

Simplified UNIX-Like Utilities

Submitted By:

Varshitha Masaram - CS22B1071

P. Naga Sripada - CS22B1018

N. Priyadarshini - CS22B1009

G. Satya Priya - CS22B1012

Date of Submission:

Nov 24, 2024

Contents

1	Objective	2
2	Introduction	2
3	Setting Up the Environment and Creating Files	2
3.1	Creating a Directory	2
3.2	Creating Code Files Using <code>nano</code>	3
3.3	Setting Up Custom Commands with <code>export</code>	3
3.4	Making Changes Persistent with <code>nano</code> <code>/.bashrc</code>	4
3.5	Applying Changes with <code>source</code> <code>/.bashrc</code>	4
3.6	Verifying Configuration with <code>echo</code>	5
3.7	Compiling and Executing Custom Utilities	5
4	Implemented Utilities	5
4.1	<code>custom_ls</code>	5
4.2	Demonstration	6
4.3	<code>custom_cat</code>	8
4.4	Demonstration	8
4.5	<code>custom_grep</code>	9
4.6	Demonstration	10
4.7	<code>custom_wc</code>	10
4.8	Demonstration	11
4.9	<code>custom_cp</code>	11
4.10	Demonstration	11
4.11	<code>custom_mv</code>	12
4.12	Demonstration	13
4.13	<code>custom_rm</code>	13
4.14	Demonstration	13
5	Custom Shell Program	14
6	Conclusion	14

1 Objective

The objective of this project is to develop a series of lightweight and simplified versions of UNIX utilities. These utilities mimic common commands like `cat`, `ls`, `grep`, and others. The purpose is to gain hands-on experience with system programming by implementing these commands from scratch in C.

Additionally, the project involves:

- Developing a **custom shell** to execute the commands interactively.
- Writing a **Makefile** for efficient compilation and testing.
- Documenting the project with execution screenshots and detailed explanations.

2 Introduction

UNIX utilities are essential tools for file and directory management. By re-implementing them from scratch, we learn:

- System calls for file operations like `open`, `read`, `write`, and `close`.
- Directory traversal and metadata extraction using APIs like `opendir` and `stat`.
- String manipulation and pattern matching.
- Error handling and edge case management.

This document provides a detailed description of the implemented utilities, their functionalities, key concepts, and usage examples.

3 Setting Up the Environment and Creating Files

This section describes the steps to set up the environment for the custom UNIX utilities project, including creating directories, writing code files, and configuring the system path for easy execution of the utilities.

3.1 Creating a Directory

To organize the custom UNIX utility programs, a dedicated directory is created.

Command Used:

```
1 mkdir custom_unix_utilities
```

Explanation:

- The `mkdir` command creates a new directory named `custom_unix_utilities`.
- Once created, navigate into this directory using the `cd` command:

```
1 cd custom_unix_utilities
```

```
varshitha@varshitha:~$  
varshitha@Varshitha:~$ mkdir custom_unix_utilities  
varshitha@Varshitha:~$ cd custom_unix_utilities
```

Figure 1: Creating directory

3.2 Creating Code Files Using nano

Code files for the custom utilities can be created and edited using the `nano` text editor.

Command Used:

```
1 nano custom_cat.c
```

Explanation:

- The `nano` command opens the `custom_cat.c` file in the terminal editor.
- Inside the editor:
 - Write the C code for the `custom_cat` utility.
 - Press `CTRL + O` to save the file and `CTRL + X` to exit.
- For other utilities, create files similarly. For example:

```
1 nano custom_ls.c
```

```
varshitha@Varshitha:~/custom_unix_utilities$ nano custom_ls.c  
varshitha@Varshitha:~/custom_unix_utilities$  
varshitha@Varshitha:~/custom_unix_utilities$  
varshitha@Varshitha:~/custom_unix_utilities$ nano custom_cat.c  
varshitha@Varshitha:~/custom_unix_utilities$  
varshitha@Varshitha:~/custom_unix_utilities$ nano custom_grep.c  
varshitha@Varshitha:~/custom_unix_utilities$ nano custom_wc.c  
varshitha@Varshitha:~/custom_unix_utilities$ nano custom_cp.c  
varshitha@Varshitha:~/custom_unix_utilities$ nano custom_mv.c  
varshitha@Varshitha:~/custom_unix_utilities$ nano custom_rm.c  
varshitha@Varshitha:~/custom_unix_utilities$ ls  
custom_cat.c  custom_grep.c  custom_mv.c  custom_wc.c  
custom_cp.c   custom_ls.c   custom_rm.c
```

Figure 2: Creating code files using nano

3.3 Setting Up Custom Commands with `export`

To run the custom utilities directly, the directory path is added to the `PATH` environment variable.

