Using filesystem tools

Hand-Formatted

Formatting programs not only take care of installing filesystems. Like well-organized toolkits, they also provide the right tool for the job. If you need more power, you can tune the mount options to squeeze the last drop of performance out of Ext 3, JFS, ReiserFS, or XFS. BY JÖRG REITTER

Filesystem tests do not take too long nowadays. JFS, ReiserFS, and XFS handle large partitions really well. The mkfs tool for JFS and XFS can format a 1 Tbyte partition in no time at all. ReiserFS takes just 30 seconds. Ext 3 is a special case, taking about an hour to do the same job. That should be no surprise as Ext 3 is really Ext 2 at its core.

Besides formatting tools there are any number of utilities for each filesystem. Some of them aim to improve performance, others help troubleshoot errors. This article looks at some of the most important command line tools. The boxes list the mount options, which can be entered on the fly in the command line as parameters of the *mount* command, or stored permanently in your */etc/fstab* file.

Ext 2/3 – Lots of Options

It is a good idea to add a journal to Ext 2, thus updating to the Ext 3 filesystem. The update is quite straightforward. But make sure you have a good backup copy, just in case.

To use Ext 3, you need to enable kernel support for Ext 3. To use Ext 3 for your root partition, you will also need to compile the Ext 3 driver into your kernel. If you boot Linux with an initial RAM disk (initrd), you can implement the driver as a module. After ensuring that your system fulfills the requirements, you can start the upgrade process. Working as the root user, first launch tune2fs, from the e2fsprogs [1] package, to update the Ext 2 system:

```
tune2fs -j /dev/hda2
```

The *-j* flag simply creates the journal, and stores the *.journal* file in the partition root. To complete the update, change the filesystem type in */etc/fstab* from *ext2* to *ext3* and reboot. If you need to mount as Ext 2, unmount the filesystem, and specify the *ext2* filesystem type when remounting. If you need to make the change permanent, revert to the original filesystem type in */etc/fstab*.

The bigger the filesystem, the more time it takes to perform a filesystem

Table 1: XFS Sysctls

Option	Min value	Default	Max value	Explanation
fs.xfs.stats_clear	0	0	1	When set to 1, deletes the statistics in */proc/fs/xfs/stat*.
fs.xfs.sync_interval	HZ	30*HZ	60*HZ	Sets the interval at which *xfssyncd* writes the metadata to disk.
fs.xfs.error_level	0	3	11	Sets the error verbosity level. Generates more or less detailed reports and back-traces for filesystem shutdowns. The following values for Errorlevel are defined: OFF: 0; LOW: 1, HIGH: 5.
fs.xfs.panic_mask	0	0	127	Debugging option. The *BUG()* is called for specific fields.
fs.xfs.irix_symlink_mode	0	0	1	Checks whether to create symbolic links with the default mode (0777), or if the mode is specified by the *umask* preset (Irix mode).
fs.xfs.irix_sgid_inherit	0	0	1	Checks the files that users store in SGID directories. The option deletes the ISGID bit if the group ID for the new file does not match the actual group ID or one of the additional group IDs of the parent directory.
fs.xfs.restrict_chown	0	0	1	Checks whether non-privileged users are allowed to use the *chown* command, to assign other users file permissions.
vm.pagebuf.stats_clear	0	0	1	When set to 1, immediately deletes the statistics in */proc/fs/pagebuf/stat*.
vm.pagebuf.flush_age	1*HZ	15*HZ	300*HZ	The age from which the content of the metadata buffer will be written out to disk.
vm.pagebuf.flush_int	HZ/2	HZ	30*HZ	The interval at which the list of meta data buffers is scanned.

check, when the default interval has expired. If a check takes too long to complete, despite the journal, or if the default interval is too short, you can modify two options. *-i* allows you to set the interval in days, weeks, or months, at which e2fsck will analyze the filesystem. You can also perform a check after a certain number of mounts. Set the *-c* option to do so. Note that a value of *0* for both options will disable filesystem checking.

