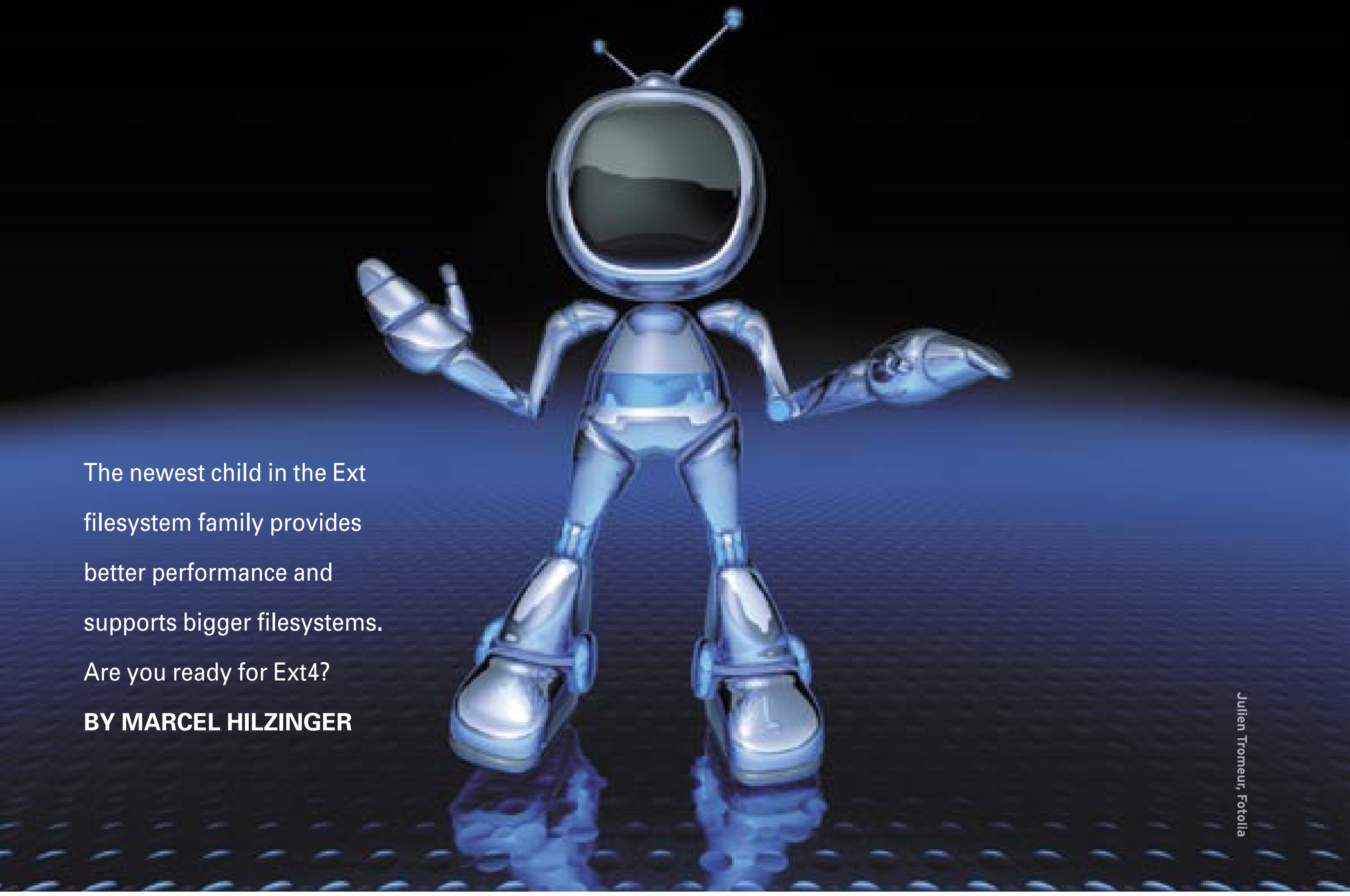
The new Ext4 filesystem

FIT FOR THE FUTURE



The newest child in the Ext filesystem family provides better performance and supports bigger filesystems. Are you ready for Ext4?

