CSE344 – System Programming

Homework 2 – Report

Ömer Faruk Bitikçioğlu

161044010

Explanation

In this homework, a shell simulator is implemented using “fork()”, “execl()”, “wait()”, and “exit()” functions.

Macro MAX_COMMANDS, defines the maximum number of commands the program can handle. It is set to 20.

Macro DEALLOCATE_MEMORY simply deallocates the memory allocated for commands to avoid memory leaks.

We have a signal handler function in this program that simply ignores SIGINT, SIGQUIT, SIGTERM, SIGTSTP signals and just prints that the signal is arrived.

The program starts by checking if any arguments were passed to it on the command line, and if so, prints out usage information and exits if any unnecessary arguments are given.

The program then sets up signal handlers for SIGINT, SIGQUIT, SIGTERM, and SIGTSTP, which are used to handle user interrupts and termination signals. It also blocks SIGCHLD, which is a signal sent when a child process terminates, to prevent it from interrupting the main program loop.

In child processes these signals are not ignored. They are treated normally. If a SIGKILL signal sent by a child process, it also kills the parent process. The other signals do not have that effect in this program.

The main program loop repeatedly prompts the user for input and reads it from standard input using fgets(). It then parses the input into separate commands, delimited by the "|" character, and stores them in an array of strings.

If the user enters a ":q" command, the loop exits and the program terminates. If an error occurs while parsing the input or allocating memory for the commands, the program prints an error message and exits.

The program supports input redirection and piping between commands.

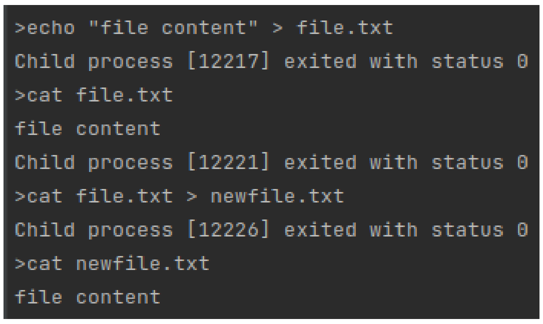
The program loop is similar to below:

- It takes a string of commands as input, splits them into individual commands, and stores them in an array.
- It creates a child process for each command, using fork().
- If a command is followed by a pipe character '|', it creates a pipe using pipe() and sets up the input and output file descriptors accordingly using dup2().
- If a command contains input redirection '<' or output redirection ‘>’, it opens the input file using open() and sets up the input/output file descriptor using dup2().
- It executes the command in the child process using execl().
- The parent process waits for the child process to finish using waitpid(), and prints the exit status of the child process. It also checks for any signals that caused the child process to exit.
- Error messages and signals that occur during execution are printed, and the program returns to the prompt to receive new commands.

- If the command “:q” is given, program terminates normally. Besides if you put ^D (EOF) to the input it causes the shell simulator to terminate normally.
- SIGINT, SIGTERM, SIGQUIT, SIGTSTP signals do not have any effect on the parent process. The parent process only prints that these signals are arrived and that’s it.
- The program continues to get inputs until stop flag is set to 1.
- You can define DEBUG_COMMANDS to print the given commands.
- In the end of the execution of a command line, the program creates a file named as timestamp at the execution time, and puts the child process information in it.

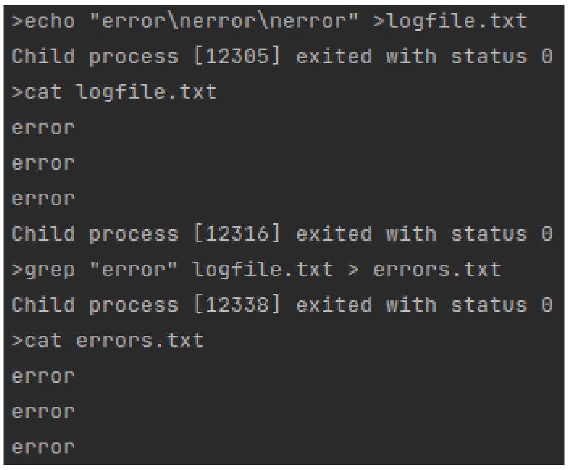
Test Cases

1. cat file.txt > newfile.txt - This command redirects the output of the cat command from file.txt to a new file called newfile.txt.



