File System and Storage Management

Foreword

- This course describes the basic concepts of file systems, drive storage, and logical volume storage, how to manage and use the file system and storage, and some common operation commands.

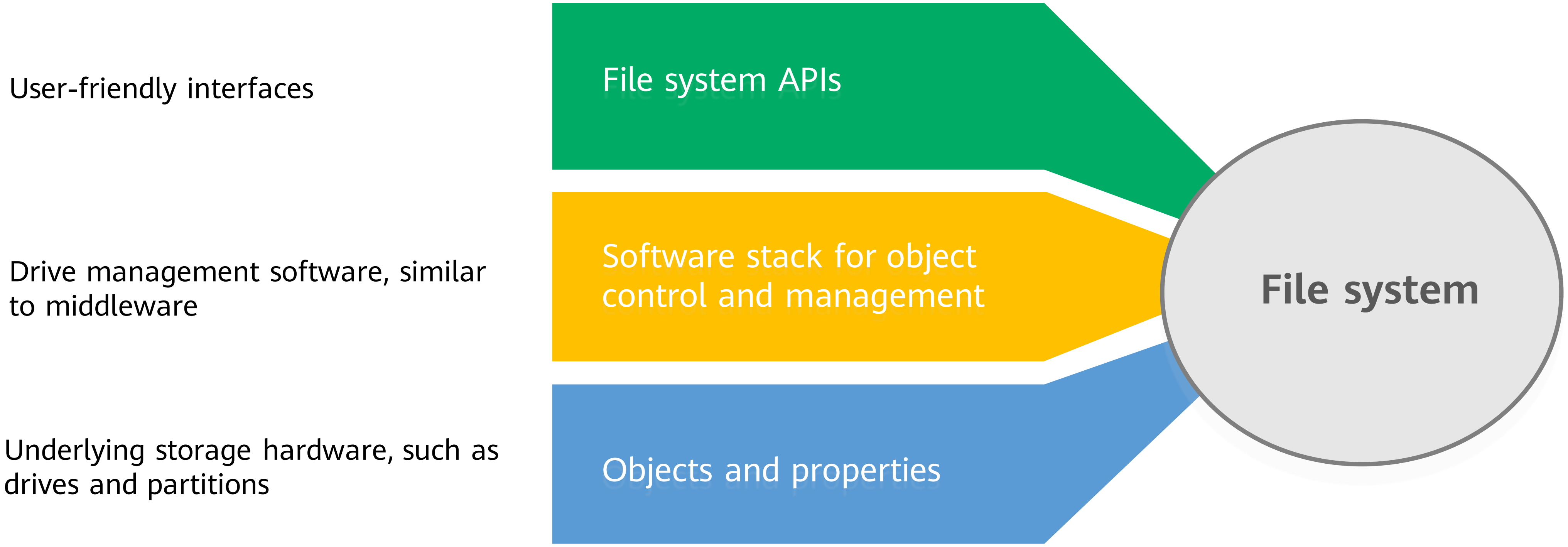
Objectives

- Upon completion of this course, you will understand:
  - Basic concepts of file systems and storage
  - How to mount and use drive storage
  - Methods of managing logical volumes

Contents

File System Overview

- A file system is a method and a data structure used by an operating system (OS) to identify files on a storage device or a partition, that is, a method of organizing files on a storage device. In an OS, a software structure that manages and stores file data is referred to as a file management system, or file system for short.


User-friendly interfaces
File system APIs
Drive management software, similar to middleware
Software stack for object control and management
Underlying storage hardware, such as drives and partitions
Objects and properties
File system

File System Types and Application Scenarios

Common File System	Application Scenario
FAT	FAT, including the FAT16 and FAT32 variants, is used by Windows 9X OSs.
NTFS	NTFS is a security-based file system. It is a unique file system structure used by Windows NT. Windows 2000 adopts the updated NTFS 5.0.
NFS	Network file system (NFS) is used for file sharing between UNIX systems over the network.
RAW	RAW file system indicates that the drive is not processed or formatted.
ext	As the standard file system on GNU/Linux, extended file system (ext) features excellent file access performance and is more effective against small- and medium-sized files. ext variants include ext2, ext3, and ext4.
XFS	A high-performance log file system developed for the IRIX OS by Silicon Graphics in 1993. Later ported to the Linux kernel, it excels in large-file processing and provides smooth data transfer.

Contents

File Systems on openEuler

- The openEuler kernel is derived from Linux. The Linux kernel supports more than 10 types of file systems, such as Btrfs, JFS, ReiserFS, ext, ext2, ext3, ext4, ISO 9660, XFS, Minix, MSDOS, UMSDOS, VFAT, NTFS, HPFS, NFS, SMB, SysV and PROC. The following table describes the common file systems.
- The default file system on openEuler is ext4.

Common File System	Description
ext	File system specially designed for Linux. The latest version is ext4.
XFS	A high-performance log file system developed for the IRIX OS by Silicon Graphics in 1993. Later ported to the Linux kernel, it excels in large-file processing and provides smooth data transfer.
VFAT	On Linux, VFAT is the name of the FAT (including FAT16 and FAT32) file systems of DOS and Windows.
NFS	Network file system (NFS) is used for file sharing between UNIX systems over the network.
ISO 9600	The standard file system for optical disc media. Linux supports this file system, allowing the system to read optical discs and ISO image files, and burn optical discs.

Contents

Swap Space

- The swap space on Linux is a partition or a file on the drive. When the physical memory is insufficient, the resources that are not frequently accessed in the memory are saved to the preset swap space so that the memory occupied by the resources is released. In this way, the system has more memory to serve each process. When the content stored in the swap space is needed, the system loads the data in the swap space to the memory.
- The sum of the physical memory and swap space size is the total virtual memory provided by the system.
- Why is the swap space required?
  - More system memory: When the physical memory is insufficient, increasing the swap space size is more cost-effective than adding physical memory.
  - Overall system performance improvement: By moving infrequently used data to the swap space, the system has more memory for caching, accelerating the system I/O.
  - Enabling Linux hibernation: In many Linux distributions (such as Ubuntu), when the system hibernates, memory data is saved to the swap space. The saved data will be loaded to the memory upon next startup.

