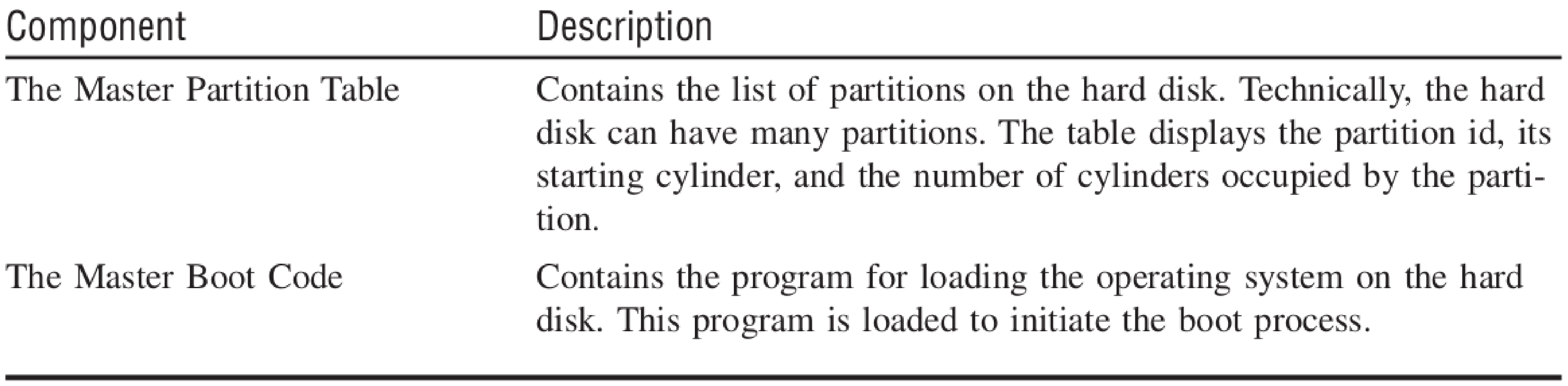
Operating Systems

Lecture 08: Managing Partitions and the Linux Filesystem

Managing Partitions and the Linux Filesystem

Device Recognition by the MBR

- Device recognition is performed by the MBR at system startup, by recognizing the hard disk and all the partitions on it.
- The MBR has two main components that help it to detect any devices that are connected to the system.



Component	Description
The Master Partition Table	Contains the list of partitions on the hard disk. Technically, the hard disk can have many partitions. The table displays the partition id, its starting cylinder, and the number of cylinders occupied by the partition.
The Master Boot Code	Contains the program for loading the operating system on the hard disk. This program is loaded to initiate the boot process.

Managing Partitions and the Linux Filesystem

The Cylinder

- The cylinder is the aggregate of all tracks that reside in the same location on every disk surface.
- On multiple-platter disks, the cylinder is the sum total of every track with the same track number on every surface. On a hard disk, a cylinder comprises the top and corresponding bottom tracks.

Managing Partitions and the Linux Filesystem

Partition Management

Partition management is the process of creating, destroying, and manipulating partitions to optimize system performance. Effective partition management enables you to keep track of the data in the partitions and avoid data overflow. Various utilities, such as sfdisk, partprobe, and GNU parted, are available for partition management.

Managing Partitions and the Linux Filesystem

Partition Management

The sfdisk Utility

The sfdisk utility is used to manipulate partitions. This utility manages partitions by listing the number of partitions and their sizes, checking the partitions, and repartitioning a storage device.

Managing Partitions and the Linux Filesystem

Partition Management

The partprobe Program

The partprobe program is used to update the kernel with changes in the partition tables. The program first checks the partition table and if there are any changes it automatically updates the kernel with the changes.

Managing Partitions and the Linux Filesystem

Navigate Through the Linux Filesystem

Paths

Definition:

A path specifies a location in the filesystem. It begins with the root directory, the

directory at the top of the directory tree, and ends with the directory or file you want

to access. You can refer to a particular file by providing a path to the specific directory that con- tains the file. For example, the directory jsmith contains a subdirectory, work, which contains a file named mywork. To refer to that file, use the following path name:

/home/jsmith/work/mywork. Notice that the forward slash (/) character is used to separate items in the path. The slash that precedes jsmith represents the root directory, from which the path to the file, mywork, begins.

