Juan Estrada ID: 923058731
Github: jjestrada2 CSC415 Operating Systems

Assignment 3 – Command Line Arguments

Description:
This assignment is about building a simple shell program that enables users to enter and execute various shell commands in a command-line interface. This basic shell supports features such as piping multiple commands together (e.g., "command1 | command2"), wildcard expansion (e.g., "ls *.c" to list all '.c' files), and executing single commands. The code employs functions to tokenize user input, manage child processes for command execution, and handle wildcard pattern matching.

Approach / What I Did:

1. First, I carefully reviewed the list of functions that our professor had emphasized in class as essential for solving the assignment. These included functions like strtok, fork, execvp, pipe, and dup2. Understanding the capabilities of these functions was crucial as they formed the building blocks of our shell.

2. My initial approach involved creating the first version of the shell, which supported single commands. I used a straightforward approach, using fgets to read the command line input, strtok to tokenize the input, and fork along with execvp to execute the instructions. This allowed me to establish a foundation for command execution within the shell.

3. To handle piped commands, I began devising a plan to implement pipes effectively. I also explored various online resources and examples to gather ideas and best practices for implementing pipes in a shell. Understanding how pipes functioned and how they could be managed between child processes was critical.

4. I took a hands-on approach to solidify my understanding of pipes. Using a whiteboard, I sketched out a step-by-step process detailing how each of my pipes would be created and how child processes would be executed while ensuring there were no resource leaks. This visual representation helped clarify the complex interactions between processes and pipes.

5. Building upon my plan and insights, I proceeded to code the solution. While implementing pipes, I realized that the code was becoming challenging to read and maintain. To enhance code organization and clarity, I decided to modularize my solution.

6. I created a separate my_pipe function to handle pipes and a splitline function to tokenize input lines. This modular approach significantly improved the readability and maintainability of my code.

7. As I believed I had completed the core requirements of the assignment with the successful implementation of pipes, I encountered an unexpected issue when testing the shell with 'make run < commands.txt.' The issue surfaced when attempting to execute 'cat .c,' which involved wildcard expansion. To address this, I extended my shell's functionality by implementing wildcard support, specifically '', using the expand_wildcards function. This addition allowed the shell to recognize and execute commands with wildcard patterns, enhancing its versatility and usability for various scenarios.

Issues and Resolutions:

1. Initially, when attempting to run the program with the command provided by the professor, 'make run < commands.txt,' I encountered a hurdle. It became evident that I had overlooked filling out the 'makefile' with my name and last name, which resulted in an error. Once I addressed this oversight by updating the 'makefile' with my information, the program ran smoothly without any issues.

2. Another challenge emerged while processing input commands. I noticed that the input lines contained a trailing '\n' character, causing difficulties in parsing and executing the commands. To resolve this, I made a simple yet effective adjustment. Instead of using a buffer size of 187 characters, I increased it to 188, allowing space for the null termination character '\0.' This modification ensured proper string termination and eliminated issues associated with the newline character.

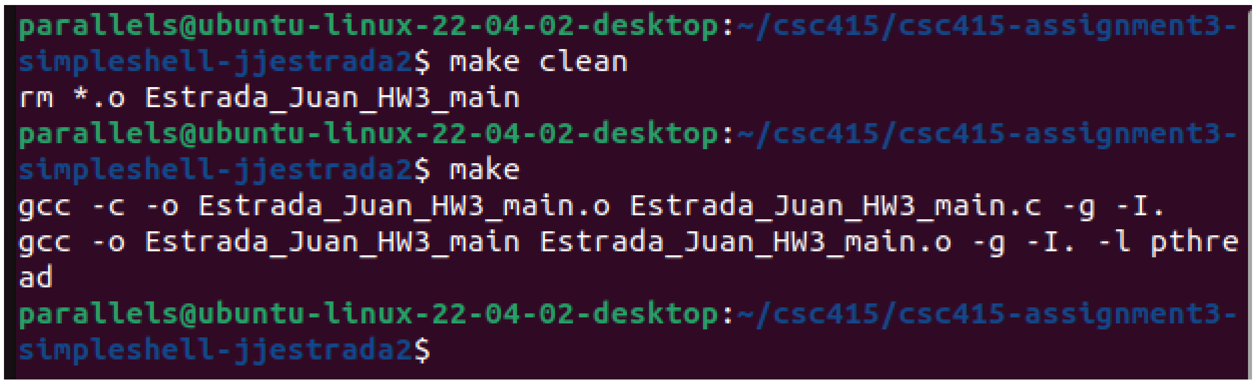
3. During the development process, I encountered confusion and difficulties with the strtok function, particularly when utilizing the statement 'token = strtok(NULL, " ");.' This confusion stemmed from a mix-up between strtok and strtok_r. I was initially using the wrong parameter implementation. To resolve this issue, I revisited the documentation and clarified the correct usage of strtok and 'strtok_r,' ensuring the accurate tokenization of input strings.

