Programming Assignment 1 Report

Hüseyin Alper Karadeniz

Operating Systems (CS 307) / Fall 2022-2023

1 Piped Commands Selection

With the rapid development of technology in recent decades, the activities we can do on our computers have changed drastically. As the capability of computer technologies continues to increase, multitasking has become an important concept of this technology age. With multitasking, a computer needs to run many applications concurrently. Moreover, in addition to software optimization, this requires a powerful processor and well-controlled RAM usage.

Among the Linux commands, there is a command called **"free"** giving the summary details of the RAM usage of the computer. It helps monitoring the resource usage, which is crucial for the management of today's computer systems. I have picked this command because it allows us to be aware of the current status of both physical and swap memory usages of our computer, making it a helpful command in daily-life cases, but also interesting to look at.

In the manual page, I have picked an option called **"--human"** since this option converts the appearance of the RAM usage in a human readable format by usually converting the amounts into megabytes and gigabytes accordingly. With this option, it becomes easier to interpret the RAM details because it makes the data easier to comprehend with more meaningful scales.

Under **"grep"** options, I have picked **"-A 14"** in order to output 11 lines following the match since there is a long description for the option I chose there. Furthermore, I have added **"-e"** option for solving the special character problem caused by the usage of "-" character inside of the second command. As a result, my piped commands for manual search became as follows:

```
man free | grep -A 14 -e --human > output.txt
```

2 Process Hierarchy

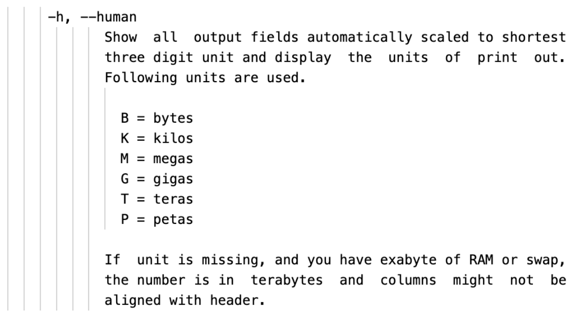
In my C program, I have used *fork*, *wait*, and *exec* system calls as we discussed in the lectures. Due to the fact that I have two child processes in addition to my main process, I had to use *fork* function. Moreover, I needed to use the *wait* function because the shell process had to wait for the first one to complete successfully in order to start the second child process and then finish the program. Furthermore, I had used *exec* function to execute my commands and receive results from them.

In my C program implementation, I have a SHELL (main) process calling two child processes as their parent process. In the program, after the main initializations, the first child process begins. The first child process is responsible for MAN process where the descriptions of "free" command is searched in the manual. It stores its output in the write-end of the pipe created in the initialization phase. After the execution of the first child process, the second child process begins. The second child process is responsible for GREP process where it founds the description of "--human" option by reading the command from the read-end of the pipe and executing it. After all these steps, the program saves the description found by the last execution into a new file called *output.txt* so that saves the execution result of the piped commands.

During the program, SHELL process waits for its child processes, MAN and GREP processes, to finish one by one because the output of the first child process is used as the input of the second child process. These processes are connected by a pipe created in the initialization phase of the program.

3 Program Result

In the *output.txt* file, there is a result as follows:



```
-h, --human
       Show  all  output fields automatically scaled to shortest
       three digit unit and display  the  units  of  print  out.
       Following units are used.

         B = bytes
         K = kilos
         M = megas
         G = gigas
         T = teras
         P = petas

       If  unit is missing, and you have exabyte of RAM or swap,
       the number is in  terabytes  and  columns  might  not  be
       aligned with header.
```