Formatting Permitted

Choose a small partition for test purposes. If you do not specify any options, the *mkfs.ext3* formatting tool will supply the missing parameters itself. That makes it really easy to format a partition:

```
mkfs.ext3 /dev/sdb2  E
```

The program will assume a block size based on the partition size. To prevent this, you need to set *-b* to specify the block size (1, 2, or 4 KBytes) yourself:

```
mkfs.ext3 -b 2048 /dev/sdb2
```

mkfs.ext3 has a number of options to deal with special requirements. *-T* allows you to specify the typical file size on the filesystem. The tool will set additional optional parameters based on this setting. The predicted size can be between 4 KBytes (*news*) and 4 MBytes (*largefile4*). The following command assumes an average value of 1 MByte:

```
mkfs.ext3 -T largefile
```

Checking for Bad Blocks

Before you install the filesystem, you should always check your hard disk for errors. *-c* tells the mkfs.ext3 program to check the disk for bad blocks. This is always a good idea if your hardware has done a fair amount of mileage. If you want to know exactly how healthy your disk is, specify the parameter twice. This tells mkfs.ext3 to perform a read-write test, rather than just a read-only text. Unfortunately, this test does take twice as long.

Mkfs.ext3 also accepts journal options. Users can select to store the journal internally in the superblock (*size=1024*), or externally on another device (*device=/dev/hdc2*). If you opt for the external variant, the journal must exist before you use mkfs.ext3 to set up the filesystem.

Two steps are required. First, you need to format the partition where you will be storing the journal. Ensure that this partition uses the same block size (1, 2, or 4 KBytes) as the filesystem that will be using the journal. Then enter the following command to create the journal for *hdb1* on the external device *hda9*:

```
mkfs.ext3 -O /dev/hdb1 /dev/hda9
```

The filesystem on *hdb1* uses the *-J* flag to find the external journal:

```
tune2fs -J device=/dev/hda9
```

After specifying the options you need, the next step is to simulate creating the

Ext 3: Mount Options

More detailed information on the Ext 3 filesystem is available from */usr/src/linux/Documentation/filesystems/ext3.txt*.

data=journal: Stores the metadata in the journal first, before writing data to the filesystem (metadata and data journaling).

data=ordered: Stores the data on the filesystem first, before modifying the metadata.

data=writeback: Varies: *sync()* writes data at set intervals. The metadata journal can be modified before or following this event.

journal=update: Update of the Ext 3 journals to the current format.

journal=*inum*: Root can use *inum* to specify the Inode number that represents the journal file. This is ignored if a journal already exists.

noload: Does not load the journal.

minixdf: The *df* command acts like on a Minix filesystem.

bsddf: The *df* command acts like on a Berkeley Unix (BSD).

check=none, nocheck: Does not check bitmaps on mounting.

debug: The kernel sends debugging information to the syslog service.

errors=continue: Will continue to mount despite filesystem errors.

errors=remount-ro: Will mount read-only in case of filesystem error.

errors=panic: Will halt the machine on filesystem error.

grpid, bsdgroups: Assigns the same group ID as the parent object to an object.

nogrpid, sysvgroups: New objects are assigned the group ID of the creator.

resuid=*UID*: The user ID of the user permitted to use reserved blocks.

resgid=*GID*: The group ID of the group permitted to use reserved blocks.

sb=*Offset*: Use an alternative superblock at this location.

Note: Ext 2 ignores the quota options, such as *grpquota*, *noquota*, *quota*, and *usrquota*, without issuing an error message.

