BIRZEIT UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

LINUX LABORATORY
ENCS313

Python Project

Title: System Manual Generator for Shell Commands

Prepared by: Jehad Hamayel
IDs: 1200348

Instructor: Dr. Amr Slimi
TA: Eng. Tareq Zidan

Section: 1
January 29, 2024

Table of Contents

Table of Figures

Abstract

The goal of the project is to create a system manual for shell commands using the Python programming language with the help of Object-Oriented Programming principles to enhance modularity, encapsulation, and reusability. Through it, 20 structured XML files are created for each command, as XML files specialize in storing data in a readable, hierarchical form. Because it makes the content easy to understand, it is a guide to shell commands.

Theory

XML Files:

Extensible Markup Language (XML) is a markup language that specifies the rules and procedures for defining, storing, and sharing data across computers. These standards make it easier to communicate data between websites, applications, and databases across any network. Because of their human and machine readability, parties participating in data sharing can quickly read and comprehend the data contained within XML files. [1]

The XML language cannot execute data-computing operations. Instead, it uses other programming languages or applications to accomplish this. As a result, it guarantees structured data management for documents and files.

These comments improve the text's usability without changing the document's content. XML relies on markup symbols or tags to provide additional information about the underlying files. This information allows other software or programs to conduct data processing tasks more efficiently. It also ensures the integrity of structured data.

Python libraries New to use:

xml.etree.ElementTree (ET) and xml.dom.minidom are two Python libraries used for parsing and manipulating XML data.

1- **xml.etree.ElementTree (ET):** This is a simple and efficient library for parsing and creating XML data.

- Library Usage:
    - Parsing XML documents into tree structures that are easy to work with.
    - Searching for elements and data within the XML tree using XPath expressions.
    - Modifying the XML tree (adding, changing, and removing elements or attributes).
    - Creating new XML documents.

2- **xml.dom.minidom**: This library is a minimal implementation of the Document Object Model interface, which is a standard for working with XML, XHTML, and HTML documents.

- Library Usage:
    - Parsing XML into a DOM structure, which represents the document as a hierarchy of objects.
    - Navigating and searching through the DOM tree.
    - Modifying the DOM tree (similar to ElementTree).
    - More suited for documents where the preservation of the XML structure and attributes is important.

Procedure and Discussion:

The complete structure of the system

1- **CommandManual** Class: When you create an Instant from it and give it a specific command name, this class creates the Manual content for the command, which is within the following functions:

- **getCommandName** Function: Through this function, the command name is extracted and stored.
- **getCommandDESCRIPTION** Function: Through this function, the command DESCRIPTION is extracted and stored.
- **getCommandVERSION** Function: Through this function, the command VERSION is extracted and stored.
- **getCommandEXAMPLE** Function: Through this function, the command EXAMPLE with the output is extracted and stored.
- **getRelatedCommandsOfTheCommand** Function: Through this function, the command Related Commands of The Command is extracted and stored.

2- **CommandManualGenerator** Class: Class is specialized in taking the command name and creating an instant from the CommandManual class, which extracts the appropriate data for each command and then creates instans of XmlSerializer class, in which the XML file is created and there are some of the following functions:

- **manualsGenerator** Function: Through this function, all commands are passed through, where the manual data for each command is created and sent to the XmlSerializer class after creating an instance from it, and an XML file is created for the command.
- **manualsGeneratorForCommand** Function: Through this function, a command manual is created for only one command, as in the previous function, an XML file is created.

3- **XmlSerializer** Class: A class specialized in creating and reading an XML file, which contains the following two functions:

- **Serialize** Function: Function specializes in creating an XML file, where it arranges the data inside it in an orderly, beautiful, and hierarchical manner.

- **readXMLFile** Function: Function specializes in reading XML files and printing their content.

4- **recomndation** Function: It is a function that creates lists so that they are classified based on functionality.

5- **commands_index** Function: Function searches for the specific manual in a specific command and then prints it.

6- **Verification** Function: Verify the correctness of the generated content. The existing document is read and verified. If there is a change, the location of the difference is shown.

Apply the system and show the results

System Interface:

When the system is turned on, the menu from which the selection is made appears.

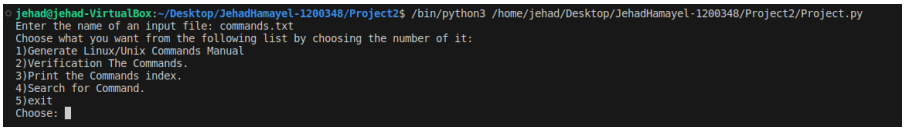

Figure 1: Interface menu

Scenarios:

Scenario1: If there is no commands manual, nothing can be done on it due to its absence, and then the user is given the options again. If it exists, the rest of the options can be applied.

