Bachelor Thesis

November 4, 2021

Comand Line Interfaces – Loved or Loathed ?

Liburn Gjonbalaj

of Waiblingen, Stuttgart, Germany (17-712-480)

supervised by

Prof. Dr. Harald C. Gall

Dr. Carol Alexandru

University of Zurich[UZH]

s.e.a.l.

software evolution & architecture lab

Bachelor Thesis

Comand Line Interfaces – Loved or Loathed ?

Liburn Gjonbalaj

University of Zurich[UZH]

s.e.a.l.
software evolution & architecture lab

Bachelor Thesis

Author: Liburn Gjonbalaj, liburn.gjonbalaj@uzh.ch

Project period: 20.05.2021 - 20.11.2021

Software Evolution & Architecture Lab
Department of Informatics, University of Zurich

Abstract

Graphical user interfaces (GUI) have surpassed command line interfaces (CLI) as the most widely used interface by software developers and are also recommended by the scientific literature as a less error prone, easier to use alternative to the CLI. However, studies show a certain percentage of software developers still choose to use the CLI on daily basis. The goal of our work is to investigate the reasons which lead these developers to choose the CLI over GUI, what are the difficulties developers are facing when learning the CLI and how to overcome these difficulties. We collected responses and opinions from 165 software developers with the help of an online survey and the experiences and thoughts of software developers from 11 one-to-one interviews. Most of our respondents are CLI users with over 10 years of experience using the CLI. Our results show that CLI aspects like automation, scripting, flexibility, parallelizing work are all areas in which CLI is superior to GUI. With man pages/documentation, discoverability, remembering commands being the biggest difficulties learning programmers face, we also give recommandations such as online courses, cheatsheets, newer help pages called tldr and alternative shells such as the fish shell to overcome these hurdles and shorten the longer learning curve that comes with using the CLI.

Zusammenfassung

Grafische Benutzeroberflächen (englisch: Graphical User Interface, kurz GUI) haben Kommandozeilen (englisch: Command Line Interface, kurz CLI) als das von Softwareentwickler:innen meistgenutzte Interface übertroffen und werden auch von der wissenschaftlichen Literatur als eine weniger fehleranfällige und einfacher zu bedienende Alternative zum CLI empfohlen. Studien jedoch zeigen, dass sich ein Prozentsatz von Softwareentwickler:innen immer noch dafür entscheidet, das CLI täglich zu verwenden. Das Ziel unserer Arbeit ist es, die Gründe zu untersuchen, warum diese Entwickler:innen das CLI dem GUI vorziehen, welche Schwierigkeiten Entwickler:innen beim Erlernen des CLI haben und wie diese Schwierigkeiten überwunden werden können. Wir haben mithilfe einer Online-Umfrage die Antworten und Meinungen von 165 Softwareentwickler:innen und durch 11 Einzelinterviews die Erfahrungen und Gedanken von Softwareentwickler:innen gesammelt. Die meisten unserer Befragten sind CLI-Benutzer mit über 10 Jahren Erfahrung in der Nutzung des CLI. Unsere Ergebnisse zeigen, dass CLI-Aspekte wie Automatisierung, Skripting, Flexibilität, Parallelisierung der Arbeit Bereiche sind, in denen das CLI dem GUI überlegen ist. Manpages / Dokumentationen, Auffindbarkeit und das Merken von Befehlen werden von lernenden Programmierer:innen als grösste Schwierigkei-ten empfunden, wir empfehlen daher auch Online-Kurse, Spickzettel (Cheatsheets), neuere Hilfeseiten namens tldr und alternative Shells wie die Fish-Shell, um diese Hürden zu überwinden und die längere Lernkurve zu verkürzen, die mit der Verwendung des CLI kommt.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The first Unix shell, the V6 shell was developed by Ken Thompson in 1971 at Bell Labs and was modeled after Schroeder's Multics shell. The Bourne shell was introduced in 1977 as a replacement for the V6 shell. Although it is used as an interactive command interpreter, it was also intended as a scripting language and contains most of the features that are commonly considered to produce structured programs. The Bourne shell led to the development of the Korn shell (ksh), Almquist shell (ash) and the popular Bourne-again shell (or Bash). Early microcomputers themselves were based on a command-line interface such as CP/M, DOS or AppleSoft BASIC. During the 1980s and 1990s, the introduction of the Apple Macintosh and of Microsoft Windows on PCs saw the command line interface as the primary user interface replaced by the Graphical User Interface. The command line remained available as an alternative user interface, often used by system administrators and other advanced users for system administration, computer programming and batch processing. [1]

Since then little research has been done on the perception of the Command Line Interface in comparison to other interfaces. The research that has been done generally favors the use of Graphical User Interfaces over CLIs in terms of usability and also preference among engineers. In 1991, a study was conducted comparing WIMP (Windows Icons Mice and Pull down menus) with command line interfaces, where WIMP was shown to be superior to command line interface. [2] More recently, in [3] when comparing the use of CLIs and GUIs over the years, the authors say that GUIs largely supplanted CLIs in the past in terms of user preference and now a significant minority of developers prefer to use CLIs. Their results also show significant advantage to using GUIs in terms of effectiveness. In [4] the authors showed similar tendencies in terms of user preference but also gave some reasons why CLI gets chosen for use from developers. Finally, in [5] software developers are advocated to take advantages of GUI to the most possible extent as an alternative to CLI.

On the other hand, some research has also been done that encourages the use of CLIs. In [4] about thirty percent of participants who used CLIs report that CLIs are flexible, efficient, transparent, reliable, and achieve ultimate functionality and a good performance. In [6] the authors found that the command line interface allows network security engineers in the intrusion detection task to better control the analysis of details of the data through the use of rich, powerful, and flexible commands. Also in [7] the author has given 5 reasons why people should use CLIs, those being: Wrangle files, Handle big data, Manipulate spreadsheets, Parallelize your work, Automate. Throughout the cited articles it can be found that one of the main drawbacks of CLIs is the long learning curve and that once you get to expert level, you start appreciating and using CLIs more.

Seeing how GUIs is much more advocated to be used, is generally more preferred and used than CLIs, we ask ourselves what is the reason that software developers today still use it and how could CLI education and integration into workflow be improved ?

With these goals in mind we will try to answer the research questions:

- RQ 1a: What causes Software Developers to stick to CLI even if GUI applications are available for the same task?
- RQ 1b: Are there tasks for CLI which cannot reasonably be replaced by GUI?
- RQ 2: What problems are software developers facing when learning CLI?
- RQ 3: Which tools and/or techniques could alleviate problems faced when learning CLI?

Our approach to answering these research questions is to conduct a survey followed by interviews. In order to get maximal information from the survey questions, each question will inspired by some previous result that can be found in the related work section and will be aimed at answering one or multiple of the research questions. After the survey will be closed, the answers for each question will be carefully analyzed, particular emphasis will be put into how the answers contributed to the research questions and what needs to be further looked into during the interviews. The analysis is done through tagging and finding similar patterns and themes in the answers and by relating the results of the likert scale questions to our research questions.

After the survey, our purpose with the interviews is to expand and gain more insight on the answers that we had found from the survey. The analysis is done by separating the questions of the interviews into separate topics and afterwards finding similar patterns and themes in the answers.

The remainder of this paper presents the related work to our study in chapter 2, describes the methodology, analysis and results of our survey in chapter 3 and describes the methodology, analysis and results of our interviews in chapter 4. Finally, a discussion of the recommandations, threats and limitations, future work and conclusions is presented in chapter 5. A complete overview of the survey questions can be found on Appendix A - Survey Questions.

Chapter 2

Related work

2.1 Method of related work research

The research on the perception of the command line interface is very scarce. To my knowledge there is no work which is considered as most famous, central work or similarly. The textbook "Linux® Command Line and Shell Scripting Bible" from Richard Blum had been used to better understand the Linux Command Line Interface. The digital libraries ACM, IEEE Computer Society, INFORMS, JSTOR, CiteCeer and Hauptbibliothek Zürich have been thoroughly searched using the keywords "Command line", "Command Line Interface", "CLI", "GUI", "perception", "human computer interaction", "learning experience", "learning", "advantages" and almost all combinations of these keywords. What resulted was a collection of 12 articles, which i thought represented and gave insight into the purpose of the thesis and another two articles "S.J. Westerman (1997) Individual Differences in the Use of Command Line and Menu Computer Interfaces, International Journal of Human-Computer Interaction, 9:2, 183-198" [8] and "Learning and Retention with a Menu and a Command Line Interface" by Ashley G. Durham and Henry H. Emurian [9] that are not included in the related work chapter, since we thought they do not contribute to the discussion that we are going to have.

2.2 Early research and views on the correlation between interface style and its perception to the user

Research into how the command line interface is perceived from the users and how the users perform in comparison to other interfaces has been conducted since the early 80s. [10] Here a **comparison between a form-filling interface and an identical command line** found the form filling interface to be faster, and noted that 11 of the 12 subjects tested preferred the form filling method.

Effects of interface style on user perceptions and behavioral intention to use computer system. [11] In this paper, the authors examine two types of interfaces: menu- and command-based interfaces and compare the effects of a menu-based with a command-based interface style on user perceptions of a new system. In this study the system interface style is treated as an external factor in the technology acceptance model to examine its direct and indirect effects on behavioral intention to accept and use a system. The results demonstrated that interface style had a sig-

nificant direct effect on perceptions of ease of use, with menu based interface style perceived as being easier to use than a command-based interface style. Perceived usefulness was found to be significantly influenced by interface style, with menu-based interface perceived as being more useful than command-based interface. Contrary to expectations, the results showed that interface style had a nonsignificant direct effect on behavioral intention. However, given that the results demonstrated that interface style had a significant effect on perceptions of ease of use and usefulness, it appears that interface affects behavioral intention indirectly through its direct effects on perceptions of ease of use and usefulness.

The results also showed that the impact of perceived ease of use on behavioral intention was slightly stronger than that of perceived usefulness. The authors suggest that when users do not have prior experience with a system, they are more concerned about their ability to use the system; once they learn the new system and are able to use it, they start to become more concerned about its usefulness.

2.3 Research favoring GUI use

In the 90s research was made that compared the perception of more modern interfaces, called WIMP (Windows Icons Mice and Pull down menus), with the perception of command line interface. This study involved students using a database program specially designed and developed to include the two interfaces, where the two interfaces were matched for functionality and as far as possible error opportunity. What was seen from the experiments was that the WIMP interface is faster, easier, less error prone and more stimulating for naive touch typists than a functionally identical command line interface. [2]

In the famous work of Human-Computer interaction **"Designing the user interface"** in the section "Command-line versus display editor versus word processors", the authors argue that the performance was improved and training times were reduced with display editors and they quote display editor enthusiasts saying "Once you have used a display editor, you will never go back to a line editor-you'll be spoiled.". But in this case they also enumerate some advantages of com-mand line approaches such as: easier history keeping, more flexible markup languages, macros tend to be more powerful, and some tasks are simpler to express (such as, changing all italics to bold). [12]

Since then research has focused on studying the performance and perception of command line interfaces in comparison with other environments with the purpose of studying them in different systems and in different professions.

