
Linux for OSINT. 21-day course for beginners
@cyb_detective 28 December 2023

While working on this course, I called a friend and asked:

- Remember six months ago when you said you started reading a book about using Kali Linux for forensics?

- Yes.

- How much have you read?

- I stopped at chapter 5.

- What's it called?

- Installing Kali Linux

After our conversation ended, I opened the file of this book and cut out all the chapters that require the use of Linux Virtual Machines. To give the reader no reason to stop, for the first 20 days, we will work in a browser only. It's not until day 21 that we think about whether it's time to install Linux.

Table of contents

Day 0. Getting ready to work

Day 1. Basic skills of working with files and directories in command line

Day 2. Basic bash script syntax

Day 3. Install and run utilities

Day 4. Batch file processing

Day 5. Downloading files and working with APIs. Curl.

Day 6. Search in files. Grep

Day 7. Sed and Awk

Day 8. Vim text editor

Day 9. Screen and Crone

Day 10. Text analyzing and editing utilities

Day 11. Video, audio and images

Who Is This Course For?

The course is primarily intended for those **who are professionally involved in or simply interested in OSINT**. And you will find in it a lot about automation of collection and analysis of various data.

On my Twitter (https://twitter.com/cyb_detective) account, I regularly write about useful tools for OSINT, including various Linux utilities. And many times readers have asked, "Cool! Is it possible to run this on Windows?".

Yes, there are different methods of running Linux utilities on Windows. But they are more complicated than running Linux on a VM (virtual machine), VPS (virtual private server) or in an online development environment.

This short tutorial is created to show you clearly that:

- Linux and the command line are very easy to use. Really easy. Extremely easy.
- hundreds of tasks (OSINT related and not only) can be simplified and automated with Linux.
- using Linux command line is a real, unparalleled pleasure.

This course is designed **for the total beginners**. I won't even ask you to install a VM with Linux until the last day, all examples can be tried in a browser. You also won't need any special knowledge. If you know how to use e-mail, you will be able to handle this course. Just relax and follow the instructions carefully.

When reading, keep in mind that this is a short course and so many things are left out. And really important things have been left out. It is designed primarily to show you the capabilities of Linux and to show you "which way to look" when faced with different tasks.

Who should avoid this course?

Linux is good because you can solve the same problem in many different ways. I am not sure that the ways described in this course are the best and fastest ways to solve problems. They are written to visually show you the magic of the command line and stimulate your inner inventor (but all examples work and solve the stated problems).

I would advise the following categories of readers to treat this book as critically as possible and certainly read other Linux books:

- students who are preparing for exams
- IT specialists who are preparing for Linux-related job interviews
- people who want to become real Linux experts
- pentesters and bug hunters

This book is primarily about OSINT.

Day 0. Getting ready to work

Eric Raymond, in his article “How to become a hacker” (1 December 1997, https://cs.fit.edu/~wds/classes/cdc/Readings/BecomeAHacker.pdf), recommended going to a local Linuxoid meeting and asking to burn discs of some distro in exchange for beer (I found this article, but link "where to get Linux" no longer working now).

In the last 26 years progress has gone a long way and I'll just ask you to open a new tab in your browser.

Gitpod is a service that provides online development environments based on the Linux distribution Ubuntu (https://ubuntu.com). You can use it for 50 hours a month for free. This is definitely enough time for you to take this course **many times**.

You should use a Github (https://github.com/) account for authorization (registration is very fast and also free).

Open this link and create a workspace with standard settings:

https://gitpod.io#https://github.com/cipher387/linux-for-OSINT-21-day

Welcome to Gitpod! Now you're ready to start the first lesson!

If you are already using some Linux distribution, you can simply clone the https://github.com/cipher387/linux-for-OSINT-21-day repository to your computer or server.

If you don't know how to clone repositories yet, I recommend using Gitpod for now. On **Day 19** you will learn how to clone repositories, and on **Day 21** you will think about which Linux distribution is best for you.

Day 1. Basic skills of working with files and directories in command line

If you did not miss **Day 0**, you should now have the command line open and the directory **linux-for-OSINT-21-day** active.

Let's see what folders and files are inside it. Type **ls** at the command line and press **Enter**:

```
ls
```

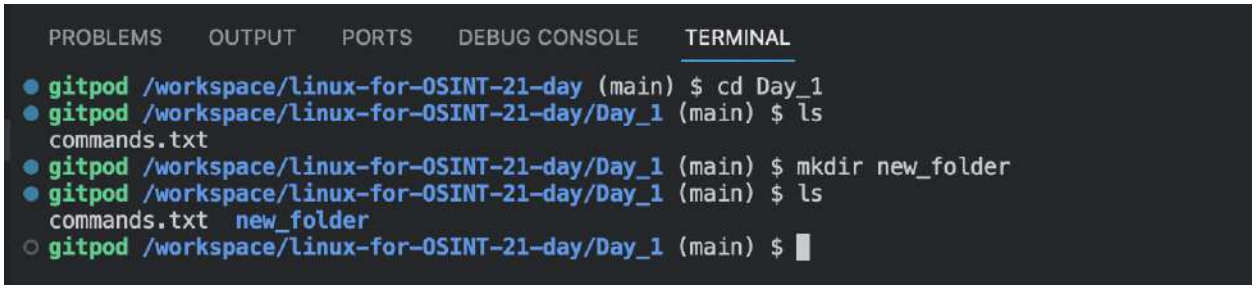
Now let's go to the **Day_1** folder:

```
cd Day_1
ls
```

At the moment there is only a text file with all the commands from this book. Let's add something else to the **Day_1** folder.

```
mkdir new_folder
```

ls



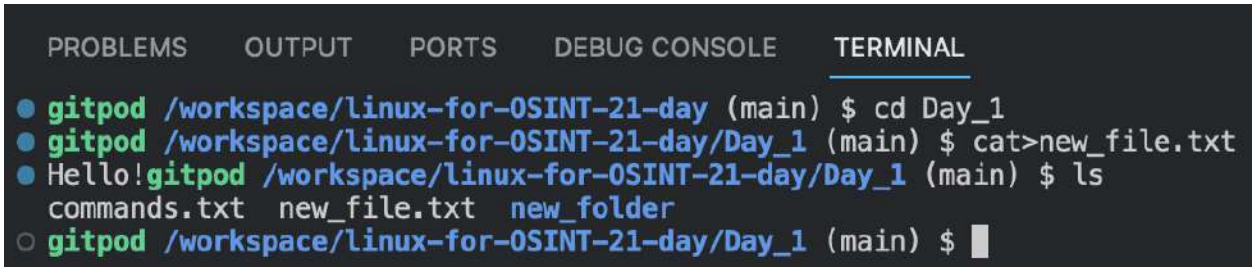
Let's create a new empty text file in this folder.

cat>new_file.txt

Enter some text and press "**Ctrl+D**".

