NAME

reiserfstune − The tunning tool for the ReiserFS filesystem.

SYNOPSIS

reiserfstune [**-f**] [**-h** | **--help**] [**-j** | **--journal-device** *FILE*] [**--no-journal-available**] [**--journal-new-device** *FILE*] [**--make-journal-standard**] [**-s** | **--journal-new-size** *N*] [**-o** | **--journal-new-offset** *N*] [**-t** | **--max-transaction-size** *N*] [**-b** | **--add-badblocks** *file*] [**-B** | **--badblocks** *file*] [**-u** | **--uuid** *UUID*] [**-l** | **--label** *LABEL*] [**-c** | **--check-interval** *interval-in-days*] [**-C** | **--time-last-checked** *timestamp*] [**-m** | **--max-mnt-count** *count*] [**-M** | **--mnt-count** *count*] *device*

DESCRIPTION

reiserfstune is used for tuning the ReiserFS. It can change two journal parameters (the journal size and the maximum transaction size), and it can move the journal's location to a new specified block device. (The old ReiserFS's journal may be kept unused, or discarded at the user's option.) Besides that **reiserfstune** can store the bad block list to the ReiserFS and set UUID and LABEL. Note: At the time of writing the relocated journal was implemented for a special release of ReiserFS, and was not expected to be put into the mainstream kernel until approximately Linux 2.5. This means that if you have the stock kernel you must apply a special patch. Without this patch the kernel will refuse to mount the newly modified file system. We will charge $25 to explain this to you if you ask us why it doesn't work.

Perhaps the most interesting application of this code is to put the journal on a solid state disk.

device is the special file corresponding to the newly specified block device (e.g /dev/hdXX for IDE disk partition or /dev/sdXX for the SCSI disk partition).

OPTIONS

-h | **--help**
Print usage information and exit.

-j | **--journal-device** *FILE*
FILE is the file name of the block device the file system has the current journal (the one prior to running reiserfstune) on. This option is required when the journal is already on a separate device from the main data device (although it can be avoided with **--no-journal-available**). If you don't specify journal device by this option, reiserfstune suppose that journal is on main device.

--no-journal-available
allows **reiserfstune** to continue when the current journal's block device is no longer available. This might happen if a disk goes bad and you remove it (and run fsck).

--journal-new-device *FILE*
FILE is the file name of the block device which will contain the new journal for the file system. If you don't specify this, reiserfstune supposes that journal device remains the same.

−s | **−-journal-new-size** *N*
N is the size parameter for the new journal. When journal is to be on a separate device - its size defaults to number of blocks that device has. When journal is to be on the same device as the filesytem - its size defaults to amount of blocks allocated for journal by *mkreiserfs* when it created the filesystem. Minimum is 513 for both cases.

−o | **−-journal-new-offset** *N*
N is an offset in blocks where journal will starts from when journal is to be on a separate device. Default is 0. Has no effect when journal is to be on the same device as the filesystem. Most users have no need to use this feature. It can be used when you want the journals from multiple filesystems to reside on the same device, and you don't want to or cannot partition that device.

−t | **−-maximal-transaction-size** *N*
N is the maximum transaction size parameter for the new journal. The default, and max possible, value is 1024 blocks. It should be less than half the size of the journal. If specifed incorrectly, it will be adjusted.

−b | **−−add-badblocks** *file*
File is the file name of the file that contains the list of blocks to be marked as bad on the fs. The list is added to the fs list of bad blocks.

−B | **−−badblocks** *file*
File is the file name of the file that contains the list of blocks to be marked as bad on the fs. The bad block list on the fs is cleared before the list specified in the *File* is added to the fs.

−f | **--force**
Normally **reiserfstune** will refuse to change a journal of a file system that was created before this journal relocation code. This is because if you change the journal, you cannot go back (without special option **--make-journal-standard**) to an old kernel that lacks this feature and be able to use your filesytem. This option forces it to do that. Specified more than once it allows to avoid asking for confirmation.

--make-journal-standard
As it was mentioned above, if your file system has non-standard journal, it can not be mounted on the kernel without journal relocation code. The thing can be changed, the only condition is that there is reserved area on main device of the standard journal size 8193 blocks (it will be so for instance if you convert standard journal to non-standard). Just specify this option when you relocate journal back, or without relocation if you already have it on main device.

-u | **--uuid** *UUID*
Set the universally unique identifier (**UUID**) of the filesystem to *UUID* (see also **uuidgen(8)**). The format of the UUID is a series of hex digits separated by hypthens, like this: "c1b9d5a2-f162-11cf-9ece-0020afc76f16".