A few of these studies are:

A command line interface versus a graphical user interface in coding VR systems [3]. In this study five programmers were asked to generate a scene, which contains a virtual 3D environment. The time spent for task completion with the GUI was shorter than the one with the CLI, the percentage of the task completed was also higher with GUI, time spent with errors was much higher with the CLI, the number of repetitions or failed commands was also higher with the CLI, as well as the time spent with consulting support personnel, and also the time spent with consulting documentation. The paper concludes by saying that the experiments of comparisons between the CLI and GUI have demonstrated significant advantages in using a GUI. Through the self-explanatory structure and layout of a GUI it is possible to support the performance of a novice programmer. Coding with the CLI requires background knowledge and practice. The GUI may provide a number of initial settings and ease the visualization of complex concepts giving some hints about the settings. Thus a programmer can focus on the task without thinking about the steps to follow. An expert programmer has the specific programming/scripting skills

for the graphic libraries and may not need a GUI as much as a novice. On the contrary, a novice programmer may require a GUI to cope with the demands of the task.

Usability of User Interface Styles for Learning Graphical Software Applications [5]. The goal of this paper, among other priorities, was to examine usability of different user interface styles (i.e. GUI and CLI) for learning a graphical software application. In this study a usability testing was conducted to evaluate the usability attributes of comparing different user interface design (i.e. GUI and CLI) of learning graphical software applications using Adobe Flash CS4 and Microsoft Expression Blend 4. CLIs were found to be more difficult to learn and less easy to use, even for software developers as well as designers. GUI was perceived to be simpler to learn for both groups. The authors conclude that software developers are advocated to take advantages of GUI to the most possible extent as an alternative to CLI. In a situation that CLI cannot be evaded; employing CLI with suggestions is far more usable rather than CLI per se.

2.4 Research analysing the strengths and weaknesses of both CLI and GUI

System Administrators Prefer Command Line Interfaces, Don't They? An Exploratory Study of Firewall Interfaces [4], in this paper an online study was made on the preferences of system administrators regarding firewall interfaces, with 300 volunteer participants. The results show that only 32 percent of the respondents prefer CLIs for managing firewalls, while the corresponding figure is 60 percent for GUIs. In this paper, the authors suggest that there may be a connection between a system administrator's proficiency with firewalls and the interface that they prefer to utilize. Results show that the stronger the firewall expertise of respondents, the lower the likelihood of utilizing GUIs. Seventy percent of the system administrators with a basic knowledge of firewalls prefer GUIs to any other interface, while this holds true for only 54 percent of firewall experts. The authors classify strengths and limitations of firewall CLIs and GUIs. The participants reported that CLIs are flexible, efficient, transparent, reliable, and achieve ultimate functionality and a good performance. However, CLIs are inconvenient for representing data, do not help users by preventing errors, and have a long learning curve. On the other hand, GUIs help users to perceive firewall configuration information more effectively and have a shorter learning curve compared to CLIs. They are also easy to use, easy to create and modify rules with and good for occasional use. The authors also finally give some design recommandations that should be taken into account by designers aiming to develop better CLIs and GUIs.

Command Line or Pretty Lines? Comparing Textual and Visual Interfaces for Intrusion Detection. [6] The authors conducted a controlled experiment comparing a representative textual and visual interface for ID to develop a deeper understanding about the relative strengths and weaknesses of each. With this understanding, the authors recommended designing a hybrid interface that combines the strengths of textual and visual interfaces for the next generation of tools used for intrusion detection. Some of the strengths of the textual interface found were: allowing direct access to fine-grain details which is critical to the task of ID, giving users the ability to control and costumizing the view of the data through use of powerful filtering commands, among the weaknesses were: identifying patterns can be difficult which can hinder the success of the task of ID, regarding this one user stated: "if all values change a lot, [it is] difficult to see [a] pattern" and another weakness was the burdening of the user with recalling the syntax of the command. The authors suggest that a combined interface of a visual and text interface could create an environment that is suitable for a variety of user expertise. Specifically, novices could be trained to use the textual interface by seeing the underlying text-based commands used to filter anomalous patterns in the visual interface. At the same time the users benefit from seeing the detailed data,

thereby gaining knowledge and expertise.

2.5 Research highlighting CLI advantages

A study was conducted more recently that **compared the mental models that novice programmers created when using visual and when using command line environments**. The authors found that the difference in feature sets between these environments can influence the learning curve. Visual environments like IDLE may impose a lower learning curve because of their features. On the other hand, command line environments like VIM may require more time for novices to learn. Despite the possibility of a lower learning curve, certain features in visual environments can potentially prevent novices from developing an appropriate mental model for programming. As seen in this study, some of the subjects who struggled with VIM were attempting to find features that they were accustomed to in IDLE. When failing to find such features, they would show confusion or frustration, which would eventually prevent them from completing the assignment. In contrast, subjects who originally used VIM were not faced with this problem. [13]

On the other side an op-ed piece which supports the use of the command line is **"Five reasons to love the command line." [7]** Here the author suggests five different ways that the command line can ease computational research, those being: wrangling files, handling big data, manipulate spreadsheets, parallelize your work, automation. Although he also writes a downside of the command line, that there is no undo in it.

2.6 Alternative Solutions

2.6.1 Overview

The first Unix shell, the V6 shell, was developed by Ken Thompson in 1971 at Bell Labs and was modeled after Schroeder's Multics shell. The Bourne shell [1] was introduced in 1977 as a replacement for the V6 shell. Although it is used as an interactive command interpreter, it was also intended as a scripting language and contains most of the features that are commonly considered to produce structured programs. The Bourne shell led to the development of the Korn shell (ksh)[2], Almquist shell (ash)[3], and the popular Bourne-again shell (or Bash) [4].

Early microcomputers themselves were based on a command-line interface such as CP/M [5], DOS [6] or AppleSoft BASIC [7]. During the 1980s and 1990s, the introduction of the Apple Macintosh and of Microsoft Windows on PCs saw the command line interface as the primary user interface replaced by the Graphical User Interface. The command line remained available as an alternative user interface, often used by system administrators and other advanced users for system administration, computer programming and batch processing.

In November 2006, Microsoft released version 1.0 of Windows PowerShell (formerly codenamed Monad), which combined features of traditional Unix shells with their proprietary object-oriented .NET Framework. MinGW and Cygwin are open-source packages for Windows that offer a Unix-like CLI. Microsoft provides MKS Inc.'s ksh implementation MKS Korn shell for Windows through their Services for UNIX add-on.

[1] https://en.wikipedia.org/wiki/Bourne_shell
[2] http://www.kornshell.org/
[3] https://www.in-ulm.de/~mascheck/various/ash/
[4] https://www.gnu.org/software/bash/
[5] http://www.digitalresearch.biz/CPM.HTM
[6] https://en.wikipedia.org/wiki/DOS
[7] https://en.wikipedia.org/wiki/Applesoft_BASIC

Since 2001, the Macintosh operating system macOS has been based on a Unix-like operating system called Darwin [8]. On these computers, users can access a Unix-like command-line interface by running the terminal emulator program called Terminal, which is found in the Utilities subfolder of the Applications folder, or by remotely logging into the machine using ssh. Z shell [9] is the default shell for macOS; bash, tcsh[10], and the Korn shell are also provided. Before macOS Catalina, bash was the default. [1]

2.6.2 Shells

The shell. **The GNU/Linux shell** is a special interactive utility. It provides a way for users to start programs, manage files on the filesystem, and manage processes running on the Linux system. The core of the shell is the command prompt. The command prompt is the interactive part of the shell. It allows you to enter text commands, interprets the commands, then executes the commands in the kernel. The shell contains a set of internal commands that you use to control things such as copying files, moving files, renaming files, displaying the programs currently running on the system, and stopping programs running on the system. Besides the internal commands, the shell also allows you to enter the name of a program at the command prompt. The shell passes the program name off to the kernel to start it. There are quite a few Linux shells available to use on a Linux system. Different shells have different characteristics, some being more useful for creating scripts and some being more useful for managing processes. The default shell used in all Linux distributions is the bash shell. The bash shell was developed by the GNU project as a replacement for the standard Unix shell, called the Bourne shell (after its creator). The bash shell name is a play on this wording, referred to as the "Bourne again shell". [14]

Linux Shells

- ash - A simple, lightweight shell that runs in low-memory environments but has full compatibility with the bash shell.
- korn - A programming shell compatible with the Bourne shell but supporting advanced programming features like associative arrays and floating-point arithmetic.
- tcsh - A shell that incorporates elements from the C programming language into shell scripts
- zsh - An advanced shell that incorporates features from bash, tcsh, and korn, providing advanced programming features, shared history files, and themed prompts.

Bash **the Bourne-Again Shell**, refers both to a particular Unix shell program and its associated scripting language. It is the default shell of the GNU Operating System (Linux) and Apple's OS X and is POSIX 1003.2 compliant.

It is a powerful shell, and features, among other things: command line editing, command history, a directory stack (pushd, popd), command substitution, special variables like PPID, autocompletion, in-process integer arithmetic: ((...)), in-process regexes, aliases, functions, arrays, expansions: tilde, brace, variable, substring awesomeness, conditional expressions, security (restricted shell mode), job control, timing, prompt customization. [15]

[8] https://en.wikipedia.org/wiki/Darwin_(operating_system)
[9] https://www.zsh.org/
[10] https://www.tcsh.org/

The Z shell (called **zsh**) is an open source Unix shell developed by Paul Falstad. It takes ideas from the Bourne, bash, ash, and tcsh shells and adds many unique features to create a full-blown advanced shell designed for programmers. Some of the features that make the zsh shell unique are: improved shell option handling, shell compatibility modes, loadable modules etc.

The zsh shell provides a core set of built-in commands, plus the ability to add additional command modules. Each command module provides a set of additional built-in commands for specific circumstances, such as network support and advanced math functions. You can add only the modules you think you need for your specific situation.

Most shells use command line parameters to define the behavior of the shell. The zsh shell uses a few command line parameters to define the operation of the shell, but mostly it uses options to customize the behavior of the shell. You can set shell options either on the command line, or within the shell itself using the set command. [14]

PowerShell is a cross-platform task automation solution made up of a command-line shell, a scripting language, and a configuration management framework. PowerShell runs on Windows, Linux, and macOS. PowerShell is a modern command shell that includes the best features of other popular shells. Unlike most shells that only accept and return text, PowerShell accepts and returns .NET objects. The shell includes the following features: robust command-line history, tab completion and command prediction, supports command and parameter aliases, pipeline for chaining commands, in-console help system similar to Unix man pages.

As a scripting language, PowerShell is commonly used for automating the management of systems. It is also used to build, test, and deploy solutions, often in CI/CD environments. PowerShell is built on the .NET Common Language Runtime (CLR). All inputs and outputs are .NET objects. No need to parse text output to extract information from output.

