****

Faculty of Engineering and Technology

Electrical and Computer Engineering Department

ENCS3130 LINUX LABORATORY

**Project 2**

Generate Linux/Unix Command Manual

Prepared by

Jenin Mansour 1200540

Instructor

Dr. Mohammad Jubran

Teacher assistant

Eng. Tareq Zidan

Section 3

BIRZEIT

December – 2023

# **Overview**:

This project aims to automate the creation of a system manual for Python commands using Python programming language. The script reads Python commands from an input file and generates 20 structured XML files for each command, containing information such as command description, version history, examples, related commands, syntax and usage patterns, and online documentation links. The script uses subprocesses to interact with shell commands and retrieve relevant data. The project also implements a verification system, a command recommendation system, and a search functionality to allow users to quickly find information. The project is structured using object-oriented programming principles to enhance modularity, encapsulation, and reusability. The Command Manual Generator is designed to streamline the process of generating manuals for various shell commands and incorporates Object-Oriented Programming principles to enhance modularity, encapsulation, and reusability. The implementation involves two main classes: CommandManualGenerator and CommandManual, and I added two classes Serializer and CommandManualVerification. And 20 command I choosed are date,grep,ss,ps,ls,printf,cat,chmod,vmstat,lspci,pwd,mkdir,rmdir,who,wc,man,uniq,paste,tr,cp. The project not only automates manual generation but also enhances user interaction by incorporating search functionalities and a command recommendation system. Python's versatility and powerful libraries make it an optimal choice for such command-line automation tasks.

# **Procedure and Discussion**

CommandManualGenerator Class:

The CommandManualGenerator class acts as the orchestrator for the manual generation process. It encapsulates logic for reading commands from an input file, generating manuals for each command, and offering additional functionalities such as manual verification. The class maintains a list of CommandManual instances, each associated with a specific command.

class commandMnaualGenerator:  
  
 def \_\_init\_\_(self, input\_file):  
 self.input\_file = input\_file  
 self.manuals = []  
  
 def read\_from\_file(self):  
 # Read commands from command.txt  
 with open(self.input\_file, "r") as file:  
 self.manuals = file.read().splitlines()  
  
 def generate\_manuals(self , option):  
 # Generate manuals for each command  
 for command in self.manuals:  
 generator2 = commandManual(command)  
 Serializer.serialize(generator2, command , option)  
  
 def show\_manual\_from\_xml(self):  
 command\_name = input("Enter the command name: ").lower()  
 deserialized\_manual =Serializer.deserialize(f"{command\_name}\_manual.xml")  
 deserialized\_manual.display\_info()

Key Methods:

read \_from\_file: Reads commands from an input file.

generate\_manuals: Generates manuals for each command. Then use Serializer method to convert the manual object to xml file

show\_manual\_form\_XML: Displays the XML file for a specific command using deserialized method to convert the xml file to manual object.

Command Manual Class:

class commandManual:  
 def \_\_init\_\_(self, command ):  
 self.command = command  
 self.\_description = ""  
 self.\_Version = ""  
 self.\_syntax\_and\_usage = ""  
 self.\_examples = ""  
 self.\_related\_commands = ""  
 self.\_documentationlink=""  
  
 def display\_info(self):  
 print(f"Command: {self.command}")  
 print(f"Description: {self.\_description}")  
 print(f"Version: {self.\_Version}")  
 print(f"Syntax and Usage: {self.\_syntax\_and\_usage}")  
 print(f"Examples: {self.\_examples}")  
 print(f"Related Commands: {self.\_related\_commands}")  
 print(f"Documentation Link: {self.\_documentationlink}")  
  
 def description(self):  
 return \_description  
  
 def set\_description(self, description):  
 self.\_description = description  
  
 # \*  
 def extract\_version(self):  
  
 return \_Version  
  
 def set\_version(self, version):  
 self.\_Version = version  
 # \*  
 def Syntax\_Usage(self):  
  
