11. Redirection

commands table

command	application
`command > file_directory`	redirect output from command to other file instead of printing to screen
`command >> file_directory`	append output from command to other file instead of printing to screen
`cat < <filename>`	Input redirecting from file
`cat < <file1> > <file2>`	to redirect input from file1 and redirect output to `file2`
`cat < <file1> >> <file2>`	to redirect input from file1 and append output to `file2`
`cat <commands> 2> error.txt`	to redirect a cat error to a file instead of prompting to the screen
`cat <commands> 2>> error.txt`	to append a cat error to a file instead of prompting to the screen

01. What actually matters in this section

- `important`
    1. Redirecting Standard Output
    2. Redirecting Standard Input
    3. Redirecting Standard Error

02. Introducing Standard Streams

- The 3 standard streams are communication channels between a computer program and it's environment. They are:
    1. standard input
    2. standard output
    3. standard error
- standard input is where a program or command gets it's input information from. By default, the shell directs standard input from keyboard. The input could come from a keyboard, a file, or even from another command.
- standard output is a place to which a program or command can send information. The information could go to a screen to be displayed, to a file, or even to a printer or other devices
- standard error is a place where commands and programs have destination to send error messages

03. Redirecting Standard Output

- The redirect output symbol `>` tells the shell to redirect the output of a command to a specific file instead of the screen.
- we use redirection after completing writing command. i.e `command > file_directory`

- By default, the command `date` will print the current date to the screen. If we instead run `date> output.txt` the output will be redirected to a file called `output.txt`
- if we redirect command again into the same file then it will re-write on the file and all previous contents will be removed.

04. Appending standard output

- we can `>>` in order to append outputs to the file. with this pre-existing file won't be overwritten

05. Redirecting Standard Input

- Input redirecting is the process where contents of file is passed to standard input
- we use `<` symbol
- example: `cat < <filename>`

06. Redirecting Stdin and Stdout together

- We can redirect standard input and output at the same time
- example:

  `cat < <file1> > <file2>` to redirect input from file1 and redirect output to `file2`

  `cat < <file1> >> <file2>` to redirect input from file1 and append output to `file2`

07. Redirecting Standard Error

- By default, error message are output to screen, but we can change this by redirecting standard error.

- The standard redirect operation is `2>`
- example:
    1. to redirect a cat error to a file instead of prompting to the screen

       `cat <non_existing_file> 2> error.txt`
    2. to append a cat error to a file instead of prompting to the screen

       `cat <non_existing_file> 2>> error.txt`

08. Putting It All together and fancy shortcuts

- we can chain input, output and error all commands together. Standard output comes first, input and then error
- example: `cat bees.txt ants.txt > insects.txt 2> error.txt`

fancy shortcuts

- if we need to redirect output and error to same file then we can use `2>&1`
- example: `ls docs > output.txt 2>&1`
- modern bash also support another fancy option to do it. which is `&>`
- example: `ls docs &> output.txt` will keep output and error both in `output.txt` file.

09. Assignments

- went well throughout