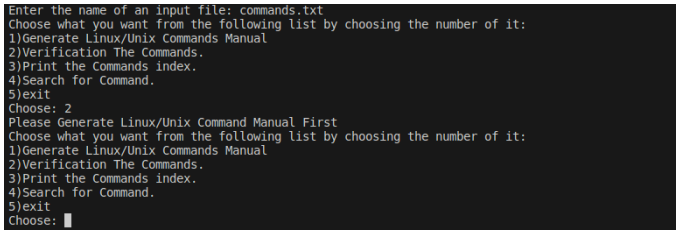

Figure 2: Scenario1 No commands Manuals

Scenario2: When you choose the first option, two options appear. You can create all manuals for commands or for one command:

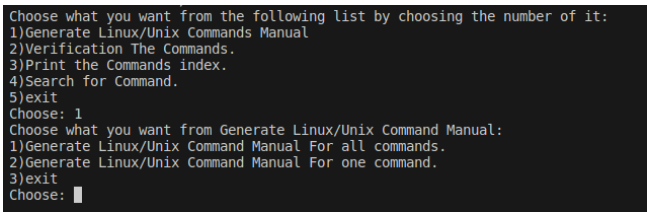

Figure 3: Scenario2 Generate commands Manuals

- Case1: If the first option is chosen. Generate all commands manuals:

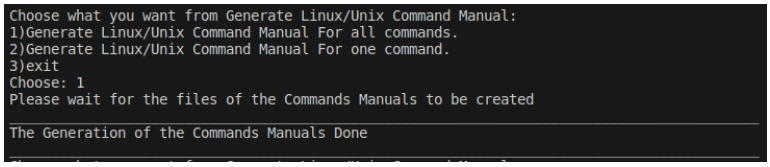

Figure 4 : Generate all commands manuals

- Case2: If the second option is chosen. Generate one command manual. The user is given a choice whether he wants to print the file content or not.

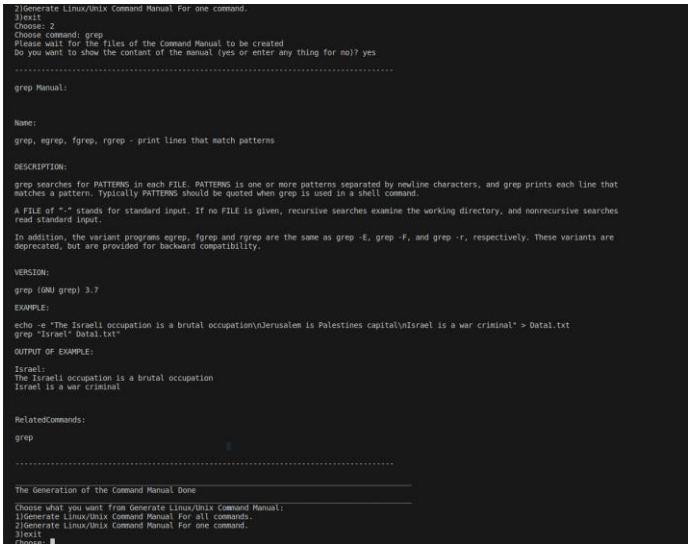

Figure 5: Generate one command manual

There is an option “exit” to undo the previous menu.

The output of this scenario Files or files depending on the case, as shown below:

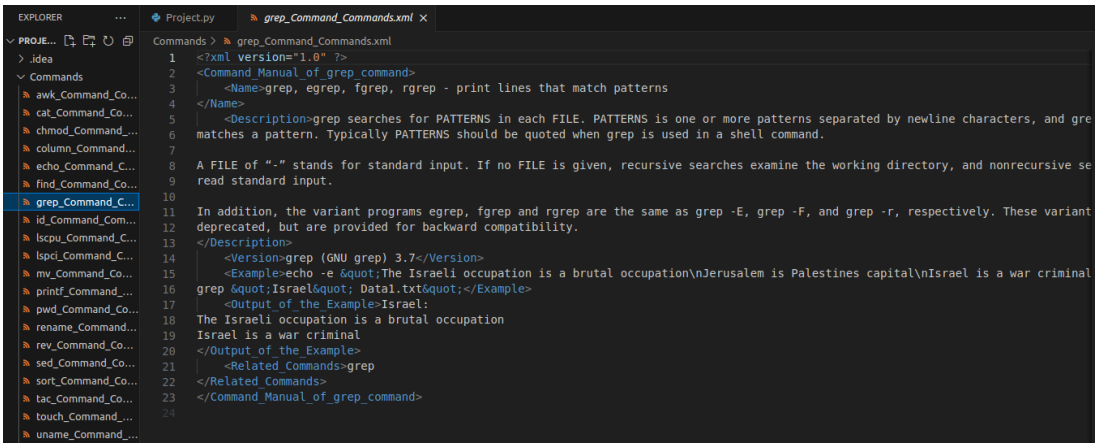

Figure 6: Output of this scenario2

Scenario3: When choosing the Verification process, two cases appear, which are Verification for all commands or for one command, as follows:

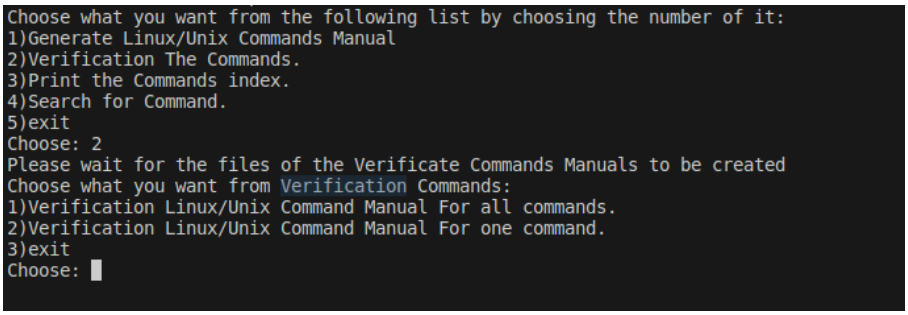

Figure 7: Scenario3 Verification commands

- Case1: When choosing the verification process for all commands, all files are checked, and if there is a difference, it appears, and if there is no difference, it is printed that the verification has been confirmed.

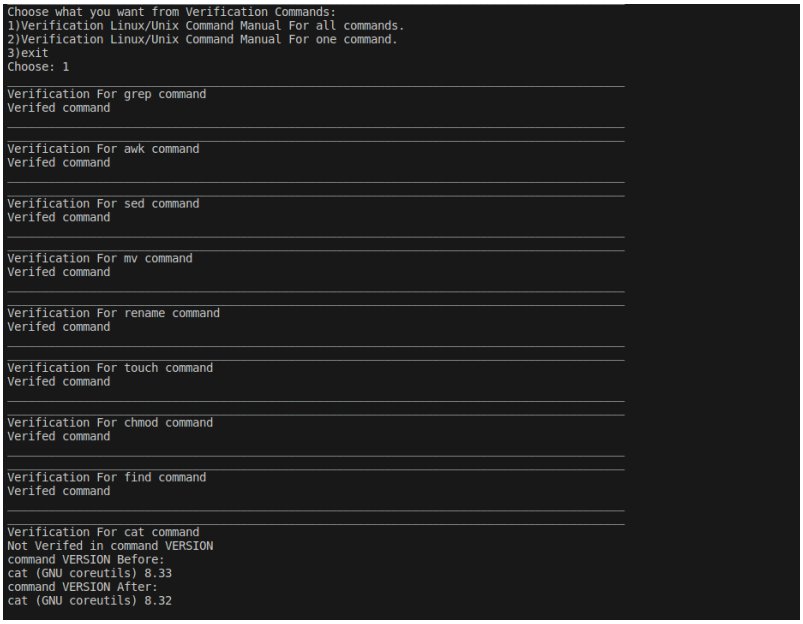

Figure 8: Scenario3 Case1 Verification all commands

- Case2: When selecting the Verification process for a single command, the file is scanned and if there is a difference it appears, and if there is no difference it is printed that Verification has been confirmed.

```
Choose what you want from Verification Commands:
1)Verification Linux/Unix Command Manual For all commands.
2)Verification Linux/Unix Command Manual For one command.
3)exit
Choose: 2
Choose command: grep
_____________________________________________________________________________
Verification For grep command
Verifed command
_____________________________________________________________________________
The Verification of the Command Manual Done
_____________________________________________________________________________
Choose what you want from Verification Commands:
1)Verification Linux/Unix Command Manual For all commands.
2)Verification Linux/Unix Command Manual For one command.
3)exit
Choose: 2
Choose command: cat
_____________________________________________________________________________
Verification For cat command
Not Verifed in command VERSION
command VERSION Before:
cat (GNU coreutils) 8.33
command VERSION After:
cat (GNU coreutils) 8.32
_____________________________________________________________________________
The Verification of the Command Manual Done
_____________________________________________________________________________
Choose what you want from Verification Commands:
1)Verification Linux/Unix Command Manual For all commands.
2)Verification Linux/Unix Command Manual For one command.
3)exit
Choose:
```