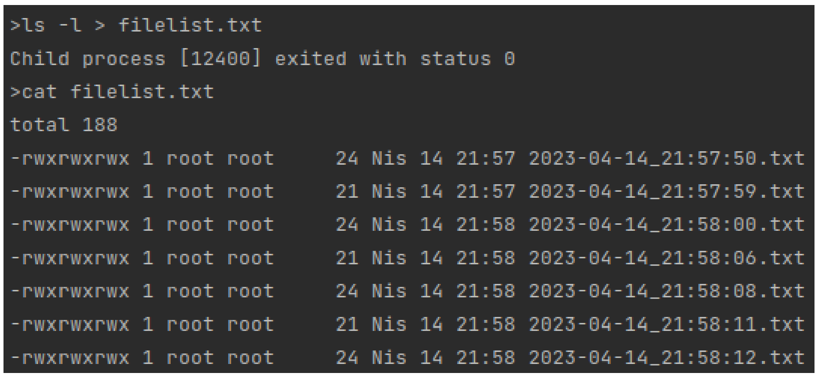
```
>echo "file content" > file.txt
Child process [12217] exited with status 0
>cat file.txt
file content
Child process [12221] exited with status 0
>cat file.txt > newfile.txt
Child process [12226] exited with status 0
>cat newfile.txt
file content
```

2. grep "error" logfile.txt > errors.txt - This command searches for the string "error" in the logfile.txt file and saves the output to a new file called errors.txt.



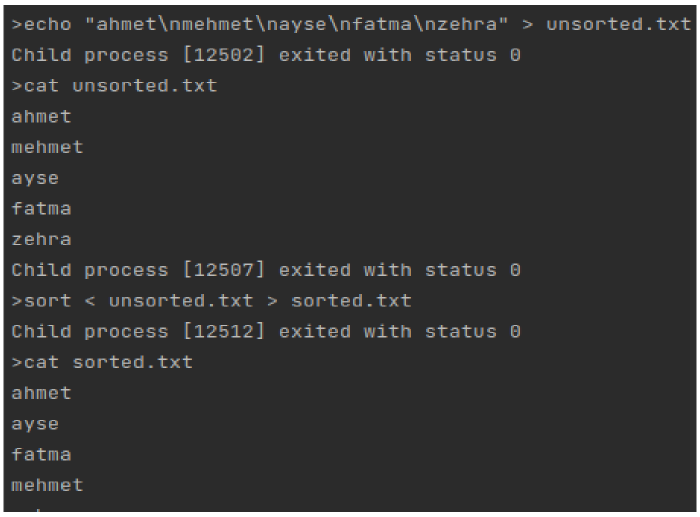
```
>echo "error\nerror\nerror" >logfile.txt
Child process [12305] exited with status 0
>cat logfile.txt
error
error
error
Child process [12316] exited with status 0
>grep "error" logfile.txt > errors.txt
Child process [12338] exited with status 0
>cat errors.txt
error
error
error
```

3. ls -l > filelist.txt - This command lists the files in the current directory in long format and redirects the output to a new file called filelist.txt.



