Terminal Velocity

A novel approach to the interface of Command Line Shells

CSC318 Group 3

Philip Bilodeau
g3bilode

Dawing Cho
g3chodaw

Ben Hu
g3benhu

Maxwell Huang-Hobbs
g4rbage

Rui Qiu
c3qiurui

ABSTRACT

Command line shells are powerful pieces of software, in that they provide a simple set of rules to operate complex systems. Moreover, they act as gateways to learning to program, as many development tools exist only as command line tools. However, typical computer users see them as intimidating and unnecessary for day to day use. Though many features exist to make command line shells more usable for experts, there is little existing work that attempts to address the initial CLI learning experience. In this document we propose a novel take on the interface of a command line shells, as well as the process of user research and iterative design taken to create it.

Categories and Subject Descriptors

Human Computer Interaction [**Interaction Paradigms**]: Command Line InterfacesGraphical User Interfaces

General Terms

Human Computer Interaction

Keywords

Command line interfaces, shell, learner, learning curve

1. INTRODUCTION

The problem space we intended to target was the initial learning experience of command line shells, focusing on new and infrequent users (i.e. those learning to use a command line shell for the first time).

This problem stems mainly from the design of command line interfaces. The primary use case where a Command Line Interface is more effective than a simple Graphical User Interface (GUI), is interfacing with complex systems that require many configuration options. [1] Command line interfaces also have the benefit of being faster than GUIs in many cases, as all interfacing is done through one focal point; thus, one's attention does not need to be drawn around a changing UI.

These traits make shell CLIs effective tools for systems administration. As a result, they are perceived and treated more as an engineer's tool, rather than an interactive program. Users are expected to read large manuals before trying to use a CLI, and something will inevitably go wrong. The debug/error messages that get displayed are targeted more to people with an existing understanding of the underlying systems, as opposed to new users just getting started with shells.

Despite being unfairly relegated to systems administration, many Command Line Tools are well suited to an average user's day-to-day operations. Many CLI-specific tools are faster, simpler ways of accomplishing common tasks such as file globbing and permissions management. Furthermore, many debug tools for common systems are usable only from a CLI (networks, etc).

2. INITIAL RESEARCH AND USER NEEDS

2.1 Research

Our research plan was to sample from random individuals, targeting mainly beginners or potential beginners in learning CLIs. We used three research instruments while conducting the test. Our sessions began with a set of tasks/observation, followed by a questionnaire for general feedback, and a detailed interview afterwards. Each session lasted about 30 minutes, and users gave full consent through our forms. By using our research instruments, we were able to probe our target audience, and understand their opinions and response towards command line interfaces.

2.2 User Needs

From the findings of our initial research results, we established what the major needs of our participant base seemed to be and their perceived issues concerning current command line interfaces. Generally, participants felt that error output with current interfaces could be cryptic and unhelpful and would like for error messages to be more straightforward and helpful for new users. A related issue, indication of successful or unsuccessful execution, was found to be a poten-

[1] The typical way of handling this in a GUI is with a series of drop-down menus, but as the number of configuration options rises, this becomes increasingly complex.

tial area of weakness with current CLIs as occasionally users would be unsure about the changes made by commands with no clear return message (e.g. 'rm', 'cd', 'touch').

State and scope representation was found to be a major issue especially among new users and could be remedied with visual cues to better bridge the transition between GUI and CLI. Command name discovery and recall was a significant issue among both the inexperienced and intermediate participants in our trials. Command names were found to be difficult for new users to grasp as the name does not seem to directly relate to the function it accomplishes (e.g. 'grep', 'touch'). Command name discovery was also a significant problem amongst newer users. For intermediate users, especially infrequent users, command name and command usage recall was the parallel to command discovery. A frequent suggestion in both cases was the incorporation of command querying in some capacity or another to remedy these issues.

Potential eye fatigue was a minor issue that was found during our research, with some participants feeling that using a CLI for long periods of time could be a very tiring experience.

3. DESIGN GOALS

After evaluating the data from our initial research, we generated a list of key problems, design principles, and requirements to keep in mind as we went forward with initial prototyping of a design for our interface.

