Lia Souris

Course Project

CT30A3370 Käyttöjärjestelmät ja systeemiohjelmointi

Project 1: Warmup to C and Unix programming

Summary

This project was a warm-up exercise in C programming. The assignment was to write a program called "reverse" that reverses text input line by line. The program was to be invoked in one of the following ways:

./reverse – reads from standard input and outputs to the screen

./reverse input.txt – reads from a file and outputs to the screen

./reverse input.txt output.txt – reads from one file and writes the reversed lines to another

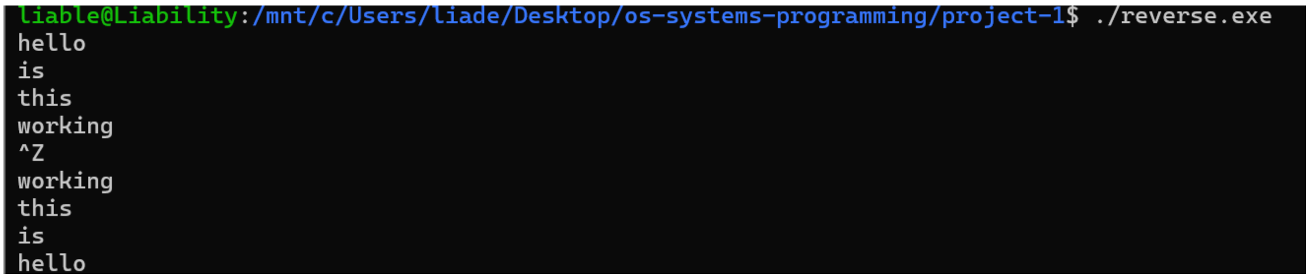
The program is written in C using dynamic memory management, a linked list, and standard file I/O functions. The program handles long lines using dynamic memory management. The "read_line()" function begins with an initially defined 128-byte buffer that then dynamically doubles its capacity using "realloc()" whenever additional space is needed to accommodate longer input lines. The lines are stored in a linked list in reverse order as they are read. The implementation includes all the necessary error handling for memory allocation failures, invalid file access, and duplicate input/output filenames. All the error messages are directed to stderr.

Source code

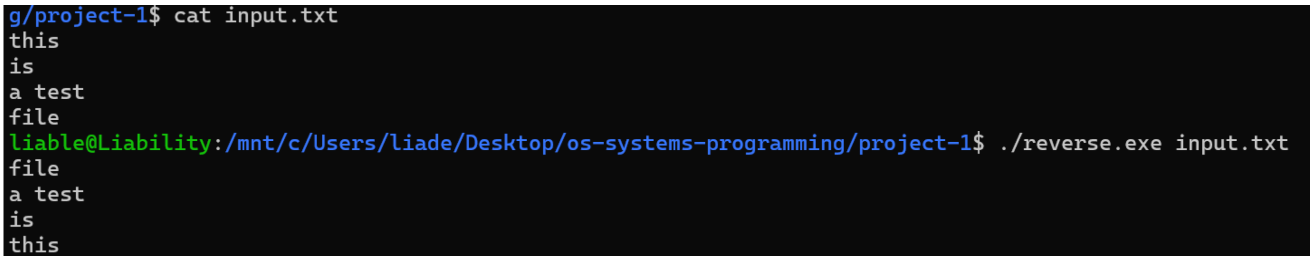
The complete source code for this project is available in the project-1 directory of my GitHub repository: https://github.com/liasouris/os-systems-programming.git

Screenshots of the program functioning:

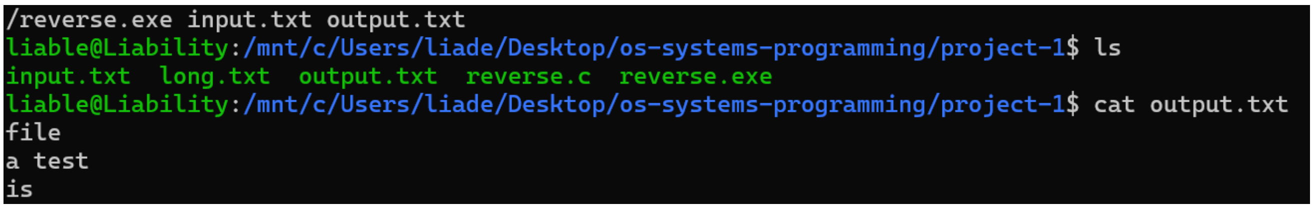
Test cases: 1) Run using standard input (no arguments), (2) Run with a single input file (1 argument), (3) Run with both input and output files (2 arguments), (4) Handle errors: too many arguments, missing input/output files, and identical input/output files, and (5) Reverse files containing long lines to verify that dynamic buffer resizing functions correctly.



