C Programming II
2024 Spring
Homework 03

Instructor: Po-Wen Chi

Due: 2024.05.08 PM 11:59

Policies:

- **Zero tolerance** for late submission.
- **Plagiarism is not allowed.** Both source and copycat will be **zero**.
- You need to prepare a README file about how to make and run your program. Moreover, you need to provide your name and your student ID in the README file.
  - Your Name and Your ID.
  - The functional description for each code.
  - Anything special.
- Please pack all your submissions in one zip file.
- For convenience, your executable programs must be named following the rule hwXXYY, where the red part is the homework number and the blue part is the problem number. For example, hw0102 is the executable program for homework #1 problem 2.
- I only accept **PDF**. MS Word is not allowed.
- Do not forget your Makefile. For convenience, each assignment needs only one Makefile.

1 Bible (20 pts)

The Bible is a collection of religious texts, writings, or scriptures sacred in Christianity. It is is widely considered to be the best-selling book of all time. This time, I want you to develop a search function for the English Bible. I provide you a text file, **bible.txt**, which contains all verses. The searching process should be case insensitive. Your program should work as follows:

```
$ ./hw0301
Please enter the search target: in the beginning
Found 1 time(s)
1. Gen 1:1 In the beginning God created the heavens and the earth.
```

Note that the above is just an example and I do not guarantee the search correctness.

2 Function Tracer (20 pts)

When we develop a large program, undoubtedly there will be lots of functions. This time, I want you to develop a program to trace the function usage in source codes.

```
$ ./hw0302 --help
Usage: hw0302 [options] ... [files] ...
  -f, --function=func   Trace the func usage in [Files].
  -i, --include=File    Trace all functions listed in the header file in [
    Files].
  -l, --linum       Display the line number.
  -c, --code        Display the code.
  -h, --help        Display this information and exit.

-F and -I are exclusive and must be at least one.
```

Let’s use **mystring.h** and **hw0101.c** for example.

```
$ ./hw0302 -i mystring.h hw0101.c
mystrchr:
  hw0101.c (count: 2)
mystrrchr:
  hw0101.c (count: 1)
...
$ ./hw0302 -i mystring.h -l -c hw0101.c
mystrchr:
  hw0101.c (count: 2)
    line 14:   char *my_p = mystrchr(s, c);
    line 34:   char *my_p = mystrchr(s, c);
mystrrchr:
  hw0101.c (count: 1)
    line 54:   char *my_p = mystrrchr(s, c);
...
```

For your simplicity, I promise that the input header file and c codes are all valid. Note that if the function name is a string, it should not be counted.

3 Image Steganography (20 pts)

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. In this problem, I want you to implement a simple image data hiding program.

The concept is simple. Given a BMP image file, where R, G, B are presented in 8 bits. To embed a secret data in the cover BMP file, you just replace the last significant bits with

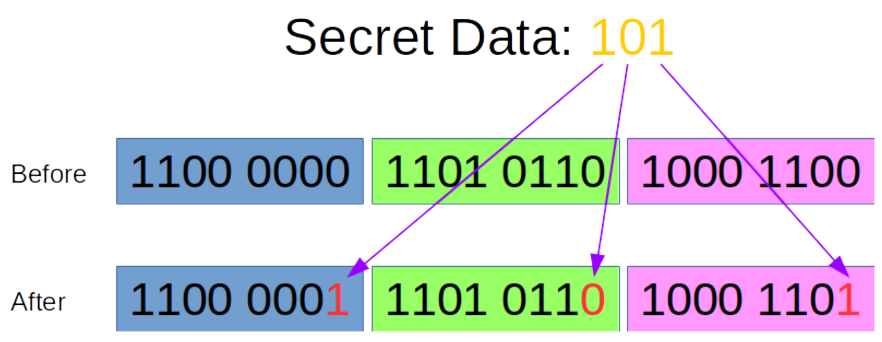


Figure 1: BMP LSB Steganography.

