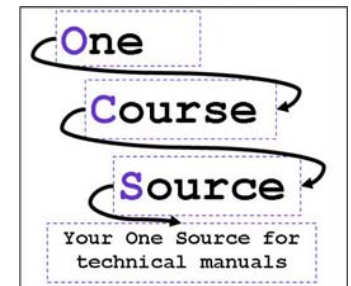
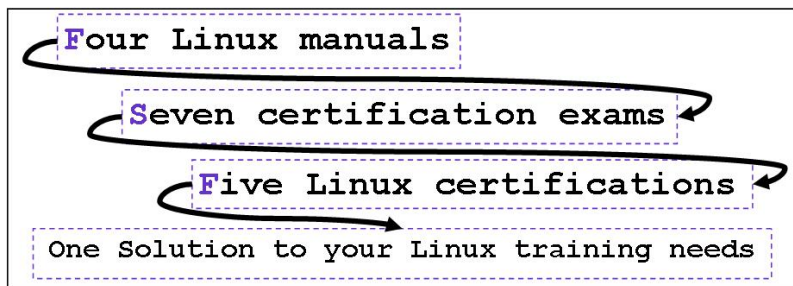


Linux Level 2

Student Manual

Sample Pages



www.onecourse.com

This publication contains proprietary and confidential information, which is the property of One Course Source, 2340 Tampa Ave, Suite J, El Cajon, CA 92020. No part of this publication is reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the prior express written consent of One Course Source.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

REFERENCES TO CORPORATIONS, THEIR SERVICES AND PRODUCTS, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. IN NO EVENT SHALL ONE COURSE SOURCE BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT ARISING OUT OF OR IN CONNECTION WITH THE USE OF THIS INFORMATION.

Descriptions of, or references to, products or publications within this publication do not imply endorsement of that product or publication. One Course Source makes no warranty of any kind with respect to the subject matter included herein, the products listed herein, or the completeness or accuracy of this publication. One Course Source specifically disclaims all warranties, express, implied or otherwise, including without limitation, all warranties of merchantability and fitness for a particular purpose.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. ONE COURSE SOURCE MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

This notice may not be removed or altered.

ver 3.1 - 1/12/06 - 2006Q1

Module Four
Administering the Filesystem

| Module topics: | <u>Page</u> |
|--------------------------------------------|-------------|
| 4.1 Filesystem details | 4 |
| 4.2 The mke2fs command | 14 |
| 4.3 The ext2 and ext3 filesystems | 18 |
| 4.4 Why filesystems break | 21 |
| 4.5 Fixing filesystems with fsck | 22 |
| 4.6 fsck examples | 23 |
| 4.7 Displaying filesystem attributes | 24 |
| 4.8 Modifying filesystem attributes | 26 |
| 4.9 Summary of commands and files | 30 |
| 4.10 Additional Resources | 31 |
| 4.11 Certification notes | 32 |
| 4.12 Lab Exercises | 33 |

Notes:

4.1 Filesystem details

The ext2 and ext3 filesystems have similar organizational structures. Both filesystems consists of the following components:

- Superbock
- Group Blocks
- Backup Superblocks
- Inode Tables
- Data blocks

In this section we will cover these components.

Note: The "disk label" (partition table) and "boot sector" (where a boot loader can be stored) are components that you will see described as part of a partition. They are not, however, part of the filesystem, and will not be covered in great detail in this module.

Notes:

Groups

For organizational reasons, the partition is divided into cylinder groups. Typically there are 16 cylinders per group, but there is a configuration option to allow for different sizes in the **mke2fs** command.

Superblock

The superblock contains information about the partition itself. This information includes:

- FS state (clean, stable or active): The FS state is used by the **fsck** utility when it is run during boot to determine what filesystems to check automatically. See more details on **fsck** in a future section.
- Size of filesystem
- Size of data blocks on filesystem
- Size of the group blocks on this filesystem
- Number of data blocks in filesystem
- Path name of last mount point
- Date and time of last update to the superblock

Typically this data is never updated or accessed by the system administrator. Knowing the contents of the superblock, however, can make understanding the functionality of the filesystem easier.