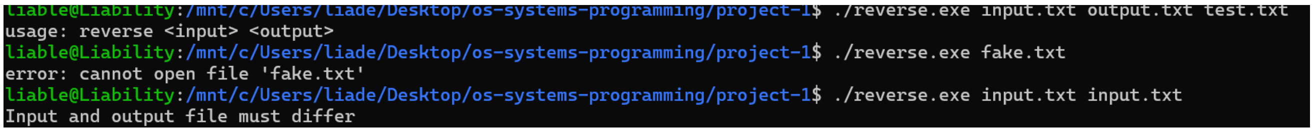
```
liable@Liability:/mnt/c/Users/liade/Desktop/os-systems-programming/project-1$ ./reverse.exe
hello
is
this
working
^Z
working
this
is
hello
```



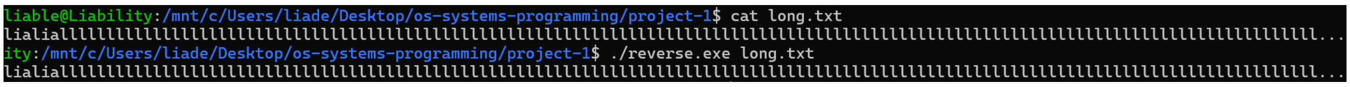
```
g/project-1$ cat input.txt
this
is
a test
file
liable@Liability:/mnt/c/Users/liade/Desktop/os-systems-programming/project-1$ ./reverse.exe input.txt
file
a test
is
this
```



```
/reverse.exe input.txt output.txt
liable@Liability:/mnt/c/Users/liade/Desktop/os-systems-programming/project-1$ ls
input.txt  long.txt  output.txt  reverse.c  reverse.exe
liable@Liability:/mnt/c/Users/liade/Desktop/os-systems-programming/project-1$ cat output.txt
file
a test
is
```



```
liable@Liability:/mnt/c/Users/liade/Desktop/os-systems-programming/project-1$ ./reverse.exe input.txt output.txt test.txt
usage: reverse <input> <output>
liable@Liability:/mnt/c/Users/liade/Desktop/os-systems-programming/project-1$ ./reverse.exe fake.txt
error: cannot open file 'fake.txt'
liable@Liability:/mnt/c/Users/liade/Desktop/os-systems-programming/project-1$ ./reverse.exe input.txt input.txt
Input and output file must differ
```



```
liable@Liability:/mnt/c/Users/liade/Desktop/os-systems-programming/project-1$ cat long.txt
lialiallllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll...
ity:/mnt/c/Users/liade/Desktop/os-systems-programming/project-1$ ./reverse.exe long.txt
lialiallllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll...
```

Project 2: Unix Utilities

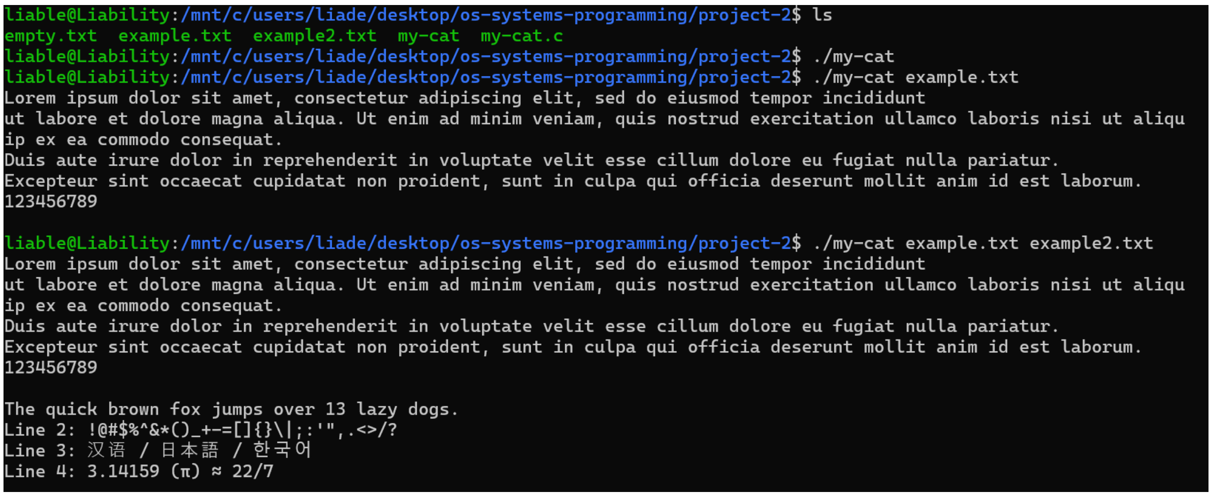
My-cat

Summary

The assignment was to implement a simplified version of the UNIX “cat” command called “my-cat”, which can read and print the contents of one or multiple files. The program checks if any files were provided if not it exits. The program opens the file in read mode using the “fopen()” function. If the file cannot be opened, it prints an error message to “stderr” and exits with a status code of 1. When a file is opened successfully, the program reads its contents line by line using the “fgets()” function and prints each line using the “printf()” function. After processing all lines, the file is closed with “fclose()”. When given multiple arguments, the program processes each specified file one after another. It prints their contents and exits with a status code of 0 when successful. I followed all the requirements detailed in the assignment description.

Source code
The full source code for this project can be found in the project-2/my-cat.c file in my GitHub repository."https://github.com/liasouris/os-systems-programming.git

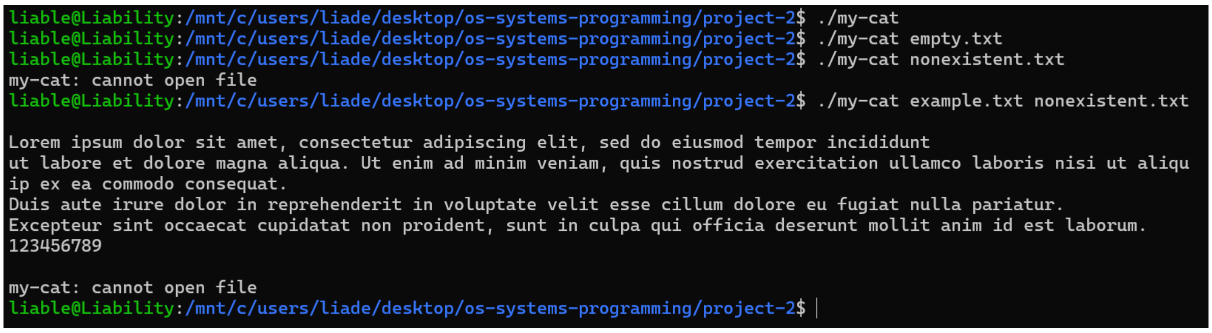
Screenshots of the program functioning:
Test cases: (1) Run with a single file (2 arguments), (2) Run with two files (3 arguments), (3) No files specified, program returns 0, (4) Input file is empty, program returns 0, and (5) Error handling: failure to open file.



