COMP 1030

WEEK 5 LESSON 1: LINUX PIPELINE COMMANDS / ARCHIVING & RESTORING FILES / MANAGING SOFTWARE

LESSON 5 TOPICS

Agenda

1. **Linux Pipeline Commands:**
   - Purpose / Symbol / tee command
2. **Archiving & Restoring Files:**
   - Purpose / Various Types of compression (**tar**, **gzip**)
3. **Software Management:**
   - Install / Update / Remove Software packages (**apt**)
   - Adding Repositories

PIPELINE COMMANDS

Pipeline Commands

Pipeline Commands use a meta symbol "|"
(called a pipe) to allow a command's **standard output**
to be redirected into the **standard input** of other commands WITHOUT having
to use **temporary** files.

Therefore, a few simple commands can be **combined** to form a
more powerful pipeline command.

Examples:

```
ls -al | more
ls | sort -r
ls | sort | more
ls -l | cut -d" " -f2 | tr 'a-z' 'A-Z"
ls | grep Linux | head -5
```

PIPELINE COMMANDS

Filters

Commands to the **right** of the pipe symbol are referred to as **filters**.

They are called *filters* since those commands are used to **modify** the stdout of the previous command.

Many commands can be "piped" together, but these commands (filters) must be chained in **a specific order**, depending on what you wish to accomplish.

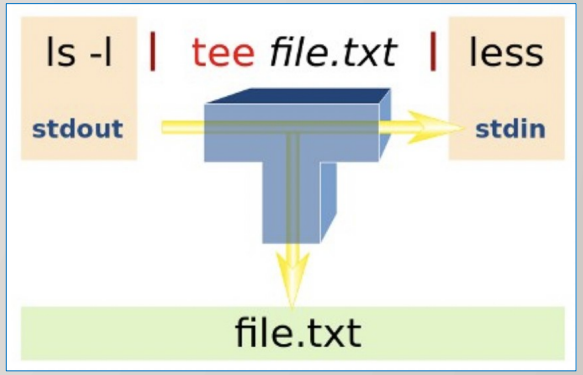
PIPELINE COMMANDS

The tee utility

The **tee** utility can be used to split the flow of standard output between a **text file** and the **terminal screen**.

The **tee** option **-a** can be used to add content to the **bottom** of an existing file as opposed to *overwriting* the file's previous contents.

The reason for the name "**tee**" is that the splitting of the flow of information resembles a capital T.



Examples:

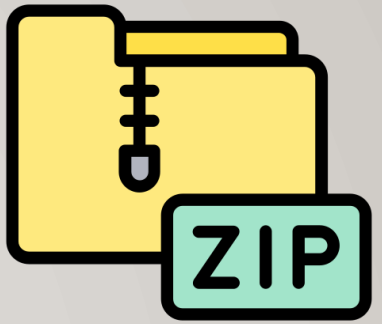
```
ls | tee unsorted.txt | sort
ls | grep Linux | tee matched.txt | more
ls | head -5 | tee -a
```

PIPELINE COMMANDS

Instructor Demonstration

Your instructor will now demonstrate how to issue **Pipeline Commands**.

RESTORING & ARCHIVING FILES



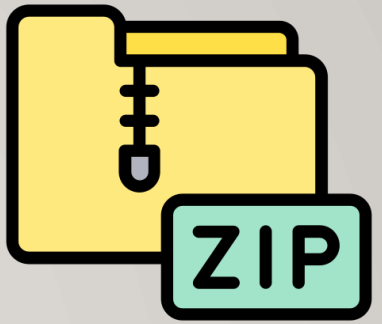
So far you have learned how to manage files and directories for a regular user.

A **Linux system administrator** needs to save disk space for the entire Linux system.
This includes monitoring disk usage so the machine doesn't run out of space.

One method to delay running out of space is **archiving** files:

- Make a compressed backup and remove the original files.
- Since those archive files are smaller they take up less space and can download faster
- Archives are also a popular method of distributing software
  (e.g. make an archive of the source code).

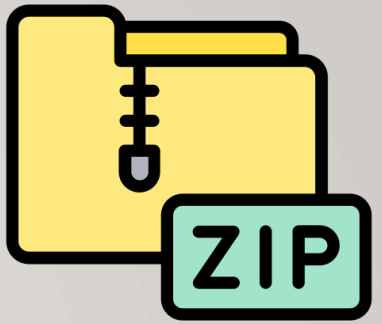
RESTORING & ARCHIVING FILES



Common Archiving / Restoring Commands

Command	Purpose
`tar cvf archivename [files/directories]`	Bundles files (including within directories) into a single archive file (with .tar) extension. argument c means to create file, f to bundle files in directories and v to explain what is being added to archive
`gzip archivename`	Compresses archived file to save space for downloads, etc
`gunzip archivename`	Decompresses zipped archived file to original archive file (to be unbundled)
`tar xvf archivename [files/directories]`	Unbundles files (including within directories) to be used. argument x means to extract archive
`tar czvf archivename [files/directories]`	Both bundle to a compressed (zipped) archive file Argument z means to zip
`tar xzvf archivename [files/directories]`	Simultaneously decompress archived file and unbundles files (including within directories) to be used. Argument z means to unzip