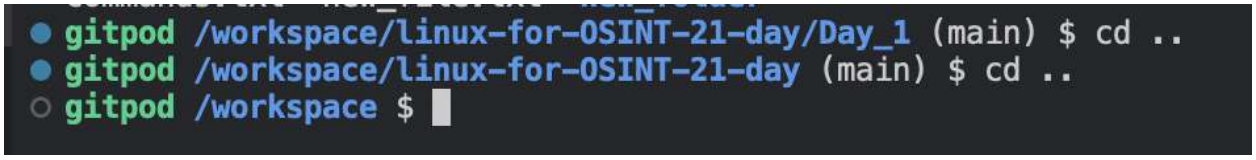
Let's check if the file was created successfully:

ls



To go to the parent directory use this command:

cd ..



Go to the directory located several levels below by specifying the path to it:

cd linux-for-OSINT-21-day/Day_1

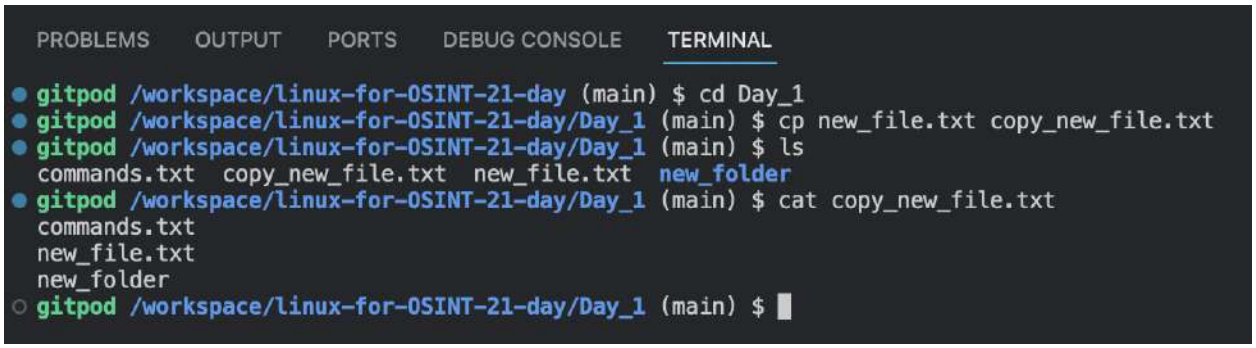
Now let's change the text of the file new_file.txt and display it on the screen (write the results of the ls command to a file):

ls>new_file.txt
cat new_file.txt

Note that this is the second time we have used the > symbol to write the results of a command to a file. Pay special attention to this.

If you just want to copy text from one file to another, use the **cp** command:

cp new_file.txt copy_new_file.txt



Directories can be copied in the same way:

cp -r new_folder copy_new_folder

```
gitpod /workspace/linux-for-OSINT-21-day/Day_1 (main) $ cp -r new_folder copy_new_folder
gitpod /workspace/linux-for-OSINT-21-day/Day_1 (main) $ ls
commands.txt  copy_new_file.txt  copy_new_folder  new_file.txt  new_folder
gitpod /workspace/linux-for-OSINT-21-day/Day_1 (main) $
```

Now let's delete the file and directory copies we created:

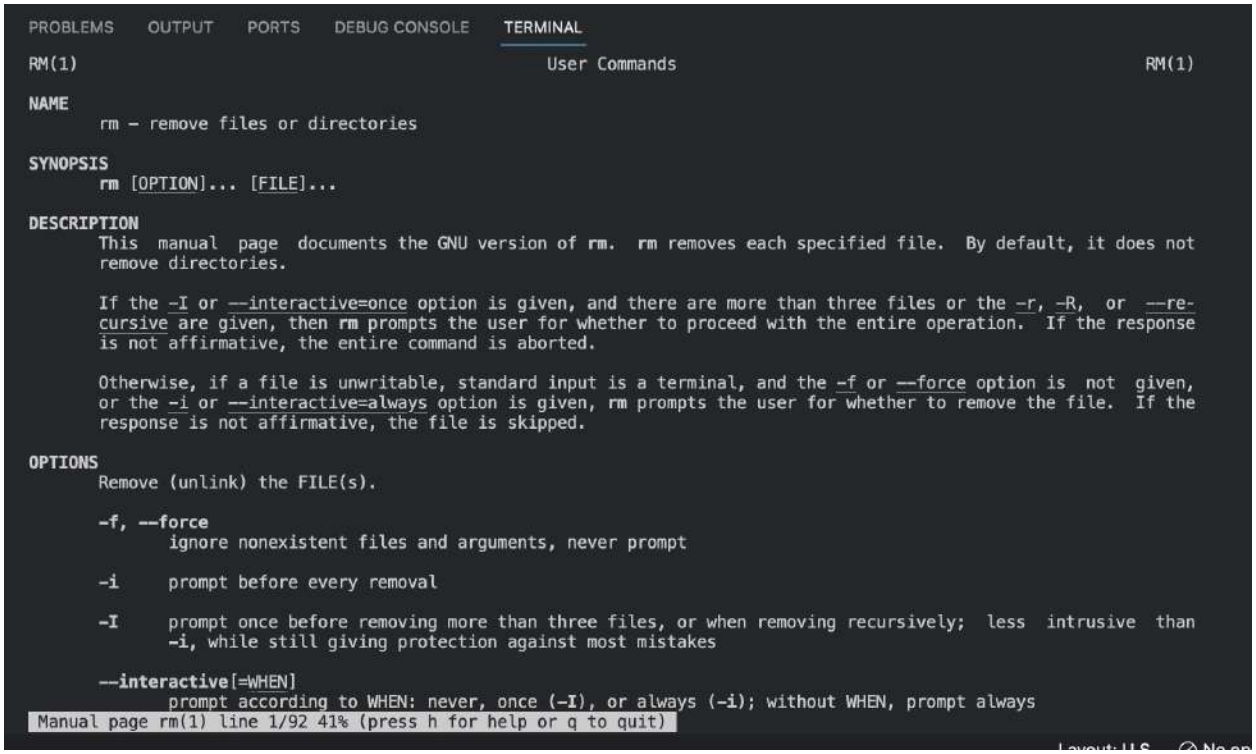
rm copy_new_file.txt
rm -d copy_new_folder

```
gitpod /workspace/linux-for-OSINT-21-day/Day_1 (main) $ ls
commands.txt  copy_new_file.txt  copy_new_folder  new_file.txt  new_folder
gitpod /workspace/linux-for-OSINT-21-day/Day_1 (main) $ rm copy_new_file.txt
gitpod /workspace/linux-for-OSINT-21-day/Day_1 (main) $ ls
commands.txt  copy_new_folder  new_file.txt  new_folder
gitpod /workspace/linux-for-OSINT-21-day/Day_1 (main) $ rm -d copy_new_folder
gitpod /workspace/linux-for-OSINT-21-day/Day_1 (main) $ ls
commands.txt  new_file.txt  new_folder
gitpod /workspace/linux-for-OSINT-21-day/Day_1 (main) $
```

Enough for now. Today we have learned the most basic commands for working with Linux files. And we tried the simplest options to use them. But all utilities mentioned in this lesson (**cat, ls, cd, rm**) have many different options.

