VE482
Introduction to Operating Systems

LAB 5

October 28, 2021

Yihua Liu 518021910998

---

1 Layer programming

- The program can be divided into three layers, what are they?
  Kernel layer, logic layer, and interface layer.
- Split the program into files according to the defined layers.
  - Kernel layer: `algorithm.h`, `algorithm.c`, `forward_list.h`, `forward_list.c`
  - Logic layer: `h3.h`, `h3.c`
  - Interface layer: `interface.h`, `interface.c`, `cli.c`, `menui.c`
- Create the appropriate corresponding header files.
- If necessary rewrite functions such that no call is emitted from lower level functions to upper level functions.
- The initial program implements a command line interface, write a "Menu interface" which (i) welcomes the user, (ii) prompts him for some task to perform, and (iii) runs it. When a task is completed the user should (i) be informed if it was successful and then (ii) be displayed the menu. From the menu he should be able to exit the program.
- Write two `main` functions, one which will "dispatch" the work to another function which will run the command line user interface and a second one which will "dispatch" the work to the Menu user interface.

2 Libraries

In order to understand libraries we first recall a few basics on compilation.

- What are the four stages performed when compiling a file?
  Preprocessing, compilation, assembler, and linker.
- Briefly describe each of them [4].

- Preprocessing: removes comments from source code, expands macros, and expands header files.
- Compilation: takes in temporary files *.i, translates into assembly language, and checks syntax.
- Assembler: tasks *.s code and translates into low-level machine code.
- Linker: takes in *.o file, links functions, and generates executable files.

A library is a collection of functions, data types, constants, etc. which are put together. When compiling, the machine code corresponding to those elements is generated. Two types of libraries exist: static and dynamic. Explain the difference between the two.

Static libraries are locked into a program at compile time while dynamic (shared) libraries exist as separate files outside of the executable file, creating a combined work at runtime [1] [3].

Generating a static library is a simple process: collect several functions and pack them into an `ar` archive.

- Search more details on how to proceed.
- Create two static libraries, one for each of the two lowest layers in the previous program.

```
add_library(kernel_static STATIC forward_list.c
↪  algorithm.c)
add_library(logic_static STATIC h3.c)
```

- Compile the command line version of the program using these two static libraries.

```
add_executable(cli_static cli.c interface.c)
target_link_libraries(cli_static logic_static
↪  kernel_static)
```

Generating shared, or dynamic, libraries is a slightly more complex process. Since the library is to be shared among various programs none of them can rely on a predefined location where to find the functions in the memory. Therefore as the library has to store its information at different memory addresses it is compiled into a Position-Independent Code (IPC). This is achieved by running `gcc` with the flag `-fpic`. Then in order to effectively create the dynamic library, `gcc` has to be re-run with the flag `-shared`.

- Generate two dynamic libraries, one for each of the two lowest layers in the previous program.

```
add_library(kernel_shared SHARED forward_list.c
↪  algorithm.c)
add_library(logic_shared SHARED h3.c)
target_link_libraries(logic_shared kernel_shared)
```

- Compile the whole program

```
add_executable(cli cli.c interface.c h3.c forward_list.c
↪  algorithm.c)
add_executable(menui menui.c interface.c h3.c
↪  forward_list.c algorithm.c)
```

- Compile the Menu version of the program using these two dynamic libraries.

```
add_executable(menui_shared menui.c interface.c)
target_link_libraries(menui_shared logic_shared
↪  kernel_shared)
```

A few extra remarks:

- What is the difference between a library and the API. Library is a collection of specific functions for some features, while API is used to interact with other applications, comprising all the definitions of publicly-declared classes, methods, and properties [2].
- Implement the API below for the two libraries.

References

[1] Erika Caoili. *Linux Basics: Static Libraries vs. Dynamic Libraries*. Dec. 17, 2019. URL: https://medium.com/swlh/linux-basics-static-libraries-vs-dynamic-libraries-a7bcf8157779.

[2] Robert Harvey. *What is the difference between an API and a library? [closed]*. Dec. 19, 2017. URL: https://softwareengineering.stackexchange.com/a/362696/380110.

[3] Stuart Kuredjian. *Static Libraries vs. Dynamic Libraries*. May 15, 2017. URL: https://medium.com/@StueyGK/static-libraries-vs-dynamic-libraries-af78f0b5f1e4.

[4] Daniela Vasquez. *The Four Stages of Compilation (C programming and gcc compiler)*. Feb. 8, 2019. URL: https://medium.com/@danielavasquez_77768/the-four-stages-of-compilation-c-programming-and-gcc-compiler-9faadf2e273a.