Yuvraj Gupta
Id : 922933190
Github : YuvrajGupta1808
CSC415 Operating Systems

Assignment 3 – Simple Shell with Pipes

Description:

This assignment involves creating a custom shell that reads user input, parses commands, and executes them using fork() and execvp(). The shell will also support piping, allowing the output of one command to be used as the input for another.

Approach:

1. Initialize the Shell Loop: The main function initializes the shell with a default or user-provided prompt and calls the shell loop to start reading and executing commands.
2. The shell loop (`shell_loop` function) continuously reads user input and handles EOF or read errors. It removes any trailing newline character from the input, skips empty inputs by printing an "Empty input" message, and continues the loop. It parses the input into separate commands if pipes (`|`) are present, determines if the input is a single command or multiple piped commands, and executes single commands directly or delegates execution of piped commands.
3. Input parsing (`parse_input` function) uses `strtok_r` to split the input string into tokens based on a given delimiter (either `|` for pipes or whitespace for command arguments) and returns tokens one by one to facilitate command and argument parsing.
4. Single command execution (`execute_command` function) handles the special "exit" command to terminate the shell, forks a new child process to execute the command using `execvp`, attempts to execute the command in the child process and exits if it fails, waits for the child process to complete in the parent process, and prints the child's exit status.
5. Piped commands execution (`execute_piped_commands` function) sets up the necessary number of pipes for the commands, forks a child process for each command in the pipeline, sets up input and output redirection using `dup2` based on the pipe configuration in each child

process, closes all pipe file descriptors to prevent resource leaks, parses and executes each command using `execvp`, closes all pipe file descriptors in the parent process, and waits for all child processes to complete, printing their exit statuses.

Issues:

Issue 1: The implementation uses various functions and types that may be complex to understand. These include `fgets`, `feof`, `strtok_r`, `strcmp`, `pid_t` type, `waitpid`, `dup2`, and `execvp`. Understanding the purpose and correct usage of these functions is crucial for proper implementation and debugging.

Solution: To facilitate better understanding of these functions and types, each function and type should be clearly explained in terms of its purpose, usage, and any nuances. `fgets` reads a line of input from a stream, while `feof` checks if the end of a file has been reached. `strtok_r` tokenizes a string with thread safety, and `strcmp` compares two strings. The `pid_t` type is used for process IDs, `waitpid` waits for a process to change state, `dup2` duplicates a file descriptor, and `execvp` executes a program, replacing the current process.

Issue 2: The shell implementation encounters a segmentation fault when executing piped commands due to improper handling of child processes and file descriptors. The issue arises because the child processes in the `execute_piped_commands`function fail to correctly manage input and output redirection using pipes, do not print their process IDs and exit statuses, and improperly parse and pass arguments to the `execvp` function. Additionally, the shell does not handle empty outputs gracefully, which can lead to crashes or incorrect behavior. To resolve this, it is essential to ensure proper argument parsing and error handling in the child processes, print necessary process information, manage file descriptors correctly to prevent resource leaks, and handle cases where commands produce empty outputs.

```
student@student:~/Desktop/csc415-assignment3-simpleshell-YuvrajGupta1808$ make run < commands.txt
./Gupta_Yuvraj_HW3_main "Prompt> "
commands.txt          Gupta_Yuvraj_HW3_main.c  Makefile
Gupta_Yuvraj_HW3_main  Gupta_Yuvraj_HW3_main.o  README.md
Prompt> Child 88622, exited with 0
"Hello World"
Prompt> Child 88623, exited with 0
total 72
drwxrwxr-x 3 student student  4096 Jun 19 18:56 .
drwxr-xr-x 6 student student  4096 Jun 14 10:50 ..
-rw-rw-r-- 1 student student    66 Jun 14 10:50 commands.txt
drwxrwxr-x 8 student student  4096 Jun 19 18:53 .git
-rwxrwxr-x 1 student student 18920 Jun 19 18:56 Gupta_Yuvraj_HW3_main
-rw-rw-r-- 1 student student  5021 Jun 19 18:56 Gupta_Yuvraj_HW3_main.c
-rw-rw-r-- 1 student student 15816 Jun 19 18:56 Gupta_Yuvraj_HW3_main.o
-rw-rw-r-- 1 student student  1865 Jun 14 10:50 Makefile
-rw-rw-r-- 1 student student  7317 Jun 14 10:50 README.md
Prompt> Child 88624, exited with 0
    PID TTY          TIME CMD
  88583 pts/2    00:00:00 bash
  88619 pts/2    00:00:00 make
  88620 pts/2    00:00:00 sh
  88621 pts/2    00:00:00 Gupta_Yuvraj_HW
  88625 pts/2    00:00:00 ps
Prompt> Child 88625, exited with 0
Prompt> args[0]: wc
args[1]: -l
args[2]: -w
     66     309
ls: cannot access 'foo': No such file or directory
Prompt> Prompt> Child 88628, exited with 2
Segmentation fault (core dumped)
make: *** [Makefile:59: run] Error 139
student@student:~/Desktop/csc415-assignment3-simpleshell-YuvrajGupta1808$
```

Solution : To resolve the segmentation fault in the shell implementation for piped commands, ensure proper argument parsing and error handling, print process IDs and exit statuses, and manage file descriptors correctly. Use dup2 for input/output redirection, close unused file descriptors, and handle empty outputs gracefully to avoid crashes. This will ensure the shell processes and executes piped commands correctly.

