CSUH-3010
Operating Systems
Eliseo Ferrante, Dena Ahmed
October 3, 2024

Project Phase 1

Description of the implementation

The implementation of the shell is based on fundamental concepts of process creation, inter-process communication, and redirection in Unix-based operating systems. Below is a more detailed breakdown of how the shell works:

- Main Program Loop: The program enters an infinite loop to continually prompt the user for input. The function display_shell_prompt() prints a custom shell prompt (my_shell>), followed by read_user_input() to capture the user's input. The loop continues until the user exits the shell (via the built-in exit command).
- Command Parsing: Once a command is entered, it is first checked for the presence of pipes (|). The function split_piped_commands() splits the command line into separate commands if pipes are detected. This function uses strtok() to tokenize the command string based on the | delimiter, allowing for support of up to three piped commands.
  If no pipes are present, the shell processes the command as a single command. The function parse_shell_command() is used to break down the command line into arguments and detect any redirection symbols (<, >, >>). This function skips leading whitespace, processes arguments, and identifies input/output files for redirection. The parsed arguments are stored in a ShellCommand structure, which also holds information about input/output files and the mode of redirection (overwrite or append).
- Handling Built-in Commands: After parsing the command, it is checked against a set of built-in commands, like cd and exit. The is_built_in_command() function handles these commands:
  - cd (Change Directory): The chdir() system call is used to change the current working directory. If the user does not provide a directory argument, an error message is displayed.
  - exit: The shell terminates by calling exit(0).
- Executing External Commands: For commands that are not built-in, the shell uses fork() to create a child process. The child process then uses execvp() to execute the specified command. This allows the shell to run external programs as subprocesses, just like a normal Linux shell.
  - In the child process:

    - Input Redirection: If the command contains input redirection (<), the file specified is opened using open(), and dup2() is used to redirect the file to stdin (standard input).
    - Output Redirection: If the command contains output redirection (> or >>), the specified output file is opened using open(). The file is either overwritten (>) or appended to (>>) depending on the mode. The output is redirected to stdout (standard output) using dup2(). Once redirection is set up, the execvp() function replaces the current process image with the new program specified by the user command. If the command fails to execute, an error message is displayed using perror().
- Handling Piped Commands: When the command line contains pipes, multiple commands need to be connected using pipes. The function execute_piped_commands() is responsible for creating the necessary pipes and coordinating communication between the commands:
    - Pipe Creation: For each pipe (|), a pair of file descriptors is created using the pipe() system call. These descriptors are used for inter-process communication.
    - Forking and Execution: For each command in the pipeline, a new child process is created using fork(). Depending on its position in the pipeline, the child process redirects stdin or stdout to the appropriate pipe. For example, the output of the first command is redirected to the input of the second command using dup2().
    - Closing Pipes: After each command is executed, the pipe file descriptors are closed to prevent resource leaks.
    - Waiting for Completion: The parent process waits for all child processes to complete using wait(), ensuring that the pipeline is executed in the correct sequence.
- Error Handling: Throughout the program, error handling is integrated to deal with common issues such as:
    - Invalid Commands: If a command is not recognized or fails to execute, an appropriate error message is printed using perror().
    - Redirection Errors: If a file specified for redirection cannot be opened, the shell prints an error and skips execution of the command.
    - Pipe Failures: If the pipe creation or process forking fails, the shell reports the error and terminates the relevant processes.

Usage

To use the shell program, first, compile the source code using a C compiler like gcc. For example, you can compile the file by running the following command in the terminal:

- gcc shell.c -o shell

Once compiled, execute the program by typing:

- ./shell

The shell will display a prompt (my_shell>) where you can type commands. The shell supports a variety of Linux commands, such as ls, pwd, mkdir, rm, etc. Additionally, it can handle input and output redirection using < and >, as well as piping commands using |. For example, you can use:

- ls - to list files in the current directory.
- pwd - to print the current working directory.
- mkdir - to create a new directory.

For redirection, input can be taken from a file using <, like:

- sort < input.txt

Output can be redirected to a file using >, like:

- ls > output.txt

Pipes are supported to connect the output of one command to the input of another. For example:

- ls | grep ".c" | wc -l

The shell also includes built-in commands such as cd for changing directories and exit to exit the shell.

Testing

#	General Command	Command Variation	Expected Outcome	Actual Outcome	Result
1	cd	cd ..	Move up one directory level.	Figure 1	Passed
		cd ./phase-1	Change directory to the phase-1 subdirectory.	Figure 2	Passed
		cd .	Navigate to the current directory, i.e., nothing changes.	Figure 3	Passed
2	exit	exit	Exit the terminal session or shell.	Figure 47	Passed
3	ls	ls	List files and directories in the current directory.	Figure 4	Passed
		ls -l	List files with detailed	Figure 5	Passed

			information (permissions, size, modification date, etc.).		
		ls ./phase-1	List files in the phase-1 subdirectory.	Figure 6	Passed
		ls -alh	List all files including hidden ones, in human-readable sizes	Figure 7	Passed
		ls -l > files.txt	Redirect the output of ls -l to the files.txt file.	Figure 8	Passed
		ls | grep ".c"	Pipe output to grep to filter .c files.	Figure 9	Passed
4	pwd	pwd	Print the current working directory.	Figure 10	Passed
		pwd > current_dir.txt	Redirect the current directory path to current_dir.txt.	Figure 11	Passed
		pwd -P	Print the current working directory with the physical path (resolving symbolic links).	Figure 12	Passed
5	mkdir	mkdir demo	Create a new directory named demo.	Figure 13	Passed
		mkdir -p dir1/dir2	Create nested directories dir1 and dir2.	Figure 14	Passed
6	touch	touch newfile.txt	Create an empty file named newfile.txt, or update its timestamp if it already exists.	Figure 15	Passed
7	echo	echo "Hello, Project!”	Print the message “Hello, Project!” to the terminal.	Figure 16	Passed
		echo "Hello, Project!" > hello.txt	Write the message “Hello, Project!” to the file hello.txt, overwriting any existing content.	Figure 17	Passed

8	cat	cat combined.txt	Display the contents of combined.txt.	Figure 19	Passed
		cat file1.txt file2.txt > combined.txt	Combine the contents of file1.txt and file2.txt into combined.txt.	Figure 18 and Figure 19	Passed
		cat combined.txt | grep "Hello"	Displays the lines from combined.txt that contain the specified "Hello".	Figure 20	Passed
		cat > newfile.txt	Create or overwrite newfile.txt with input provided from the terminal..	Figure 21 and Figure 22	Passed
9	rm	rm demofile.txt	Remove the file named demofile.	Figure 23	Passed
		rm -r dir1	Recursively delete the directory dir1 and all its contents.	Figure 24	Passed
		rm -f hello.txt	Forcefully delete hello.txt	Figure 25	Passed
		rm -f combined.txt file1.txt file2.txt	Forcefully delete multiple files (combined.txt, file1.txt, and file2.txt).	Figure 26	Passed
10	grep	grep "fine" newfile.txt	Search for lines in newfile.txt containing the word "fine".	Figure 27	Passed
		grep -i "FINE" newfile.txt	Perform a case-insensitive search for "FINE" in newfile.txt.	Figure 28	Passed
		ls -l | grep "^d"	List directories (lines starting with "d") in the current directory in a detailed format.	Figure 29	Passed
		cat newfile.txt | grep "fine" | wc -l	Count and display the number of lines in newfile.txt that contain the word "fine".	Figure 30	Passed

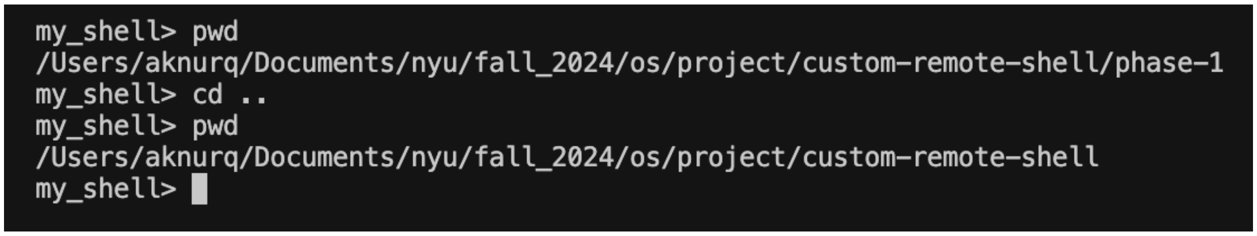
11	mv	mv oldfile.txt newfile.txt	Rename or move oldfile.txt to newfile.txt.	Figure 31	Passed
		mv newfile.txt ./phase-1	Move the file to a new directory	Figure 32	Passed
12	cp	cp newfile.txt backupfile.txt	Copy newfile.txt to backupfile.txt.	Figure 33	Passed
		cp -r phase-1 phase-1-copy	Recursively copy directory phase-1 to phase-1-copy.	Figure 34	Passed
13	tail	tail newfile.txt	Display the last 10 lines of newfile.txt.	Figure 35	Passed
		tail -n 2 newfile.txt	Display the last 2 lines of newfile.txt.	Figure 35	Passed
14	head	head newfile.txt	Display the first 10 lines of newfile.txt.	Figure 36	Passed
		head -n 2 newfile.txt	Display the first 2 lines of newfile.txt.	Figure 36	Passed
		ps aux | head -n 3	Display the first 3 lines of the output from the ps aux command, which shows detailed information about all running processes on the system	Figure 37	Passed
15	wc	wc newfile.txt	Display the number of lines, words, and characters in newfile.txt.	Figure 38	Passed
		wc -l newfile.txt	Display the number of lines in newfile.txt.	Figure 38	Passed
		ls | wc -l	Count and display the total number of files and directories in the current directory.	Figure 39	Passed
		grep "fine" newfile.txt | wc -w	Count and display the total number of words in the lines from newfile.txt that	Figure 40	Passed

			contain the word "fine".		
16	sort	sort unsorted.txt	Sort the contents of unsorted.txt in alphabetical order	Figure 41	Passed
		sort -r unsorted.txt	Sort the contents of unsorted.txt in reverse alphabetical order.	Figure 42	Passed
		sort unsorted.txt > sorted.txt	Sort unsorted.txt and save the result in sorted.txt.	Figure 43	Passed
		cat unsorted.txt | sort | uniq	Sort the contents of unsorted.txt and then remove any duplicate lines, displaying only unique lines.	Figure 44	Passed
17	ps	ps	Display information about currently running processes.	Figure 45	Passed
		ps aux	Display detailed information about all running processes on the system.	Figure 46	Passed
18	clear	clear	Clear the terminal screen.	Terminal is cleared.	Passed
19	1 pipe	grep "fine" newfile.txt | wc -w	Count and display the total number of words in the lines from newfile.txt that contain the word "fine".	Figure 40	Passed
20	2 pipes	cat unsorted.txt | sort | uniq	Sort the contents of unsorted.txt and then remove any duplicate lines, displaying only unique lines.	Figure 44	Passed
21	3 pipes	ps aux | grep "user" | sort -k3 -n | head -n 5	Filters the list of running processes (ps aux) for those that contain "user," sorts them by the 3rd	Figure 48	Passed

			column (usually CPU usage) in numerical order, and then displays the first 5 lines of the sorted result.		
22	Input Redirection	wc -l < file.txt	Counts the number of lines in file.txt.	Figure 49	Passed
23	Output Redirection	sort unsorted.txt > sorted.txt	Sort unsorted.txt and save the result in sorted.txt.	Figure 43	Passed

