Permanent Aliasing

By: Ian Green, Jacob Longo, and Bitao Jin

Project Description

- The `alias` command currently only creates an alias while the bash terminal is open
- We would modify this command to affect the user's `.bash_alias` file and add the alias as a permanent feature
- To use this feature while maintaining backwards compatibility, we will make it available via the `-s` option
    - Permanently add alias: `alias -s hello='echo'`
    - Temporarily add alias (during lifetime of bash terminal): `alias hello='echo'`

Why are we interested?

- The `-s` command will save inexperienced users time and headache by editing `~/.bash_aliases` or the `~/.bashrc` file
- For experienced users, this change will help reduce the time necessary to add a permanent alias
    - eliminate the need to edit the file in the home directory

Building the Kernel/Bash 1: Research

- Professor Johnson's resources including his and group 04 Kernel builds
  - Thanks to Isaiah, Ona, and Moriah!
- We found the following documentation: https://www.gnu.org/software/bash/manual/bash.html#Basic-Installation
  - Aided in the installation of Bash and how to implement our solutions to the problems that we encountered
- https://linux.die.net/man/

Building the Kernel/Bash 2: Problems

1. It was initial very hard to find the alias.c file
2. Bash did not install on VM OS properly (in /bin)
    a. Occasionally successful, hard to replicate with identical commands
3. File editing
4. Multi-character CLI arguments
5. Figuring out what a .def file does and is used for
6. Internal BASH commands (getopt(), usage())

Building the Kernel/Bash 3: Solutions

1. Alias.c is not part of the Kernel internally and had to be installed through Bash
2. Built all the functionality externally from bash and worked into alias files
3. Built bash separately from ‘terminal’ then ran it
    a. Interestingly, we used Bash to run our custom Bash
4. Changed from --perm to -s to match other alias CLI arguments
5. Read GNU documentation on internal Bash functions
6. Using ‘make clean’ to cleanup before compilation

Demonstration!

Key Takeaways

- More experience in C
  - File manipulation ‘in-place’
  - String manipulation (strcmp, strstr, strcat, strcpy)
- The distinction between Bash and the Kernel and how command line processes are implemented
- File paths processings - finding $HOME without using standard relative paths
- Buffer sizing for worst case scenarios
  - Finding max string lengths

Hope you enjoyed!

Any questions?