

Book of Specification Vol-TeX-Sh

February 2021



VolTeX-sh\$
not the bourn-again shell

by VolTeX Studio

BOSE Abhishek

BEKHAT Sofiane

JOHN Rajat

JY Loic Mahatsangy

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 1 |
| 1.1 | The Members | 2 |
| 1.1.1 | BOSE "σ" Abhishek | 2 |
| 1.1.2 | JY L0c Loic Mahatsangy | 2 |
| 1.1.3 | JOHN "ω"Rajat | 3 |
| 1.1.4 | BEKHAT "γ"Sofiane | 3 |
| 2 | The Shell | 4 |
| 2.1 | About the Project | 4 |
| 2.1.1 | Basic Lifetime of a shell | 4 |
| 2.1.2 | Basic Loop of a shell | 5 |
| 2.1.3 | Reading a Line | 5 |
| 2.1.4 | Parsing the Line | 5 |
| 2.1.5 | How the shell processes: | 5 |
| 2.1.6 | User interface | 6 |
| 2.2 | Website | 7 |
| 2.3 | Goals | 8 |
| 3 | Task Distribution | 9 |
| 4 | Conclusion: Few Words from the leader | 9 |

1 Introduction

VolteX Studios was formed in 2019. Since that day, as a group we had lots of ups and downs. Our last project, Optical character recognition was well polished project. And to carry on the legacy of this group, we will embark upon another journey which again requires as much hard work if not more. YUP! More frustration and rants, but of course with an output.

The project we decided to work on is to create a Shell, which will be eventually called as Vol-Tex-Sh. We come across different kind of shells every time in our life specially being engineers. And as a group, it always has fascinated us how shells actually work.

In the end we want our shell to be like bash. We would like to focus more on the interactive part of the shell more than the scripting-oriented aspect of it. We would like to have our own small-interpreted language. That's what we are striving to emulate.

Before diving into the details of our progress. Here is a quick introduction from the team and what they have to say about the new journey we have embarked upon.

1.1 The Members

1.1.1 BOSE "σ" Abhishek

Bonjour! I would like to introduce myself. My name is Abhishek BOSE and as of writing this I am a S4 student at EPITA and the leader of VolTeX Studios. Last project was a learning space. Creating an optical character recognition when one has no idea about how neural networks work was kinda daunting at first. But sheer hard work in researching about it helped us surpass that daunting phase.

The last semester gave me a chance to learn about C and how it works. I won't say we as a group have mastered C as a low level language. But we feel we know enough to embark upon this challenge. Honestly, I would have preferred doing this project on C or JavaScript as file manipulation is much simpler. But no hard feelings.

As a group leader, I will strive to make my group and myself work to the mentioned goals that we will discuss later on this document.

1.1.2 JY L0c Loic Mahatsangy

Hi i am Loic and in this project I ,like all my colleagues, will implement the commands and builtins for our shell. I worked on the neural network for the last epita project and from it i learned a great value in team work and good communication.

I will do my best to help this new team. I am specially interested in this project because it would help us have a deeper grasp on shells, how they work and how they are implemented. I learned a lot about shells in SUP thanks to an ACDC, he got me into zsh and OhMyZsh.

1.1.3 JOHN "ω"Rajat

Hello there! My name is Rajat John and I will be helping with the access rights and some network aspects.

I have had no experience with shells other than the tps provided in Epita, hence this will be a challenge but one that I am ready to face

The implementation of custom commands interested me the most as it gives the team a way to express their ideas in a practical way.

As much as I would like to toot my own horn (with good reason of course), I would be nowhere without my amazing team. Working with the same people tends to create bonds that will last a lifetime and hopefully we learn a little about a shell along the way.

1.1.4 BEKHAT "γ"Sofiane

Ciao! I am Bekhat Sofiane and I am a second year student at EPITA. Last year, I had fun doing the Game project and I believe that I learnt and gained a lot from it. Recently, I had a chance to work in an OCR project with this amazing team and it wasn't an easy task to fulfill. But with determination and hard work we managed to finish it. Thanks to the last two projects, I am more confident in my programming knowledge and I believe that with my team, VolTeX Studios, we can tackle this project with confidence and no fear.

Now, let's talk about my programming skills, shall we?. I have one year of experience in C# and Python. I also have a bit knowledge in HTML5 and CSS. I gained experience in C thanks to the OCR project. I don't like C very much because it is an outdated language and also does not give a lot of freedom in terms of coding experience. But hey!, sometimes we need to get through some tough times in life.

I am looking forward to improve my knowledge and learn new concepts in this project. Forza VolTeX Studios !!!!!

2 The Shell

2.1 About the Project

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

The shell is an interactive interface that allows users to execute other commands and utilities in Linux and other UNIX-based operating systems. When you login to the operating system, the standard shell is displayed and allows you to perform common operations such as copy files or restart the system.

2.1.1 Basic Lifetime of a shell

Let's look at a shell from the top down. A shell does three main things in its lifetime.

- **Initialize:** In this step, a typical shell would read and execute its configuration files. These change aspects of the shell's behavior.
- **Interpret:** Next, the shell reads commands from stdin (which could be interactive, or a file) and executes them.
- **Terminate:** After its commands are executed, the shell executes any shutdown commands, frees up any memory, and terminates.

These steps are so general that they could apply to many programs, but we're going to use them for the basis for our shell. Our shell will be so simple that there won't be any configuration files, and there won't be any shutdown command. So, we'll just call the looping function and then terminate. But in terms of architecture, it's important to keep in mind that the lifetime of the program is more than just looping.

2.1.2 Basic Loop of a shell

So we've taken care of how the program should start up. Now, for the basic program logic: what does the shell do during its loop? Well, a simple way to handle commands is with three steps:

- **Read:** Read the command from standard input.
- **Parse:** Separate the command string into a program and arguments.
- **Execute:** Run the parsed command.

2.1.3 Reading a Line

Reading a line from stdin sounds so simple, but in C it can be a hassle. The sad thing is that one doesn't know ahead of time how much text a user will enter into their shell. One can't simply allocate a block and hope they don't exceed it. Instead, one need to start with a block, and if they do exceed it, reallocate with more space.

2.1.4 Parsing the Line

As the name suggests, we parse the line after reading the line into arguments which will be used to do specific tasks.

2.1.5 How the shell processes:

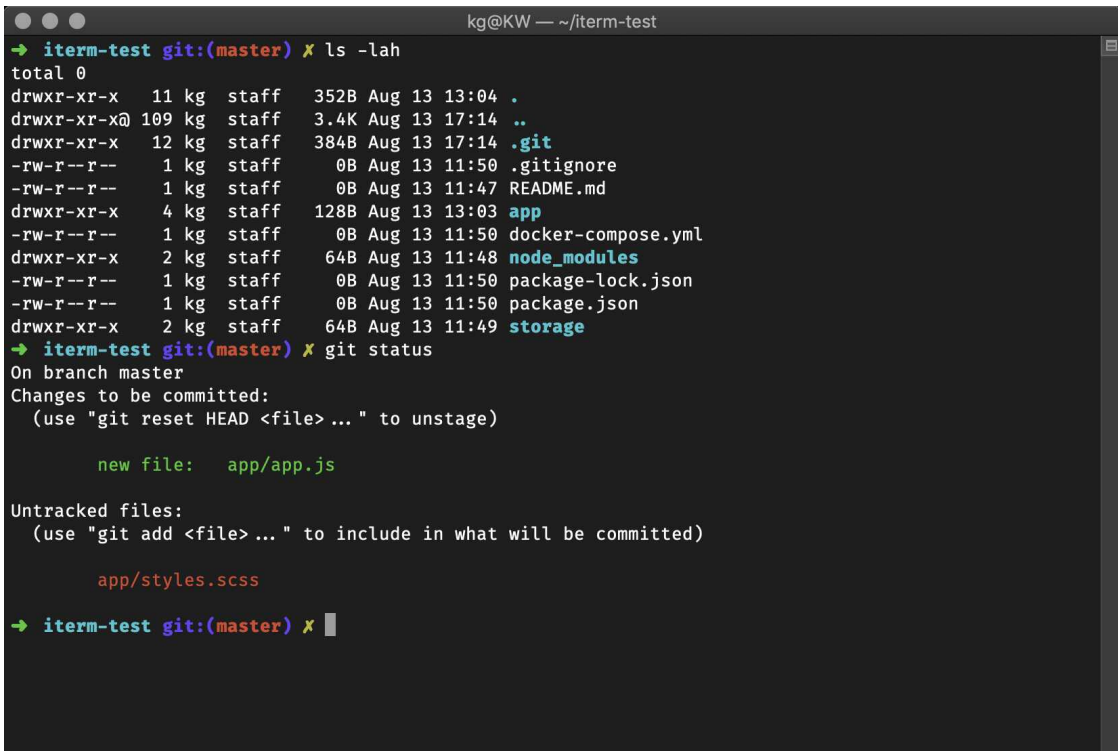
Now, we are really at the heart of what a shell does. Starting process is the main function of shells. So writing a shell means that one need to know exactly what is going on with processes and how they start.

We will use one way for processes to get started: the `fork()` system call. When this function is called, the operating system makes a duplicate of the process and starts them both running. The original process is called the "parent", and the new one is called the "child". `fork()` returns 0 to the child process. In essence, this means that the only way for new processes is to start is by an existing one duplicating itself.

2.1.6 User interface

Regarding the UI of the shell, we will implement a basic, simple user interface that is appealing to the user. The UI will have a theme, with different colors that serve different purposes. For this end, the GTK library will be used in order to implement the UI.

