To install the XFS packages on a system:

1. Log in to ULN, and subscribe your system to the ol6\_x86\_64\_latest channel.
2. On your system, use **yum** to install the xfsprogs and xfsdump packages:

# **yum install xfsprogs xfsdump**

1. If required, use **yum** to install the XFS development and QA packages:

# **yum install xfsprogs-devel xfsprogs-qa-devel**

Creating an XFS File System

You can use the **mkfs.xfs** command to create an XFS file system, for example.

# **mkfs.xfs /dev/vg0/lv0**

meta-data=/dev/vg0/lv0 isize=256 agcount=32, agsize=8473312 blks

= sectsz=512 attr=2, projid32bit=0

data = bsize=4096 blocks=271145984, imaxpct=25

= sunit=0 swidth=0 blks

naming =version 2 bsize=4096 ascii-ci=0

log =internal log bsize=4096 blocks=32768, version=2

= sectsz=512 sunit=0 blks, lazy-count=1

realtime =none extsz=4096 blocks=0, rtextents=0

To create an XFS file system with a stripe-unit size of 32 KB and 6 units per stripe, you would specify the **su** and **sw** arguments to the **-d** option, for example:

# **mkfs.xfs -d su=32k,sw=6 /dev/vg0/lv1**

For more information, see the mkfs.xfs(8) manual page.

Modifying an XFS File System

**Note**

You cannot modify a mounted XFS file system.

You can use the **xfs\_admin** command to modify an unmounted XFS file system. For example, you can enable or disable lazy counters, change the file system UUID, or change the file system label.

To display the existing label for an unmounted XFS file system and then apply a new label:

# **xfs\_admin -l /dev/sdb**

label = ""

# **xfs\_admin -L "VideoRecords" /dev/sdb**

writing all SBs

new label = "VideoRecords"

**Note**

The label can be a maximum of 12 characters in length.

To display the existing UUID and then generate a new UUID:

# **xfs\_admin -u /dev/sdb**

UUID = cd4f1cc4-15d8-45f7-afa4-2ae87d1db2ed

# **xfs\_admin -U generate /dev/sdb**

writing all SBs

new UUID = c1b9d5a2-f162-11cf-9ece-0020afc76f16

To clear the UUID altogether:

# **xfs\_admin -U nil /dev/sdb**

Clearing log and setting UUID

writing all SBs

new UUID = 00000000-0000-0000-0000-000000000000

To disable and then re-enable lazy counters:

# **xfs\_admin -c 0 /dev/sdb**

Disabling lazy-counters

# **xfs\_admin -c 1 /dev/sdb**

Enabling lazy-counters

For more information, see the mkfs\_admin(8) manual page.

Freezing and Unfreezing an XFS File System

If you need to take a hardware-based snapshot of an XFS file system, you can temporarily stop write operations to it.

**Note**

You do not need to explicitly suspend write operations if you use the **lvcreate** command to take an LVM snapshot.

To freeze and unfreeze an XFS file system, use the **-f** and **-u** options with the **xfs\_freeze** command, for example:

# **xfs\_freeze -f /myxfs**

# # ... Take snapshot of file system ...

# **xfs\_freeze -u /myxfs**

**Note**

You can also use the **xfs\_freeze** command with btrfs, ext3, and ext4 file systems.

For more information, see the xfs\_freeze(8) manual page.

Setting Quotas on an XFS File System

[6.7.1 Setting Project Quotas](https://docs.oracle.com/cd/E37670_01/E37355/html/ol_prjquo_xfs.html)

The following table shows the **mount** options that you can specify to enable quotas on an XFS file system:

| **Mount Option** | **Description** |
| --- | --- |
| **gqnoenforce** | Enable group quotas. Report usage, but do not enforce usage limits. |
| **gquota** | Enable group quotas and enforce usage limits. |
| **pqnoenforce** | Enable project quotas. Report usage, but do not enforce usage limits. |
| **pquota** | Enable project quotas and enforce usage limits. |
| **uqnoenforce** | Enable user quotas. Report usage, but do not enforce usage limits. |
| **uquota** | Enable user quotas and enforce usage limits. |

