ROM drives. My Fedora Core 4 system has loadable kernel modules that support two dozen varieties of file systems - primarily to ensure compatibility. You can insert or connect a disk drive from almost any other computer system and operate it under Linux. But what if you are installing a new drive in a Linux system and you do not need to use it in a Windows®, QNX, Mac OS X or Minix?

Then it is necessary to know a little more about some of the basic Linux file systems, such as ext2 (second extended file system), ext3 (third extended file system), and reiserfs 4 (prospective file system with many interesting features that improve the system's file processing capabilities).

Preparation

If you run a Linux distribution that does not support Reiser4 (such as Arch, Linspire, or SUSE), you have to perform quite complex operations to rebuild the kernel. Instructions on recompiling the kernel would require me to write a separate manual, so consult the manual for your particular distribution. It will guide you to perform the steps required to recompile the kernel.

Before you actually start to compile the Linux kernel, you must visit the Reiser4 ~~Namesys~~ https://reiser4.wiki.kernel.org home page (see. Section Resources) and download the Reiser4 patch appropriate for your kernel. These patches contain instructions on how to apply them before you configure and compile the kernel.

To create the Reiser4 file system and manipulate it, you will need reiser4progs package. If your Linux distribution does not provide the reiser4progs package, please visit ~~Namesys~~ https://reiser4.wiki.kernel.org again and download it (See Resources.).

If you want to experiment with Reiser4, the perfect solution - Gentoo Linux Live CD with Reiser4 support. See Link in the Resources section.

Linux File Systems

At the beginning of the birth of Linux (in late 1993), its kernel only supported one file system - the port was a very minimalist Minix file system. This file system had a number of limitations: including a 14-character file name length and maximum file system size limited to 64MB. It even did not support all the attributes inherent in UNIX file systems, in particular the creation, modification, and access time stamps required for compliance with the POSIX standard file system (Portable Operating System Interface).

Due to the limitations of Minix file system, developers begun work on its replacement. The result was a level of abstraction of a virtual file system (VFS), which simplified the writing of file systems for Linux. With the new VFS, the Minix file system was enhanced by the addition of support for longer file names and an increase in the file system capacity (up to 2 GB). This version is called the extended file system (ext), but it still had limitations.

Many of these limitations were overcome in the ext2 file system, which is still used in many systems and was the default file system for Linux for a long time. By adding journaling functionality to ext2, its successor ext3 was created.

ReiserFS (also known as Reiser3) is the first advanced file system in Linux that supports journaling and more efficient disk usage than any other journal-based file system used in Linux. Its successor, Reiser4, has been thoroughly redesigned and rewritten, while maintaining data security and efficiency, focusing primarily on scalability, security and performance. Reiser4 is not included in the Linux kernel 2.6, which is usually a sign of a possible instability or other reasons for caution. Regardless of the file system you are using, remember to always make backup copies of important data.

Let's do a quick look at ext2, ext3, and Reiser4 file systems.

Traditional file system: ext2

Ext2 – was the default file system for Linux and is a traditional UNIX file system (based on Berkeley Fast Filesystem, FFS). It has a maximum file name length of 255 characters, and the theoretical maximum file system size of 4 terabytes. (Linux block device driver can not exceed 2047 gigabytes, when you can buy a disk with such a large storage space, do not forget to tell me).

Because of its widespread use, there are ext2 drivers for Windows and Mac OS X. They allow you to read and write ext2 partitions directly from these operating systems, making it an excellent format for sharing devices such as portable hard drives.

Ext2 file system supports all standard UNIX features:

- Owner user ID and group ID.
- Bits to control user, group, and other permissions and operating system flags.
- Keeping records creation, modification and access time (although most systems run with a disabled access time control to improve performance at the cost of compatibility with the standard POSIX 1003.1).