XFS: Mount Options

More detailed information on the XFS-filesystem is available from */usr/src/linux/Documentation/filesystems/xfs.txt*

biosize=*value*: Sets the preferred size for buffered I/O (default: 64 KByte), where *value* represents the log of the required size.

ikeep: Keep empty Inode clusters instead of assigning them to free disk pool.

logbufs=*value*: Sets the number of log buffers in main memory. Values *2* through *8*. Can increase performance while at the same time impacting free memory.

logbsize=*value*: Sets the log buffer size. Values 16, 32, 64, 128, 256 KBytes. Systems with more than 32 MBytes RAM use 32 KByte by default.

logdev=*device*: Location of external log (journal).

noalign: This option will not align data allocations at the borders of stripe units.

noatime: Do not update the timestamp for read access.

norecovery: Mounts the file system without log recovery. The filesystem must be mounted read-only, otherwise the mount operation will fail.

osyncisosync: Updates the timestamp for files where the *O_SYNC* flag is set. Performance may be better without this flag, but files will lose timestamp information in case of a crash..

quota, usrquota, uqnoenforce: Enables user quotas and optionally enforces the limits.

grpquota, gqnoenforce: Enables group quotas and optionally enforces limits.

sunit=*value* and swidth=*value*: Sets the size of the stripe unit in steps of 512 bytes (block units). *swidth* must be set to a multiple of *sunit*.

For Raid devices note that root can overwrite the information in the superblock, if the disk layout is modified after creating the filesystem. nouuid: No UUID check for multiple mounting of filesystems. Useful for mounting LVM snapshots.

filesystem by setting the *-n* flag. This is a good way of finding out if your options work. There are more options. For example, the root user can reserve more than five percent of the partition space for his own use (*-m*), or disable some default Ext 3 functions (*-O*).

Dumpe2fs outputs the content of the superblock and the block groups. e2fsck, performs filesystem checks. This does not take long to complete on an Ext 3 system, thanks to the journal. The exception to this rule is the case where the superblock is damaged, and needs to be checked. With LVM, you can use e2fsadm to reduce the size of the device; the tool can handle online and offline resizing. The Ext 2 dump and restore utilities will work with Ext 3.

JFS Low-Fat Diet

The JFS port is advanced enough for test purposes, but the lack of options means it is not suitable for production use on Linux. On the other hand, JFS produces excellent benchmark results. The kernel code has a debug interface for advanced analysis. In user-space, admins can try the jfs_fscklog tool, a jfsutils [2] tool that extracts the content of the fsck service logfile and outputs it to the console.

JFS does not provide a lot of user options. For example, there is no way of setting the block size. This said, the manpages are well maintained, and provide useful examples, in contrast to the documentation for other filesystems. Some mount options are also aimed at error handling, and are much discussed topics in various mailing lists.

Lack of JFS Tools

The *mkfs.jfs* tool creates a filesystem. You can set the *-c* flag to check the disk for bad blocks before formatting. JFS also supports exporting the journal to another device (*-J*):

```
mkfs.jfs -J journal_dev ➥
external-journal
```

will create the journal on the specified device. The *device* option is then used to bind the journal to the filesystem:

```
mkfs.jfs -J device=external-➥
journal device
```

The *device* referred to at the end of this line, represents the partition where the filesystem will reside. You can either opt for *journal_dev* or *external-journal* in a command. If you need to retain OS/2 compatibility, set the *-O* flag to enable case-insensitive support.

There are another three JFS applications: *jfs_tune* displays the volume label and UUID. *jfs_debugfs* to replace the data at a specific offset with a hex string. *jfs_fscklog* extracts and displays the contents of the fsck service logfile.

ReiserFS v3.6 is a stable filesystem that has enough performance to satisfy power users. The Reiserfsprogs [3] package contains various tuning and error recovery tools. However, there are no dump or restore programs, although it is possible to resize the filesystem.

Although ReiserFS is the standard filesystem for Suse Linux 9.0, there is no supporting documentation on the discs. Suse includes the formatting tool, *mkfs.reiserfs*, but don't expect to discover manpages or help files in the */usr/share/doc* directory.

The kernel documentation is just as taciturn. Your only options are to surf the Web, especially the developer, Hans Reiser's website [3]. This is worth doing anyway, as the page has patches for the brave, and mailing list subscribers provide answers to various questions.

