東南大學
SOUTHEAST UNIVERSITY

OPERATING SYSTEM CONCEPTS

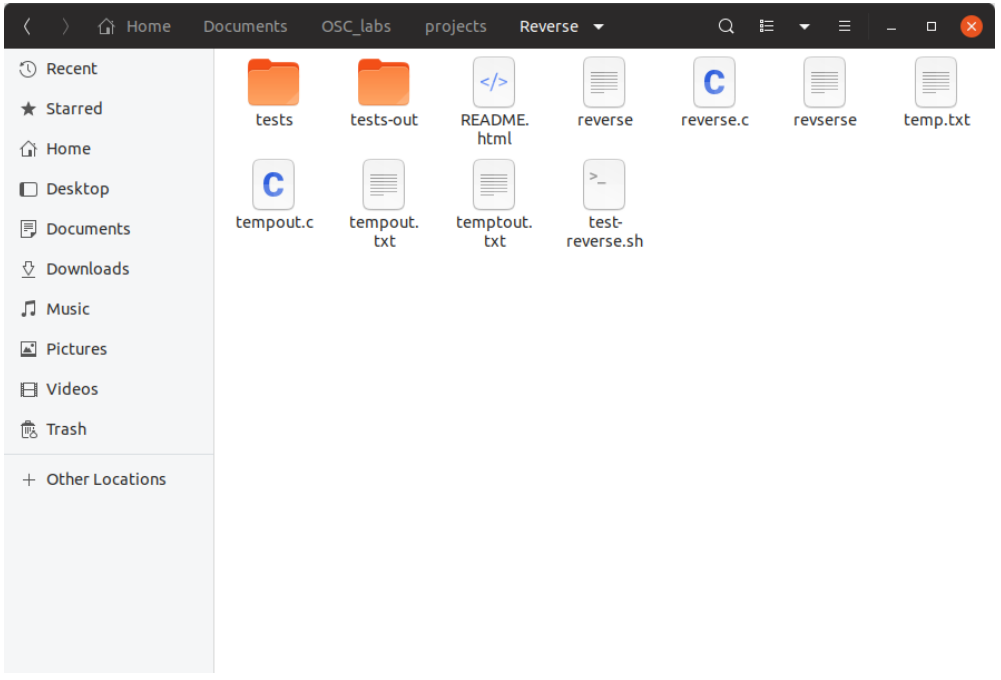
Chapter A0. Guide to Labworks

A/Prof. Kai Dong

Contents

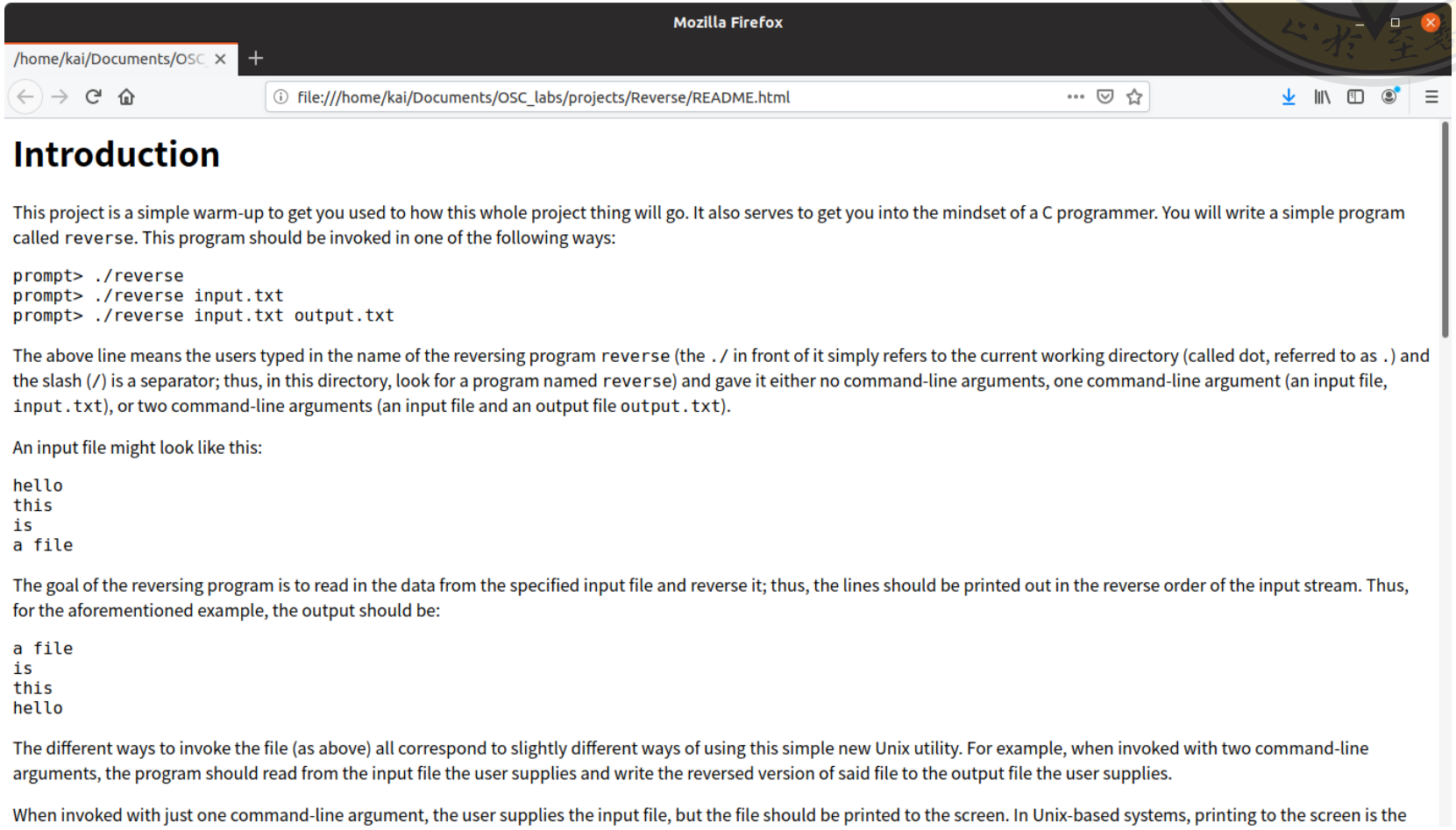
General Guide: Step #1

```
1  cd projects/Reverse/
```



Step #2

- Read carefully the corresponding “README.html” file.



Introduction

This project is a simple warm-up to get you used to how this whole project thing will go. It also serves to get you into the mindset of a C programmer. You will write a simple program called `reverse`. This program should be invoked in one of the following ways:

```
prompt> ./reverse
prompt> ./reverse input.txt
prompt> ./reverse input.txt output.txt
```

The above line means the users typed in the name of the reversing program `reverse` (the `./` in front of it simply refers to the current working directory (called dot, referred to as `.`) and the slash (`/`) is a separator; thus, in this directory, look for a program named `reverse`) and gave it either no command-line arguments, one command-line argument (an input file, `input.txt`), or two command-line arguments (an input file and an output file `output.txt`).

An input file might look like this:

```
hello
this
is
a file
```

The goal of the reversing program is to read in the data from the specified input file and reverse it; thus, the lines should be printed out in the reverse order of the input stream. Thus, for the aforementioned example, the output should be:

```
a file
is
this
hello
```

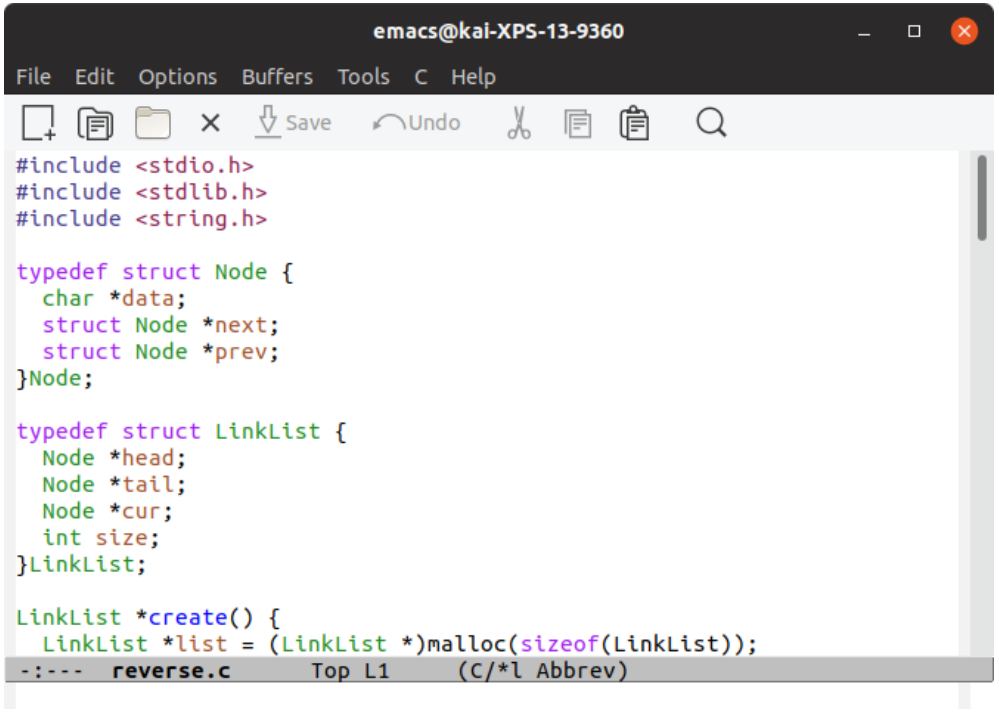
The different ways to invoke the file (as above) all correspond to slightly different ways of using this simple new Unix utility. For example, when invoked with two command-line arguments, the program should read from the input file the user supplies and write the reversed version of said file to the output file the user supplies.

When invoked with just one command-line argument, the user supplies the input file, but the file should be printed to the screen. In Unix-based systems, printing to the screen is the

Step #3

- Write your own reverse.c file.

```
1  emacs reverse.c
```



How to Use emacs

1. Type/edit as you wish.
2. Two commands you have to know (C is for Ctrl):

C-x C-s	保存文本
C-x C-c	退出 Emacs

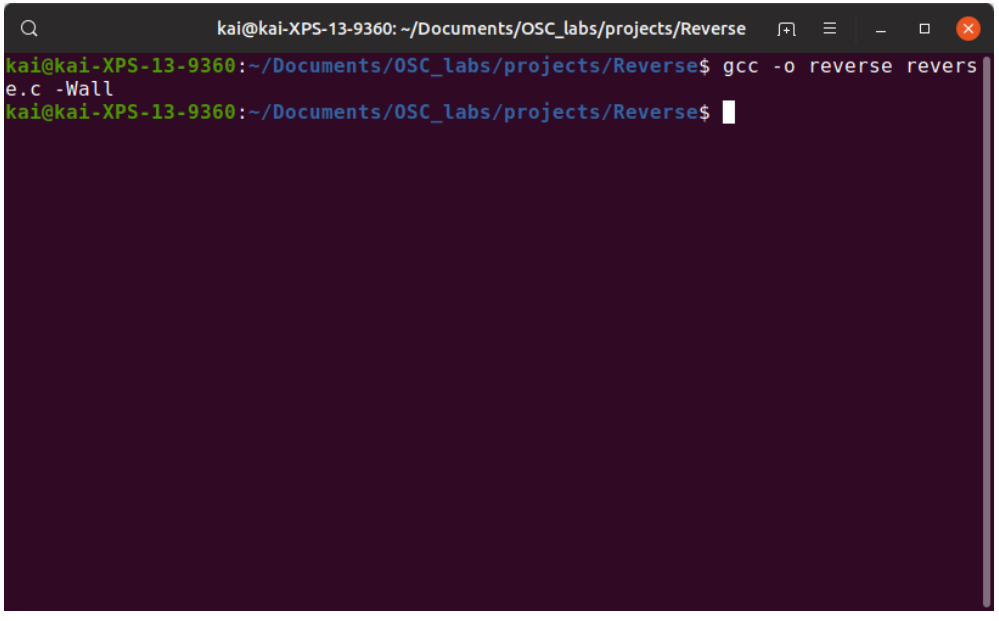
3. Check the other commands by yourself if needed.