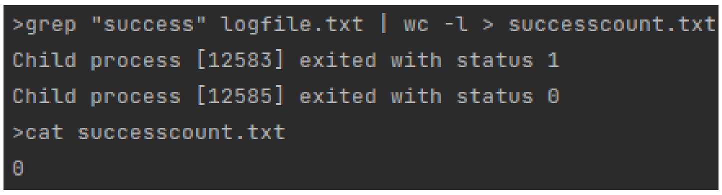
```
>ls -l > filelist.txt
Child process [12400] exited with status 0
>cat filelist.txt
total 188
-rwxrwxrwx 1 root root     24 Nis 14 21:57 2023-04-14_21:57:50.txt
-rwxrwxrwx 1 root root     21 Nis 14 21:57 2023-04-14_21:57:59.txt
-rwxrwxrwx 1 root root     24 Nis 14 21:58 2023-04-14_21:58:00.txt
-rwxrwxrwx 1 root root     21 Nis 14 21:58 2023-04-14_21:58:06.txt
-rwxrwxrwx 1 root root     24 Nis 14 21:58 2023-04-14_21:58:08.txt
-rwxrwxrwx 1 root root     21 Nis 14 21:58 2023-04-14_21:58:11.txt
-rwxrwxrwx 1 root root     24 Nis 14 21:58 2023-04-14_21:58:12.txt
```

4. sort < unsorted.txt > sorted.txt - This command sorts the contents of the unsorted.txt file and saves the output to a new file called sorted.txt.



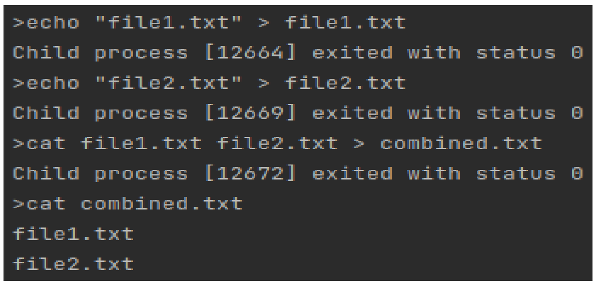
```
>echo "ahmet\nmehmet\nayse\nfatma\nzehra" > unsorted.txt
Child process [12502] exited with status 0
>cat unsorted.txt
ahmet
mehmet
ayse
fatma
zehra
Child process [12507] exited with status 0
>sort < unsorted.txt > sorted.txt
Child process [12512] exited with status 0
>cat sorted.txt
ahmet
ayse
fatma
mehmet
```

5. grep "success" logfile.txt | wc -l > successcount.txt - This command searches for the string "success" in the logfile.txt file and then counts the number of occurrences using the wc -l command. The output is then redirected to a new file called successcount.txt.



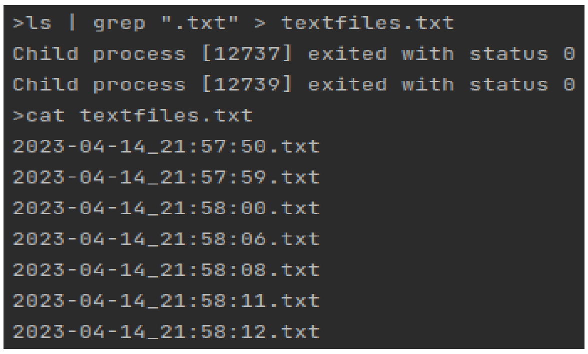
```
>grep "success" logfile.txt | wc -l > successcount.txt
Child process [12583] exited with status 1
Child process [12585] exited with status 0
>cat successcount.txt
0
```

6. cat file1.txt file2.txt > combined.txt - This command concatenates the contents of file1.txt and file2.txt and saves the output to a new file called combined.txt.



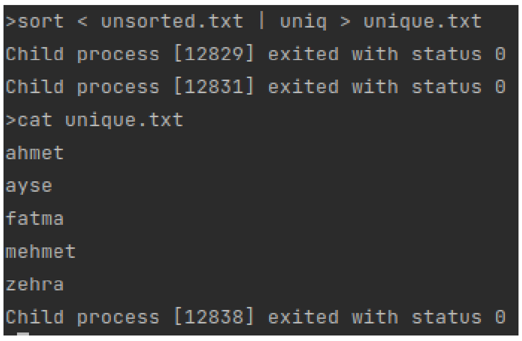
```
>echo "file1.txt" > file1.txt
Child process [12664] exited with status 0
>echo "file2.txt" > file2.txt
Child process [12669] exited with status 0
>cat file1.txt file2.txt > combined.txt
Child process [12672] exited with status 0
>cat combined.txt
file1.txt
file2.txt
```

7. ls | grep ".txt" > textfiles.txt - This command lists the files in the current directory and then searches for only the files that end with ".txt". The output is then redirected to a new file called textfiles.txt.