The main disadvantage of ext2 is that the size of hard drives is much larger than its original design. After a system crash or power failure, the file system has to be checked with the help of fsck -- a very time consuming operation in modern drives with a lot of folders and files.

Traditional, but journaled file system: ext3

The ext3 file system for Linux is used by default in most modern distributions. Compared with ext2, it adds:

- Journal metadata, providing a reliable state of the filesystem. This eliminates the need to carry out lengthy checks using fsck after a system crash or power failure.
- Indexing directory tree to speed access to large catalogs.
- Resize on the fly and the ability to upgrade the file system from ext2 to ext3 without reformatting the hard disk.
- Increasing the maximum size of the file and the file system (2 and 32 terabytes, respectively).

Although ext3 is inferior in speed and scalability to its competitors (such as Reiser3 or SGI's excellent XFS), it is compatible with ext2, which makes it attractive because it has a lot of mature ext2 utilities for support and administration.

Batman Machine: Reiser4

Although the Reiser3 file system has gained some popularity, thanks to its speed and journaling support (today it has become the default file system for some Linux distributions), its creators did not relax. Reiser4 is written from the ground up and includes interesting additional features:

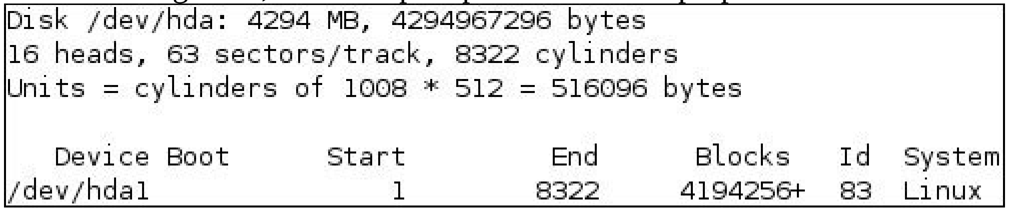
- Effective logging with event logging.
- Effective storage of small files, resulting in increased speed and better use of disk space.
- Fast processing of very large directories with hundreds of millions of files (yes, millions of files in one directory without loss of performance).
- A flexible plug-in infrastructure that makes it easy to add compression and encryption

functionality at any time in the future.
- Atomic modification of the file system, which always guarantees its consistent state.
- Dynamic disk optimization on the fly.
- Transaction support in the style of databases.

But wait, why did I call it "Batman's car"? Reiser4 supports a lot of interesting features that you might never need, because Linux VFS does not expose this functionality - as well as many possibilities of Batman's car will not be needed for you on your way from home to work.

Familiarity with Reiser4

Before you do anything interesting with the Reiser4 file system, you need to format a partition with it. As seen in Figure 1 , I have a spare partition for this purpose:



```
Disk /dev/hda: 4294 MB, 4294967296 bytes
16 heads, 63 sectors/track, 8322 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1               1        8322     4194256+  83  Linux
```

Figure 1: Waiting to format the partition

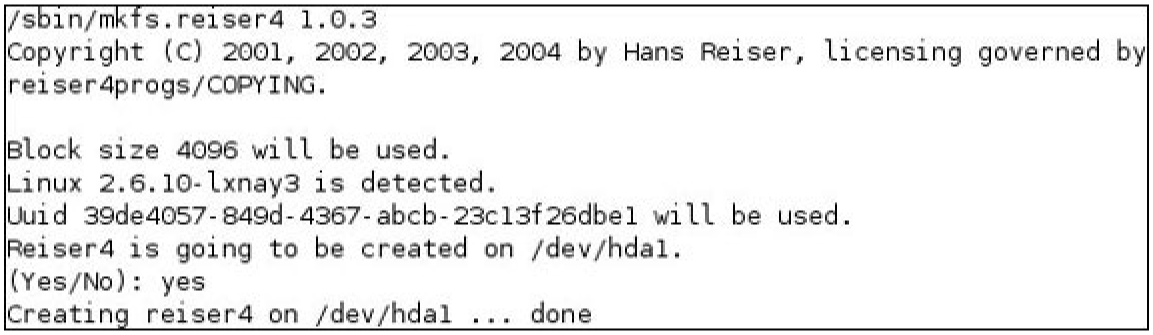
We need to create a new Reiser4 file system on that partition, and then mount it.