4. Implementing pipes introduced several complexities, and though I encountered multiple issues, I don't recall all the specific details. However, one significant problem I faced was related to the structure of the 'do-while' loop. Initially, I placed the 'waitpid' function within the parent process immediately after 'commandc++,' causing the program to wait indefinitely. This happened because there were no child processes to wait for, leading to a deadlock. To address this issue, I removed the 'waitpid' call from that location, resolving the deadlock and ensuring the proper execution of child processes.

5. Another challenge arose when implementing wildcard support. I initially overlooked the distinction between the '*' character used in 'command *.c' and other cases. This oversight led to an error message indicating 'no such file or directory.' To rectify this, I

devised a solution by introducing a new instruction format that could handle wildcard patterns. This adjustment allowed the shell to differentiate between various types of wildcard usage, such as 'command *.c,' ensuring the correct execution of commands involving wildcards.

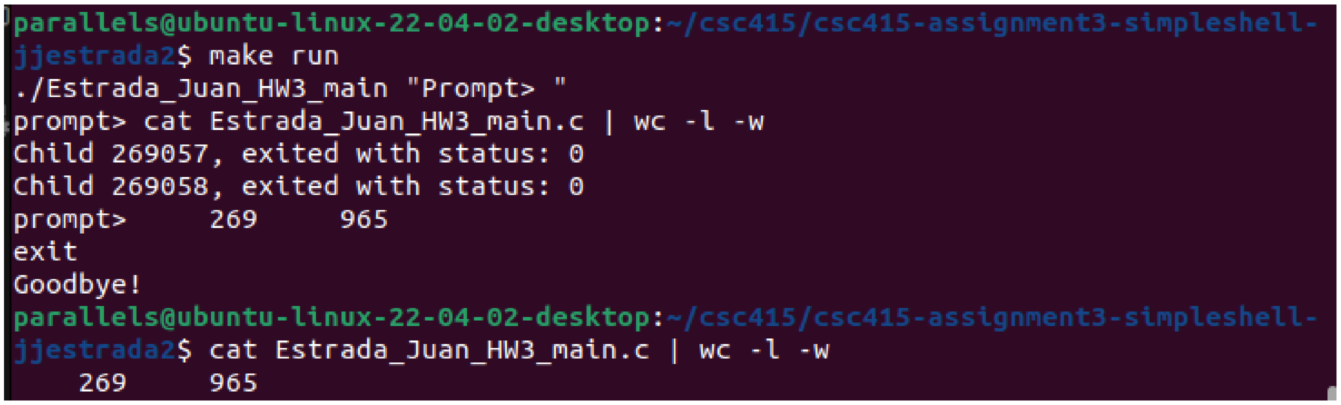
Screen shot of compilation:



```
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-
simpleshell-jjestrada2$ make clean
rm *.o Estrada_Juan_HW3_main
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-
simpleshell-jjestrada2$ make
gcc -c -o Estrada_Juan_HW3_main.o Estrada_Juan_HW3_main.c -g -I.
gcc -o Estrada_Juan_HW3_main Estrada_Juan_HW3_main.o -g -I. -l pthre
ad
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-
simpleshell-jjestrada2$
```

