Lab 5

Xu Shengyuan

518370910200

1 Layer programming

- The program can be divided into three layers, what are they?
  - Top: User Command-Line Interface
  - Medium: Sorting Implemetation
  - Bottom: Linked List Structure
- File devision
  - `list.c/h`: Linked list structure/implementation
  - `sort.c/h`: Sorting implementation
  - `ui.c/h`: User interface
  - `cmd.c`: Command-line user interface
  - `menu.c`: Menu user interface

2 Libraries

- **What are the four stages performed when compiling a file? Briefly describe each of them.**
  1. The first stage is the **pre-processing stage**, which takes place before the formal compilation stage. The preprocessing phase modifies the contents of the source file based on the preprocessing instructions that have been placed in the file. The #include directive, for example, is a preprocessing directive that adds the contents of a file to a.cpp file. This way of modifying source files before compilation provides great flexibility to adapt to the limitations of different computer and operating system environments. The code required by one environment may differ from that required by another because the hardware or operating system available is different. In many cases, code for different environments can be placed in the same file and then modified during the preprocessing phase to fit the current environment.
  2. In the second stage of **compilation and optimization**, there are only constants in the output file obtained by pre-compilation. Such as the definition of numbers, strings, variables, and C keywords such as main,if,else,for,while,{,}, +,-,*,\ and so on. The compiler's job is to translate all the instructions into equivalent intermediate code representation or assembly code, after confirming that they conform to the syntax rules through lexical analysis and grammar analysis. Optimization is a difficult technique in compilation system. The problems it involves are not only related to compilation technology itself, but also to the hardware environment of the machine. The

optimization part is the optimization of the intermediate code. This optimization is not computer - specific. Another type of optimization focuses on object code generation.

3. **Assembly** actually refers to the process of translating assembly language code into target machine instructions. For every C language source program processed by the translation system, it will eventually get the corresponding object file through this process. The object file stores the target machine language code equivalent to the source program. The object file consists of segments. There are usually at least two segments in an object file:

   Code snippet: This snippet contains mainly program instructions. This paragraph is usually readable and executable, but generally not writable.

   Data segment: mainly stored in the program to use a variety of global variables or static data. Generally, data segments are readable, writable, and executable.

4. **Link process** The object file generated by the assembler is not immediately executable and may have many unresolved problems.

   For example, a function in one source file may refer to a symbol (such as a variable or function call) defined in another source file; You might call a function from a library file in your program, and so on. All of these problems need to be dealt with by the link program.

   The main job of the linker is to connect the relevant object files with each other, that is, to connect the symbol referenced in one file with the symbol defined in another file, so that all these object files become a unified whole that can be loaded and executed by the operating system.

- **Explain the difference between static and dynamic library**

  1. **Static linking**

     In this mode, the code for a function is copied from its local static link library into the final executable so that it is loaded into the process's virtual address space when it is executed A static linked library is actually a collection of object files, each of which contains the code for one or a group of related functions in the library.

     For example,

     ```
     gcc -c list.c sort.c
     ar crv liblist.a list.o
     ar crv libsort.a sort.o

     gcc -c cmd.c ui.c -o cmd.o
     gcc -static -o cmd cmd.o -llist -lsort
     ```

  2. **Dynamic linking**

     In this way, the code for a function is placed in an object file called a dynamic linked library or shared object Link the program at this time the only in the final executable programs to record the name of the Shared object and small amounts of other registration information In the executable file is executed, the entire content of the dynamic link library will be mapped to the runtime process virtual address space of the dynamic linker will according to record information in an executable program to find the corresponding function code.

     For example,

```
gcc -fPIC -shared -o liblist.so list.c
for list.o
gcc -fPIC -shared -o libsort.so sort.c
for sort.o

gcc -o menu menu.c ui.c -llist -lsort
```

- **Explain the difference between a library and the API**

  C library functions are basic functions supported by the C language itself, usually implemented directly in assembly.

  API functions are functions provided by the operating system for users to design application programs and achieve some specific functions. API functions are also implemented by C language functions.

  The difference is that THE API functions are specific to the operating system, while the C runtime functions are specific to THE C language itself

- **Implement the API below for the two libraries.**