The PowerShell scripting language includes the following features: extensible through functions, classes, scripts, and modules; extensible formatting system for easy output, extensible type system for creating dynamic types, built-in support for common data formats like CSV, JSON and XML. [16]

Fish, the friendly interactive shell offers a command-line interface focused on usability and interactive use. Unlike other shells, fish does not follow the POSIX standard, but still uses roughly the same model. Some of the special features of fish are:

- No configuration needed: fish is designed to be ready to use immediately, without requiring extensive configuration.
- Easy scripting: New functions can be added on the fly. The syntax is easy to learn and use.
- Fish has an extensive help system. Use the help command to obtain help on a specific subject or command. For instance, writing help syntax displays the syntax section of this documentation. Fish also has man pages for its commands, and translates the help pages to man pages. For example, *man set* will show the documentation for *set* as a man page.
- Extensive UI (see bulletpoints below)
- Autosuggestions. fish suggests commands as you type, based on command history, completions, and valid file paths. Autosuggestions are a powerful way to quickly summon frequently entered commands,by typing the first few characters. They are also an efficient technique for navigating through directory hierarchies.
- Syntax highlighting - Fish interprets the command line as it is typed and uses syntax highlighting to provide feedback. The most important feedback is the detection of potential

errors. By default, errors are marked red. Detected errors include: non existing commands, reading from or appending to a non existing file, incorrect use of output redirects, mismatched parenthesis.

- Tab completion is a time saving feature of any modern shell. When you type Tab, fish tries to guess the rest of the word under the cursor. If it finds just one possibility, it inserts it. If it finds more, it inserts the longest unambiguous part and then opens a menu (the "pager") that you can navigate to find what you are looking for.
- The fish editor features copy and paste, a searchable history and many editor functions that can be bound to special keyboard shortcuts.

[17]

tldr pages are a community effort to simplify the man pages with practical examples. An example of the tldr can be seen in figure 2.1, while an example of the man page can be seen in figure 2.2. [18]

Figure 2.1: tldr example

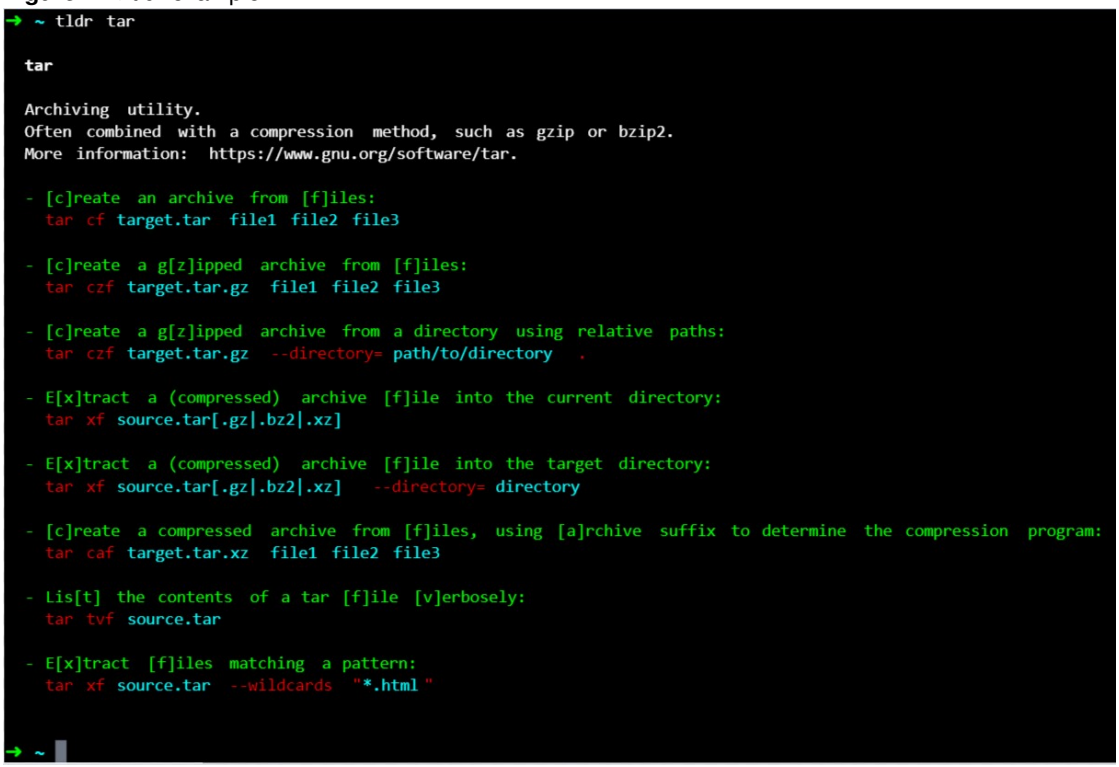
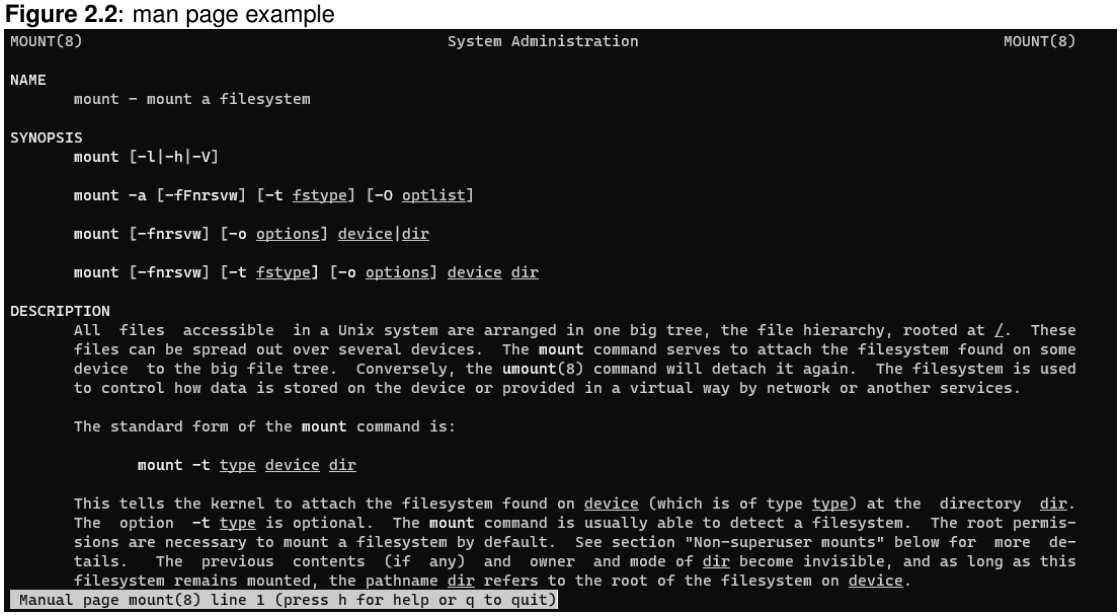


Figure 2.2: man page example



```
MOUNT(8)                                System Administration                                MOUNT(8)

NAME
       mount - mount a filesystem

SYNOPSIS
       mount [-l|-h|-V]

       mount -a [-fFnrsvw] [-t fstype] [-O optlist]

       mount [-fnrsvw] [-o options] device|dir

       mount [-fnrsvw] [-t fstype] [-o options] device dir

DESCRIPTION
       All  files  accessible  in a Unix system are arranged in one big tree, the file hierarchy, rooted at /.  These
       files can be spread out over several devices.  The mount command serves to attach the filesystem found on some
       device  to the big file tree.  Conversely, the umount(8) command will detach it again.  The filesystem is used
       to control how data is stored on the device or provided in a virtual way by network or another services.

       The standard form of the mount command is:

              mount -t type device dir

       This tells the kernel to attach the filesystem found on device (which is of type type) at the  directory  dir.
       The  option  -t type is optional.  The mount command is usually able to detect a filesystem.  The root permis-
       sions are necessary to mount a filesystem by default.  See section "Non-superuser mounts" below for  more  de-
       tails.   The  previous  contents  (if  any)  and  owner  and mode of dir become invisible, and as long as this
       filesystem remains mounted, the pathname dir refers to the root of the filesystem on device.
 Manual page mount(8) line 1 (press h for help or q to quit)
```

Chapter 3

Survey

3.1 Introduction

After doing research in the use and perception of the command line and identifying the research questions, we chose to conduct a survey followed by interviews to give answers to them. In order to get maximal information from the survey questions, each question was inspired by some previous result that can be found in the related research section and was aimed at answering one or multiple of the research questions. Our survey was a success as it had reached over 100 responses in 2 days and 165 total responses for the total of less than six days before it was expired. After the survey was closed for access, the answers for each question were carefully analyzed, how they contributed to the research questions and what needed to be further looked into during the interviews was particularly emphasized in the analysis. A complete overview of the survey questions can be found on Appendix A - Survey Questions.

3.2 Method

3.2.1 Overview

A web-based questionnaire was created on 28.07.2021 with 22 questions and shared over to multiple pages in Reddit in particular in the reddit communities: r/commandline, r/bash, r/hci and r/zsh, it was also shared in LinkedIn and also personally sent to multiple acquainted software engineers, as well as also sent out to a company in Zurich, Polygon Software, that develops code for the digitalization of companies, in order for the employees to fill it out if they want. People had access to fill the survey out until 02.08.2021. The responses to the survey were anonymized and the survey was presented question by question. The web-based questionnaire app LimeSurvey was used, thus also the data collection was made easier since it was done automatically from the app. A total of 165 respondents filled out the survey, among which, 113 filled the survey out completely and 52 partially, as can be seen from table 3.1.

The sequence of questions in the survey was such that in the beginning there are descriptive questions that gave us information about the respondents level of acquaintance with the command line, her/his overall perception of the CLI and how they learned the command line as a programmer. The questions 1 to 7 and 22 are such questions, the following questions, i.e after question 7, are all substantive questions which address the research questions. A mix of hybrid questions, closed-ended and open-ended questions was used.

Surveys filled		
Partial	Full	Total
52	113	165

Table 3.1: Survey completion

3.2.2 Survey questions

Introductory Questions

Q01 - "How many years of experience do you have in programming ?"
Q02 - "Which is your primary operating system?"
Q03 - "How would you rate your familiarity with CLI?"
Q04 - "Which kinds of CLI are you comfortable with?"
Q05 - "Which of the following best describes your CLI learning experience?"
Q06 - "Did you switch from GUI to CLI or vice-versa during your programming career ?"
Q07 - "I started learning CLI from..."
Q22 - "How did you find this survey ?"

Hybrid questions were mainly used for descriptive questions that informed us about the background that the respondent has with the command line but also her/his experience with the command line. These questions were multiple choice with an "Other" option to add more to the answer. Questions about the background were those that asked the respondents years of experience in programming, which operating system and which shell she/he uses, her/his level of familiarity with the command line, more important for our survey were questions that informed us about the respondents experience with the CLI, which were whether the respondent had initially spent a lot of effort learning the CLI or did she/he quickly become comfortable with it, did they switch from GUI to CLI or vice-versa or did they always use the same interface and from what resource they learned the CLI.