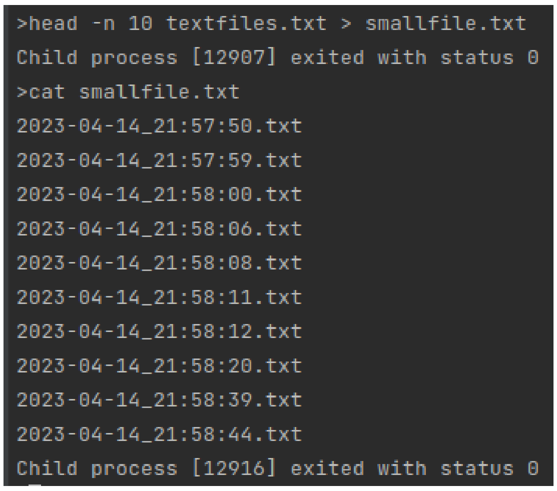
```
>ls | grep ".txt" > textfiles.txt
Child process [12737] exited with status 0
Child process [12739] exited with status 0
>cat textfiles.txt
2023-04-14_21:57:50.txt
2023-04-14_21:57:59.txt
2023-04-14_21:58:00.txt
2023-04-14_21:58:06.txt
2023-04-14_21:58:08.txt
2023-04-14_21:58:11.txt
2023-04-14_21:58:12.txt
```

8. sort < unsorted.txt | uniq > unique.txt - This command sorts the contents of the unsorted.txt file and then removes any duplicate lines using the uniq command. The output is then redirected to a new file called unique.txt.



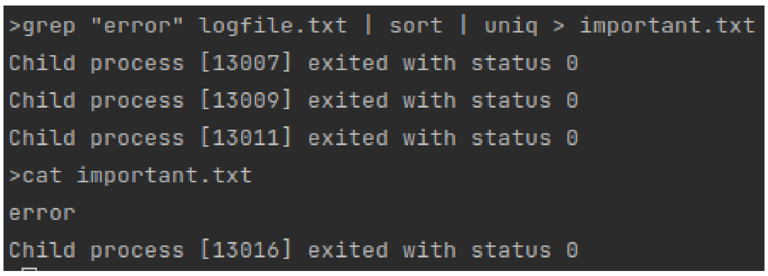
```
>sort < unsorted.txt | uniq > unique.txt
Child process [12829] exited with status 0
Child process [12831] exited with status 0
>cat unique.txt
ahmet
ayse
fatma
mehmet
zehra
Child process [12838] exited with status 0
```

9. head -n 10 bigfile.txt > smallfile.txt - This command selects the first 10 lines from the bigfile.txt file and saves the output to a new file called smallfile.txt.



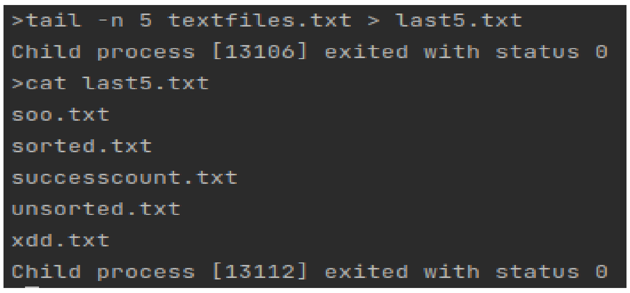
```
>head -n 10 textfiles.txt > smallfile.txt
Child process [12907] exited with status 0
>cat smallfile.txt
2023-04-14_21:57:50.txt
2023-04-14_21:57:59.txt
2023-04-14_21:58:00.txt
2023-04-14_21:58:06.txt
2023-04-14_21:58:08.txt
2023-04-14_21:58:11.txt
2023-04-14_21:58:12.txt
2023-04-14_21:58:20.txt
2023-04-14_21:58:39.txt
2023-04-14_21:58:44.txt
Child process [12916] exited with status 0
```

10. grep "important" logfile.txt | sort | uniq > important.txt - This command searches for the string "important" in the logfile.txt file, sorts the results, removes any duplicates, and saves the output to a new file called important.txt.