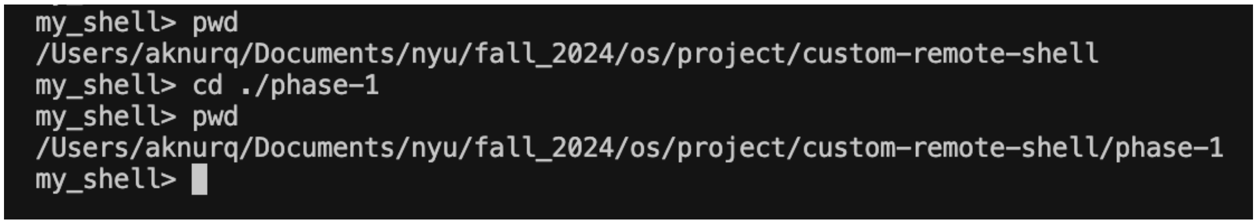
Screenshots of the tests

- cd



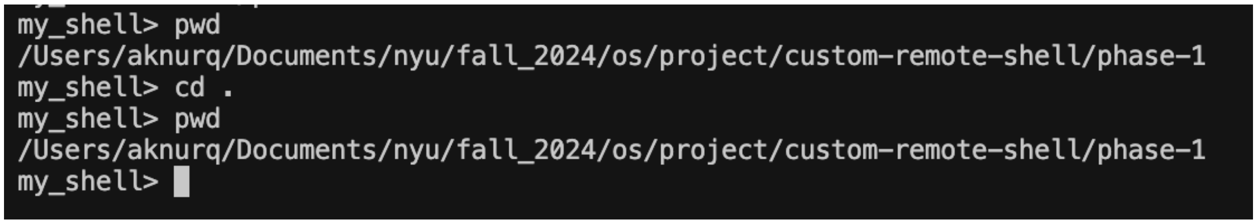
```
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1
my_shell> cd ..
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell
my_shell>
```

Figure 1



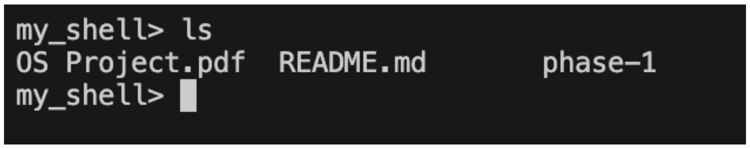
```
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell
my_shell> cd ./phase-1
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1
my_shell>
```

Figure 2



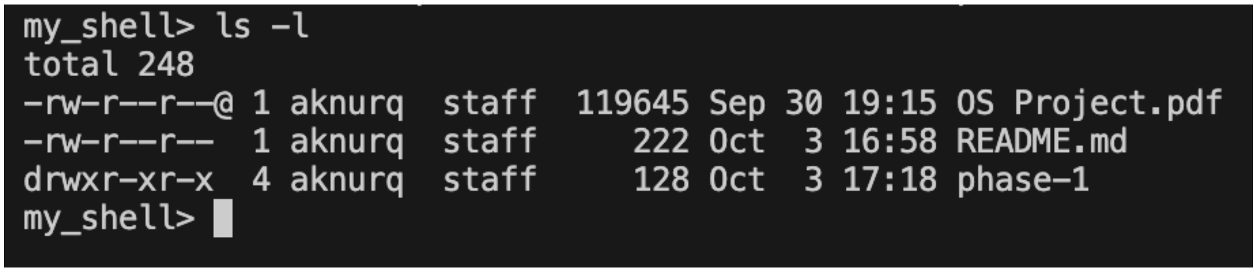
```
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1
my_shell> cd .
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1
my_shell>
```

Figure 3



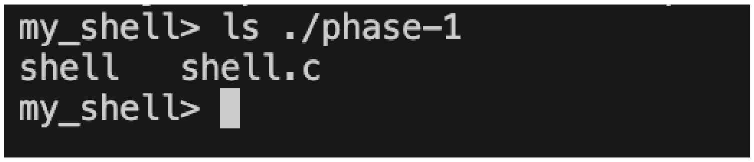
```
my_shell> ls
OS Project.pdf  README.md       phase-1
my_shell> 
```

Figure 4



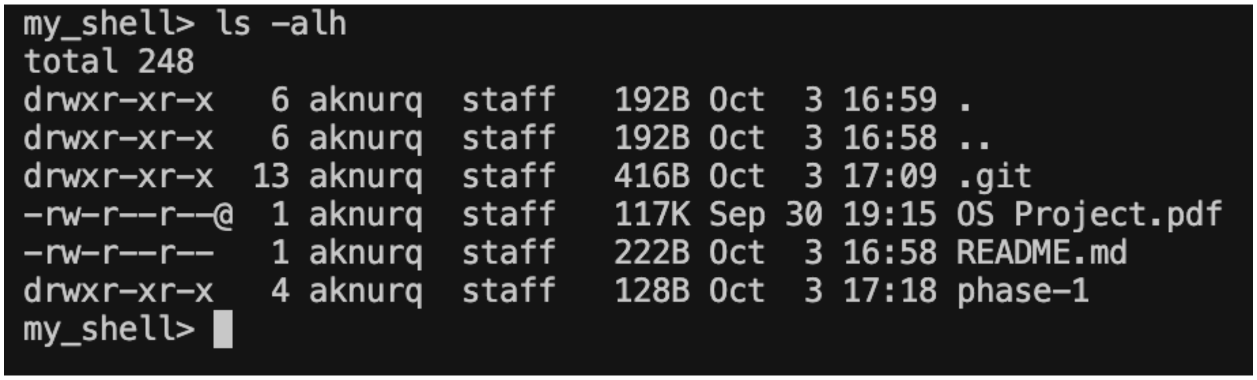
```
my_shell> ls -l
total 248
-rw-r--r--@ 1 aknurq  staff  119645 Sep 30 19:15 OS Project.pdf
-rw-r--r--  1 aknurq  staff     222 Oct  3 16:58 README.md
drwxr-xr-x  4 aknurq  staff     128 Oct  3 17:18 phase-1
my_shell> 
```

Figure 5



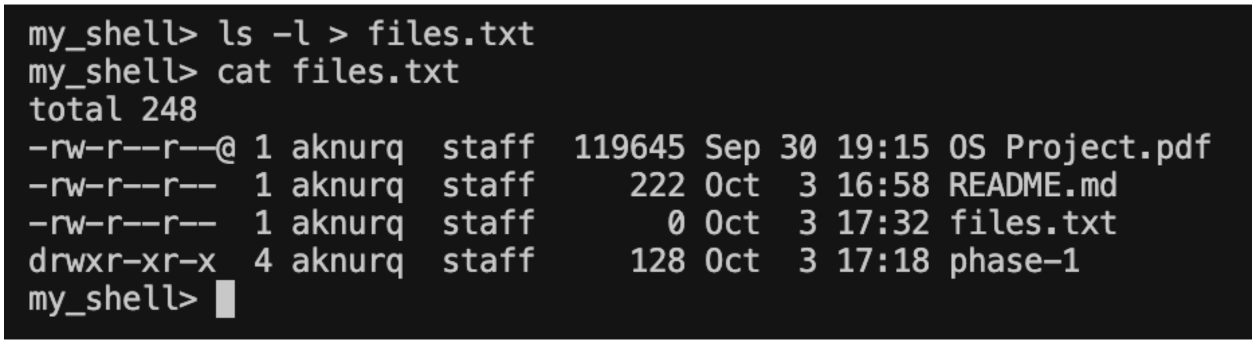
```
my_shell> ls ./phase-1
shell   shell.c
my_shell> 
```

Figure 6



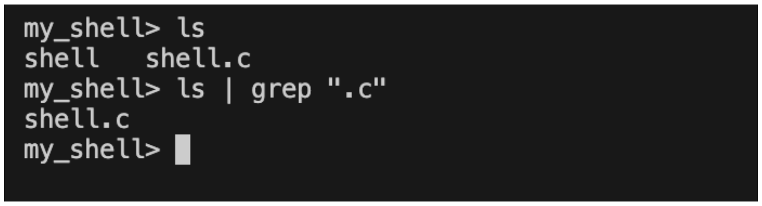
```
my_shell> ls -alh
total 248
drwxr-xr-x   6 aknurq  staff   192B Oct  3 16:59 .
drwxr-xr-x   6 aknurq  staff   192B Oct  3 16:58 ..
drwxr-xr-x  13 aknurq  staff   416B Oct  3 17:09 .git
-rw-r--r--@  1 aknurq  staff   117K Sep 30 19:15 OS Project.pdf
-rw-r--r--   1 aknurq  staff   222B Oct  3 16:58 README.md
drwxr-xr-x   4 aknurq  staff   128B Oct  3 17:18 phase-1
my_shell> 
```

Figure 7



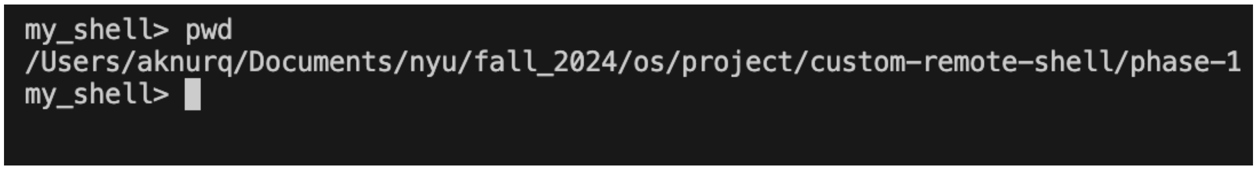
```
my_shell> ls -l > files.txt
my_shell> cat files.txt
total 248
-rw-r--r--@ 1 aknurq  staff  119645 Sep 30 19:15 OS Project.pdf
-rw-r--r--  1 aknurq  staff     222 Oct  3 16:58 README.md
-rw-r--r--  1 aknurq  staff       0 Oct  3 17:32 files.txt
drwxr-xr-x  4 aknurq  staff     128 Oct  3 17:18 phase-1
my_shell> 
```