To find out all the options of any Linux command use the man (manual) command. For example:

man rm



Press **q** to exit.

If you did something wrong when typing some commands, always remember that you can just create a new workspace, if you are working on Gitpod.

If you are working on your computer, VM or VPS, delete the folder with the files for this course and then copy it again from Github.

Day 2. Basic Bash syntax

What is the most important point of using the command line to perform various tasks instead of the GUI?

The ability to record and repeatedly perform different sequences of actions to automate your work.

Bash (https://www.gnu.org/software/bash/) is a Unix shell and command language written by Brian Fox and first released in 1989. This is what we will use to write commands to automate various tasks in Linux, combining Bash scripts with tools written in other programming languages (Python, Go etc).

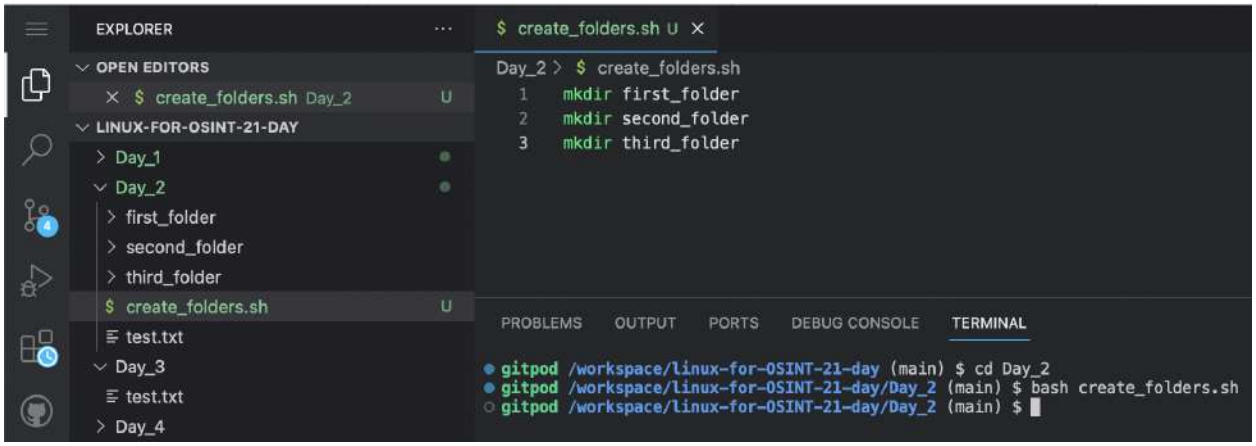
Commands for later execution can be stored in files with the extension **.sh**. Let's take a look at how this works right away.

Go to current lesson folder:

```
cd Day_2
```

And run file create_folders.sh:

```
bash create_folders.sh
```



If you didn't miss yesterday's lesson, you can see that this script simply creates three directories with different names.

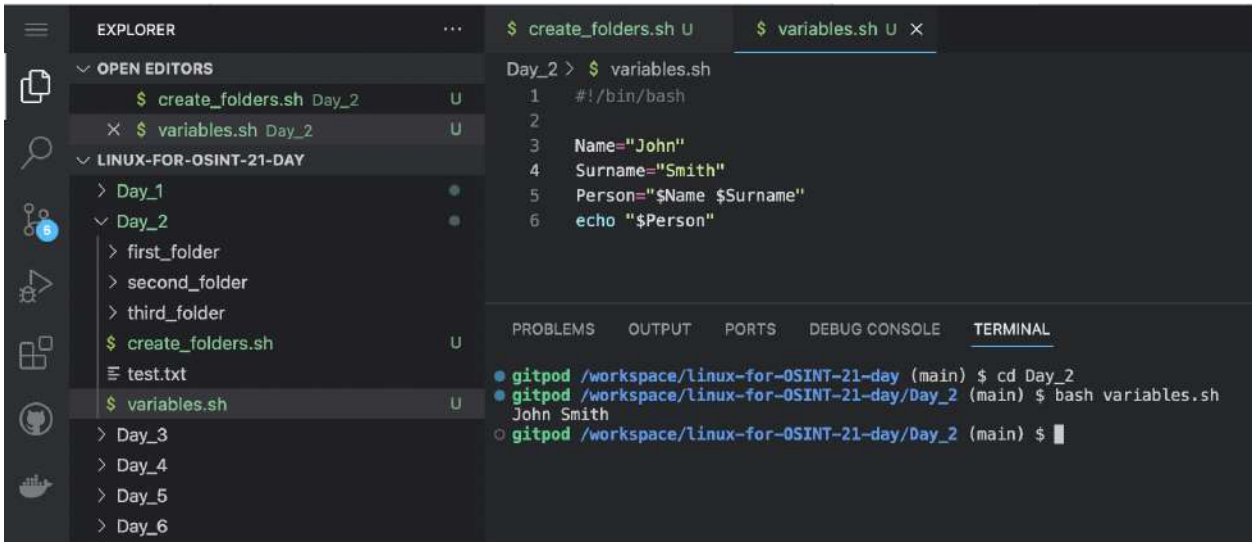
The **.sh** extension is used to store files with code written in the Shell Script language or Bash script language.

Now let's break down the most basic elements of Bash script syntax.

Variables and string concatenation

Run in command line:

bash variables.sh



Comments in the code denote by the symbol #.

In most cases this will not affect the scripts, but I recommend that you indicate on the first line of the .sh files that it is a Bash script:

#!/bin/bash

Create variable contain a string “John”:

Name="John"

Create variable contain a string “Smith”:

```
Surname="Smith"
```

Create variable contain Name variable, space and Surname variable:

```
Person="$Name $Surname"
```

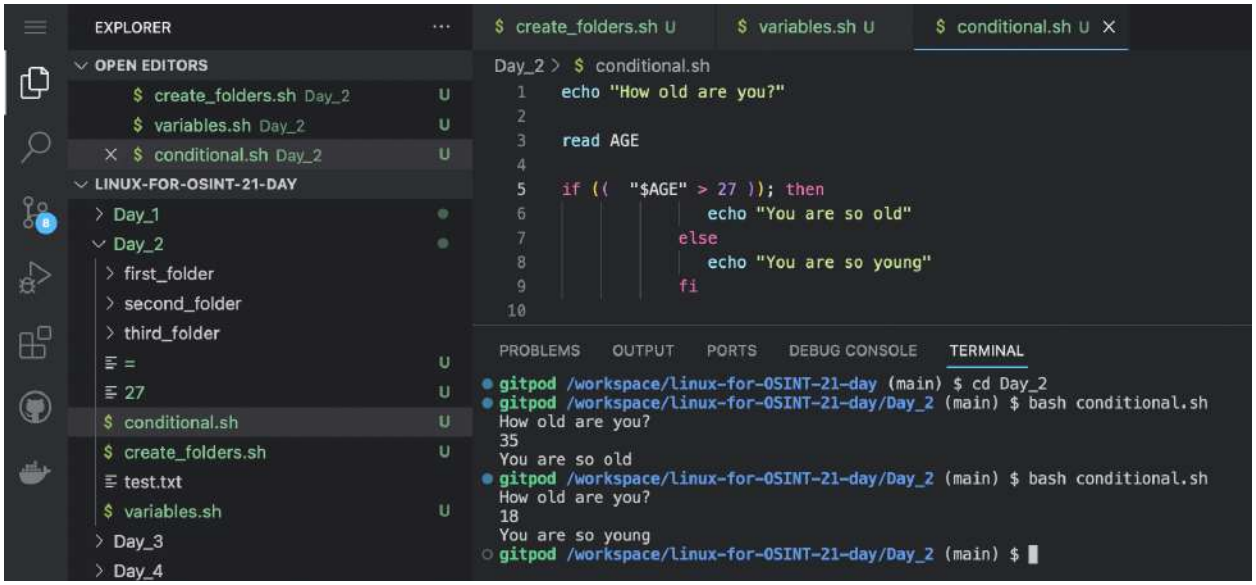
Print Person variable:

```
echo "$Person"
```