Managing Partitions and the Linux Filesystem

Absolute and Relative Paths

Paths are of two types—absolute and relative. Absolute path refers to the specific location, including the domain name, irrespective of the current working directory or combined paths.

These paths are usually written with reference to the root directory, and therefore start with a forward slash. Paths that do not begin with a forward slash are called relative paths. A relative path is the path relative to the current working directory; therefore, the full absolute path need not be included. These paths can contain the period [.] and double period [..], which are indications for the current and parent directories.

Managing Partitions and the Linux Filesystem

Basic Filesystem Commands

Command	Enables You To
`cd`	Traverse the directory structure. There are several ways to specify the path name of the directory you need to switch to. The syntax of the *cd* command is `cd` *`{absolute or relative path}`*.
`ls`	List the files in the current working directory. This command displays only the file name when the command is run without any options. However, it can be used to list information such as size, file type, and permissions by running the command with the respective options. The syntax of the *ls* command is `ls` *`[options] [absolute or relative path of the directory]`*.
`mv`	Move files and directories from one directory to another, or renames a file or directory. The syntax of the *mv* command is `mv` *`{absolute or relative path}/{file or directory name} {absolute or relative path}/{new file or directory name}`*.

Managing Partitions and the Linux Filesystem

Basic Filesystem Commands

`cp`	Copy a file. The syntax of the *cp* command is `cp [options]` *`{absolute or relative path of the file or directory to be copied}/{file or directory name} {absolute or relative path of the destination}`*. You can use the `-R` option of the `cp` command to copy files along with the source directory recursively. The syntax is: `cp -R /`*`{source directory }/{target directory}`*.
`rm`	Delete files or directories. The syntax of the *rm* command is `rm [options]` *`{absolute or relative path of file or directory}/{file or directory name}`*. You can use the `-R` option of the `rm` command to recursively remove files, subdirectories, and the directory itself. The syntax is: `rm -R` *`{directory and content that needs to be deleted}`*.
`touch`	Change the time of access or modification time of a file to the current time. In addition, the `touch` command creates an empty file if the file name specified as an argument does not exist. The syntax of the *touch* command is `touch` *`{file name}`*.

Managing Partitions and the Linux Filesystem

Basic Filesystem Commands

`touch`	Change the time of access or modification time of a file to the current time. In addition, the `touch` command creates an empty file if the file name specified as an argument does not exist. The syntax of the *touch* command is `touch {file name}`.
`mkdir`	Create a directory. The syntax of the *mkdir* command is `mkdir {directory name}`.
`rmdir`	Delete directories. The syntax of the *rmdir* command is `rmdir {directory name}`.
`pushd`	Add a directory at the top of a stack of directories or rotate a stack of directories. The syntax of the *pushd* command is `pushd [options] {directory name}`.
`popd`	Remove entries from a stack of directories. When no option is specified, it removes the top directory from the stack. The syntax of the *popd* command is `popd [options]`.

Managing Partitions and the Linux Filesystem

Mount Points

Definition:

- A mount point is an access point to information stored on a local or remote storage device. The mount point is typically an empty directory on which a filesystem is loaded, or mounted, to make the filesystem accessible to users. If the directory already has content, the content becomes invisible to the users until the mounted filesystem is unmounted.
- You can use the /etc/fstab file to list the filesystem to be mounted and unmounted when the Linux system boots and shuts down, respectively

Managing Partitions and the Linux Filesystem

Mount Points

The mount Command

- In Linux, a filesystem cannot be accessed directly. It has to be associated with a directory to make it accessible to users. This association is brought about by loading, or mounting, the filesystem in a directory by using the mount command. After using the filesystem, it needs to be disassociated from the directory by unloading, or unmounting, the filesystem using the umount command.

Managing Partitions and the Linux Filesystem

Mount Points

Binaries

- Binaries are source codes that are compiled into executable programs, or are assembled so that they are readable by the computer system. Binaries are encoded so that they can be transmitted over the Internet. In addition, binaries can be pictures, word processing files, or spreadsheet files. Some binaries may contain viruses that can harm the system.

