Command Line Functions for Ubuntu

Slightly advanced use of the bash shell.

`sudo`

Running commands with administrative privileges:

- **`sudo`**: Execute a command as the superuser.

  ```
  sudo command
  sudo apt update
  ```

  [!CAUTION] Superuser privileges are normally password protected because you can brick your system with them.

Package Management (APT and Snap)

APT

- **`apt update`**: Update the package list.

  ```
  sudo apt update
  ```

- **`apt upgrade`**: Upgrade all packages.

  ```
  sudo apt upgrade
  ```

  [!TIP] If you use a linux system, you should regularly update and upgrade for security.

- **`apt install`**: Install a package.

  ```
  sudo apt install package_name
  ```

- **`apt remove`**: Remove a package.

  ```
  sudo apt remove package_name
  ```

  [!WARNING] There are very many packages that do important background stuff. Don't remove a package just because you don't recognize it.

Snap

- **`snap install`**: Install a Snap package.

  ```
  sudo snap install package_name
  ```

- **`snap remove`**: Remove a Snap package.

  ```
  sudo snap remove package_name
  ```

- **`snap list`**: List installed Snap packages.

  ```
  snap list
  ```

Accessing Manuals and Help

- **`man` (manual):** View the manual for a command.

  ```
  man command_name
  ```

  Example:

  ```
  man ls
  ```

  Press `q` to quit the manual.

- **`--help`:** Many commands provide a quick help summary.

  ```
  command_name --help
  ```

  Example:

  ```
  ls --help
  ```

- **`info`:** Access detailed documentation for certain commands.

  ```
  info command_name
  ```

  Example:

  ```
  info coreutils
  ```

- **Searching Manuals:** Use `man -k` to search the manual pages for a keyword.

  ```
  man -k keyword
  ```

  Example:

  ```
  man -k copy
  ```

Common Shortcuts for Efficiency

- **Tab Completion:** Use the `Tab` key to auto-complete file names, commands, or directories.
- **Command History:** Use the `Up` and `Down` arrow keys to browse previous commands. You can also search with `Ctrl + R` (reverse search).
- **Cancel a Command:** Press `Ctrl + C` to stop a running command.

`alias`

The `alias` command in Linux allows you to create shortcuts for frequently used commands, making them easier and faster to type.

Syntax

```
alias name='command'
```

Example

Create a shortcut for a long command:

```
alias ll='ls -alF'
```

- Now, typing `ll` runs the `ls -alF` command, which lists all files in long format and classifies file types.

Create a shortcut for a command that makes me feel icky:

```
alias boop='touch'
```

They can get more complex

```
alias please='sudo $(fc -ln -1)'
```

Viewing Existing Aliases To see all current aliases:

```
alias
```

Removing an Alias To remove an alias, use the `unalias` command:

```
unalias name
```

Example:

```
unalias ll
```

Making Aliases Permanent Aliases created in a terminal session are temporary. To make them permanent, add them to your shell configuration file: - For `bash`: Add to `~/.bashrc` - For `zsh`: Add to `~/.zshrc`

Example:

```
echo "alias ll='ls -alF'" >> ~/.bashrc
source ~/.bashrc
```

This ensures the alias is available every time you open a new terminal session.

Search

The command line offers powerful tools to search for files and text.

`grep` `grep` searches for patterns in text files or output. - **Syntax:** `bash    grep 'pattern' filename` - **Example:** Search for the word "error" in a log file: `bash    grep 'error' /var/log/syslog`

`locate` `locate` quickly searches a pre-built database for file names. - **Syntax:** `bash    locate filename` - **Example:** Find files containing "config": `bash    locate config` - **Note:** Update the database with `sudo updatedb` if necessary.

`find` `find` searches for files and directories based on various criteria. - **Syntax:** `bash    find path options` - **Example:** Find all `.txt` files in the current directory: `bash    find . -name "*.txt"`

Combining Commands with Pipes Use `grep` with `history` to search your command history: - **Example:** Search for all previous commands using `git`: `bash    history | grep git`

These tools allow you to efficiently locate files and text, even in large systems or output streams.

Cron

Automating tasks with `cron`:

- `crontab -e`: Edit the crontab file to schedule tasks.

  ```
  # Example crontab entry
  # m h  dom mon dow   command
    0 5   *   *   *   /path/to/script.sh  # Run daily at 5 AM
  ```

- `crontab -l`: List current cron jobs.

  ```
  crontab -l
  ```

SSH

Securely connecting to remote machines:

- `ssh`: Open an SSH session.

  ```
  ssh user@hostname_or_IP
  ```

- `scp`: Copy files over SSH.

```bash
scp file.txt user@hostname:/remote/path/
```

Git

Version control and collaboration:

- **git clone**: Clone a repository.

  ```bash
  git clone https://github.com/user/repository.git
  ```

- **git add**: Stage changes.

  ```bash
  git add file.txt
  ```

- **git commit**: Commit staged changes.

  ```bash
  git commit -m "Descriptive message"
  ```

- **git push**: Push changes to a remote repository.

  ```bash
  git push origin branch_name
  ```

- **git pull**: Pull updates from a remote repository.

  ```bash
  git pull origin branch_name
  ```

- **git status**: Check the status of the repository. “‘bash git status