The pipe (|) and tee (tee) commands in Linux are both used to manipulate the flow of data between commands. Here's a breakdown of their individual functionalities and how they can be used together:

Pipe (|)

- **Purpose:** Redirects the standard output of one command as the standard input for another command.
- **Example:**
  Bash
  ```
  ls | grep txt  # Lists files with ".txt" extension (output of ls piped to grep)
  ```
  In this example, the output of ls (listing files) is piped to grep, which searches for lines containing ".txt".

Tee (tee)

- **Purpose:** Duplicates the standard output of a command, sending it to both the terminal and a specified file.
- **Example:**
  Bash
  ```
  ls | tee file_list.txt  # Lists files and saves them to file_list.txt
  ```
  Here, ls lists files, and the output is piped to tee. tee writes the output to both the terminal (so you see it) and the file file_list.txt.

Using Pipe and Tee Together

While not as common, you can combine | and tee to achieve specific workflows. Here's a scenario:

Bash

```
command1 | tee output.txt | command2
```

In this example:

1. command1 executes, sending its output to both the terminal and output.txt using tee.
2. The **same output** from command1 is then piped to command2 for further processing.

Important Note:

This specific usage pattern with tee in the middle might not be the most efficient approach, depending on the commands involved. tee typically reads from standard input, so in the above example, command2 might receive a partial output depending on how tee buffers the data.

Alternative Approaches:

- If you only need to save the output and don't necessarily need to see it in the terminal, redirect the output directly to a file:
  Bash
  ```
  command1 > output.txt
  ```
- If you need the output for both another command and viewing, consider saving it to a temporary file first:
  Bash
  ```
  command1 > temp_output.txt
  cat temp_output.txt  # View the temporary file
  command2 < temp_output.txt  # Use the temporary file as input for command2
  rm temp_output.txt  # Clean up the temporary file
  ```

By understanding the use cases of | and tee, you can effectively manipulate the flow of data in your Linux commands.