Figure 9: Scenario3 Case2 Verification one command

Scenario4: When you choose the Print the Commands index process, the index for the commands will be printed, divided according to functionalty, to facilitate access to and search for the commands.

```
1)Generate Linux/Unix Commands Manual
2)Verification The Commands.
3)Print the Commands index.
4)Search for Command.
5)exit
Choose: 3
Special commands for printing, display, standard output, and listing information:
1) lspci_Command
2) rev_Command
3) pwd_Command
4) cat_Command
5) printf_Command
6) lscpu_Command
7) tac_Command
8) sort_Command
9) echo_Command
10) uname_Command
11) grep_Command
12) id_Command

Special commands in sorting and printing in specific way:
1) rev_Command
2) tac_Command
3) sort_Command
4) grep_Command

Special commands for creating files and changing the file mode:
1) touch_Command
2) chmod_Command

Special commands in filtering, searching, and scanning:
1) sed_Command
2) find_Command
3) awk_Command
4) grep_Command

Special commands in renaming and moving the file location:
1) mv_Command
2) rename_Command
```

Figure 10: Scenario4 Print the Commands index process

Scenario5: When choosing Search for Command, the user is asked to enter the name of the command. If it exists, its content is printed and given the Recommended Commands. If it does not exist, the Recommended Commands in the existing one and anything similar to the Command are given.

- Case1: The first case is when the command is found.

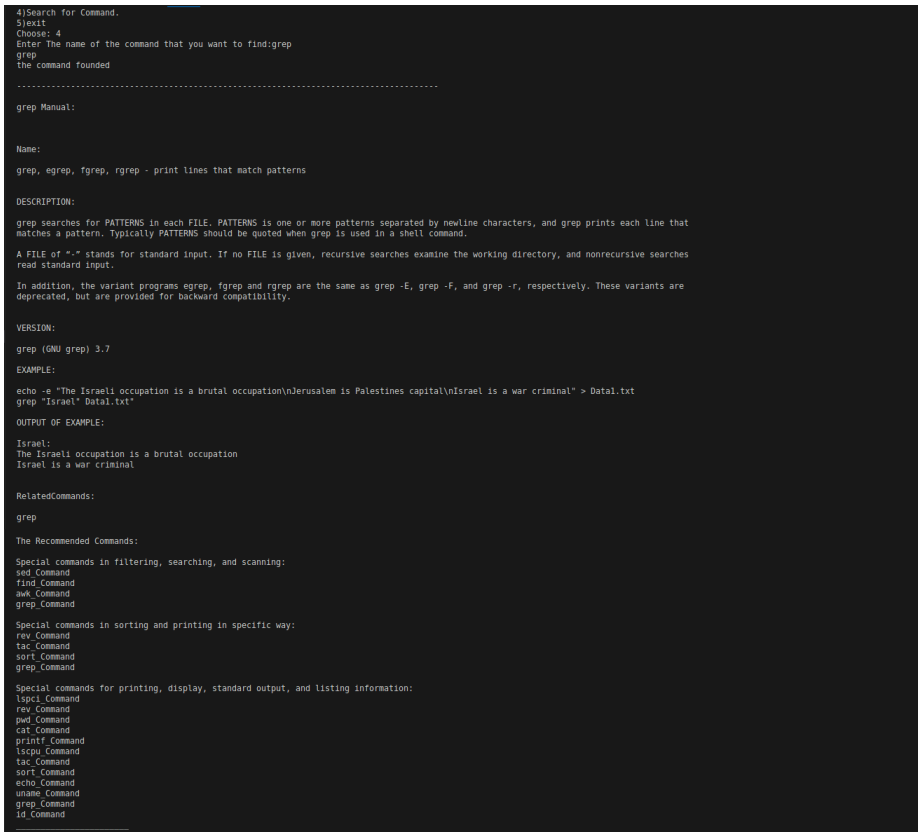

Figure 11: Scenario4 Case1 when the command is found

- Case2: The first case is when the command is not found.

Figure 12: Scenario4 Case2 when the command is not found

Conclusion:

In conclusion, this project brilliantly combines Python's talent for working with XML with the intelligent syntax of object-oriented programming. What we get is a clear, easy-to-navigate guide to shell commands. Each command gets its own neat XML file, which makes the whole thing not only easy on the eye, but also very practical to update or extend due to the hierarchical structure of the file. It's a great example of how a little programming knowledge can turn a complex topic into something accessible and easy to use.

References

[1] Chrome, 8 March 2023. [Online]. Available: https://www.spiceworks.com/tech/tech-general/articles/what-is-xml/#:~:text=Extensible%20Markup%20Language%20(XML)%20is%20a%20markup%20language%20that%20outlines,and%20databases%20on%20any%20network.. [Accessed 29 January 2024].