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ls
empty.txt  example.txt  example2.txt  my-cat  my-cat.c
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-cat
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-cat example.txt
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliqu
ip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
123456789

liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-cat example.txt example2.txt
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliqu
ip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
123456789

The quick brown fox jumps over 13 lazy dogs.
Line 2: !@#$%^&*()_+-=[]{}\|;:'",.<>/?
Line 3: 汉语 / 日本語 / 한국어
Line 4: 3.14159 (π) ≈ 22/7
```



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-cat
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-cat empty.txt
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-cat nonexistent.txt
my-cat: cannot open file
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-cat example.txt nonexistent.txt

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliqu
ip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
123456789

my-cat: cannot open file
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ |
```

My-grep

Summary

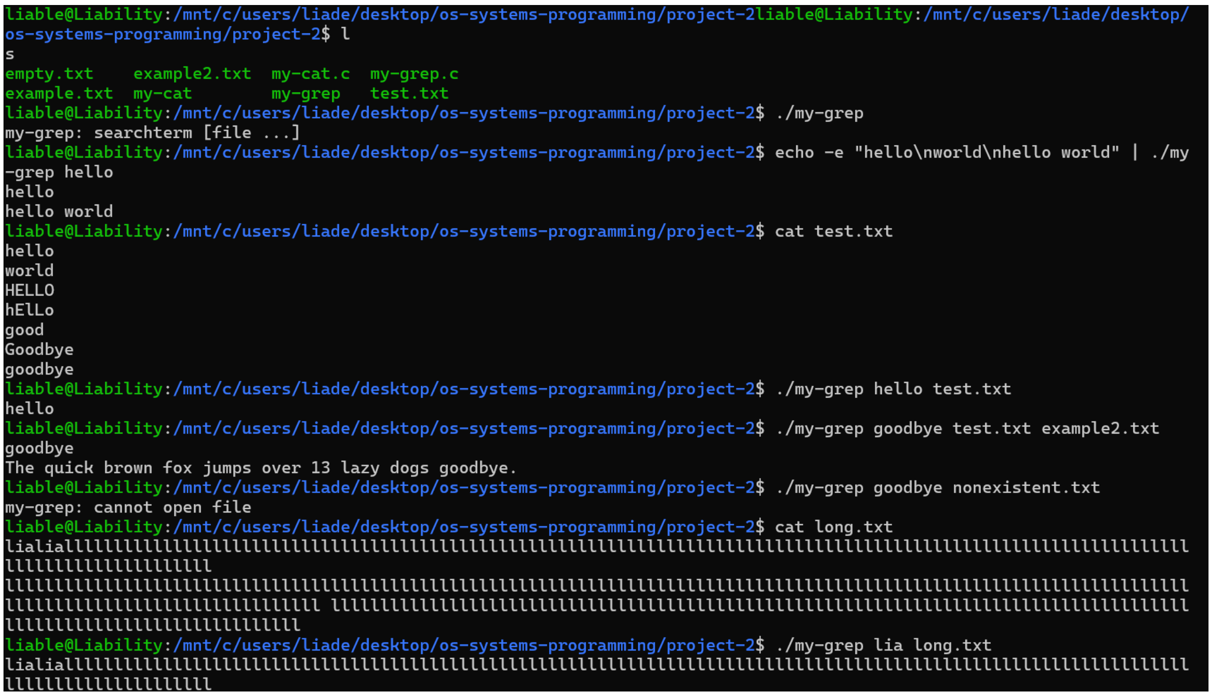
The assignment was to implement a simplified version of the UNIX grep command called my-grep, which searches for a specified term in one or multiple files and then prints the matching lines. The program first checks if a search term was provided, if not, it prints an error to stderr and exits with status 1. If files are provided, it attempts to open each in read mode using “fopen()”. If any file fails to open, it prints an error and exits with status 1. When given a valid file as an argument, the program reads each line using getline() also handling arbitrarily long lines. It prints the lines containing the exact case-sensitive search term using “strstr()”. The program handles multiple files sequentially, exiting with status 0 when execution is successful. All requirements including case sensitivity, long-line support, and proper error handling were implemented as specified.