Figure 8



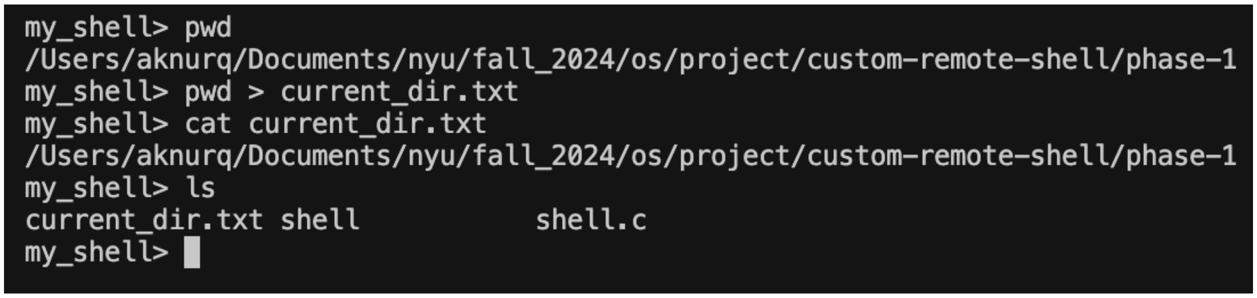
```
my_shell> ls
shell   shell.c
my_shell> ls | grep ".c"
shell.c
my_shell> 
```

Figure 9



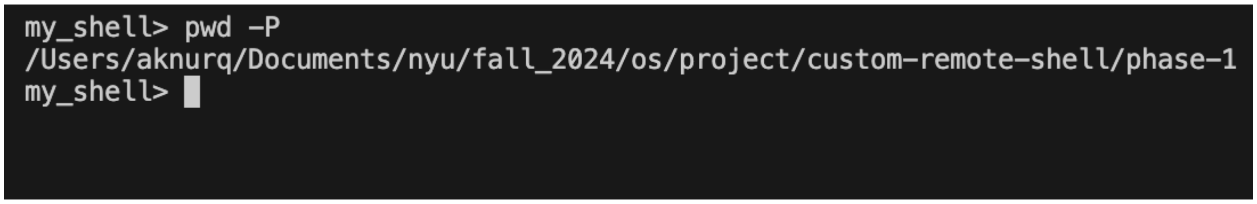
```
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1
my_shell> 
```

Figure 10



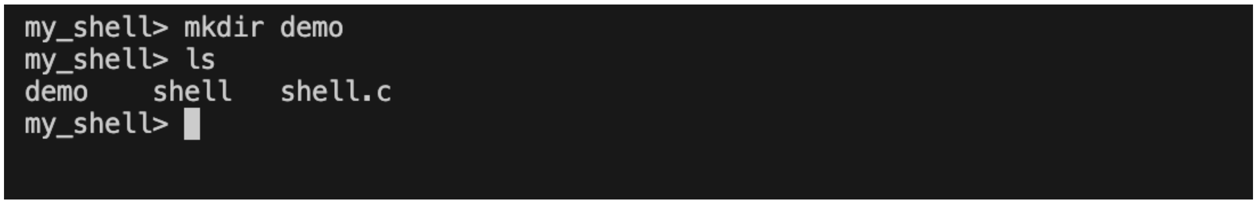
```
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1
my_shell> pwd > current_dir.txt
my_shell> cat current_dir.txt
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1
my_shell> ls
current_dir.txt shell           shell.c
my_shell> 
```

Figure 11



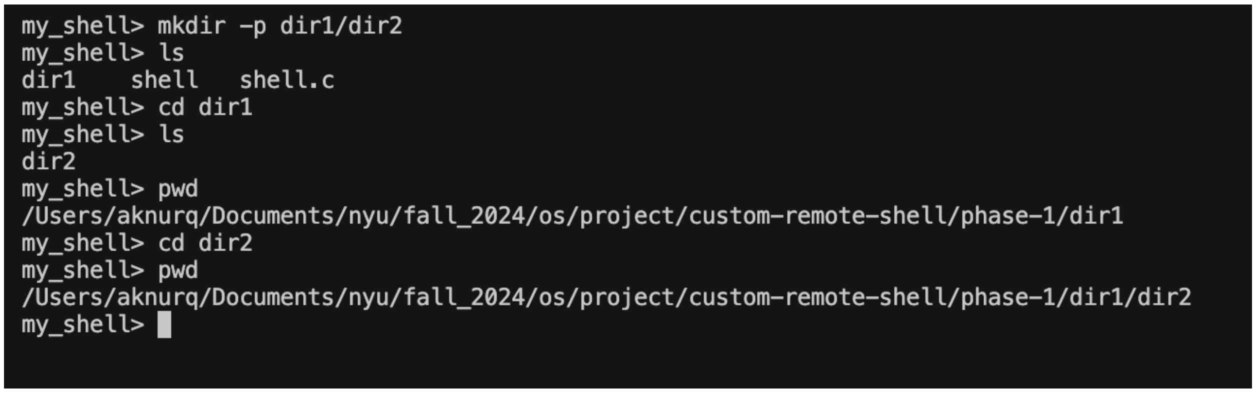
```
my_shell> pwd -P
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1
my_shell> 
```

Figure 12



```
my_shell> mkdir demo
my_shell> ls
demo    shell   shell.c
my_shell> 
```

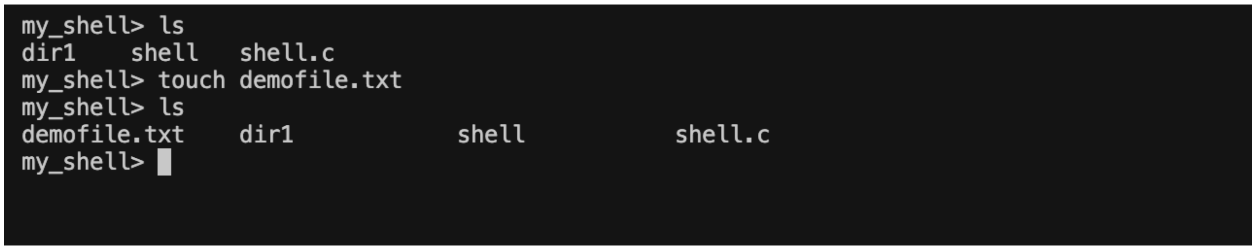
Figure 13



```
my_shell> mkdir -p dir1/dir2
my_shell> ls
dir1    shell   shell.c
my_shell> cd dir1
my_shell> ls
dir2
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1/dir1
my_shell> cd dir2
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1/dir1/dir2
my_shell> 
```

Figure 14

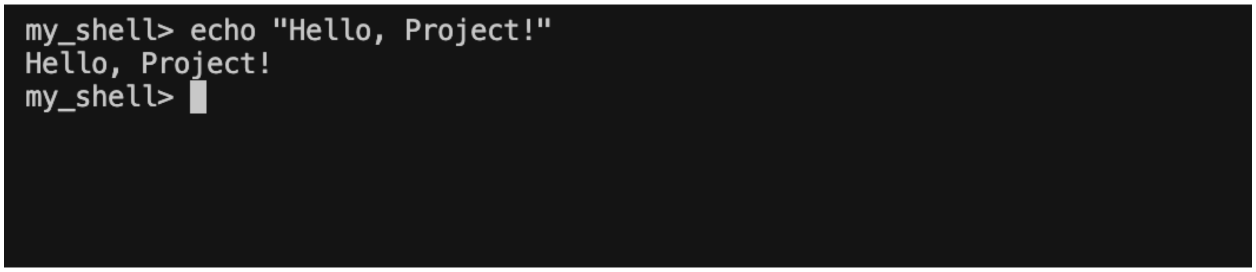
touch



```
my_shell> ls
dir1    shell   shell.c
my_shell> touch demofile.txt
my_shell> ls
demofile.txt    dir1            shell           shell.c
my_shell> 
```

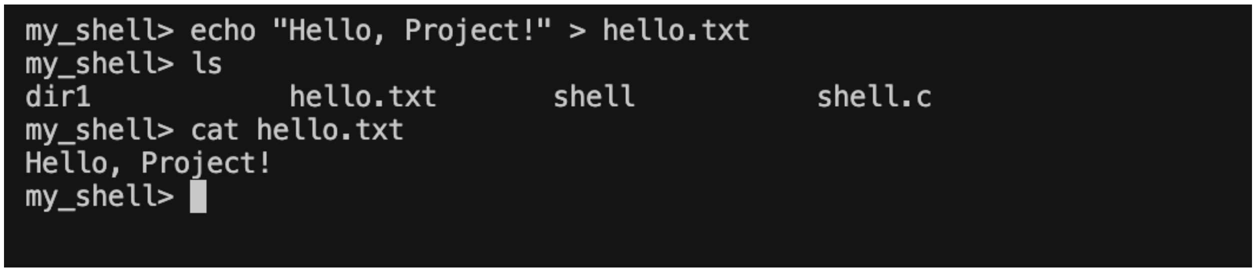
Figure 15

echo



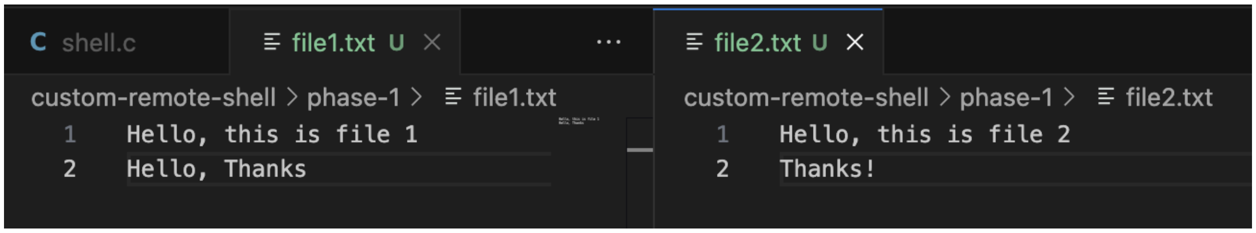
```
my_shell> echo "Hello, Project!"
Hello, Project!
my_shell> 
```

Figure 16



```
my_shell> echo "Hello, Project!" > hello.txt
my_shell> ls
dir1            hello.txt       shell           shell.c
my_shell> cat hello.txt
Hello, Project!
my_shell> 
```

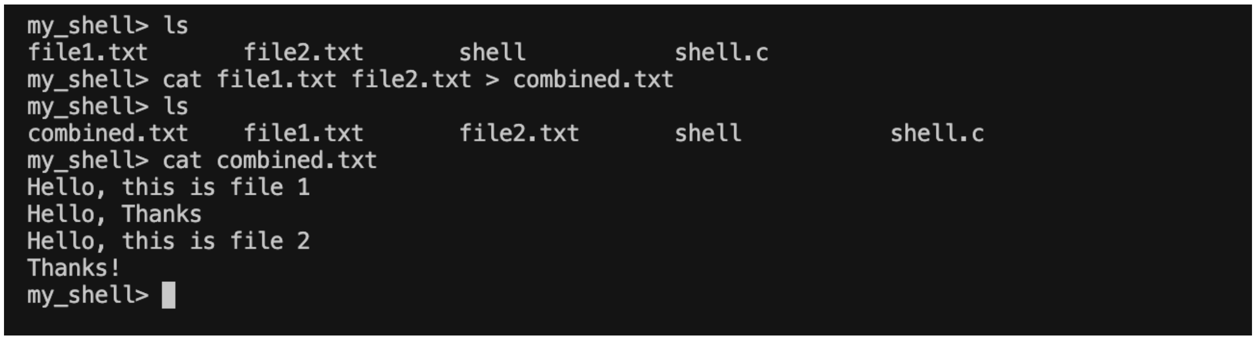
Figure 17


shell.c
file1.txt U
file2.txt U
custom-remote-shell > phase-1 > file1.txt
custom-remote-shell > phase-1 > file2.txt

```
Hello, this is file 1
Hello, Thanks
```

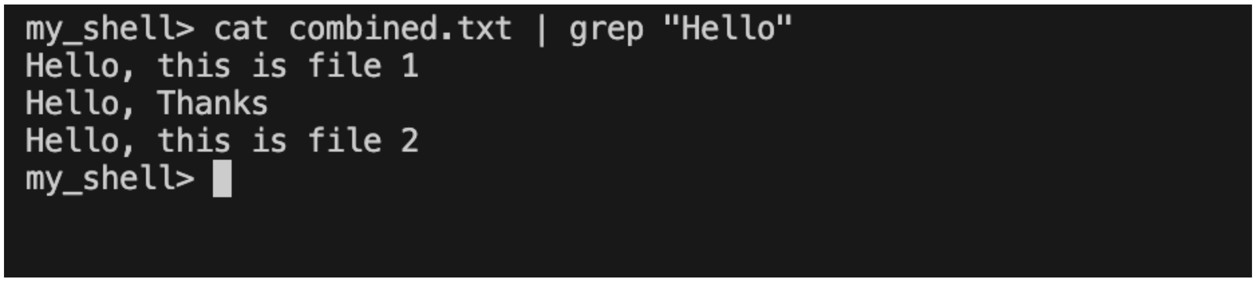
```
Hello, this is file 2
Thanks!
```

Figure 18



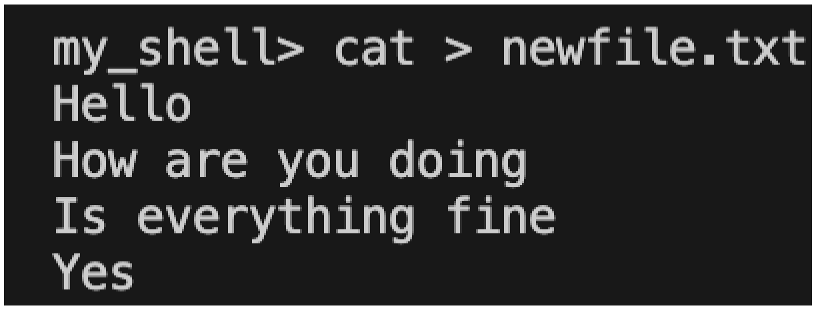
```
my_shell> ls
file1.txt       file2.txt       shell           shell.c
my_shell> cat file1.txt file2.txt > combined.txt
my_shell> ls
combined.txt    file1.txt       file2.txt       shell           shell.c
my_shell> cat combined.txt
Hello, this is file 1
Hello, Thanks
Hello, this is file 2
Thanks!
my_shell> 
```

Figure 19



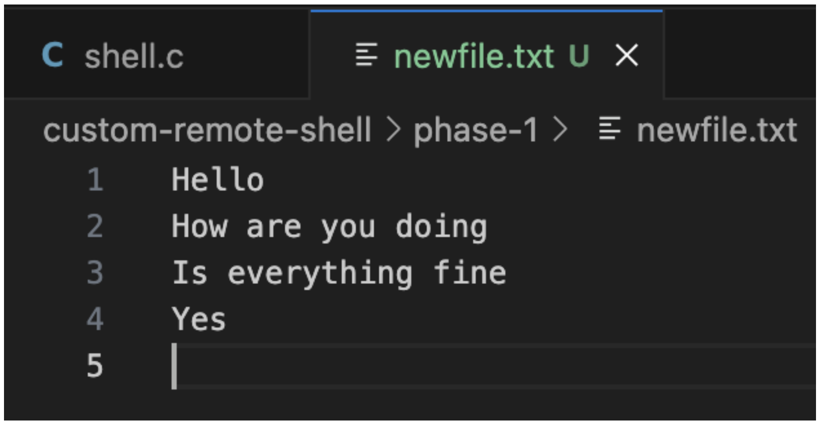
```
my_shell> cat combined.txt | grep "Hello"
Hello, this is file 1
Hello, Thanks
Hello, this is file 2
my_shell> 
```

Figure 20



```
my_shell> cat > newfile.txt
Hello
How are you doing
Is everything fine
Yes
```

Figure 21

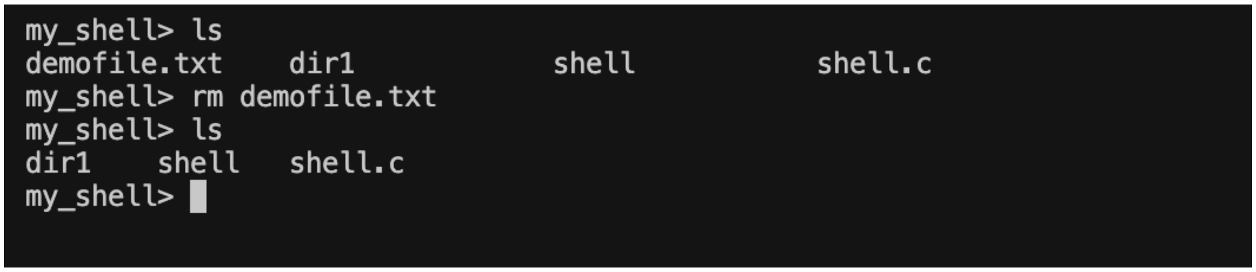


```
Hello
How are you doing
Is everything fine
Yes

```

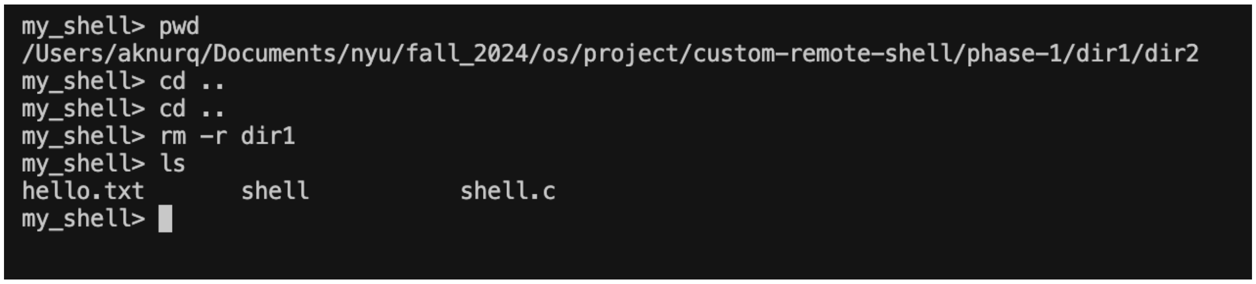
Figure 22

rm



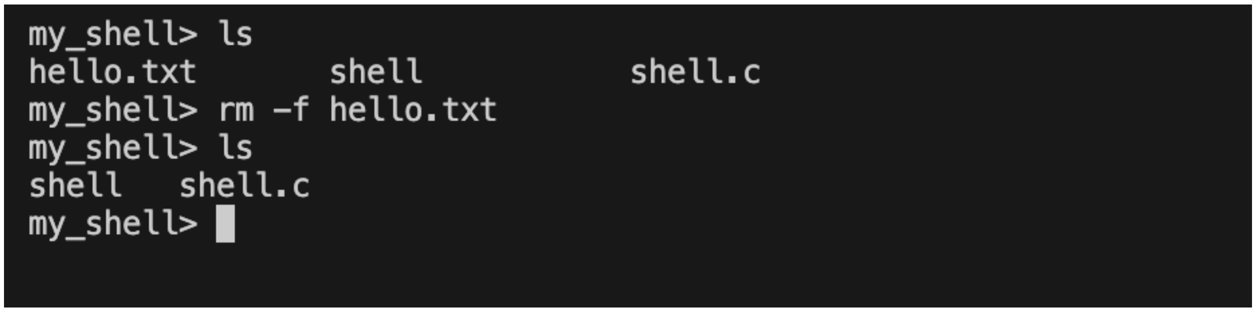
```
my_shell> ls
demofile.txt    dir1            shell           shell.c
my_shell> rm demofile.txt
my_shell> ls
dir1    shell   shell.c
my_shell> 
```

Figure 23



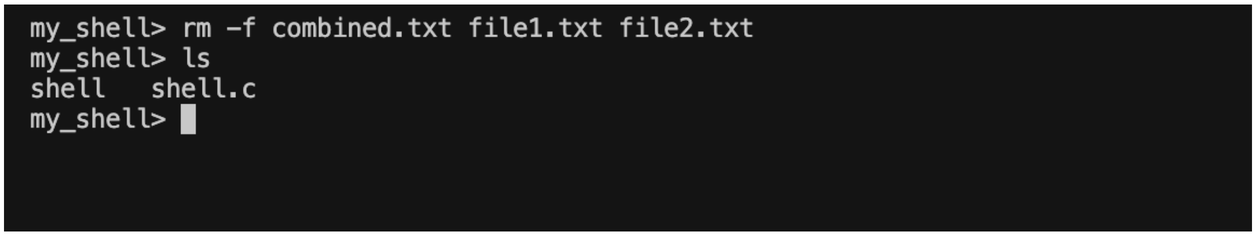
```
my_shell> pwd
/Users/aknurq/Documents/nyu/fall_2024/os/project/custom-remote-shell/phase-1/dir1/dir2
my_shell> cd ..
my_shell> cd ..
my_shell> rm -r dir1
my_shell> ls
hello.txt       shell           shell.c
my_shell> 
```

Figure 24



```
my_shell> ls
hello.txt       shell           shell.c
my_shell> rm -f hello.txt
my_shell> ls
shell   shell.c
my_shell> 
```

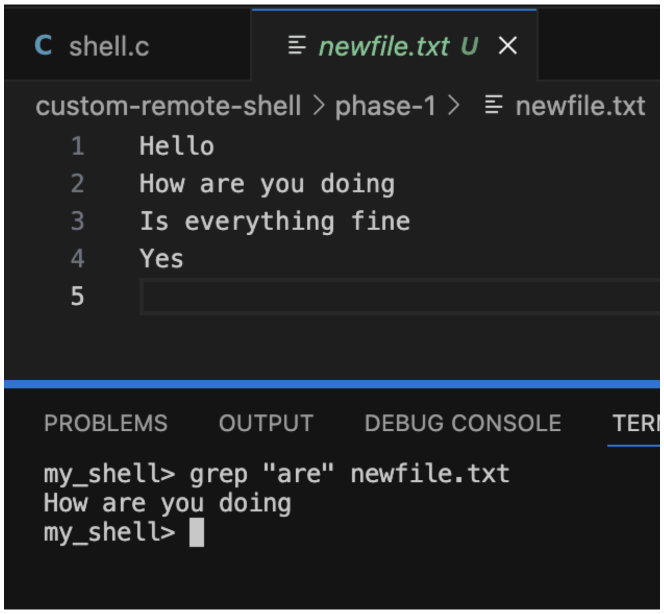
Figure 25



```
my_shell> rm -f combined.txt file1.txt file2.txt
my_shell> ls
shell   shell.c
my_shell> 
```

