HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY, VNU HCM
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEM

Lab 2: C Programming on Linux

Pham Minh Tuan - MSSV: 1752595

1 Exercises:

1.1 Questions:

1. Briefly explain each stage of the compilation process.

   - The compilation process includes 4 stages: Preprocessing, Compilation, Assembly, and Linking.
     - Preprocessing:
       * Lines starting with a # character are interpreted by the preprocessor as preprocessor commands.
       * Before interpreting commands, the preprocessor does some initial processing including joining continued lines and stripping comments.
       * The result of preprocessing stage is generated by command:

         gcc -E sourcefile.c
     - Compilation:
       * In this stage, the preprocessed code is **translated** into **assembly instructions** specific to the target processor architecture.
       * This step allows C code to contain inline assembly instructions and different assemblers to be used.
       * The result of compilation stage is generated by command:

         gcc -S sourcefile.c
       * After executing the above command, it will create the output file "sourcefile.s".
     - Assembly:
       * During this stage, an assembler is used to translated the assembly instructions to object code.
       * The output consists of actual instructions to be run by the target processor.
       * The result of the assembly stage is generated by command:

         gcc -c sourcefile.c
       * After running the above command will create a file name sourcefile.o containing the object code of program, the contents of this file is in binary format.
     - Linking:
       * The object code generated in compilation stage is composed of machine instructions that the processor understands, however some files of the program are missing. Those missing files needed to be rearranged and filled in → Linking stage.
       * In this stage, The linker will arrange the pieces of object code so that functions in some pieces can successfully call functions in other ones, the linker also add pieces containing the instructions for library functions used by the program.
       * The result of this stage is the final **executable program**.

         gcc -o main sourcefile.c

2. Illustrate the compilation process when we have multiple files:

   - % Preprocessed sourcefile
     $ gcc -E transmit.c
     $ gcc -E receive.c
     $ gcc -E main.c
   - % Assembly code
     $ gcc -S transmit.c
     $ gcc -S receive.c
     $ gcc -S main.c

- % Binary file
  $ gcc -c transmit.c
  $ gcc -c receive.c
  $ gcc -c main.c
- % Executable file
  $ gcc -o main main.o transmit.o receive.o

3. Compiling a program in the first time usually takes a longer time in comparison with the next re-compiling. What is the reason?
   - A program takes long time to compile depend on the structures of the program (Ex: struct, ...) or linking header files and object files together.
   - However since next time recompile will take less time due to linking already built as the first time compilation stage.
4. How do we make Makefile for other programming languages ? Use specific examples to support languages.
   - To use Makefile for other languages, we can make a general-purpose Makefile to support all types of languages.
   - We can define macros and general-purpose command in Makefile to support it. (Ex: "CC=gcc" to define the compilation of C language).

```
JFLAGS = -g
JC = javac
.SUFFIXES: .java .class
.java.class:
        $(JC) $(JFLAGS) $*.java

CLASSES = \
        Foo.java \
        Blah.java \
        Library.java \
        Main.java

default: classes

classes: $(CLASSES:.java=.class)

clean:
        $(RM) *.class
```

5. When source code files are located in different directories, how can we write a Makefile ?
   - Suppose we have .h files in a specific directory and some source code files in src directory, we can define paths to the specific places where .h files were stored.
   - To include any libraries, Makefile has macros defined for them (Ex: the math library as "LIBS=-lm").
   - Beside, the created Makefile should be located in the src directory.

1.2 Programming Exercises

1. I use the KMP algorithm to implement the file findsubstr.c, so the O-notation is O(m + n), for m is the size of the string, n for the size of the substring.

2. The result after running some several commands:

input.txt

```
1  Hello, world
2  Operating system
3  Computer Science and Engineering
```