

Week2

This section is inspired by [Welcome to Linux command line for you and me!](#) and [The Missing Semester of Your CS Education](#).

Disclaimer: This week's source code should not be used in any real project, as it does not follow the proper use of the C language.

Before we start

1. What is a pointer?
 - Size of data type
 - Why is it useful?

A rating system for C-programmers. The more indirect your pointers are (i.e. the more "*" before your variables), the higher your reputation will be. No-star C-programmers are virtually non-existent, as virtually all non-trivial programs require use of pointers. Most are one-star programmers. In the old times (well, I'm young, so these look like old times to me at least), one would occasionally find a piece of code done by a three-star programmer and shiver with awe.

Some people even claimed they'd seen three-star code with function pointers involved, on more than one level of indirection. Sounded as real as UFOs to me.

Just to be clear: Being called a ThreeStarProgrammer is usually not a compliment. [Three Star Programmer](#)

2. What is a function pointer?
 - How to define a type that stores a function pointer?
 - How to assign a function pointer to variable of that type?

Processes in Linux

1. What is a process?
 - Process ID (PID)
2. How to find a process?
 - Created by current shell
 - View all the processes
 - Select One from the list
3. How to kill a process
 - `kill` command
 - What's the meaning of `-9` ?
 - Signal Handler
4. `lsuf` and its options
 - How to open a file in C?

- Different modes
 - Inside `FILE` struct
 - File Descriptors
 - Finally, `lsof`
5. `top` command (`uptime`)
- How to use `?`
 - What is Load Average?
6. How to create a process?
- How programs are started by shells?
 - Unix-like API and Windows API

```
// Windows
bool CreateProcessA(
    [in, optional] LPCSTR          lpApplicationName,
    [in, out, optional] LPSTR      lpCommandLine,
    [in, optional] LPSECURITY_ATTRIBUTES lpProcessAttributes,
    [in, optional] LPSECURITY_ATTRIBUTES lpThreadAttributes,
    [in]           BOOL              bInheritHandles,
    [in]           DWORD             dwCreationFlags,
    [in, optional] LPVOID            lpEnvironment,
    [in, optional] LPCSTR            lpCurrentDirectory,
    [in]           LPSTARTUPINFOA    lpStartupInfo,
    [out]           LPPROCESS_INFORMATION lpProcessInformation
);

// Unix-like
pid_t fork(void);
// and exec family
```

- Unix-like API vs. Windows API, and their design philosophy (quoted from [CS 110L: Safety in Systems Programming](#))

Data Wrangling

1. `grep`
 - Print lines Before/After current lines
 - Basic Regular Expressions
2. `sort`
 - Different sort options
3. `uniq`
 - Deal with unique/duplicate elements
4. `head`
 - Extract the starting n lines
5. `tail`
 - Extract the ending n lines
6. `rev`
 - Reverse each line
7. `tr`

- Replace characters
- 8. **cut**
 - Select the starting n characters
 - Select any character in between
 - Select based on field
- 9. **awk**
 - A programming language for working on files
 - [What does \\$0=\\$2 in awk do? learn awk](#)
 - Equality
- 10. **sed**
 - A stream editor that is used to perform basic text transformations on an input stream
 - [The Basics of Using the Sed Stream Editor to Manipulate Text in Linux](#)
 - [Sed Command to Delete a Line](#)

Command-line environment

1. Jobs
 - See [Pausing and backgrounding processes](#)
 - Plus and Minus Sign: [What do those +/- mean if linux job in background finishes \(started with &\)](#)
2. Dotfiles
3. Aliases
 - How to set an alias?
 - How to unset an alias?
4. Command Substitution
 - Why we need it?
 - How to use it?
5. Globbing

Random Stuff

1. **which**
2. Differences between **.bashrc** and **.bash_profile**
 - Login Shell vs. Non-Login Shell
 - Read order of the configuration file
 - Read [Difference between Login Shell and Non-Login Shell?](#) for more details
3. she-bang
 - What is a scripting language?
4. **less**
 - Run command: Save content to files
5. **time** command

Exercises

1. Count the number of lines in the file **/usr/share/dict/words**
2. [Leetcode: Word Frequency](#)
3. Use only the tools from 1-8 in Data Wrangling session to find the first version number in awk example. (See **./others/q3.txt**)
4. Find the number of words (in **/usr/share/dict/words**) that contain two consecutive as. What are the three most common last two letters of those words? How many of those two-letter combinations are there?

5. Find the number of words (in [/usr/share/dict/words](#)) that contain at least three as and don't have a 's ending. What are the three most common last two letters of those words? How many of those two-letter combinations are there?
6. [Leetcode: Tenth Line](#)

License

MIT License

Copyright (c) 2022 Yuxiang Qiu

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.