

Linux File system

Ahmed Elhoseeny (a_elhoseeny@hotmail.com)

8 Jun 2004

what is a file system:

File system is a place to store files and refer to them in Hierarchical structure.

A file system can be stored on any block device (hard disk, floppy disk, RAID and LVM).

There is a difference between partition and file system where:

partition is a part of the block device (like hard disk) even we can say it is a block device.

Example:

/dev/hdan (where n is the number of the partition) is a block device, but you must know that this partition is not accessible yet before some procedures where creating the file system on the partition is one of them

Types of file system:

there is a wide variety of supported file system types in linux such as:

ext2 – ext3 – ReiserFS – JFS (IBM) – Fat16,32 – NTFS (read-only)

ISO 9660 (CD-ROM)

NFS (network file system for sharing)

SMB (windows share)

each type of those file systems has its own features and here we have an *example*:

lets say we have 2 partitions where the first one is ext3 and the second one is VFAT

now we are going to create directories in each one respectively

```
# mkdir ahmed
```

```
# mkdir Ahmed
```

in the ext3 file system both of the commands will run successfully while in the VFAT only the first command will run successfully but the second one will failed as the file is already exist

Hence we knew that ext3 file system is case sensitive while VFAT is not, and no matter what is the operating system in such case.

File System Structure:

Partition in linux by default is divided into blocks of 1024 Bytes where every block do a specific function according to its type:

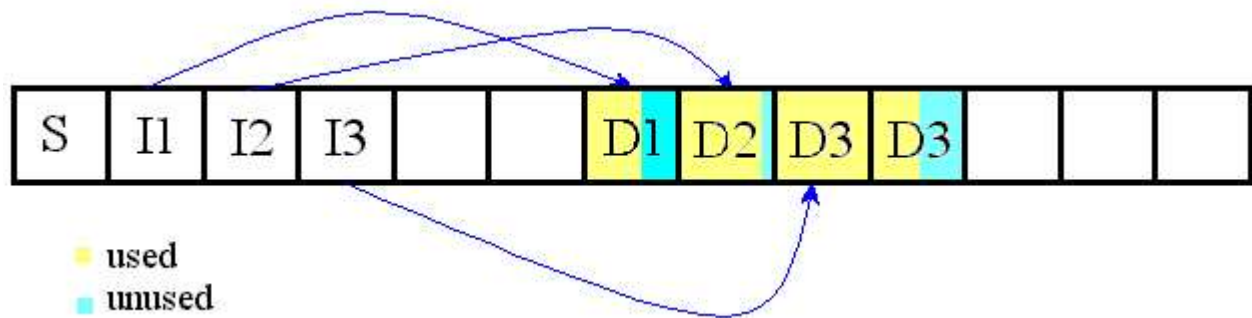
super block – index node block (Inode) – data block

Super block: is the first block of the file system which contains information about the whole file system (FS type – FS size – Block size – last mounted time – pointers to the free i nodes and free blocks) as you can see it is a very important block and if that block get corrupted the file system will never work

so there are multiple copies of that block kept on block 8193, 1683 and so on where if the first block get corrupted, then at the mount process the other blocks will be used.

Inode: each file on the file system has his own Inode (unless we are talking about hard links which is out of topic) where the Inode contains the information about that file (size – permission – creation time – modification time – user&group)

actually the Inode is not a pointer to the file itself but to the data block where that file is located (look at the following figure).



As you can see the first block is the super block then the Inodes as a pointers to the data blocks where:

In1 refer to D1

In2 refer to D2

In3 refer to D3

first of all you must know that there is no data block can hold more than one file and here is the *example*:

1) we have a file (512 Bytes) so here is the status of his data block:

512 Bytes used space

512 Bytes unused space

that block will be pointed by an Inode

2) and we have another file (1536 Bytes) so that file will be located on 2 blocks where the first one will hold 1024 Bytes of the file and the other 512 Bytes will be in another block and here is the status of the first block:

1024 Bytes used space

0 Bytes unused space

that block will be pointed by an Inode

the second block:

512 Bytes used space

512 Bytes unused space

that block will not be pointed by any Inode

so we can say that the Inode is a pointer to the first block of the file

the size of each Inode is 256 Bytes where each Inode block contains 4 Inodes.