Conditional operator

Run **conditional.sh**:

```
cd Day_2
bash conditional.sh
```



Ask the user how old he is:

```
echo "How old are you?"
```

Read the value of the AGE variable entered by the user:

```
read AGE
```

If AGE is greater than 27, we tell the user that he is too old:

```
if ((  "$AGE" > 27 )); then
            echo "You are so old"
```

If AGE is less than 27, we tell the user that he is too young:

```
        else
            echo "You are so young"
        fi
```

Loops

There are three basic kinds of loops in a bash script: “while loop”, “until loop” and “for loop”.

While loop

Run in command line:

```
bash while_loop.sh
```

Assign the value 1 to the variable x:

```
x=1
```

As long as the variable x does not equal 5, perform the following action:

```
while [ $x -le 5 ]
do
```

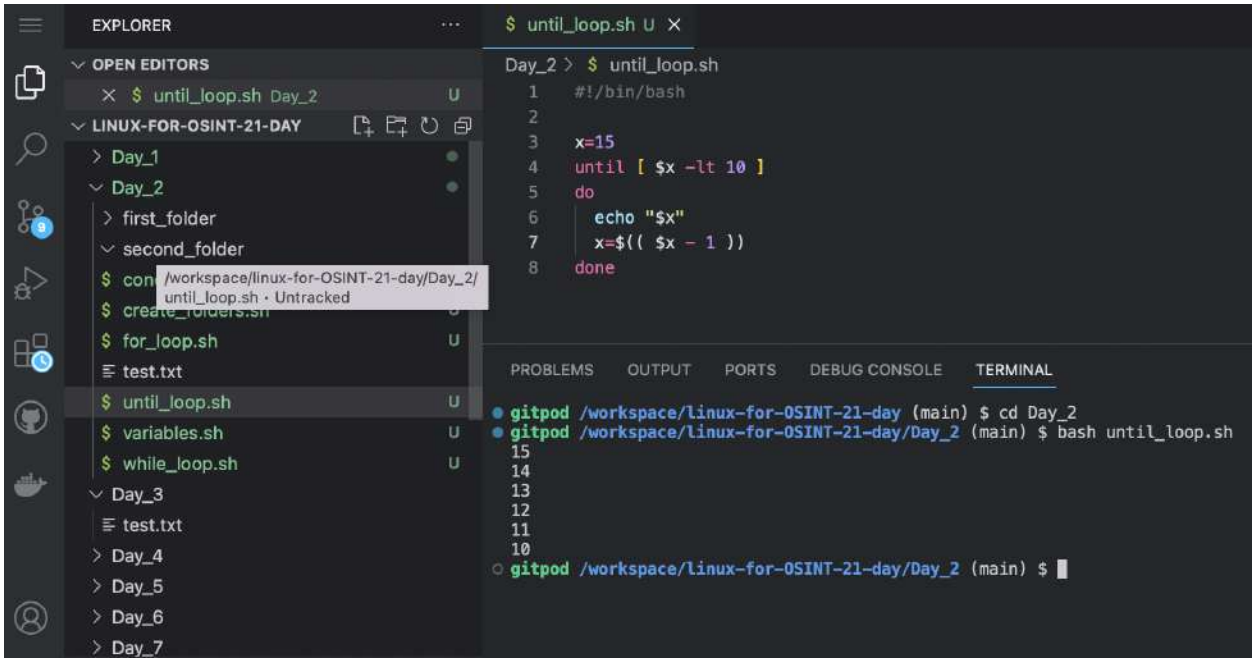
Display the variable x on the screen and then add 1 to it:

```
  echo "$x"
  x=$(( $x + 1 ))
done
```

Until loop

Run:

```
bash until_loop.sh
```



Assign the value 15 to the variable x:

```
x=15
```

As long as the variable x greater than 1-, perform the following action:

```
until [ $x -lt 10 ]
Do
```

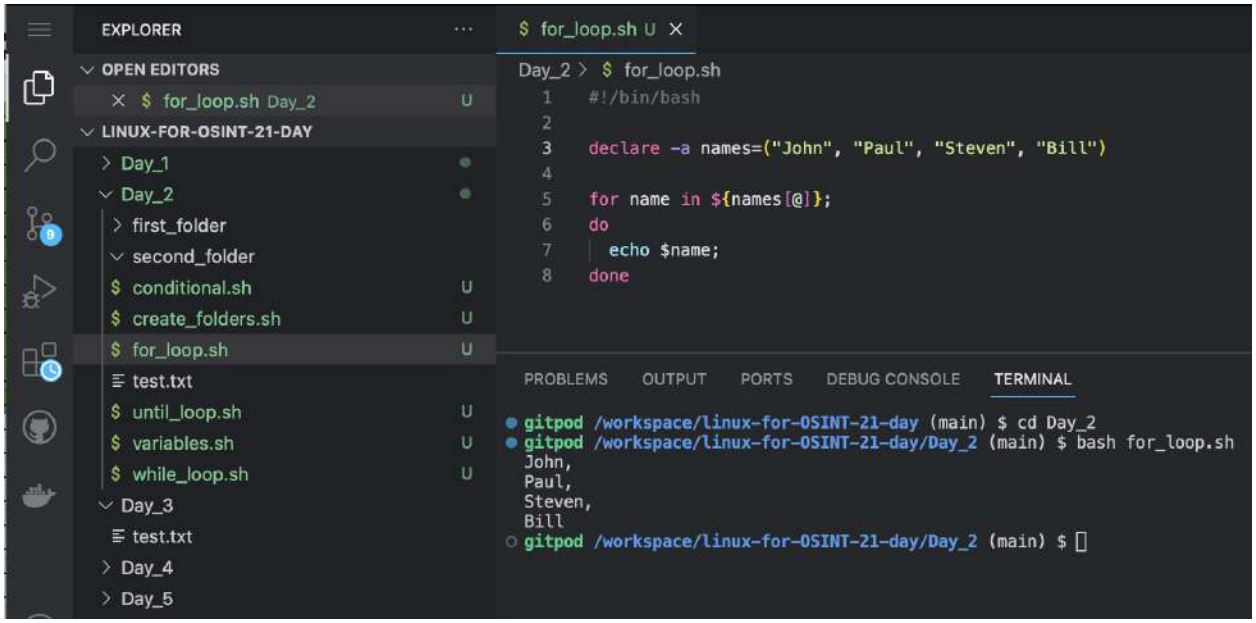
Display the variable x on the screen and then subtract 1 to it:

```
 echo "$x"
 x=$(( $x - 1 ))
done
```

