OPERATING SYSTEMS – ASSIGNMENT 2

Writing Your Own Shell

Submission Date: 16.04.2020, 23:55

הנחיות:

- קראו היטב את העבודה עד הסוף.
- ניתן להגיש את העבודה בזוגות. לא ניתן להגיש את העבודה בקבוצה מעל שני אנשים.
- יש להגיש את העבודה בקובץ zip שבתוכו נמצאים קובץ/קבצי קוד C.
- שם הקובץ שיוגש למערכת ההגשה יהיה מורכב מת"ז של המגיש/ים. לדוגמה:

עבור הגשה ביחיד - 111111111.zip

עבור הגשה בזוג - 111111111_22222222.zip

- במקרה של הגשה בזוגות, רק אחד מבני הזוג יגיש את העבודה במודל.
- גם בתחילת כל קובץ קוד יש לרשום את שמות ומספרי ת.ז. של המגישים.
- איחור במועד ההגשה יגרור הורדה של ציון, 5 נק' לכל יום איחור או חלק ממנו.

  בכל מקרה לא יהיה ניתן להגיש מעבר ל-3 ימי איחור ממועד ההגשה המקורי.

  במקרים חריגים בלבד יש לפנות למרצה כדי לקבל אישור על הגשה באיחור.
- שאלות לגבי העבודה יש לשאול בפורום באתר הקורס ("מודל") או בשעות קבלה.

Introduction:

The object of this assignment is to gain experience with some advanced programming techniques such as process creation and termination, and file descriptors. To do this, you will be writing your own command shell - much like bash.

What is Shell?

A shell program is an application that allows interacting with the computer. In a shell, the user can run programs and also redirect the input and output. Shell additionally offers features such as line editing, history, file completion, etc.

"A shell is an interface that allows you to interact with the kernel of an operating system."

Shell Workflow:

A shell parses commands entered by the user and executes this. The workflow of the shell will look like this:

1. Startup the shell
2. Wait for user input
3. Parse user input
4. Execute the command and return the result
5. Go back to 2

The shell is the parent process. This is the main process in our program which is waiting for user input. However, we cannot execute the command in the parent process itself, because of the following reasons:

1. A command with error will cause the shell to stop working. We want to avoid this.
2. Independent commands should have their own process blocks. This is known as isolation principle.

Shell pseudo-code:

```
int main (int argc, char **argv)
{
 int childPid;
 char * cmdLine;
 parseInfo *info;
 while (1){
      cmdLine= readline(">");
      info = parse(cmdLine);

      childPid = fork();

      if (childPid == 0)
      {
         executeCommand(info); //calls  execvp
      }
      else
      {
         waitpid(childPid);
      }
  }
 }
```

Required Features:

Here are the features you should support in your shell:

1) The shell should indicate the current working directory using **pwd**.

2) **cd** should change the working directory

3) Create a file by the command **Cat >** <filename> or **nano** <filename>.

4) **cat** <filename>. View the contents of the file.

5) **wc** [options/flags] filename. The **wc** command should support the following flags **wc -l** <filename>, **wc -c** <filename>, **wc -w** <filename>.

6) **cp** <file1> <file2>. Copy the file <file1> to file <file2>

7) The shell has to support **pipe** operator "|" between processes.

8) **sort** <filename>, **sort -r** <filename>. **sort** Command should sort a file, and support the **-r** flag. (printing the file after sort)

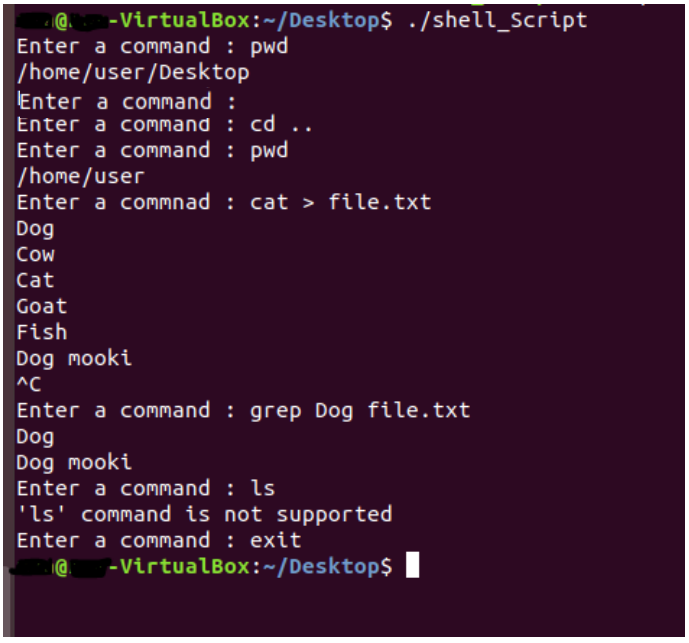
9) **grep** Return lines with a specific word/string/pattern in a file. grep [options/flags] [Pattern] <filename>. Should support in the following options(flags): **grep -c** [pattern] <filename>

10) **man** [command name]. A man command can have one argument (command name) and then it will display a manual only for the commands that you have to implement in your own shell, and then it will display a manual for all the commands we requested to execute.

11) **exit** should terminate your shell process.

- Each command must check whether the parameters are correct, and whether or not to print an appropriate error message (much like a real shell does).

example:



Helpful Resources:

Here is a list of resources you may find helpful.

Linux/UNIX

The following functions are some helpful functions (consult the man pages for details):
fork, exec, execvp, wait, waitpid, kill, dup, pipe, strncmp, strlen, malloc, free, getcwd, chdir , open, close, readline, gets, fgets, getchar.

Enjoy!