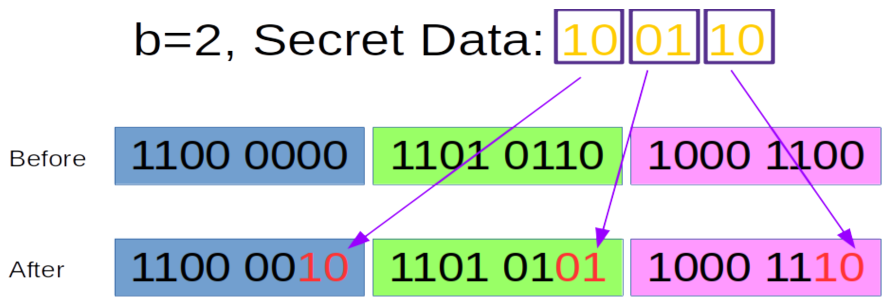


Figure 2: BMP Last Two LSBs Steganography.

the secret data bit, as shown in Fig. 1. If you want more capacity, you can use the last two significant bits, as shown in Fig. 2. Since we do not want to affect the original image file too much, the embedding order is from LSB to MSB.

Now I want you to embed and extract a secret data in/from the cover bmp image file.

```
./hw0303 [option] [cover_bmp] [secret_data]
 -w, --write: Write the secret data to the cover_bmp.
 -e, --extract: Extract the secret data from the cover_bmp to the secret_data.
 -b, --bits=N: use last N bits. N is from 1 to 8. The default N is 1.
```

There are some notes here.

1. The secret data size should be recorded in the BMP header reserved field which is 4 bytes long.
2. If the secret data is larger than the cover BMP's hiding capacity, print an error message.
3. For your simplicity, I guarantee that the DPP is 24-bits.

4 Game Cheater (20 pts)

炎龍騎士團 II is one of my favorite DOS game. Fig. 3 is its playing screen and you can get its information from the following link.

https://zh.wikipedia.org/wiki/%E7%82%8E%E9%BE%8D%E9%A8%8E%E5%A3%AB%E5%9C%98II_%E9%BB%83%E9%87%91%E5%9F%8E%E4%B9%8B%E8%AC%8E

Figure 3: The screen of 炎龍騎士團 II.

Unfortunately, I am a bad game player. So, I want you to develop a game cheater for me. I will show you how to do this. First, you need to play the game

1. Install **dosbox-staging**[1] on your computer. This is an emulation tool that can help you to play DOS games.
2. Download the game from the following link:
   `https://legacy.dos.zczc.cz/games/%E7%82%8E%E9%BE%99%E9%AA%91%E5%A3%AB%E5%`
   `9B%A22%E9%BB%84%E9%87%91%E5%9F%8E%E4%B9%8B%E8%B0%9C/download`
3. Run the game from **dosbox-staging**. Do not worry, I will show you how in my Youtube video.

Now let's see what I want you to do. I want you to develop a program to **modify** the game process memory directly. As you guess, all characters' abilities are variables stored in the memory. That is, if I change the memory value, I can modify the character ability directly. This is cool, right? Do not worry, I will show you how step by step.

1. Run **dosbox-staging** in the terminal and you will get **the memory address it allocated for the game**, as shown in Fig. 4a.
2. Get **dosbox-staging** process ID from the command **ps**, as shown in Fig. 4b.
3. Open the memory file of the given pid. The proc filesystem is a pseudo-filesystem which provides an interface to kernel data structures. This mem file can be used to

[1]Not dosbox!!

access the pages of a process's memory through open(2), read(2), and lseek(2). Note that you must have root privilege for accessing this file. So TA will use **sudo** to run your cheater.

Now you need to find the character abilities from the memory and design an interface for the user to use your cheater. This time, there will be **NO** interface specification. **Please design the interface yourself**. Undoubtedly, you need to provide a user manual for TAs to test your program. The program should be named **hw0304**. Your cheating program must support the following functions:

- Modify abilities: HP, MP, MT, DF, MV, EX, DX
- User items.