Notes:

Group Blocks

Each group block has a table of information about the group including:

- The number of inodes in this group block
- The number of directories in this group block
- The number of data blocks in this group block
- A free data block map
- A inode table

Typically this data is never updated or accessed by the system administrator. Knowing the contents of the group block, however, can make understanding the functionality of the filesystem easier.

Backup Superblocks

Since the superblock is critical to the filesystem, backups are made of it in each group block. The backups are made in a “cascade” method in which the first group block has the most recent backup; the second group block has the next most recent backup, etc.

These backup superblocks can be used by the **fsck** utility to recreate the primary superblock. A future section in this manual will cover more details regarding this procedure.

Notes:

Inode Tables

Inode tables are used to store data about files. They basically contain everything about a file except for the data itself and the file name. This information includes:

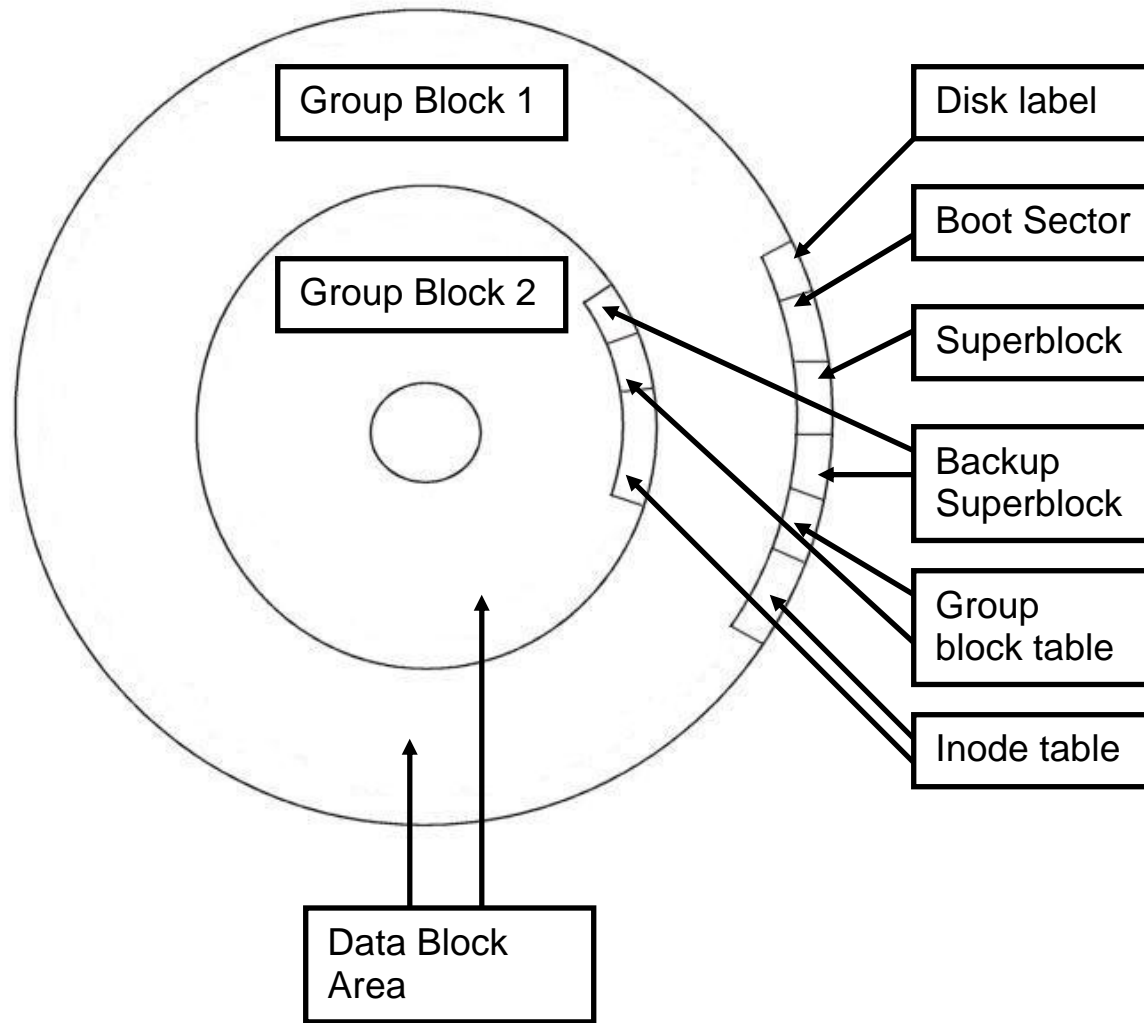
- inode number – a unique number to identify the file
- File type
- Permissions
- Hard link count
- UID of the file owner
- GID of the group owner
- Size of file
- Date/time stamp of last time file was accessed
- Date/time stamp of last time file was modified
- Date/time stamp of last time inode table was modified
- 12 Direct block pointers
- 1 single-indirect pointer
- 1 double-indirect pointer