3.1 Problems to Address

- The complexity of the shell for new or casual users with limited experience.
- The difficulty of learning and remembering command names and techniques.
- The limited feedback provided by current command-line interfaces (mainly recovery from error and acknowledgement of success)
- Quality of life improvements that experienced users may appreciate

3.2 Design Principles

- Simplicity and clarity without sacrificing usability or power.
- Intuition-building.
- Fully-customizable.
- Minimalist and lightweight in design.

3.3 Functional Requirements

- Improve and provide easier to use command-line interfaces
- Requires users to have an understanding of computers

3.4 Technical Requirements

- Must be compatible with or have similar functionality as existing libraries of commands
- Should be relatively lightweight (similar to current interfaces) in terms of system resource usage
- Should be able to be run natively or be easily accessible from a Microsoft Windows operating system as this is the most common operating system used by our target user base

3.5 Usability Requirements

- Must be easier to use and learn than a regular command-line interface
- Provide features that will make it easier to use (natural language search, auto-comp, etc.)
- Provide error correction or suggestions on incorrect input
- Return helpful, specific error messages when required.
- Be able to easily 'undo' commands if a mistake is made (e.g. rm'ing the wrong batch of files)

4. INDIVIDUAL EVALUATION AND INITIAL PROTOTYPE

The initial design was built on top of the metaphor of a back-and-forth chat. This was done in an attempt to replace the now defunct metaphor of a teletype printer that command line shells are currently based on. On top of this, we added features to help users deal with common problems encountered during the interactive trial.

One feature that experienced and beginner users liked was the persistent graphical file representation. Users had an easier time distinguishing file types, and liked that they did not need to type 'ls' each time they changed directories. Beginners occasionally tried to type out their desired actions in English (e.g., 'find file)', so the built-in keyword query-able feature helped users not only use commands, but also discover new ones. We planned for a large library of aliases that link to existing commands (e.g.,. 'ls', 'dir', 'list', and 'directory' all print the contents of the current working directory), to help ease the required memorization and learning curve of using different commands. Another related feature that was helpful to users was a clear conveyance and feedback for successful commands as well as program exit statuses. The 'chat' metaphor was added in to create a distinction between users and non-users, as well as make it easier to distinguish input and output.

There were additional features that we also believed would be beneficial, but were not able to due to time and implementation complexity issues. One of the features is command auto-completion and auto-correction as users typed. Users found that they often mistyped commands, and this feature is frequently used in many text editors. We also considered an 'undo' feature, which would allow users to revert their actions if they made a mistake. Finally, syntax highlighting for commands/flags/arguments would be a simple quality of life improvement that would help all users when using the command line.

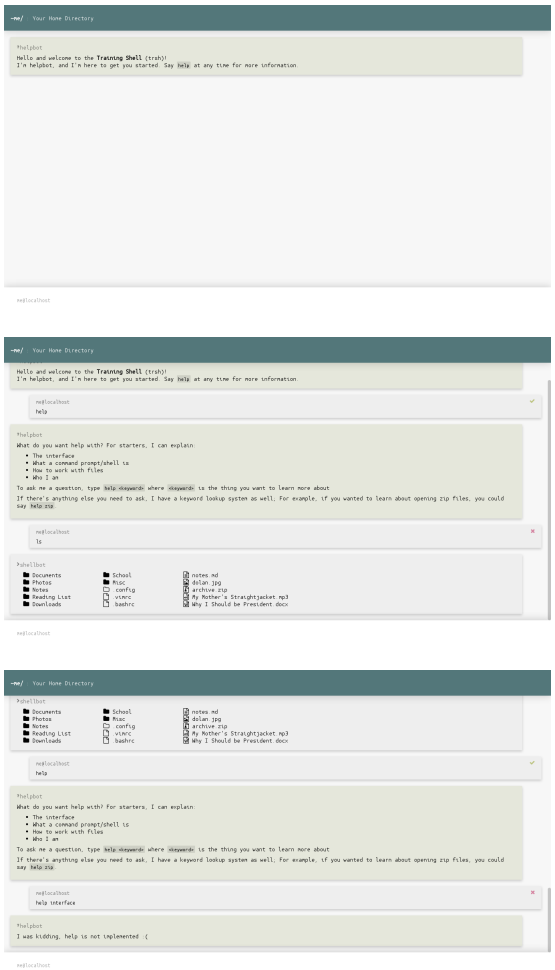

Figure 1: The Initial Prototype. Note the segmented messages, and the return statuses indicated in the top right of the command blocks

Combining our added features and semi-implemented quality of life improvements resulted in a more intuitive and fluid interface for our prototype. We aimed to add onto the command line interface, enhancing the feeling and features that existing users know, as well as provide a beginner friendly layout for new users to use.