Command Used:

```
1 export PATH=$PATH:/home/varshitha/custom_unix_utilities
```

Explanation:

- The `export` command temporarily appends `/home/varshitha/custom_unix_utilities` to the `PATH`.
- This allows the utilities (e.g., `custom_cat`) to be executed without specifying the full path.
- Verify this by checking the updated `PATH` variable:

```
1 echo $PATH
```

```
varshitha@varshitha:~/custom_unix_utilities$ pwd
/home/varshitha/custom_unix_utilities
varshitha@varshitha:~/custom_unix_utilities$ export PATH=$PATH:/home/varshitha/custom_unix_utilities
varshitha@varshitha:~/custom_unix_utilities$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c/Program Files/Python3
12/Scripts:/mnt/c/Program Files/Python312:/mnt/c/Program Files/Common Files/Oracle/Java/javapath:/mnt/c/Windows/system32:/mnt/c/Wind
ows:/mnt/c/Windows/System32/Wbam:/mnt/c/Windows/System32/WindowsPowerShell/v1.0:/mnt/c/Windows/System32/OpenSSH:/mnt/c/Program Files
/dotnet:/mnt/c/Program Files/Git/cmd:/mnt/c/Program Files/Git/mingw64/bin:/mnt/c/Program Files/Git/usr/bin:/mnt/c/MinGW/bin:/mnt/c/Pr
ogram Files/MySQL/MySQL Server 8.0/bin:/mnt/c/Program Files/Java/jdk-22/bin:/mnt/c/Program Files/nodejs:/mnt/c/ProgramData/chocolatey
/bin:/mnt/c/Program Files/MySQL/MySQL Shell 8.0/bin:/mnt/c/Users/varsh/AppData/Local/Microsoft/WindowsApps:/mnt/c/Program Files/Git/b
in:/mnt/c/MinGW/bin:/mnt/c/Users/varsh/AppData/Local/Programs/Microsoft VS Code/bin:/mnt/c/Users/varsh/AppData/Local/Programs/MiKTeX/m
iktex/bin/x64:/mnt/c/verilog/bin:/mnt/c/verilog/gtkwave/bin:/mnt/c/Users/varsh/AppData/Roaming/npm:/snap/bin:/home/user/custom_unix
_utilities:/home/user/custom_unix_utilities:/home/varshitha/custom_unix_utilities
```

3.4 Making Changes Persistent with `nano` `/.bashrc`

To make the `PATH` changes permanent, update the `/.bashrc` file.

Command Used:

```
1 nano ~/.bashrc
```

Explanation:

- The `/.bashrc` file is a shell script that initializes the terminal environment.
- Add the following line at the end of the file to make the `PATH` update permanent:

```
1 export PATH=$PATH:/home/varshitha/custom_unix_utilities
```

- Save the file and exit the editor.

3.5 Applying Changes with `source` `/.bashrc`

To reload the updated `/.bashrc` file without restarting the terminal:

Command Used:

```
1 source ~/.bashrc
```

Explanation:

- The `source` command applies changes made to the `/.bashrc` file.
- This ensures the updated `PATH` configuration is immediately available.

