HW3: Due 02/20/2022

TechShell

A simple shell supports shell commands with I/O redirection. Write a C program that repeatedly allows a user to enter input (interpreted as a shell command). Your program should parse the command line and determine whether it contains I/O redirection (< for input from a file, > for output to a file). It should also have built-in support (i.e., not executed in a forked child process that uses an **exec** call) for three commands:

(1) **cd**: changes the current working directory to the one specified;
(2) **pwd**: displays the current working directory; and
(3) **exit**: exits from the shell.

All other commands must be executed in a forked child process that uses an **exec** call. Your program should provide an informative shell prompt that contains the current working directory, followed by a $, and ending with a space. The prompt should be updated if the current working directory changes. Here's an example that illustrates this:

```
/home/ibrahim$ cd ..
/home$
```

You must include a separate file called README (a text file with **no** extension) that provides the following details:

(1) Your name and your partner's name;
(2) The inner workings of your program; and
(3) If you have not fully implemented shell functionality as described above, list the parts that work (and how to test them if it's not obvious) so that you can receive partial credit.

You must include a **Makefile** so that your program can be easily compiled via **make**. A typical **Makefile** for your program may contain the following (supposing that your single source file is **techshell.c**):

```
all: techshell

techshell: techshell.c
	gcc -o techshell techshell.c

clean:
	rm -f techshell
```

You must also include output from testing your program using **script**, a program that records terminal sessions. See the sample at the end of this document. A **script** session of your program is started (and saved in a file called **typescript**) by starting **script** from the command line, running your program, testing it, exiting it, and finally exiting **script** as follows:

```
ibrahim@ibrahim-latech:~$ script
Script started, file is typescript
ibrahim@ibrahim-latech:~$ ./techshell
/home/ibrahim$ cd Desktop
/home/ibrahim/Desktop$ exit
ibrahim@ibrahim-latech:~$ exit
exit
```

```
Script done, file is typescript
ibrahim@ibrahim-latech:~$
```

Make sure to implement good programming practices. For example, your **main** function should merely be a driver (i.e., farm out various functions to subroutines).

Submit a single **.zip** archive file that contains the following:

(1) The README file described above;
(2) The Makefile described above;
(3) All **.c** source files required to compile your program (do not include any executable or object files); and
(4) Output from testing your program via **script**; make sure to demonstrate:
   1. Simple UNIX commands;
   2. The three built-in commands (**cd**, **pwd**, and **exit**);
   3. I/O redirection (< and >); and
   4. Error conditions (e.g., invalid commands, errors, etc).

Your program should be able to handle the following examples (via sample **script** output). Note that your program will be tested with more than these commands (including more error handling cases):

```
ibrahim@ibrahim-latech:~$ ./techshell
/home/ibrahim$ cd Desktop
/home/ibrahim/Desktop$ pwd
/home/ibrahim/Desktop
/home/ibrahim/Desktop$ cd ..
/home/ibrahim$ pwd
/home/ibrahim
/home/ibrahim$ ls -lh techshell.c
-rw-r--r-- 1 ibrahim ibrahim 3.9K Oct 27 15:08 techshell.c
/home/ibrahim$ chmod 400 techshell.c
/home/ibrahim$ ls -lh techshell.c
-r-------- 1 ibrahim ibrahim 3.9K Oct 27 15:08 techshell.c
/home/ibrahim$ ls /root
ls: cannot open directory /root: Permission denied
/home/ibrahim$ horseface
Error 2 (No such file or directory)
/home/ibrahim$ cd /root
Error 13 (Permission denied)
/home/ibrahim$ ps
  PID TTY          TIME CMD
 6271 pts/3    00:00:00 bash
 7415 pts/3    00:00:00 techshell
 7460 pts/3    00:00:00 ps
/home/ibrahim$ whereis ps
ps: /bin/ps /usr/share/man/man1/ps.1.gz
/home/ibrahim$ /bin/ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ibrahim     444  0.0  0.0   7236  5244 pts/1    Ss+  07:57   0:01 bash
ibrahim    6271  0.0  0.0   7220  5032 pts/3    Ss   14:18   0:00 bash
ibrahim    7415  0.0  0.0   2172  1324 pts/3    S+   15:08   0:00 ./techshell
ibrahim    7483  0.0  0.0   5228  2372 pts/3    R+   15:12   0:00 ps u
/home/ibrahim$ ls
Desktop  Documents  Music  techshell  Downloads  Pictures  techshell.c
Videos
/home/ibrahim$ ps u > ps.out
/home/ibrahim$ wc -l < ps.out
```

```
5
/home/ibrahim$ wc -l < ps.out > wc.out
/home/ibrahim$ cat wc.out
5
/home/ibrahim$ ls
Desktop  Documents  Music  techshell  Downloads  Pictures  techshell.c
Videos   ps.out     wc.out
/home/ibrahim$ rm ps.out wc.out
/home/ibrahim$ ls
Desktop  Documents  Music  techshell  Downloads  Pictures  techshell.c
Videos
/home/ibrahim$ find . -name Makefile
./Makefile
/home/ibrahim$ ./techshell
/home/ibrahim$ cd Desktop
/home/ibrahim/Desktop$ exit
/home/ibrahim$
/home/ibrahim$ exit
ibrahim@ibrahim-latech: ~$
```

Make sure to comment your source code appropriately, and to include a header providing your name.

Hints:

(1) Your previous C programs will greatly help;
(2) To get the value of an environment variable, **getenv** is your friend;
(3) The **exec** function of most use may very well be **execvp**;
(4) The following may just work to redirect **stdin** from a file:

```
FILE* infile = fopen(my_input_file, "r");
dup2(fileno(infile), 0);
fclose(infile);
```

(5) The following may just work to redirect **stdout** to a file:

```
FILE* outfile = fopen(my_output_file, "w");
dup2(fileno(outfile), 1);
fclose(outfile);
```

(6) To build command line arguments, **malloc** and **realloc** are your friend;
(7) To copy strings, you've probably used **strcpy**; however, you may want to try **strdup**;
(8) To handle errors, **errno** and **strerror** are your friends; and
(9) **`#define DEBUG`** may greatly help with debugging (i.e., use toggle-able print statements).