|  |  |  |
| --- | --- | --- |
| Ext2 | Ext3 | Ext4 |
| * Ext2 stands for second extended file system. * This was developed to overcome the limitation of the original ext file system. * Ext2 does not have journaling feature. * On flash drives, usb drives, ext2 is recommended, as it doesn’t need to do the overhead of journaling. * Maximum individual file size can be from 16 GB to 2 TB * Overall ext2 file system size can be from 2 TB to 32 TB | * Ext3 stands for third extended file system. * Starting from Linux Kernel 2.4.15 ext3 was available. * The main benefit of ext3 is that it allows journaling. * Journaling has a dedicated area in the file system, where all the changes are tracked. When the system crashes, the possibility of file system corruption is less because of journaling. * Maximum individual file size can be from 16 GB to 2 TB * Overall ext3 file system size can be from 2 TB to 32 TB * There are three types of journaling available in ext3 file system. * Journal – Metadata and content are saved in the journal. * Ordered – Only metadata is saved in the journal. Metadata are journaled only after writing the content to disk. This is the default. * Writeback – Only metadata is saved in the journal. Metadata might be journaled either before or after the content is written to tlhe disk. * You can convert a ext2 file system to ext3 file system directly (without backup/restore). | * Ext4 stands for fourth extended file system. * Starting from Linux Kernel 2.6.19 ext4 was available. * Supports huge individual file size and overall file system size. * Maximum individual file size can be from 16 GB to 16 TB * Overall maximum ext4 file system size is 1 EB (exabyte). 1 EB = 1024 PB (petabyte). 1 PB = 1024 TB (terabyte). * Directory can contain a maximum of 64,000 subdirectories (as opposed to 32,000 in ext3) * You can also mount an existing ext3 fs as ext4 fs (without having to upgrade it). * Several other new features are introduced in ext4: multiblock allocation, delayed allocation, journal checksum. fast fsck, etc. All you need to know is that these new features have improved the performance and reliability of the filesystem when compared to ext3. * In ext4, you also have the option of turning the journaling feature “off”. |

**Differences between ext2, ext3 and ext4 file system types:**

**What is Journaling in Linux:-**

A journaling filesystem keeps a journal or log of the changes that are being made to the filesystem during disk writing that can be used to rapidly reconstruct corruptions that may occur due to events such a system crash or power outage.

**Creating an ext2, or ext3, or ext4 filesystems:-**

Once you’ve partitioned your hard disk using fdisk command, use mke2fs to create either ext2, ext3, or ext4 file system.

* 1. **Create an ext2 file system:**

mke2fs /dev/sda1

* 1. **Create an ext3 file system:**

mkfs.ext3 /dev/sda1

**(or)**

mke2fs –j /dev/sda1

* 1. **Create an ext4 file system:**

mkfs.ext4 /dev/sda1

(or)

mke2fs -t ext4 /dev/sda1

**Converting ext2 to ext3**

For example, if you are upgrading /dev/sda2 that is mounted as /home, from ext2 to ext3, do the following.

* umount /dev/sda2
* tune2fs -j /dev/sda2
* mount /dev/sda2 /home

Note: You really don’t need to umount and mount it, as ext2 to ext3 conversion can happen on a live file system. But, it is better doing the conversion offline.

**Converting ext3 to ext4**

If you are upgrading /dev/datavg/testlv01 that is mounted as /myfs03, from ext3 to ext4, do the following.

umount /myfs03

* e2fsck -f /dev/datavg/testlv01
* tune2fs -O extent,flex\_bg,huge\_file,dir\_nlink,extra\_isize,metadata\_csum /dev/datavg/testlv01
* mount /dev/datavg/testlv01 /myfs03
* df -hT

**How to set ulimit valueS**

**Settings in /etc/security/limits.conf take the following form:**

**# vi /etc/security/limits.conf**

**#<domain> <type> <item> <value>**

**\* - core <value>**

**\* - data <value>**

**\* - priority <value>**

**\* - fsize <value>**

**\* soft sigpending <value> eg:57344**

**\* hard sigpending <value> eg:57444**

**\* - memlock <value>**

**\* - nofile <value> eg:1024**

**\* - msgqueue <value> eg:819200**

**\* - locks <value>**

**\* soft core <value>**

**\* hard nofile <value>**

**@<group> hard nproc <value>**

**<user> soft nproc <value>**

**%<group> hard nproc <value>**

**<user> hard nproc <value>**

**@<group> - maxlogins <value>**

**<user> hard cpu <value>**

**<user> soft cpu <value>**

**<user> hard locks <value>**

**Where**

**<domain> can be:**

* an user name
* a group name, with @group syntax
* the wildcard \*, for default entry
* the wildcard %, can be also used with %group syntax, for maxlogin limit

**<type> can have the two values:**

* "soft" for enforcing the soft limits
* "hard" for enforcing hard limits

**<item> can be one of the following:**

* core - limits the core file size (KB)
* data - max data size (KB)
* fsize - maximum filesize (KB)
* memlock - max locked-in-memory address space (KB)
* nofile - max number of open files
* rss - max resident set size (KB)
* stack - max stack size (KB)
* cpu - max CPU time (MIN)
* nproc - max number of processes
* as - address space limit (KB)
* maxlogins - max number of logins for this user
* maxsyslogins - max number of logins on the system
* priority - the priority to run user process with
* locks - max number of file locks the user can hold
* sigpending - max number of pending signals
* msgqueue - max memory used by POSIX message queues (bytes)
* nice - max nice priority allowed to raise to values: [-20, 19]
* rtprio - max realtime priority

**Exit and re-login from the terminal for the change to take effect.**

**fuser command:-**

The fuser command is used to find which process is using a file, a directory or a socket. It also gives information about the user owning the process and the type of access. The fuser tool displays the process id(PID) of every process using the specified files or file systems.

Examples:-

1.To Identify which processes are using a particular file or directory.

**$ fuser .**

We see that the output consists of process IDs of the processes using fuser but all the PIDs are followed by a character ‘c’. This indicates the type of access. The type of access can be any one of the following:

c current directory

e executable being run

f open file. f is omitted in default display mode

F open file for writing. F is omitted in default display mode

r root directory

m mmap’ed file or shared library

So ‘c’ in the output would mean that these processes are using this directory as their current directory.

Use Option -v to display detailed information in the output

2.$ fuser -v ./

3.fuser -v -n tcp 5000

4.fuser -v -k socket\_serv

5.Interactively Kill Processes using fuser

$ fuser -v -k -i socket\_serv

6.$ fuser -v -k -i ./

7.fuser -k 123/tcp

**fsck**

Linux fsck utility is used to check and repair Linux filesystems (ext2, ext3, ext4, etc.). Depending on when was the last time a file system was checked, the system runs the fsck during boot time to check whether the filesystem is in consistent state. System administrator could also run it manually when there is a problem with the filesystems. Make sure to execute the fsck on an unmounted file systems to avoid any data corruption issues.

Examples:-

fsck -t ext4 -y

e2fsck /dev/vg/lvname

fsck /dev/sda3 –n

Below are phases of fsck command:-

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