CSCI 347 - Computer Systems II

Assignment 1

Total Points: 130

Part 1

Find the attached microshell on Canvas (microshell.c). Extend this microshell to do simple argument processing in the shell. Specifically, do the following:

• Read the man pages on execve(2) and related functions (exec(3)). (Yes, you will need to use a different exec... function than what is currently used in the microshell.)
• Read the man page on malloc and if needed, consult a C programming text on how to use malloc(3) and the C sizeof function. Also, read about free(3).

(10 points)
• Create a new private project on gitlab.cs.wwu.edu with the name of "csci347-fall- 2022"
• Clone your new project to a lab machine. After you clone your project, add a README.md that has your name and quarter in it. Next, a add directory named "ush" to your project. All ush files must be in this directory. Copy the microshell.c to your project's ush directory and name it ush.c. Commit and push both files.

(60 points)
Write a function "arg parse" that must have the prototype:

char ** arg_parse (char *line, int *argcptr).

The "line" parameter points to the line (character array) that contains the command to be processed. The "argcptr" parameter points to an integer variable where this function must put the number of arguments found in the "line". The return value must be a pointer to a malloced area that points into the line parameter and matches what is needed by execve for the "argv" parameter. The malloced area should be the exact size needed, not too small and not too big. You should call malloc(3) only once. Make sure you check for malloc(3) errors. (Note: You should use argcptr only just before a return. You should have a local "argc" variable that keeps track of the number of arguments and then use argcptr to save argc to the correct place just before arg_parse() returns.)
Do your processing by breaking up the line on spaces. Do not do more complex processing at this time. You should not copy any characters out of the line parameter. You only put zero characters in the line parameter and set pointers to characters in the line parameter in your malloced area.

Notice that two or more consecutive spaces is equivalent to a single space.
For example, a call to arg_parse with the parameter line

mycommand arg1 nextarg arg3 not-arg-5-but-4

should return a pointer to an “array” of 6 elements. The first element (index 0) should point to the character ’m’ in mycommand, the second element should point to the ’a’ of arg1, and so forth. The last element should be a NULL pointer. The characters “mycommand” are not copied to any other storage. The space just after “mycommand” is turned into a zero character so “mycommand” is a zero terminated string. Also, leading and trailing spaces should be ignored! (Before the command name and after the last argument.)
Note: Do not use strtok(3) from the string library.

(10 points)
Modify the microshell function processline() to use arg parse to run commands with arguments. Call arg_parse before the fork() call and if there are no parameters on the line, do not fork(), but just return from processline() without doing more. Do not forget to free the pointer returned by arg_parse() in the parent shell process. You will also need to modify the child code in processline() to call a different exec function. Do not use execve. You must not have any “int *” variables in processline().

(10 points)
At this point, you should have a shell that can run commands with command line arguments. Create a script to show you have a working shell at this point.

Part 2

(40 points)
Now, add double quote processing to arg_parse(). Implementing arg_parse() with quote processing first will cost you points. If a quote is found, look for a matching quote and everything between the two quotes is considered to be consecutive characters in the current argument.

For example:

prog "this is a single arg"

has 2 arguments and

prog this" "is" "a" "single" "arg

also has exactly 2 arguments. And again,

prog thi"s is a s"ingl"e a"rg

has exactly 2 arguments and is identical in result to the above 2. Also, you can make a program name with spaces like:

"prog name with spaces"

Note: a quote may appear in the middle of an argument as you can see by the second example above. Also, the actual quote character is removed from the final version of the argument. If you find an odd number of quotes in the line, print an error message to “standard error” and stop processing the current line.

• Make sure you have no warnings when compiled with “gcc -Wall” on Ubuntu as provided by the department in the labs. (Remember, you have remote access to the linux-01 to linux-12 machines. Domain is cs.wwu.edu. Some kind of ssh client is required to remote access these machines.)