Data blocks

Data blocks are where the file's data is stored. The default size of data blocks in most distributions is 1 kilobyte.

Notes:

Visual Example of an ext2/ext3 filesystem:



Notes:

File Attributes

File attributes tell the Linux kernel how to treat the file. The following shows the most common attributes:

| Attribute | Description |
|-----------|-------------|
|-----------|-------------|

| | |
|---|-----------------------------------------------------------------------------------------------------------|
| A | Don't update the atime data in the inode |
| a | Only allow file to be opened to append data |
| c | File data is automatically compressed when file is written to and uncompressed when read from |
| d | Don't backup this file when dump command is run |
| i | The file can't be deleted or renamed nor can it's data be changed. |
| j | Forces data and metadata of file to be journaled, despite how filesystem was mounted. |
| s | When the file is deleted, the data blocks are overwritten with zeros. This is good for security concerns. |

To set attributes, use the **chattr** command. The "+" character allows you to add an attribute while the "-" character allows you to remove an attribute:

```
[root@linux2 root]# chattr +d install.log
```

To list file attributes, use the **lsattr** command:

```
[root@linux2 root]# lsattr install.log  
-----d----- install.log
```

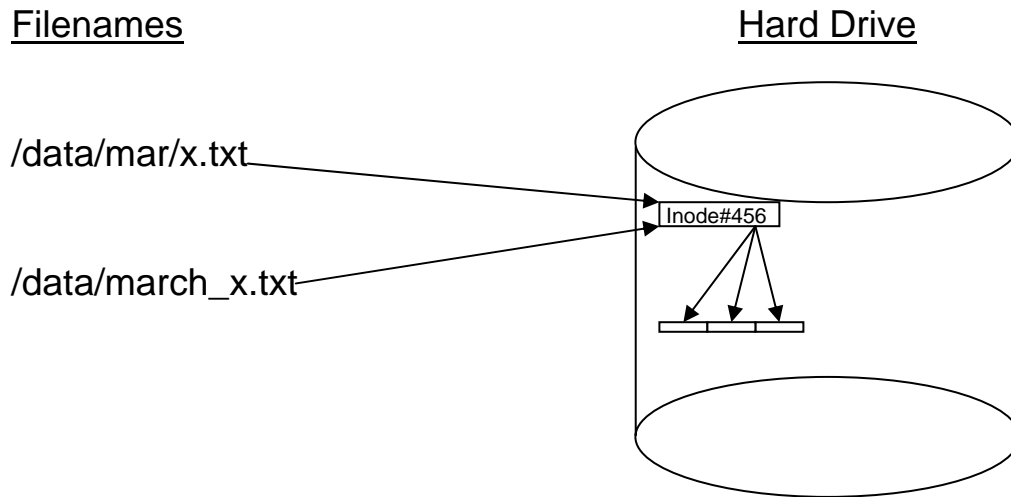
Notes:

Hard Links

When two "files" are said to be hard linked it means that they share the same inode table.