Swap Space Configuration

- Linux has two forms of swap space: the swap partition and swap file. The swap partition is an independent drive that has no other file or content. The swap file is a special file in the file system and is independent from the system and data files.
  - Creating a swap partition: Run **fdisk** to create a partition, **mkswap** to create a swap partition, and **swapon** to enable the swap partition.
  - Creating a swap file: Create a file, run **mkswap** to format the file, and then run **swapon** to enable the swap file.
- Recommended swap sizes

RAM Size	Recommended Swap Size
≤ 2 GB	2x RAM size
2 GB to 8 GB	Same as RAM size
> 8 GB	8 GB

Contents

Drive Types

Storage media

Hard disk drive (HDD)

Solid state drive (SSD)

Interface types

IDE

SCSI

SATA

SAS

FC

Drive Interface Description

Drive Interface	Description
Integrated Drive Electronics (IDE)	The earliest general standard for HDDs. All electronic integrated drives, including SCSI, adopt the IDE standard.
Serial ATA (SATA)	SATA is distinct from IDE, which is also known as Parallel ATA (PATA). Therefore, the IDE interface is often called the parallel port and the SATA interface is called the serial port.
Small Computer System Interface (SCSI)	Drives that adopt the SCSI interface are SCSI drives. Serial Attached SCSI (SAS) is a serial implementation of the SCSI interface. SCSI drives are usually used on servers. Compared with PATA and SATA drives, SCSI drives provide higher performance, stronger stability, and support hot swap. However, SCSI drives have disadvantages such as small capacity, loud noise, and high cost.
Fibre Channel (FC)	FC interfaces enable drives to connect directly using optical fibers, improving the I/O performance of high-throughput performance-intensive systems.

Viewing Drive Information on Linux (1)

- **fdisk –l** is used to view information about all drives in the system, including mounted and unmounted drives.

```
[root@openEuler ~]# fdisk -l
Disk /dev/vda: 40 GiB, 42949672960 bytes, 83886080 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: AA321D82-C833-4D3E-885C-52FC0ADF3860

Device       Start      End  Sectors  Size Type
/dev/vda1     2048   411647   409600  200M EFI System
/dev/vda2   411648  2508799  2097152    1G Linux filesystem
/dev/vda3  2508800 83884031 81375232 38.8G Linux LVM

Disk /dev/mapper/openeuler-root: 34.82 GiB, 37367054336 bytes, 72982528 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mapper/openeuler-swap: 4 GiB, 4294967296 bytes, 8388608 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Viewing Drive Information on Linux (2)

- **df –h** is used to check the mounting status, spaces, and usage of the drives.