Creating a Reiser4 file system

To create the file system, log on as an administrator - root (or use the command sudo to obtain administrator rights) and use the command mkfs.reiser4:

```
/sbin/mkfs.reiser4 -L myLabel /dev/hda1
```

This command creates a Reiser4 file system on the specified partition (I chose /dev/hda1) Tagged "mylabel" and a random unique identifier, as shown in Figure 2 .



```
/sbin/mkfs.reiser4 1.0.3
Copyright (C) 2001, 2002, 2003, 2004 by Hans Reiser, licensing governed by
reiser4progs/COPYING.

Block size 4096 will be used.
Linux 2.6.10-lxnay3 is detected.
Uuid 39de4057-849d-4367-abcb-23c13f26dbe1 will be used.
Reiser4 is going to be created on /dev/hda1.
(Yes/No): yes
Creating reiser4 on /dev/hda1 ... done
```

Figure 2. Creating a Reiser4 file system

Freshly Reiser4 file system is ready! Now you need to mount it to start using.

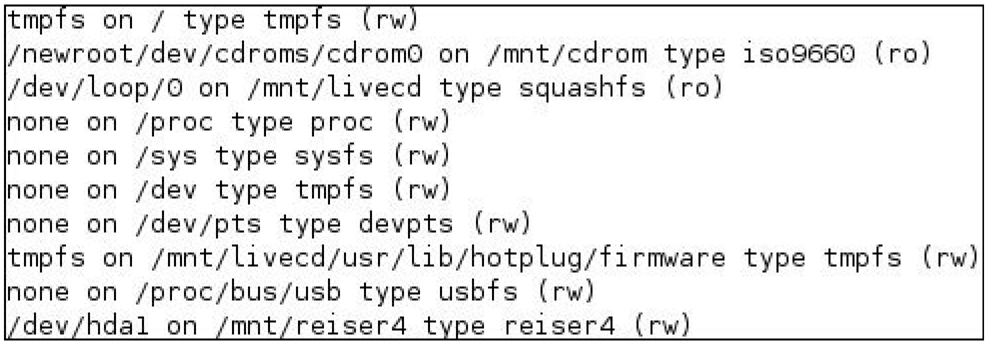
Mounting

To mount a new file system, you need to log in as the administrator (or use the command sudo for the root) and execute the commands mkdir and then mount:

```
mkdir /mnt/reiser4
mount /dev/hda1 /mnt/reiser4
```

First mkdir creates a mount point and then the mount command mounts the device on our file system at the mount point we just created.

You can run the command mount without any arguments to get a list of currently-mounted file systems, which should include a new Reiser4 file system (see. Figure 3).



```
tmpfs on / type tmpfs (rw)
/newroot/dev/cdroms/cdrom0 on /mnt/cdrom type iso9660 (ro)
/dev/loop/0 on /mnt/livecd type squashfs (ro)
none on /proc type proc (rw)
none on /sys type sysfs (rw)
none on /dev type tmpfs (rw)
none on /dev/pts type devpts (rw)
tmpfs on /mnt/livecd/usr/lib/hotplug/firmware type tmpfs (rw)
none on /proc/bus/usb type usbfs (rw)
/dev/hda1 on /mnt/reiser4 type reiser4 (rw)
```

Figure 3. The mounted file systems, including Reiser4

Now that we manually mounted a new file system, we will make sure that it is mounted automatically.

Automatically mount a file system

For the system to automatically mount the new Reiser4 volume, you must register the appropriate instructions in the file /etc/fstab.

Use of the administrator account (or use sudo for root) is needed to make changes to the /etc/fstab, in a text editor, by adding the string specified below:

```
/dev/hda1    /mnt/reiser4    reiser4    defaults    0 0
```

The /etc/fstab directive should specify the desired device and the mount point for the file system. After the device and mount point, you need to determine the file system type and file system options (the defaults setting is usually the best if you do not know what you are doing and do not have good documentation on the file system). The last two parameters are "backup flag" and "flag of the fsck check", necessary for historical reasons.

By unmounting the file system and using the mount command to automatically mount all the file systems, you can check the accuracy of editing /etc/fstab correctly:

```
umount /mnt/reiser4
mount -a
```

Now, when you type the command mount without arguments, the result should look exactly the same as before (see. Figure 3). The new file system will be automatically mounted at boot time along with other file systems.

Improved performance and customize the behavior of the volume

Like many other file systems for Linux, Reiser4 has a number of options that can be used to improve the overall performance and modify its behavior. These options can be passed to the command mount using the parameter -o, as can be seen from the following example:

mount -t reiser4 -o option**1**,option**2**,...,option**n** /dev/hda1 /mnt/reiser4

You can include multiple file system options, separated by commas.

In this way, the system can read these parameters at startup, or you can include these file system options in the file /etc/fstab:

```
/dev/hda1   /mnt/reiser4      reiser4     option1,option2,...,optionn   0 0
```

Most common options include:

- defaults - Standard file system parameters for Linux. This option is equivalent to specifying the following parameters: rw, suid, dev, exec, auto, nouser, async. The file system will be mounted in read-write mode, the set-UID bit will take effect, the handling of special block and character devices will take place in the usual manner, binary files will be executed, the file system will be mounted automatically, all input-output operations are performed asynchronously.

- noatime – Do not update the access time field when reading a file or directory. This is not strict POSIX behavior, but can greatly increase the speed of file system operations, particularly in file systems with a large number of directories and files that are typically used to read, but not write.

- noexec – Do not to run binaries from the file system. The file system is deemed to contain only data. This can be useful if you do not really trust the source files and binary code, located in the file system.

- nosuid - Ignore user and group IDs of files stored in this directory; Another security option in case you do not trust the source files.

- ro - File system is mounted read-only mode. Attempts to burn or create a new file or directory will not work.

- data=journal – All data will be diverted before writing data to the file system, not just writing the file system metadata to the journal. This ensures the integrity of the data after emergency situations, but seriously reduces write performance.

Typically, you can use the default values for these options, but you can add noatime setting to safely improve execution speed. data=journal option can be useful for very important CVS-servers or file systems used for backup, where data integrity is more important than performance.

Summary

Adding a new file system to Linux can be a daunting task, especially if we are talking about one of the plurality of alternative supported Linux file systems. Understand the relevant features of the popular file systems to help you make a sensible decision.

Reiser4 file system should still be regarded as experimental (despite the fact that many people use it without problems), because it has not yet been included in the Linux kernel. Namesys developers are working hard to make their code part of the kernel, so the emergence of distributions based on Reiser4 is only a matter of time.

After you create a file system using the appropriate mkfs command, use the mount command (and add entry in the file /etc/fstab) to mount it -- so you can start using the file system. After completing these tasks, remember to make regular backups of important data without waiting for the hard drive to fail.

[Article originally appeared at IBM DeveloperWorks in English; notwithstanding, since it is not there anymore, it was translated from Russian – and enhanced from Chinese -- via Google Translate.]
https://www.ibm.com/developerworks/ru/library/au-unix-reiserFS/
https://www.ibm.com/developerworks/cn/aix/library/au-unix-reiserFS/

Chris Herborth for more than 10 years has been writing about operating systems and programming. He won awards as a senior technical writer. If he does not play with his son Alex or just spends time with his wife, Chris devotes his spare time writing articles and researching video games (that is, the game).