A visual representation of hard linked files:

Filenames



To create a hard link to an existing file, use the **ln** command:

```
ln org_filename new_file_name
```

Notes:

Finding Hard Links

Since hard links share inode tables (and inode numbers) you can find what filenames are hard linked together by looking for files with the same hard links:

```
[root@linux2 root]# ls -li install.log
290882 install.log
[root@linux2 root]# find /root -inum 290882 -ls
290882 40 -rw-r--r-- 2 root root 38369 Dec 16 11:38 /root/install.log
290882 40 -rw-r--r-- 2 root root 38369 Dec 16 11:38 /root/new.install.log
```

Additional notes regarding Hard Links:

- Since each partition (filesystem) has its own inode tables, hard links cannot "span" across partitions:

```
In /etc/inet.d/lp /boot/lp
{results in an error}
```

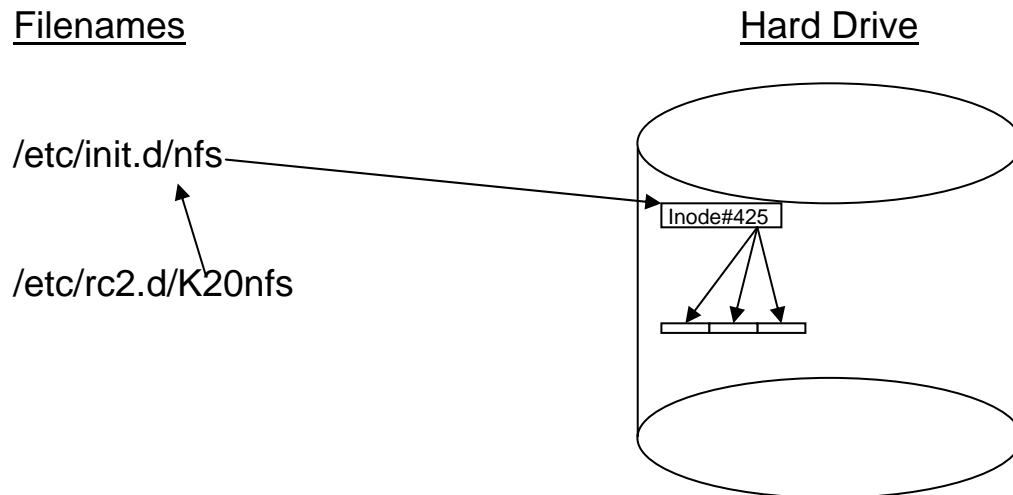
- An advantage hard links is that once you hard link two filenames together, you can remove either one and still have the file.
- You cannot hard link directories.

Notes:

Soft Links

When two "files" are said to be soft linked (also known as symbolic linked) it means one file "points" to the location of another file. A visual representation of soft linked files:

Filenames



Creating soft Links

To create a soft link to an existing file, use the **ln** command with the **-s** option:

```
ln -s org_filename new_file_name
```

Notes:

Finding Soft Links

The **ls -l** command displays soft links.

Soft Link Pros and Cons

- Since soft links point to files via that directory structure, soft links can span filesystems.
- If you delete the original file, the soft link is meaningless.
- You can soft link directories.

Notes:

4.2 The mke2fs command

The **mke2fs** command provides you with more flexibility when creating ext2 and ext3 filesystems. The following features are commonly used.

Specifying the size of the data blocks

On most Linux distributions, the default size of the data blocks in the filesystem is 1024 bytes (or 1 kilobyte), 2048 bytes (2K) or 4096 bytes (4K). The default size is based on the overall size of the filesystem (smaller filesystems getting 1K data blocks, medium getting 2K and very large getting 4K).

When a file is stored in a filesystem, it will "take up" at least one data block. When there are many small files in a filesystem, the result can be a lot of wasted space.

For example, 50 files of 75 bytes of space would take up 204,800 bytes (about 200K) of space in the filesystem rather than 3,750 bytes of space. This problem is inherent to just about all filesystems, not just ext2 and ext3 filesystems.

Typically, 50 small files won't be a large problem. But thousands of small files might be a problem. If you know that you will be storing many small files on the filesystem, consider making smaller block sizes by using the **-b** option to the **mke2fs** command. For example, the following command will create 1024 byte data blocks:

```
mke2fs -b 1024 /dev/hda16
```

Notes:

Specifying the number of inodes

Each file in a filesystem must have an inode table to define it. Each inode table takes up space on the filesystem. Because of the layout of the filesystem, the inode tables must be defined when the filesystem is created.