The question whether the developers initially spent a lot of effort or they quickly became comfortable with it or it was overall an enjoyable or frustrating experience was inspired from research ([13], [4]) which suggested that using the command line requires a long learning curve prior to being able to using it effectively whereas the GUI's learning curve is relatively short in comparison to that of the CLI's. We could thus see from this questions whether this learning curve was really so for our respondents and dependent on the responses from the other questions we could see what lead to the curve being short or long. The question of whether the respondents had switched from GUI to the CLI or vice-versa during their career was also inspired from research [3] which said that for beginning developers using GUI is more suitable as the CLI requires more expertise and it would also let young developers focus more on the task at hand rather then focusing on the CLI. We wanted to see if this was also the case for the respondents but also try to draw relationships from this questions responses and responses from other questions, such as: if there was a switch from GUI to CLI or vice-versa, what caused the switch ?

Q07 - "I started learning CLI from..."

Question 7 was one of the key questions of the survey as we wanted to find out what are the main sources that developers are learning/learned the CLI from and also what would follow from relating this with other questions was whether these methods are effective and if yes what makes them so and if not, what could have been done to improve the learning curved.

Substantive Questions

Recent experiments have tested the performance of CLIs against that of GUIs in different settings and for different purposes such as in firewall security tasks [4] or in cli vs gui in coding vr systems [3] where the results of the experiments have largely favored GUI, especially in [3] where GUI topped CLI in terms of performance in every category. Also [7] lists advantages that CLI has over GUI in computational research. The purpose of the closed-ended questions was to answer the research questions. Each closed-ended question was taken as a result from an already published article and tested whether it was valid for our case and for our research purposes, i.e why developers do not select GUI applications instead of CLI, are there tasks for the CLI which cannot reasonably replaced by GUI, what problems developers face when learning GUI and how to alleviate these problems.

Q08 - "While learning CLI, i felt that CLI more closely resembled my mental programming model than GUI, so I felt more inclined to use it"

Question 8 was inspired from the paper [13] which compared the different mental models that novices can acquire from learning visual and command line environments. A study which concluded that command line environments can potentially allow novices to develop a more helpful mental model for programming than visual environments. What we wanted to test through this question was whether the command line felt more like programming for developers and whether that is a reason that they use it. This question would prove helpful in the interviews where more information about the command line was extracted based on this information.

Q09 - "While learning CLI, i felt that I can achieve more in less time using the CLI"

Question 9 was inspired from the paper [3] where a group of five programmers was asked to first generate a 3D environment with the command line and then the participants were expected to generate a similar 3d environment with the help of a GUI and what resulted was that the average for task completion with the GUI was 13.9 minutes versus 62 minutes with the CLI. We wanted to see if programmers that use the CLI believe they can finish tasks quicker using it rather than using GUI.

Q10 - "While learning CLI, i felt that the CLI offers me freedom to express anything that I want, while I felt GUI limited me in this aspect."

Question 10 was inspired from the paper [4] where the number 1 strength of CLI was flexibility according to the respondents.

Q11 - "While learning CLI, i felt that GUI's speed of operation was much slower than CLI's, and it bothered me."

Question 11 has been also inspired from paper [4] where the qualitative data showed that GUI's are not very useful for experts one of the reasons being the low operation speed.

Q12 - "While learning CLI, i felt that the number of times I made mistakes in CLI bothered me."

Q13 - "While learning CLI, i felt that the number of times I had to consult documentation while using CLI bothered me."

Q14 - "While learning CLI, i felt that having to remember CLI commands by heart bothered me."

Questions 12, 13 and 14 have all been inspired from the paper [3] where all of these are aspects where developers in the study performed better with the GUI than the CLI. Other than this study, being error prone and relying on remembering commands are also weaknesses of the CLI that have been reported by respondents in the paper [4] and also in [5] with the context switched in our case to that when developers were still learning the command line in order to be in alignment with our research goals.

Q16 - "In data handling tasks such as handling big data, manipulate spreadsheets, paral-

lelize your work, automation, what do you prefer to use more ?"

Question 16 had directly been taken from the opinion that Jeffrey M. Perkel expressed in [7] over the reasons CLI is better than GUI for computational research, we wanted to test this opinion in a more general context.

Q17 - "In solving repeated tasks, which do you prefer to use more ?"

Question 17 was taken from [3] where it is said that repetitive tasks can be easily achieved with loops, which would take hours in a GUI environment.

Q18 - "To build small case-specific tools for later re-use I prefer..."

Question 18 was taken from [4] where one of the respondents said that on of the strengths of the CLI is bash scripting. In an attempt to answer the research question 1b, we included it in this survey.

Open-ended questions were used with the same purpose that closed-ended questions were used but here the respondents had the possibility to give further and more detailed insight into the CLI working and learning experience.

Q15 - "Other aspects where i had difficulties while learning CLI are..."

On Question 15 the respondents could also cover what was not mentioned in the previous questions about the difficulties of learning the CLI or they could even add specifics that had to do with one of the already mentioned suggestions.

Q19 - "I don't think GUI can replace CLI in..."

Also question 19 was very similar to question 15 in that also here the respondents could add something that was not already mentioned in the previous questions regarding the aspects of the CLI which could not reasonably be replaced by GUI or confirm one of the already proposed suggestions.

Q20 - "Would you like that the CLI had an integrated GUI like assistant while you were learning or using it ? Examples: Command suggestion, faulty command correction, pop-ups with documentation or running examples, etc."

Question 20 is a hybrid question where we tried to find out whether graphical tools would have an effect on learning the command line. If the responses for this question would be positive, we would then be able to further investigate on this question.

Q21 - "If you were to learn CLI again for the first time today, what would you do different ?"

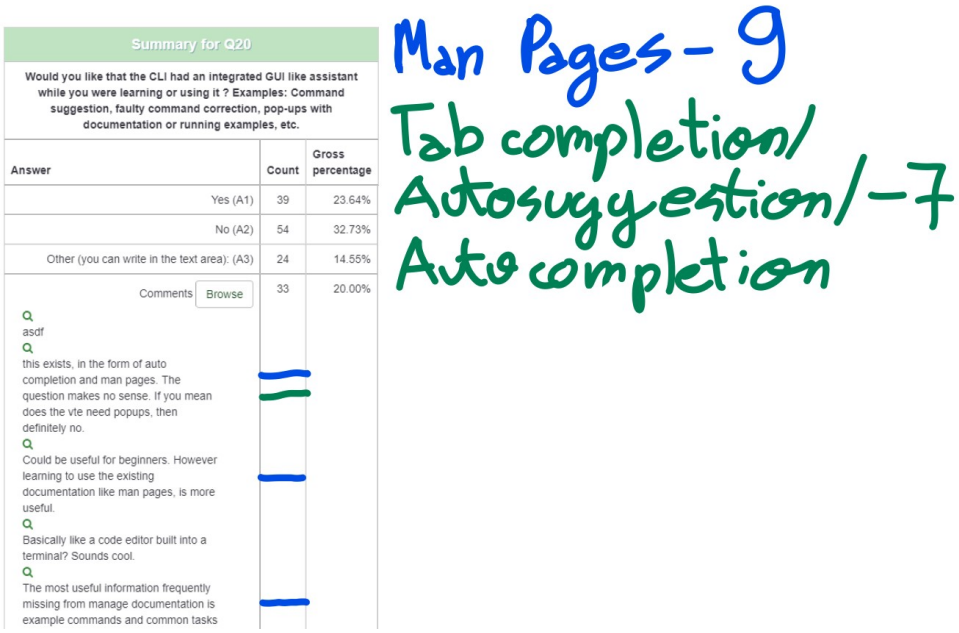
Question 21 is also one of the key questions of the survey as it tries to identify the mistakes while learning the CLI but also what are the best methods for learning it.

3.2.3 Analysis

Closed-ended questions were all Likert scale, strongly disagree/mildly disagree/neutral/mildly agree/strongly disagree. In these questions the distribution of the answers was analyzed relative to prior assumptions we had, which were based on existing research and the research goals. If a large majority strongly agreed upon in a question, then we took that as a true statement. Yet, as the questions were Likert scale, there remained additional insight to be found into why the respondents think how they do. If, in other cases, in the answers there were divergences from prior assumptions we had from research, we prepared to further investigate about these divergences in the interviews phase, which would follow the survey analysis one. The open-ended questions, question 20 ("Yes", "No" and "Other" hybrid question) and question 7 were all analyzed in the same manner. First, all comments were carefully read and a list of keywords (or most commonly used terms) was made. Each keyword was associated to a color and near each comment where

that keyword appeared a mark with that color was made so that we could enumerate all appearances of that keyword in the answers. Similarly to figure 3.1.

Figure 3.1: Analysis example



The purpose was to notice patterns and themes in the answers as much as possible. There were cases where a particular keyword was implicitly stated, we also took those appearances into account. We carefully read each instance of each keyword and looked for why and how a particular keyword was used in the answers in order to answer our research questions.

3.3 Results

3.3.1 Introductory questions

Reddit was the platform where we got the most responses from, over 150 people entered and filled our the survey through it. Luckily for us, a majority of the respondents of the survey had more than 10 years of experience in programming and considered themselves expert CLI power-users as can be seen from figure 3.2.

A large majority of them also had Linux as their primary operating system and used the linux shells (bash, sh, zsh etc.) when interacting with the command line. Another interesting information that we found out was that the respondents did not struggle learning the command line. Most of them found it an enjoyable experience and they said that they quickly became comfortable with it. Another interesting fact is that a lot of them also responded that even though they still use it regularly they still have to look up commands and syntax on a regular basis. As can be seen from figure 3.3.

It is interesting since literature reports that there is a steep learning curve to the CLI ([13], [4]), which was not the case for our respondents and the immediate question from this result was,

Figure 3.2: Respondent experience with the CLI

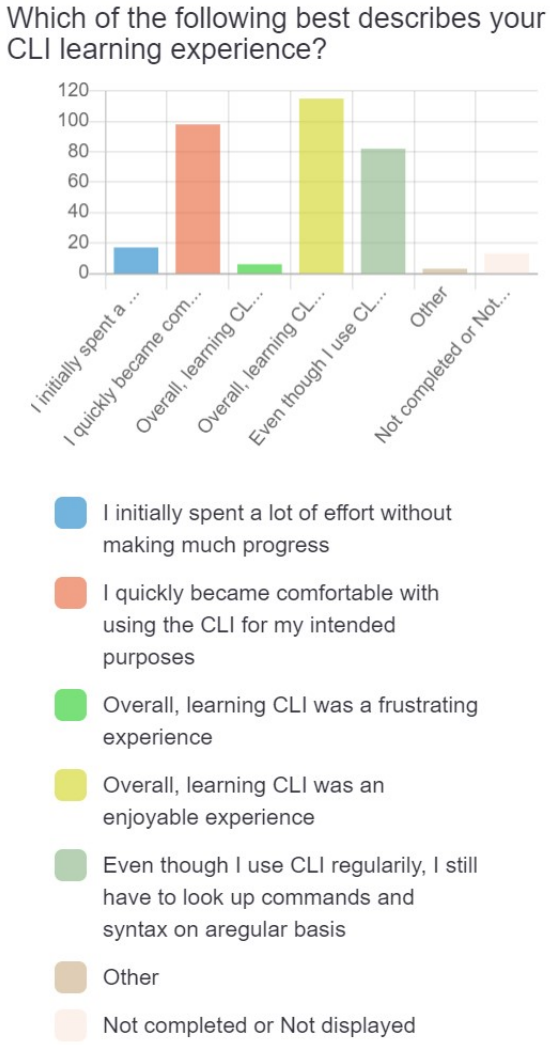
what were the reasons that these software engineers did not have a steep learning curve with the command line. What resulted from these introductory questions was also that we knew we could get concrete answers to our research questions based on the long experience and expertise of the respondents with the command line.