```
[root@openEuler ~]# df -h
Filesystem                   Size  Used  Avail  Use% Mounted on
devtmpfs                     1.2G     0   1.2G    0% /dev
tmpfs                        1.5G     0   1.5G    0% /dev/shm
tmpfs                        1.5G   18M   1.5G    2% /run
tmpfs                        1.5G     0   1.5G    0% /sys/fs/cgroup
/dev/mapper/openeuler-root    35G  4.4G    28G   14% /
tmpfs                        1.5G   64K   1.5G    1% /tmp
/dev/vda2                    976M  125M   785M   14% /boot
/dev/vda1                    200M  5.8M   195M    3% /boot/efi
tmpfs                        298M     0   298M    0% /run/user/0
tmpfs                        298M     0   298M    0% /run/user/993
```

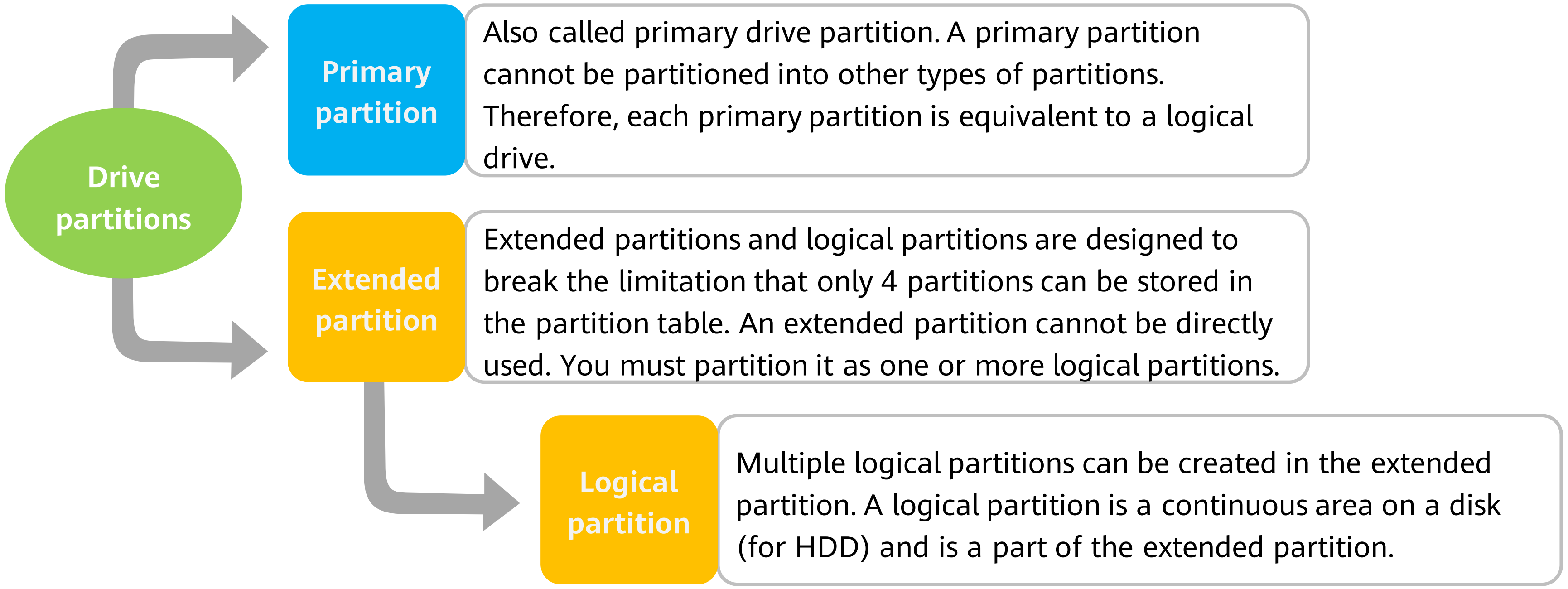
Contents

Drive Partitioning

- Through drive partitioning, a drive is divided into multiple logical storage units called partitions. The system administrator can use different partitions for different functions.
- Advantages of drive partitioning:
  - The available space of applications or users can be restricted.
  - The machine can boot into multiple OSs from different partitions on the same drive.
  - OS files are separated from program and user files.
  - A separate area can be created for OS virtual memory swapping.
  - Drive space usage can be restricted to improve the performance of diagnosis tools and image backups.

Drive Partition Types

- By partitioning the drive, you are modifying the drive partition table. Pay attention to the following while partitioning:
  - For the continuity of data, you are advised to place the extended partition in the last cylinders.
  - A drive has only one extended partition. Except the primary partition space, the rest space is allocated to the extended partition.
  - Drive capacity = primary partition capacity + extended partition capacity; Extended partition capacity= Sum of the capacities of all logical partitions.



Primary partition: Also called primary drive partition. A primary partition cannot be partitioned into other types of partitions. Therefore, each primary partition is equivalent to a logical drive.

Extended partition: Extended partitions and logical partitions are designed to break the limitation that only 4 partitions can be stored in the partition table. An extended partition cannot be directly used. You must partition it as one or more logical partitions.

Logical partition: Multiple logical partitions can be created in the extended partition. A logical partition is a continuous area on a disk (for HDD) and is a part of the extended partition.

Drive Partition Naming

- There are no drive letters in Linux. You can access a device through the device name. The device names are stored in the **/dev** directory.
- The devices are named as follows:

/dev/xxyN

- *xx* indicates the device type, which can be **hd** (IDE drive), **sd** (SCSI drive), **fd** (floppy disk drive), or **vd** (virtio drive).
- y indicates the device where the partition is located. For example, **/dev/hda** indicates the first IDE drive, and **/dev/sdb** indicates the second SCSI drive.
- *N* indicates the partition number. The first four partitions (primary partitions or extended partition) are numbered from 1 to 4. The logical partitions start from 5. For example, **/dev/hda3** is the third primary partition or extended partition on the first IDE drive, and **/dev/sdb6** is the second logical partition on the second SCSI drive.

- Note: On Linux, SAS drives, SATA drives, and SSDs are identified by **sd**, and IDE drives are identified by **hd**.

Drive Partitioning Scheme - MBR

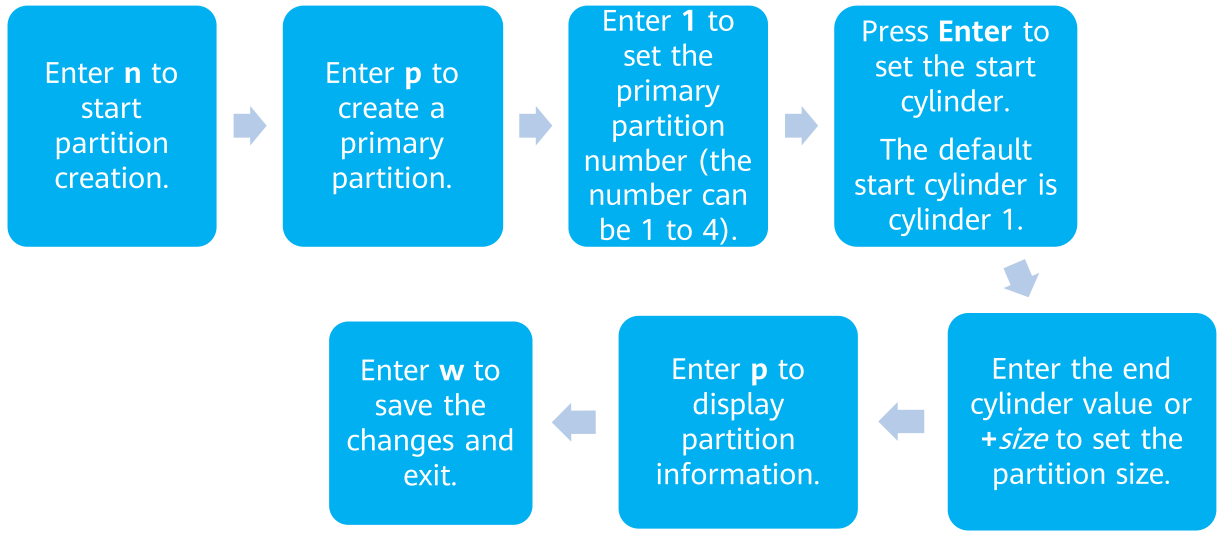
- Master Boot Record (MBR)
  - The MBR partitioning scheme specifies how the drive of a system running BIOS is partitioned. The MBR is a special boot sector at the beginning of the drive.
  - A SCSI drive partitioned using the MBR scheme can have a maximum of 15 partitions, among which a maximum of 4 primary partitions or 12 logical partitions can be created. Extended partitions cannot be directly used and therefore are not counted. An IDE drive partitioned using the MBR scheme can have a maximum of 63 partitions, among which a maximum of 4 primary partitions or 60 logical partitions can be created. Extended partitions cannot be directly used and therefore are not counted.