Since each inode table take up 128 bytes of space, having a large number of inode tables can take up a lot of space in the filesystem. To provide the maximum space in the filesystem, it is best of avoid creating too many inode tables.

The **-i** option allows you to specify how many inodes to create. With this option you specify a number of bytes. The **mke2fs** command determines how many inodes to create by dividing the total size of the filesystem by the number of bytes that you specify with the **-i** option. Therefore, the larger the value you provide is, the less number of inode tables will be created.

Most distributions have a default value of 8192 (8K) for the byte-to-inode ratio. The following command would generate less inode tables than by default:

```
mke2fs -i 16384 /dev/hda16
```

Notes:

Checking for bad blocks

The **-c** option to the **mke2fs** command will check the partition for bad partition blocks before creating the filesystem. Keep in mind that a partition block (typically 512 bytes in size) is not the same as a filesystem block. A bad partition block would typically indicate a physical problem on the hard drive.

Reserving disk space for the superuser

To avoid lack of space problems on filesystems, a percentage of the filesystem is reserved by default for the superuser only (or any daemon that can create files as "root"). By default, 5% of the filesystem will be reserved for the root user. This can be overridden with the **-m** option to **mke2fs**.

Reserving space for root is vital on critical filesystems such as /, /var, and /tmp. In the event that a regular user "fills up" one of these filesystems, the entire OS may crash as critical processes need to be able to write to these filesystems. Space for the superuser should always be reserved on these filesystems.

Other filesystems, such as /home, are not quite as critical. In fact, reserving space for the superuser may be considered a "waste". Consider a 5GB /home partition. Since the root user rarely needs to write to /home, reserving 5% (250MB) results in wasted space.

The argument to **-m** must be given in whole number and is always a percent of disk space to reserve. A value of "0" is allowed on most distributions.

Notes:

Reserving inode tables

Some inode tables must also be reserved for system use. If no inode tables are left in a filesystem, no files can be created by the system process regardless of how much space is left on the filesystem.

The **-N** option to **mke2fs** allows you to specify how many inodes to reserve for system use. Without this option, mke2fs uses a calculation that takes into account the byte-per-inode ratio and the total space available on the partition. With the **-N** option, you specify how many inodes to reserve (not some sort of ratio).

Unlike other options mentioned so far, the output of the **mke2fs** command doesn't mention how many inodes are reserved for system use. We will see a command in a later section that does display this information.

Making an ext3 filesystem

By default both **mkfs** and **mke2fs** create ext2 filesystems. To create an ext3 filesystem with **mke2fs**, use the **-j** option.

4.3 The ext2 and ext3 filesystems

On older distributions, the primary Linux filesystem was ext2. The ext3 filesystem is an improvement over ext2, providing better data integrity, speed and reliability.

Journal modes

The primary difference between ext2 and ext3 filesystems is that ext3 filesystem uses journaling. Journaling is the process of keeping a "log" of changes made to the filesystem. Typically this method allows for quicker and more stable recovery of corrupted filesystem with a possible performance penalty during file operations (adding and deleting files in the filesystem). When you use an ext3 filesystem, you have three choices of journal modes:

ordered: This is the default mode if no mode is specified. This mode journals metadata (attributes of files and directories), but not the contents of the files.

journalled: This mode journals metadata (attributes of files and directories), and the contents of the files. Takes more disk space but can provide quicker database access.

writeback: Doesn't guarantee proper journaling of metadata, but performs quicker than ordered and journalled.

Notes:

Convert an ext2 filesystem to an ext3 filesystem

You can convert an ext2 filesystem to ext3 (even if it is currently mounted) with the **tune2fs** command. The following example will convert the filesystem on the `/dev/hda7` partition to ext3:

```
[root@linux2 /root]# /sbin/tune2fs -j /dev/hda7  
{Output omitted}
```

Remember that your `/etc/fstab` file contains the filesystem type that is to be mounted. You will need to change the filesystem type to “ext3” or else the filesystem won’t be mounted automatically during boot.