In **question 7**, "I started learning CLI from...", as can be seen from figure 3.4, a majority, 97 out of 267 given answers (the question was multiple choice with an "Other" option) answered that they had learned the CLI from written tutorials in the internet, Youtube tutorials and cheat sheets were selected 32 and 49 times respectively, the answers in the "other" option were as seen in table 3.2.

Keyword	# of times mentioned
man pages	29
books	15
colleagues/friends/mentor	11
learn by doing/experimenting/figuring out on my own	10
university/college/school	6
documentation	3

Table 3.2: Q7, option "Other" most mentioned keywords

Figure 3.3: Question 5

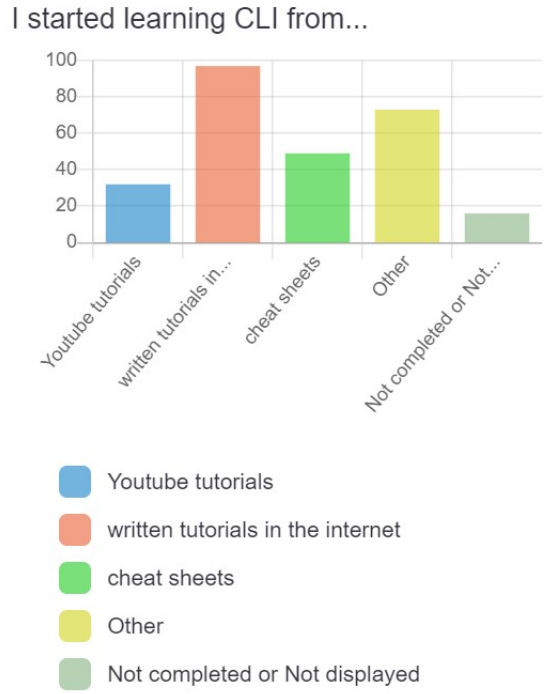


This would also give us the main resources where could the long learning curve of the CLI be shortened from and it motivated us to further look into how do they improve the CLI learning process and how can even they be improved.

3.3.2 Substantive questions

With **question 8** began the Likert scale questions(strongly disagree/mildly disagree/neutral/ mildly agree/strongly agree). In question 8, the respondents did not confidently select one side, 92 of the 165 responses were either neutral or mildly agree. We believe this has more to do with the respondents not properly understanding the question, however, with this question reiterated in the interviews, we delved deeper into similarities of learning programming languages and learning the CLI. Besides this, the remaining 44 people strongly agreed that the CLI more closely resembled a mental model similar to programming than GUI.

Figure 3.4: Question 7



In **question 9**, more than half (83) of the respondents strongly agreed that they thought they could achieve more in less time using the CLI, 36 mildly agreed with this, also there was a number of people which complete this question, 26, 14 were neutral, 5 mildly disagreed and one strongly disagreed.

In **question 10**, the results were very similar to that of question 9 with 84 out of 165 strongly agreeing, 30 mildly agreeing, 15 being neutral, 29 not completing this question and 6 either strongly or mildly disagreeing.

In **question 11**, as 25 people answered that they are neutral to GUI's speed of operation being slower than that of the CLI's, 39 people mildly agreed and 29 people did not complete this answer and 10 people mildly or strongly disagreed with this statement, we understood that programmers do not feel as strongly about this particular advantage of CLI's over GUI's.

The question 12 resulted as an even more balanced answer with mildly disagree being the most selected option, 32.12 percent or 53 votes, strongly disagree 19 votes, neutral and not completed 32 votes each, mildly agree 26 votes and strongly 3. Seeing as it has been reported in literature that the CLI is more error prone than GUI [3], [4], we expected this would be a reason people would choose to shy away from using the command line. The results did not prove this to be true. This made us also question how is the fact that the CLI is more error prone than GUI interpreted from programmers ?

Although we expected that **question 13 and 14** would give us more reasons that developers avoid the CLI, the results of the survey showed otherwise as in both of these questions the majority of the answers tended from Neutral to Strongly Disagree. In particular for the thirteenth question, 33 strongly disagreed, 52 answered that they mildly disagreed, 21 were neutral, 18 mildly agreed, 7 strongly agreed and 32 did not complete the question. And in the fourteenth question 53 strongly disagreed, 45 mildly disagreed, 15 were neutral, 17 mildly agreed, 3 strongly

agreed and 32 did not complete the answer. Remembering commands was a problem noted by respondents also in other questions.

Question 15 was an open-ended question. Among the most mentioned keywords, as can be seen from table 3.3, the most mentioned was saynone, 48 times, the second most mentioned was "man page"/"documentation", 19 times, the third most mentioned was "searching"/"googling"/ "remembering", 13 times, and the fourth most mentioned was "discoverability", 6 times. Some of the comments that were made about each of these keywords can be seen in table 3.4.

Keyword	# of times mentioned
none	48
man page/documentation	19
searching/googling/remembering	13
discoverability	6

Table 3.3: Q15, most mentioned keywords

Keyword	Selected comments
man pages/documentation	"hard to filter the man pages looking for the parameter you want" "poor man pages for programs" "some/many man pages OS manuals are obtuse and require foundational experiences to fully grok what is being documented." "Documentation is often extremely verbose/complex/dense"
searching/googling/remembering	"I have to google most of the time" "Spending time to research what commands do" "memorize all possibilities" "Not able to find a tool and need to write one for work."
discoverability	"Again discoverability, otherwise it was a pleasing experience" "Discoverability. Knowing what commands exist in Bash can be tricky, whereas a GUI can be much more discoverable" "difficult shell script syntax, lack of discoverability(as one gets from menus)"

Table 3.4: Q15, keywords (on the left) and respective comments made from respondents about them (on the right)

Question 16 only confirmed what was written by Jeffrey M. Perkel, where in total 8 people answered that they either use GUI with a little CLI or Only GUI for tasks such as handling big data, manipulate spreadsheets, parallelize your work and automation, 104 people answered that they either use Only CLI or CLI with a little GUI (47 Only CLI, 57 CLI with a little GUI), 12 people said It does not matter and 41 did not complete the question, this could be attributed to the developers not having to do such tasks in their professional life.

Question 17 and 18 results were very lobsided in favor of the CLI with 100 respondents answering Only CLI in the case of question 17, 19 CLI with a little GUI, 3 it does not matter, 1 GUI with a little CLI, 0 Only GUI and 42 did not complete this answer. While in question 18 102 responded Only CLI, 17 CLI with a little GUI, 3 It does not matter, 1 GUI with a little CLI, 0 Only GUI and 42 did not complete this question.

Question 19 was an open-ended question. The keyword "automation" was the most mentioned with 18 appearances, "repeated tasks" with 9 appearances, "scripting" with 9,"flexibility"/"versatility", 8 appearances and "video"/
"image editing for GUI" with 15 appearances. Answers where these keywords did not appear were analyzed individually. Some of the comments that were made for each of these keywords can be seen in table 3.6.

Keyword	# of times mentioned
automation	18
repeated tasks	9
scripting	9
flexibility/versatility	8

Table 3.5: Q19, most mentioned keywords

Question 20 was a hybrid question with a "Yes", "No" and "Other" option. 39 respondents answered "Yes", 54 answered "No", 24 clicked on the button "Other", 33 left a comment and 48 did not complete the question. Among the 33 comments, the general trend of the answers was that improvements in the documentation could be made and that tab completion/autosuggestion/syntax highlighting that shells like the fish shell offer already do a great job at that and that such an application could also not be as useful as it is thought. "Man Pages"/"Docs" were mentioned 9 times, "tab completion"/"autosuggestion"/"auto completion" were mentioned 7 times.

Question 21 was an open-ended question. The most commonly used keyword in this question was "nothing", 42 times, the second most used were "man pages"/"docs" and "learning alternative shells", both 13 times, "books", 4 times, "tldr", 2 times. Remaining comments that did not include any of these keywords were individually treated and analyzed.

3.4 Summary

A web-based questionnaire was created with 22 questions and distributed over to multiple different channels. The survey was a success as it reached 165 total responses in less than six days. The questions 1 to 7 and 22 are questions about the respondent's prior experience with the CLI, the following questions, i.e after question 7, are all substantive questions which address the research questions with prior assumptions based on existing research. The analysis of the results was made by testing if those assumptions were met or not.

Keyword	Selected comments
automation	"Automation and scripting, CLI is way better than any GUI offered in this term. Want to convert hundreds of PPTX file to PDF in short amount of times. Yeah, just try that using GUI and see how long it will take to do it. CLI is just superior." "Automation on repeated tasks, processing huge amount of structured data" "anything that involves immense amount of user configuration, user created input, and the chaining of arbitrary programs together into a powerful computational pipeline"
repeated tasks/ scripting	"Custom scripts to solve infrequent problems. The cost, in time spent, of building a GUI to handle a once-in-awhile task is huge compared to the cost of duct taping together a half dozen CLI programs to accomplish the same goal" "piping data through multiple commands(easily), data cleaning and exploration (eg find/grep/jqet. for looking through data to find structured or semistructured information)"
flexibility/versatility	"Flexibility. Also robustness. If a command fails on the cli, there are obvious steps to debug it. If a gui fails, it is often more difficult to find the root cause." "Versatility and portability for small case-specific tasks. Sometimes it's necessary to create something really quickly, and it doesn't make sense to instantiate a new application and write dedicated code in a container. That's when I choose a CLI over an IDE or hybrid-IDE-editor."

Table 3.6: Q19, keywords (on the left) and respective comments made from respondents about them (on the right)

In open-ended questions we could get a more detailed insight and answers on our research questions. What we found out from the respondents was that they were mostly very experienced developers with more than 10 years of experience and that they found learning the CLI an en-

joyable experience. Most of the respondents had learned the CLI from written tutorials in the internet (referring to man page like documents), cheat sheets, and man pages, Youtube tutorials was also selected. A result from the survey was that the areas where the CLI performs worse than GUI (questions 12-14) did not prove to be a difficulty for a majority of the respondents, however, as it was noted as a problem by over 10 percent of the respondents in all three cases, they were difficulties that needed further investigation. In addition other difficulties were identified from following questions, man pages/documentation, searching/googling/remembering and discoverability were among the problems that programmers faced while learning the CLI, not necessarily the number of times that you make errors or the number of times you have to consult documentation. Areas where the respondents do not think CLI can be replaced from GUI were also identified. Terms like: “automation”, “manipulating spreadsheets”, “scripting”, “repeated tasks”, “flexibility” were dominant and respondents felt strongly that these are terms where CLI cannot be replaced by GUI. As of what is the best way to alleviate the difficulties developers face, we found results upon which we could further investigate and expand on during the interviews phase. Among the recommandations were improved man pages/ documentation, learning newer more user friendly shells, reading quality textbooks etc.

