Working with Files: Takeaways

Syntax

- Create a file

  ```
  touch [name of file]
  ```

- Print text

  ```
  echo [string of text]
  ```

- Write text to file

  ```
  echo [string of text] > [name of file]
  ```

- Edit a file without redirection

  ```
  nano [name of file]
  ```

- View permissions on files and folders

  ```
  ls -l
  ```

- Get info about a file

  ```
  stat [file name]
  ```

- Modify file permissions

  ```
  chmod [octal notation integer] [file name]
  ```

- Move file

  ```
  mv [file name] [destination path]
  ```

- Copy file

  ```
  cp [file name] [new file name]
  ```

- Delete file

  ```
  rm [name of file]
  ```

- Switch and run as root user

  ```
  sudo
  ```

Concepts

- Every program writes to standard output and receives input through standard output.
- If the program throws an error while running, it writes it to standard error.
- `stderr` and `stdout` usually display on the monitor, while `stdin` is the input from the keyboard.
- `stdout` , `stderr` , and `stdin` exist because these standard streams allow the interfaces to be abstract.
- We can redirect standard streams to connect them to different sources.
- In Unix, every file and folder has permissions associated with it. These permissions have three scopes:
    - `owner` : The user who created the file or folder
    - `group` : Users in the owner's group (on Unix systems, an owner can place users in groups)
    - `everyone` : All other users on the system who aren't the user or in the user's group
- Each scope can have any of three permissions (a scope can have multiple permissions at once):
    - `read` : The ability to see what's in a file (if defined on a folder, the ability to see what files are in a folder)
    - `write` : The ability to modify a file (if a folder, the ability to delete, modify, and rename files in the folder)
    - `execute` : The ability to run a file (some files are executable, and need this permission to run)
- Each permission can be granted or denied to each scope.
- The character for read is `r` , the character for write is `w` , and the character for execute is `x` .
- If a scope doesn't have a permission, a dash takes the place of it instead.
- We can use octal notation to represent permissions for all scopes with 4 digits.
    - `---` : No permissions; corresponds to 0
    - `-- x` : Execute only permission; corresponds to 1
    - `-w-` : Write only permissions; corresponds to 2
    - `-wx` : Write and execute permissions; corresponds to 3
    - `r--` : Read only permissions; corresponds to 4
    - `r- x` : Read and execute permissions; corresponds to 5
    - `rw-` : Read and write permissions; corresponds to 6

    - `rwx` : Read, write, and execute permissions; corresponds to 7

- Files typically have extensions like .txt and .csv that indicate the file type.
- Rather than relying on extensions to determine file type, Unix-based operating systems like Linux use media types, which are also called MIME types.
- The root user has all permissions and access to all files by default.

Resources

- Standard streams
- Octal