For your simplicity, I will give you some notes.
https://chiuinan.github.io/game/game/intro/ch/c31/fd2/

5 RGB Line Chart (20 pts)

When analyzing images, we typically use histograms to examine color distribution. In Figure 5, the x-axis represents RGB color values ranging from 0 to 255, while the y-axis indicates the number of pixels corresponding to each RGB value. By histograms, we could easy to analyze the image exposure and adjust the image color. However, drawing histograms can be too simplistic, requiring three separate figures to convey information.

Today, I'd like you to generate an RGB Line Chart similar to Figure 6 using a given BMP file. You'll need to specify the figure's height, width, file name, and line width.

If you're unsure how to map the data to pixels in the figure, interpolation methods can be employed. The program usage should be:

```
./main -i maldives.bmp -o output.bmp -w 1280 -h 864 -l 3
./main -H
```

The result will Figure.6 The options are

- **-i, - -input, mandatory:** input file path
- **-o, - -output, mandatory:** output file path
- **-w, - -width, mandatory:** output bmp file widths
- **-h, - -height, mandatory:** output bmp file height
- **-l, - -line, mandatory:** the radius of line
- **-H, - -help, option:** show help message

Here are some additional require:

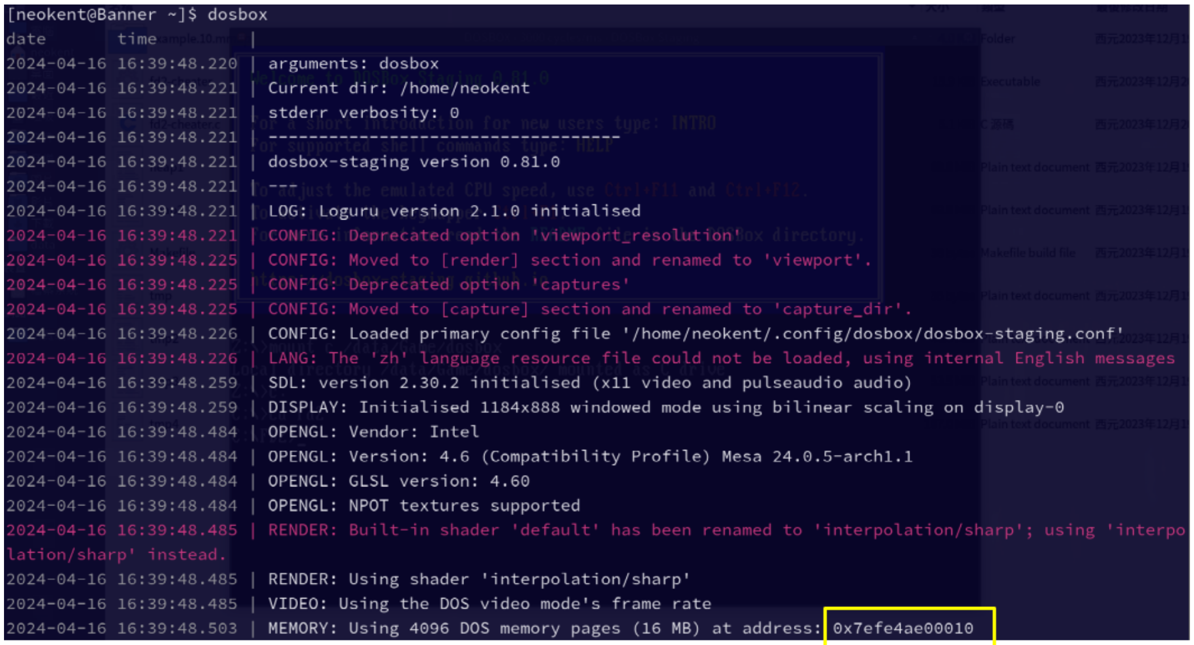
1. The maximum number should in the top of figure.

2. The line need to be continuous.
3. Short and long option should be support.
4. When the link cross over, you need to show the overlay color.
5. The radius of line means the width of line. If -l is 3, the width of line is 5
6. The line need Gradient from middle of line.