Chapter 4

Interviews

4.1 Introduction

As the survey analysis were over, our purpose with the interviews was to expand and gain more insight on the answers that we had found from the survey. Especially of interest were:

- What makes the CLI necessary to use in the workplace and why do programmers start using it ?
- Give further insight into the everyday difficulties that software developers have while learning and using the CLI ?
- How to improve and better learn from the man pages/documentation ?
- To explore other forms of learning, besides man pages, that need further investigation.

These were still questions that needed answering and a more detailed discussion to get to those answers. As a result of the findings of our survey and the questions that still remained open, we decided to employ semi-structured interviews where a combination of specific (questions about the background) and open-ended questions were used.

4.2 Method

Acquainted experienced professional software developers were contacted, other experienced professional software developers were reached out through LinkedIn. We received 11 confirmations in total. We conducted 11 face-to-face online interviews. The interviews were conducted through the Zoom platform and except the first interview, which was documented through notetaking by hand, all interviews were recorded and transcribed afterwards. All interviews were conducted for a time-span of 12 to 30 minutes, where the majority of the interviews lasted between 20 and 30 minutes. The subjects included bachelors, masters and doctoral students and professionals with multiple years of experience.

A **core concept of the interviews** was developed in order for us to ensure that our goals with the interviews were met. We asked the interviewees roughly all survey questions, but we also asked the interviewees what their take on the results of the open questions from the survey is as we believed through this we could get more detailed insight into a software developers thinking on our research questions.

4.3 Analysis

We did a thematic analysis of user interviews. [19] [20] Which includes familiarizing yourself with the data, assigning preliminary codes to the data in order to describe content, searching for pattern or themes in the code across the different interviews, reviewing themes and defining and naming themes.

In **the first phase**, we had audio recordings. Although we knew what we were looking for, we transcribed all of the interviews in separate excel sheets, as we did not want to miss out any information and to be able to review everything that was said during the interviews. During this transcription phase we got our first ideas of what the content and the themes of the answers are.

In **the second phase** we had to develop a new excel sheet, where we created a table with all the information from all the interviews (initial table). As can be seen from figure 4.1.

Figure 4.1: Initial table

Interview	years of experience in professional programming	what is your primary operating system	how would you rate you familiarity with the cli	how did you start learning it	command li	which of the following would best descibe your cli experience ?
Interviewee 1	2	macos	i know my way around it, google stuff when i dont know something, its not intuitive	youtube, stack overflow	mac terminal	have to look up non git related tasks
Interviewee 2	2	macos	I know the basic stuff, creating directories, navigation and stuff.		So I suppose with the macos terminal with the zsh right ? I1: Exactly	I1: I for sure have ot look up command, I did not invest a lot of time learning it, so that's why I only know the basic stuff so it guess that quite fun, but that's quite limited
Interviewee 3	I2: In programming 4 years of experience me: From the internships until now ? i2: that would be like 5 months of programming experience	i2: when on university I use windows, but in my internships I always use linux distributions	i2: I would say I could do simple tasks		Im supposing with bash since you use linux right ? i2: exactly yeah	at the beginnning of my first internship I spent a lot of time without progressing, until my supervisor showed me some insights, tricks
Interviewee 4	I would say around 7 years	mostly windows and linux variants based on dbm mostly like dbn variants or ubuntu, so the servers are mostly in dbn and working personally mostly on windows	I know my way around it but I wouldn't say an absolute cli expert Id say somewhere in the middle average	i3: I use the man pages a lot, the built in man pages but I think the man oages are very good if you have already used the command and need to look up some flag or syntax but if its your first time the man page may not be the best point to start,so I had one class at university which was about unix systems where we had a lot of how fundementally unix systems worked and there i learned a lot and then also usual sources like stackoverflow	me: ok and im supposing the cli youre most comfortable with would be the bash ? i3: yeah	regularly I still have to look up certain flags or syntac and so me: ok and did you initally spent a lot of effort without making much progress or did you have a smooth transition into the command line i3: I would say it was smooth, I started mostly using git in the command line and then it went over to unix essentials like grep and stuff like that and then it even at this went to writing scrips and stuff like that yes but I would say it was mostly smooth yeah i think the documentation is rather good on unix command line toold so yeah

The rows represented the answers of each interviewee, while the columns represented all the answers to a particular question.

After this, in order to study the questions more effectively by the topic that they treated, different columns were joined with one another based on the topic of the questions.

Such as in figure 4.2.

The following topics could be recognized:

- Prior experience
- How did you start
- Efficiency and flexibility
- GUI vs CLI
- Cheatsheets and tldr

Prior Experience

Interviewee	years of experience in professional programming	what is your primary operating system	command line shell ?	which of the following would best descibe your cli experience ?
Interview 1	2	macos	mac terminal	have to look up non git related tasks
Interview 2	2	macos	So I suppose with the macos terminal with the zsh right ? I1: Exactly	I1: I for sure have ot look up command, I did not invest a lot of time learning it, so that's why I only know the basic stuff so it guess that quite fun, but
Interview 3	I2: In programming 4 years of experience me: From the internships until now ? i2: that would be like 5 months of programming experience	i2: when on university I use windows, but in my internships I always use linux distributions	Im supposing with bash since you use linux right ? i2: exactly yeah	at the beginnning of my first internship I spent a lot of time without progressing, until my supervisor showed me some insights, tricks

Efficiency, Flexibility

achieve more in less time	While learning CLI, i felt that the CLI offers me freedom to express anything that I want, while I felt GUI limited me in this aspect.
no disagree, i use cli because it gives me more flexibility, i can see logs more	yes, cuz i can code and code is more familiar
yeah I guess, it depends on what, there are some command that you have to type a lot, so maybe not for them, but for the others sure.	
absolutely, if you have some experience with the command line, you can get those commands faster and do everything more effectively	internship for example , I have to start docker and everything else in my cli, and to be honest I cant imagine how I would have to do it with a gui, for some reason I would say yes

Figure 4.2: Topics tables

- Difficulties
- Indispensable to CLI
- Discoverability
- If you were to learn the CLI

Each of the topics contained multiple of the columns of the initial table but a column of the initial table did not appear twice or more on the new tables that were formed according to the topics they represented. Each topic was analysed separately.

In each topic, next to each answer of each question codes were developed. Thus the second phase of our thematic analysis was finished.

In **the third phase**, in order for general patterns and themes to be recognized, we, very carefully, analysed the codes that we had written and searched for similarities and differences between them. Additional notes were taken noting down these patterns and tendencies. These comprised our results and were taken as base for our recommandations.

4.4 Results

In **the Prior Experience and How did you start questions**, most of the interviewees (7 out of 11) answered that they had 4 years of programming experience the other having 7, 5, 5 and 9. They also almost all used MacOS and windows and their corresponding shells. Asked how did they start learning, most of the interviewees said they started from university courses much more as a help tool rather than something that they focused exclusively on. Most of the interviewees expressed that they had learned CLI from university first, which in most cases was only an introduction, and after that they searched for commands as they needed them. One interviewee said "yes[i started learning it as a help tool] because you needed it for a project or something","i would just google[when i needed to look something up] and see what the answers are, i don't think i've ever seen a video on the command line".

Asked **how would they describe their CLI learning experience**, most of the interviewees answered they still have to regularly look up commands and that it was a frustrating learning experience for them. During the interviews there was a strong sense that the interviewees never

needed to have a good big picture of the CLI, that they learned it out of necessity and only learned it as much as they needed it for practical purposes and that is what caused the slow learning curve. Different interviewees said: "for sure i have to look up commands, i did not invest a lot of time learning it, so that's why i know the basic stuff", "i would say at first it was pretty much a black box, like i didn't know what to do, what to type, i just copied command that i found on the internet and pasted it", "i wasn't that comfortable in the beginning, maybe not too confident and i was a bit scared of what it does".

In terms of **efficiency and flexibility** most interviewees believe you can achieve more in less time using the CLI and that the CLI is more flexible than GUI, as they did in the survey, but they also note that you need a bit of experience for that, some interviewees said: "absolutely, if you have some experience with the command line, you can get those commands faster and do everything more effectively" and for flexibility the same interviewee said "in my current internship, i have to start docker and everything else in my CLI and to be honest, i can't imagine how i would have to do it with a GUI".

When it came to **whether the interviewees use more the GUI or the CLI** in their daily professional life or whether they use GUI applications for tasks they could do with the CLI, there was a noticeable pattern that less experienced programmers said they use GUI only or Only GUI and CLI only for certain tasks with one developer even saying, "i never felt the need to dig deep into the CLI", and for more experienced programmers they use either only the CLI or Only the CLI and GUI to execute files/trigger processes. One programmer who had 7 years of professional experience said, " if you take Git there i like the CLI more, i don't like using GUI that much since i feel like it hides the details from me, it hides what it is doing, but there are cases for example when i have to release some branch or deploy it on a server somewhere, i don't always execute the scripts by hand on the command line but i have a button that triggers it and then runs the whole process, for those kinds of things like deploying and releasing, i like to have a GUI, but as soon as i have to do more detailed stuff i prefer the CLI".

The Difficulties topic was the most discussed topic. The difficulties that the interviewees expressed varied from one another, one answer that appeared multiple times was knowing a lot of commands from the beginning, and "The CLI has very little visual feedback after executing a command right". Another interviewee said "the problem with man pages is missing examples and overflowing information, if you don't know shell scripting, you won't understand anything.". Other comments were, "don't know the proper commands and typing errors", "man pages too complicated/too much information". Asked how they overcame such difficulties some of the comments were: "at some point i could remember the commands...just by repetition and learning by doing", another interviewee noted "i would create a notes file where i copied and paste all the commands that i wanted to use so that i could remember it like the entire workflow". A few of the interviewees also noted that man pages can be considered as a help tool, but they would not consider man pages a good learning resource.

After making these questions about the difficulties the programmers had, they were asked about **their take on the results** of the same question in the survey, namely question 15. One pattern was noticeable: discoverability. Comments such as: "hard to discover where to start", "i can definitely relate to the third point[discoverability] especially in the beginning you don't even really know what commands even exist and what could be done at all in the command line", "i don't get any stuff around what's happening around CLI, i don't have the mental model i have for pandas or numpy".

In searching for possible solutions to overcome the discoverability problem, we started to propose to the interviewees two possible solutions: **cheat sheets and tldr**. Cheat sheets meaning a page with separate sections, where each section treated a certain topic and in that section the com-

mands that would be related to that topic would be written, along with that a short description of the command would be given. We found websites on the internet that contained such cheat sheets like [21]. Tldr description can be found on the Alternative solutions chapter. These two solutions were particularly praised by the interviewees. Among the given comments were: “i definitely need something like this cheat sheet, that’s good”, “man, this looks amazing", "this is awesome, i would also do my own cheat sheet”.

In terms of **what is indispensable to GUI** all of the interviewees agreed with the survey results with one of them responding: “flexibility, GUI will never include all the functionality you can have in something, it can only some, but all no, of course not, if i add a new feature to my program, i can use it directly in CLI, in GUI i have to create a button”. Another one said: “I agree with the statements that you just made with all of those statements, and i would add to those the amount of control that you have, GUI feels like it hides the details from you, but for certain tasks you need tight control of what youre doing, you need to really know what you’re doing and then i prefer to do it in the command line”.