  - Because partition size data is stored as 32-bit values, the maximum size of a drive or partition cannot exceed 2 TB when the MBR scheme is used.

fdisk Drive Partitioning Utility

- fdisk is a commonly used traditional Linux drive partitioning utility. fdisk does not support partitions larger than 2 TB.
- The command syntax is as follows:
  - **fdisk [options]** *parameter*
  - Common options are as follows:
    - **-b <partition size>**: specifies the size of each partition.
    - **-l**: lists the partition tables of the specified peripheral devices.
    - **-s <partition number >**: prints the specified partition size in sectors to the standard output.
    - **-u**: used with **-l** to display the start address of each partition in the unit of sectors instead of cylinders.
    - **-v**: displays version information.

fdisk Interactive Mode

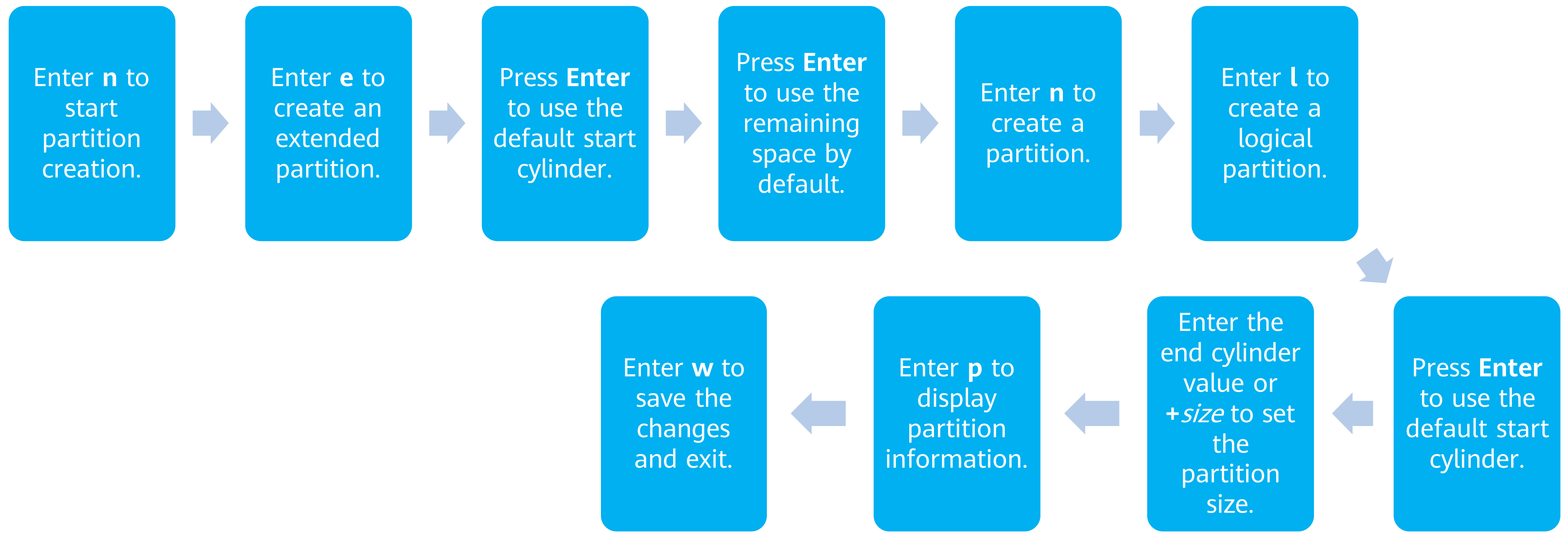
- Select a drive to enter the interactive mode for drive partitioning.
- For example: **fdisk /dev/sdb**. The following table lists the interactive mode commands.
- The process of creating a primary partition is as follows:


Enter **n** to start partition creation.
Enter **p** to create a primary partition.
Enter **1** to set the primary partition number (the number can be 1 to 4).
Press **Enter** to set the start cylinder.
The default start cylinder is cylinder 1.
Enter the end cylinder value or +*size* to set the partition size.
Enter **p** to display partition information.
Enter **w** to save the changes and exit.

Command	Description
a	Set a bootable flag.
b	Edit the BSD disklabel.
c	Set the DOS compatibility flag.
d	Delete a partition.
l	Lists known file system types. 82 indicates a Linux swap partition, and 83 indicates a Linux partition.
m	Displays help menu.
n	Create a partition.
o	Create an empty DOS partition table.
p	List the partitions.
q	Quit without saving changes.
s	Creating an empty SUN partition table.
t	Change the system ID of a partition.
u	Change the unit of the displayed records.
v	Verify the partition table.
w	Save and exit.

fdisk Interactive Mode

- The process of creating an extended partition is as follows:


Enter **n** to start partition creation.
Enter **e** to create an extended partition.
Press **Enter** to use the default start cylinder.
Press **Enter** to use the remaining space by default.
Enter **n** to create a logical partition.
Enter **l** to create a logical partition.
Press **Enter** to use the default start cylinder.
Enter the end cylinder value or **+***size* to set the partition size.
Enter **p** to display partition information.
Enter **w** to save the changes and exit.

- Note: The extended partition cannot be directly used after creation. You must create a logical partition.

Drive Partitioning Scheme - GPT

- GUID Partition Table (GPT)
  - As drives continue to increase in capacity, the 2 TB drive and partition size restriction of the old MBR partitioning is no longer a theoretical limit, but an actual problem that is frequently encountered in the production environment. As a result, GPT is replacing the traditional MBR solution for drive partitioning.
  - GPT assigns a globally unique identifier (GUID) to each partition on a drive. For systems that run Unified Extensible Firmware Interface (UEFI) firmware, GPT is the standard for arranging partition tables on physical drives.
  - There are no primary or logical partitions with GPT, and each drive can have a maximum of 128 partitions. As GPT uses 64 bits for logical block addresses, a maximum partition size of 18 EB is supported.

parted Drive Partitioning Utility

- parted is a partitioning utility commonly used in Linux to create drive partitions larger than 2 TB. Compared with fdisk, parted is more convenient and enables the partition size to be dynamically adjusted. The command syntax is as follows:
- **parted [options] [*device* [*command* [options...]...]]**
  - The command options are as follows:
    - **-h**: displays help information.
    - **-i**: enters the interactive mode.
    - **-s**: enters the script mode.
    - **-v**: displays parted version information.
    - *device*: indicates the drive device name, for example, **/dev/sda**.
    - *command*: indicates a parted command. If no command is specified, parted enters the interactive mode.

parted Interactive Mode

- Select a drive to enter the interactive mode for drive partitioning.