BY MARCEL HILZINGER

Julien Tromeur, Fotolia

Linux came of age with the introduction of the Ext2 filesystem, and Ext3 was the first journaling filesystem to make it into the kernel. Through the years, the Ext (Extended) filesystem family has held an important place in Linux, and Ext remains a popular choice for users around the world. But the venerable Ext code base is showing signs of age. Several months ago, developers began to discuss the problem of adding new features to Ext3 while maintaining the stability and compatibility of the code [1]. Meanwhile, several initiatives offered enhancements to the filesystem that were too fundamental to integrate easily but too important to ignore.

The solution proposed by the developers was to fork the code and start work on a new filesystem called Ext4. This strategy lets users around the world continue to use the stable Ext3 while the developers integrate and test a new round of enhancements. Ext4, the next generation of the Extended filesystem, is now available for users. We took a look at Ext4 to see how it compares with other popular filesystem options.

What Changes?

Ext4 moves to better 64-bit support, which leads to several other enhancements. One important benefit unveiled with Ext4 is an increase in the maximum filesystem size. Ext3's maximum filesystem size of 8TB (with a default block size of 4KB) was once considered huge, but contemporary hardware cries out for more. Ext4 now supports filesystems of up to 1024 petabytes (PB).

Another major change with Ext4 is the use of *extents*. An extent is a contiguous area of additional disk space saved with a file. The purpose of an extent is to ensure that later changes will be stored in the same location as the original file. Extents reduce fragmentation and improve write speeds. Several other popular filesystems, such as Reiser4, NTFS, and the Macintosh HPS filesystem, already support extents.

Backward compatibility was also a goal for the Ext4 developers: You can mount an Ext4 partition as Ext3, and you can mount an Ext3 partition as Ext4 (with the ext4dev filesystem type). Unfortunately, to achieve this Ext3/Ext4 compatibility, you have to turn off Ext4's extent feature, which is one of the major performance benefits of Ext4.

If you are interested in testing Ext4, you need a current kernel and the usual developer tools: make, GCC, and glibc-devel. In our lab, we used kernel 2.6.20. Working as root, unpack the sources in the */usr/src* directory and create a link of */usr/src/linux* that points to the sources:

```
cp linux-2.6.20.tar.bz2 /usr/src
cd /usr/src
tar xvfj linux-2.6.20.tar.bz2
ln -s linux-2.6.20 linux
```

To let the kernel know which kernel elements you want to compile as modules, you need to create a *.config* file in */usr/src/linux*. To do so, you can issue either the *make menuconfig* or *make xconfig* command. Because the manual system with its multitude of options is not easy, you might prefer to use the current kernel as the basis for the new configuration and just add the differences for the new kernel version. To do so, copy the */boot/config-kernelversion* file to */usr/src/linux* and save the file as *.config*. The following command should do the trick for most distributions:

```
cp /boot/config$(uname -r) ↩
/usr/src/linux/.config
```

Again working as root, enter the *make oldconfig* command in the */usr/src/linux* directory to create a new kernel from the existing configuration file. For each change, the system will prompt you for whether you want the change and how you want to implement it. Depending on your kernel version, you might need to answer a number of questions. If you only want to add Ext4, just keep pressing Enter to accept the defaults. Note that the system will not use Ext4 by default. To change this, you need to open the *.config* file and look for the following entry:

```
# CONFIG_EXT4DEV_FS ↩
is not set
```

Remove the pound sign at the start of the line and change the entry to *CONFIG_EXT4DEV_FS = m* (Figure 1). To experiment with Ext4 on your root partition, you will need to build the filesystem into the kernel rather than using a module. In this case, change the entry to *CONFIG_EXT4DEV_FS = y*, launch *make oldconfig* again, and then answer a few questions about Ext4 (Figure 2).

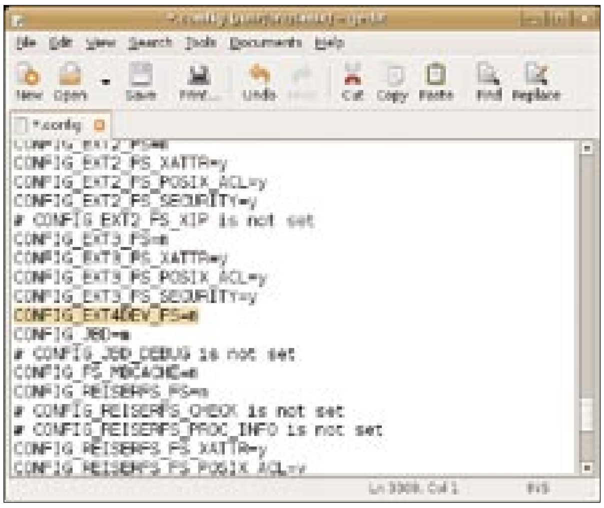
Most current distributions use access control lists (ACLs) and extended attributes to support more granular control over user privileges. If the *mount* command output shows you entries for *acl* or *user_xattr* at the command line, you will need to enable these features on Ext4 too. So when you are prompted to enable *EXT4DEV_FS_XATTR* and *EXT4DEV_FS_POSIX_ACL*, don't forget to say *y*.

You need security labels (*EXT4DEV_FS_SECURITY*) to support SElinux. Make sure you avoid enabling debugging support for JBD2 because this will slow the filesystem down. *make* enables debugging support for JBD (*JBD_SUPPORT*) by default.

After making your decisions, type *make oldconfig* to write the new *.config* file, which now has Ext4 support enabled. Then run *make* to start building the kernel. Depending on how fast your machine is, this can take a couple of hours. After completing the kernel build, enter the following commands to install the module and the kernel:

```
make modules_install
make install
```

Most distributions will set up the initial ramdisk (*initrd*) at this point and also modify the bootloader. When you reboot your machine, you should be able to select the new kernel from the Grub menu. Installing the new kernel will not affect your existing installation.


File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
*.config
Ln 3308, Col 1 INS

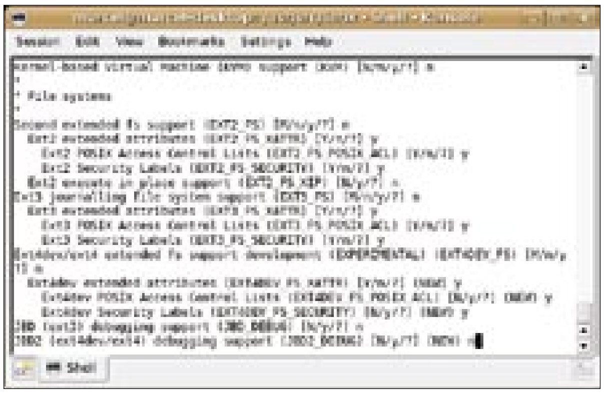
```
CONFIG_EXT2_FS=m
CONFIG_EXT2_FS_XATTR=y
CONFIG_EXT2_FS_POSIX_ACL=y
CONFIG_EXT2_FS_SECURITY=y
# CONFIG_EXT2_FS_XIP is not set
CONFIG_EXT3_FS=m
CONFIG_EXT3_FS_XATTR=y
CONFIG_EXT3_FS_POSIX_ACL=y
CONFIG_EXT3_FS_SECURITY=y
CONFIG_EXT4DEV_FS=m
CONFIG_JBD=m
# CONFIG_JBD_DEBUG is not set
CONFIG_FS_MBCACHE=m
CONFIG_REISERFS_FS=m
# CONFIG_REISERFS_CHECK is not set
# CONFIG_REISERFS_PROC_INFO is not set
CONFIG_REISERFS_FS_XATTR=y
CONFIG_REISERFS_FS_POSIX_ACL=y
```

Figure 1: To make sure the kernel understands Ext4, you need to enable the corresponding option in your .config file.

Working with Ext4

To create an Ext4 partition, you need a recent version of E2fsprogs-Tools [2]. Users of SUSE Linux will find the current SUSE kernel version 2.6.20 and matching E2fsprogrs at *suse.com* [3] or *openSUSE.org* [4]. When you install this kernel, make sure you don't enter the *rpm -Uvh* command, but use *rpm -ivh* instead to keep the openSUSE 10.2 kernel.