In the question **If you were to learn the CLI again** some of the dominating thoughts were to understand how CLI works first, then to do examples and exercises. Online courses were also mentioned. One interviewee noted, “I would when it comes to bash scripting, i would do a more systematic approach, like not ad hoc and on demand when i need it look stuff up but maybe do a proper course or learning resource on bash scripting” others said “i’d probably watch some videos, during the bachelor’s i learned that youtube can be so great, you can see everything in very short amount of time”, “maybe i would learn it a bit more, watch a video or something, so that i have an understanding of how it works and not just go straight into the commands”, “i would look up proper docs or tutorials... codeacademy or coursera and they guide you through exercises and they show you commands that you can type and explore, something like that would be helpful instead of just typing and waiting for what happens”.

As a **final thought** about the process of learning CLI and it’s learning curve, we will leave one of the comments that one of our interviewees said: “My personal experience is whenever you get into programming you get into it not because of CLI, so your primary task is not the CLI, so while it’s probably best to read docs or read books, this approach is much more likely for a programming language like for C#, Java, Javascript, CLI is something like git for me because you just need it to get the job done you’re less likely to read the entire docs before you start programming, so theoretically i agree but practically i think it’s always gonna be a bit of learning by doing when it comes to the CLI”.

4.5 Summary

As the survey was over we needed a more detailed perspective about the results of the survey. Of interest was to have more perspective from experienced software developers on our research questions. Acquainted experienced professional software developers were contacted, other experienced professional software developers were reached out through LinkedIn.

We conducted 11 face-to-face online interviews which, for the majority of the cases, lasted between 20 and 30 minutes. A core concept of the interviews was developed to ensure our research goals were met. A thematic analysis of user interviews was done. Which included familiarizing with the data, assigning preliminary codes to the data in order to describe content, searching for pattern or themes in the code across the different interviews, reviewing themes and defining and naming themes.

Most of our interviewees turned out to have 4 years of programming experience, we also had

interviewees that had 5 to 9 years of experience in programming. Most of them had learned the CLI from university as a help tool and said that they struggled with it. The interviewees believed that CLIs overpowers GUIs when it comes to efficiency and flexibility, but they also noted the learning curve needed for that. Experienced programmers endorsed CLI use, noting that they would use GUI only for easy tasks. Among the difficulties expressed by the programmers was knowing so many commands from the beginning and the little visual feedback that CLI has. Discoverability proved to be the biggest problem for the programmers with knowing too many commands from the beginning posing a problem for the programmers. Cheatsheets and tldr were liked as learning tools from the interviewees. The results from the survey about what is indispensable to GUI were strongly agreed to by the interviewees and if they would learn the CLI again most of the programmers said that they would spend more time to understand how CLI works first, and not start just by copying and pasting commands found through googling.

Chapter 5

Discussion

5.1 Summary of Results

A majority of the respondents of the survey had more than 10 years of experience in programming and considered themselves expert CLI power-users. While in the interviews, most of the interviewees (7 out of 11) answered that they had 4 years of programming experience the others having 7,5,5 and 9. They also almost all used MacOS and windows and their corresponding shells, in contrast to the survey where a large majority had Linux as their primary operating system and used the linux shells (bash, sh, zsh etc.) when interacting with the command line.
Asked how would they describe their CLI learning experience, most of the interviewees answered they still have to regularly look up commands and that it was a frustrating learning experience for them. During the interviews there was a strong sense that the interviewees never needed to have a good big picture of the CLI, that they learned it out of necessity and only learned it as much as they needed it for practical purposes and that is what caused the slow learning curve. While in the survey we found out that the respondents did not struggle learning the command line. Most of them found it an enjoyable experience and they said that they quickly became comfortable with it.
In the question how did they start learning, most of the interviewees said they started from university courses much more as a help tool rather than something that they focused exclusively on. Most of the interviewees expressed that they had learned CLI from university first, which in most cases was only an introduction with basic commands, and after that they searched for commands as they needed them. While in the survey, the methods to learning the CLI varied from one respondent to another, written tutorials in the internet, cheat sheets, man pages, Youtube tutorials, books, mentors etc were all represented.
In question 9, more than half (83) of the respondents strongly agreed that they thought they could achieve more in less time using the CLI, 36 mildly agreed with this, also there was a number of people which did not complete this question, 26, 14 were neutral, 5 mildly disagreed and one strongly disagreed. In question 10, the results were very similar to that of question 9 with 84 out of 165 strongly agreeing, 30 mildly agreeing, 15 being neutral, 29 not completing this question and 6 either strongly or mildly disagreeing. In the interviews this evidence, more than half agreeing that the CLI is more efficient and flexible than GUI, was also supported but the interviewees noted that there is a certain level of experience needed to get to that level.
The question 12 was not one-sided and tended towards the respondents disagreeing to the statement. Seeing as it has been reported in literature that the cli is more error prone than gui [3] [4], we expected this would be a reason people would choose to shy away from using the command line. The results did not prove this to be true. This made us also question how is the fact that

the CLI is more error prone than GUI interpreted from programmers ? One of our interviewees responded that making errors in the CLI is just one of the ways to better learn it and makes learning CLI fun and generally the statement of the question was not supported from interviewees also. Although we expected that question 13 and 14 would give us more reasons that developers avoid the CLI, the results of the survey did not prove this claim as in both of these questions the majority of the answers tended from Neutral to Strongly Disagree.
The Difficulties topic was the most discussed topic in the interviwes. The difficulties that the interviewees expressed varied from one another, one answer that appeared multiple times was knowing a lot of commands from the beginning, and "The CLI has very little visual feedback after executing a command right". Another interviewee said "the problem with man pages is missing examples and overflowing information, if you don't know shell scripting, you won't understand anything.". Other comments were, "don't know the proper commands and typing errors", "man pages too complicated/too much information". A few of the interviewees also noted that man pages can be considered as a help tool, but they wouldn't consider man pages a good learning resource.
Much more information on this topic we received from question 15, where the most mentioned difficulty was man pages/documentation, after that was searching/googling/remembering and discoverability. Among the things that were said about the man pages/documentation were: "hard to filter the man pages looking for the parameter you want", "poor man pages for programs", "some/many man pages OS manuals are obtuse and require foundational experiences to fully grok what is being documented.", "Documentation is often extremely verbose/complex/-dense" etc. In the interviews, the were the programmers were asked about their take on the results of the same question in the survey. One pattern was noticeable: discoverability. Comments such as: "hard to discover where to start", "i can definitely relate to the third point[discoverability] especially in the beginning you don't even really know what commands even exist and what could be done at all in the command line", "i don't get any stuff around what's happening around CLI, i don't have the mental model i have for pandas or numpy" were made.
Question 16 only confirmed what was written by Jeffrey M. Perkel, where 104 people answered that they either use Only CLI or CLI with a little GUI (47 Only CLI, 57 CLI with a little GUI). Question 17 and 18 results were very lobsided in favor of the CLI.
Question 19 was an open-ended question. The keyword "automation" was the most mentioned with 18 appearances, "repeated tasks" with 9 appearances, "scripting" with 9, "flexibility"/ "versatility" 8 appearances and "video"/"image editing for GUI" with 15 appearances. For the same question in the interviews, all of the interviewees agreed with the survey results with one of them responding: "flexibility, GUI will never include all the functionality you can have in something, it can only some, but all no, of course not, if i add a new feature to my program, i can use it directly in CLI, in GUI i have to create a button". Another one said: "I agree with the statements that you just made with all of those statements, and i would add to those the amount of control that you have, GUI feels like it hides the details from you, but for certain tasks you need tight control of what you're doing, you need to really know what you're doing and then i prefer to do it in the command line".
Question 20 was a hybrid question with a "Yes", "No" and "Other" option. 39 respondents answered "Yes", 54 answered "No", 24 clicked on the button "Other", 33 left a comment and 48 did not complete the question. Among the 33 comments, the general trend of the answers was that improvements in the documentation could be made and that tab completion/autosuggestion/syntax highlighting that shells like the fish shell offer already do a great job at that and that such an application could also not be as useful as it is thought. "Man Pages"/"Docs" were mentioned 9 times, "tab completion"/"autosuggestion"/"auto completion" were mentioned 7 times. Question 21 was an open-ended question. The most commonly used keywords, not including "nothing", in this question were "man pages"/"docs" and "learning alternative shells",

then “books” and “tldr”.

5.2 Recommendations

5.2.1 Learning methods

In the survey, most of the respondents were programmers that had learned programming a long time ago, with some even mentioning that there was not internet when they learned the CLI, also the methods that they used to learn the command line were methods that generally take more time, such as written tutorials in the internet, books, man pages, documentation etc. In the interviews, the interviewees were programmers that had started practicing professional software development in the recent years and the methods that they used to learn the command line were all methods that do not take a large amount of time like university slides (which in most cases taught only the basic commands), youtube videos and searching stack overflow. When it came to methods that were used from our survey respondents, they were all considered either as too time consuming, too complicated or similarly by our interviewees.

This difference in this particular result (“How would you describe you CLI learning experience ?”) may be due to the fact that over time programmers have replaced tasks in CLI with those in GUI, while earlier the CLI was the only option. As most of our respondents in the survey had 10+ years of experience in programming, learning the CLI was the only option for them, hence they also devoted more time to it, whereas programmers in our interviews started to program when there was an abundance of GUI applications from which they could select, causing the CLI to be less necessary to use, thus also the programmers having less motivation to use it, thus also the worsened learning experience.

We can look at this question’s answers more closely. While (for the inexperienced) the latter methods (university slides, youtube videos and searching stack overflow) may be more efficient at finding commands you need, the former (written tutorials in the internet, books, man pages) are better at giving the learner a broader overview of the command line and a better understanding of what is possible with the command line, which also was one of the biggest hurdles for programmers reported during the interviews.

While the learning methods that programmers in the survey used were, most likely, the only ones they could use, today programmers can use similar but newer methods to tackle the same hurdles.

5.2.2 Online courses

One suggestion would be to use online courses, which were not mentioned as a learning method in the interviews nor the survey and one of our interviewees noted: "I would look up codeacademy or coursera since they guide you through exercises, show you commands that you can type and explore, something like that would be helpful, instead of just typing and waiting". This idea was supported by multiple other interviewees.

5.2.3 Cheatsheets

Another suggestion we have is cheatsheets. Either ready found somewhere or doing your own cheat sheet. As much as this can sound as a classic helping tool, it was not mentioned in the survey or interviews as a tool programmers use in their daily life, but after showing them a cheat sheet where the page was separated into sections separated by topics, where each section had

commands belonging to that topic, all of the programmers loved it as an idea with some one them even saying that they will start to use it right away in their work. We believe this would help in replacing stack overflow as a tool that programmers have to return to when they have forgotten a command they used a while ago.

5.2.4 tldr

One of the problems we found from the interviews was man pages. All of our interviewees were professional software developers and only one of them reported that he used the man pages in his work. Among other programmers most of them reported that they try to avoid the man pages as much as they can, only using it very rarely. The reasons being that it is too complicated or too difficult to find information or similarly.