Figure 26

- grep

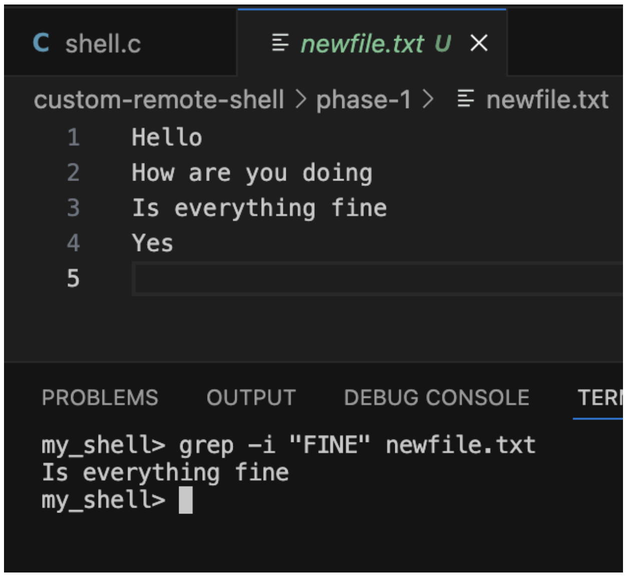


```
Hello
How are you doing
Is everything fine
Yes

```

```
my_shell> grep "are" newfile.txt
How are you doing
my_shell> 
```

Figure 27

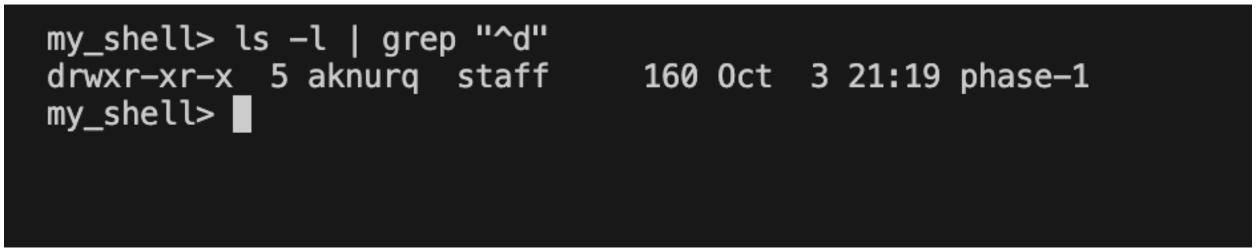


```
Hello
How are you doing
Is everything fine
Yes

```

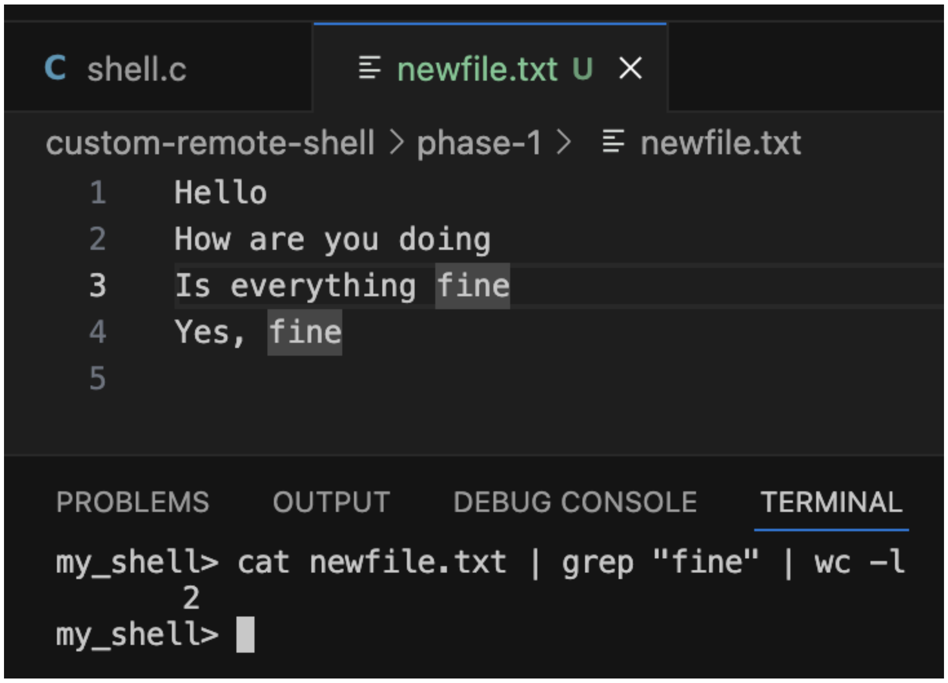
```
my_shell> grep -i "FINE" newfile.txt
Is everything fine
my_shell>
```

Figure 28



```
my_shell> ls -l | grep "^d"
drwxr-xr-x  5 aknurq  staff     160 Oct  3 21:19 phase-1
my_shell>
```

Figure 29



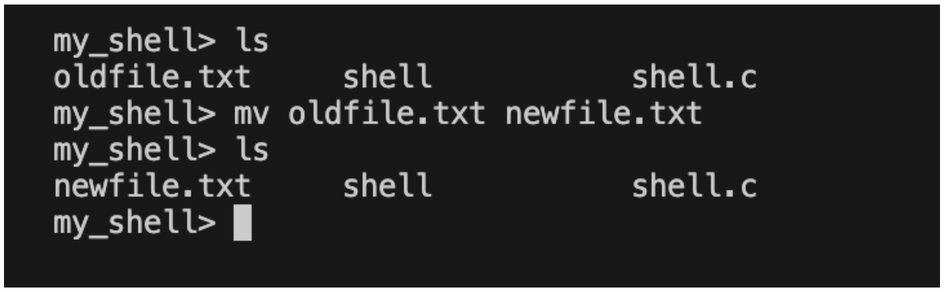
```
Hello
How are you doing
Is everything fine
Yes, fine

```

```
my_shell> cat newfile.txt | grep "fine" | wc -l
       2
my_shell> 
```

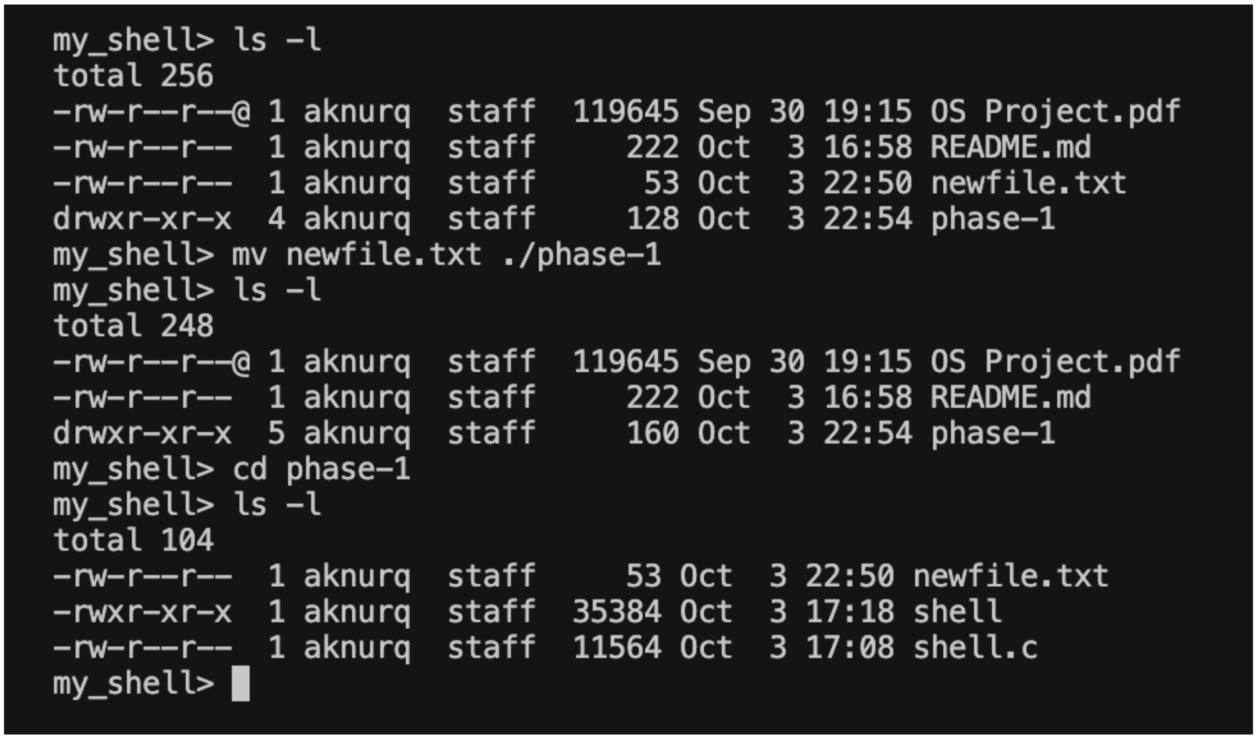
Figure 30

- mv



```
my_shell> ls
oldfile.txt     shell           shell.c
my_shell> mv oldfile.txt newfile.txt
my_shell> ls
newfile.txt     shell           shell.c
my_shell> 
```