Managing Partitions and the Linux Filesystem

Swap Space

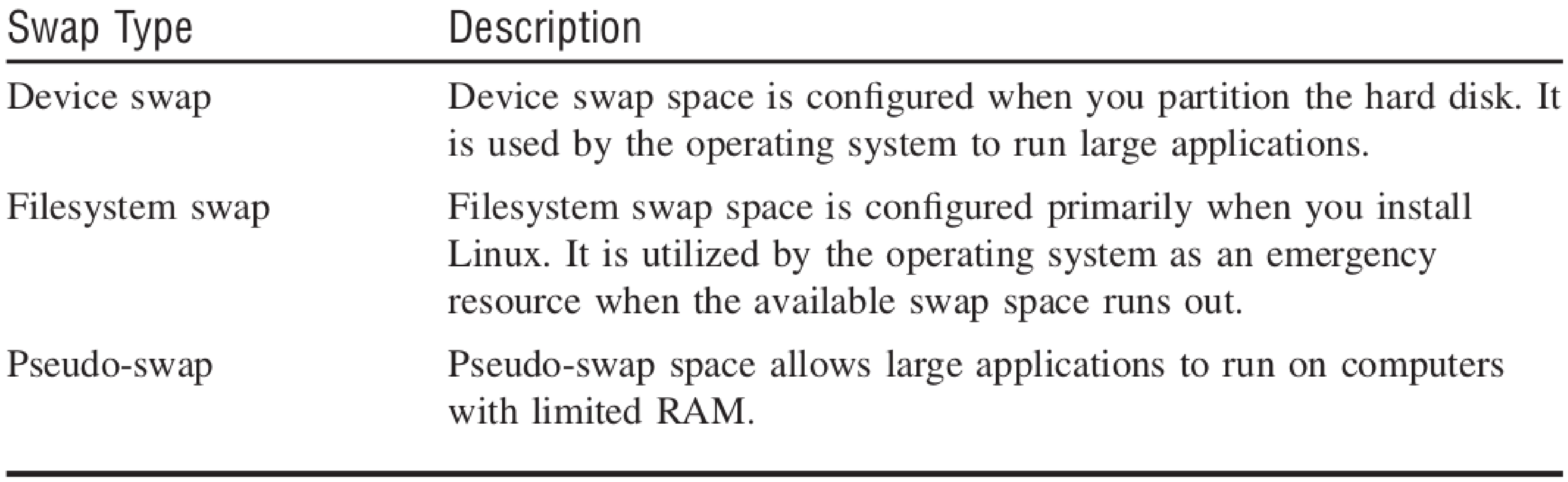
Definition:

Swap space is a partition on the hard disk that is used when the system runs out of physical memory. Linux pushes some of the unused files from the RAM to the swap space to free up memory. Usually, the swap space equals twice the RAM capacity.

Managing Partitions and the Linux Filesystem

Swap Space

Swap space can be one of three types.



Swap Type	Description
Device swap	Device swap space is configured when you partition the hard disk. It is used by the operating system to run large applications.
Filesystem swap	Filesystem swap space is configured primarily when you install Linux. It is utilized by the operating system as an emergency resource when the available swap space runs out.
Pseudo-swap	Pseudo-swap space allows large applications to run on computers with limited RAM.

Managing Partitions and the Linux Filesystem

Swap Files

Swap files are created for storing data that is to be transferred from a system's memory to a disk. It is dynamic and changes in size when data is moved in and out of the memory. It is used as a medium to transfer data from the RAM on to the hard disk.

Swap Partitions

A swap partition is an area of virtual memory on a hard disk to complement the physical RAM in the computer. Swap partitions are created by Linux because they perform better than swap filesystems.

Managing Partitions and the Linux Filesystem

Maintain the Filesystem

Maintaining a filesystem in Linux involves the following tasks:

- Checking the integrity of the filesystem.
- Managing the size of partitions.
- Removing temporary files.
- And, performing system recovery.

Managing Partitions and the Linux Filesystem

Mass Storage Devices

- Mass storage devices are types of storage devices that provide fast access to large amounts of data in a small, reasonably reliable, physical package.
- Hard disks, tape drives, flash drives, CD-R(W), DVD-R(W), and zip drives are some of the common mass storage devices.

ATAPI

- AT Attachment Packet Interface (ATAPI) is a protocol for controlling mass storage devices.
- ATAPI provides commands that are used for hard disks, CD-ROM drives, tape drives, and other devices.

Managing Partitions and the Linux Filesystem

Journaling Filesystems

A journaling filesystem is a method that is used by an operating system to quickly recover after an unexpected interruption, such as a system crash. Journaling filesystems can remove the need for a filesystem check when the system boots. By using journaling filesystems, the system does not write modified files directly on the disk. Instead, a journal is maintained on the disk.