 return \_syntax\_and\_usage  
  
 def set\_syntax\_and\_usage(self, syus):  
 self.\_syntax\_and\_usage = syus  
 # \*  
  
 def related\_commands(self):  
 return \_related\_commands  
  
 def set\_related\_commands(self, related\_commands):  
 self.\_related\_commands = related\_commands  
 # \*  
  
 def extract\_examples(self):  
  
 return \_examples  
  
  
 def set\_examples(self, example\_lines):  
 self.\_examples = example\_lines  
 # \*\*  
 def documentation\_link(self):  
 return documentationlink  
  
  
  
 def set\_doc(self, documentation\_lines):  
 self.\_documentationlink = documentation\_lines  
 # /////////////////////////////////////////////////////////

Key Methods:

encapsulate and organize information related to a specific shell command. It acts as a container for details such as command description, version history, examples, related commands, syntax and usage patterns, and online documentation links.

CommandManualVerificationclass: that compare between two XML files.

The XML file comparison methodology involves parsing two XML files, extracting root elements, converting them to strings, performing a unified diff, and displaying the differences. Errors can be handled by ensuring the XML strings provided are Unicode strings and specifying encoding as "unicode" when converting XML elements to strings. Additional considerations include proper parsing of XML files, checking for unexpected characters or encoding-related issues, and validating the structure and content of XML files.