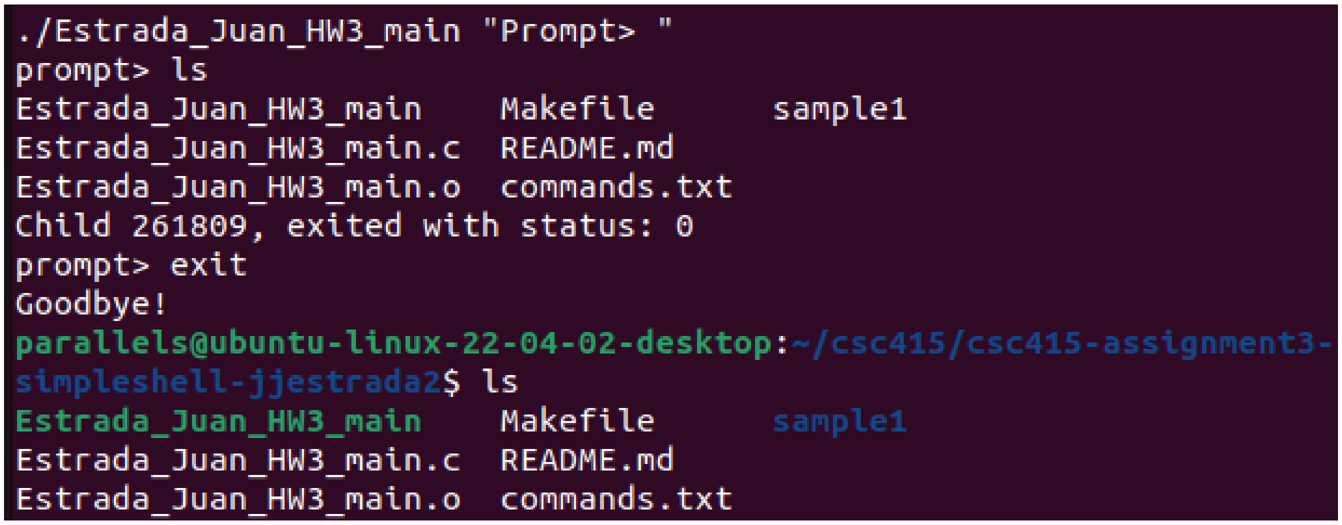
Screen shot(s) of the execution of the program:

ls



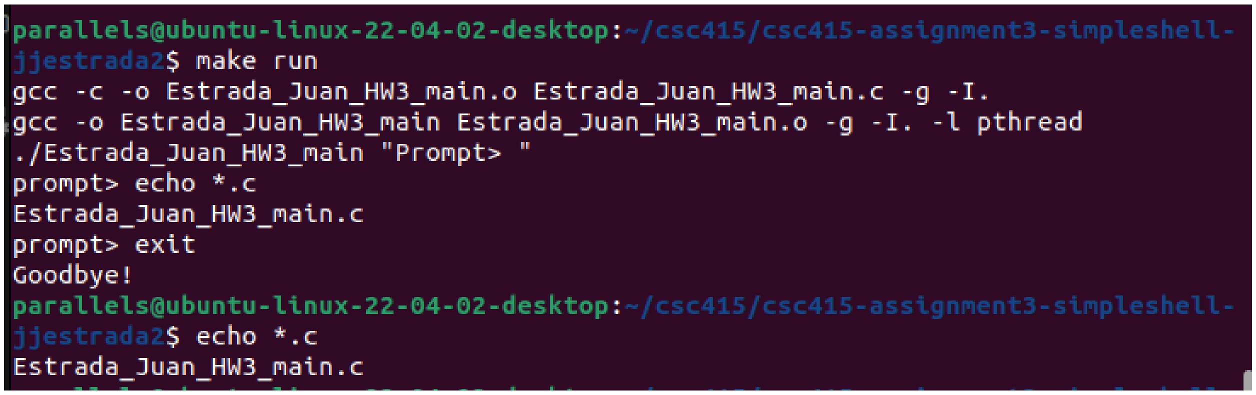
```
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-simpleshell-
jjestrada2$ make run
./Estrada_Juan_HW3_main "Prompt> "
prompt> cat Estrada_Juan_HW3_main.c | wc -l -w
Child 269057, exited with status: 0
Child 269058, exited with status: 0
prompt>     269     965
exit
Goodbye!
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-simpleshell-
jjestrada2$ cat Estrada_Juan_HW3_main.c | wc -l -w
    269     965
```