Source code

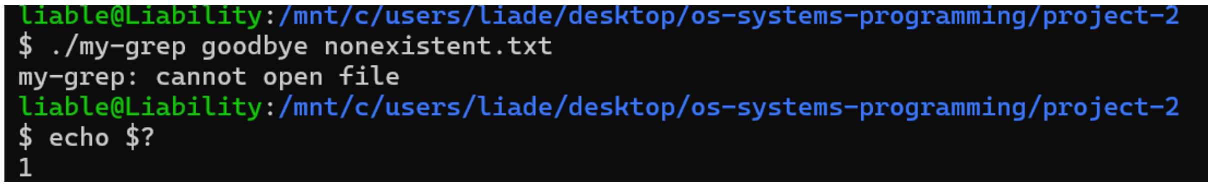
The full source code for this project can be found in the project-2/my-grep.c file in my GitHub repository**:** https://github.com/liasouris/os-systems-programming.git

Screenshots of the program functioning:

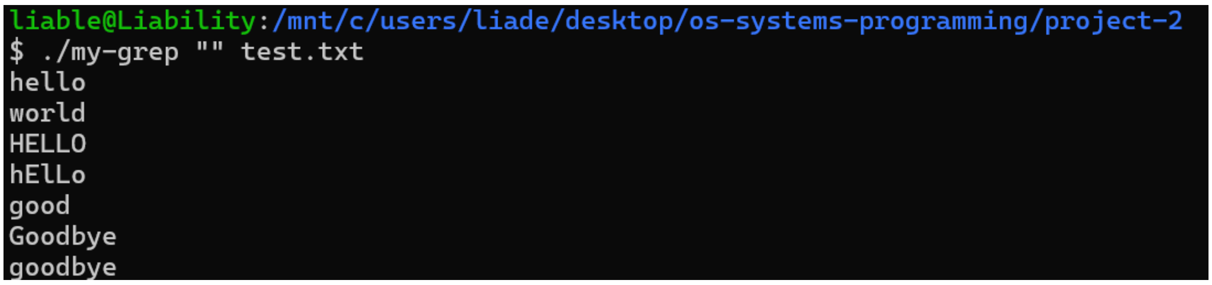
Test cases: (1) Run with no arguments, verifying error handling, (2) Search term only to confirm stdin reading functionality, (3) Searching within a single file, (4) Searching with multiple files, (5) attempting to open nonexistent files to validate error messaging, (6) Checking program is in fact case-sensitive, (7) Handling of long lines to test buffer management, (8) empty search term input to verify edge case behavior, prints the full content of the file.



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2liable@Liability:/mnt/c/users/liade/desktop/
os-systems-programming/project-2$ l
s
empty.txt    example2.txt  my-cat.c  my-grep.c
example.txt  my-cat        my-grep   test.txt
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-grep
my-grep: searchterm [file ...]
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ echo -e "hello\nworld\nhello world" | ./my
-grep hello
hello
hello world
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ cat test.txt
hello
world
HELLO
hElLo
good
Goodbye
goodbye
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-grep hello test.txt
hello
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-grep goodbye test.txt example2.txt
goodbye
The quick brown fox jumps over 13 lazy dogs goodbye.
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-grep goodbye nonexistent.txt
my-grep: cannot open file
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ cat long.txt
lialiallllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllll
llllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
llllllllllllllllllllllllllllllll lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
llllllllllllllllllllllllllllll
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-grep lia long.txt
lialiallllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllll
```



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2
$ ./my-grep goodbye nonexistent.txt
my-grep: cannot open file
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2
$ echo $?
1
```