5. CONTINUED USER RESEARCH AND ITERATION ON DESIGN

After finalizing our initial prototype design, we went on to evaluate the design by having expert shell users participate in another round of trials and testing. Users stepped through the initial prototype and asked to use it as if they were using a normal shell. From this round of testing the following suggestions and feedback were given from participants:

- Auto-completion & syntax highlighting would be welcome features if implemented, among both inexperienced and intermediate users. Help querying based on keywords, partial commands, or natural language was suggested (e.g. someone could 'ask' 'How do I go to root directory?')
- Natural language search & keyword searching would flatten the learning curve for beginners with respect to command discovery, recall, and usage significantly
- More concise man pages are a welcome quality of life addition for intermediate and inexperienced users
- The Red / Green colour scheme is potentially difficult to use for individuals who are color blind.
- Error reporting could be more visible and obvious.
- Users found sectioning output off into distinct blocks to be helpful when trying to parse output.
- The alignment scheme taken from modern chat UIs (user on right, response on left) runs counter to existing conventions for formatting blocks of information (header on left, sub-information indented).
  - Users were reluctant to pick up on the idea of a shell as a back & forth conversation, so the metaphor of a chat is not a helpful one
- Sidebar should be toggle-able, as it may take up too much space when working in small windows.

The final prototype included a persistent display of the current working directory in a 'sidebar', ' an annotated full path in the top bar, in order to ease user confusion as to their current working directory and path. It also included keyword based help searches, in order to find commands based on the functionality instead of based on their names. It also included a changed message display scheme for displaying the 'command, response' relationship in a more standard way. This all but discarded the metaphor of shell-as-chat, though visual divides between messages were preserved.

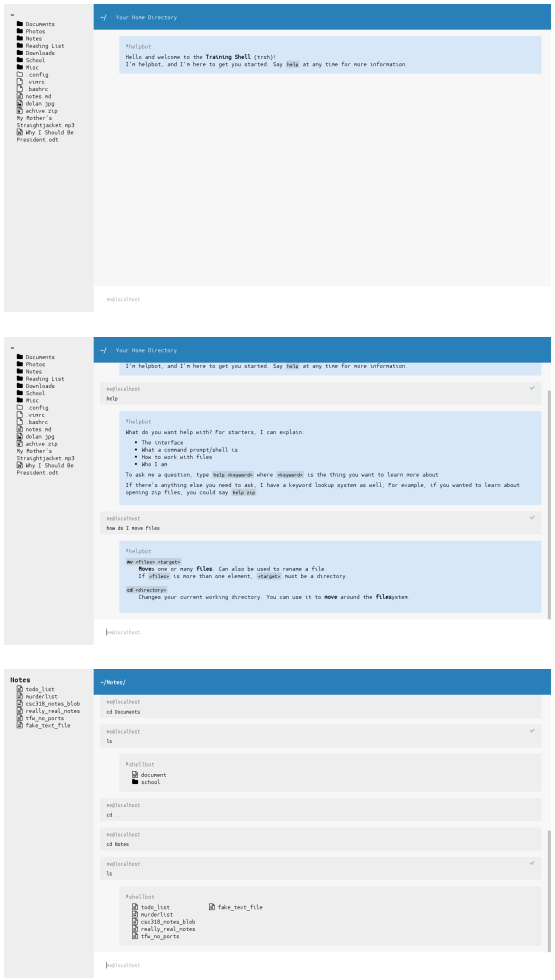

Figure 2: The Final Prototype

6. SUMMARY

Generally, users of command line interfaces like the power and flexibility they achieve when using one but find the learning experience to be lacking as the learning curve is extremely steep. Even intermediate users will often choose to use GUIs to accomplish tasks that can be done through a command line interface due to the more simplistic and intuitive nature of a graphical interface.