Managing Partitions and the Linux Filesystem

Journaling Filesystems

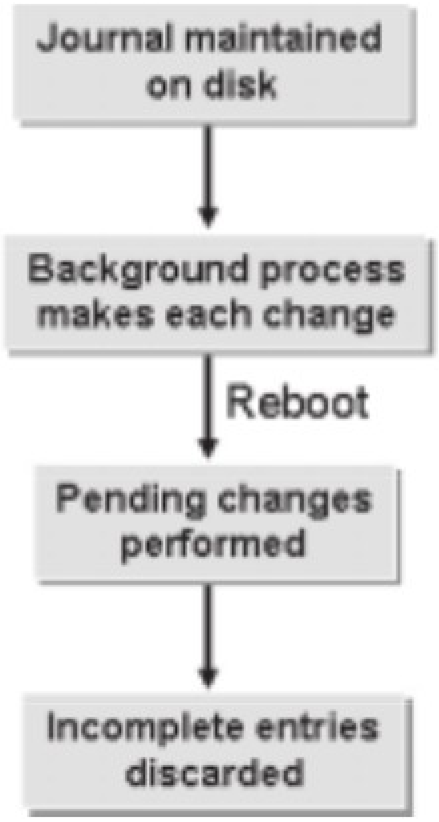
The journaling filesystem process involves the following phases:

1. The journal describes all the changes that must be made to the disk.

2. A background process makes each change as and when it is entered in the journal.

3. If the system shuts down, pending changes are performed when it is rebooted.

4. And, incomplete entries in the journal are discarded.


Journal maintained on disk
Background process makes each change
Reboot
Pending changes performed
Incomplete entries discarded

Performance Issues with Journaling

A journaled filesystem works well with small files and small drives. With the growth of file and drive sizes, performance will suffer. Some of the reasons for poor performance include:

- Filesystem recovery time after a power failure or improper shutdown.
- Bitmap method of tracking the filesystem.
- Wasted space and fragmentation.

Managing Partitions and the Linux Filesystem

The fsck Command

- The fsck command is used to check the integrity of a filesystem.
- Filesystem integrity refers to the correctness and validity of a filesystem.
- Most systems automatically run the fsck command at boot time so that errors, if any, are detected and corrected before the system is used.
- Filesystem errors are usually caused by power failures, hardware failures, or improper shutdown of the system.

Managing Partitions and the Linux Filesystem

The fsck Command

- The syntax of the fsck command is *fsck -t {filesystem type} [options]*.
- You can use the *fsck -r /dev/{filesystem}* command to repair a filesystem.
  - The command will prompt you to confirm your actions. If you are simultaneously checking multiple filesystems, you should not use this option because it allows you to repair only a single filesystem at a time.

Managing Partitions and the Linux Filesystem

The e2fsck Command

- The e2fsck command allows you to check the ext2, ext3, and ext4 filesystems.
- You need to unmount the filesystem before running the e2fsck command to prevent damage to the filesystem.
- The syntax of the e2fsck command is
  - *e2fsck /dev/{filesystem}*

Managing Partitions and the Linux Filesystem

The tune2fs Utility

- The tune2fs utility helps tuning parameters associated with a Linux filesystem.
- Using this utility, a journal can be added to an existing ext2 or ext3 filesystem.
- If the filesystem is already mounted, the journal will be visible in the root directory of the filesystem.
- If the filesystem is not mounted, the journal will be hidden.
- The tune2fs utility is available with most Linux distributions.
- The syntax of the tune2fs utility is ***tune2fs [options] {device name}.***

Managing Partitions and the Linux Filesystem

The dumpe2fs Utility

- The dumpe2fs utility is used for managing ext2, ext3, and ext4 (extended) filesystems.
- It dumps the status of the extended filesystem onto the standard output device and prints the block group information for the selected device.
- The syntax of the dumpe2fs command is ***dumpe2fs [options] [blocksize] {device name}.***

Managing Partitions and the Linux Filesystem

The debugfs Utility

- The debugfs utility allows you to examine and modify ext2, ext3, and ext4 filesystem.
- When executed, the debugfs utility opens an interactive shell that can be used to examine and modify the extended filesystem.

Thanks For Attention