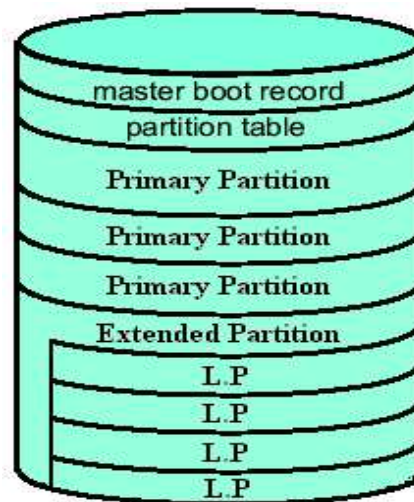
Well, Now we have a question??

when can we say that the file system is full ???

the file system is full if there is no more Inodes or there is no more data blocks, so when you create your file system you have to consider the size of the file which will be located on that file system and here is an *example*:

if you are suspecting small files (< 300 Bytes) then you better make your block size about 512 Bytes where that will save the wasted unused space of the blocks.

Well, now we have to start managing the file systems but before that we have to learn more about partitions and partition table. (look at the following figure).



As we can see the first sector of the hard disk is the MBR (Master Boot Record) where the boot image is stored.

At the end of the MBR (last 64 Bytes) the partition table is stored.

The partition table is a part of the hard disk contains something like a chart for the hard disk and that table has some restrictions that we can not ignore during partitioning our hard disk *Such as:*

- 1) Any partition table can't hold more than 4 Partitions so we can't create more than 4 partitions per hard disk. These partitions may be one of two types (primary/extended).
- 2) The extended partition could be split up into several (logical) partitions.
- 3) The following partition numbers (1,2,3,4) reserved to the (primary/extended) partitions.
- 4) Logical partitions start from number (5).

Partitioning tools:

There are several tools for partitioning the hard disk (FDISK – Parted.... and so on).

But now we will use FDISK to manage our hard disk. At the command prompt run the following command:

`fdisk /dev/hdx` where x is (a, b, c, d) depending on the hard disk location.

```
# fdisk /dev/hda
```

```
The number of cylinders for this disk is set to 2586.
```

```
There is nothing wrong with that, but this is larger than 1024,  
and could in certain setups cause problems with:
```

- 1) software that runs at boot time (e.g., old versions of LILO)
- 2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)

```
Command (m for help):
```

Now we are in the FDISK

Type **m** to list the available commands in FDISK and you will get something like that:

```
Command (m for help): m
Command action
  a  toggle a bootable flag
  b  edit bsd disklabel
  c  toggle the dos compatibility flag
  d  delete a partition
  l  list known partition types
  m  print this menu
  n  add a new partition
  o  create a new empty DOS partition table
  p  print the partition table
  q  quit without saving changes
  s  create a new empty Sun disklabel
  t  change a partition's system id
  u  change display/entry units
  v  verify the partition table
  w  write table to disk and exit
  x  extra functionality (experts only)
Command (m for help):
```

The previous list shows the options that we can use to manage our hard disk and we are going to use some of them now.

First of all we must print the partition table by typing **p** at the prompt.

```
Command (m for help): p

Disk /dev/hda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1  *           1       4462   35840983+   c   Win95 FAT32 (LBA)
/dev/hda2                4463        9729   42307177+   f   Win95 Ext'd (LBA)
/dev/hda5                4463        7012    20482843+   c   Win95 FAT32 (LBA)
/dev/hda6                7013        7650    5124703+   c   Win95 FAT32 (LBA)
/dev/hda7                7651        8288    5124703+   c   Win95 FAT32 (LBA)
/dev/hda8                8289        8301     104391    83   Linux
/dev/hda9                8302        9066    6144831    83   Linux
/dev/hda10               9067        9321    2048256    83   Linux
/dev/hda11               9322        9576    2048256    82   Linux swap
/dev/hda12               9577        9601     200781    83   Linux

Command (m for help):
```

Now lets do some analysis to that output,

The first three lines describe the size of the hard disk (80 GB in our case) and some information about the total number of sectors and so on.

The rest of the output describes the partition table in seven fields as follow:

1. The 1st field is the partition and in our case we have one primary partition and one extended partition splitted up into eight logical partitions.
2. The 2nd field telling us if that partition has a bootable image or not and in our case the first partition is the C partition where it is the active partition and windows is installed on it
3. The 3rd and 4th fields determine the first and the last cylinder in that partition.
4. The 5th field determines how many blocks in that partition.
5. The 6th field lists the ID of that partition where every partition has a specific Hexadecimal Code (ID) and we will learn more about those Ids later.
6. The last field determine which operating system that partition belongs to.

Creating partition:

Type **n** to create new partition

```
Command (m for help): n
Command action
  l   logical (5 or over)
  p   primary partition (1-4)
```

The **n** command prompt you to select the type of the partition you want to create, and you have two choices here, (logical or primary).

So you have to choose which type you want by typing **l** or **p**

Note: in our case we can't create primary partition as the whole space of that hard disk is divided into two partitions (primary /dev/hda1 and extended /dev/hda2) so there is no space for more partitions. But the extended partition has a free space and we can use it to create new logical partition.

