LESSON NOTES

Intro to Linux

System Management

1.2.2 File Compressing and Archiving

Lesson Overview:

Students will:

- Understand the methods used to compress, archive, and backup files of most Linux-based operating systems

Guiding Question: What are some ways to compress files in order to save space or archive, create a backup for Linux systems for later use?

Suggested Grade Levels: 9 - 12

Technology Needed: None

CompTIA Linux+ XK0-005 Objective:

1.2 - Given a scenario, manage files and directories

- File compression, archiving, and backup
  - gzip
  - bzip2
  - zip
  - tar
  - xz
  - cpio
  - dd

This content is based upon work supported by the US Department of Homeland Security's Cybersecurity & Infrastructure Security Agency under the Cybersecurity Education Training and Assistance Program (CETAP).

File Compressing and Archiving

File compression, archiving, and creating backups are fundamental processes for any Linux system administrator or user. These techniques not only help in optimizing disk space utilization but also ensure the safety and integrity of vital data. By compressing files, the size of data is reduced, allowing for efficient storage and faster transmission. Archiving helps in organizing files and directories into a single, compact package, making it easier to manage and share large volumes of data. Creating backups is crucial for safeguarding against data loss or accidental deletion, ensuring that critical information can be restored in the event of hardware failure or malicious attacks. This lesson will explore various tools and methods available in the Linux environment to perform file compression, archiving, and creating backups, thus equipping users with the skills necessary to keep their Linux systems running smoothly and securely.

Backups, sometimes called archives, are crucial for restoring a system or data in the event that the data is destroyed or corrupted. There are multiple types of backups and each has its own rewards and drawbacks. A System Image serves as a clone of the operating system and configuration. While it usually offers no way to recover directories or files, it is a fast way to get a system up and functioning.

A Full Backup makes a copy of everything on the system regardless of when it was created. Although this is useful for recovering files, it can be very time consuming and requires more storage space because of the large volume associated with the full backup.

An Incremental Backup makes a backup of only the data that has been modified since the last backup by cross checking the timestamps of the backup and the file or directory. It takes less time, and each backup uses less storage than a full backup, but it can be very time consuming if using these incremental backups to recover a system.

Often considered the middle ground between a full and incremental backup, a Differential Backup only backups up what has been modified since the last full backup and is performed less often than an incremental backup. The downside is that because the backup is performed less often there is a stronger chance that data could be lost in the event of a corruption or other data compromise.

A Snapshot is a considered a variation of backing up a system in that it first creates a full backup that is typically read-only then it focuses more on the links and the where and how the data is stored. In essence, the snapshot is recording the metadata of the files and data being backed up. Snapshots can occur multiple times a day but since they are only updating the changes between backups, the time and space is reduced dramatically. A snapshot clone is often paired with a snapshot backup. The clone is created from the snapshot and not the original data. This helps minimize performance issues since the original data is not being accessed and it allows modification when a normal snapshot is typically read-only. Regardless of the backup type, the best way to protect data and have it readily available should a system failure occur is to implement a combination of these backup methods.

Archiving files and directories has some similarities to backing up a system but is typically more limited and specific to what data is being archived. In some cases, it may be that archiving data may serve as a space saver for a system. In other cases, it may be that archiving sensitive data or data that has time-retention regulations, such as financial information, may need to be kept long-term but is not necessarily

needed to be accessed continuously. Even though there are several GUI based programs for archiving, the cpio, tar, and dd utilities are used in the command line interface.

The cpio utility stands for copy in and out. It has several options but the -o option is used to archive while the -I option is used to extract. It works by first listing the files using ls then use pipe to tie them to the cpio command. An example of this could be to run the command in the directory containing the files you desire to archive as **`ls | cpio -o > ArchiveFiles.cpio`** where we are listing the files, telling cpio to create the archive file and then naming the archive file as ArchiveFiles.cpio. The cpio utility does maintain the directory reference and because of this, it is useful in creating system image files and full backups but requires specific options when restoring from a different directory.

Another archive utility that is often used is tar which stands for tape archiver. It is similar to cpio in that it can combine several files into one; however, tar also adds the option to compress the files all in one command using the options available with the tar utility. When creating an uncompressed archive, it is referred to as a tar archive file, while a compressed tar file is referred to as a tarball. Using the command **`tar -zcf Archive.tar.gz File?.txt`** compresses multiple files named 'File' with any single-character extension into a single compressed archive file named 'Archive.tar.gz'. The '?' serves as a wildcard, allowing for variations in the file numbering. The '-z' option specifies compression. The option -c is telling tar to create the archive file. Exacting the files would have a similar command with tar -zxf Archive.tar.gz where the -z option tells the command to unzip the file and the -x option is to extract the file.

Regardless of whether backing up or archiving data is desired, both will use a form of file compression. In short, file compression works by getting rid of any data redundancies and instead has the file refer back to the initial data point. Compressing directories can package the files into one file type, eliminating the need for some specific metadata tied to individual files from taking up space.

The gzip, bzip2, and xz utilities are popular when it comes to file compression on Linux systems . The gzip utility is easy and efficient to use. The command **`gzip example.txt`** would compress the file and apply a .gz file extension while using the command **`gunzip example.gz`** would expand the file.

The bzip2 utility offers a higher compression value, increasing the amount of space saved but does take longer to compress files. The command is simply **`bzip2 example.txt`** to compress the file and **`bunzip2 example.bz2`** to expand the file.

The xz utility has become more popular with file compression because it offers a higher compression rate but also offers options to change the rate as desired. The xz utility can be run by typing **`xz example.txt`** to compress the file and **`unxz example.xz`** to expand the file. When working with multiple files, the zip utility is a go to tool. Instead of compressing the files themselves, zip creates a folder and places a copy of the files in the folder, then compresses it into one package and adds a .zip extension. All the aforementioned compression methods are known as lossless compression tools, meaning the original data is compressed but nothing is removed, making them useful for providing backups when needed.

The last utility dd, allows you to copy the entire contents of a disk or partition onto a new drive or partition. The command is simple in that it consists of the command dd then the name of the input device, drive or partition needing to be copied, signified by if=, and then the output device, or the drive or partition being copied to, signified by of=. An example of the command would be **`dd if=input_drive of=output_drive`**. This utility does have some additional steps such as ensuring the drive being

copied and the one being copied to are not mounted. In some cases this may involve booting from an external source such as a disc or USB drive.

Using these methods, the user's data and information can be stored and backed up efficiently allowing for effective use of resources and puts redundancies in place to ensure safety, integrity, and availability of that data.