- For example: **parted /dev/sdb**. The following table lists the interactive mode commands.

Enter **mklabel gpt**

- Create a GPT partition table. GPT is the only option for a partition size greater than 2 TB. After confirming other information, run the **print** command to check whether the partition table is in GPT format.

Enter **mkpart**

- Create a partition. If **mkpart** is used without any parameter, parted prompts you to enter related information step by step.

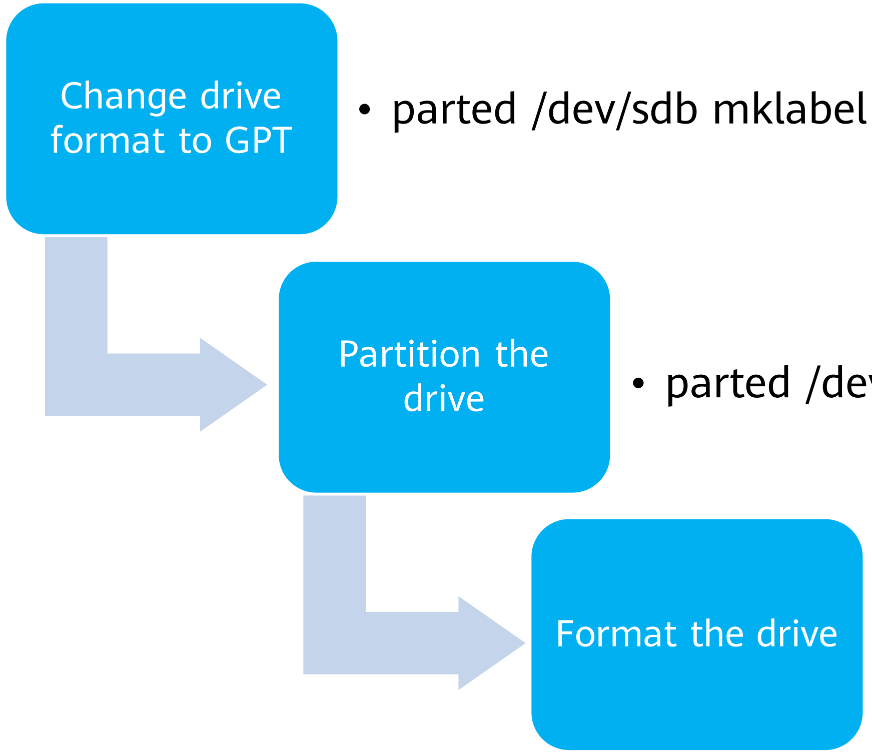
Enter **q**

- Save the settings and exit.

Command	Description
align-check	Check if partition **N** satisfies the alignment constraint of type **min** or **opt**.
mklabel	Create a disklabel (partition table).
name	Name a specified partition.
print	Display partition the table or partitions.
rescue	Rescue a lost partition.
resizepart	Resize a partition.
rm	Delete a partition.
select	Select the device to be edited. By default, only the specified device is operated. You can change the specified device here.
disk_set	Changes the flag on the selected device.
disk_toggle	Switch the state of the flag on the selected device.
quit	Exit.
set	Change the flag on a partition.
toggle	Set or unset the flag on a partition.
unit	Set the default unit.
version	Display version information.

parted Non-Interactive Mode

- Configure a selected drive using the non-interactive mode, that is, the command line mode.
- For example: **parted /dev/sdb**

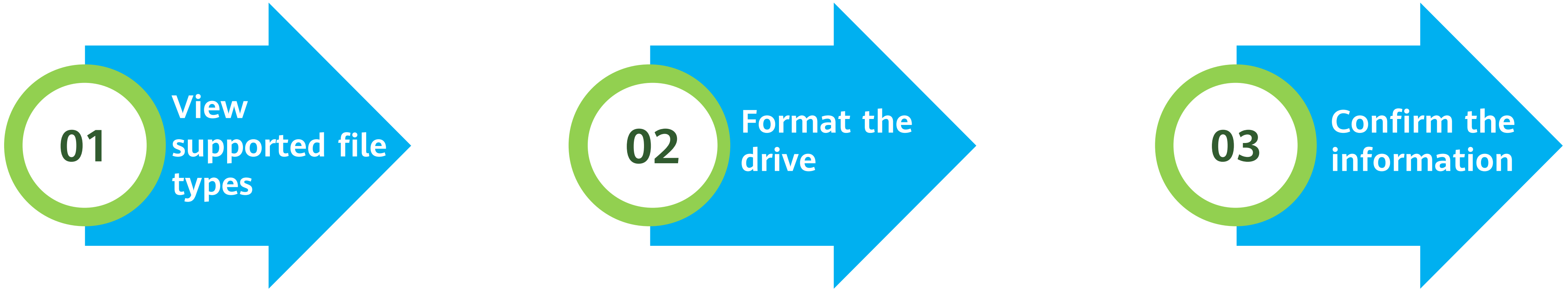


- parted /dev/sdb mklabel gpt
- parted /dev/sdb mkpart primary 0 1000
- mkfs -t ext4 /dev/sdb1

Contents

Drive Formatting

- Formatting involves initializing a drive or a partition in a drive, and formatting a partition into different file systems. This operation usually causes all files in an existing drive or partition to be deleted.



For details about the file types supported by Linux, enter **mkfs** and press **Tab** twice to view commands for supported file types. Select a required command to format the drive.

Run **mkfs.ext4 /dev/sdb2** to format the sdb2 partition under the root into ext4.

After the formatting is completed, run the **ll** command to view drive information, for example, **ll /dev/sdb2**.

mkfs Drive Formatting Command

- Short for "make file system", **mkfs** is used to create a Linux file system in a specified partition.
- Syntax: **mkfs [-V] [-t fstype] [fs-options]** ***filesys*** **[*blocks*]**
  - The command options are as follows:
    - ***filesys***: the drive partition to be checked. For example, **/dev/sda1**.
    - **-V**: produces verbose output.
    - **-t**: specifies the type of file system. The default value for Linux is **ext2**.
    - **-c**: checks the partition for bad blocks before building the file system.
    - **-l** ***bad_blocks_file***: reads information about bad blocks from the **bad_blocks_file** file.
    - ***blocks***: specifies the number of blocks to be used.

Drive Mounting

- A formatted drive has to be mounted before you can use it. The reasons are as follows:
  - Linux adopts "Everything is a file" design. To use the drive, you must set up a link between the drive and a file (directory). The process of setting up a link is called mounting.
  - When you access the directory linked to the sdb2 drive, you are accessing the **sdb2** device file. This directory is equivalent to an entry or interface for accessing sdb2. The drive can be accessed only when this interface is available.

Mount Point Directory

The **media** and **mnt** directories in the root directory are mount point directories. You can also create a directory as a mount point directory.

Temporary Mounting

The **mount /dev/sda5 /test** command mounts **/dev/sda5** to the **test** directory. The mounting becomes invalid after the system is restarted.

Permanent Mounting

A permanently mounted device is automatically mounted when the system is started. Edit the **/etc/fstab** file with Vim to configure permanent mounting.

Introduction to fstab

- Functions of the **/etc/fstab** file
  - This file stores static information about a file system. Upon startup, the OS automatically reads information from the file and mounts the specified file systems to directories accordingly. By writing mounting information to the file, you do not need to manually mount the drives after each startup.
- Format of **fstab**