cat Estrada_Juan_HW3_main.c | wc -l -w



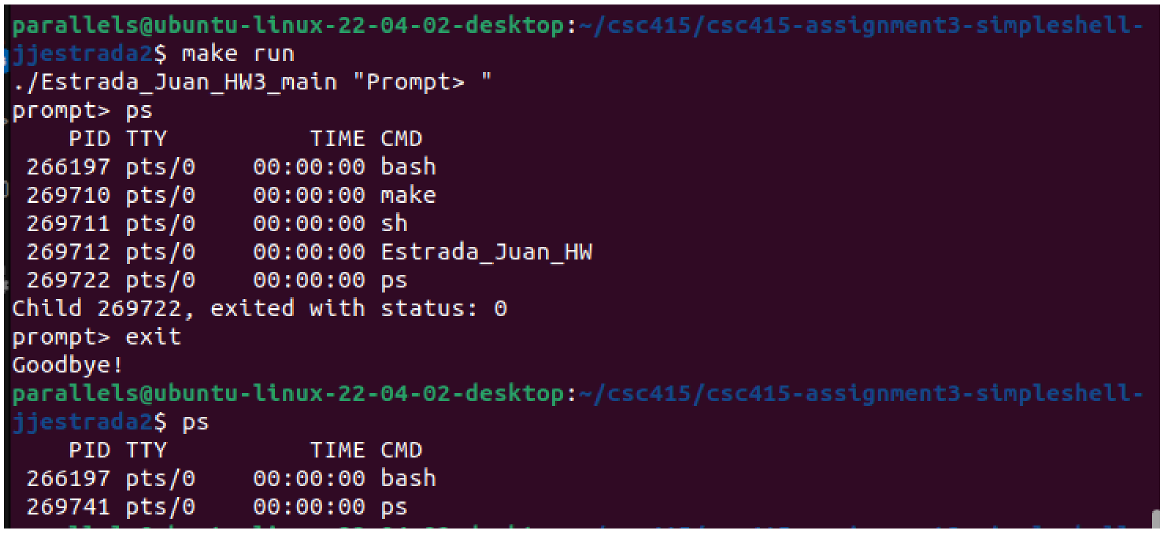
```
./Estrada_Juan_HW3_main "Prompt> "
prompt> ls
Estrada_Juan_HW3_main    Makefile     sample1
Estrada_Juan_HW3_main.c  README.md
Estrada_Juan_HW3_main.o  commands.txt
Child 261809, exited with status: 0
prompt> exit
Goodbye!
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-
simpleshell-jjestrada2$ ls
Estrada_Juan_HW3_main    Makefile     sample1
Estrada_Juan_HW3_main.c  README.md
Estrada_Juan_HW3_main.o  commands.txt
```