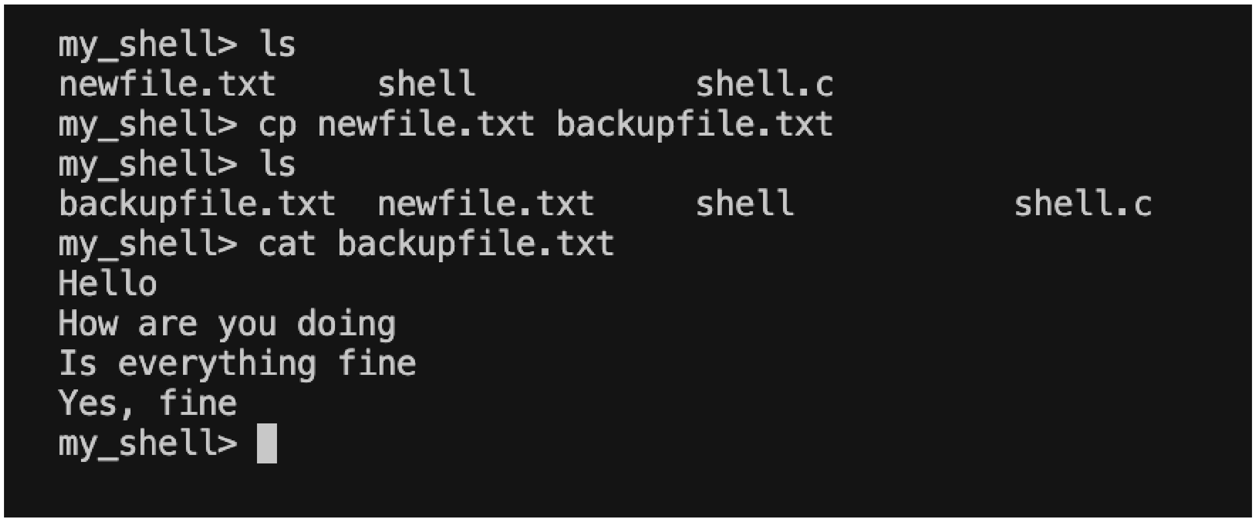
Figure 31



```
my_shell> ls -l
total 256
-rw-r--r--@ 1 aknurq  staff  119645 Sep 30 19:15 OS Project.pdf
-rw-r--r--  1 aknurq  staff     222 Oct  3 16:58 README.md
-rw-r--r--  1 aknurq  staff      53 Oct  3 22:50 newfile.txt
drwxr-xr-x  4 aknurq  staff     128 Oct  3 22:54 phase-1
my_shell> mv newfile.txt ./phase-1
my_shell> ls -l
total 248
-rw-r--r--@ 1 aknurq  staff  119645 Sep 30 19:15 OS Project.pdf
-rw-r--r--  1 aknurq  staff     222 Oct  3 16:58 README.md
drwxr-xr-x  5 aknurq  staff     160 Oct  3 22:54 phase-1
my_shell> cd phase-1
my_shell> ls -l
total 104
-rw-r--r--  1 aknurq  staff     53 Oct  3 22:50 newfile.txt
-rwxr-xr-x  1 aknurq  staff  35384 Oct  3 17:18 shell
-rw-r--r--  1 aknurq  staff  11564 Oct  3 17:08 shell.c
my_shell> 
```

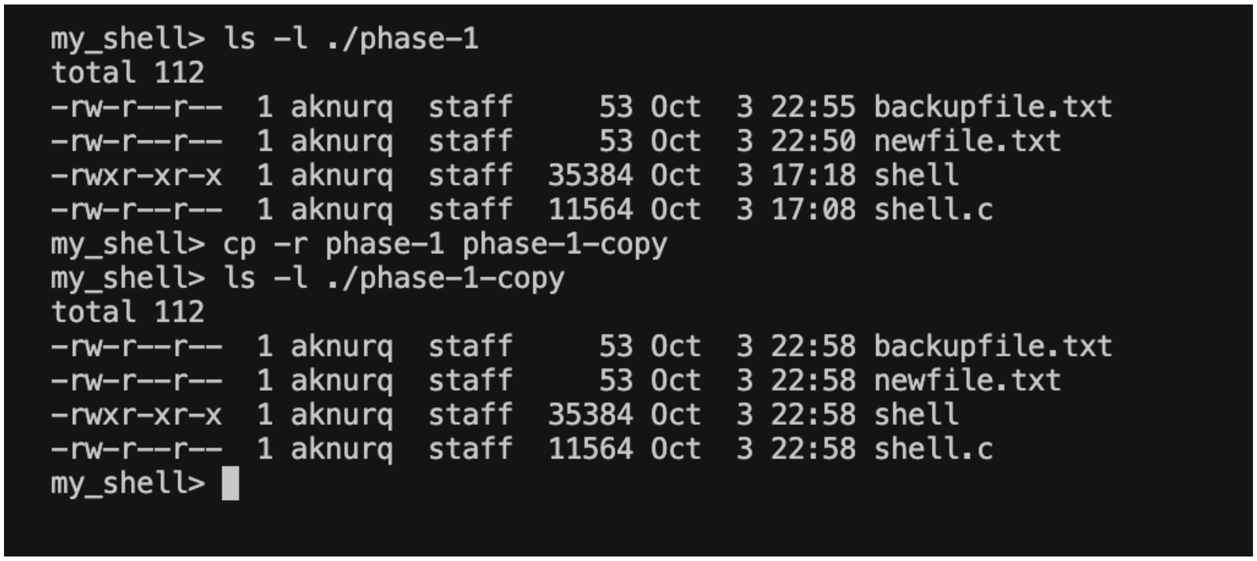
Figure 32

- cp



```
my_shell> ls
newfile.txt     shell           shell.c
my_shell> cp newfile.txt backupfile.txt
my_shell> ls
backupfile.txt  newfile.txt     shell           shell.c
my_shell> cat backupfile.txt
Hello
How are you doing
Is everything fine
Yes, fine
my_shell> 
```

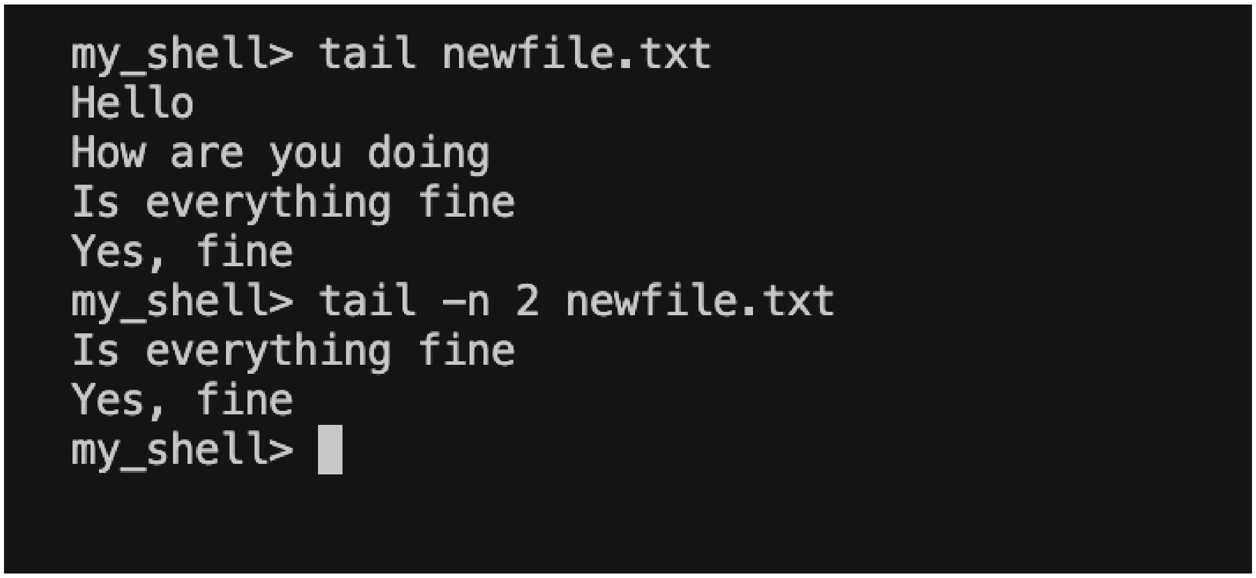
Figure 33



```
my_shell> ls -l ./phase-1
total 112
-rw-r--r--  1 aknurq  staff     53 Oct  3 22:55 backupfile.txt
-rw-r--r--  1 aknurq  staff     53 Oct  3 22:50 newfile.txt
-rwxr-xr-x  1 aknurq  staff  35384 Oct  3 17:18 shell
-rw-r--r--  1 aknurq  staff  11564 Oct  3 17:08 shell.c
my_shell> cp -r phase-1 phase-1-copy
my_shell> ls -l ./phase-1-copy
total 112
-rw-r--r--  1 aknurq  staff     53 Oct  3 22:58 backupfile.txt
-rw-r--r--  1 aknurq  staff     53 Oct  3 22:58 newfile.txt
-rwxr-xr-x  1 aknurq  staff  35384 Oct  3 22:58 shell
-rw-r--r--  1 aknurq  staff  11564 Oct  3 22:58 shell.c
my_shell> 
```

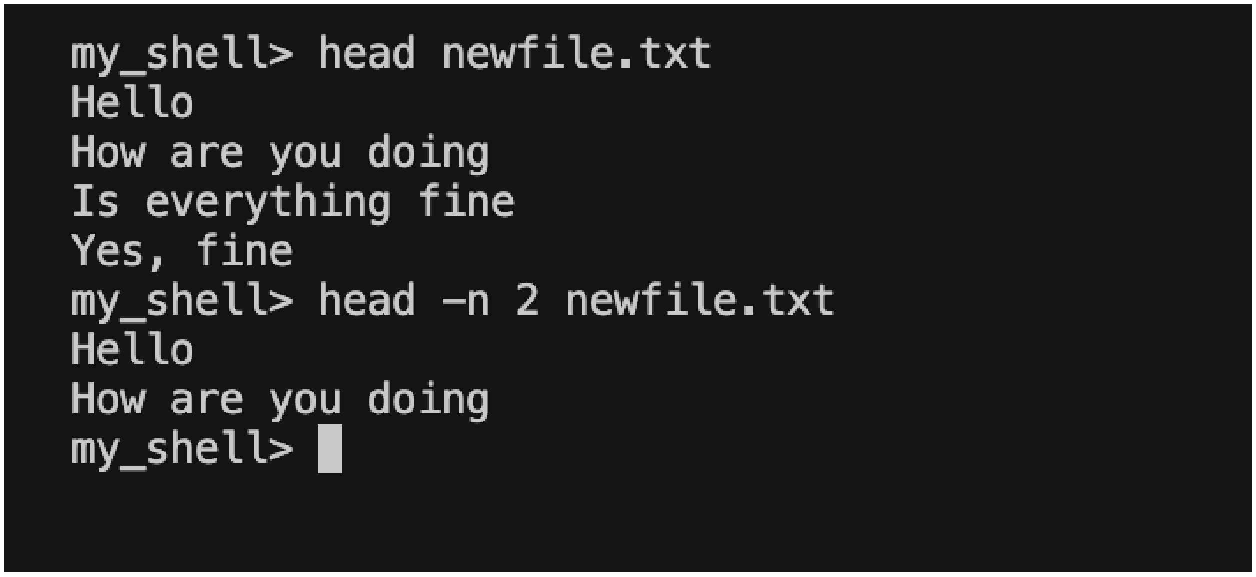
Figure 34

- tail



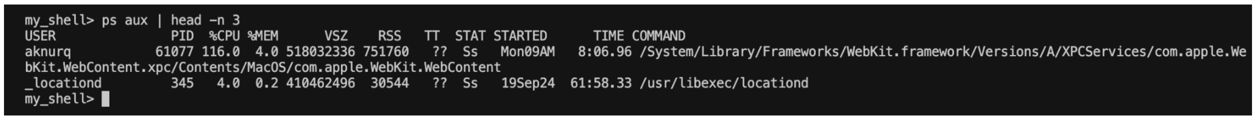
```
my_shell> tail newfile.txt
Hello
How are you doing
Is everything fine
Yes, fine
my_shell> tail -n 2 newfile.txt
Is everything fine
Yes, fine
my_shell> 
```

Figure 35



```
my_shell> head newfile.txt
Hello
How are you doing
Is everything fine
Yes, fine
my_shell> head -n 2 newfile.txt
Hello
How are you doing
my_shell> 
```