To install Ext2-Tools, just follow the usual steps: *./configure*, *make*, and (as root) *make install*. If an error occurs at the *make* step, you will probably need



```
Kernel-based Virtual Machine (KVM) support (KVM) [N/m/y/?] n
*
* File systems
*
Second extended fs support (EXT2_FS) [M/n/y/?] m
  Ext2 extended attributes (EXT2_FS_XATTR) [Y/n/?] y
    Ext2 POSIX Access Control Lists (EXT2_FS_POSIX_ACL) [Y/n/?] y
    Ext2 Security Labels (EXT2_FS_SECURITY) [Y/n/?] y
  Ext2 execute in place support (EXT2_FS_XIP) [N/y/?] n
Ext3 journalling file system support (EXT3_FS) [M/n/y/?] m
  Ext3 extended attributes (EXT3_FS_XATTR) [Y/n/?] y
    Ext3 POSIX Access Control Lists (EXT3_FS_POSIX_ACL) [Y/n/?] y
    Ext3 Security Labels (EXT3_FS_SECURITY) [Y/n/?] y
Ext4dev/ext4 extended fs support development (EXPERIMENTAL) (EXT4DEV_FS) [M/n/y
/?] m
  Ext4dev extended attributes (EXT4DEV_FS_XATTR) [Y/n/?] (NEW) y
    Ext4dev POSIX Access Control Lists (EXT4DEV_FS_POSIX_ACL) [N/y/?] (NEW) y
    Ext4dev Security Labels (EXT4DEV_FS_SECURITY) [N/y/?] (NEW) y
JBD (ext3) debugging support (JBD_DEBUG) [N/y/?] n
JBD2 (ext4dev/ext4) debugging support (JBD2_DEBUG) [N/y/?] (NEW) n
```

Figure 2: The second time you run make oldconfig, you only need to answer questions on Ext4.

Roadwork Ahead!

Athough Ext4 has officially been part of Linux since kernel 2.6.19, work on the new filesystem is far from complete. Takashi Sato, for example, is currently working on an online re-sizer that checks Ext4 for fragmentation on the fly and corrects any issues it finds. The finished Ext4 product will also include an undelete function. Some of these new features, however, mean changing the filesystem format, and this could cause issues with existing Ext4 partitions. Given the current pace of development, you might want to think twice before moving to Ext3's successor, although our tests did not identify any stability problems.

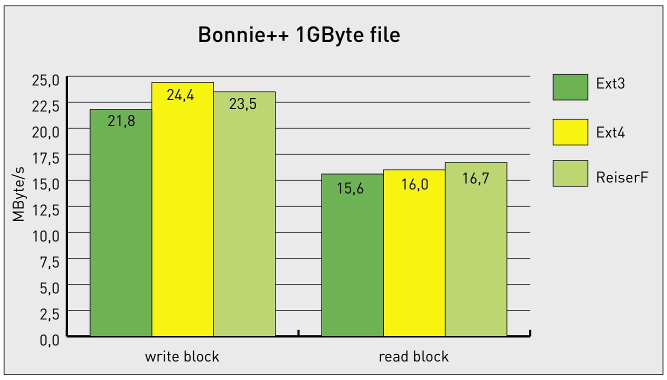


Figure 3: Ext4 writes a 1GB file slightly faster than ReiserFS in a blockwise write operation, and Ext4 is far faster than Ext3.

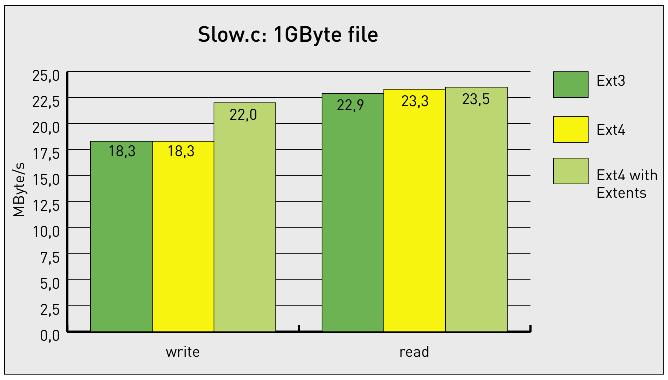


Figure 5: Without extents, Ext4 writes far more slowly and drops down to the level of Ext3.

to install the texinfo package on your machine. After installing the package, just run *make* again.

To create a new partition in Ext4 format, enter *mke2fs -j <partition>*, replacing *partition* with the required device file name. You could also use the GUI to create a new Ext3 partition. After doing so, mount the partition by entering *mount -t ext4dev <partition> <mountpoint>*. The kernel will automatically load the ext4dev module.