For loop

Run:

```
bash for_loop.sh
```



Create array with list of names:

```
declare -a names=("John", "Paul", "Steven", "Bill")
```

Go through the elements of the list one by one and do the following for each one:

```
for name in ${names[@]};
do
```

Display current array element on the screen:

```
  echo $name;
done
```

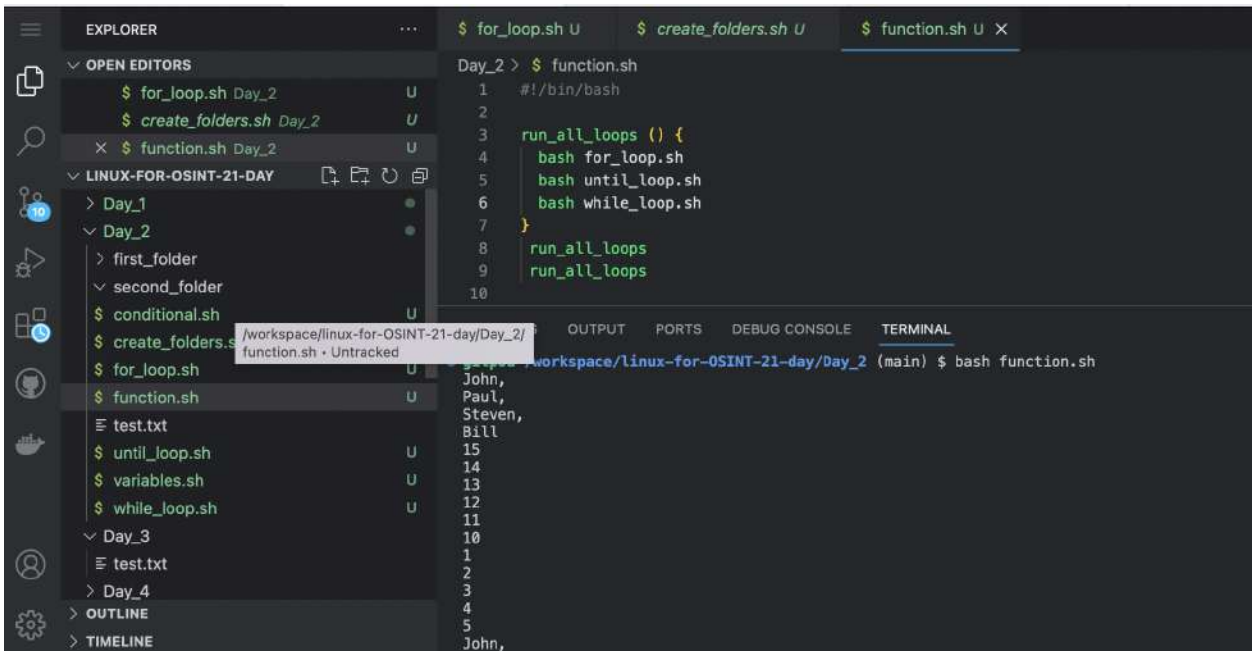
Functions

Like many other programming languages, bash allows you to combine long programs into functions, so that you can then trigger their execution with a short function name.

Let's look at an example.

Run in command line:

```
bash function.sh
```



Create a function that runs scripts to demonstrate how loops work:

```
run_all_loops () {
 bash for_loop.sh
 bash until_loop.sh
 bash while_loop.sh
}
```

Call this function twice:

```
run_all_loops
Run_all_loops
```

As in other programming languages, bash allows you to use arguments in functions and return the values obtained after processing these arguments. But we will not discuss this point in detail, as it is already one of the longest lessons of this course (yes, it will be easier later).

Don't try to learn the basics of bash script syntax right now, we will barely use it in this course (except a little on **Day 4**). This lesson is created primarily to give you an overview of its basic features so that you can stimulate your brain to generate ideas in the future

The most interesting thing about Bash script functions is that you can combine scripts written in different programming languages. For example, combine different Python OSINT tools with a Nuclei scanner written in Go as well as with a wide variety of Linux utilities. Tomorrow we will finally learn how to install them.

Day 3. Install and run utilities

Linux utility is a command that can be run on the command line or on bash scripts. We have already used some of these in past lessons: **cat**, **bash**, **cd**, **man**, **cp**, **mkdir**. These are utilities that are installed by default in every Linux distribution.

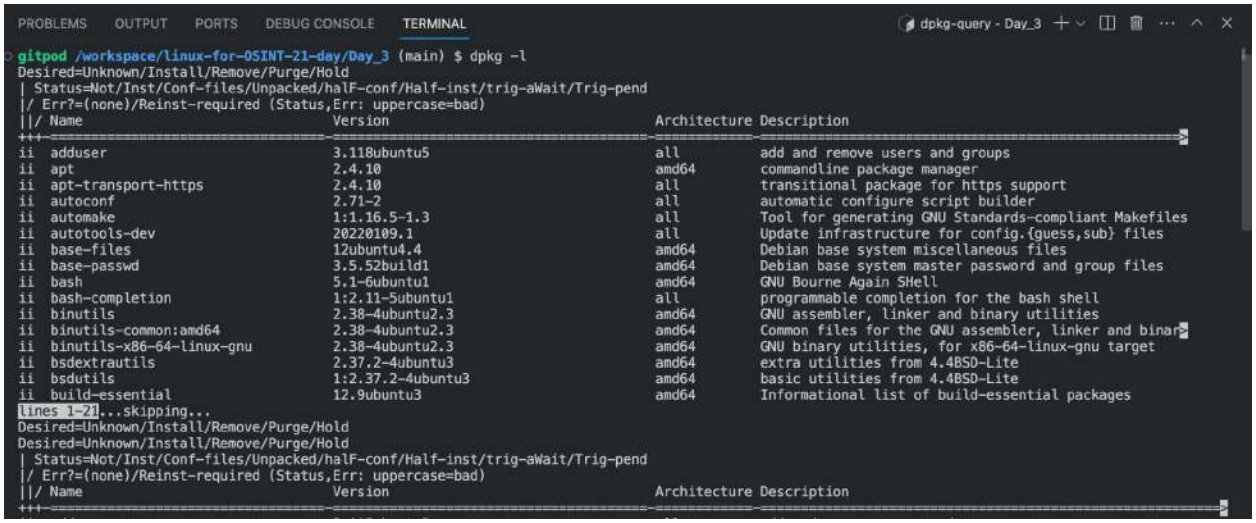
You can see the full list of utilities and applications installed in your own distribution using a special command. Which one depends on the type of your distribution. For example:

Ubuntu, Debian: **dpkg -l**
Fedora, RHEL: **rpm -qa**
OpenBSD, FreeBSD: **pkg_info**
Gentoo: **equery** list or **eix -l**
Arch Linux: **pacman -Q**

If you are using Gitpod, you need to use the commands for Ubuntu.