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2
$ ./my-grep "" test.txt
hello
world
HELLO
hElLo
good
Goodbye
goodbye
```

My-zip and my-unzip

Summary

The assignment was to implement a simplified version of the UNIX zip and unzip commands called my-zip and my-unzip, which allow users to compress and decompress files. The implemented tools use run-length encoding (RLE) for compression and decompression.

The my-zip tool reads one or multiple input files and compresses them by replacing consecutive duplicate characters with a 4-byte binary count followed by a single instance of the character. The compressed output is written to stdout. If no input files are provided, the tool exits with status 1 and prints an error message to stderr.

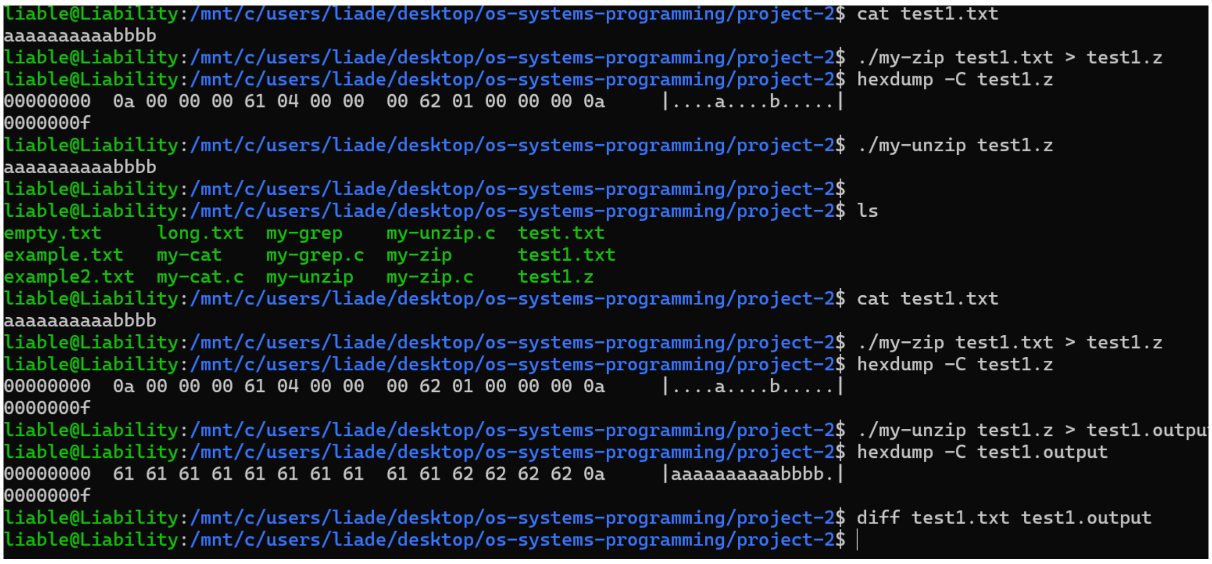
The my-unzip tool reverses this process by reading the 5-byte RLE entries from the compressed files and reconstructs the original content to stdout. It also handles missing files by displaying an appropriate error message.

Both my-zip and my-unzip support processing multiple files one after another, combining their output into a single stream. They use binary file mode ("rb") to make sure they work correctly on different systems. All functional and error-handling requirements specified in the assignment were fully implemented and tested.

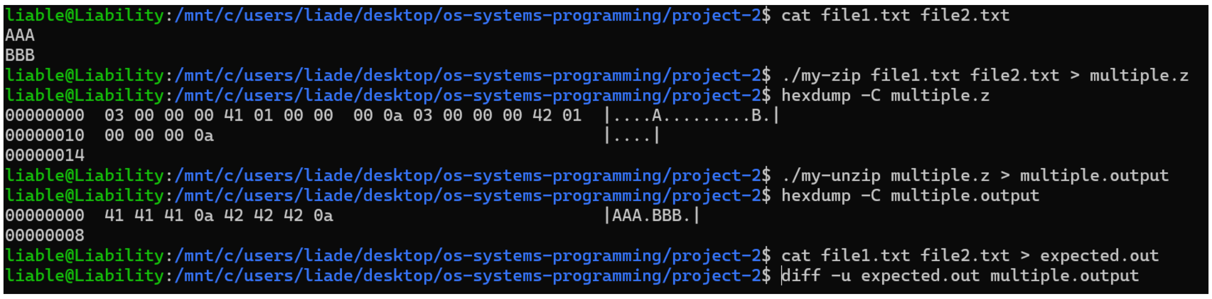
Source code
The full source code for this project can be found in the project-2/my-zip.c and my-unzip.c file in my GitHub repository: https://github.com/liasouris/os-systems-programming.git

Screenshots of the program functioning:

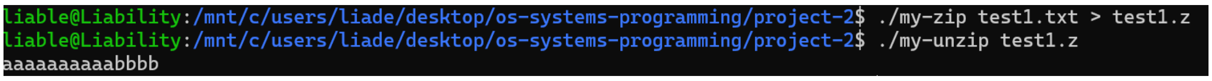
Test cases: (1) Run with no arguments, verifying error handling (2) Single-file compression and decompression to confirm correct output, (3) Combining the contents of multiple input files into one compressed output, (4) Restoring the compressed data and checking that it's the same as the original files joined together, and (5) Attempting to compress/decompress non-existent files (6)



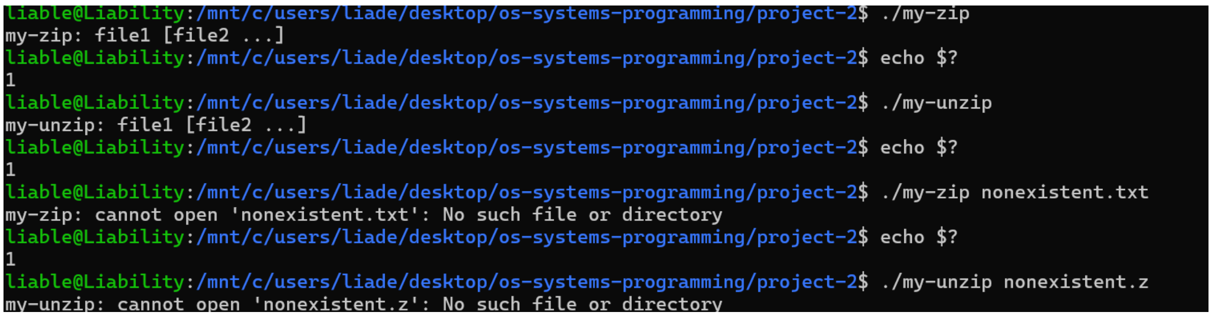
```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ cat test1.txt
aaaaaaaaaabbbb
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-zip test1.txt > test1.z
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ hexdump -C test1.z
00000000  0a 00 00 00 61 04 00 00  00 62 01 00 00 00 0a     |....a....b.....|
0000000f
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-unzip test1.z
aaaaaaaaaabbbb
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ls
empty.txt     long.txt  my-grep    my-unzip.c  test.txt
example.txt   my-cat    my-grep.c  my-zip      test1.txt
example2.txt  my-cat.c  my-unzip   my-zip.c    test1.z
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ cat test1.txt
aaaaaaaaaabbbb
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-zip test1.txt > test1.z
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ hexdump -C test1.z
00000000  0a 00 00 00 61 04 00 00  00 62 01 00 00 00 0a     |....a....b.....|
0000000f
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-unzip test1.z > test1.outpu
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ hexdump -C test1.output
00000000  61 61 61 61 61 61 61 61  61 61 62 62 62 62 0a     |aaaaaaaaaabbbb.|
0000000f
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ diff test1.txt test1.output
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$
```



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ cat file1.txt file2.txt
AAA
BBB
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-zip file1.txt file2.txt > multiple.z
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ hexdump -C multiple.z
00000000  03 00 00 00 41 01 00 00  00 0a 03 00 00 00 42 01  |....A.........B.|
00000010  00 00 00 0a                                       |....|
00000014
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-unzip multiple.z > multiple.output
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ hexdump -C multiple.output
00000000  41 41 41 0a 42 42 42 0a                           |AAA.BBB.|
00000008
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ cat file1.txt file2.txt > expected.out
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ diff -u expected.out multiple.output
```