So, type **l** to create logical partition as follow:

```
Command (m for help): n
Command action
  l   logical (5 or over)
  p   primary partition (1-4)
l
First cylinder (9602-9729, default 9602):
```

Now you can create the partition using the cylinders or press **Enter** again to get another prompt that allow you to create your partition by M or K as follow:

```
Last cylinder or +size or +sizeM or +sizeK (9602-9729, default 9729): +100M
```

```
Command (m for help):
```

Now we have created logical partition and we can enter the **p** command again to print the partition table after that modification.

```
Command (m for help): p

Disk /dev/hda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot   Start    End  Blocks  Id System
/dev/hda1  *        1   4462  35840983+  c  Win95 FAT32 (LBA)
```

```

/dev/hda2    4463    9729  42307177+  f  Win95 Ext'd (LBA)
/dev/hda5    4463    7012  20482843+  c  Win95 FAT32 (LBA)
/dev/hda6    7013    7650  5124703+  c  Win95 FAT32 (LBA)
/dev/hda7    7651    8288  5124703+  c  Win95 FAT32 (LBA)
/dev/hda8    8289    8301   104391  83  Linux
/dev/hda9    8302    9066  6144831  83  Linux
/dev/hda10   9067    9321  2048256  83  Linux
/dev/hda11   9322    9576  2048256  82  Linux swap
/dev/hda12   9577    9601  200781  83  Linux
/dev/hda13   9602    9614   104391  83  Linux

```

Command (m for help):

Now please take a look at the first output of **p** command and back here again, You will see that the new partition is (/dev/hda13) and by default it has the (83) ID which means that the file system of that partition will be ext3

Well, it is time to learn more about that Hexadecimal ID.

As we said before, each file system has its own ID and we can list all the available IDs by typing **l** at the FDISK prompt.

Once you know the ID of the specified file system you want. type **t** at the FDISK prompt to assign that ID to your partition as follow:

Command (m for help): t

Partition number (1-13):

the prompt will ask you about the number of the partition you want and we will enter 13 (the new partition)

Command (m for help): t

Partition number (1-13): 13

Hex code (type L to list codes):

Now you can enter the Hexadecimal code or you can list them if you want to take another look before assign it, and we will choose RAID where the Hexadecimal is **fd**

Hex code (type L to list codes): fd

Changed system type of partition 13 to fd (Linux raid autodetect)

Enter **p** to print the partition table and look at our partition

```

/dev/hda13    9602    9614   104391  fd  Linux raid autodetect

```

Finally type **w** to save and exit FDISK or type **q** to exit without saving.

Remove a partition:

To remove a partition enter the FDISK utility again and enter **p** to print the partition table and choose the partition you want to delete and we are going now to remove our partition we have just created.

To do that enter **d** at the FDISK prompt as follow:

Command (m for help): d

Partition number (1-13):

the prompt will ask you about the partition number you want to delete and we will enter **13** and again enter **p** to print the partition table, if you didn't find (/dev/hda13) then the partition has been removed successfully.

Note: after any change using FDISK you have to reboot your system.

File System Management:

As we said before, to create a file system we must have a partition, and now we already know how to do that using FDISK and we will consider we already have that partition (/dev/hda13)

Creating a file System:

At the command prompt you will run the following command:

```
# mke2fs /dev/hda13
```

The previous command will create an ext2 file system, so i think it is time to talk about ext2/ext3 file systems.

The ext3 file system is an enhanced version of the ext2 with some improvements as follow.

1. The most important advantage of ext3 file system is the journaling. Where if an unexpected system shutdown occurs because of power failure or something like that, a file system check takes place by a program called **e2fsck** where that program checks all the hard disk and no matter if there is need to do that or not.

As you can see this process wasting a lot of time by delaying the system boot and that will be worst if we have many large hard disks contains a lot of files.

With the journaling feature in the ext3 file system, this check is not necessary anymore unless we have a file system damage because of that unclean shutdown.

2. Speed: also the performance of data integrity and reading/writing on the ext3 is better that ext2 file system.

Now i think it is better to replace the previous command to create ext3 instead of ext2 file system as follow:

```
# mke2fs -j /dev/hda13
```

Note: to change the default block size of that file system use the option -b with the previous command.

```
# mke2fs -j -b 512 /dev/hda13
```

now you have ext3 file system on the partition /dev/hda13 and you asking your self now how can i access that partition

Mounting a File System:

This process is very important where you will be able to access your partition (read, write, and so on) after that process

Well, to do this you need a directory (as a mount point) to the file system. So, run the following command:

```
# mkdir /testfs
```

if you are going to use that directory now you will use it as an ordinary directory and you will be limited with the size of the (/) file system.

then you have to run the mount command with two arguments (file system/mount point):

```
# mount /dev/hda13 /testfs
```

if you are going to use that directory now you will use the partition (/dev/hda3) and you will be limited with its size.