Screen shot of compilation:

```
student@student:~/Desktop/csc415-assignment3-simpleshell-YuvrajGupta1808$ make
gcc -c -o Gupta_Yuvraj_HW3_main.o Gupta_Yuvraj_HW3_main.c -g -I.
gcc -o Gupta_Yuvraj_HW3_main Gupta_Yuvraj_HW3_main.o -g -I. -l pthread
student@student:~/Desktop/csc415-assignment3-simpleshell-YuvrajGupta1808$
```

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

Test 1: Base Case

```
student@student:~/Desktop/csc415-assignment3-simpleshell-YuvrajGupta1808$ make run < commands.txt
./Gupta_Yuvraj_HW3_main "Prompt> "
commands.txt  Gupta_Yuvraj_HW3_main  Gupta_Yuvraj_HW3_main.c  Gupta_Yuvraj_HW3_main.o  Makefile  README.md
Prompt> Child 88844 exited with status 0
"Hello World"
Prompt> Child 88845 exited with status 0
total 72
drwxrwxr-x 3 student student  4096 Jun 19 19:20 .
drwxr-xr-x 6 student student  4096 Jun 14 10:50 ..
-rw-rw-r-- 1 student student    66 Jun 14 10:50 commands.txt
drwxrwxr-x 8 student student  4096 Jun 19 18:53 .git
-rwxrwxr-x 1 student student 19008 Jun 19 19:20 Gupta_Yuvraj_HW3_main
-rw-rw-r-- 1 student student  6469 Jun 19 19:20 Gupta_Yuvraj_HW3_main.c
-rw-rw-r-- 1 student student 16016 Jun 19 19:20 Gupta_Yuvraj_HW3_main.o
-rw-rw-r-- 1 student student  1865 Jun 14 10:50 Makefile
-rw-rw-r-- 1 student student  7317 Jun 14 10:50 README.md
Prompt> Child 88846 exited with status 0
    PID TTY          TIME CMD
  88583 pts/2    00:00:00 bash
  88841 pts/2    00:00:00 make
  88842 pts/2    00:00:00 sh
  88843 pts/2    00:00:00 Gupta_Yuvraj_HW
  88847 pts/2    00:00:00 ps
Prompt> Child 88847 exited with status 0
Prompt> Child 88848 exited with status 0
     64     304
Child 88849 exited with status 0
ls: cannot access 'foo': No such file or directory
Prompt> Child 88850 exited with status 2
Prompt> Empty input
Prompt>
student@student:~/Desktop/csc415-assignment3-simpleshell-YuvrajGupta1808$
```

Test 2: Other Check Cases

```
student@student:~/Desktop/csc415-assignment3-simpleshell-YuvrajGupta1808$ make run
./Gupta_Yuvraj_HW3_main "Prompt> "
Prompt> date
Wed Jun 19 08:05:12 PM PDT 2024
Child 89128 exited with status 0
Prompt> echo "Hello"
"Hello"
Child 89129 exited with status 0
Prompt> cat commands.txt
ls
echo "Hello World"
ls -l -a
ps
cat Makefile | wc -l -w
ls foo

Child 89130 exited with status 0
Prompt> uname -a
Linux student 6.5.0-41-generic #41~22.04.2-Ubuntu SMP PREEMPT_DYNAMIC Mon Jun  3 16:28:24 UTC 2 aarch64 aarch64 aarch64 GNU/Linux
Child 89133 exited with status 0
Prompt> cat commands.txt | grep "echo"
Child 89135 exited with status 0
Child 89136 exited with status 1
Prompt> /bin/ls -l /var/tmp
total 36
drwx------ 3 root root 4096 Jun  5 12:56 systemd-private-d258252bbc164ec7816ad1bb04342ea9-colord.service-pgOivt
drwx------ 3 root root 4096 Jun  5 12:56 systemd-private-d258252bbc164ec7816ad1bb04342ea9-ModemManager.service-MOP7oG
drwx------ 3 root root 4096 Jun  5 12:56 systemd-private-d258252bbc164ec7816ad1bb04342ea9-power-profiles-daemon.service-r1pVLr
drwx------ 3 root root 4096 Jun  5 12:56 systemd-private-d258252bbc164ec7816ad1bb04342ea9-switcheroo-control.service-no1WS6
drwx------ 3 root root 4096 Jun  5 12:56 systemd-private-d258252bbc164ec7816ad1bb04342ea9-systemd-logind.service-b83kqS
drwx------ 3 root root 4096 Jun  5 05:57 systemd-private-d258252bbc164ec7816ad1bb04342ea9-systemd-oomd.service-IKMlHd
drwx------ 3 root root 4096 Jun  5 05:57 systemd-private-d258252bbc164ec7816ad1bb04342ea9-systemd-resolved.service-1odTum
drwx------ 3 root root 4096 Jun  5 05:57 systemd-private-d258252bbc164ec7816ad1bb04342ea9-systemd-timesyncd.service-S7MRzf
drwx------ 3 root root 4096 Jun  5 12:56 systemd-private-d258252bbc164ec7816ad1bb04342ea9-upower.service-iW9Rq6
Child 89142 exited with status 0
Prompt> exit
student@student:~/Desktop/csc415-assignment3-simpleshell-YuvrajGupta1808$
```