Proposal for Topic 1: Simulated Operation System

1. Module/Target:

The target of this project is to design a software that simulates several basic functions of a real OS using the C++ language. This software can implement a fake GUI system to some degree. It contains a graphical window (indicating a fake "screen") that allows user interaction and an embedded application that displays the real-time condition of the system's memory usage, executing tasks and error warnings.

This project generally simulates four basic functions:

- **memory allocation**: For a real operating system, memory needs to be allocated and protected executing any new application. Our program will borrow the memory allocation system from a real OS. Current memory usage (how much memory is used by this system and where this memory is allocated at in main memory) of fake OS is checked periodically and displayed it in our window.
- **file system**: This software can read and process files in two ways. All the files inside a specific file folder can be read directly. Users can upload their files to the fake OS by putting the files into this folder. After transporting the files, the user prompts this fake OS with a load order. If new icons appeared successively on-screen, corresponding files can be opened and changed. Users are also allowed to open their files in their computers by entering the direct address of files. This would generate a copy of that file under the file folder. A further change of file would only change this copy. The original file would not be changed in this case.
- **task scheduling**: For a real OS, it has an algorithm to schedule running tasks to optimize CPU and avoid potential errors. Our fake OS has a simplified version task scheduling, such that no more than two tasks can be run at the same time. This is realized by recording the condition of all processes, including both embedded applications and loaded files. All of them have two states, running or not running. If more then two tasks are detected to be running, the program will

close the earliest task automatically. This can help prevent running out of memory to some extent.

- **exception system:** The exception handler in a real operating system can be invoked in many cases. The exception system can detect exceptions and report the error. For example, if a running task causes a memory leak, error messages would be displayed, and that task terminated.

If the above functions are successfully realized and tested, we would add other functions to this project:

1) A simulated start-up process is designed. When users enter in this system, they are asked to log in or sign up by inputting a username and password. The system also gives different jurisdictions to different users, for examples, the right of administrator is the highest. Once the system is "powered on", the display panel of memory allocation would start running automatically. This simulated reading and evoking the OS from disk in a highly simplified way.
2) Mini video games: small game apps such as Chinese sliding-block and Retro Snake can be added for users to play, this is intended to make this OS more interesting.
3) Calendar and memorandum: a simple calendar can be embedded in the OS. Reminders can be created into this calendar.
4) Calculator: a single graphic calculator is provided, which can perform basic addition and multiplication.

2. Method to display:

The program will be displayed in software with a simulated GUI environment. A graphic window is shown to users when he or she opens our software. The icons representing files and mini applications are put on the "desktop". For more detailed description of the fake GUI, please refer to "visualization" part.

3. Procedures to build this OS:

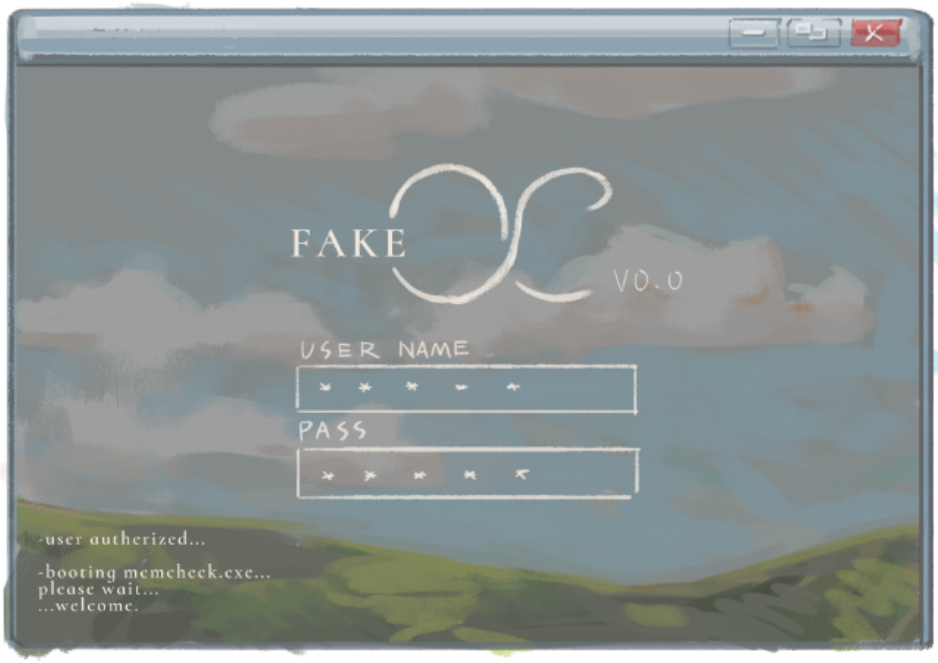
1） Construct a framework:

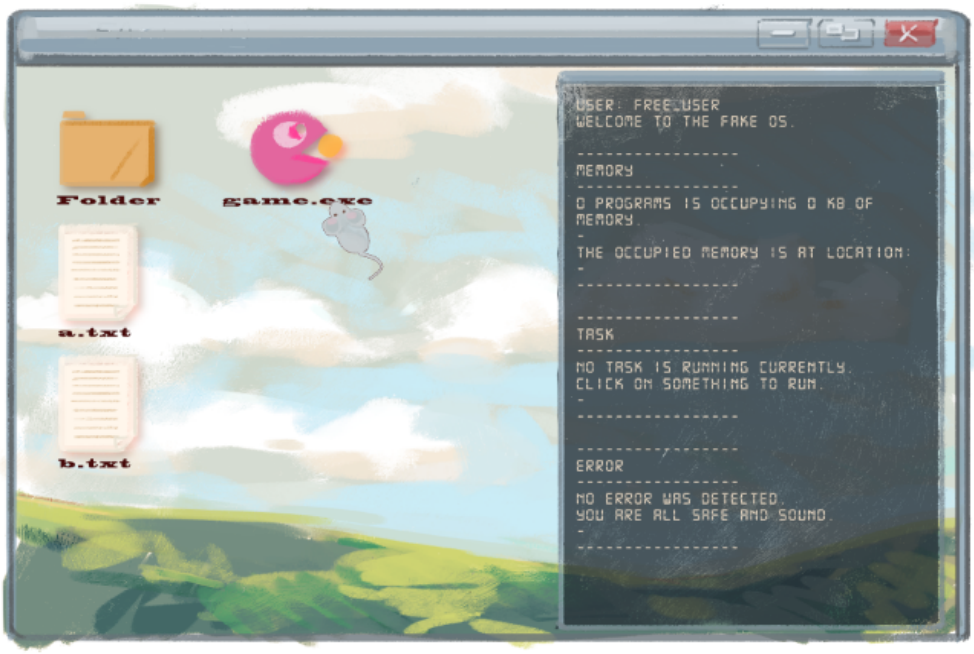
Write a program that can connect to the real operating systems. This program can obtain information about text files in the file folder, such as the size and creation time of each file. If the file is open in the fake OS, CPU time and CPU percentage are also needed.

Write a program that can read and write files. This part simulates the basic functions of file management in a real operating system. The program will scan through all the files in the file folder and create an object for each file. For example, this object will contain the name of this file, the content of this file and other vital attributes. Therefore, users can perform operations on the files through these objects. Once the user has made changes to the object, this program will transmit the information to the user's real OS. Then, the real OS will update the file relatively.

2） Visualization:

This project uses a fake GUI system. This system will be built using Qt. A sample display of this fake GUI is given below:





3） Add Error-Related Algorithms:

In this section , we need to write:

- Algorithms for solving errors.
- Algorithms that keep updating the information of the error detection, memory usage, and file information.

4） Add more functions:

Apply more functions for the user. For example, small games that can run in this program, a calculator and a calendar are also accessible in this software. These functions simulates the variety of functions a real OS bear. By clicking relative icons, user can utilize these functions.

5） Overall examination:

The whole program should be integrated and tested in every possible situation.

6） User manual & Report:

The user manual can help users to prepare the environment for this program and learn how to use it. The report will show our work and more detail information about this operating system.

4. Timetable:

Procedure	Time
Construct a framework	Finish by 4/20
Visualization	Finish by 4/21
Exception	Finish by 4/25
Function extension	Finish by 5/15
Overall examination	Finish by 5/18
User manual & Report	Finish by 5/18

5. Work distribution:

<table>
<tr><td colspan="2">Group Member</td><td colspan="3">Work distribution</td></tr>
<tr><td>Name</td><td>ID</td><td>Part 1 (~4/22)</td><td>Part 2 (~5/15)</td><td>Part 3 (~5/18)</td></tr>
<tr><td>Yiru Mao</td><td>118010223</td><td>Framework construction</td><td rowspan="4">Function extension (calendar, calculator, mini video game and start-up process)
Each person writes one function.</td><td rowspan="4">Overall examination;
User manual & Report</td></tr>
<tr><td>Ruichun Yang</td><td>118010368</td><td>Visualization</td></tr>
<tr><td>Zhanxin Wu</td><td>118010336</td><td>Framework construction</td></tr>
<tr><td>Yawei Xu</td><td>118010360</td><td>Visualization</td></tr>
</table>

6. Relative works:

- We can learn the basic operating system design and have a better understanding of OS by reading the two books: Operating Systems Design and Implementation, Third Edition, and The Design and implementation of the FreeBSD operating system, Second Edition.

- Teaching materials provided for the course Operating Systems: A Design-Oriented Approach that gives an introduction to the services included in the OS: https://www.cs.unm.edu/~crowley/osbook/begin.html
- Introduction to the file system in the OS: https://www.geeksforgeeks.org/file-systems-in-operating-system/
- Conclusion of the file access method in the OS: https://www.geeksforgeeks.org/file-access-methods-in-operating-system/?ref=rp
- We could learn the basic visualization design from the https://demo.os-js.org/
- Reference book: C++ GUI Programming with Qt 4, Jasmin Blanchette, Mark Summerfield (2006).
- Qt creator manual: https://doc.qt.io/qtcreator/index.html
- Using a designer UI file in your application in Qt: https://doc.qt.io/qt-5/designer-using-a-ui-file.html