The *mkreiserfs* command creates a filesystem. There is an option to set the block size, but ReiserFS v3.6 only supports a block size of 4096 Bytes. This restriction will be lifted with the release of version 4, which will be some time this year. ReiserFS selects a few additional options, such as the hash algorithm (*r5*, *rupasov*, *tea*), or whether to install version 3.5 or 3.6, itself, if you fail to specify them.

Full Speed Ahead with Reiser

The journaling options are restricted to the size (*-s*) of the journal, storing the file on an external device (*-j*), and the offset at which the journal starts (*-o*). Experts will typically specify the maximum transaction size. It is fairly safe to

ReiserFS: Mount Options

More detailed information on the ReiserFS-filesystem is available from *http://www.namesys.com/mount-options.html*

conv: Allows a ReiserFS v3.6 to mount a v3.5 filesystem using v3.6 code for any objects to be created. The v3.5 utilities will not work on this filesystem after this operation.

dontpanic: Ignores input/output errors while editing the journal. This option is only available in *raw* mode.

hash=rupasov: Very quick hash function for enormous directories and lots of unusual filenames. Dangerous, as hash collisions are to be expected.

hash=tea: Provides a high degree of randomness and a low probability of hash collisions. Impacts performance, but should be used if the *r5* hash issues a *EHASHCOLLISION* message.

hash=detect: Only makes sense when mounting an older filesystem for the first time. Recognizes the hash function and writes it to the superblock.

hash=r5: Standard hash.

hashed_relocation: Attempts to tune the block allocator. May improve performance. See also, *no_unhashed_relocation*.

no_unhashed_relocation: Attempts to tune the block allocator. May improve performance. See also, *hashed_relocation*.

noborder: Disables the *border allocator algorithm*. May improve performance.

nolog: Disables journaling. May improve performance. Caution: This automatically disables quick recovery after a crash.

notail: Disables the *tail* feature. If you have trouble with a program (such as Lilo) in normal operations, you should avoid storing extremely small files directly in the tree with ReiserFS.

pgc=low, high: Sets the activity level of the Passive Garbage Collector. Only works with the *reiserfs-raw* driver, which is not included with the standard kernel.

raw: Mounts the filesystem in *raw* mode. Not included with the standard kernel.

replayonly: Restores the transactions in the journal without mounting the filesystem. This option is only used by *fsck* under normal circumstances.

resize=*value*: Expand a partition on the fly, especially for LVM devices (Logical Volume Management). Needs the Resizer utility from the Reiserfs-utils package. Download from the Namesys FTP Site at: *ftp://ftp.namesys.com/pub/reiserfsprogs*

experiment with these options as mkreiserfs automatically selects a safe size, if the user supplies an invalid option. If you want to set a UUID (Universally Unique Identifier), you can use the *-u* option to specify this parameter.

Repairing ReiserFS

The Reiserfsck check and repair tool is launched on booting the system. If a system crashes, the tool checks the filesystem consistency, using the default *check* setting and restores the latest status from the journal. If *check* discovers a serious error, such as an inconsistency of the filesystem, it prompts the user to do something about it. *Check* displays the option required to repair the system on the console.

It recommends *fix-fixable* for more minor inconsistencies, such as invalid directory entries. If the filesystem is more seriously damaged, the user can try the *rebuild-tree* option. This tells Reiserfsck to rebuild the whole filesystem. This operation can take a lot longer, and lead to total loss of the filesystem.

You need to enter the following if you see a message such as *read_super_block: can't find a reiserfs file system*,:

```
reiserfsck -rebuild-sb
```

The message typically indicates a corrupt superblock, which makes it impossible to access to the filesystem. *rebuild-sb* restores the superblock, if the partition really does contain a filesystem.

Resizing a Filesystem

The resizing tool, *resize_reiserfs*, is still beta. The utility can expand or reduce a filesystem's size. This assumes that the filesystem is not mounted during the operation. If you do not specify a size (*-s*), the tool will expand the filesystem as far as possible. The command syntax is quite straightforward at first glance.

```
resize_reiserfs -s -2G /dev/hdb2
```

reduces the filesystem on *hdb2* by 2 GBytes from its original size. To expand the filesystem by 2 GBytes, simply type *-2G* instead of *+2G*. This is not quite as simple as it seems. Users need to run fdisk and modify the partition size to reflect the filesystem size. When expanding the filesystem size, the partition size needs to be modified beforehand. When reducing, the partition size can be modified later. The Reiserfstune program allows users to define the journal and maximum transaction sizes and to move the journal to another block device.

Debugreiserfs is a troubleshooting tool. If you call the tool without specifying any options, it outputs the superblock for a given device. The *-p* option is quite interesting in this context. It tells Debugreiserfs to extract the filesystem metadata. The ReiserFS team uses the option to remove errors:

```
debugreiserfs -p /dev/hda2/ | ➥
gzip -c > hda2.gz
```

Ext 2/3 users thinking of migrating to ReiserFS should be aware of one restriction. There is no dump and restore tool. The ReiserFS FAQ recommends tar for incremental backups. Suse users can perform a full backup with YaST 2.

XFS: For the Exacting

With the exception of Ext 3, XFS creates an impression of being most complete. It not only provides numerous options, the documentation is also exemplary. XFS is the only filesystem that allows you to set sysctls in the proc filesystem (see Table 1). XFS spoils power users with options for various buffer sizes. Its other advantages are kernel based quota support, and tools to match in the Xfsprogs [4] package. Also, the XFS package has its own dump and restore tools.

Fully Featured

The simplest syntax for formatting a partition is as follows:

```
mkfs.xfs /dev/sda1
```

The following command places the journal on another file system:

```
mkfs.xfs -l logdev=/dev/sdb2,➥
size=10000b /dev/sda1
```

formats the first partition on the first SCSI hard disk, and creates the journal in the second partition on the second disk.

XFS also has a very flexible approach to block sizes, but not on Linux, unfortunately. Block sizes between 512 Bytes and 64 KBytes are possible, but the page size defines the upper limit on Linux. x86 Linux systems have a page size of 4 KBytes, but at least you have the option to choose a smaller size.

XFS groups its options in categories such as filesystem, inodes, journal, and real-time. Users can assign location or size values to these groups, allowing them to define the size of the stripe unit for a RAID system, or specify the percentage of variable inodes. *mkfs.xfs -N* displays an overview.

You can expand an XFS filesystem. Xfs_growfs with the *-d* flag set maximizes the data section. *-D* allows admins to specify the desired size in blocks. The *-r* and *-R* options do the same thing for the real-time section.

Experts have a lot to gain from using XFS. The Xfs_db tool allows you to read or write blocks. The *blockuse* option returns the file using a given block. The wealth of xfs_db options underlines the fact that Linux can hold sway with major Unix systems, thanks to XFS. ■

JFS: Mount Options

More detailed information on the JFS-filesystem is available from */usr/src/linux/Documentation/filesystems/jfs.txt* .

iocharset=*Name*: Character set the system should use, to convert from Unicode to ASCII.

resize=*number*: Expands the volume on the fly by this number of blocks.

nointegrity: Prevents writes to the journal. Useful if you need to restore a volume backup.

integrity: The journal logs changes to the metadata. After a restore users should set this and the *nointegrity* option on remounting.

errors=continue: Will continue to mount despite filesystem errors.

errors=remount-ro: Will mount read-only in case of filesystem error.

errors=panic: Will halt the machine on filesystem error.

INFO

[1] E2fsprogs: *http://e2fsprogs.sourceforge.net/ext2.html*

[2] Jfsutils: *http://www-124.ibm.com/jfs*

[3] Reiserfsprogs: *http://www.namesys.com/download.html*

[4] Xfsprogs: *http://oss.sgi.com/projects/xfs*