For ensuring permissions we used `chmod`:

```
varshitha@Varshitha:~/custom_unix_utilities$ nano ~/.bashrc
varshitha@Varshitha:~/custom_unix_utilities$ source ~/.bashrc

varshitha@Varshitha:~/custom_unix_utilities$ chmod +x custom_*
```

3.6 Verifying Configuration with echo

To confirm that the directory has been added to the PATH variable:

Command Used:

```
1 echo $PATH
```

Explanation:

- The echo command displays the current value of the PATH variable.
- Verify that /home/varshitha/custom_unix_utilities is listed.
- Example output:

```
1 /usr/local/bin:/usr/bin:/bin:/home/varshitha/custom_unix_utilities
```

3.7 Compiling and Executing Custom Utilities

After writing the C code, compile the utility and execute it:

Command Used:

```
1 gcc -o custom_cat custom_cat.c
2 custom_cat sample.txt
```

Explanation:

- The gcc command compiles the C code into an executable file.
- The compiled utility (custom_cat) is then executed with a file (sample.txt) as input.

4 Implemented Utilities

4.1 custom_ls

Description: Displays the contents of a directory, similar to the ls command.

Functionalities:

- Lists all files and subdirectories in the specified path.
- Displays file types (e.g., directories, regular files).
- Recursively lists subdirectory contents using the -l flag.
- Displays hidden files using the -a flag.

- Display files with its size in bytes **-s** flag

Key Concepts:

- **Directory Traversal:** Implemented using `opendir`, `readdir`, and `closedir`.
- **File Metadata:** Extracted using the `stat` system call.

Usage:

```
1 custom_ls [options] [path]
2 # Examples:
3 custom_ls           # List files in the current directory.
4 custom_ls -l        # Recursively list files.
5 custom_ls -a        # Show hidden files.
6 custom_ls -s        #List files with its size in bytes
```

4.2 Demonstration

The following figure illustrates how the

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_ls
custom_wc.c
Makefile
custom_cp.c
custom_ls
custom_rm.c
custom_cp
custom_mv.c
custom_mv
custom_wc
custom_cat.c
custom_grep.c
custom_rm
custom_ls.c
custom_cat
custom_grep
```

Figure 3: `custom_ls` demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_ls -a
custom_wc.c
Makefile
custom_cp.c
custom_ls
..
custom_rm.c
sample.txt
custom_cp
custom_mv.c
custom_mv
custom_wc
custom_cat.c
custom_grep.c
custom_rm
custom_ls.c
.
custom_cat
custom_grep
varshitha@Varshitha:~/custom_unix_utilities$ custom_ls -l
custom_wc.c
Makefile
custom_cp.c
custom_ls
custom_rm.c
sample.txt
custom_cp
custom_mv.c
custom_mv
custom_wc
custom_cat.c
custom_grep.c
custom_rm
custom_ls.c
custom_cat
custom_grep
```

Figure 4: custom_ls -l,-a demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_ls -s
custom_wc.c           1383 bytes
Makefile               624 bytes
custom_cp.c            1477 bytes
custom_ls              16376 bytes
custom_rm.c            674 bytes
sample.txt             441 bytes
custom_cp              16376 bytes
custom_mv.c            766 bytes
custom_mv              16200 bytes
custom_wc              16328 bytes
custom_cat.c           910 bytes
custom_grep.c          1560 bytes
custom_rm              16200 bytes
custom_ls.c            1121 bytes
custom_cat              16392 bytes
custom_grep             16496 bytes
varshitha@Varshitha:~/custom_unix_utilities$ █
```

Figure 5: custom_ls -s demonstration

4.3 custom_cat

Description: Reads and displays the contents of files, similar to the `cat` command.

Functionalities:

- Displays the contents of one or more files.
- Concatenates multiple files and displays them together.
- Provides options to display text in uppercase (`-u`) or lowercase (`-l`).

Key Concepts:

- **File Reading:** Implemented using `fopen` and `fgets`.
- **Text Manipulation:** Uses `toupper` and `tolower`.

Usage:

```
1 custom_cat [-u|-l] file1 file2 ...
2 # Examples:
3 custom_cat file1.txt           # Display file contents.
4 custom_cat -u file1.txt        # Display contents in uppercase.
5 custom_cat file1.txt file2.txt # Concatenate and display multiple
                               files.
```

4.4 Demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_cat sample.txt
Hello IIITDM KANCHEEPURAM. This is just a sample text file to test and evaluate all the system utility function calls.
There are 7 system utilities:
1. custom_ls
2. custom_cat
3. custom_grep
4. custom_wc
5. custom_cp
6. custom_mv
7. custom_rm

And we are trying to successfully develop these from scratch and execute and evaluate these.

Thank You,
Varshitha - CS22B1071
Priyadarshini - CS22B1009
Satya Priya - CS22B1012
Sripada - CS22B1018
```