-l | **--label** *LABEL*
Set the volume label of the filesystem.*LABEL* can be at most 16 characters long; if it is longer than 16 characters, reiserfstune will truncate it.

-c | **--check-interval** *interval-in-days*
Adjust the maximal time between two filesystem checks. A value of "disable" will disable the time-dependent checking. A value of "default" will restore the compile-time default.

It is strongly recommended that either **−m** (mount-count dependent) or **−c** (time-dependent) checking be enabled to force periodic full **fsck.reiserfs(8)** checking of the filesystem. Failure to do so may lead to filesystem corruption (due to bad disks, cables, memory, or kernel bugs) going unnoticed, ultimately resulting in data loss or corruption.

-C | **--time-last-checked** *timestamp*
Set the time the filesystem was last checked using fsck.reiserfs. This can be useful in scripts which use a Logical Volume Manager to make a consistent snapshot of a filesystem, and then check the filesystem during off hours to make sure it hasn't been corrupted due to hardware problems, etc. If the filesystem was clean, then this option can be used to set the last checked time on the original filesystem. The format of time-last-checked is the international date format, with an optional time specifier, i.e. YYYYMMDD[HH[MM[SS]]]. The keyword **now** is also accepted, in which case the last checked time will be set to the current time.

-m | **--max-mnt-count** *max-mount-count*
Adjust the number of mounts after which the filesystem will be checked by **fsck.reiserfs(8).** If max-mount-count is "disable", the number of times the filesystem is mounted will be disregarded by **fsck.reiserfs(8)** and the kernel. A value of "default" will restore the compile-time default.

Staggering the mount-counts at which filesystems are forcibly checked will avoid all filesystems being checked at one time when using journaled filesystems.

You should strongly consider the consequences of disabling mount-count-dependent checking entirely. Bad disk drives, cables, memory, and kernel bugs could all corrupt a filesystem

without marking the filesystem dirty or in error. If you are using journaling on your filesystem, your filesystem will never be marked dirty, so it will not normally be checked. A filesys- tem error detected by the kernel will still force an fsck on the next reboot, but it may already be too late to prevent data loss at that point.

This option requires a kernel which supports incrementing the count on each mount. This feature has not been incorporated into kernel versions older than 2.6.25.

See also the **−c** option for time-dependent checking.

-M | **--mnt-count** *count*

Set the number of times the filesystem has been mounted. If set to a greater value than the max-mount-counts parameter set by the**−m** option, **fsck.r eiserfs(8)** will check the filesystem at the next reboot.

POSSIBLE SCENARIOS OF USING REISERFSTUNE:

1. You have ReiserFS on /dev/hda1, and you wish to have it working with its journal on the device /dev/journal

boot kernel patched with special "relocatable journal support" patch
reiserfstune /dev/hda1 −−journal−new−device /dev/journal −f
mount /dev/hda1 and use.
You would like to change max transaction size to 512 blocks
reiserfstune −t 512 /dev/hda1
You would like to use your file system on another kernel that doesn't
contain relocatable journal support.
umount /dev/hda1
reiserfstune /dev/hda1 −j /dev/journal −−journal−new−device /dev/hda1 −−make−journal−standard
mount /dev/hda1 and use.

2. You would like to have ReiserFS on /dev/hda1 and to be able to switch between different journals including journal located on the device containing the filesystem.

boot kernel patched with special "relocatable journal support" patch
mkreiserfs /dev/hda1
you got solid state disk (perhaps /dev/sda, they typically look like scsi disks)
reiserfstune −−journal−new−device /dev/sda1 −f /dev/hda1
Your scsi device dies, it is three in the morning, you have an extra IDE device
lying around
reiserfsck −−no−journal−available /dev/hda1
or
reiserfsck −−rebuild-tree −−no−journal−available /dev/hda1
reiserfstune −−no−journal−available −−journal−new−device /dev/hda1 /dev/hda1
using /dev/hda1 under patched kernel

AUTHOR

This version of **reiserfstune** has been written by Vladimir Demidov <vova@namesys.com> and Edward Shishkin <edward@namesys.com>.

BUGS

Please report bugs to the ReiserFS developers <reiserfs-devel@vger.kerne.org>, providing as much information as possible--your hardware, kernel, patches, settings, all printed messages; check the syslog file for any related information.

SEE ALSO

reiserfsck(8), **debugreiserfs**(8), **mkreiserfs**(8)