```
<file system>     <dir>      <type>       <options>            <dump> <pass>
 tmpfs            /tmp        tmpfs      nodev,nosuid              0        0
/dev/sda1         /           ext4       defaults,noatime          0        1
/dev/sda2         none        swap       defaults                  0        0
```

Key Parameters in fstab

Field	Parameter	Function
options	auto	Automatically mounts at startup or after the **mount -a** command is entered.
	ro	Mounts a file system as read-only.
	rw	Mounts a file system as read/write.
	user	Allows any user to mount the file system.
	nouser	Allows only the **root** user to mount the file system.
	dev/nodev	Does or does not interpret block special devices on the file system.
	noatime/ nodiratime	Does not update directory inode access records on this file system or directory. This option improves the performance.
	defaults	Uses the default mounting parameters for the file system.
	sync/async	Specifies synchronous/asynchronous I/O.
	suid/nosuid	Does or does not honor the SUID and SGID bits. This option is typically used for certain special tasks so that common users can temporarily escalate their privileges when running programs.
dump	0 /1	Specifies whether to back up the file system (**1**) or not (**0**) As most users do not install dump, the value of <dump> should be set to 0.
pass	0, 1, 2	Sets the fsck check priority for a file system. This value should be set to 1 for the root partition and 2 for other devices that need to be checked. 0 indicates that the device will not be checked by fsck.

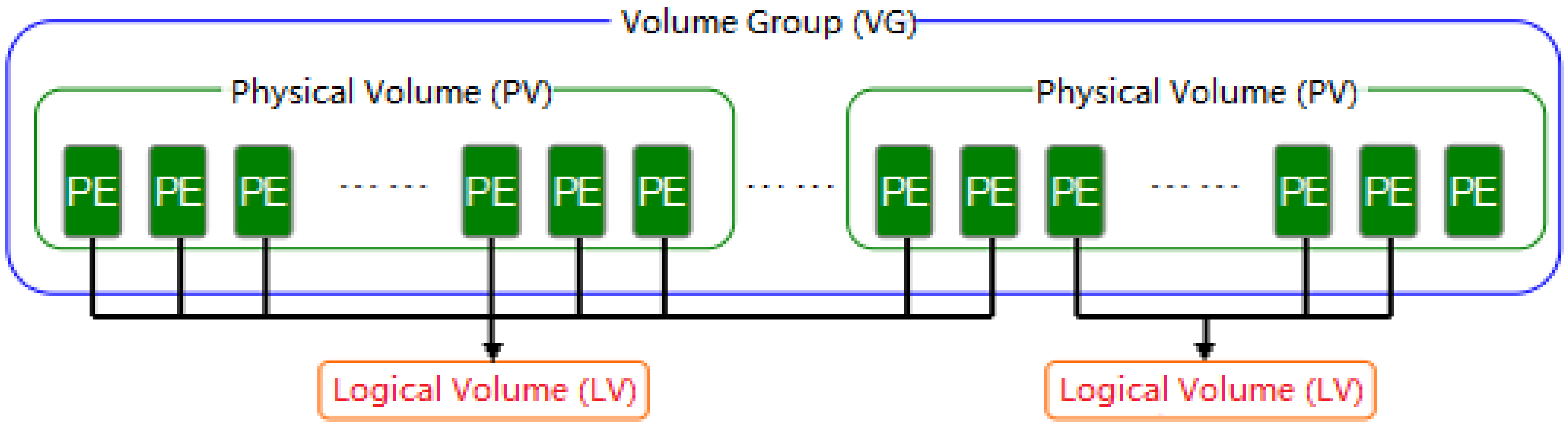
Contents

Logical Volume

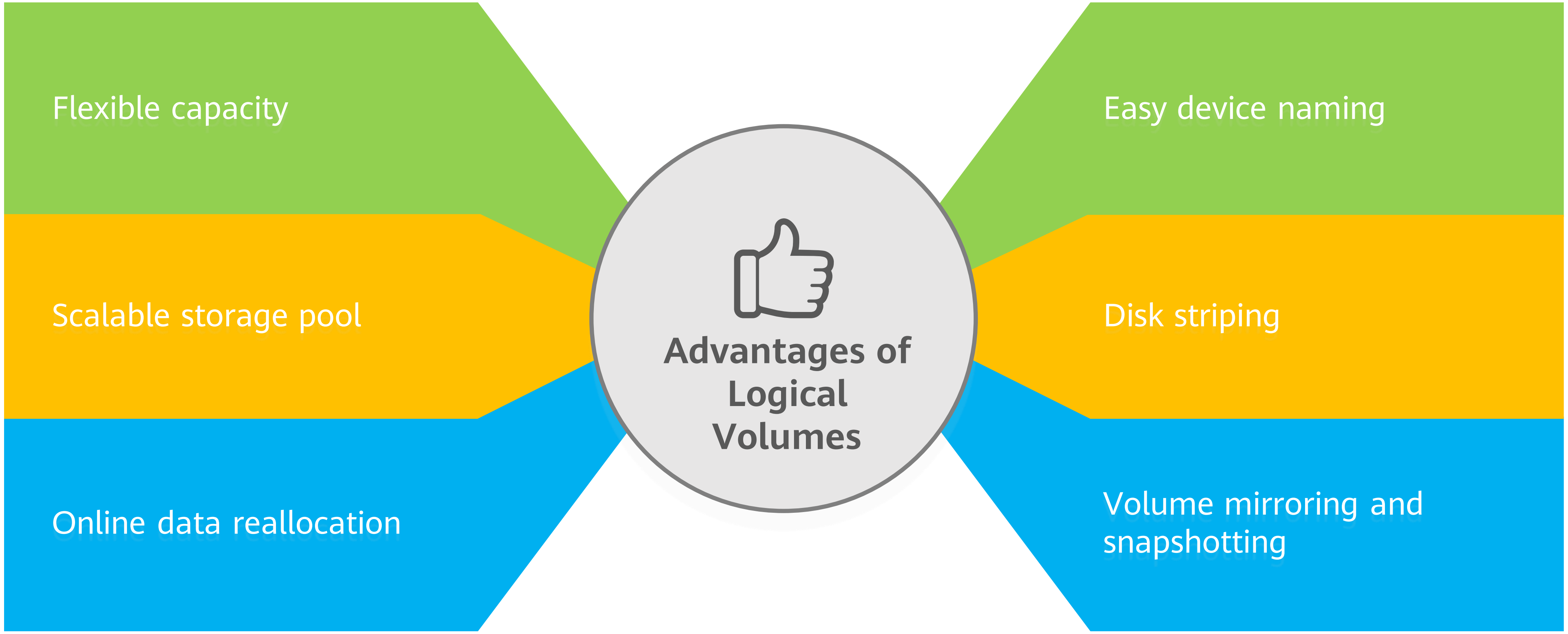
- Logical volume management (LVM) is a mechanism for managing drive partitions in Linux. It is a logical layer built above drives and partitions and below the file systems, which improves the flexibility of drive partition management.
  - Physical extents (PE): a PE with a unique number is the smallest unit that can be addressed by LVM. The sizes of PEs can be specified, with 4 MB being the default. Once the PE size is determined, it cannot be changed. The PE sizes of all physical volumes in the same volume group are the same.
  - Logical extents (LE): the minimum storage unit that can be allocated in LVM. The size of LE depends on the size of PE in the volume group (VG) where the logical volume (LV) is located. Within the same VG, the size of the LE is the same as that of the PE. Generally, the LE and PE are in one-to-one correspondence.
  - Physical volume (PV): an underlying device that provides capacity and stores data. A PV can be an entire drive or a partition on a drive.
  - Volume group (VG): a VG is created based on PVs and consists of one or more of them. Specifically, PVs are combined to provide capacity allocation. An LVM system can contain one or more VGs.
  - Logical volume (LV): an LV is created based on a VG. It is a logical device used by end users, and its size can be expanded or reduced.

Logical Volume Principles

- An LV is a large extended partition (VG) consisting of several drive partitions or block devices (PVs). Note that the PVs can be located in different drive partitions and their sizes can be different, and a VG must contain at least one PV. The extended partition cannot be used directly, and the LV can only be used after it is divided into LVs. The LVs can be formatted into different file systems and can be directly used after being mounted.

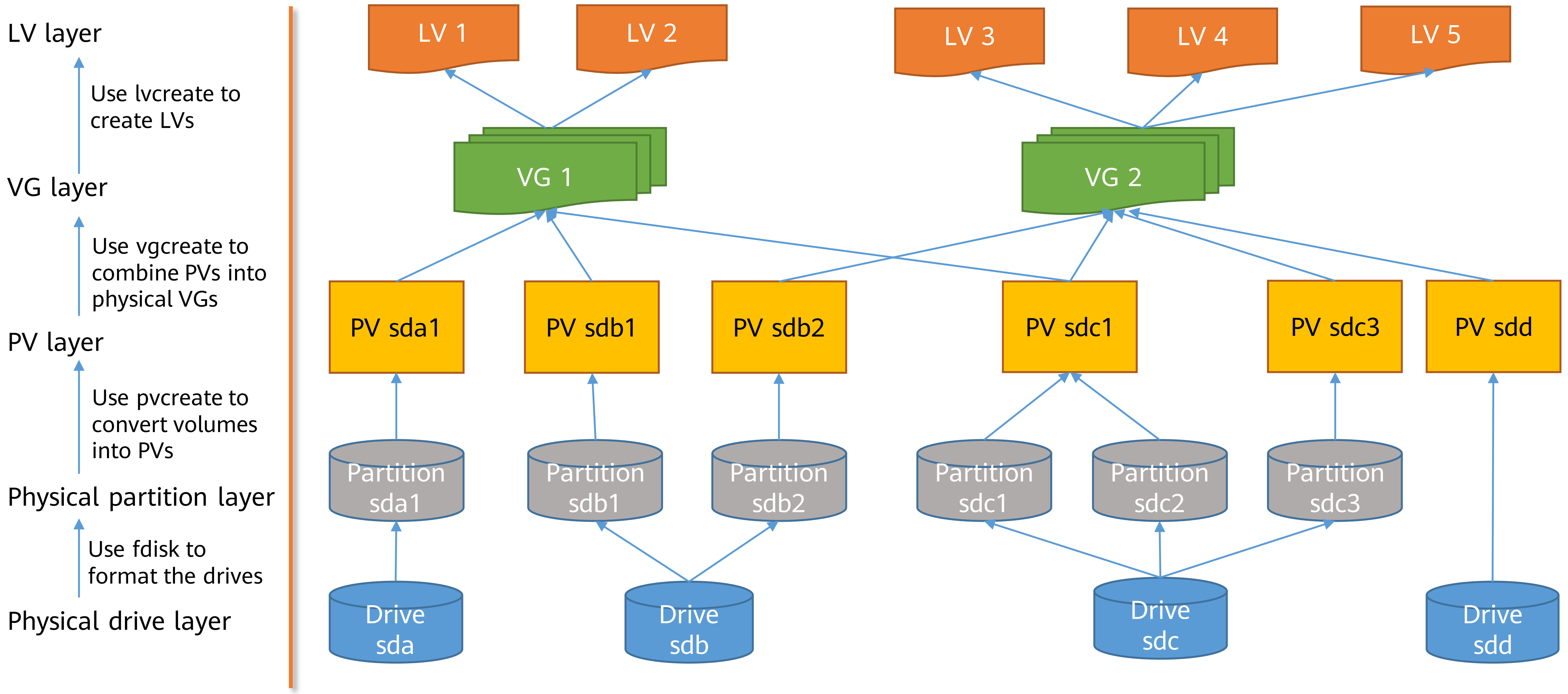


Advantages of Logical Volumes



Contents

Process of Creating a Logical Volume



Logical Volume Management - Using PVs

- The **pvcreate** command is used to create a PV using a physical drive or drive partition.
- Syntax: **pvcreate [option]** ***device_file_name***
  - The command options are as follows:
    - **-f**: forcibly creates a PV without user confirmation.
    - **-u**: specifies the UUID of the device.
    - **-y**: answers yes to all questions.
  - The command parameter is as follows:
    - ***device_file_name***: specifies the device file name for the PV to be created.

Note: After a partition is created, the partition ID is 83 by default. You need to change the ID to 8e to create a PV based on the partition. You can run the **fdisk** command to change the ID.

Logical Volume Management - PV Usage Example

- Create PVs based on drive partitions 6 to 9. Note the use of the braces.
  - Run the following command:

```
[root@openEuler ~]#pvcreate /dev/hda{6,7,8,9}
```

- Run the **pvdisplay**, **pvscan**, or **pvs** command to view PV information.
  - For example, run the **pvs** command:

```
[root@openEuler ~]#pvs # View the general information about the PVs
```

  - Output:

```
 PV          VG       fmt    Attr   PSize     PFree
/dev/sdb1   vg1000   lvm2    --    100.00M   100.00M
/dev/sdb2            lvm2    --    101.98M   101.98M
```

Logical Volume Management - Using VGs

- The **vgcreate** command is used to create a VG of LVM. A VG combines multiple PVs into a whole, shielding the details of the underlying PVs. When creating an LV on a VG, you do not need to consider the PVs.
- Syntax: **vgcreate [option]** ***VG_name PV_list***
  - The command options are as follows:
    - **-l**: specifies the maximum number of LVs that can be created on the VG.
    - **-p**: specifies the maximum number of PVs that can be added to the VG.
    - **-s**: specifies the PE size of a PV in the VG.
  - The command parameter is as follows:
    - ***VG_name***: name of the VG to be created.
    - ***PV_list***: list of PVs to be added to the VG.

Logical Volume Management - VG Usage Example

- Run the **vgcreate** command to create the **vg1000** VG, and then add the **/dev/sdb1** and **/dev/sdb2** PVs.
  - Run the following command:

```
[root@openEuler ~]#vgcreate vg1000 /dev/sdb1 /dev/sdb2
```

- Run the **vgdisplay** or **vgscan** command to view VG information.
  - For example, run the **vgdisplay** command:

```
[root@openEuler ~]#vgdisplay vg1000
```

Note: If **vg1000** is not specified, information about all VGs is displayed.

Logical Volume Management - Using LVs

- The **lvcreate** command is used to create an LV of LVM. LVs are created on VGs.
- Syntax: **lvcreate [option]** ***LV_name***
  - The command options are as follows:
    - **-L** specifies the size of the LV. The unit can be K, M, G, or T (case-insensitive).
    - **-l**: specifies the size of the LV (number of LEs).
  - The command parameter is as follows:
    - ***LV_name***: specifies the name of the LV to be created.

    Note: A created LV has to be formatted and mounted before use. The method is the same as that described in section 2.4. That is, use **mkfs** to format the LV and use **mount** to mount the LV to a directory.

Logical Volume Management - LV Usage Example

- Run the **lvcreate** command to create a 200 MB LV on the **vg1000** VG.
  - Run the following command:

```
[root@openEuler ~]#lvcreate -L 200M vg1000
```

- Run the **lvdisplay** or **lvscan** command to view LV information.
  - For example, run the **lvscan** command:

```
[root@openEuler ~]#lvscan # Scan all LVs.
```

Contents

Expanding Logical Volume Capacity

Online capacity expansion without unmounting

01 Before capacity expansion, run the **vgs** command to check whether the VG space is sufficient.

02 Run the **lvextend –L +SIZE *lv_device*** command to expand the capacity.

03 Run the **resize2fs *device*** command to resize the file system.

Reducing Logical Volume Capacity

Capacity reduction is risky.
Exercise caution.

01 Determine the target size after reduction and ensure that the target LV has sufficient space to store all the original data.

02 Run the **umount** command to unmount the file system, and run the **e2fsck -f** command to perform a forcible check.

03 Run the **resize2fs*device*** command to resize the file system.

04 Run **lvreduce** to reduce the size of the LV.

05 Mount the file system again.

Changing Logical Volume Capacity

- The **lvresize** command is used to adjust (increase or decrease) the size of an LV. As the **lvresize** command provides the **lvextend** and **lvreduce** functions, the procedures for expanding and reducing the capacity of an LV are the same.
- Syntax: **lvresize [option]** ***LV_name***
  - The command options are as follows:
    - **-L** specifies the size of the LV. The unit can be K, M, G, or T (case-insensitive).
    - **-l**: specifies the size of the LV (number of LEs).
  - The command parameter is as follows:
    - ***LV_name***: specifies the name of the LV to be created.
- For example, run the **lvresize** command to increase the capacity:

```
[root@openEuler ~]# lvresize -L +200M /dev/vg1000/lvol0 # Increase the LV space by 200 MB.
```

Quiz

- (Single-answer) Which of the following commands can be used to format the **/dev/hdb6** partition?
  - A. mkfs -t ext4 /dev/hdb6
  - B. format -t ext4 /dev/hdb6
  - C. format /dev/hdb6
  - D. makefile -t ext4 /dev/hdb6

2. (True or false) Because reducing LV capacity is risky, you need to unmount and forcibly check the file system.

Summary

- This course describes the concept of the file system, how to mount and use drives, and how to manage logical volumes.

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI