Chapter 1: Introduction

The purpose of this work is to make a step towards the common user in what refers using terminal for executing different commands - that is to imitate at its top level the communication in plain text between the user and his Linux distribution command line interface (CLI).

The trouble starts where GUI ends, and it does end pretty soon after the initial setup of the system. Most of the common tasks: install/uninstall software, edit a file, unzip, create a symlink, etc.; require root privileges and are much easier executed using the terminal. The difficulty is that all these bash commands are pretty long and far from easy to remember.

The research done and described in the paragraphs below was meant to serve this only goal – of making using Linux terminal a pleasure even for beginners and Windows users.

As the script could not support all possible commands right off the start, it was of great importance to choose which should be implemented first, then how should this be
done.

This document will not include any information on how Linux, Linux terminal or any of its components are built. It will only tell how commands are parsed and how these are linked to actual code that is to be executed. Beside these, a sneak peak view over the process of auto-completion and determining the right keywords is included. Also, all the struggles that were overcome, all the problems that were proposed for solution and the ones that didn't find an answer will be mentioned. Relevant snippets of code are included along with explanations.

1.1 Background

In the philosophy of language, a natural language (or ordinary language) is any language which arises in an unpremeditated fashion as the result of the innate facility for language possessed by the human intellect. A natural language is typically used for

communication, and may be spoken, signed, or written. In computer science we are speaking of slightly different area related to natural language - Natural language processing (NLP).

NLP is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve natural language understanding -- that is, enabling computers to derive meaning from human or natural language input.

The ability to use natural language in order to perform common tasks in CLI is essential for simplifying human-computer interaction.

As of yet, I did not find any software or script that had exactly the same purpose as SIUNL has. This is why the topic appears to be a fine field of study with potential of further development.

Nevertheless, there are several package managers that were built specifically to make certain tasks easier. Two of them are *pip* and *easy_install*.

pip allows installing and managing software packages written in Python. Most of the packages are located in the Python Package Index (PyPI).

EasyInstall is a package manager for the Python programming language that provides a standard format for distributing Python programs and libraries

The features that were implemented in SIUNL were determined based on the main ones of simplified package management tools, as *pip* and *easy_install* are. To name a few: autocomplete, easy command line interface, flexibility (refering to the option of installing a batch of certain packages via *pip*, having a strictly structured .txt file)

1.2 Motivation

The main advantage of *pip* is the ease of its command-line interface, which makes installing Python software packages as easy as issuing one command: *pip install some-package-name*. *EasyInstall* has quite the same advantages, although it has less functions (for example it doesn't have the possibility to update packages, what *pip* does).

Both package managers have the single purpose of making managing Pyhton packages a breeze. Also, both package managers have autocomplete functions, especially useful when one is choosing the module to be installed.

As mentioned before, simplifying interaction between a user and his Linux distribution should increase the spread of this free OS, especially the possibility of using the same commands not depending on pre-installed package management tool or distribution in use. Consistency is the key.

This research is important because the main goal of any computer is to make life easier, and making it, doesn't only imply building an awesome computer that understands only very special technical or even non-technical – machine language, but also the plain human or Natural Language. Being able to execute commands using simple words, will lead to an efficient work and easier learning. Also, it will facilitate the use of a PC for any user. This is what SIUNL does.

It will be of great use for making sitting in front of the black window or screen more productive, elevate the use of terminal to another, higher level. Help users save time on routine and not only. Make the use of bash so simple and intuitive that even a child who knows how to type could get some stuff done, because instead of writing *sudo apt-get install some-package*, one could write *install some-package*. That is much easier, and what's more important, is much shorter. The difference is quite big, to be precise – 10 characters, almost one half shorter, but the important thing is that it is much simpler and more intuitive.

1.3 Aim and Objective

The goal of this thesis is building and testing a tool that will make easy the use of command line interface even for a non-experienced Linux user or the one who wishes to migrate from another OS and to analyze the possibilities of further use and expansion of the script. As the development process goes on, the viability and the correctness of chosen direction of this project can be determined.

In order to achieve this goal several objectives must be followed:

- To research the package management tools in different Linux distributions.
- Determine similarities and differences of analyzed distributions.
- Determine what are the advantages of package managers focused on a smaller

area (e.g. *pip, easy_install*).

- Determine the list of common functions that are executed via CLI.
- Write a script that implements easy aliases for these functions.
- Implement auto-complete function.
- Testing and reviewing the script.

1.4 Contribution

In the end of the thesis we will learn and develop a script using the following functionality:

- Learn in detail about how a aliases are created and how auto-complete function can be implemented in bash
- Learn where a bash script can be added in order to be accessible from different location. (e.g. login shell, interactive non-login shell)
- Learn how the script can be tested in real environment.

1.5 Thesis Organization

Chapter X1. Determining constraints.

In this chapter we discuss why certain decisions were taken and why some functions were implemented and the other ones were not.

It will explain what lead to the choice of the first batch of functions, distribution and package manager tool that served as base for further development.

Chapter X2. Conclusion

This chapter gives the conclusion part of the thesis. In this part we have to explain the concluding points which we get during the implementation as well as testing of the project.

Chapter X3. Further Work

This chapter suggests some of the future work to be done with this thesis which could be useful for further research in this field.

A more descriptive structure of the thesis is TBD