Assignment 1

Deadline: Due at 11:59PM on Mar 2, 2024

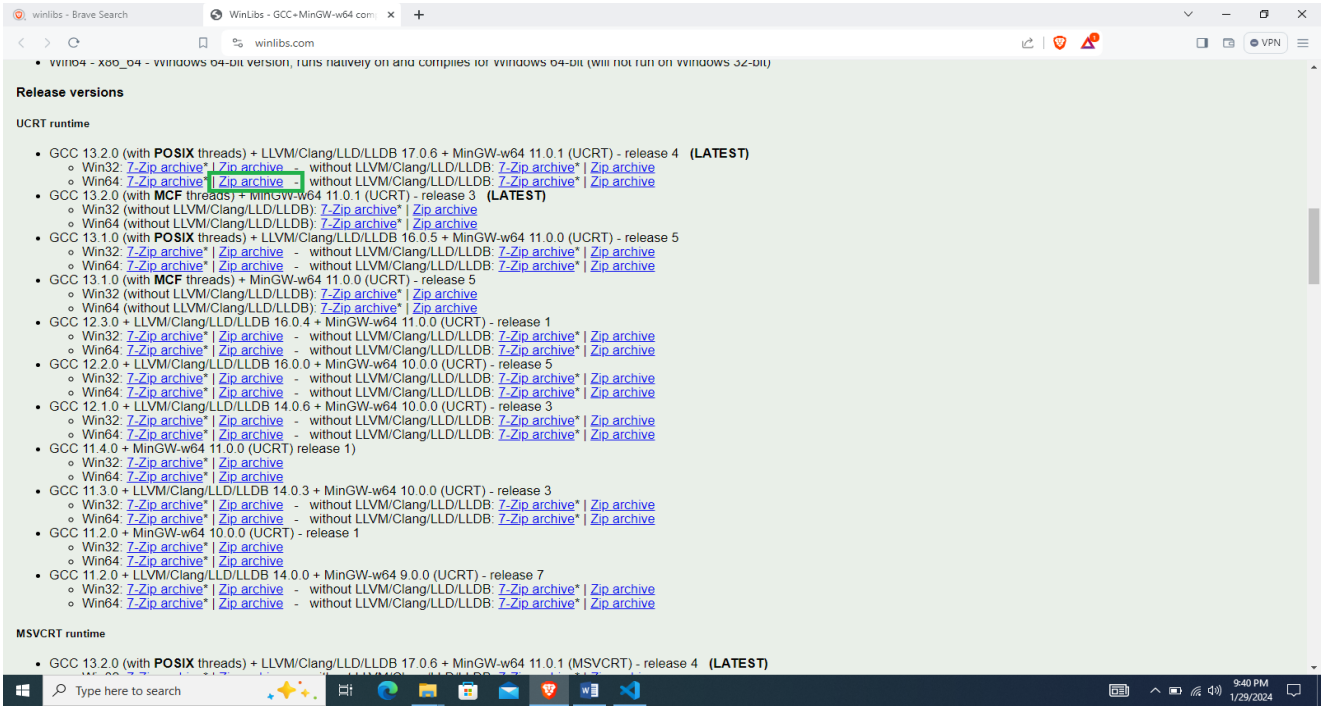
This assignment aims to enhance your familiarity with the Unix system call interface and the shell by implementing various features in a small shell program. The assignment focuses on parsing shell commands and implementing their functionalities. You can complete this assignment on any operating system supporting the Unix API, such as a Linux Athena machine, a computer with Linux or MacOS, etc. You can work in a group of maximum 3 people.