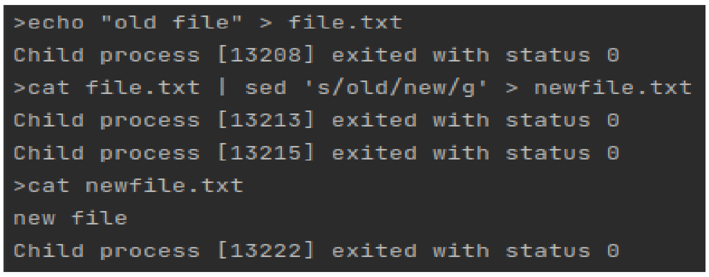
```
>grep "error" logfile.txt | sort | uniq > important.txt
Child process [13007] exited with status 0
Child process [13009] exited with status 0
Child process [13011] exited with status 0
>cat important.txt
error
Child process [13016] exited with status 0
```

11. tail -n 5 logfile.txt > last5.txt - This command selects the last 5 lines from the logfile.txt file and saves the output to a new file called last5.txt.



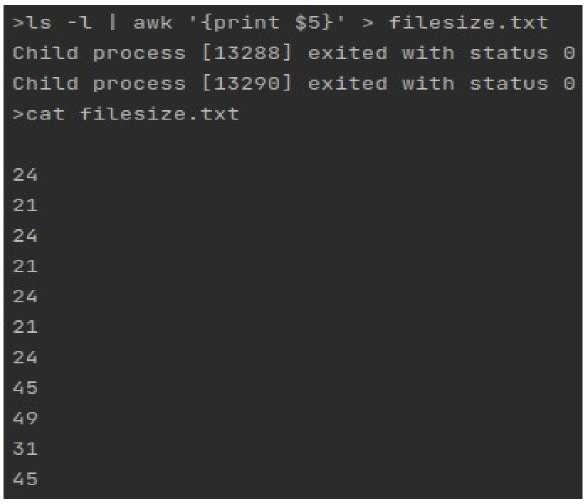
```
>tail -n 5 textfiles.txt > last5.txt
Child process [13106] exited with status 0
>cat last5.txt
soo.txt
sorted.txt
successcount.txt
unsorted.txt
xdd.txt
Child process [13112] exited with status 0
```

12. cat file.txt | sed 's/old/new/g' > newfile.txt - This command reads the contents of file.txt, replaces every occurrence of "old" with "new" using the sed command, and saves the output to a new file called newfile.txt.



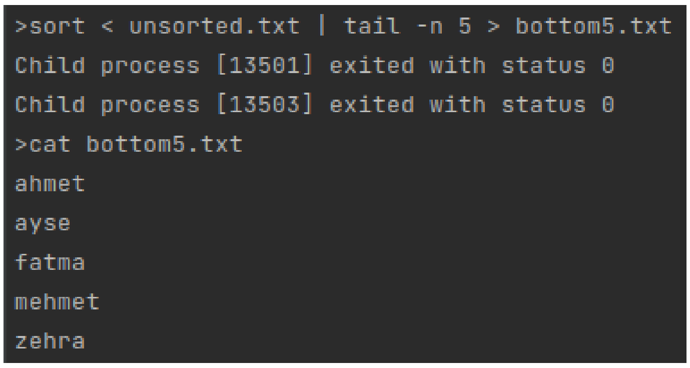
```
>echo "old file" > file.txt
Child process [13208] exited with status 0
>cat file.txt | sed 's/old/new/g' > newfile.txt
Child process [13213] exited with status 0
Child process [13215] exited with status 0
>cat newfile.txt
new file
Child process [13222] exited with status 0
```

13. ls -l | awk '{print $5}' > filesize.txt - This command lists the files in the current directory in long format and then uses the awk command to select only the fifth column (the file size). The output is then redirected to a new file called filesize.txt.



