Code Review

Author Name: Preston Duffield

Reviewer(s) Name: Sky Duryee

May 5, 2023

Project Name: Project 2

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 |
| N/A | 0 | L | The checklist says there should be “Commenting: top of file.” I don’t know what we’re supposed to put there though |
| exitProgram | 59 | L | I do not know if Elglaly has any expectations for what should happen when the user *doesn’t* enter a number. Seems to me like bash exit will print an error “numeric argument required” but then exit even if a number was not provided. You might want to check if you have to recreate this behavior |
| exitProgram | 61 | L | Even though I would not consider this a memory leak, it seems like we are expected to free() our args before exiting the program. This is something you might want to consider once you get everything else done. |

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

Your code is very good! It is compact and easy to read and you used perror in a good way.

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

For the exit function, I noted in the reqs that My function exits correctly when no input is sent.  
For the exit function, I made modifications to free the args.  
I also commented at the top of each file.

## 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?