# **Runing:**

First there is a menu to choose betwee 5 options

1- Generate Manuals

2- Search Manual for Specific Command

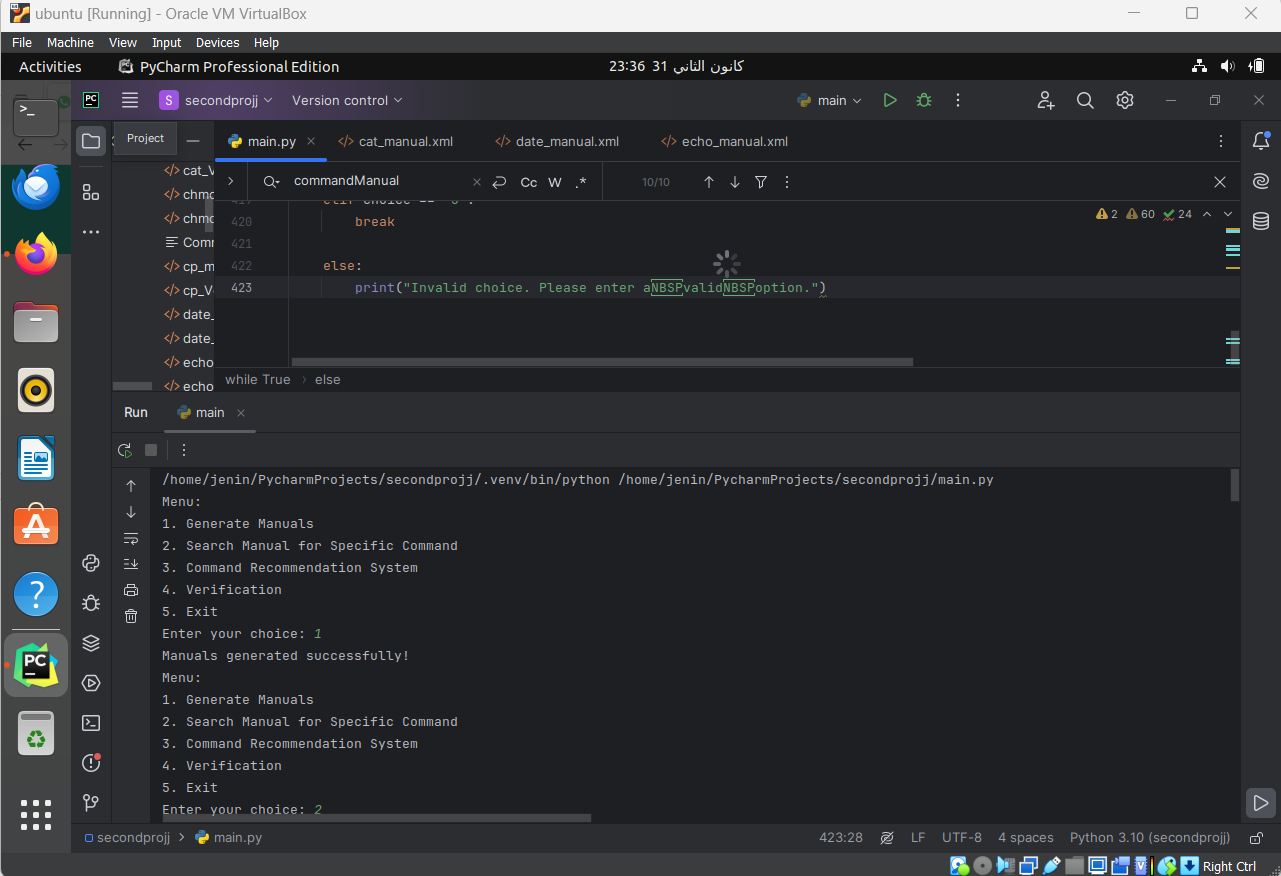
3- Command Recommendation System

4- Verification

5- exit

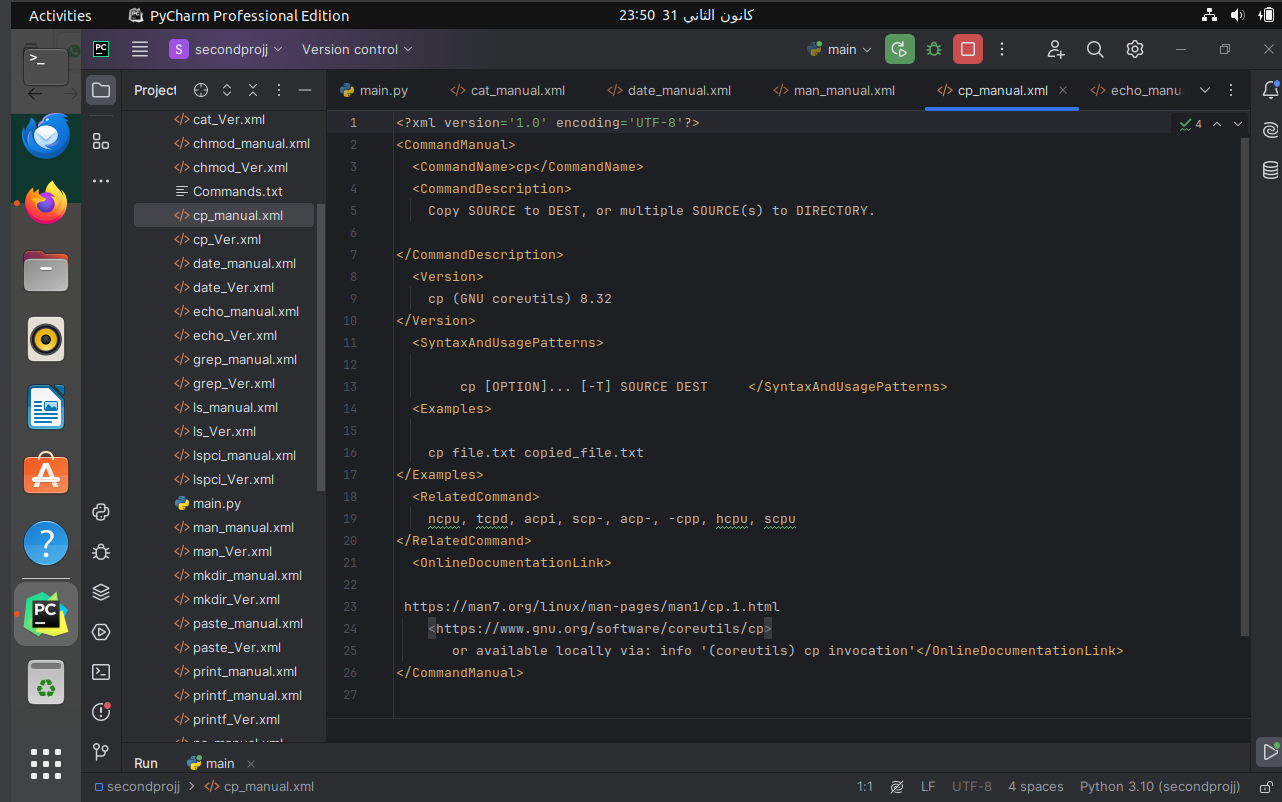
For option 1 :