Notes:

Convert an ext3 filesystem to an ext2 filesystem

Converting an ext3 filesystem to an ext2 filesystem is a bit more complex. The following commands will convert the ext3 filesystem on the `/dev/hda7` partition to ext2:

```
[root@linux2 /root]# umount /dev/hda7
[root@linux2 /root]# /sbin/tune2fs -O ^has_journal /dev/hda7
{Note: need to modify the /etc/fstab file here}
[root@linux2 /root]# mount /dev/hda7
[root@linux2 /root]# rm -f .journal
```

The `-O` is used to specify an option. The “has_journal” option would specify that the filesystem is using journaling (the backbone of the ext3 filesystem). The caret (“^”) in front of this option it specifies “not”.

You may consider backing up the filesystem prior to converting it to an ext2 filesystem. Don't forget to modify the `/etc/fstab` file!

4.4 Why filesystems break

File corruption

Whenever the filesystem is modified, inode tables and other filesystem "metadata" are changed. To increase the system's speed when performing these actions, these changes are not stored directly on the hard drive; these changes are stored in memory to make the process of filesystem modification quicker.

Every 30 seconds the kernel flushes the memory of these filesystem changes and updates the hard drive. If the system is improperly brought down, the kernel never gets the chance to update the hard drive. This results in filesystem corruption: when filesystem metadata is not correct.

Note: You can inform the kernel that you want to have the filesystem metadata flushed to the hard drive by running the **sync** command. This is rarely needed as unmounting a partition or properly bringing the system down will automatically perform a "sync" on the filesystem(s).

Improper shutdown

The following actions can cause filesystem corruption:

1. Power failure.
2. Turning off the power to the system.
3. Removing a device without unmounting it.

Notes:

4.5 Fixing filesystems with fsck

The **fsck** utility checks for and will fix filesystem inconsistencies. It can be run in two different "modes": non-interactive and interactive.

Non-interactive Mode

The non-interactive mode is typically run only during the boot process. When run in this mode, **fsck** will attempt to fix "simple" problems to the filesystem. Any serious problems **fsck** encounters forces it to abort because fixing such problems would require "human interaction".

Interactive Mode

When **fsck** is run in the interactive mode, the utility lists filesystem errors and a suggested solution. It also prompts the user to answer "yes" to have the utility fix the problem.

Note: On some distributions of Linux, the **fsck** utility executes the **e2fsck** utility. You can also execute the **e2fsck** command directly. Review the man pages for **fsck** and **e2fsck** to see the differences between the options that each utility provides.

Notes:

4.6 fsck examples

Note: If you run **fsck** on an active filesystem, it will almost always return an error. The **fsck** command should always be run on an unmounted filesystem.

Typical interactive example

When you run **fsck** in the interactive mode, it prompts the user to fix problems it encounters. In the following example, the filesystem was properly unmounted and, therefore, contains no problems:

```
[root@linux2 /root]# umount /opt  
[root@linux2 /root]# fsck /opt  
Parallelizing fsck version 1.14 (9-Jan-1999)  
e2fsck 1.14, 9-Jan-1999 for EXT2 FS 0.5b, 95/08/09  
/dev/hda5: clean, 11/1189888 files, 161532/4755208 blocks
```

Responding "yes" to all inquiries

Use the **-y** option to answer "yes" to all of **fsck**'s inquiries:

```
[root@linux2 /root]# fsck -y /opt  
Pass 1: Checking inodes, blocks, and sizes  
Pass 2: Checking directory structure  
Pass 3: Checking directory connectivity  
Pass 4: Checking reference counts  
Pass 5: Checking group summary information  
/dev/hda6: 66167/385560 files (0.5% non-contiguous), 1008404/1542208 blocks
```

Notes:

4.7 Displaying filesystem attributes

The **dumpe2fs** command can be useful to display information about an existing ext2 or ext3 filesystem:

```
[root@linux2 root]# dumpe2fs /dev/hda8
dumpe2fs 1.32 (09-Nov-2002)
Filesystem volume name:   /boot
Last mounted on:         <not available>
Filesystem UUID:         b680f171-9d06-4b08-85d5-d627fe2d79b0
Filesystem magic number: 0xEF53
Filesystem revision #:   1 (dynamic)
Filesystem features:     has_journal filetype_needs_recovery sparse_super
Default mount options:   (none)
Filesystem state:        clean
Errors behavior:         Continue
Filesystem OS type:      Linux
Inode count:             26104
Block count:             104391
Reserved block count:    5219
Free blocks:             91896
Free inodes:             26069
First block:             1
Block size:              1024
Fragment size:          1024
Blocks per group:        8192
Fragments per group:     8192
```