Requirements

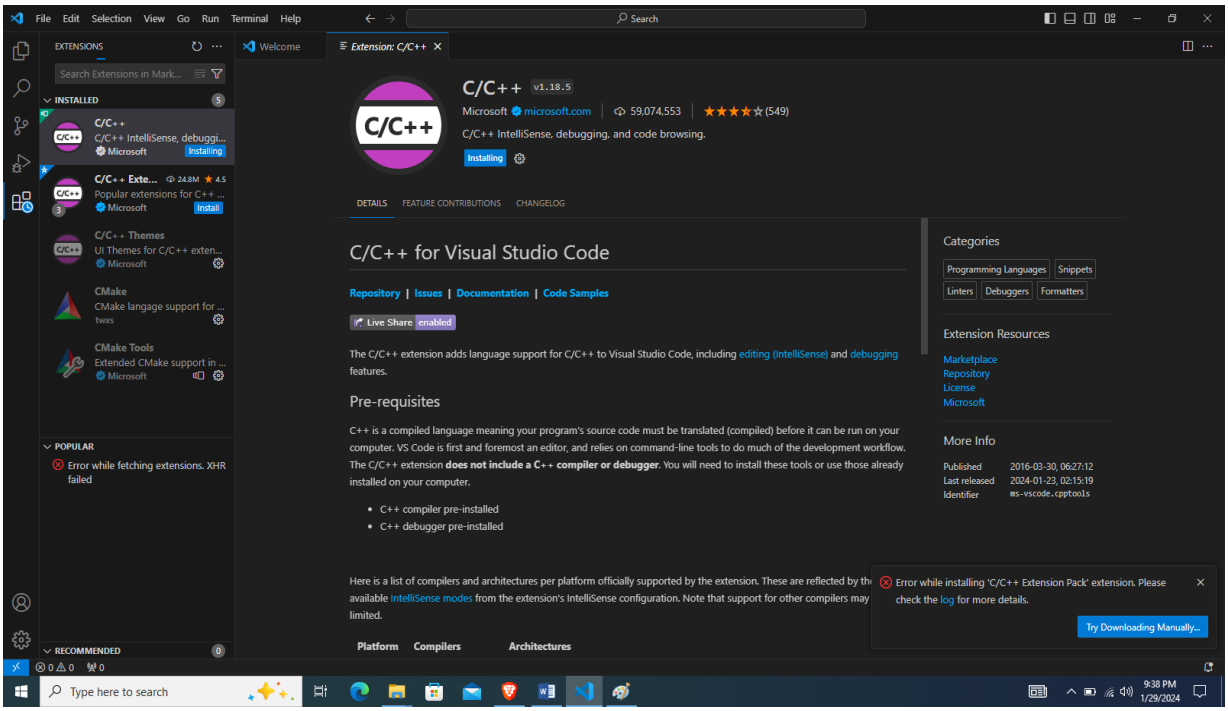
- Obtain the skeleton of the xv6 shell sh.c, which consists of parsing shell commands and implementing their execution.