Note: there is some options with the mount command:

```
# mount -t ext3 -o rw /dev/hda13 /testfs
```

-t specify the file system type

-o and option will apply that option to the file system such as (rw – ro)

to learn more about mount see the manual page of mount command.

```
# man mount
```

Note: The mount command has a temporary effect i.e. If you will reboot your system for any reason you will have to remount your file system again.

To make your file system mounted permanently you have to edit the /etc/fstab file, where every time you start your system up, it reads that file and mount the file systems in it.

```
# vi /etc/fstab
```

LABEL=/	/	ext3	defaults	1 1
LABEL=/boot	/boot	ext3	defaults	1 2
none	/dev/pts	devpts	gid=5,mode=620	0 0
LABEL=/elhoseeny	/elhoseeny	ext3	defaults	1 2
none	/proc	proc	defaults	0 0
none	/dev/shm	tmpfs	defaults	0 0
/dev/hda11	swap	swap	defaults	0 0
/dev/cdrom	/mnt/cdrom	udf,iso9660	noauto,owner,kudzu,ro	0 0
/dev/fd0	/mnt/floppy	auto	noauto,owner,kudzu	0 0
/dev/hda1	/mnt/c	vfat	defaults	0 0
/dev/hda5	/mnt/d	vfat	defaults	0 0
/dev/hda6	/mnt/e	vfat	defaults	0 0
/dev/hda7	/mnt/f	vfat	defaults	0 0

now you can add your file system to the fstab file as follow:

/dev/hda13	/testfs	ext3	defaults	0 0
------------	---------	------	----------	-----

after that your file system will be mounted every time your system will start up.

Unmounting File system:

To unmount a file system just make sure you are not using that file system i.e your working directory is not in that file system hierarchy and run the following command:

```
#umount /dev/hda13
```

Now your file system is unmounted and you can not access it and the directory /testfs is an ordinary directory limited with the size of the (/) file system.

Checking a File system:

Some times we need to check our file systems for errors, first of all you must remember that you have to unmount your file system before you start any file system check. And here is the command:

```
# e2fsck -y /dev/hda13
```

where -y will automatically repair minor errors.

Removing a File system:

We discussed how to do that before but here is it again for a mounted file system:

to remove (/dev/hda13) file system do the following:

-umount the file system

```
# umount /dev/hda13
```

-Then use any partitioning tool to remove the /dev/hda13 partition (we did that before using FDISK)

-Remove its entry from /etc/fstab file

-Remove the mount point (/testfs).

Labeling the partition:

Some times we have a lot of file systems and we may lost the ability to recognize them.

So there is a utility called **e2label** which allow us to assign names to our file systems as follow:

```
#e2label /dev/hda13 /testfs
```

The previous command will assign the /testfs label to the /dev/hda13 partition and you can use (/testfs) instead of (/dev/hda13) in most commands especially mount.

Edit the (/etc/fstab) file and replace the line of /dev/hda13 by its label. As follow.

```
# vi /etc/fstab
```

LABEL=/	/	ext3	defaults	1 1
LABEL=/boot	/boot	ext3	defaults	1 2
none	/dev/pts	devpts	gid=5,mode=620	0 0
LABEL=/elhoseeny	/elhoseeny	ext3	defaults	1 2
none	/proc	proc	defaults	0 0
none	/dev/shm	tmpfs	defaults	0 0
/dev/hda11	swap	swap	defaults	0 0
/dev/cdrom	/mnt/cdrom	udf,iso9660	noauto,owner,kudzu,ro	0 0
/dev/fd0	/mnt/floppy	auto	noauto,owner,kudzu	0 0
/dev/hda1	/mnt/c	vfat	defaults	0 0
/dev/hda5	/mnt/d	vfat	defaults	0 0
/dev/hda6	/mnt/e	vfat	defaults	0 0
/dev/hda7	/mnt/f	vfat	defaults	0 0
LABEL=/testfs	/testfs	ext3	defaults	1 0

Now you can mount and unmount /dev/hda13 file system as follow:

```
# mount /testfs
```

```
# umount /testfs
```

References

- ◆ Linux System Administration Student Notebook (IBM Learning Services).
- ◆ Redhat System Administration Guide.

<http://www.redhat.com/docs/manuals/enterprise/RHEL-3-Manual/sysadmin-guide/>