6 Bonus: ncurses (5 pts)

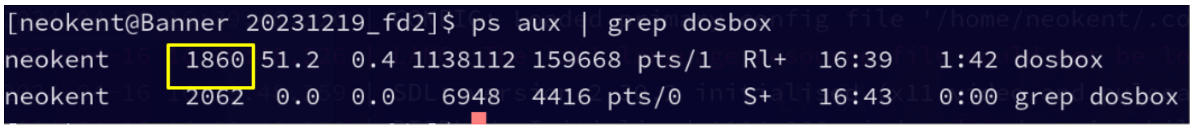
It is boring to write a console program, right. I totally understand. So I want to show you some interesting library: **ncurses**. ncurses (new curses) is a programming library providing an application programming interface (API) that allows the programmer to write text-based user interfaces (TUI) in a terminal-independent manner. It is a toolkit for developing ”GUI-like” application software that runs under a terminal emulator. It also optimizes screen changes, in order to reduce the latency experienced when using remote shells. Fig. 7 is an example.

Now I want you to write a tutorial about how to make an interface like Fig. 7. For your simplicity, you **do not** need to provide your code so that TAs will not check your code. I assign this problem to help you be familiar with the tool you may use in your final project.



```
[neokent@Banner ~]$ dosbox
date       time            |
2024-04-16 16:39:48.220 | arguments: dosbox
2024-04-16 16:39:48.221 | Current dir: /home/neokent
2024-04-16 16:39:48.221 | stderr verbosity: 0
2024-04-16 16:39:48.221 |
2024-04-16 16:39:48.221 | dosbox-staging version 0.81.0
2024-04-16 16:39:48.221 | ---
2024-04-16 16:39:48.221 | LOG: Loguru version 2.1.0 initialised
2024-04-16 16:39:48.221 | CONFIG: Deprecated option 'viewport_resolution'
2024-04-16 16:39:48.225 | CONFIG: Moved to [render] section and renamed to 'viewport'.
2024-04-16 16:39:48.225 | CONFIG: Deprecated option 'captures'
2024-04-16 16:39:48.225 | CONFIG: Moved to [capture] section and renamed to 'capture_dir'.
2024-04-16 16:39:48.226 | CONFIG: Loaded primary config file '/home/neokent/.config/dosbox/dosbox-staging.conf'
2024-04-16 16:39:48.226 | LANG: The 'zh' language resource file could not be loaded, using internal English messages
2024-04-16 16:39:48.259 | SDL: version 2.30.2 initialised (x11 video and pulseaudio audio)
2024-04-16 16:39:48.259 | DISPLAY: Initialised 1184x888 windowed mode using bilinear scaling on display-0
2024-04-16 16:39:48.484 | OPENGL: Vendor: Intel
2024-04-16 16:39:48.484 | OPENGL: Version: 4.6 (Compatibility Profile) Mesa 24.0.5-arch1.1
2024-04-16 16:39:48.484 | OPENGL: GLSL version: 4.60
2024-04-16 16:39:48.484 | OPENGL: NPOT textures supported
2024-04-16 16:39:48.485 | RENDER: Built-in shader 'default' has been renamed to 'interpolation/sharp'; using 'interpo
lation/sharp' instead.
2024-04-16 16:39:48.485 | RENDER: Using shader 'interpolation/sharp'
2024-04-16 16:39:48.485 | VIDEO: Using the DOS video mode's frame rate
2024-04-16 16:39:48.503 | MEMORY: Using 4096 DOS memory pages (16 MB) at address: 0x7efe4ae00010
```