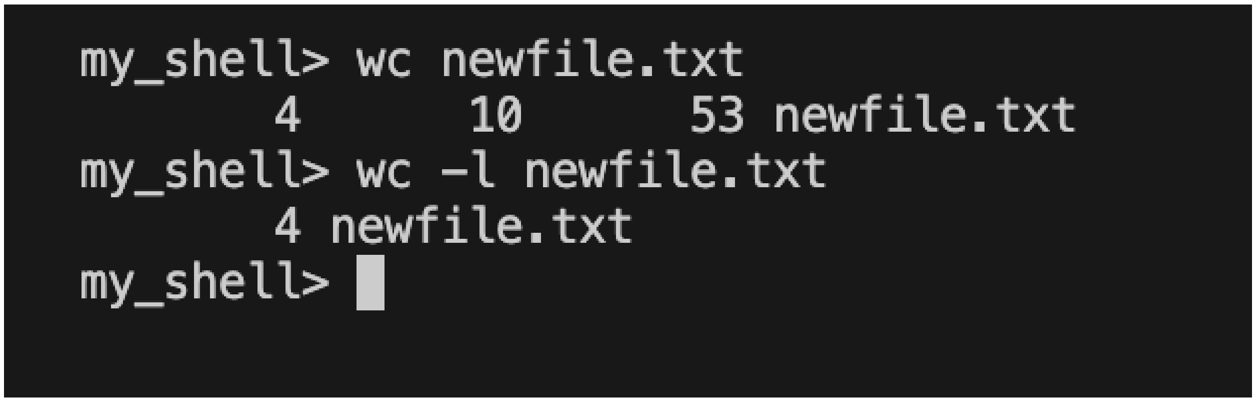
Figure 36



```
my_shell> ps aux | head -n 3
USER               PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
aknurq           61077 116.0  4.0 518032336 751760   ??  Ss   Mon09AM   8:06.96 /System/Library/Frameworks/WebKit.framework/Versions/A/XPCServices/com.apple.We
bKit.WebContent.xpc/Contents/MacOS/com.apple.WebKit.WebContent
_locationd         345   4.0  0.2 410462496  30544   ??  Ss   19Sep24  61:58.33 /usr/libexec/locationd
my_shell> 
```

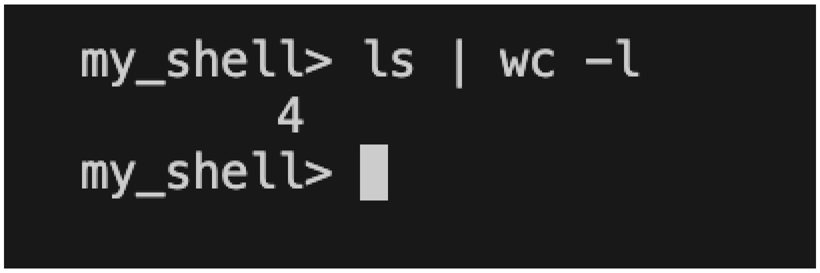
Figure 37

- wc



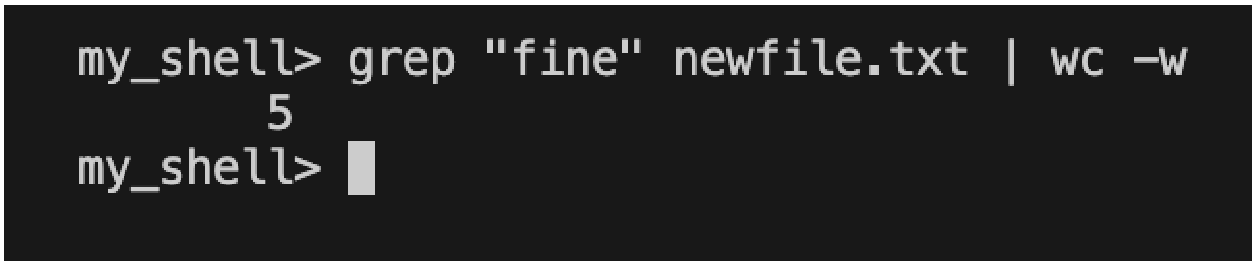
```
my_shell> wc newfile.txt
       4      10      53 newfile.txt
my_shell> wc -l newfile.txt
       4 newfile.txt
my_shell> 
```

Figure 38



```
my_shell> ls | wc -l
       4
my_shell> 
```

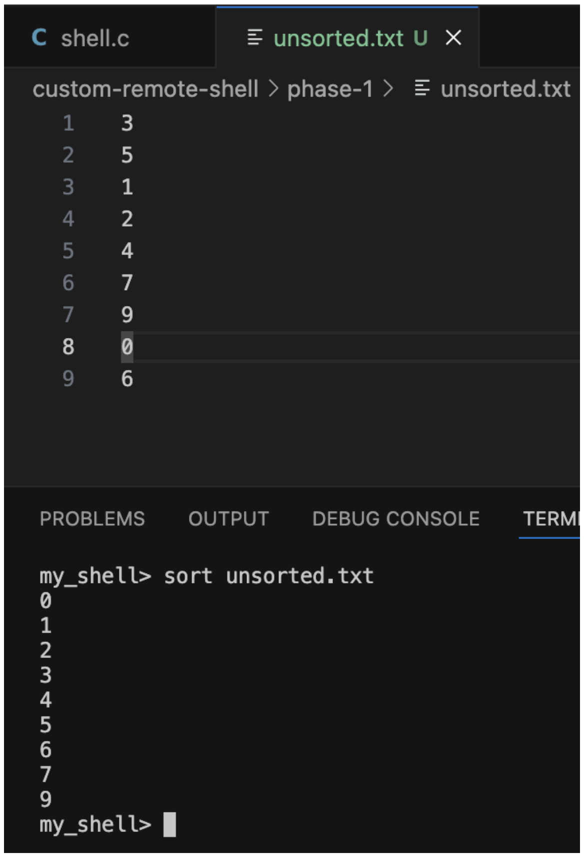
Figure 39



```
my_shell> grep "fine" newfile.txt | wc -w
       5
my_shell> 
```

Figure 40

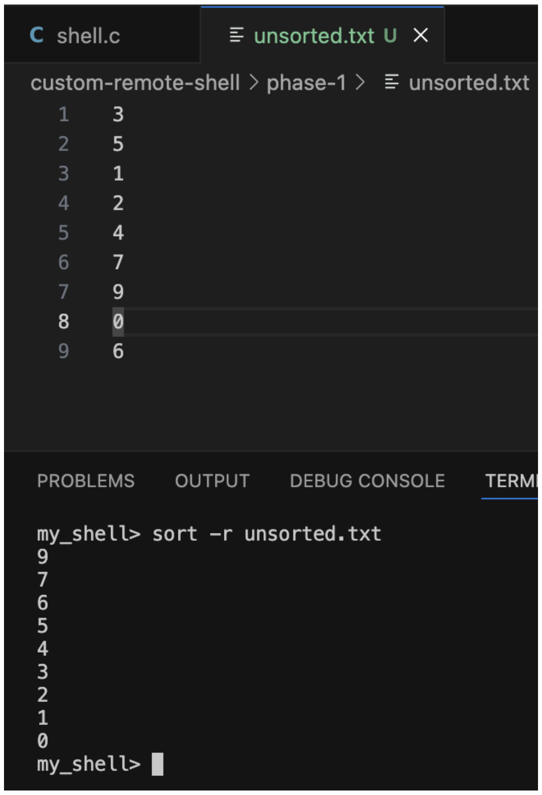
- sort



```
3
5
1
2
4
7
9
0
6
```

```
my_shell> sort unsorted.txt
0
1
2
3
4
5
6
7
9
my_shell> 
```

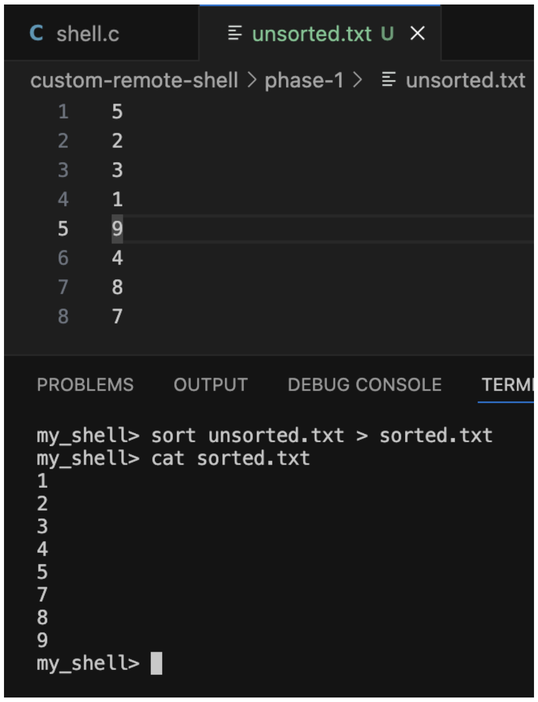
Figure 41



```
3
5
1
2
4
7
9
0
6
```

```
my_shell> sort -r unsorted.txt
9
7
6
5
4
3
2
1
0
my_shell> 
```

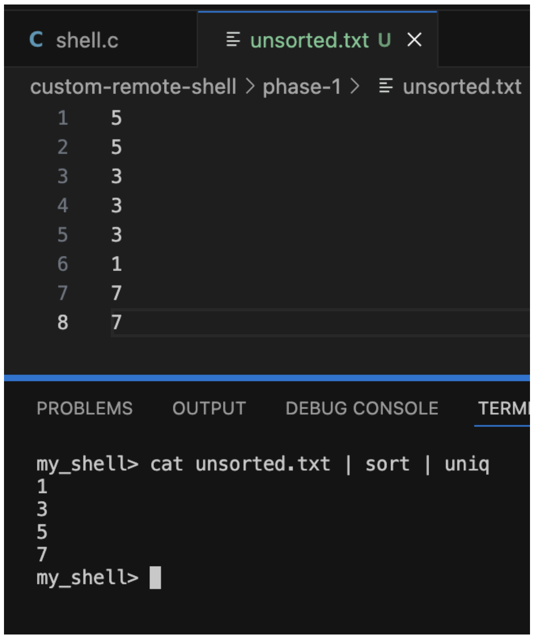
Figure 42



```
5
2
3
1
9
4
8
7
```

```
my_shell> sort unsorted.txt > sorted.txt
my_shell> cat sorted.txt
1
2
3
4
5
7
8
9
my_shell> 
```

Figure 43

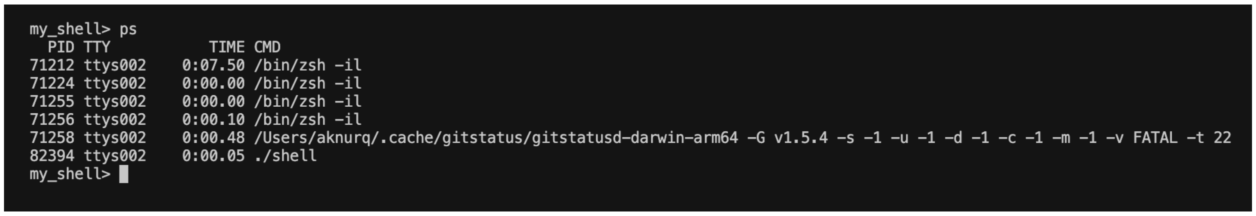


```
5
5
3
3
3
1
7
7
```

```
my_shell> cat unsorted.txt | sort | uniq
1
3
5
7
my_shell>
```