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-zip test1.txt > test1.z
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-unzip test1.z
aaaaaaaaaabbbb
```



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-zip
my-zip: file1 [file2 ...]
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ echo $?
1
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-unzip
my-unzip: file1 [file2 ...]
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ echo $?
1
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-zip nonexistent.txt
my-zip: cannot open 'nonexistent.txt': No such file or directory
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ echo $?
1
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-2$ ./my-unzip nonexistent.z
my-unzip: cannot open 'nonexistent.z': No such file or directory
```

Project 3: Unix Shell

Summary

The assignment was to implement a simplified version of a UNIX shell called wish, which supports basic command execution, output redirection, parallel command handling, and batch file processing. The shell starts in interactive mode invoked by the command $ ./wish this then displays the wish> prompt and accepts commands from standard input. If a batch file is provided, it reads and executes commands sequentially from the file.

The shell supports built-in commands cd, path and exit. The cd command changes the current working directory, path sets the list of directories to search for executable commands and exit terminates the shell. For external commands, the shell uses “fork()” to create a new process and “execv()” to run the command. If the user includes a > symbol, the shell redirects the output to a file using “dup2()”. When multiple commands are separated by &, the shell runs them at the same time by creating separate processes for each one and uses “waitpid()” to make sure all of them finish before continuing.

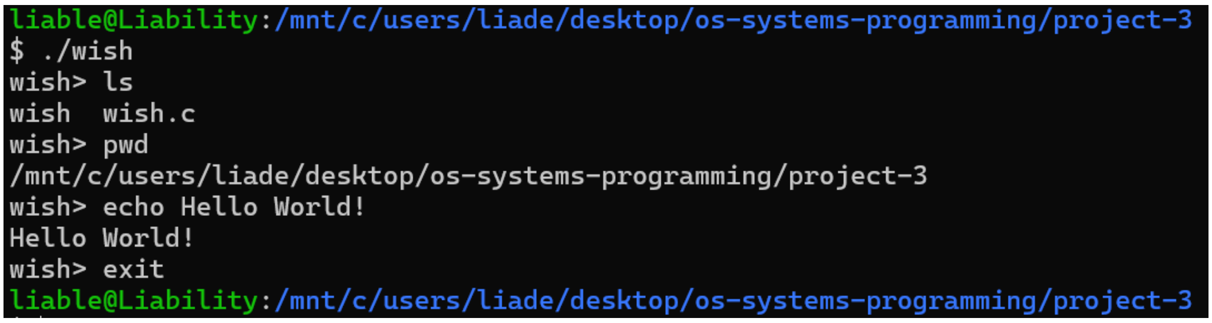
One challenge I came across while implementing this project was getting parallel commands to work. I had to manage several child processes at the same time and make sure each command ran on its own. I also needed to wait for all of them to finish without stopping the rest of the shell. This required careful use of process IDs and error checks. However it helped me better understand how running multiple commands works in Unix-like systems.

All functional and error-handling requirements, including built-in command behavior, output redirection, parallel execution, and batch mode processing were implemented according to the project details.

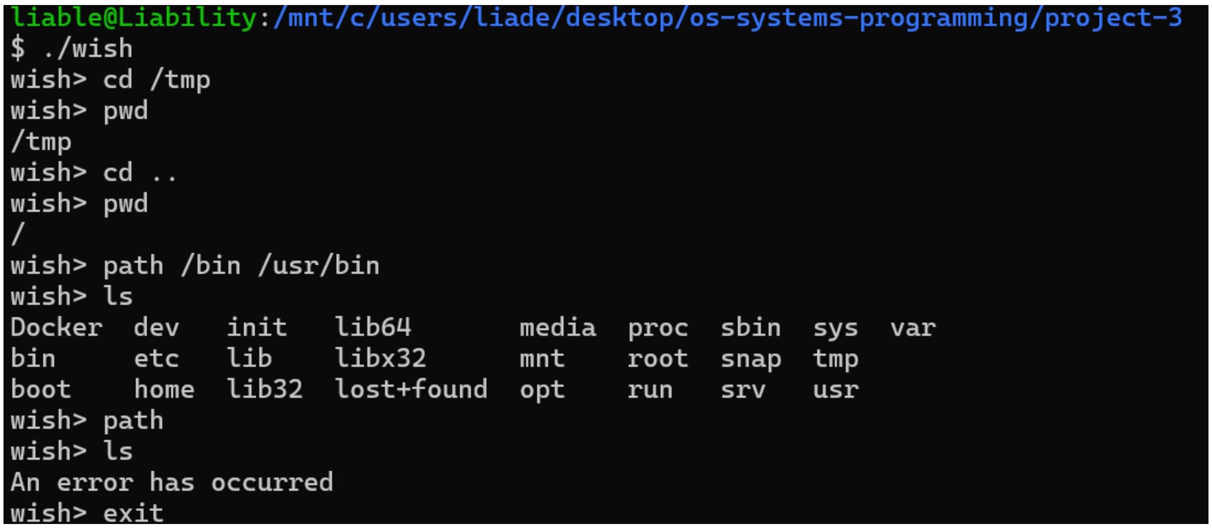
Source code
The full source code for this project can be found in the project-3/wish.c file in my GitHub repository**:**
https://github.com/liasouris/os-systems-programming.git

Screenshots of the program functioning:

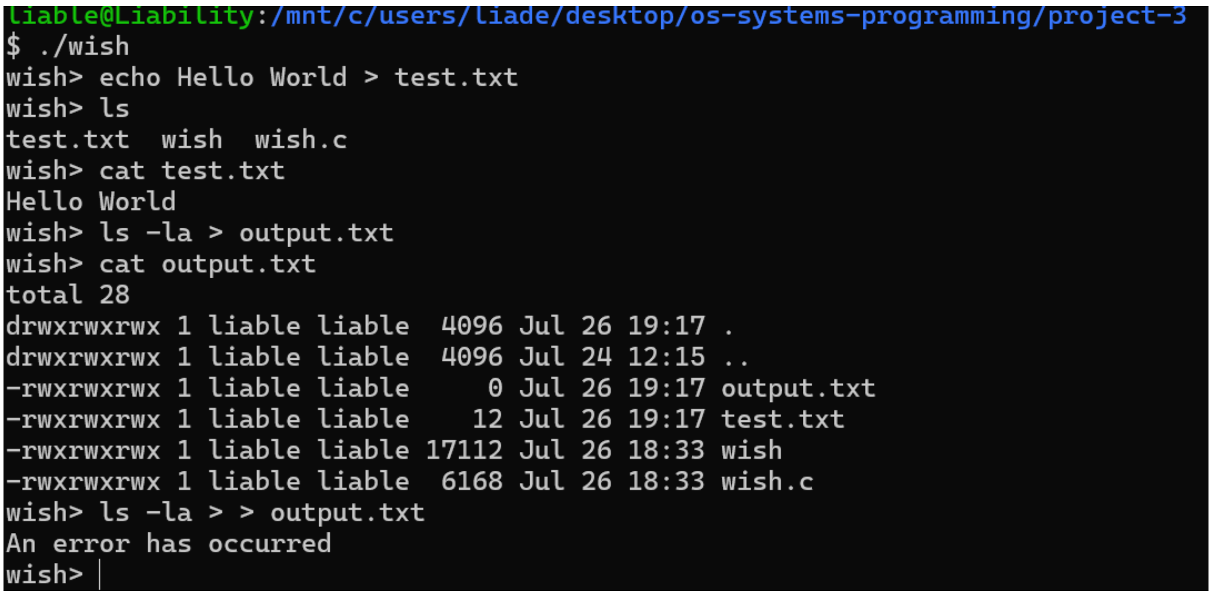
1) Run the program with no arguments to check that it starts in interactive mode, shows the wish> prompt, and handles basic commands like ls, pwd, and exit correctly, (2) Test built-in commands: exit with no arguments, cd with one argument, and path to make sure directory searching works as expected, (3) Test output redirection: check that it creates or overwrites files correctly, and shows errors when needed, (4) Run multiple commands in parallel to check that they execute at the same time and the program waits for all of them to finish, (5) Test batch mode by running a script file (like test_batch.txt) and making sure all commands run in order. Also test what happens when the batch file doesn’t exist, (6) Test error cases, like running invalid commands, using exit with extra arguments, giving cd the wrong number of arguments, or using incorrect redirection, and (7) Test edge cases, like lines with extra spaces, empty lines, and very long commands to make sure input is handled properly.



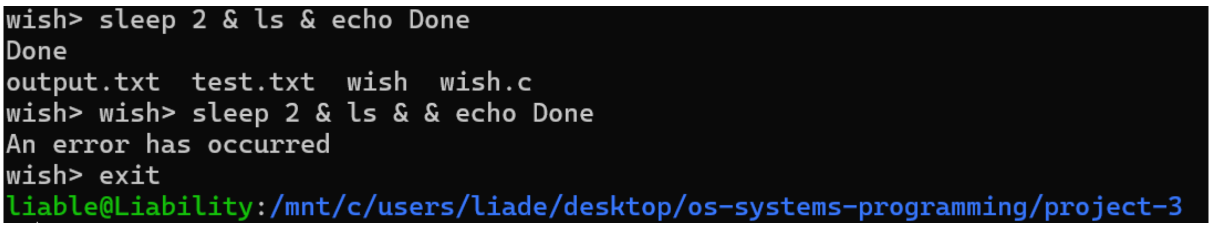
```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
$ ./wish
wish> ls
wish  wish.c
wish> pwd
/mnt/c/users/liade/desktop/os-systems-programming/project-3
wish> echo Hello World!
Hello World!
wish> exit
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
```



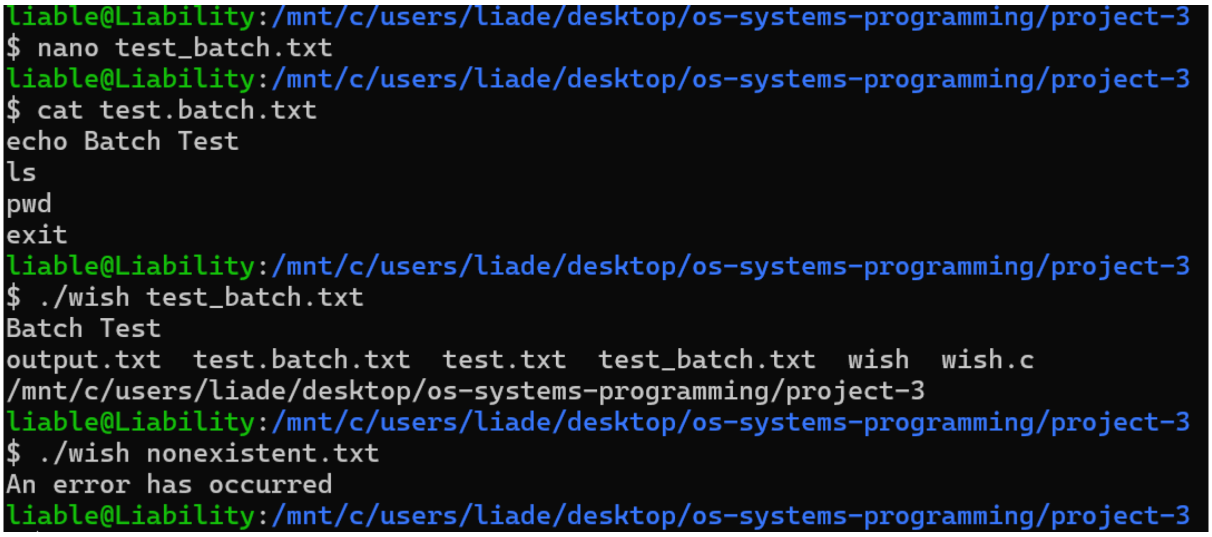
```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
$ ./wish
wish> cd /tmp
wish> pwd
/tmp
wish> cd ..
wish> pwd
/
wish> path /bin /usr/bin
wish> ls
Docker  dev   init  lib64       media  proc  sbin  sys  var
bin     etc   lib   libx32      mnt    root  snap  tmp
boot    home  lib32 lost+found  opt    run   srv   usr
wish> path
wish> ls
An error has occurred
wish> exit
```