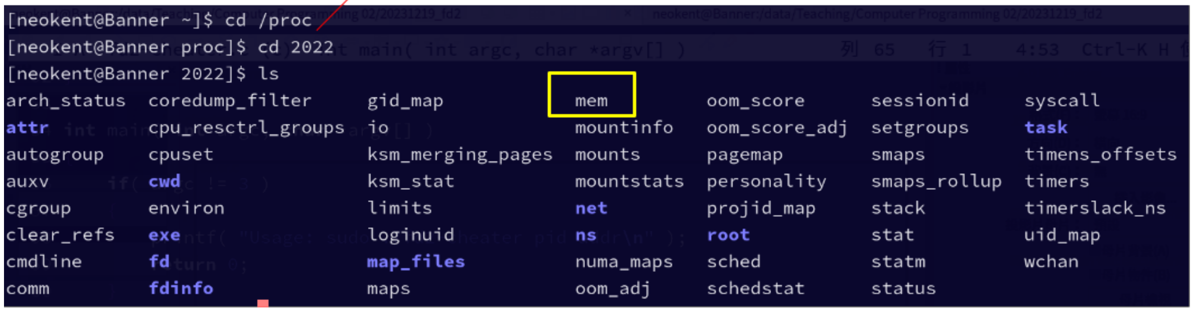
(a) Game memory address.



```
[neokent@Banner 20231219_fd2]$ ps aux | grep dosbox
neokent    1860 51.2  0.4 1138112 159668 pts/1  Rl+  16:39   1:42 dosbox
neokent    2062  0.0  0.0   6948  4416 pts/0   S+   16:43   0:00 grep dosbox
```

(b) Dosbox-staging PID.

這邊應該是 1820，但因為我遊戲關了，所以 PID 消失了



```
[neokent@Banner ~]$ cd /proc
[neokent@Banner proc]$ cd 2022
[neokent@Banner 2022]$ ls
arch_status   coredump_filter    gid_map           mem          oom_score      sessionid     syscall
attr          cpu_resctrl_groups io                mountinfo    oom_score_adj  setgroups     task
autogroup     cpuset             ksm_merging_pages mounts       pagemap        smaps         timens_offsets
auxv          cwd                ksm_stat          mountstats   personality    smaps_rollup  timers
cgroup        environ            limits            net          projid_map     stack         timerslack_ns
clear_refs    exe                loginuid          ns           root           stat          uid_map
cmdline       fd                 map_files         numa_maps    sched          statm         wchan
comm          fdinfo             maps              oom_adj      schedstat      status
```

(c) /proc/pid/mem.

Figure 4: Some important values.