Run in command line:

dpkg -l



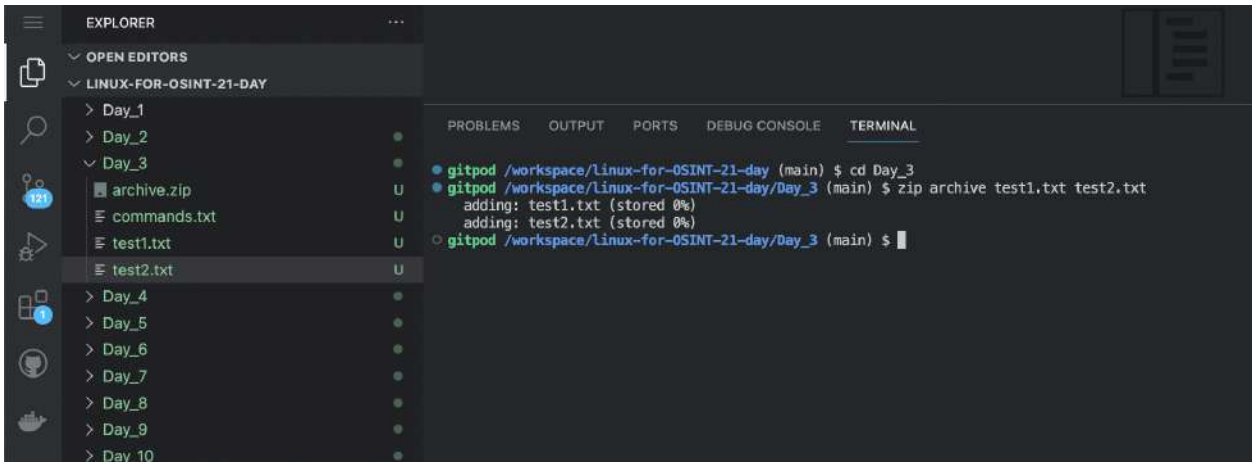
There are also thousands of utilities for a wide variety of purposes that you can install yourself.

As an example, let's install two utilities for working with ZIP archives:

```
sudo apt install zip unzip
```

In this example it is not necessary (we could just use **apt install**), but we add the word **sudo** before the command.

This way we run the command with superuser privileges (maximum permissions to write, read and delete files). In Gitpod, you can use it just like that, but if you are working on your own computer and are not logged in as an administrator, you may have to enter a password.



Now let's try to convert the file into a zip archive:

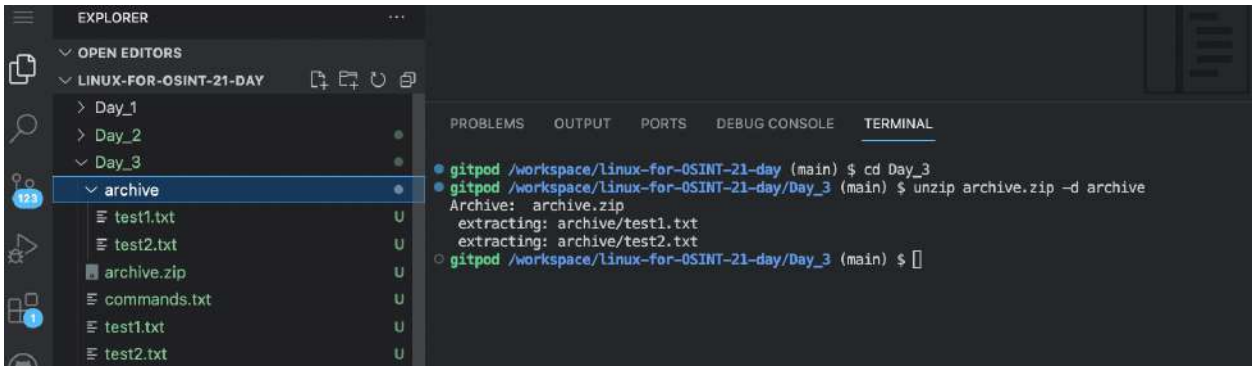
```
cd Day_3
zip archive test1.txt test2.txt
```

Instead of file names, we could specify directory names or put all txt files from the active directory into the archive using *.txt instead of listing filenames.

```
gitpod /workspace/linux-for-OSINT-21-day/Day_3 (main) $ zip -sf archive.zip
Archive contains:
  test1.txt
  test2.txt
Total 2 entries (0 bytes)
gitpod /workspace/linux-for-OSINT-21-day/Day_3 (main) $
```

Now let's see what files are inside the archive using **-sf** flag:

zip -sf archive.zip



You can unzip the archive to a specific folder using the unzip command:

unzip archive.zip -d archive

You can use the **rar** and **unrar** utilities to work with WinRAR archives.

In the example above, we used apt package manager (https://ubuntu.com/server/docs/package-management), which is installed by default in Ubuntu. But there are other package managers that can be used both in Ubuntu and other distributions. Few examples:

PyPi (https://pypi.org) - Python Package Index.

pip3 install ddgr

Snap Store (https://snapcraft.io/store) - software store developed by Canonical.