Tool Setup

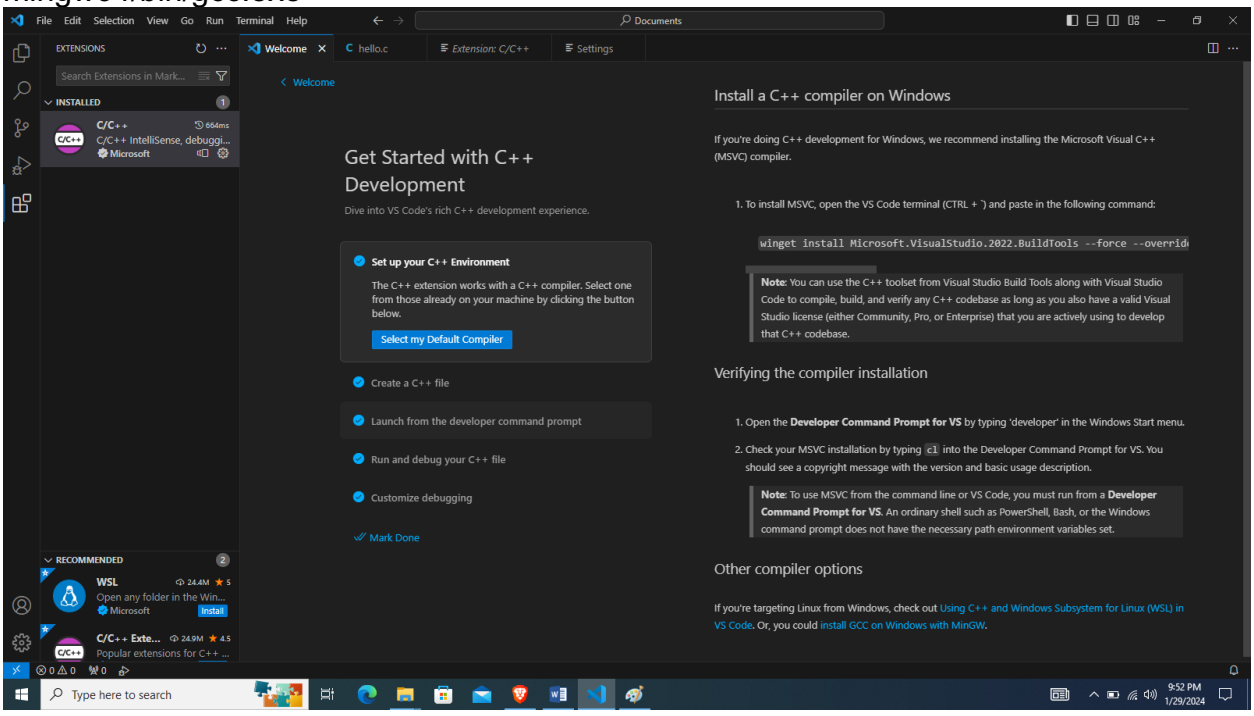
- WinLibs standalone build of GCC and MinGW-w64 for Window (https://winlibs.com/)
- Download and extract anywhere in user directory or usb drive.