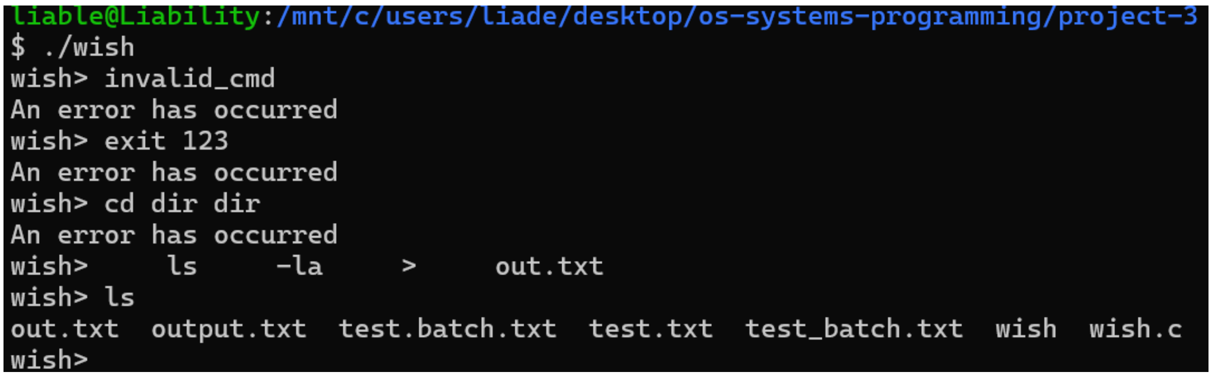
```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
$ ./wish
wish> echo Hello World > test.txt
wish> ls
test.txt  wish  wish.c
wish> cat test.txt
Hello World
wish> ls -la > output.txt
wish> cat output.txt
total 28
drwxrwxrwx 1 liable liable  4096 Jul 26 19:17 .
drwxrwxrwx 1 liable liable  4096 Jul 24 12:15 ..
-rwxrwxrwx 1 liable liable     0 Jul 26 19:17 output.txt
-rwxrwxrwx 1 liable liable    12 Jul 26 19:17 test.txt
-rwxrwxrwx 1 liable liable 17112 Jul 26 18:33 wish
-rwxrwxrwx 1 liable liable  6168 Jul 26 18:33 wish.c
wish> ls -la > > output.txt
An error has occurred
wish>
```



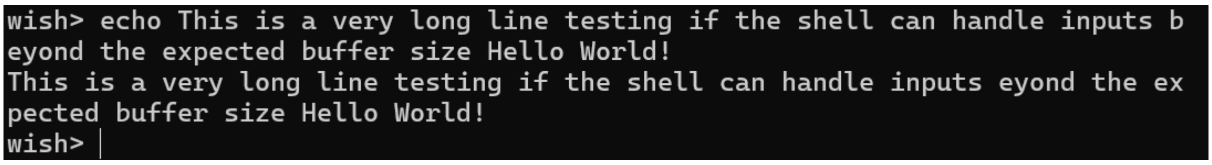
```
wish> sleep 2 & ls & echo Done
Done
output.txt  test.txt  wish  wish.c
wish> wish> sleep 2 & ls & & echo Done
An error has occurred
wish> exit
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
```



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
$ nano test_batch.txt
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
$ cat test.batch.txt
echo Batch Test
ls
pwd
exit
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
$ ./wish test_batch.txt
Batch Test
output.txt  test.batch.txt  test.txt  test_batch.txt  wish  wish.c
/mnt/c/users/liade/desktop/os-systems-programming/project-3
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
$ ./wish nonexistent.txt
An error has occurred
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
```



```
liable@Liability:/mnt/c/users/liade/desktop/os-systems-programming/project-3
$ ./wish
wish> invalid_cmd
An error has occurred
wish> exit 123
An error has occurred
wish> cd dir dir
An error has occurred
wish>     ls     -la     >     out.txt
wish> ls
out.txt  output.txt  test.batch.txt  test.txt  test_batch.txt  wish  wish.c
wish>
```



```
wish> echo This is a very long line testing if the shell can handle inputs b
eyond the expected buffer size Hello World!
This is a very long line testing if the shell can handle inputs eyond the ex
pected buffer size Hello World!
wish>
```