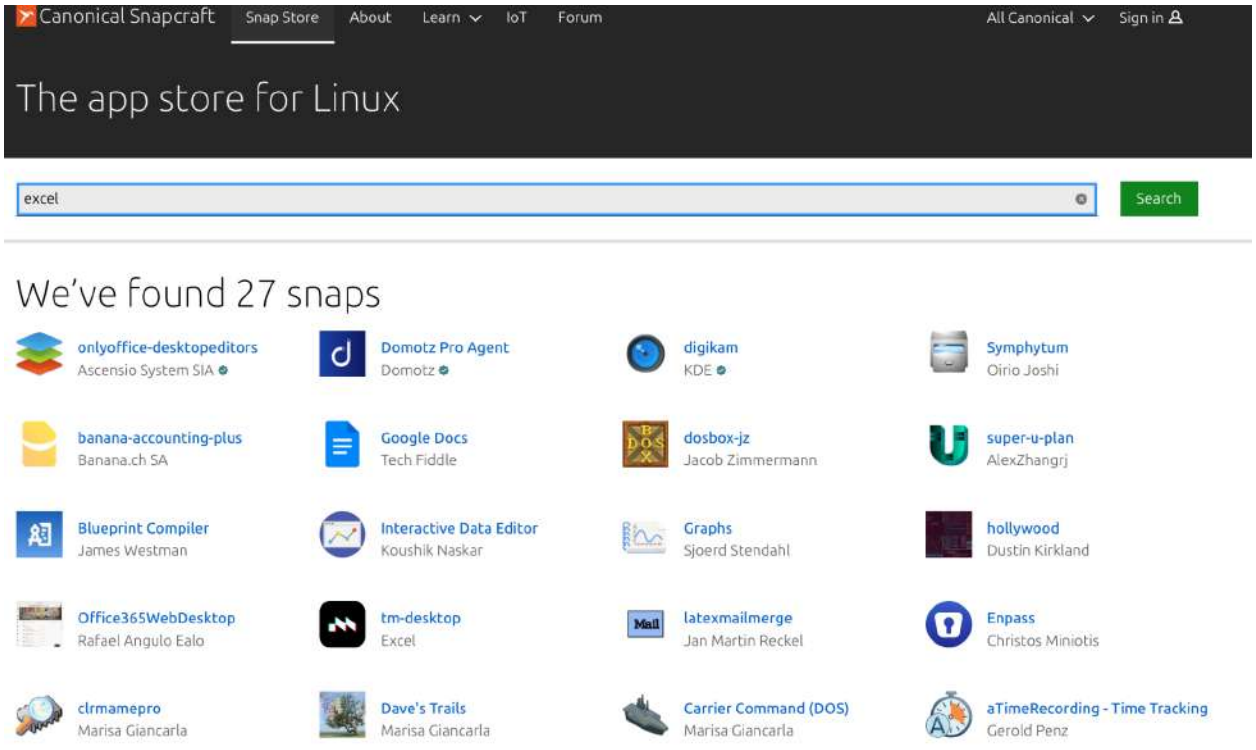
snap install ddgr

Cast (https://sourcemage.org) - package manager for Source Mage Linux distribution.

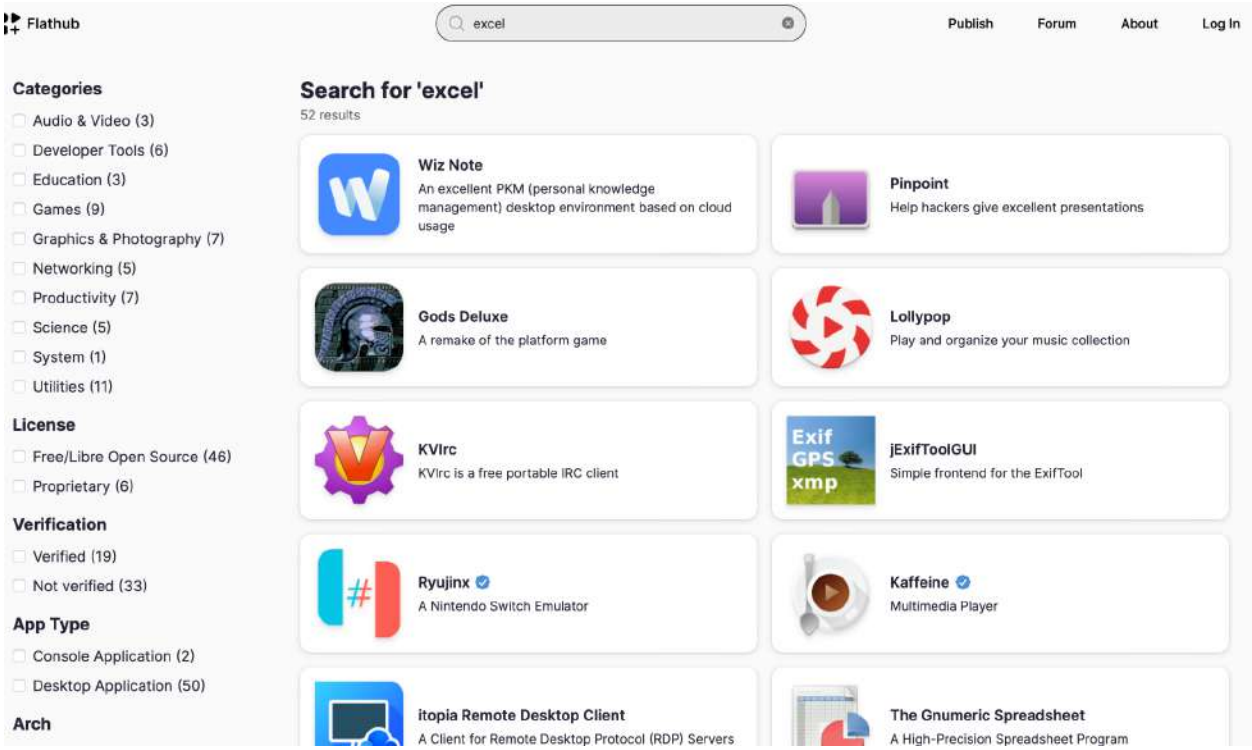
cast ddgr

* (for each package manager, a sample command for installing the ddgr utility is listed, but you do not need to run them. We will install it on **Day 16** of this course)

In this course, we focus on working with the command line. Well for Linux there are a huge number of GUI applications. They are very easy to install through the app stores.



Snap is a software packaging and deployment system developed by Canonical. Snapcraft (http://snapcraft.io) contain thousands of snaps (apps) for 41 Linux distributions.



Flathub (http://flathub.org) - another app store that contains more apps. But unlike Snap, which allows you to install command line utilities, Flathub only supports GUI applications.

If you use Gitpod, you won't be able to try them right now. Return to this chapter at **Day 21** after you select a VM to continue using Linux with a graphical interface.

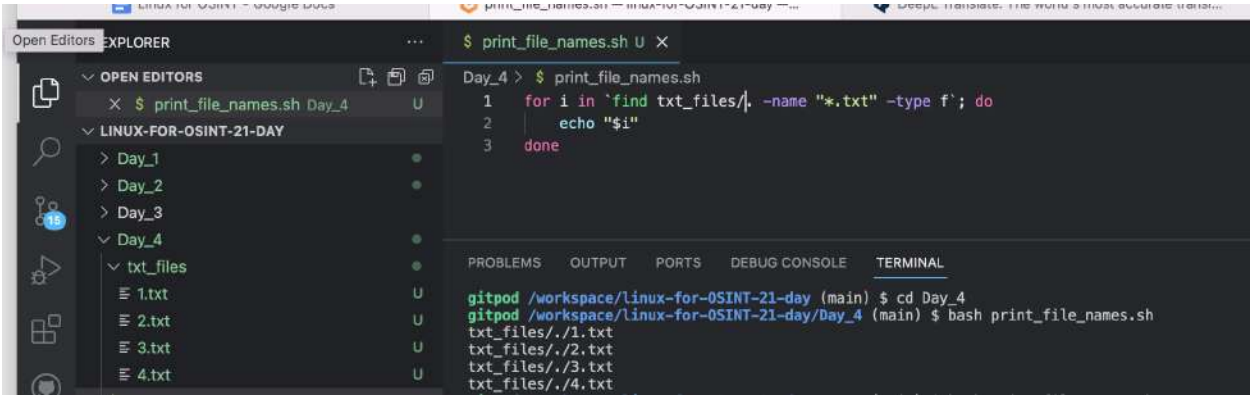
Day 4. Batch file processing

With Bash scripts you can manipulate thousands of files with a few lines of code.