Approximately, our shell will look like the following images :

A screenshot of a terminal window with a dark background and light-colored text. The window title is 'kg@KW — ~/item-test'. The prompt is '→ item-test git:(master) ✕'. The first command is 'ls -lah', which lists files with permissions, sizes, dates, and names. The second command is 'git status', which shows the current branch and staged changes. The output shows a new file 'app/app.js' staged for commit. The prompt is '→ item-test git:(master) ✕' again.

```
kg@KW — ~/item-test
→ item-test git:(master) ✕ ls -lah
total 0
drwxr-xr-x  11 kg  staff   352B Aug 13 13:04 .
drwxr-xr-x@ 109 kg  staff   3.4K Aug 13 17:14 ..
drwxr-xr-x  12 kg  staff   384B Aug 13 17:14 .git
-rw-r--r--   1 kg  staff    0B Aug 13 11:50 .gitignore
-rw-r--r--   1 kg  staff    0B Aug 13 11:47 README.md
drwxr-xr-x   4 kg  staff   128B Aug 13 13:03 app
-rw-r--r--   1 kg  staff    0B Aug 13 11:50 docker-compose.yml
drwxr-xr-x   2 kg  staff   64B Aug 13 11:48 node_modules
-rw-r--r--   1 kg  staff    0B Aug 13 11:50 package-lock.json
-rw-r--r--   1 kg  staff    0B Aug 13 11:50 package.json
drwxr-xr-x   2 kg  staff   64B Aug 13 11:49 storage
→ item-test git:(master) ✕ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file> ..." to unstage)

    new file:   app/app.js

Untracked files:
  (use "git add <file> ..." to include in what will be committed)

    app/styles.scss

→ item-test git:(master) ✕
```



```

kg@KW — ~/iterm-test
-----
~/iterm-test(master*) » ls -lah
total 0
drwxr-xr-x  11 kg  staff   352B Aug 13 13:04 .
drwxr-xr-x@ 109 kg  staff   3.4K Aug 13 17:14 ..
drwxr-xr-x  12 kg  staff   384B Aug 13 17:14 .git
-rw-r--r--   1 kg  staff    0B Aug 13 11:50 .gitignore
-rw-r--r--   1 kg  staff    0B Aug 13 11:47 README.md
drwxr-xr-x   4 kg  staff  128B Aug 13 13:03 app
-rw-r--r--   1 kg  staff    0B Aug 13 11:50 docker-compose.yml
drwxr-xr-x   2 kg  staff   64B Aug 13 11:48 node_modules
-rw-r--r--   1 kg  staff    0B Aug 13 11:50 package-lock.json
-rw-r--r--   1 kg  staff    0B Aug 13 11:50 package.json
drwxr-xr-x   2 kg  staff   64B Aug 13 11:49 storage

~/iterm-test(master*) » git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file> ..." to unstage)

    new file:   app/app.js

Untracked files:
  (use "git add <file> ..." to include in what will be committed)

    app/styles.scss

-----
~/iterm-test(master*) » █

```

2.2 Website

For the website we have thought of making it responsive, basically which means it will respond to different screen size. We will be using of course HTML and CSS, along with that a bit of JavaScript to make it more interactive.

Structure of the website:

- Home Page
- A project presentation (history, members, completion timing, problems encountered and possible solutions).
- The links on the website (members, software, images, sounds, libraries, applets and other elements you may have used)
- A download of the report and the project including a light version (without music files, avi files or other useless elements.)

We will be using GitHub pages to display our website.

2.3 Goals

Here are our goals for each defence:

1. First defence (March 29nd):

For the Shell:

- A basic shell with a custom theme (colors and symbols).
- 20 basic commands (bash builtins).
- Done pipes and fd redirection to our shell.

For the website:

- A presentation of the project and the members.
- A journal and links to our Github repository and sources.
- Everything discussed in the specification

2. Second defence (May 3rd):

For the Shell:

- One more custom theme.
- 20 more commands.
- Done job control and signals handling.
- Done quoting and expansion (globbing).

For the website:

- Updated website.

3. Last defence (June 14th):

For the Shell:

- 30 more commands.
- Done commands history and auto-completion

For the website:

- Final version of the website.

Some tasks may be added during the process.

3 Task Distribution

Below is a table of the global task distribution so far. However all members will touch a bit of everything since we will add more tasks and the tasks can change.

| Task | Abhishek | Rajat | Sofiane | Loic |
|-------------------------|----------|-------|---------|------|
| Themes (shell themes) | | | X | |
| Interactive usability | X | X | X | X |
| System Access | | X | | X |
| Website | X | | | |
| concurrency(jobs/pipes) | | | | X |

Themes: the different shell themes.

Interactive usability: the commands we will implement, including some bash built ins.

System Access: working with files, processes, windows, databases, system configuration, signals...

Concurrency: pipes and re directions.

4 Conclusion: Few Words from the leader

Obviously as one can witness, it is kind of difficult project to implement. But that's where Voltex Studios have the edge at. We are one of the best out there that makes difficult jobs into simpler ones. Maybe that's because of our problem solving strategies and methods. Jokes aside, we are really excited to complete this project to our expectation and execute it. There are lot of details to cover as discussed above in the specifications.

But we believe in ourselves. I believe in our boys. We are totally committed to leave everything out there in the battlefield. We hope you will be fascinated and excited after reading this and we hope to maintain the same amount of excitement in your face in the last defense when we will be ready with our shell.

Thanks