Figure 6: `custom_cat` demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_cat -l sample.txt
Hello IIITDM KANCHEEPURAM. This is just a sample text file to test and evaluate all the system utility function calls.
There are 7 system utilities:
1. custom_ls
2. custom_cat
3. custom_grep
4. custom_wc
5. custom_cp
6. custom_mv
7. custom_rm

and we are trying to successfully develop these from scratch and execute and evaluate these.

thank you,
varshitha - cs22b1071
priyadarshini - cs22b1009
satya priya - cs22b1012
sripada - cs22b1018
```

Figure 7: `custom_cat -l` demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_cat -u sample.txt
HELLO IIITDM KANCHEEPURAM. THIS IS JUST A SAMPLE TEXT FILE TO TEST AND EVALUATE ALL THE SYSTEM UTILITY FUNCTION CALLS.
THERE ARE 7 SYSTEM UTILITIES:
1. CUSTOM_LS
2. CUSTOM_CAT
3. CUSTOM_GREP
4. CUSTOM_WC
5. CUSTOM_CP
6. CUSTOM_MV
7. CUSTOM_RM

AND WE ARE TRYING TO SUCCESFULLY DEVELOP THESE FROM SCRATCH AND EXECUTE AND EVALUATE THESE.

THANK YOU
VARSHITHA - CS22B1071
PRIYADARSHINI - CS22B1009
SATYA PRIYS - CS22B1012
SRIPADA - CS22B1018
```

Figure 8: custom_cat -u demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_cat -l sample2.txt
*****
-----
this is second sample text file used for
custom_cat and custom_cp like those utilities
for ensuring concatenation and copying the files

*****
```

Figure 9: custom_cat -l demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_cat -u sample2.txt
*****
-----
THIS IS SECOND SAMPLE TEXT FILE USED FOR
CUSTOM_CAT AND CUSTOM_CP LIKE THOSE UTILITIES
FOR ENSURING CONCATENATION AND COPYING THE FILES

*****
```

Figure 10: custom_cat -u demonstration

4.5 custom_grep

Description: Searches for a pattern in files, similar to the grep command.

Functionalities:

- Searches for lines matching a given pattern.
- Supports case-insensitive search (-i).
- Displays lines that do not match the pattern using the -v flag.

Key Concepts:

- **Pattern Matching:** Implemented using `strstr`.
- **Case Insensitivity:** Achieved by converting strings to lowercase using `tolower`.

Usage:

```
1 custom_grep [-i|-v] pattern file1 file2 ...
2 # Examples:
3 custom_grep "pattern" file1.txt      # Search for a pattern in file1.txt
4 .
5 custom_grep -i "pattern" file1.txt    # Perform case-insensitive search.
6 custom_grep -v "pattern" file1.txt    # Perform case-insensitive search
7 and displays all the lines which are not matched.
```

4.6 Demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_grep -i "satya" sample.txt
Satya Priya - CS22B1012
```

Figure 11: custom_grep -i demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_grep -v "Satya" sample.txt
Hello IIITDM KANCHEEPURAM. This is just a sample text file to test and evaluate all the system utility function calls.
There are 7 system utilities:
1. custom_ls
2. custom_cat
3. custom_grep
4. custom_wc
5. custom_cp
6. custom_mv
7. custom_rm

And we are trying to successfully develop these from scratch and execute and evaluate these.

Thank You,
Varshitha - CS22B1071
Priyadarshini - CS22B1009
Sripada - CS22B1018
varshitha@Varshitha:~/custom_unix_utilities$
```

Figure 12: custom_grep -v demonstration

4.7 custom_wc

Description: Counts words, lines, and characters in a file, similar to the `wc` command.

Functionalities:

- Counts and displays the number of lines, words, and characters in a file.
- Processes multiple files with individual and total counts.

