CP2023-I HW05

Hsin-Jui Lin

November 13, 2023

1 TAS Editor (25 pts)

TAS (Tool-assisted speedrun)link generally defined as a speedrun or playthrough composed of precise inputs recorded with tools such as video game emulators. e.g. Super Mario Bros and Pokemon. It is commonly employed to achieve records in various categories like any%link, 100%link, glitch end runlink and arbitrary code executionlink. TAS can do the console actions that would be impossible for a human to perform in reality.

Your TA was very love to play DS and Wii which are Nintendo video game consoles in his childhood. One of his favorite series game is Super Mario BrosFig5.1. He repeatedly completing the game 100%, even attempting speedrun on it. Of course, he sometimes watches something cool video about TAS speedrun. And he is also curious about the principles behind it. He finds out that TAS control based on reading the contents inside files.

Figure 1: Super Mario Bros game

So this time, I want you implement some functions about TAS edit action.

For your convenient, it is not necessary to test TAS on DS or Wii platform which are require high performance. You are required to use your program to generate a file named with the extension fm2link for testing on the NESlink platform, with the classic game being Super Mario Bros. TA will prepare tas.h for you. There is the tas.h is as:

```c
#pragma once

#include <stdint.h>
#include <stdbool.h>

/*
    Set NES button within fm2 file content by frame range.

    Parameters:
        src: The source from fm2 file content.
        size: src size for pointer.
        button: Button string to set. (Order doesn't matter)
        start_frame: Represent start frame to set button.
        end_frame: Represent end frame to set button.
        unset: If true, clear button but not delete it. Otherwise,
    as above description.

    Examples:
        Set R and B button from frame 381 to frame 480.

        $ button_set_frame(src, size, "RB", 381, 480, 0);

    Note:
        If the frame range is overflow, please increase the size of
    src automatically
*/
void button_set_frame(char ***src, size_t *size, const char *button,
     const uint64_t start_frame, const uint64_t end_frame, const
    bool unset);

/*
    Set NES button within fm2 file content by second range.

    Parameters:
        src: The source from fm2 file content.
        size: src size for pointer.
        button: Button string to set. (Order doesn't matter)
        start_sec: Represent start second to set button.
        end_sec: Represent end second to set button.
        unset: If true, clear button but not delete it. Otherwise,
    as above description.

    Examples:
        Set L, B, and A button from 5.5 seconds to 15.232 seconds.

        $ button_set_second(src, size, "LBA", 5.5, 15.232, 0);

    Note:
        If the second range is overflow, please increase the size of
     src automatically

*/
void button_set_second(char ***src, size_t *size, const char *button
    , const double start_sec, const double end_sec, const bool unset
    );

/*
    Set subtitle within fm2 file content by specific frame.

    Parameters:
        src: The source from fm2 file content.
        size: src size for pointer.
        subtitle: Subtitle string to set.
        frame: Set subtitle at specific frame.

    Examples:
        Set "Start Speed Run!!" subtitle string at frame 0.

        $ button_set_frame(src, size, "Start Speed Run!!", 0);
*/
void subtitle_set_frame(char ***src, size_t *size, const char *
    subtitle, const uint64_t frame);

/*
    Set subtitle within fm2 file content by specific second.

    Parameters:
        src: The source from fm2 file content.
        size: src size for pointer.
        subtitle: Subtitle string to set.
        second: Set subtitle at specific second.

    Examples:
        Set "1-1 clear!!" subtitle string at 55.413 seconds.

        $ button_set_frame(src, size, "1-1 clear!!", 55.413);
*/
void subtitle_set_second(char ***src, size_t *size, const char *
    subtitle, const double second);

/*
    Read from fm2 file source.

    Parameters:
        src: The pointer to store fm2 file content.
        size: src size for pointer.

    Note:
        No matter the origin of src, src will be replaced by new fm2
     file source.
*/
void tas_read(char*** src, size_t *size);

/*
    Save fm2 file source

    Parameters:
        src: The source from fm2 file content.
        size: src size for pointer.
*/
void tas_save(const char** src, const size_t size);

/*
    Set 1-1 map clear step fm2 content to src

    Parameters:
        src: The source from fm2 file content.
        size: src size for pointer.

    Note:
        - This function should be implement by the other function.
        - No matter the origin of src, src will be replaced by demo
    content.
*/
void demo(char*** src, size_t *size);
```

I'll use the following commands when running your code:

- If generating fm2 file:

```
$ ./hw0505 > output.fm2
```

- If editing existing fm2 file:

```
$ ./hw0505 < input.fm2 > output.fm2
```

So you should know how to implement tas_read and tas_write function right? I'll use a NES emulator called FCEUX_{Fig5.2 link} while running your fm2 file.

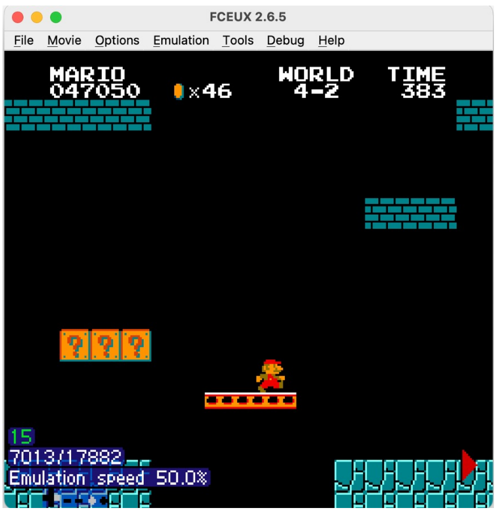

Figure 2: FCEUX on my macbook

This is how to run it:

```
$ fceux --playmov example.fm2 SuperMarioBros.nes
```

There are some FCEUX shortcuts:

- NES Gamepad (keyboard map to gamepad button):
  - D: B
  - F: A
  - Enter: T (Start)
  - S : S (Select)
  - Keypad up: U (Up)
  - Keypad left: L (Left)
  - Keypad down: D (Down)
  - Keypad right: R (Right)
- Show Frame Count: .
- Speed Up: +
- Speed Down: -
- Enter or Quit Fullscreen:
  - Alt + Enter (For Ubuntu or Windows)
  - Option + Enter (For macOS)

Note: In fm2 file, the button order will be "R, L, D, U, T, S, B, A" to set. To install it, you can run the following command:

```
$ sudo apt install fceux #Ubuntu
$ sudo brew install fceux #macOS
```

If you still have a little confusion on it. Don't worry. TA will prepares something like these:

- tas.h (as your header file to include)
- example.fm2 (an example fm2 file as for your reference)
- SuperMarioBros.nes (NES game rom file for emulator to read)

You should also prepare a header file called tas.h. TA will prepare hw0505.c which includes tas.h and uses these functions. **Do not forget to make hw0505.c to hw0505 in your Makefile.**