echo *.c

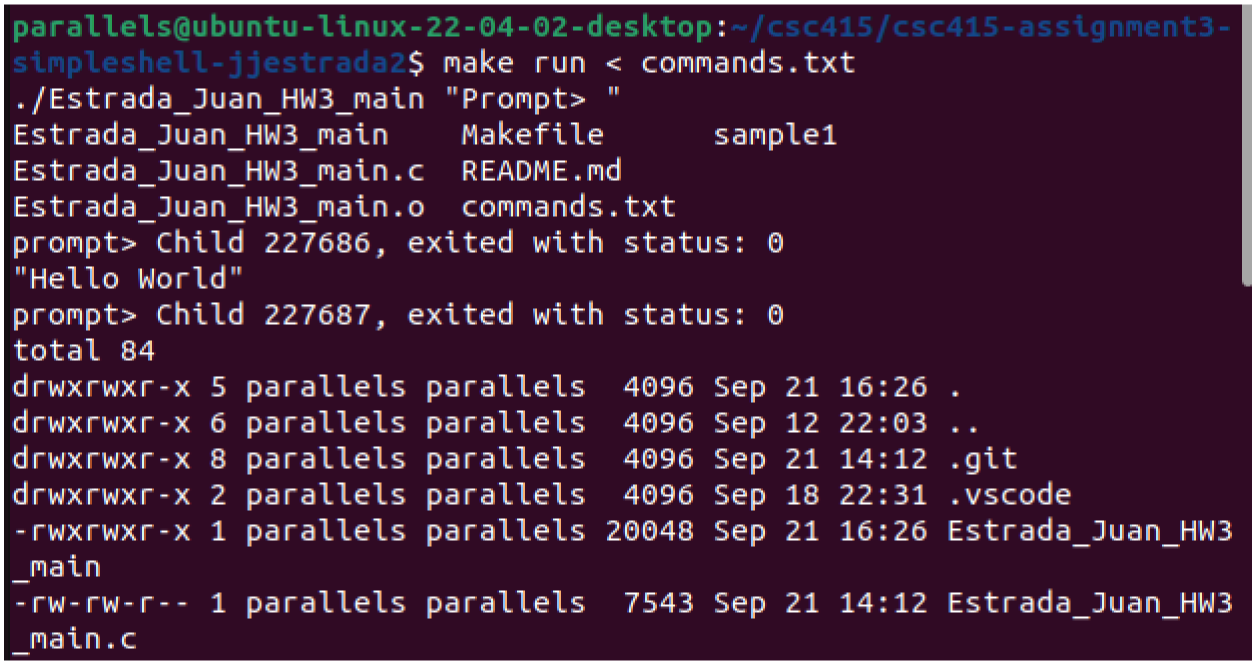


```
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-simpleshell-
jjestrada2$ make run
gcc -c -o Estrada_Juan_HW3_main.o Estrada_Juan_HW3_main.c -g -I.
gcc -o Estrada_Juan_HW3_main Estrada_Juan_HW3_main.o -g -I. -l pthread
./Estrada_Juan_HW3_main "Prompt> "
prompt> echo *.c
Estrada_Juan_HW3_main.c
prompt> exit
Goodbye!
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-simpleshell-
jjestrada2$ echo *.c
Estrada_Juan_HW3_main.c
```