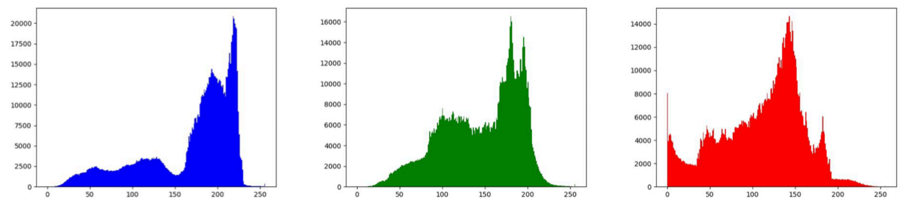


Figure 5: The RGB histogram

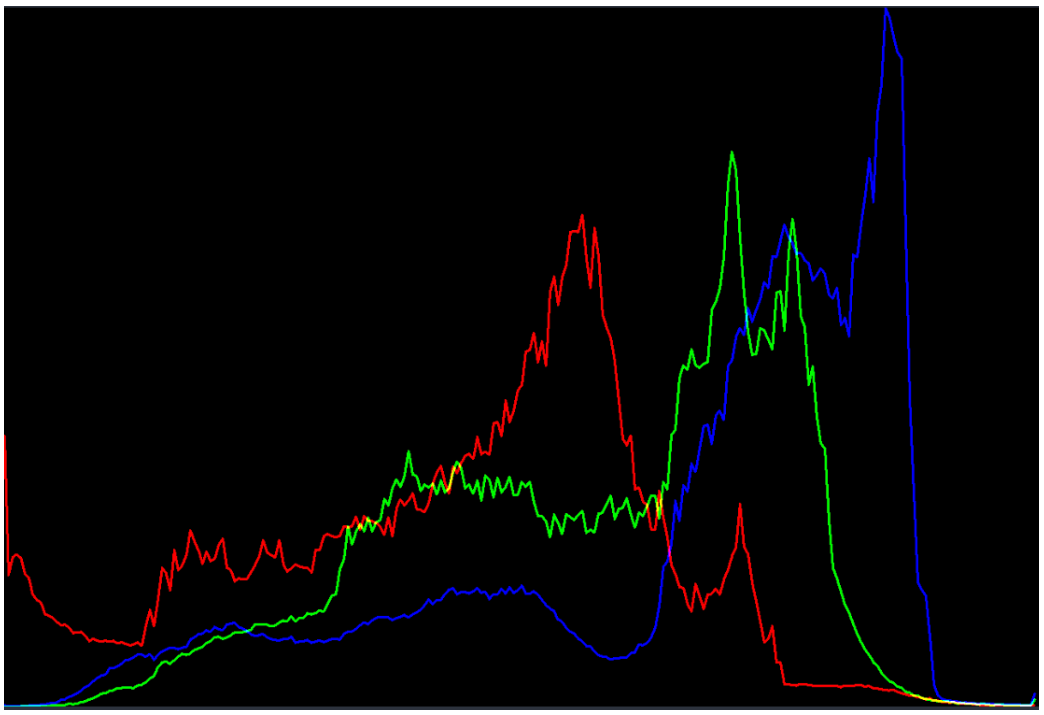
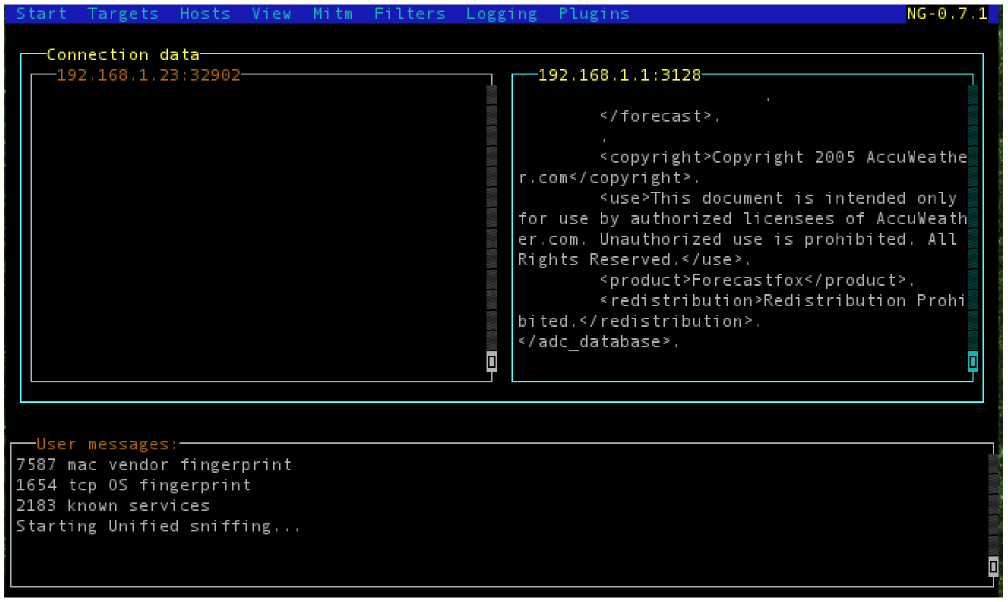


Figure 6: The RGB line chart



```
        </forecast>.
        .
        <copyright>Copyright 2005 AccuWeathe
r.com</copyright>.
        <use>This document is intended only
for use by authorized licensees of AccuWeath
er.com. Unauthorized use is prohibited. All
Rights Reserved.</use>.
        <product>Forecastfox</product>.
        <redistribution>Redistribution Prohi
bited.</redistribution>.
</adc_database>.
```

```
7587 mac vendor fingerprint
1654 tcp OS fingerprint
2183 known services
Starting Unified sniffing...
```

Figure 7: Ncurses example.