18. The Environment

commands table

command name	application
`printenv`	prints the list of environment variables
`echo $<name_of_variable>`	will print the value of variable
`variable=<value>`	will create a new shell variable
`export <shell_variable_name>`	will create a new environment variable from the shell variable.
`export variable=<value>`	will create a new environment variable
`alias <custom_name>='<standard_command>'`	will create a new alias as `custom_name` for the `standard_command` passed

01. What actually matters in this section?

- `important`: understanding environment variables, parameter expansion, defining aliases
- `useful`: startup files like .profile, .bash_profile, .bashrc,
- `nice to have`: remembering the specific environment variables

02. Introducing The Environment and variables

- `environment`: The shell maintains a set of information during a shell session, known as the environment. It's a series of key-value pairs that define properties like:
    1. your home directory
    2. your working directory
    3. The name of your shell
    4. The name of the logged-in user
- `printenv`: prints the list of environment variables
- environment variables values changes from session to session and user to user

03. Parameter expansion

- `echo $<name_of_variable>`: will print the value from the environment variable
- all the variable names are case sensitive
- example: `echo $USER`: will print `pank` since that is my username

04. Defining Variables and Export

shell variables

- shell variables exists on particular shell session only.
- `variable=<value>`: will create a new shell variable
- Built-in variables are upper-cased, so it's a common convention to lowercase custom variables to prevent confusion.
- example:
    1. `color="purple"`
    2. `num=56`
- `export <variable_name>`: will create a new environment variable from the shell variable.

environment variables

- environment variables are global variables for the shell.
- `export variable=<value>`: will create a new environment variable
- example:
    1. `export color="purple"`
    2. `export num=45`

05. The mysterious startup files

- when we log in, the shell reads information from startup files. First, the shell reads from global config files that affect the environment for all users. Then, the shell reads startup files for specific users.
- The specific files the shell reads from depends on the type of session: login vs non-login shell sessions

Startup files for login sessions:

- A login session is the period of activity between a user logging in and logging out of a (multi-user) system.
- `/etc/profile`: global config for all users
- `~/.bash_profile`: user's personal config file
- `~/.bash_login` : read if `bash_profile` isn't found
- `~/.profile` : used if previous two aren't found

Startup files for non-login sessions:

- For non-login sessions (typical session when you launch the terminal via the GUI):
  - `etc/bash.bashrc`: global config for all users
  - `~/.bashrc`: specific settings for each user. This is where we can define our own settings and configuration

06. Customizing your prompts

- we can tweak things on the terminal while starting up
- `man bash` has its own documentation regarding prompt customization
- we can go to a website like ezpromt which gives a good option to edit `PS1` to edit the configuration for better quality

07. Defining Aliases

- We can define our own commands using `alias` keyword
- example:
  1. `alias ll='ls -al'`: will make keyword `ll` to execute the command `ls -al`

- we can define an alias by editing the `.bashrc` file. command: `nano ~/.bashrc`

08. Useful Aliases & The bash_aliases File

- we can look for useful aliases on cheat sheets or any other online resources
- we can also put all our additions of commands into a separate file like `~/.bash_aliases` instead of adding on `~/.bashrc`
- always use comments while writing your own aliases so that they can be better understandable