Command Line Basics for Ubuntu

An overview of essential command-line functions for Ubuntu, categorized by purpose.

Navigation

Commands to move around and explore the filesystem:

- **pwd**: Print the current working directory.

  ```
  pwd
  ```

- **ls**: List directory contents.

  ```
  ls -l  # Long format
  ls -a  # Show hidden files
  ```

- **cd**: Change directory.

  ```
  cd /path/to/directory
  cd ..  # Move up one level
  cd ../ # Equivalent
  cd .   # "Move" to the current directory (i.e., do nothing)
  cd ./  # Equivalent
  cd ./././.  # Equivalent
  cd ~   # Move to home directory (also with no input argument)
  cd -   # Back to previous
  ```

File Management

Commands to manage files and directories:

- **touch**: Create an empty file or set its last edited date to now.

  ```
  touch newfile.txt
  ```

- **mkdir**: Create a directory.

  ```
  mkdir newdir
  ```

- **rm**: Remove files or directories.

  ```
  rm file.txt
  rm -r directory  # Remove a directory and its contents, -r for recursive
  ```

  > [!CAUTION] The command line is a harsh and unforgiving environment. File deletion is immediate and permanent!

- **cp**: Copy files or directories.

  ```
  cp source.txt destination.txt
  cp -r sourcedir/ destinationdir/
  ```

- **mv**: Move or rename files or directories.

  ```
  mv oldname.txt newname.txt
  mv file.txt /new/location/
  ```

  [!CAUTION] The `cp` and `mv` commands can overwrite files immediately and permanently!

- Always double-check the source and destination paths.
- Use the `-i` option (interactive) to confirm overwrites before proceeding:

```
cp -i source.txt destination.txt
mv -i oldname.txt newname.txt
```

Understanding File Permissions

Viewing Permissions

Use `ls -l` to view file permissions.

```
ls -l
```

Example output:

```
-rwxr-xr-- 1 user group 1234 Jan 1 12:34 file.txt
```

Interpreting Permissions

Permissions are displayed in a 10-character string. Here's what each part means:

1. **File Type:** The first character indicates the type of file:
   - `-`: Regular file
   - `d`: Directory
   - `l`: Symbolic link
2. **Owner, Group, and Others:**
   - The next 9 characters are divided into three sets of three:
     - **Owner permissions** (characters 2-4)
     - **Group permissions** (characters 5-7)
     - **Other permissions** (characters 8-10)

   Each set can contain:
   - `r`: Read permission
   - `w`: Write permission
   - `x`: Execute permission
   - `-`: No permission

Example:

```
-rwxr-xr--
```

- `rwx`: Owner can read, write, and execute.
- `r-x`: Group can read and execute but not write.
- `r--`: Others can only read.

Changing Permissions

Use `chmod` to modify permissions:

```
chmod 755 file.txt  # rwxr-xr-x
chmod +x script.sh  # Add execute permission.
```

[!NOTE] By default, new scripts can't be run.

Changing Ownership

Use `chown` to change file ownership:

```
chown user:group file.txt
```

```
- **`ls -l`**: View permissions.
```

```bash
ls -l file.txt
```

Text Editing

Commands for working with text files:

- **nano**: Simple command-line text editor.

  ```
  nano file.txt
  ```

- **cat**: Display file contents.

  ```
  cat file.txt
  ```

- **less**: View file contents page by page.

  ```
  less file.txt
  ```

- **echo**: Print text to the console or append to a file.

  ```
  echo "Hello, World!" > file.txt
  echo "Additional line" >> file.txt
  ```

Pipes (|)

A pipe (|) is used in Linux to pass the output of one command as input to another. This allows chaining multiple commands together for efficient processing.

Syntax

```
command1 | command2
```

Examples

1. **View Long Output:**

   ```
   ls -l | less
   ```

   Sends the long output of `ls -l` to `less` for easy navigation.

2. **Filter and Count Lines:**

   ```
   grep "pattern" file.txt | wc -l
   ```

   Counts the number of lines matching "pattern" in `file.txt`.

3. **Combine Commands:**

   ```
   cat file.txt | sort | uniq
   ```

   Sorts `file.txt` and removes duplicate lines.

Redirection (> and >>)

Redirection (> or >>) sends the output of a command to a file instead of the terminal.

Operators

- >: Overwrites the file with the command output.
- >>: Appends the command output to the file.

Examples

1. **Write Output to a File:**

   ```
   echo "Hello, World!" > output.txt
   ```

   Writes "Hello, World!" to `output.txt` (overwrites if it exists).

2. **Append Output to a File:**

   ```
   echo "Additional line" >> output.txt
   ```

   Adds "Additional line" to the end of `output.txt`.

3. **Combine Commands with Redirection:**

   ```
   ls -l | grep "pattern" > filtered_output.txt
   ```

   Filters the `ls -l` output for "pattern" and writes it to `filtered_output.txt`.