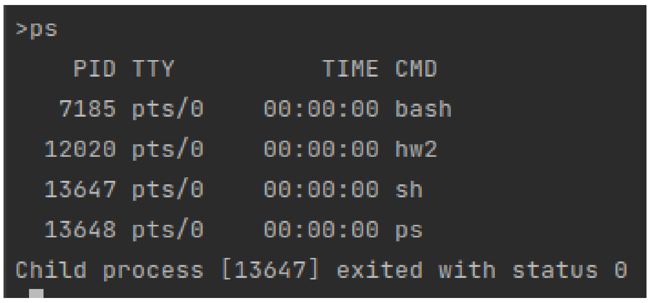
```
>ls -l | awk '{print $5}' > filesize.txt
Child process [13288] exited with status 0
Child process [13290] exited with status 0
>cat filesize.txt

24
21
24
21
24
21
24
45
49
31
45
```

14. sort < unsorted.txt | tail -n 5 > bottom5.txt - This command sorts the contents of the unsorted.txt file and then selects only the last 5 lines using the tail command. The output is then redirected to a new file called



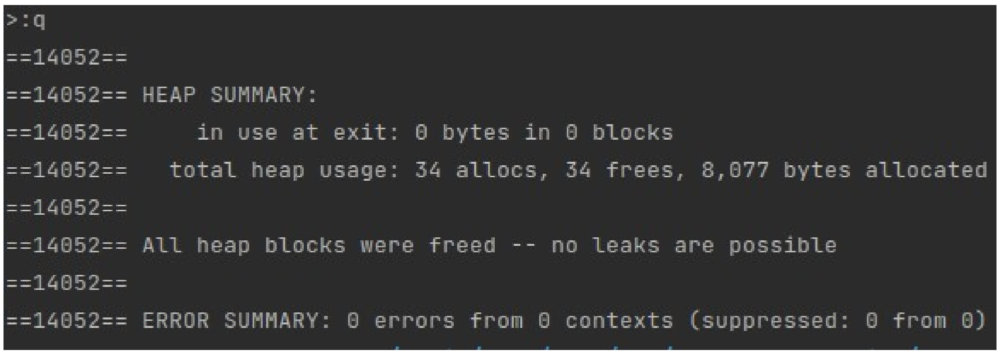
```
>sort < unsorted.txt | tail -n 5 > bottom5.txt
Child process [13501] exited with status 0
Child process [13503] exited with status 0
>cat bottom5.txt
ahmet
ayse
fatma
mehmet
zehra
```

Zombies ? No.



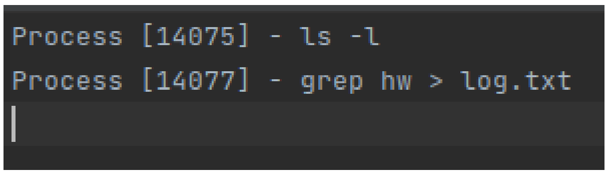
```
>ps
    PID TTY          TIME CMD
   7185 pts/0    00:00:00 bash
  12020 pts/0    00:00:00 hw2
  13647 pts/0    00:00:00 sh
  13648 pts/0    00:00:00 ps
Child process [13647] exited with status 0
```

Memory leaks? We do not have.



```
>:q
==14052==
==14052== HEAP SUMMARY:
==14052==     in use at exit: 0 bytes in 0 blocks
==14052==   total heap usage: 34 allocs, 34 frees, 8,077 bytes allocated
==14052==
==14052== All heap blocks were freed -- no leaks are possible
==14052==
==14052== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Log sample for a command “ls -l | grep hw > log.txt”



```
Process [14075] - ls -l
Process [14077] - grep hw > log.txt
```