Usage:

```
1 custom_wc file1 file2 ...
2 # Examples:
3 custom_wc file1.txt                  # Count lines, words, and
4     characters in file1.txt.
5 custom_wc file1.txt file2.txt        # Count for multiple files.
```

4.8 Demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_wc sample.txt
Lines: 17
Words: 69
Characters: 441
varshitha@Varshitha:~/custom_unix_utilities$ custom_wc -l sample.txt
Lines: 17
varshitha@Varshitha:~/custom_unix_utilities$ custom_wc -w sample.txt
Words: 69
varshitha@Varshitha:~/custom_unix_utilities$ custom_wc -c sample.txt
Characters: 441
varshitha@Varshitha:~/custom_unix_utilities$
```

Figure 13: custom_wc demonstration

4.9 custom_cp

Description: Copies files or directories, similar to the `cp` command.

Functionalities:

- Copies a single file from source to destination.
- Recursively copies directories using the `-r` flag.

Usage:

```
1 custom_cp [-r] source destination
2 # Examples:
3 custom_cp file1.txt file2.txt          # Copy file1.txt to file2.txt.
4 custom_cp -r dir1 dir2                # Copy dir1 to dir2 recursively.
```

4.10 Demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ ls
Makefile  custom_cat.c  custom_cp.c  custom_grep.c  custom_ls.c  custom_mv.c  custom_rm.c  custom_wc.c  sample2.txt
custom_cat  custom_cp  custom_grep  custom_ls  custom_mv  custom_rm  custom_wc  sample.txt  samplecopy.txt
varshitha@Varshitha:~/custom_unix_utilities$
```

Figure 14: Performing ls before executing custom_mv utility

Here we performed custom_mv. We can see that after ls it has listed the newly created file file_cpy.txt

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_cp sample.txt file_copy.txt
varshitha@Varshitha:~/custom_unix_utilities$ ls
Makefile  custom_cat.c  custom_cp.c  custom_grep.c  custom_ls.c  custom_mv.c  custom_rm.c  custom_wc.c  sample.txt  samplecopy.txt
custom_cat  custom_cp  custom_grep  custom_ls  custom_mv  custom_rm  custom_wc  file_copy.txt  sample2.txt
varshitha@Varshitha:~/custom_unix_utilities$
```

Figure 15: Performing ls after executing custom_mv utility

Here we can see that after performing custom_mv it has displayed the original file content in the newly copied file(file_copy.txt) using custom_cat

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_cat file_copy.txt
Hello IIITDM KANCHEEPURAM. This is just a sample text file to test and evaluate all the system utility function calls.
There are 7 system utilities:
1. custom_ls
2. custom_cat
3. custom_grep
4. custom_wc
5. custom_cp
6. custom_mv
7. custom_rm

And we are trying to successfully develop these from scratch and execute and evaluate these.

Thank You,
Varshitha - CS22B1071
Priyadarshini - CS22B1009
Satya Priya - CS22B1012
Sripada - CS22B1018
varshitha@Varshitha:~/custom_unix_utilities$
```

Figure 16: Copied file(file_copy.txt) content

The same procedure is performed with directories. Below are the images for demonstration for directories.

```
varshitha@Varshitha:~/custom_unix_utilities$ ls
Makefile  custom_cat.c  custom_cp.c  custom_grep.c  custom_ls.c  custom_mv.c  custom_rm.c  custom_wc.c  file_copy.txt  sample2.txt  samplecopy.txt
custom_cat  custom_cp  custom_grep  custom_ls  custom_mv  custom_rm  custom_wc  directory_copy  sample.txt  sample_for_copy
```

Figure 17: ls before copying

Here we can see that after performing custom_cp the new directory is listed.

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_cp -r sample_for_copy copy_directory
varshitha@Varshitha:~/custom_unix_utilities$ ls
Makefile  custom_cat  custom_cp  custom_grep  custom_ls  custom_mv  custom_rm  custom_wc  directory_copy  sample.txt  sample_for_copy
copy_directory  custom_cat.c  custom_cp.c  custom_grep.c  custom_ls.c  custom_mv.c  custom_rm.c  custom_wc.c  file_copy.txt  sample2.txt  samplecopy.txt
```

Figure 18: ls after copying

Here we can see that the same original file is there inside newly created file and also the content in this is same as original file content.