To show the block usage limits and the current usage in the myxfs file system for all users, use the **xfs\_quota** command:

# **xfs\_quota -x -c 'report -h' /myxfs**

User quota on /myxfs (/dev/vg0/lv0)

Blocks

User ID Used Soft Hard Warn/Grace

---------- ---------------------------------

root 0 0 0 00 [------]

guest 0 200M 250M 00 [------]

The following forms of the command display the free and used counts for blocks and inodes respectively in the manner of the **df -h** command:

# **xfs\_quota -c 'df -h' /myxfs**

Filesystem Size Used Avail Use% Pathname

/dev/vg0/lv0 200.0G 32.2M 20.0G 1% /myxfs

# **xfs\_quota -c 'df -ih' /myxfs**

Filesystem Inodes Used Free Use% Pathname

/dev/vg0/lv0 21.0m 4 21.0m 1% /myxfs

If you specify the **-x** option to enter expert mode, you can use subcommands such as **limit** to set soft and hard limits for block and inode usage by an individual user, for example:

# **xfs\_quota -x -c 'limit bsoft=200m bhard=250m isoft=200 ihard=250 guest' /myxfs**

Of course, this command requires that you mounted the file system with user quotas enabled.

To set limits for a group on an XFS file system that you have mounted with group quotas enabled, specify the **-g** option to **limit**, for example:

# **xfs\_quota -x -c 'limit -g bsoft=5g bhard=6g devgrp' /myxfs**

Backing up and Restoring XFS File Systems

The xfsdump package contains the **xfsdump** and **xfsrestore** utilities. **xfsdump** examines the files in an XFS file system, determines which files need to be backed up, and copies them to the storage medium. Any backups that you create using **xfsdump** are portable between systems with different endian architectures. **xfsrestore** restores a full or incremental backup of an XFS file system. You can also restore individual files and directory hierarchies from backups.

**Note**

Unlike an LVM snapshot, which immediately creates a sparse clone of a volume, **xfsdump** takes time to make a copy of the file system data.

You can use the **xfsdump** command to create a backup of an XFS file system on a device such as a tape drive, or in a backup file on a different file system. A backup can span multiple physical media that are written on the same device, and you can write multiple backups to the same medium. You can write only a single backup to a file. The command does not overwrite existing XFS backups that it finds on physical media. You must use the appropriate command to erase a physical medium if you need to overwrite any existing backups.

For example, the following command writes a level 0 (base) backup of the XFS file system, /myxfs to the device /dev/st0 and assigns a session label to the backup:

# **xfsdump -l 0 -L "Backup level 0 of /myxfs `date`" -f /dev/st0 /myxfs**

You can make incremental dumps relative to an existing backup by using the command:

# **xfsdump -l *level* -L "Backup level *level* of /myxfs `date`" -f /dev/st0 /myxfs**

A level 1 backup records only file system changes since the level 0 backup, a level 2 backup records only the changes since the latest level 1 backup, and so on up to level 9.

If you interrupt a backup by typing Ctrl-C and you did not specify the **-J** option (suppress the dump inventory) to **xfsdump** , you can resume the dump at a later date by specifying the **-R** option:

# **xfsdump -R -l 1 -L "Backup level 1 of /myxfs `date`" -f /dev/st0 /myxfs**

In this example, the backup session label from the earlier, interrupted session is overridden.

You use the **xfsrestore** command to find out information about the backups you have made of an XFS file system or to restore data from a backup.

The **xfsrestore -I** command displays information about the available backups, including the session ID and session label. If you want to restore a specific backup session from a backup medium, you can specify either the session ID or the session label.

For example, to restore an XFS file system from a level 0 backup by specifying the session ID:

# **xfsrestore -f /dev/st0 -S c76b3156-c37c-5b6e-7564-a0963ff8ca8f /myxfs**

If you specify the **-r** option, you can cumulatively recover all data from a level 0 backup and the higher-level backups that are based on that backup:

# **xfsrestore -r -f /dev/st0 -v silent /myxfs**

The command searches the archive looking for backups based on the level 0 backup, and prompts you to choose whether you want to restore each backup in turn. After restoring the backup that you select, the command exits. You must run this command multiple times, first selecting to restore the level 0 backup, and then subsequent higher-level backups up to and including the most recent one that you require to restore the file system data.