For novice programmers or even experienced ones we would recommend the tldr-pages, which are help pages for command line tools, which aim to be a simpler, more approachable complement to traditional man pages. Showing only short explanations for commands along with an example of that command, we believe, would greatly increase the programmers interaction with the command line, it would also shorten the time programmers spend searching on google for particular commands and how to use them. This recommandation was very much liked from the interviewees and they were amazed by it, as none of them had heard about or used it before.

5.2.5 fish shell

The little visual feedback which was reported from some of the programmers in the interviews, we believe would be best addressed by using newer more interactive shells like the fish shell, which they did not report to use or know about, which offer help tools such as: autosuggestion, syntax highlighting, tab completion etc. As one of the respondents of the survey said about these helping tools, "auto completion and syntax highlighting already goes a long way".

While these recommandations are more inclined toward novice programmers, we believe command line designers and developers could also benefit greatly from the results of our survey and interviews, as a lot of aspects of the user experience with the CLI have been covered there.

5.3 Threats and Limitations

One of the limitations of this article can be the convenience sampling used for the survey and the interviews. In the survey's case, the respondents found the survey on the reddit communities where the survey was posted, these communities were: `r/commandline`, `r/bash`, `r/hci`, `r/zsh`. So these respondents were admirers of the CLI who followed each news that came out about the CLI, making it unprobable that these respondents had bad experiences with the CLI. Thus also their answer that they enjoyed their CLI learning experience.

Meanwhile, the interviews participants were selected without regarding their background or preference of the CLI. Thus also their answer that their CLI learning experience was frustrating.

While our survey respondents were probably the best suited to answer our research questions, a more random sampling would more accurately represent the distribution of the usage of the CLI among programmers today. Because of time constraints, this was not possible to do in this project.

Another point from the survey results that is worth mentioning is that one of the most given answers in the 19th question of the survey, "I don't think GUI can replace CLI in...", was "video/image editing for GUI", it was mentioned 15 times. After also discussing this with the software developers in the interviews, we believe the question was misunderstood from these respondents

as most of the developers from the interviews agreed that this keyword does not make a lot of sense in this question.

5.4 Future work

The possibilities for future research in CLI are ample. From further researching the man pages to enhancing the user interactivity. One possible project would be separating two groups of novice developers who do not use the command line, where both of the groups have the same tasks related to the documentation, but one group has to do the tasks using the man page and the other using the tldr, that way the effectiveness of the tldr could be measured.

Another one would be to gather two groups of developers who do not use the command line, where both of these groups have the same tasks related to the command line, but one group uses the bash shell without the interactive tools and the other uses the fish shell. This way the added interactivity of the fish shell could really be measured.

5.5 Conclusion

In conclusion, we could give answers to our research questions. Developers learn the CLI out of necessity for their programming projects, the learning process mostly consists of them searching stack overflow, and typing commands without having a clear comprehension of the commands they are typing. They do not spend the time required to acquire easiness of use as they do with programming languages like C, C++, Java etc. It seems that with time when they deal with more advanced tasks they start appreciating the command line more and that is when they also put more effort into understanding it.
Although these terms are related to one another and a future study could more in depth analyse these aspects of the command line, automation, scripting, repeated tasks, flexibility, handling big data, manipulating spreadsheets, parallelize your work all are aspects of the command line which experienced programmers do not believe that GUI can replace the CLI in.
The areas where programmers have the most difficulties with the CLI is the man pages/documentation, discoverability, rememebering of the commands, little visual feedback and such. The tools that we proposed to overcome these problems were online tutorials, creating by yourself or finding cheatsheets, using the tldr instead of man pages and using newer more interactive shells like the fish shell.

Appendix A

Survey Questions

1. How many years of experience do you have in programming ?
    - Less than 2
    - 2-5
    - 5-10
    - 10+
2. Which is your primary operating system?
    - Windows
    - MacOS
    - Linux
    - Other
3. How would you rate your familiarity with CLI?
    - None or very limited
    - I can perform simple tasks using CLI
    - I know my way around CLI
    - I'm an expert / CLI power-user
    - Other
4. Which kinds of CLI are you comfortable with? (multiple choice)
    - UNIX-style Shell (Bash, sh, zsh, etc.)
    - Windows command prompt (cmd)
    - Windows PowerShell
    - R / GNU Octave / MATLAB
    - Bloomberg Terminal
    - Other
5. Which of the following best describes your CLI learning experience? (multiple choice)
    - I initially spent a lot of effort without making much progress

- I quickly became comfortable with using the CLI for my intended purposes
- Overall, learning CLI was a frustrating experience
- Overall, learning CLI was an enjoyable experience
- Even though I use CLI regularly, I still have to look up commands and syntax on a regular basis
- Other

6. Did you switch from GUI to CLI or vice-versa during your programming career?

   - Yes, I switched from GUI to CLI
   - Yes, I switched from CLI to GUI
   - I started with CLI and I started to also use GUI later on.
   - I started with GUI and I started to also use CLI later on.
   - Other

7. I started learning CLI from...

   - Youtube tutorials.
   - written tutorials in the internet.
   - cheat sheets.
   - Other

   - While learning CLI, i felt that...

8. CLI more closely resembled my mental programming model than GUI, so I felt more inclined to use it

   - Strongly Disagree
   - Mildly Disagree
   - Neutral
   - Mildly Agree
   - Strongly Agree

9. I can achieve more in less time using the CLI.

   - Strongly disagree
   - Mildly disagree
   - Neutral
   - Mildly agree
   - Strongly agree

10. The CLI offers me freedom to express anything that I want, while I felt GUI limited me in this aspect

    - Strongly disagree
    - Mildly disagree
    - Neutral
    - Mildly agree

- Strongly disagree

11. GUI's speed of operation was much slower than CLI's, and it bothered me.
    - Strongly disagree
    - Mildly disagree
    - Neutral
    - Mildly agree
    - Strongly agree
12. The number of times I made mistakes in CLI bothered me.
    - Strongly disagree
    - Mildly disagree
    - Neutral
    - Mildly agree
    - Strongly agree
13. The number of times I had to consult documentation while using CLI bothered me.
    - Strongly disagree
    - Mildly disagree
    - Neutral
    - Mildly agree
    - Strongly agree
14. Having to remember CLI commands by heart bothered me.
    - Strongly disagree
    - Mildly disagree
    - Neutral
    - Mildly agree
    - Strongly agree
15. Other aspects where i had difficulties while learning CLI are...
    - Opinion.

    The following statements relate to advantages CLI might have over GUI...

16. In data handling tasks such as handling big data, manipulate spreadsheets, parallelize your work, automation, what do you prefer to use more ?
    - Only CLI
    - CLI with a little GUI
    - It does not matter
    - GUI with a little CLI
    - Only GUI

17. In solving repeated tasks, which do you prefer to use more ?
    - Only CLI
    - CLI with a little GUI
    - It does not matter
    - GUI with a little CLI
    - Only GUI
18. To build small case-specific tools for later re-use I prefer...
    - Only CLI
    - CLI with a little GUI
    - It does not matter
    - GUI with a little CLI
    - Only GUI
19. I don't think GUI can replace CLI in...
    - Opinion
20. Would you like that the CLI had an integrated GUI like assistant while you were learning or using it ? Examples: Command suggestion, faulty command correction, pop-ups with documentation or running examples, etc.
    - Yes
    - No
    - Other
21. If you were to learn CLI again for the first time today, what would you do different?
    - Opinion
22. How did you find this survey ?
    - Reddit
    - YCombinator
    - Recommended from someone
    - Other

Bibliography

[1] "HistoryOfTheCommandLineInterface," https://en.wikipedia.org/wiki/Command-line_interface, 2021, Accessed: 2021-11-03.

[2] K. Morgan, R. Morris, and S. Gibbs, "When does a mouse become a rat? or... comparing performance and preferences in direct manipulation and command line environment," *The Computer Journal*, vol. 34, no. 3, pp. 265–271, 1991.

[3] T. Fellmann, M. Kavakli *et al.*, "A command line interface versus a graphical user interface in coding vr systems," in *Proceedings of the Second IASTED International Conference on Human Computer Interaction. ACTA Press*, 2007.

[4] A. Voronkov, L. A. Martucci, and S. Lindskog, "System administrators prefer command line interfaces, don't they? an exploratory study of firewall interfaces," in *Fifteenth Symposium on Usable Privacy and Security ({SOUPS} 2019)*, 2019, pp. 259–271.

[5] A. Feizi and C. Y. Wong, "Usability of user interface styles for learning a graphical software application," in *2012 International Conference on Computer & Information Science (ICCIS)*, vol. 2. IEEE, 2012, pp. 1089–1094.

[6] R. S. Thompson, E. M. Rantanen, W. Yurcik, and B. P. Bailey, "Command line or pretty lines? comparing textual and visual interfaces for intrusion detection," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007, p. 1205.

[7] J. M. Perkel, "Five reasons why researchers should learn to love the command line." *Nature*, vol. 590, no. 7844, pp. 173–174, 2021.

[8] S. J. Westerman, "Individual differences in the use of command line and menu computer interfaces," *International Journal of Human-Computer Interaction*, vol. 9, no. 2, pp. 183–198, 1997.

[9] A. G. Durham and H. H. Emurian, "Learning and retention with a menu and a command line interface," *Computers in human behavior*, vol. 14, no. 4, pp. 597–620, 1998.

[10] W. C. Ogden and J. M. Boyle, "Evaluating human-computer dialog styles: command vs. form/fill-in for report modification," in *Proceedings of the Human Factors Society Annual Meeting*, vol. 26, no. 6. SAGE Publications Sage CA: Los Angeles, CA, 1982, pp. 542–545.

[11] B. Hasan and M. U. Ahmed, "Effects of interface style on user perceptions and behavioral intention to use computer systems," *Computers in Human Behavior*, vol. 23, no. 6, pp. 3025–3037, 2007.

[12] S. Ben and P. Catherine, "Designing the user interface," *Strategies for*, 2005.

[13] E. Dillon, M. Anderson, and M. Brown, "Comparing mental models of novice programmers when using visual and command line environments," in *Proceedings of the 50th Annual Southeast Regional Conference*, 2012, pp. 142–147.

[14] R. Blum, *Linux command line and shell scripting bible*. John Wiley & Sons, 2008, vol. 481.

[15] "BashIntro," https://cs.lmu.edu/~ray/notes/bash/, 2021, Accessed: 2021-11-03.

[16] "PowerShellIntro," https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1, 2021, Accessed: 2021-11-03.

[17] "fishShellIntro," https://fishshell.com/docs/current/index.html, 2021, Accessed: 2021-11-03.

[18] "tldrIntro," https://tldr.sh/, 2021, Accessed: 2021-11-03.

[19] H. Nekkanti and S. Reddy, "Surveys in software engineering a systematic literature review and interview study," Ph.D. dissertation, MSc thesis, Blekinge Institute of Technology, Sweden, 2016.

[20] "HowToDoAThematicAnalysis," https://www.interaction-design.org/literature/article/how-to-do-a-thematic-analysis-of-user-interviews, 2021, Accessed: 2021-11-03.

[21] "Cheatsheet," https://www.cheatsheet.wtf/bash/, 2021, Accessed: 2021-11-03.