Project 0: Linux Warm Up

Objective

This project is to help you getting familiar with Linux, which is, in fact, the operating system in the course title **Operating System Labs**. We only cover several topics, but highly recommend you to play with it as much as possible.

> "For the things we have to learn before we can do them, we learn by doing them."
> — Aristotle, The Nicomachean Ethics

Overview

The project contains two subparts and two bonus problems. In part 1, you will write shell scripts to accomplish simple tasks. Understanding some basic shell commands (cd, cp, ...) and concepts (redirection, pipe ...) might be helpful. In part 2, you will practise with linux C programming environment (gcc compiler, gdb debugger, text editors). Finally, you can try some advanced text processing tools in bonus problems.

Readings

OSTEP Lab Tutorial

Program Specifications

Part 1

You need to write two shell scripts, both of them should be executable in a Linux shell.

- `s1.sh` :

  ```
  % ./s1.sh foo
  ```

  The script first builds a new directory `foo` in the current directory (the one contains `s1.sh`), then creates two files: "name.txt" containing your name; "stno.txt" containing your student number. Finally, it makes a copy of "name.txt" and "stno.txt" in the same directory of `foo` .

- `s2.sh` :

  ```
  % ./s2.sh
  ```

  It generates a text file `output` . Each row of `output` represents a file in directory `/bin` , and contains three fields which are name, owner and permission of that file. The fields should be separated by white spaces. Furthermore,
  - All file names in `output` should start with letter "b" (other files should be excluded)
  - The file `output` is sorted by the file name field (in order of alphabet, from small to large)
  - The file `output` is read only for other users

Here is an example output

```
bacman root -rwxr-xr-x
badblocks root -rwxr-xr-x
baobab root -rwxr-xr-x
base64 root -rwxr-xr-x
basename root -rwxr-xr-x
bash root -rwxr-xr-x
bashbug root -r-xr-xr-x
bayes.rb root -rwxr-xr-x
bbox root -rwxr-xr-x
```

Part 2

We have a C program set_operation.c, which contains several implementation bugs. You will first try to compile/run the program, then detect and correct the bugs using the gdb debugger.

- set_operation.c computes $(A - B) \cup (B - A)$, where A and B are two sets input by users.
- The algorithm is simple:
  - copy A to a temporary set A_2
  - compute $A = A - B$ and $B = B - A_2$
  - union A and B
- There are two sub-functions: output and check
  - "output" is used to output all elements in a linked list
  - "check" is used to check if an integer belongs to a linked list

A correct (bug-free) output should be:

```
--------------------------------
----Computing (A-B)union(B-A)----
--------------------------------
----input the number of elements of A: 3
1-th element: 3
2-th element: 4
3-th element: 5
----input the number of elements of B: 2
1-th element: 1
2-th element: 4
---- elements of (A-B)union(B-A) ----
1-th element: 3
2-th element: 5
3-th element: 1
```

Bonus 1

Using `awk` to print the 10th line of a file. For example, assume that a file has the following content:

```
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
```

Your script should output the tenth line, which is:

```
Line 10
```

Bonus 2

Given a text file, transpose its content. The file is in "column format", which contains a bunch of rows, each row has a fixed number of columns, and the columns are separated by the `' '` character. For example, if a file has the following content:

```
name age
alice 21
ryan 30
```

Your script should output:

```
name alice ryan
age 21 30
```

Hints

Part 1

- In the first script, you can use `echo` and redirection to output some texts into a file.
- In the second script, you may use `xargs` (between `find` and `ls`) or pipe.
- For text files, you can read lines with a loop. Or you may choose to use `awk` .
- To sort the data, see command `sort` .
- Print some information for debugging.
- Use `man` to get detail usages of commands.

Part 2

- Set breakpoints around important statements (e.g., loop, if)
- Print out values of variables (e.g., loop variables, contents of pointers).

Hand In

- The two script files `s1.sh` and `s2.sh` .
- The source file set_operation.c after debugging.
- A lab report contains:
  - A simple introduction of your project
  - For part 1, explain the shell commands in your scripts (e.g., usage, options).
  - For part 2, explain the meaning of gcc options for compiling your program. Explain the detailed process of debugging (where are your breakpoints located, which variables you've watched, etc.) and the bugs you found.

General Advice

Try to use and get familiar with the shell commands first. Make good use of "man", "help" and "--help".

Start small, and get things working incrementally. For example, you can try to write some sub-functions in your script first, then test them before you combine them together. For debugging, you could build several test cases, and see why the running process of the program is different from your expectation.