Command line

Navigation

- The *command line* is a text interface for the computer's operating system. To access the command line, we use the terminal.
- A *filesystem* organizes a computer's files and directories into a tree structure. It starts with the *root directory*. Each parent directory can contain more child directories and files.
- From the command line, you can navigate through files and folders on your computer:
- **pwd** outputs the name of the current working directory.
- **ls** lists all files and directories in the working directory.
- **cd** switches you into the directory you specify.
- **mkdir** creates a new directory in the working directory.
- **touch** creates a new file inside the working directory.

Manipulation

- Options modify the behavior of commands:
- **ls -a** lists all contents of a directory, including hidden files and directories
- **ls -l** lists all contents in long format
- **ls -t** orders files and directories by the time they were last modified
- Multiple options can be used together, like **ls -alt**
- From the command line, you can also copy, move, and remove files and directories:
- **cp** copies files
- **mv** moves and renames files
- **rm** removes files
- **rm -r** removes directories
- Wildcards are useful for selecting groups of files and directories

Redirection

- *Redirection* reroutes standard input, standard output, and standard error.
- The common redirection commands are:
- > redirects standard output of a command to a file, overwriting previous content.
- >> redirects standard output of a command to a file, appending new content to old content.
- < redirects standard input to a command.
- | redirects standard output of a command to another command.
- A number of other commands are powerful when combined with redirection commands:

- **sort:** sorts lines of text alphabetically.
- **uniq:** filters duplicate, adjacent lines of text.
- **grep:** searches for a text pattern and outputs it. Stands for "global regular expression print". It searches files for lines that match a pattern and returns the results. It is also case sensitive.
- **grep -i** enables the command to be case insensitive.

The above commands are a great way to get started with grep. If you are familiar with regular expressions, you can use regular expressions to search for patterns in files.

- **grep -R** searches all files in a directory and outputs filenames and lines containing matched results. -R stands for "recursive".
- **grep -Rl** searches all files in a directory and outputs only filenames with matched results. -R stands for "recursive" and l stands for "files with matches".
- **sed :** searches for a text pattern, modifies it, and outputs it.

Environment

The *environment* refers to the preferences and settings of the current user.

- The *nano* editor is a command line text editor used to configure the environment.
- The ^ stands for the Ctrl key.
- **Ctrl + O** saves a file. 'O' stands for output.
- **Ctrl + X** exits the nano program. 'X' stands for exit.
- **Ctrl + G** opens a help menu.
- **clear** clears the terminal window, moving the command prompt to the top of the screen.
- **~/.bash_profile** is where environment settings are stored. You can edit this file with nano.
- The **~** represents the user's home directory.
- The **.** indicates a hidden file.
- The command **nano ~/.bash_profile** opens up **~/.bash_profile** in nano.
- The command **source ~/.bash_profile** activates the changes in **~/.bash_profile** for the current session. Instead of closing the terminal and needing to start a new session, source makes the changes available right away in the session we are in.
- *environment variables* are variables that can be used across commands and programs

and hold information about the environment.

- **export VARIABLE="Value"** sets and exports an environment variable.
- **alias pd="pwd"** - The alias command allows you to create keyboard shortcuts, or aliases, for commonly used commands. Creates the alias pd for the pwd command, which is then saved in the bash profile. Each time you enter **pd**, the output will be the same as the **pwd** command. The command **source ~/.bash_profile** makes the alias **pd** available in the current session. Each time we open up the terminal, we can use the alias.
- **alias hy="history"** - **hy** is set as alias for the history command in the bash profile. The **alias** is then made available in the current session through source. By typing **hy**, the command line outputs a history of commands that were entered in the current session.
- **alias ll="ls -la"** - **ll** is set as an alias for **ls -la** and made available in the current session through source. By typing **ll**, the command line now outputs all contents and directories in long format, including all hidden files.
- **USER** is the name of the current user.
- **PS1** is a variable that defines the makeup and style of the command prompt.
- **export PS1=">> "** sets the command prompt variable and exports the variable. Here we change the default command prompt from $ to >>.

After using the **source** command, the command line displays the new command prompt.

- **HOME** is the home directory. It is usually not customized.
- **echo $HOME**, the terminal displays the path **/home/ccuser** as output.

You can customize the HOME variable if needed, but in most cases this is not necessary.

- **PATH** returns a colon separated list of file paths. It is customized in advanced cases.
- **echo $PATH** lists the following directories
- **env** returns a list of environment variables.
- **env | grep PATH** is a command that displays the value of a single environment variable.

Git

The most commonly used git commands are:

add Add file contents to the index

bisect Find by binary search the change that introduced a bug

branch List, create, or delete branches

checkout Checkout a branch or paths to the working tree

clone Clone a repository into a new directory

commit Record changes to the repository

diff Show changes between commits, commit and working tree, etc

fetch Download objects and refs from another repository

grep Print lines matching a pattern

init Create an empty Git repository or reinitialize an existing one

log Show commit logs

merge Join two or more development histories together

mv Move or rename a file, a directory, or a symlink

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

rebase Forward-port local commits to the updated upstream head

reset Reset current HEAD to the specified state

rm Remove files from the working tree and from the index

^Chow Show various types of objects

$ git add filename.txt

$ git commit -m "Comment"

[master 3c9ee1e] Finalyzing dialog

1 file changed, 4 insertions(+)

git checkout HEAD filename restore the file in your working directory to look as it did when you last made a commit

Three different ways to backtrack in Git. You can use these skills to undo changes made to your Git project.

- **git checkout HEAD** filename: Discards changes in the working directory.
- **git reset HEAD** filename: Unstages file changes in the staging area.
- **git reset SHA**: Can be used to reset to a previous commit in your commit history.

Git *branching* allows users to experiment with different versions of a project by checking out separate *branches* to work on.

The following commands are useful in the Git branch workflow.

- **git branch**: Lists all a Git project's branches.
- **git branch branch_name**: Creates a new branch.
- **git checkout branch_name**: Used to switch from one branch to another.
- **git merge branch_name**: Used to join file changes from one branch to another.
- **git branch -d branch_name**: Deletes the branch specified.

Enter git remote -v to list the remotes.

Notice the output:

```
origin    /home/ccuser/workspace/curriculum/science-quizzes (fetch)
origin    /home/ccuser/workspace/curriculum/science-quizzes (push)
```

- Git lists the name of the remote, **origin**, as well as its location.
- Git automatically names this remote **origin**, because it refers to the remote repository of origin. However, it is possible to safely change its name.
- The remote is listed twice: once for (**fetch**) and once for (**push**). We'll learn about these later in the lesson.

An easy way to see if changes have been made to the remote and bring the changes down to your local copy is with:

git fetch this command will not *merge* changes from the remote into your local repository. It brings those changes onto what's called a *remote branch*. Learn more about how this works

below.

A *remote* is a Git repository that lives *outside* your Git project folder. Remotes can live on the web, on a shared network or even in a separate folder on your local computer.

- The *Git Collaborative Workflow* are steps that enable smooth project development when multiple collaborators are working on the same Git project.

We also learned the following commands

- **git clone**: Creates a local copy of a remote.
- **git remote -v**: Lists a Git project's remotes.
- **git fetch**: Fetches work from the remote into the local copy.
- **git merge origin/master**: Merges origin/master into your local branch.
- : Pushes a local branch to the origin remote.

Git projects are usually managed on Github, a website that hosts Git projects for millions of users. With Github you can access your projects from anywhere in the world by using the basic workflow you learned here.