- Install standalone version of VS Code
- Install C/C++ Extension in VS Code

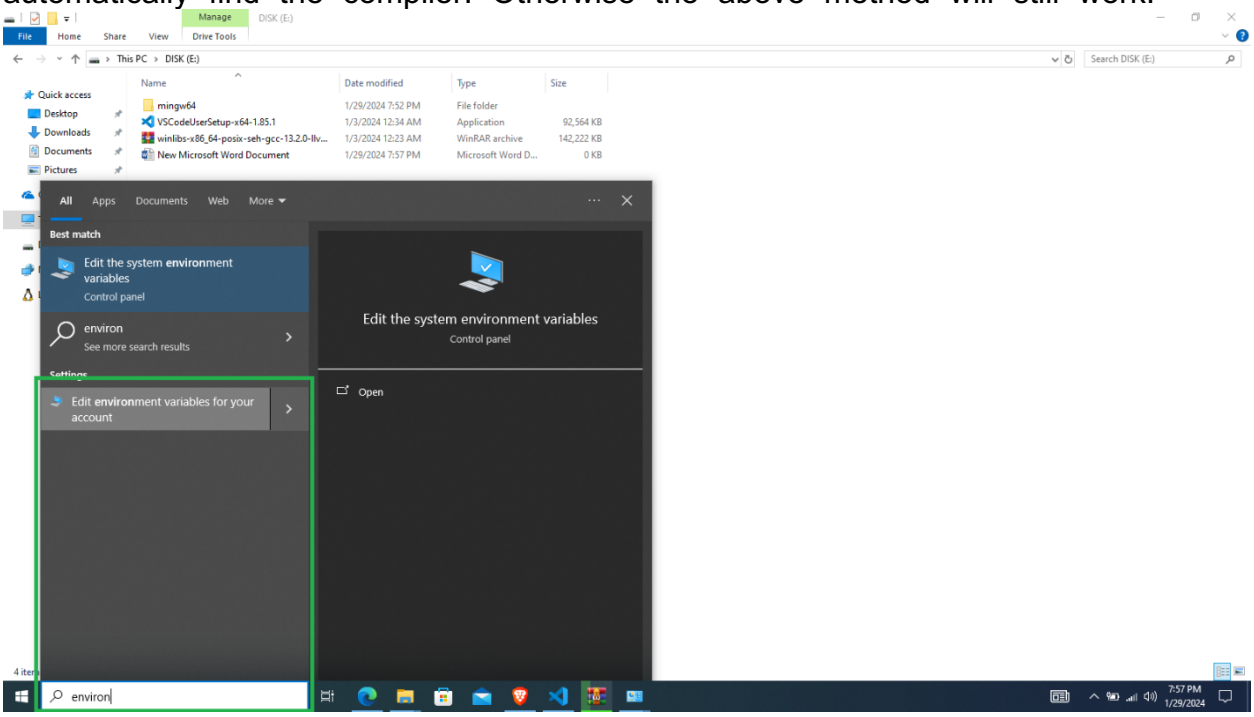


- In VS Code, open a folder (e.g Documents, It is like opening a file)
- Click trust the authors.
- Configure the installed extension and select the default compiler. It is mingw64/bin/gcc.exe



- If you don't find extension configuration, it could be found on welcome page.

- (Optional) Just in case if you could add environment variables for user account, they are different from system variables, it would be just an extra for vs code to automatically find the compiler. Otherwise the above method will still work.



Assignment Instructions

1. Understanding the Skeleton Shell

- Examine the skeleton shell, which is divided into two main parts: parsing shell commands and implementing their functionalities.
- The parser recognizes simple shell commands, examples of which include:

```
ls > y
cat < y | sort | uniq | wc > y1
cat y1
rm y1
ls |  sort | uniq | wc
rm y
```

- Copy and paste these commands into a file named t.sh.

2. Compile and Execute

- Compile the skeleton shell using the following command:
  > $ gcc sh.c
- This produces an a.out file, which you can run using:
  > $ ./a.out < t.sh

Note: The execution may result in a panic as several features are yet to be implemented.

3. Task 1: Executing Simple Commands

- Implement the execution of simple commands, e.g., $ ls.

- The parser already constructs an execcmd for you, so your task is to write the code for the ' ' case in runcmd.
- Test your implementation by running "ls". Refer to the exec manual page for guidance (man 3 exec).

4. Task 2: I/O Redirection

- Implement I/O redirection commands, allowing you to run commands like:
  >echo "6.828 is cool" > x.txt
  >cat < x.txt

- The parser already recognizes > and <, constructing a redircmd for you. Your task is to complete the missing code in runcmd for these symbols.
- Ensure your implementation runs correctly with the provided test input. Refer to the open and close man pages for assistance.

5. Task 3: Implement Pipes

- Implement command pipelines using pipes, for example:
  >$ ls | sort | uniq | wc

- The parser recognizes | and constructs a pipecmd for you. Your responsibility is to write the code for the '|' case in runcmd.
- Test your implementation with the provided pipeline. Refer to the man pages for pipe, fork, close, and dup for guidance.

6. Execute

- Now you should be able the following command correctly:
  >$ ./a.out < t.sh

7. README

- General guidelines for the README file is in this URL.
  - https://datamanagement.hms.harvard.edu/collect-analyze/documentation-metadata/readme-files
  - https://android.googlesource.com/kernel/common.git/+/bcmdhd-3.10/README

8. Submission Guidelines

- Submit the modified shell code with added functionalities.
- Include a README file describing any features implemented and the testing process.
- Include your student numbers at the top of your code as comments and in the README file as well.
- Place all files in a directory. Compress the directory as a zipped directory and upload .zip file to Learning Hub.
- Make sure that your group self enroll into a group. Please make only one submission per group.

9. Grading Guidelines

- Correct implementation of executing simple commands, I/O redirection, and pipes. **(5 Pts)**
- Proper handling of edge cases and errors. **(5 Pts)**
- Code organization, readability, and adherence to best practices. **(3 Pts)**
- Documentation quality in the README file. **(2 Pts)**

Good luck with your assignment!