SOFE 3200U: Systems Programming

Tutorial 2

Fall 2023

Somayya Elmoghazy

Slides acknowledged to: Abdelrahman Elewah, Teaching Assitant, Faculty of Engineering and Applied Science

Agenda

- Downloading This Tutorial Using wget Command
- Shell Plumbing
- Pipes
- Redirects
- Basic Commands
- makefile
- Activity

Downloading This Tutorial Using wget Command

Downloading This Tutorial using wget Command

1. wget command : wget http://ericdube.com/sofe3200/2/INDEX.md
2. Github is software development and version control
3. Markdown language

Good online course to learn Git and GitHub :

-https://www.udacity.com/course/version-control-with-git--ud123

-https://www.udacity.com/course/version-control-with-git--ud456

wget https://github.com/elewah/elewah.github.io/blob/master/courses/system%20programming/Tutorial2.pdf

Shell Plumbing

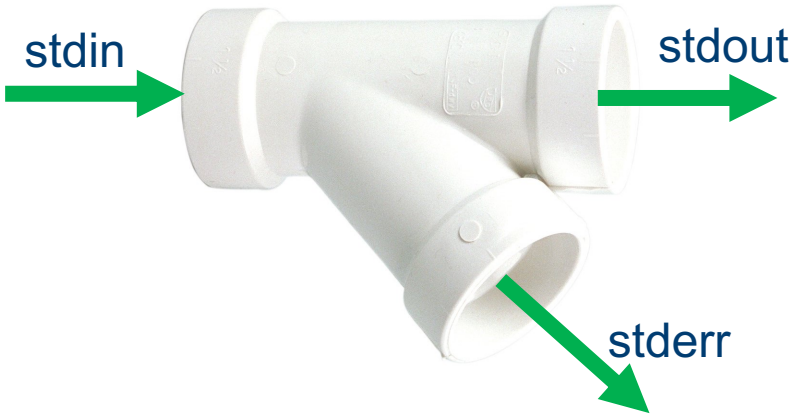
Shell Plumbing

A **shell** is a **CLI** (**Command-Line Interface**)

- **Streams**

When you run any program in the shell, it will have access to three streams: -

1. One input stream:stdin (standard in)
2. Two output streams: stdout and stderr (standard out and standard error)



Shell Plumbing

- **Why two output streams?**
  - It is important to distinguish between valid output of a command and output due to errors. Like a function, you can make some assumptions about the output of a command. If errors were mixed in with the output, you wouldn't be able to make these assumptions.
    - For example, the `ls` command will always write a list of files (or nothing) to `stdout`. Any error message is sent over `stderr`.
  - **For instance try this command**
    - **`ls /root /var (stdout)`**

Redirects

Redirects

- The output of a command can also be directed to a file (or stream) using redirection.
- Using **<** will redirect a file's contents to **stdin**
- **while** using **>** will redirect **stdout** and write it to a file.

```
echo "Hello, world!" > helloworld.txt
printf "This is cool! \n This is awesome!" > myfile.txt
grep "a" < myfile.txt
```

Pipes

Pipes

- Pipes are a very powerful tool which allow you to create your own data flow in the terminal. A pipe connects two streams together.
- The **piped commands run concurrently**. This can be very helpful for basic multiprocessing.
- For example, running the following will display all files in the working directory which contain the letter "a".
  - `ls | grep "a"`

Basic Commands

Basic Commands

- **man**
  - `man` will display usage instructions for a command. For example `man man` displays the usage of man.
- **pwd and ls**
  - `pwd` stands for Print Working Directory. It displays the current working directory. This command normally doesn't take any parameters.
  - `ls [OPTION]... {FILE]...` lists the contents of the working directory, or a directory passed as an argument. This command also has flags to change the behaviour, which you can see by running `ls--help` or `man ls`.
- **date**
  - `date` is a command which can print the date and time. A format can be passed preceded by a plus symbol. For example, `date +%I:%M:%S\ %p` displays the time in this format: 9:00:00 PM

Basic Commands

- **cat**
  - `cat` is a command intended for concatenating files, but there are many useful applications of cat. This command can take input from multiple files and the input stream, and will write all of these to the output stream.
- **examples:**
  - `cat file1.txt file2.txt` - Display the contents of two files
  - `cat file1.txt file2.txt > file1and2.txt` - Concatenate files
  - `echo "hello" | cat file1.txt -` - Append "hello" to a file
- **more**
  - `more` is a useful command to display an input one screen at a time. For example: `ls -R / | more` displays all files, one screen at a time.

Basic Commands

- **head and tail**
  - `head` and `tail` display the first lines and last lines of a file respectively. By default, they display the last 10 lines, but passing the `-n` flag (ex: `head -n 5`) can override this behaviour. `tail` also has a flag `-f` to follow the end of a file. In this mode, `tail` will stay open and display every new line added to a file
- **mkdir, rmdir, cd**
  - `mkdir` and `rmdir` provide creation and deletion of directories respectively. Note that `rmdir` requires the directory to be empty for removal.
  - `cd [PATH]` changes the current directory. With no parameters, `cd` will set the working directory to the user's home directory.
- **mv, cp**
  - `mv` is the rename command. It takes two arguments: source and destination.

Basic Commands

- **wc**
  - `wc` is a useful command for counting words, lines, or characters. It can receive input from stdin, or a filename as an argument. There are three flags available: `l`, `w`, and `c`; for lines, words, and characters respectively. For example, to display the number of lines and words in some input text, try `printf "Hello\nThis is text" | wc`.
  - **Note:** Even if the flags are passed in a different order, wc will always display lines, words, then characters.
- **sort**
  - `sort` will sort input from stdin, or from a filename accepted as a paramters.

Activity