The manual generated for the 20 command

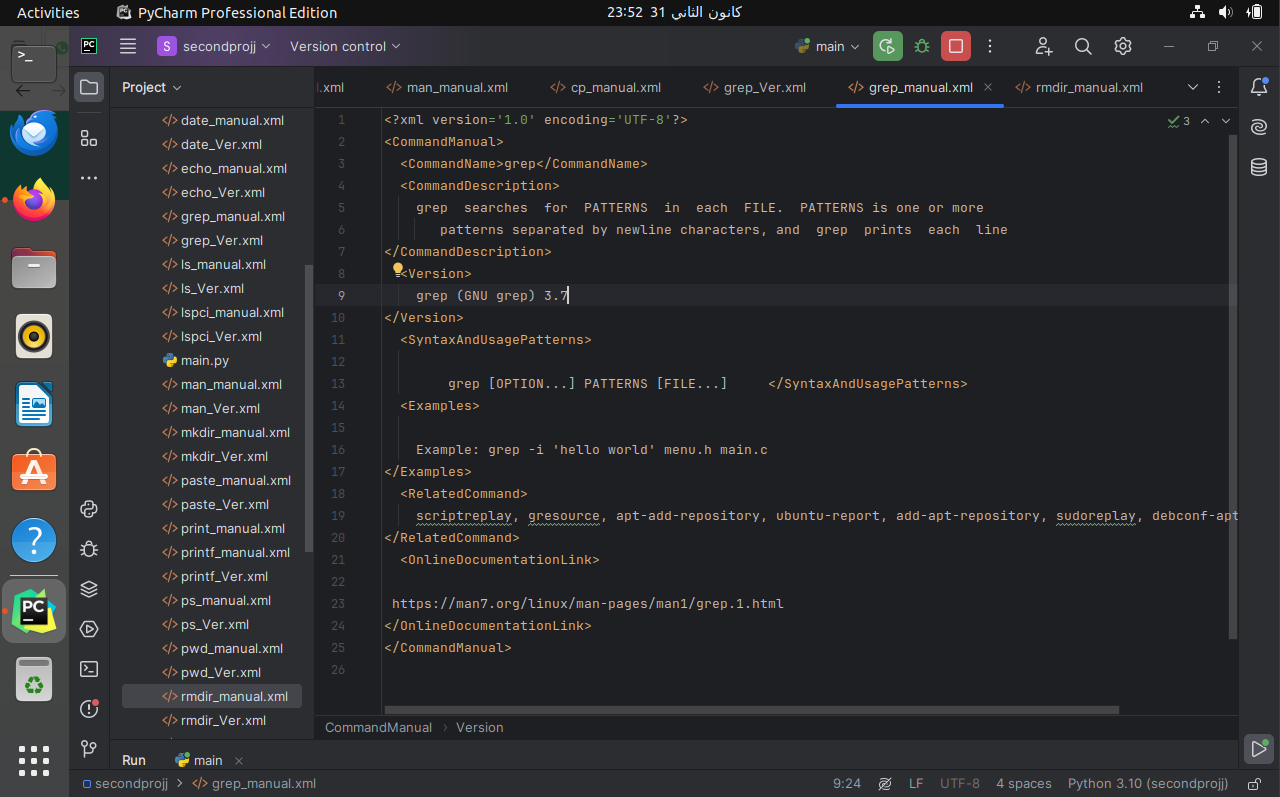


As we can see the manual is generated in Xml file in this format:

cp command:

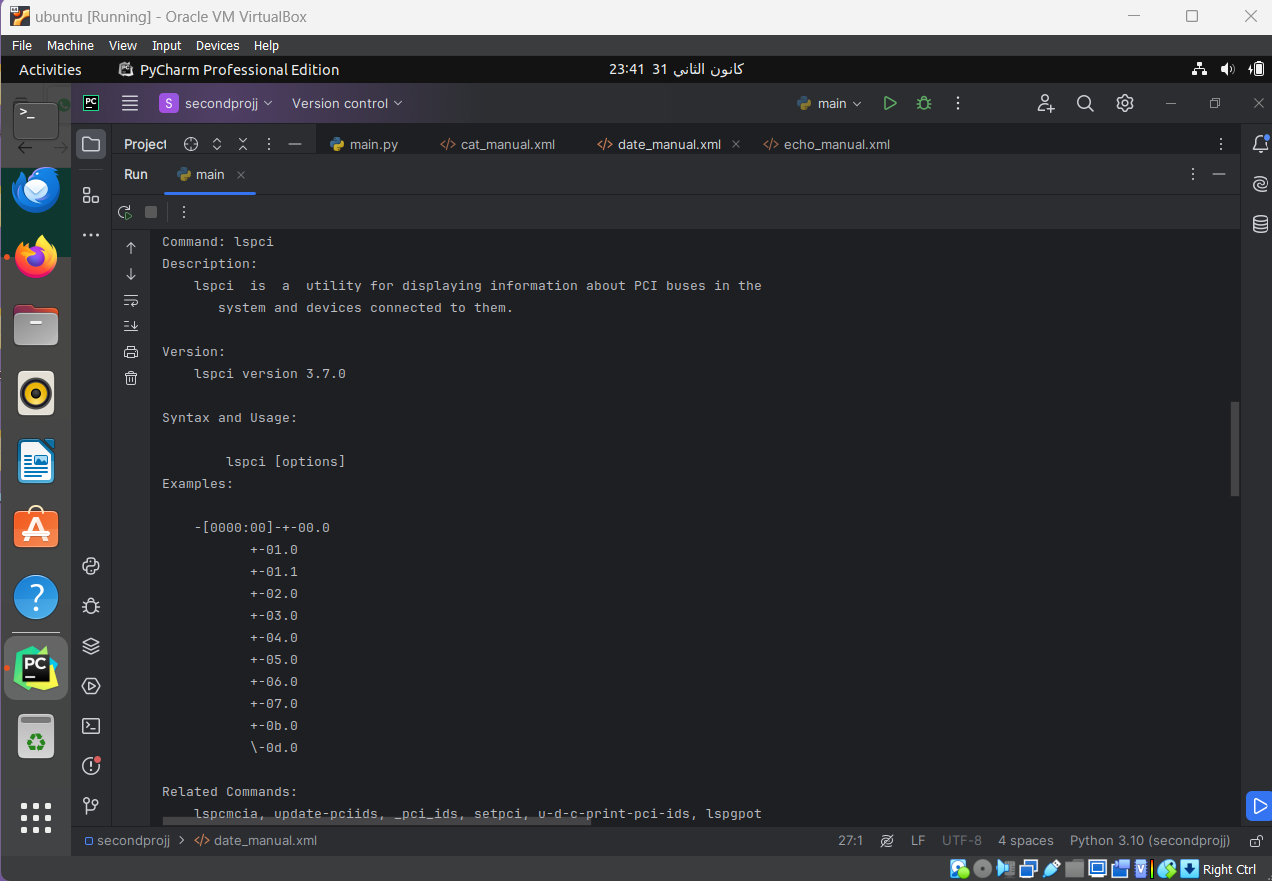


grep command:

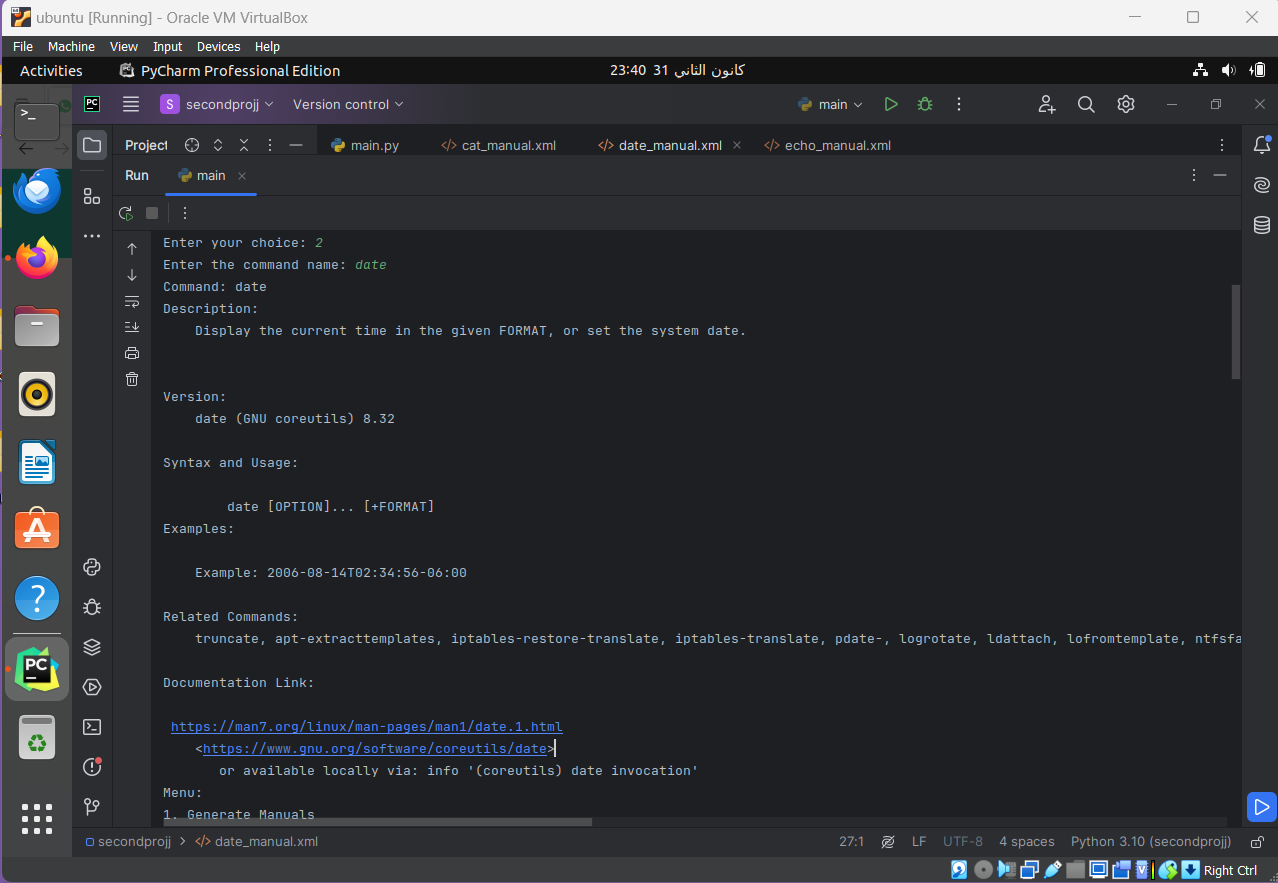


Using option 2 we can see specific command manual :

lspci command for example:

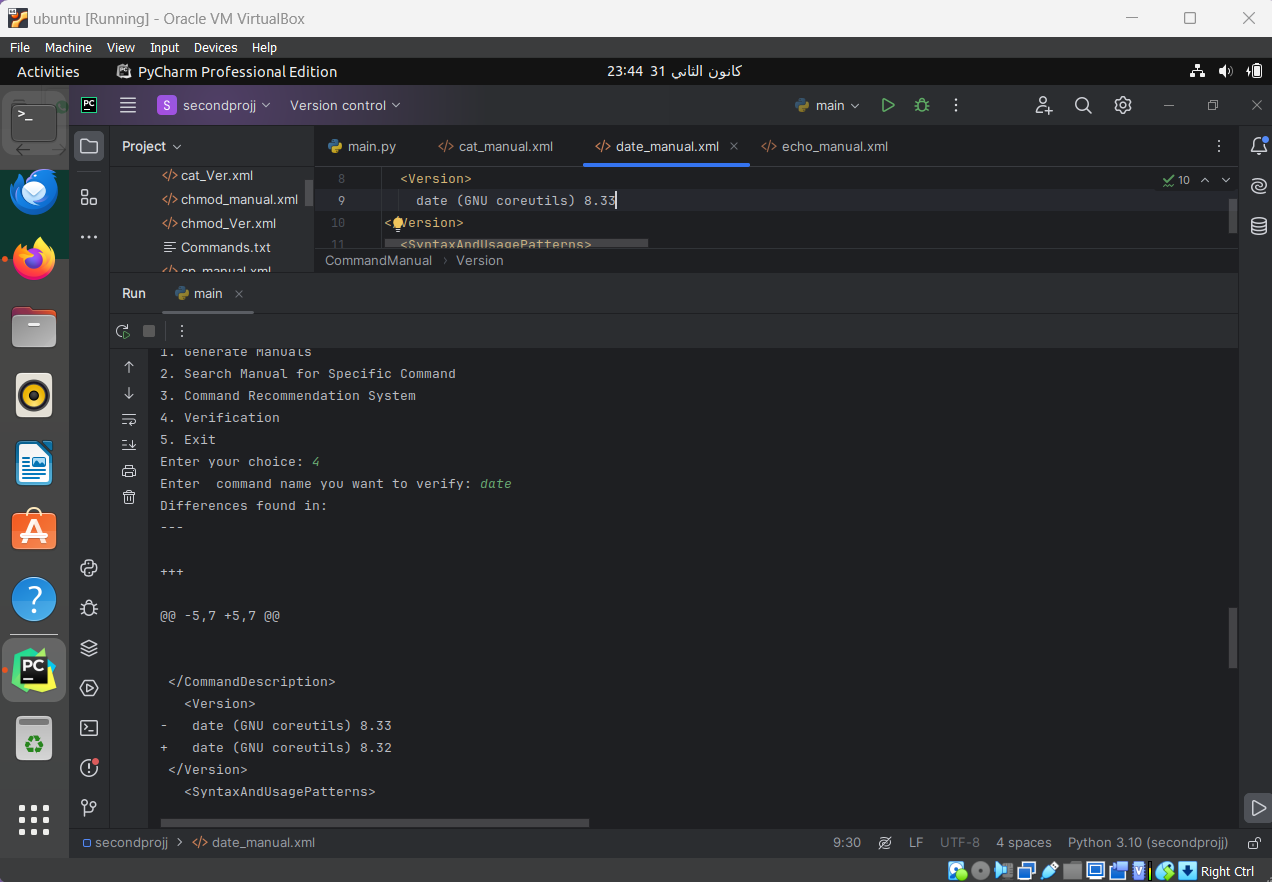


Date command for example



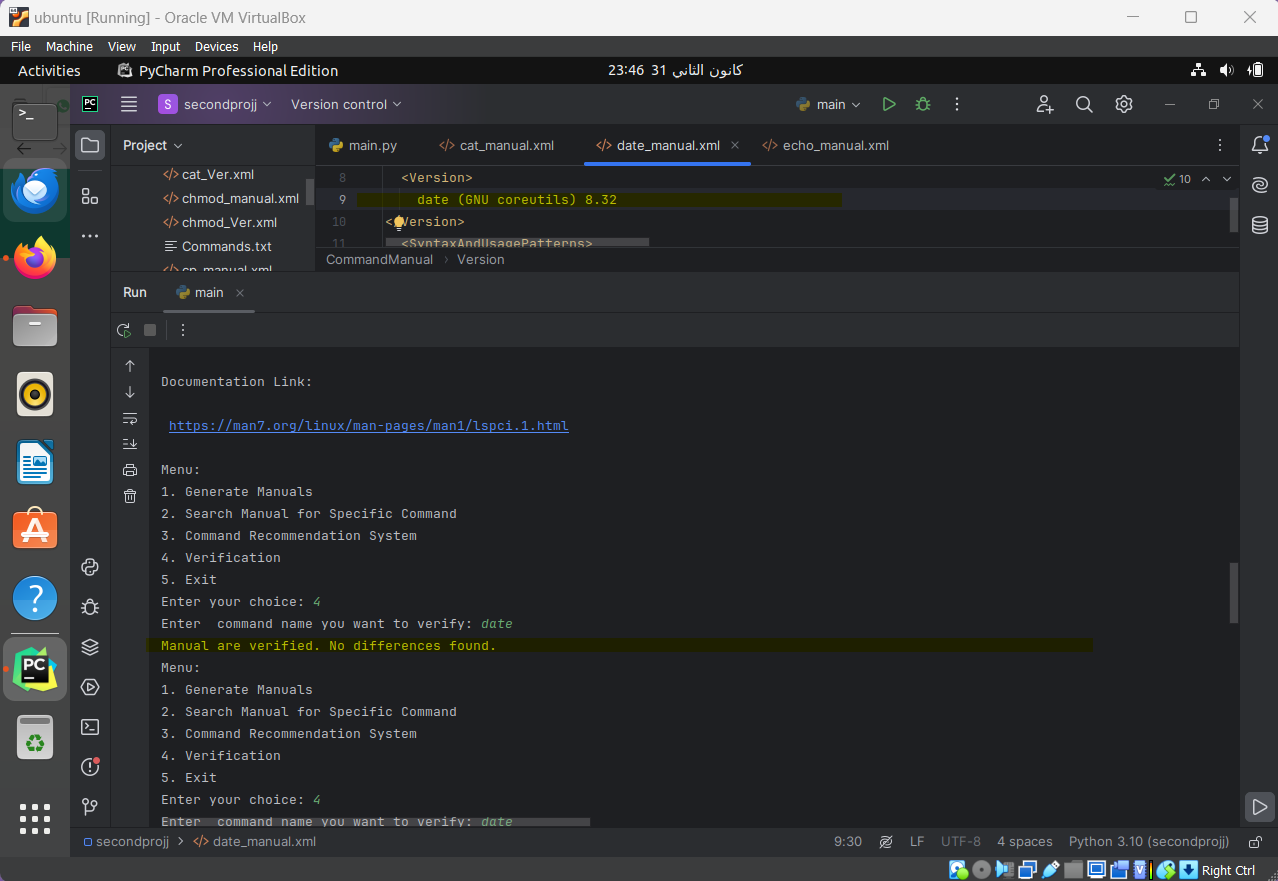
As we can see from the above fig the version is 8.32 I change it to 8.33

Using option 4 we do the verification:



And we notice there is a change for specific in the version of the command

After we regenerate the manual the command now verified :



Now for option 3: recomandation