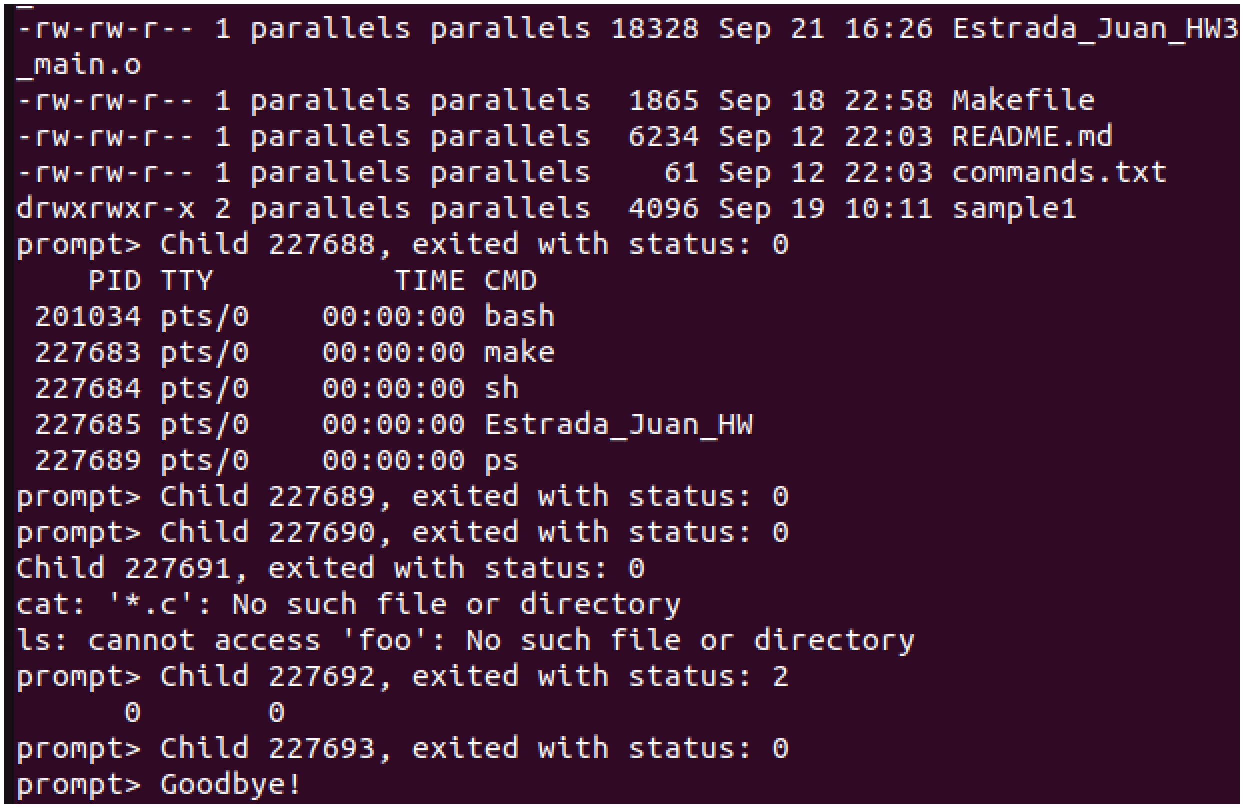
ps



```
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-simpleshell-
jjestrada2$ make run
./Estrada_Juan_HW3_main "Prompt> "
prompt> ps
    PID TTY          TIME CMD
 266197 pts/0    00:00:00 bash
 269710 pts/0    00:00:00 make
 269711 pts/0    00:00:00 sh
 269712 pts/0    00:00:00 Estrada_Juan_HW
 269722 pts/0    00:00:00 ps
Child 269722, exited with status: 0
prompt> exit
Goodbye!
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-simpleshell-
jjestrada2$ ps
    PID TTY          TIME CMD
 266197 pts/0    00:00:00 bash
 269741 pts/0    00:00:00 ps
```



```
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment3-simpleshell-jjestrada2$ make run < commands.txt
./Estrada_Juan_HW3_main "Prompt> "
Estrada_Juan_HW3_main    Makefile      sample1
Estrada_Juan_HW3_main.c  README.md
Estrada_Juan_HW3_main.o  commands.txt
prompt> Child 227686, exited with status: 0
"Hello World"
prompt> Child 227687, exited with status: 0
total 84
drwxrwxr-x 5 parallels parallels  4096 Sep 21 16:26 .
drwxrwxr-x 6 parallels parallels  4096 Sep 12 22:03 ..
drwxrwxr-x 8 parallels parallels  4096 Sep 21 14:12 .git
drwxrwxr-x 2 parallels parallels  4096 Sep 18 22:31 .vscode
-rwxrwxr-x 1 parallels parallels 20048 Sep 21 16:26 Estrada_Juan_HW3
_main
-rw-rw-r-- 1 parallels parallels  7543 Sep 21 14:12 Estrada_Juan_HW3
_main.c
```



```
-rw-rw-r-- 1 parallels parallels 18328 Sep 21 16:26 Estrada_Juan_HW3
_main.o
-rw-rw-r-- 1 parallels parallels  1865 Sep 18 22:58 Makefile
-rw-rw-r-- 1 parallels parallels  6234 Sep 12 22:03 README.md
-rw-rw-r-- 1 parallels parallels    61 Sep 12 22:03 commands.txt
drwxrwxr-x 2 parallels parallels  4096 Sep 19 10:11 sample1
prompt> Child 227688, exited with status: 0
    PID TTY          TIME CMD
 201034 pts/0    00:00:00 bash
 227683 pts/0    00:00:00 make
 227684 pts/0    00:00:00 sh
 227685 pts/0    00:00:00 Estrada_Juan_HW
 227689 pts/0    00:00:00 ps
prompt> Child 227689, exited with status: 0
prompt> Child 227690, exited with status: 0
Child 227691, exited with status: 0
cat: '*.c': No such file or directory
ls: cannot access 'foo': No such file or directory
prompt> Child 227692, exited with status: 2
      0       0
prompt> Child 227693, exited with status: 0
prompt> Goodbye!
```