**Note**

After completing a cumulative restoration of an XFS file system, you should delete the housekeeping directory that **xfsrestore** creates in the destination directory.

You can recover a selected file or subdirectory contents from the backup medium, as shown in the following example, which recovers the contents of /myxfs/profile/examples to /tmp/profile/examples from the backup with a specified session label:

# **xfsrestore -f /dev/sr0 -L "Backup level 0 of /myxfs Sat Mar 2 14:47:59 GMT 2013"** \

**-s profile/examples /usr/tmp**

Alternatively, you can interactively browse a backup by specifying the **-i** option:

# **xfsrestore -f /dev/sr0 -i**

This form of the command allows you browse a backup as though it were a file system. You can change directories, list files, add files, delete files, or extract files from a backup.

To copy the entire contents of one XFS file system to another, you can combine **xfsdump** and **xfsrestore**, using the **-J** option to suppress the usual dump inventory housekeeping that the commands perform:

# **xfsdump -J - /myxfs | xfsrestore -J - /myxfsclone**

For more information, see the xfsdump(8) and xfsrestore(8) manual pages.

Defragmenting an XFS File System

You can use the **xfs\_fsr** command to defragment whole XFS file systems or individual files within an XFS file system. As XFS is an extent-based file system, it is usually unnecessary to defragment a whole file system, and doing so is not recommended.

To defragment an individual file, specify the name of the file as the argument to **xfs\_fsr**.

# **xfs\_fsr *pathname***

If you run the **xfs\_fsr** command without any options, the command defragments all currently mounted, writeable XFS file systems that are listed in /etc/mtab. For a period of two hours, the command passes over each file system in turn, attempting to defragment the top ten percent of files that have the greatest number of extents. After two hours, the command records its progress in the file /var/tmp/.fsrlast\_xfs, and it resumes from that point if you run the command again.

For more information, see the xfs\_fsr(8) manual page.

Checking and Repairing an XFS File System

**Note**

If you have an Oracle Linux Premier Support account and encounter a problem mounting an XFS file system, send a copy of the /var/log/messages file to Oracle Support and wait for advice.

If you cannot mount an XFS file system, you can use the **xfs\_check** command to check its consistency. Usually, you would only run this command on the device file of an unmounted file system that you believe has a problem. If **xfs\_check** displays any output when you do not run it in verbose mode, the file system has an inconsistency.

# **xfscheck *device***

If you can mount the file system and you do not have a suitable backup, you can use **xfsdump** to attempt to back up the existing file system data, However, the command might fail if the file system's metadata has become too corrupted.

You can use the **xfs\_repair** command to attempt to repair an XFS file system specified by its device file. The command replays the journal log to fix any inconsistencies that might have resulted from the file system not being cleanly unmounted. Unless the file system has an inconsistency, it is usually not necessary to use the command, as the journal is replayed every time that you mount an XFS file system.

# **xfs\_repair *device***

If the journal log has become corrupted, you can reset the log by specifying the **-L** option to **xfs\_repair**.

**Warning**

Resetting the log can leave the file system in an inconsistent state, resulting in data loss and data corruption. Unless you are experienced in debugging and repairing XFS file systems using **xfs\_db**, it is recommended that you instead recreate the file system and restore its contents from a backup.

If you cannot mount the file system or you do not have a suitable backup, running **xfs\_repair** is the only viable option unless you are experienced in using **xfs\_db**.

**xfs\_db** provides an internal command set that allows you to debug and repair an XFS file system manually. The commands allow you to perform scans on the file system, and to navigate and display its data structures. If you specify the **-x** option to enable expert mode, you can modify the data structures.

# **xfs\_db [-x] *device***

For more information, see the xfs\_check(8), xfs\_db(8) and xfs\_repair(8) manual pages, and the **help** command within **xfs\_db**