```
varshitha@Varshitha:~/custom_unix_utilities$ cd copy_directory
varshitha@Varshitha:~/custom_unix_utilities/copy_directory$ ls
dir_file_copy.txt
varshitha@Varshitha:~/custom_unix_utilities/copy_directory$ custom_cat dir_file_copy.txt
This is for simulating custom_cp for directories. This is a sample file inside the source directory.
varshitha@Varshitha:~/custom_unix_utilities/copy_directory$
```

Figure 19: After copying the directory

4.11 custom_mv

Description: Moves or renames files and directories, similar to the mv command.

Functionalities:

- Renames a file or directory.
- Moves a file or directory to a new location.

Usage:

```
1  custom_mv  source  destination
2  # Examples:
3  custom_mv  old.txt  new.txt          # Rename old.txt to new.txt.
4  custom_mv  dir1  /home/user/dir2      # Move dir1 to a new location.
```

4.12 Demonstration

```
varshitha@Varshitha:~/custom_unix_utilities$ ls
Makefile      custom_cat  custom_cp  custom_grep  custom_ls  custom_mv  custom_rm  custom_wc  directory_copy  sample.txt  sample_for_copy
copy_directory  custom_cat.c  custom_cp.c  custom_grep.c  custom_ls.c  custom_mv.c  custom_rm.c  custom_wc.c  directory_copy.txt  sample2.txt  samplecopy.txt
```

Figure 20: ls before performing moving

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_mv sample.txt sample2.txt
varshitha@Varshitha:~/custom_unix_utilities$ ls
Makefile      custom_cat.c  custom_grep  custom_ls.c  custom_rm  custom_wc.c  sample2.txt
copy_directory  custom_cp  custom_grep.c  custom_mv  custom_rm.c  directory_copy  sample_for_copy
custom_cat      custom_cp.c  custom_ls  custom_mv.c  custom_wc  file_copy.txt  samplecopy.txt
```

Figure 21: ls after performing moving

Here we can see that the sample.txt content is moved into sample2.txt.

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_cat sample2.txt
Hello IIITDM KANCHEEPURAM. This is just a sample text file to test and evaluate all the system utility function calls.
There are 7 system utilities:
1. custom_ls
2. custom_cat
3. custom_grep
4. custom_wc
5. custom_cp
6. custom_mv
7. custom_rm

And we are trying to successfully develop these from scratch and execute and evaluate these.

Thank You,
Varshitha - CS22B1071
Priyadarshini - CS22B1009
Satya Priya - CS22B1012
Sripada - CS22B1018
varshitha@Varshitha:~/custom_unix_utilities$
```

Figure 22: Displaying content of sample2.txt

4.13 custom_rm

Description: Deletes files or directories, similar to the `rm` command.

Functionalities:

- Deletes files.
- Recursively deletes directories using the `-r` flag.

Usage:

```
1 custom_rm [-r] file_or_directory
2 # Examples:
3 custom_rm file1.txt                      # Delete file1.txt.
4 custom_rm -r dir1                         # Recursively delete dir1.
```

4.14 Demonstration

Here we can see that after performing `custom_rm` `file_copy.txt` that file is not listed which represents it has removed

```
varshitha@Varshitha:~/custom_unix_utilities$ custom_rm file_copy.txt
varshitha@Varshitha:~/custom_unix_utilities$ ls
Makefile      custom_cat  custom_cp  custom_grep  custom_ls  custom_mv  custom_rm  custom_wc  directory_copy  sample.txt  sample_for_copy
copy_directory  custom_cat.c  custom_cp.c  custom_grep.c  custom_ls.c  custom_mv.c  custom_rm.c  custom_wc.c  directory_copy.txt  sample2.txt  samplecopy.txt
varshitha@Varshitha:~/custom_unix_utilities$
```

Figure 23: ls after performing `custom_rm`

5 Custom Shell Program

A custom shell program was developed to execute these utilities interactively. Users can type commands to invoke the utilities, which are processed and executed by the shell.

6 Conclusion

This project provided hands-on experience with system programming concepts and file operations. By re-implementing standard UNIX commands, we gained a deeper understanding of their functionality and inner workings.