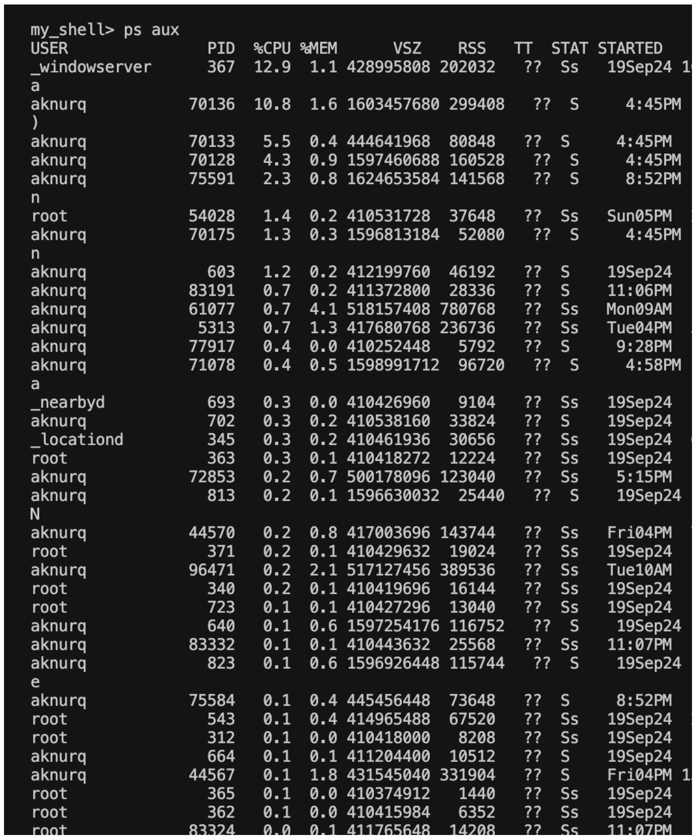
Figure 44

- ps



```
my_shell> ps
  PID TTY           TIME CMD
71212 ttys002    0:07.50 /bin/zsh -il
71224 ttys002    0:00.00 /bin/zsh -il
71255 ttys002    0:00.00 /bin/zsh -il
71256 ttys002    0:00.10 /bin/zsh -il
71258 ttys002    0:00.48 /Users/aknurq/.cache/gitstatus/gitstatusd-darwin-arm64 -G v1.5.4 -s -1 -u -1 -d -1 -c -1 -m -1 -v FATAL -t 22
82394 ttys002    0:00.05 ./shell
my_shell>
```

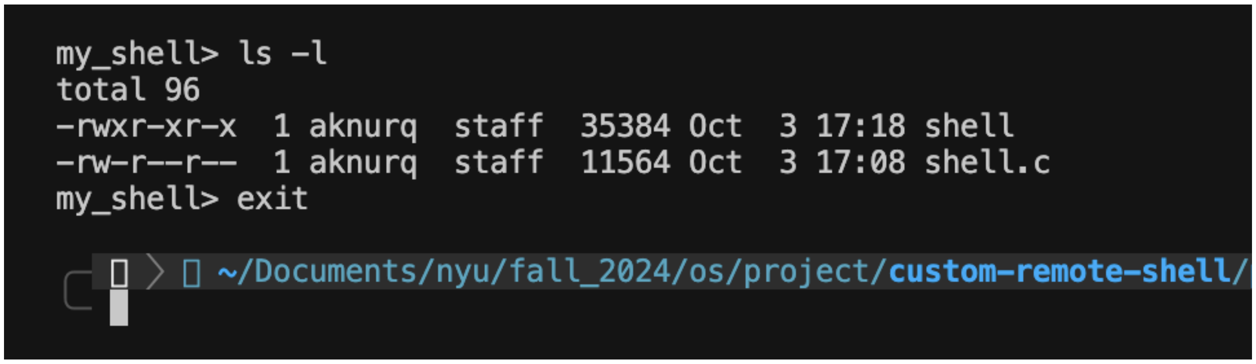
Figure 45



```
my_shell> ps aux
USER               PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED
_windowserver      367  12.9  1.1 428995808 202032   ??  Ss    19Sep24 1
a
aknurq           70136  10.8  1.6 1603457680 299408   ??  S      4:45PM
)
aknurq           70133   5.5  0.4 444641968  80848   ??  S      4:45PM
aknurq           70128   4.3  0.9 1597460688 160528   ??  S      4:45PM
aknurq           75591   2.3  0.8 1624653584 141568   ??  S      8:52PM
n
root             54028   1.4  0.2 410531728  37648   ??  Ss   Sun05PM
aknurq           70175   1.3  0.3 1596813184  52080   ??  S      4:45PM
n
aknurq             603   1.2  0.2 412199760  46192   ??  S     19Sep24
aknurq           83191   0.7  0.2 411372800  28336   ??  S     11:06PM
aknurq           61077   0.7  4.1 518157408 780768   ??  Ss   Mon09AM
aknurq            5313   0.7  1.3 417680768 236736   ??  Ss   Tue04PM
aknurq           77917   0.4  0.0 410252448   5792   ??  S      9:28PM
aknurq           71078   0.4  0.5 1598991712  96720   ??  S      4:58PM
a
_nearbyd           693   0.3  0.0 410426960   9104   ??  Ss    19Sep24
aknurq             702   0.3  0.2 410538160  33824   ??  S     19Sep24
_locationd         345   0.3  0.2 410461936  30656   ??  Ss    19Sep24
root               363   0.3  0.1 410418272  12224   ??  Ss    19Sep24
aknurq           72853   0.2  0.7 500178096 123040   ??  Ss     5:15PM
aknurq             813   0.2  0.1 1596630032  25440   ??  S     19Sep24
N
aknurq           44570   0.2  0.8 417003696 143744   ??  Ss   Fri04PM
root               371   0.2  0.1 410429632  19024   ??  Ss    19Sep24
aknurq           96471   0.2  2.1 517127456 389536   ??  Ss   Tue10AM
root               340   0.2  0.1 410419696  16144   ??  Ss    19Sep24
root               723   0.1  0.1 410427296  13040   ??  Ss    19Sep24
aknurq             640   0.1  0.6 1597254176 116752   ??  S     19Sep24
aknurq           83332   0.1  0.1 410443632  25568   ??  Ss    11:07PM
aknurq             823   0.1  0.6 1596926448 115744   ??  S     19Sep24
e
aknurq           75584   0.1  0.4 445456448  73648   ??  S      8:52PM
root               543   0.1  0.4 414965488  67520   ??  Ss    19Sep24
root               312   0.1  0.0 410418000   8208   ??  Ss    19Sep24
aknurq             664   0.1  0.1 411204400  10512   ??  S     19Sep24
aknurq           44567   0.1  1.8 431545040 331904   ??  S    Fri04PM 1
root               365   0.1  0.0 410374912   1440   ??  Ss    19Sep24
root               362   0.1  0.0 410415984   6352   ??  Ss    19Sep24
root             83324   0.0  0.1 411765648  14208   ??  Ss    11:07PM
```

Figure 46

- exit

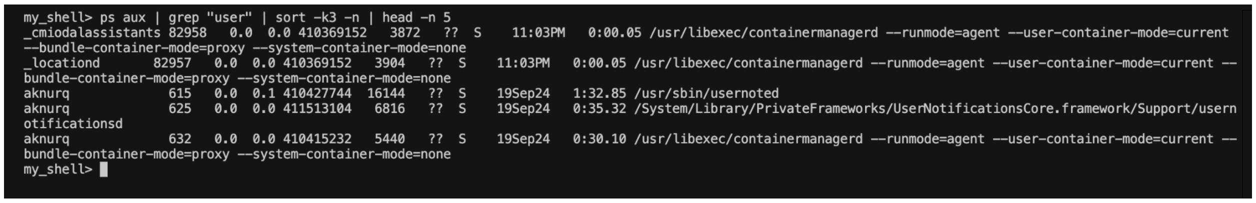


```
my_shell> ls -l
total 96
-rwxr-xr-x  1 aknurq  staff  35384 Oct  3 17:18 shell
-rw-r--r--  1 aknurq  staff  11564 Oct  3 17:08 shell.c
my_shell> exit

  ~/Documents/nyu/fall_2024/os/project/custom-remote-shell/
```

Figure 47

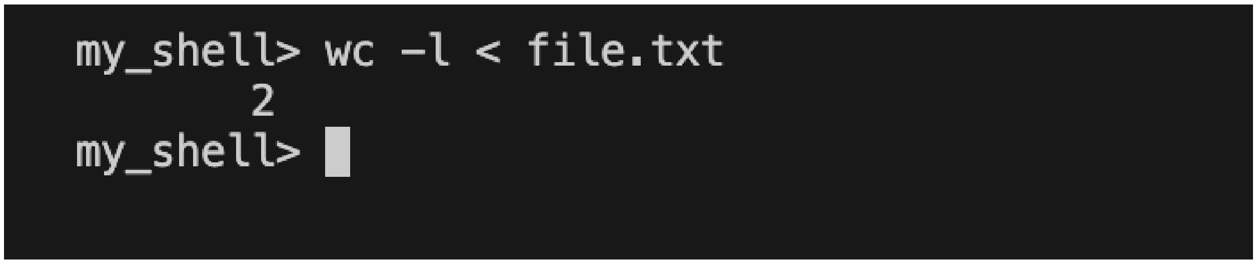
- 3 pipes



```
my_shell> ps aux | grep "user" | sort -k3 -n | head -n 5
_cmiodalassistants 82958   0.0  0.0 410369152   3872   ??  S     11:03PM   0:00.05 /usr/libexec/containermanagerd --runmode=agent --user-container-mode=current
--bundle-container-mode=proxy --system-container-mode=none
_locationd       82957   0.0  0.0 410369152   3904   ??  S    11:03PM   0:00.05 /usr/libexec/containermanagerd --runmode=agent --user-container-mode=current --
bundle-container-mode=proxy --system-container-mode=none
aknurq             615   0.0  0.1 410427744  16144   ??  S    19Sep24   1:32.85 /usr/sbin/usernoted
aknurq             625   0.0  0.0 411513104   6816   ??  S    19Sep24   0:35.32 /System/Library/PrivateFrameworks/UserNotificationsCore.framework/Support/usern
otificationsd
aknurq             632   0.0  0.0 410415232   5440   ??  S    19Sep24   0:30.10 /usr/libexec/containermanagerd --runmode=agent --user-container-mode=current --
bundle-container-mode=proxy --system-container-mode=none
my_shell> 
```

Figure 48

- Input Redirection



```
my_shell> wc -l < file.txt
       2
my_shell> 
```

Figure 49