Step #4

- Compile it.

```
1  gcc -o reverse reverse.c -Wall
```

- If you are lucky:



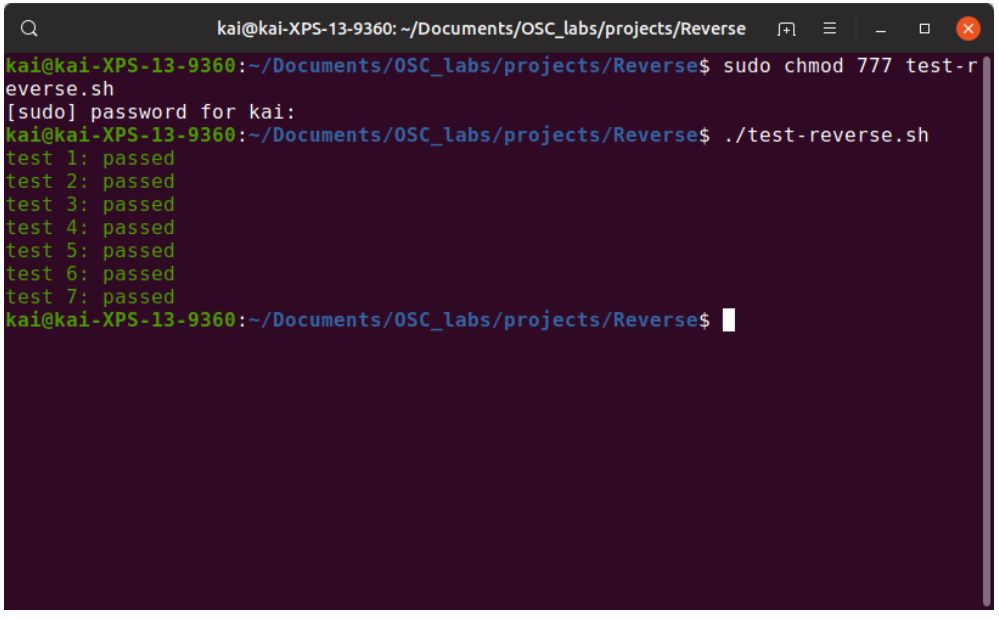
- Otherwise, debug it.

Step #5

- Run/test it.

```
sudo chmod 777 test-reverse.sh
./test-reverse.sh
```

- If you are lucky:



- Otherwise, debug it.

Contents

Guide to Debugging

This small guide can be very important!

- If you got errors/warnings when you compile your code, this guide is not at all helpful.
- If you pass the compiler check, but fail in test ♯*n*.
  1. Check the files “n.err”, “n.out”, “n.rc” in the folder named “/tests-out”, this is what you put to stderr, stdout, and the return value.
  2. Compare to those files in “/tests”, to see what they should be.
  3. If you still have difficulties, check “n.desc”, “n.run” in “/texts” to see what test is actually performed in the testscript.

Contents

Guide to Kernel Hacking

Never hesitate to kernel hacking.

- Follow the Tips in “README.html” in “/Xv6-Syscall”.
  - 30 ~ 60 mins for downloading.
- Read carefully “background.html” in “/Xv6-Syscall”
  - This project is indeed SIMPLE! If you are doing a lot of labor working — you are probably doing something wrong.
- Follow the Running Tests section in “README.html” in “/Xv6-Syscall”.
- You might fail for test #2.
  - No worry, this is designed to. You can fix it in the next labwork.