RESTORING & ARCHIVING FILES

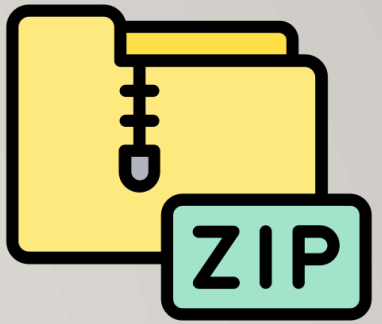


Instructor Demonstration

Your instructor will now demonstrate how to

Archive and Restore Files

RESTORING & ARCHIVING FILES (EXAMPLE)



Practical Use of Using Archives

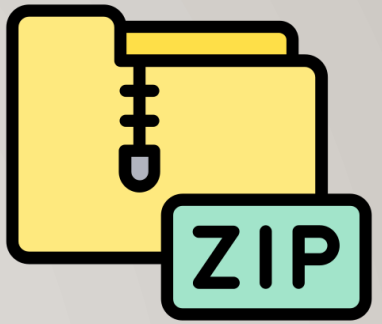
An excellent example of using the tar command is to **download, unpack, and compile source code** in order to install a program (application).

Downloading, unpacking and compiling source code was the **traditional method** of installing programs for Unix/Linux. This method can be frustrating since it does NOT resolve dependency issues (missing programs or libraries).

Although generally considered obsolete, you will sometimes need software that isn't nicely prepared in a package manager for you.

Some software may be so old or new that it isn't available for online repositories.

RESTORING & ARCHIVING FILES (EXAMPLE)



Downloading / Compiling Source Code

Here are the typical steps in installing software by compiling source code:

- Download archive from the internet (web browser, or **wget** command)
- Extract the archive (**tar**)
- Go to directory containing source code
- Compile source code (**./configure** && **make**)
- Make the software available as a command (**make install**)

SOFTWARE MANAGEMENT

Package Managers

A **package manager** or **package-management system** is a collection of **software tools** that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner.

A package manager deals with *packages*, distributions of software and data in **archive files**. Packages contain **metadata**, such as the software's name, description of its purpose, version number, vendor, checksum, and a list of dependencies necessary for the software to run properly.

SOFTWARE MANAGEMENT

Package Managers

Upon installation, metadata is stored in a **local package database**.

Package managers typically maintain a database of software dependencies and version information to prevent software mismatches and missing prerequisites. They work closely with software repositories, binary repository managers, and app stores.

Package managers **simplify** the management of software.

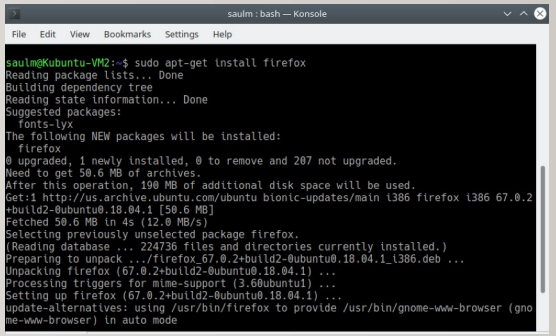
SOFTWARE MANAGEMENT

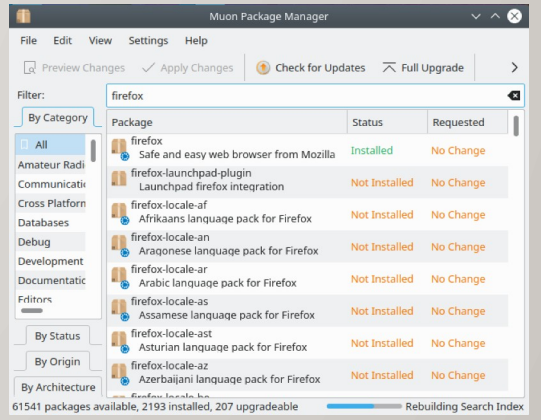
The apt Package Manager

Ubuntu (*Kubuntu*) is based on the **Debian Linux** distribution. Both Ubuntu and Debian-based Linux distributions use the same package manager called **apt** (**Advanced Packaging Tool**)

This **automates** the retrieval, configuration, management and installation of software packages. Software is contained via online repositories via the Internet. The repository allows for checking, downloading and installing dependency programs and library files required by the program set to install.

There are even graphical-based user interfaces to search for and install software (such as **synaptic**), but it is important to use command-line (i.e. apt) since Linux servers usually do not support graphical desktop environments.





SOFTWARE MANAGEMENT

The **apt** utility is itself the package name containing the set of **tools** (and requiring the libraries) that support its functionality.

Command	Purpose
apt-get install package-name	Installs package-name
apt-get remove package-name	Removes package-name
apt-get purge package-name	Removes package-name and its configuration files
apt-cache search keyword	Search online repository for package
add-apt-repository	Add additional online repository
apt-update	Downloads newest updates and dependencies
apt-cache showpkg package-name	Display package information (including dependencies)
apt-rpm	The apt utility is also modified to work with the RPM Package Manager system

SOFTWARE MANAGEMENT

Software Repositories

Repositories are added into the **/etc/apt/sources.list** file.

Usually a main (basic) repository is added up to this file upon installation. Other repositories can be added to access other software.

Refer to the apt command to add an additional repository
in the previous slide.

SOFTWARE MANAGEMENT

Instructor Demonstration

Your instructor will now demonstrate how to

Manage Software on your Kubuntu Linux VM

MOVING FORWARD

In week 8 (after the reading week), we will be learning about **User Account / Group Management**, and **Managing Permissions**.