If you intend to use Ext4's extents feature, you need to add *-o extents* to the mount command. Note that a partition mounted with extents cannot be mounted as Ext3 later. Be sure not to mount existing Ext3 partitions with *-o extents*. If this happens, unmount the partition without making any changes to files; otherwise, you will only be able to mount the partition as Ext4 in the future. The developers are aiming to migrate support for extents to the *mke2fs* and *tune2fs* tools in the future, which will remove the need for the mount option.

Besides these options, Ext4 supports familiar mount parameters for journaling filesystems. If you have a laptop, the *-o commit=seconds* option might be of interest. The interval you specify here determines how often Ext4 will write data to disk. The default value is every five seconds. Higher values improve performance and can also save power. But note that this compromises the safety of any data that have not yet been saved. This said, our benchmarks failed to reveal any significant difference when we set this option to *commit=60*.

For our first tests with the developer version of Ext4, we put the filesystem through its paces with the *Bonnie++* [5] and *Slow.c* [6] benchmarks. Our choice of distribution was openSUSE 10.2 with a home-grown kernel 2.6.20 and ext4dev as a module. For our tests, we used the *-o extents* mount option for Ext4, unless specified otherwise, and we used defaults for all other filesystems.

Ext4 achieves far faster write speeds than its predecessors and actually overtakes its direct competitor ReiserFS (Figure 3) on blockwise write operations for the first time. This said, ReiserFS still reads slightly faster than Ext4. Ext4 is not much faster than Ext3 when it comes to sequential creating and deleting of files (Figure 4).

The performance effect of the extents feature is clearly demonstrated by the results of the "Random Create" test. It makes sense that Ext4 loses out to Ext3 here because it needs extra time to reserve storage space for the extents. This extra work is not required on deleting files, and this is where Ext4 overtakes Ext3 and ReiserFS, which comes in last in the Bonnie++ stress tests.

Finally, we wanted to know how fast Ext4 is without extents. Figure 5 shows the results of the Slow.c benchmarks with a 1GB test file. The results show that Ext4's exceptional write performance is mainly due to extents.

Conclusions

Ext4 is a fast and future-oriented journaling filesystem that is still at an early stage of development. Although the lack of full downward compatibility to Ext3 might cause occasional confusion, this is no reason not to deploy Ext4 for production use. You might, however, prefer to stick with Ext3 as long as the Ext4 developers refer to the kernel module as ext4dev. ■

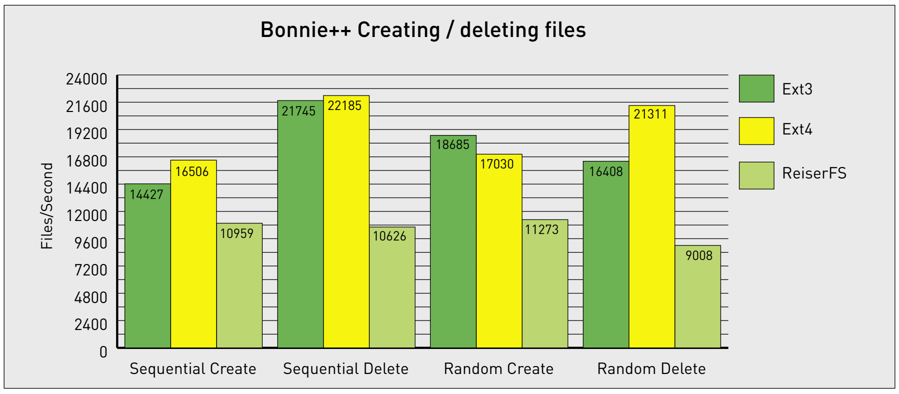


Figure 4: A special Bonnie++ stress test evaluates how many files the filesystem can create or delete in a second. Ext4 scores top marks.

INFO

[1] Ext4 proposal: *http://lkml.org/lkml/2006/6/28/454*

[2] E2fsprogs: *ftp://ftp.kernel.org/pub/linux/kernel/people/tytso/e2fsprogs-interim/*

[3] SUSE kernel: *ftp://ftp.suse.com/pub/projects/kernel/kotd/i386/HEAD*

[4] SUSE E2fsprogs: *http://download.opensuse.org/distribution/SL-OSS-factory/inst-source/suse/i586/*

[5] Bonnie++: *http://www.coker.com.au/bonnie++/*

[6] Slow.c: *http://www.jburgess.uklinux.net/slow.c*