Notes:

```
Inodes per group:      2008
Inode blocks per group: 251
Filesystem created:   Fri Dec 10 11:28:59 2004
Last mount time:      Thu Dec 30 09:18:03 2004
Last write time:      Thu Dec 30 09:18:03 2004
Mount count:          3
Maximum mount count:  -1
Last checked:         Fri Dec 10 11:28:59 2004
Check interval:       0 (<none>)
Reserved blocks uid:  0 (user root)
Reserved blocks gid:  0 (group root)
First inode:          11
Inode size:           128
Journal UUID:         <none>
Journal inode:        8
Journal device:       0x0000
First orphan inode:   0
```

Group 0: (Blocks 1-8192)

```
Primary superblock at 1, Group descriptors at 2-2
Block bitmap at 3 (+2), Inode bitmap at 4 (+3)
Inode table at 5-255 (+4)
0 free blocks, 1988 free inodes, 2 directories
Free blocks:
Free inodes: 21-2008
{remaining output omitted}
```

Notes:

4.8 Modifying filesystem attributes

As demonstrated in an earlier section, the **tune2fs** command can be used to modify some of an ext2 or ext3 filesystem's attributes.

Displaying attributes

To display the attributes of a filesystem, use the **-l** option:

```
[root@linux2 root]# tune2fs -l /dev/hda1
tune2fs 1.32 (09-Nov-2002)
Filesystem volume name:   /boot
Last mounted on:         <not available>
Filesystem UUID:         5adf9ad9-17be-4e40-95ad-2a1174f94e09
Filesystem magic number: 0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:     has_journal filetype needs_recovery sparse_super
Default mount options:   (none)
Filesystem state:        clean
Errors behavior:         Continue
Filesystem OS type:      Linux
Inode count:             26104
Block count:             104391
Reserved block count:    5219
Free blocks:             91886
Free inodes:             26066
First block:             1
```

Notes:

| | |
|--------------------------------|--------------------------|
| <i>Block size:</i> | 1024 |
| <i>Fragment size:</i> | 1024 |
| <i>Blocks per group:</i> | 8192 |
| <i>Fragments per group:</i> | 8192 |
| <i>Inodes per group:</i> | 2008 |
| <i>Inode blocks per group:</i> | 251 |
| <i>Filesystem created:</i> | Tue Oct 5 19:04:37 2004 |
| <i>Last mount time:</i> | Sun Dec 26 21:41:48 2004 |
| <i>Last write time:</i> | Sun Dec 26 21:41:48 2004 |
| <i>Mount count:</i> | 10 |
| <i>Maximum mount count:</i> | -1 |
| <i>Last checked:</i> | Tue Oct 5 19:04:37 2004 |
| <i>Check interval:</i> | 0 (<none>) |
| <i>Reserved blocks uid:</i> | 0 (user root) |
| <i>Reserved blocks gid:</i> | 0 (group root) |
| <i>First inode:</i> | 11 |
| <i>Inode size:</i> | 128 |
| <i>Journal UUID:</i> | <none> |
| <i>Journal inode:</i> | 8 |
| <i>Journal device:</i> | 0x0000 |
| <i>First orphan inode:</i> | 0 |

Notes:

Other common **tune2fs** options include:

| <u>Option</u> | <u>Description</u> |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -J | Override the journal parameters. Most commonly used to specify the size of the journal. See the man page for tune2fs for more details. |
| -m | Change the percent of blocks reserved for the superuser (much like the -m option for the mke2fs command). |
| -o | Used to specify an ext2 or ext3 mount option. For example, you can specify the journal type (ordered, journaled or writeback) with this option. Note: There is also an -O (capital "o") option that is used to specify a filesystem feature. See the man page for tune2fs for more details. |

Notes:

You can also display and modify filesystem features with the **debugfs** utility. This interactive utility is commonly used to directly see filesystem data. For example, the "stat" command can be used to display inode information about a file:

```
[root@linux2 root]# debugfs /dev/hda1
debugfs 1.32 (09-Nov-2002)
debugfs: stat message
Inode: 12  Type: regular  Mode: 0644  Flags: 0x0  Generation: 1562980251
User: 0  Group: 0  Size: 23108
File ACL: 0  Directory ACL: 0
Links: 1  Blockcount: 48
Fragment: Address: 0  Number: 0  Size: 0
ctime: 0x4163b63f -- Wed Oct 6 02:09:19 2004
atime: 0x3f6024f1 -- Thu Sep 11 00:32:01 2003
mtime: 0x3f6024f1 -- Thu Sep 11 00:32:01 2003
BLOCKS:
(0-11):4383-4394, (IND):4395, (12-22):4396-4406
TOTAL: 24
```

See the man page for **debugfs** for more details.

4.9 Summary of commands and files

| Command | Description |
|----------------|---------------------------------------------|
| chattr | Changes file attributes |
| debugfs | Displays filesystem information |
| dumpe2fs | Displays filesystem information |
| fsck | Fixes filesystems |
| ln | Create hard or soft links |
| lsattr | Displays file attributes |
| mke2fs | A flexible command that creates filesystems |
| tune2fs | Modifies filesystems |

| File | Description |
|-------------|--------------------|
| None | None |

4.10 Additional Resources

Books

Running Linux

By Matt Welsh, Matthias Kalle Dalheimer, Terry Dawson, Lar Kaufman

Publisher: O'Reilly

ISBN: 0-596-00272-6

Chapter #6

Web sites

<http://www.tldp.org/HOWTO/Partition/index.html> - Linux Partition HOWTO

<http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/admin-guide/ch-storage.html> - Red Hat Enterprise Linux 4 Introduction to System Administration - Chapter 5: Managing Storage

Man pages

| | |
|----------|---------|
| chattr | ln |
| debugfs | lsattr |
| dumpe2fs | mke2fs |
| fsck | tune2fs |

Notes:

4.11 Certification notes

Review the following charts to determine what sections in this module are relevant for the exam that you are preparing for:

| Topic | RHCT | RHCE | Linux+ | LPI 1-1 | LPI 1-2 | LPI 2-1 | LPI 2-2 |
|--------------------------------------|------|------|--------|---------|---------|---------|---------|
| 4.1 Filesystem details | X | X | X | X | N | N | N |
| 4.2 The mke2fs command | X | X | X | X | N | N | N |
| 4.3 The ext2 and ext3 filesystems | X | X | X | X | N | N | N |
| 4.4 Why filesystems break | X | X | X | X | N | N | N |
| 4.5 Fixing filesystems with fsck | X | X | X | X | N | N | N |
| 4.6 fsck examples | X | X | X | X | N | N | N |
| 4.7 Displaying filesystem attributes | X | X | X | X | N | N | N |
| 4.8 Modifying filesystem attributes | X | X | X | X | N | N | N |

| Key | |
|-----|--------------------------------------------------------------------------------------------------------------------|
| B | Background - May not be on exam itself, but contains information that aids in the understanding of other topics. |
| X | eXam - A topic that is "testable" for this exam |
| N | Not on exam - Indicates that this topic isn't on the exam and isn't needed to understand other topics on the exam. |

4.12 Lab Exercises

Reboot your system to your Lab install.

Scenario #1: You have discovered that the partition you created in Module #2 and #3 labs needs more inodes. In a "real life" situation, you would back up all data in the partition and then recreate the partition. In this lab there isn't any need to back up any data since the `/beta` partition should be empty (except for the swap file). Recreate this filesystem with the following parameters:

- #1. The filesystem should be an **ext3** filesystem.
- #2. Don't reserve any data block for the root user.
- #3. Use 2K block sizes
- #4. Use a byte-to-inode ratio of 4K

Notes:

- Make sure you "clean up" work from the previous lab before proceeding
- Reboot your machine when finished to verify that your partition is automatically mounted during the boot process

IMPORTANT: Reboot your system to your lecture install when you have finished this lab.

For answers to these exercises see the "answers" directory on your floppy disk.

For additional "after class" exercises, see the "exercises" directory on your floppy disk.