Code Review

Author Name: Bryan Pham

Reviewer(s) Name: Samantha Murray Tuesta

Date: May 9, 2022

Project Name: Microshell

Reviewed File(s): builtin.c

## Code Review

The reviewer shall use the below table to document code defects and suggestions for improvement. Severity records how serious the infraction is: high (H), medium (M) or low (L). The first row (after the header) is an example. Remove it before submission. Check the appendix for a suggested C code review checklist.

|  |  |  |  |
| --- | --- | --- | --- |
| Function name | Line number | Severity | Brief Description |
| builtIn | 44 | L | Implementation is clean and follows requirements |
| exitProgram | 89 | L | Great catch with the conditional instead of just exiting! |
| pwd | 104 | L | Nice use of library functions to configure your file path! |
| cd | 122 | M | You are printing the change of directory but I am not clear as to if it actually does change the directory! But it looks good! |
| statData | 144 | L | This implementation looks great! Is there a way to reduce the amount of print statements used? If not, this is still cool, I like how you included the conditional into the print statement! |
| Timeprint | 190 | L | Great use of time functions! And great formatting! I know this is just a helper function but it’s cool so I wanted to say good job! |
| tailPrint | 201 | M | I think as long as we are allowed to use the fopen and fclose funcitons that this is great! If not then maybe find another way of implementing the open file command! Looks great! |
| envProgram | 239 | L | Helper function that looks really helpful! |

The reviewer can add any other notes they have in the below space.

Great job Bryan!

## Plan for improvement (to be written by the code author)

My plan for improving is to go back and look at my tailPrint and implementing it a different way. Going to go back and revisit my cd function to make sure, it actually does what it is suppose to.

## Appendix – C Code Review Checklist

1. Commenting:  top of file, start of function, code that needs an explanation
2. Style is consistent and follows style guidelines
3. No redundant, dead, commented out, unused code & variables
4. Conditional expressions evaluate to a Boolean value; no assignments
5. Parentheses used to avoid operator precedence confusion
6. All switch statements have a default clause; preferably an error trap
7. Single point of exit from each function
8. Loop entry and exit conditions correct; minimum continue/break complexity
9. Conditionals should be minimally nested (generally only one or two deep)
10. Are "magic numbers" avoided? (use named constants rather than numbers)
11. Variables have well-chosen names and are initialized at definition
12. Input parameter checking is done
13. Error handling for function returns is appropriate
14. Null pointers, division by zero, null strings, boundary conditions handled
15. Buffer overflow safety (bound checking, avoid unsafe string operations)
16. Large arrays are dynamically allocated on the heap.
17. Pointer variables are named in a consistent fashion.
18. Pointers are initialized to NULL.
19. Pointers are tested for NULL before being referenced.
20. Dynamically allocated memory is deallocated when no longer needed.
21. Does the code match the detailed design (correct functionality)?
22. Is the code as simple, obvious, and easy to review as possible?