One of the biggest issues that we found was that users disliked remembering commands, especially during the learning period, and that some of the commands were seemingly irrelevant to the action being performed. Beginners and first time users felt uneasy due to the lack of visual cues, but recognized the usefulness that command lines offered to those that would use it on a regular basis. Overall, our focus on improving the learning curve allowed us to address other issues that other users brought up.

From what we discovered during in our extended research, some missing and potentially beneficial features that we haven't implemented in **Initial Design** section could be added, for instance:

- Processing of natural language help queries & more complete searching via tags & keywords.
- Command Auto-Completion, Auto-Correct, and auto-help as you type.
- Undo features (Perhaps alias rm to 'rm -i' or 'mv ~/Trash' instead of rm).
- File-globbing highlighting: visual indication of affected files for a given command prior to and after execution.
- Syntax highlighting and coloring as a default feature.
- Themes and user customization of the UI's layout and appearance

Moreover, further iteration based on more feature complete revisions including these features or a subset of them would allow additional refinement of our design.

7. REFERENCES

1. Durham, A., & Emurian, H. (1998). Learning and retention with a menu and a command line interface. *Computers in Human Behaviour*, 14(4), 597-620.

2. Westerman, S. (1997). Individual Differences in the Use of Command Line and Menu Computer Interfaces. *International Journal of Human-Computer Interaction*, 9(2), 183-183.

3. Herrera, D. A. (2004). Using guided interaction to support learning a command language (Order No. EP10787).

4. Davison, B. D., & Hirsh, H. (1997). Toward an adaptive command line interface. *HCI* (2), 505-508.

5. Davison, B. D., & Hirsh, H. (1998, July). Predicting sequences of user actions. *In Notes of the AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Analysis* (pp. 5-12).

6. Hirsh, H., Basu, C., & Davison, B. D. (2000). Learning to personalize. *Communications of the ACM, 43(8)*, 102-106.

7. Korvemaker, B., & Greiner, R. (2000, August). Predicting Unix command lines: adjusting to user patterns. *In AAAI/IAAI* (pp. 230-235).

8. Jacobs, N., & Blockeel, H. (2001). The learning shell: Automated macro construction. *In User Modeling 2001* (pp. 34-43). Springer Berlin Heidelberg.

9. Ekstrand, M., Li, W., Grossman, T., Matejka, J., & Fitzmaurice, G. (2011, October). Searching for software learning resources using application context. *In Proceedings of the 24th annual ACM symposium on User interface software and technology* (pp. 195-204). ACM.

10. R. Eberts , L. Villegas , C. Phillips & C. Eberts (1992) Using neural net modeling for user assistance in HCI tasks, International Journal of Human-Computer Interaction, 4:1, 59-77, *DOI*: 10.1080/10447319209526028

11. K. R. Mahach , D. BoehmâĂŘDavis & R. Holt (1995) The effects of mice and pullâĂŘdown menus versus commandâĂŘdriven interfaces on writing, International Journal of Human-Computer Interaction, 7:3, 213-234, *DOI*: 10.1080/10447319509526122

12. D. M. Lane , H. Albert Napier , S. Camille Peres & A. Sandor (2005) Hidden Costs of Graphical User Interfaces: Failure to Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts, *International Journal of Human-Computer Interaction, 18:2, 133-144, DOI*: 10.1207/s15327590ijhc1802_1

13. Morgan, K., Morris, R. L., & Gibbs, S. (1991). When does a mouse become a rat? orâĂę comparing performance and preferences in direct manipulation and command line environment. The Computer Journal, 34(3), 265-271.

14. Chen, J.-W., & Zhang, J. (2007). Comparing Text-based and Graphic User Interfaces for Novice and Expert Users. AMIA Annual Symposium Proceedings, 2007, 125-129.

15. Dillon, E. C., Jr. (2009). Which environment is more suitable for novice programmers: Editor / command line / console environment vs. integrated development environment?(Order No. 1468008). Available from ProQuest Dissertations & Theses Global. (304830577). Retrieved from http://search.proquest.com/docview/304830577