For example, you can perform the same action on all files with a certain extension located in a certain folder. For this we will use the **Find** utility (https://man7.org/linux/man-pages/man1/find.1.html). It is installed in Linux by default.

Please, type in command line:

```
cd Day_4
bash print_file_names.sh
```



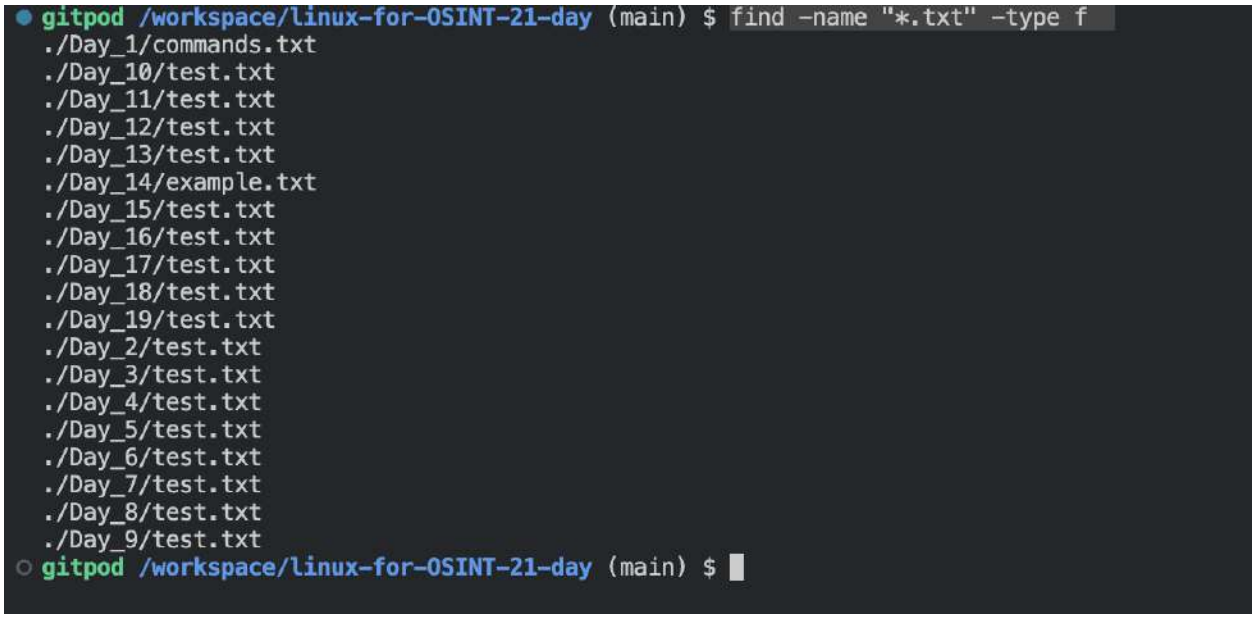
Source code of print_file_names.sh:

```
for i in `find txt_files/. -name "*.txt" -type f`; do
    echo "$i"
done
```

You can replace .txt with any other extension. You can also search for files whose names contain a certain word.

In truth, the script above doesn't make much sense on its own, since the file names are displayed simply by running the find command. Try this:

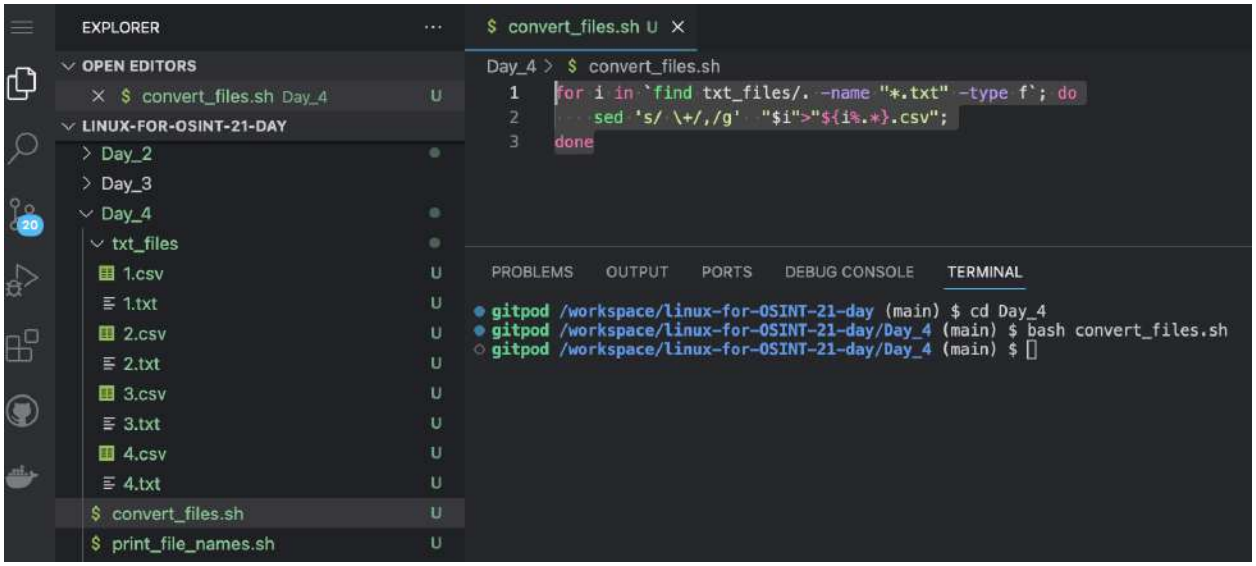
find -name "*.txt" -type f



But by going through the file names one by one, we can perform a certain repetitive action on each file (process file with any Linux utility). Just replace “echo” command with something else.

As example, let's try to convert all txt files to csv format.

Save .txt files as .csv



Enter in command line:

```
bash convert_files.sh
```

Source code of **convert_files.sh**:

```
for i in `find txt_files/. -name "*.txt" -type f`; do
    sed 's/ \+/,/g'  "$i">"${i%.*}.csv";
done
```

Note the sed (https://www.gnu.org/software/sed/manual/sed.html) command. We use to replace spaces with commas and return to it in **Day 7**.

Xargs

Xargs (https://man7.org/linux/man-pages/man1/xargs.1.html) is a utility that allows you to run different Linux commands with certain input parameters. This includes reading these parameters from lists stored in files.

Let's look at examples of how it works.

Let's delete all the archives from the txt_files folder:

find txt_files -name "*.zip" -type f | xargs rm -f

Now let's create archives using Xargs:

cat file_names.txt | xargs -I filenamevar zip archive txt_files/filenamevar

As you can see, in the right part of the command, we read the file line by line using the **Cat** command. And on the left side, we run Xargs with the **-I** flag to use each line as a **filenamevar** variable and use that variable as the filename in the parameters when we run the zip command.

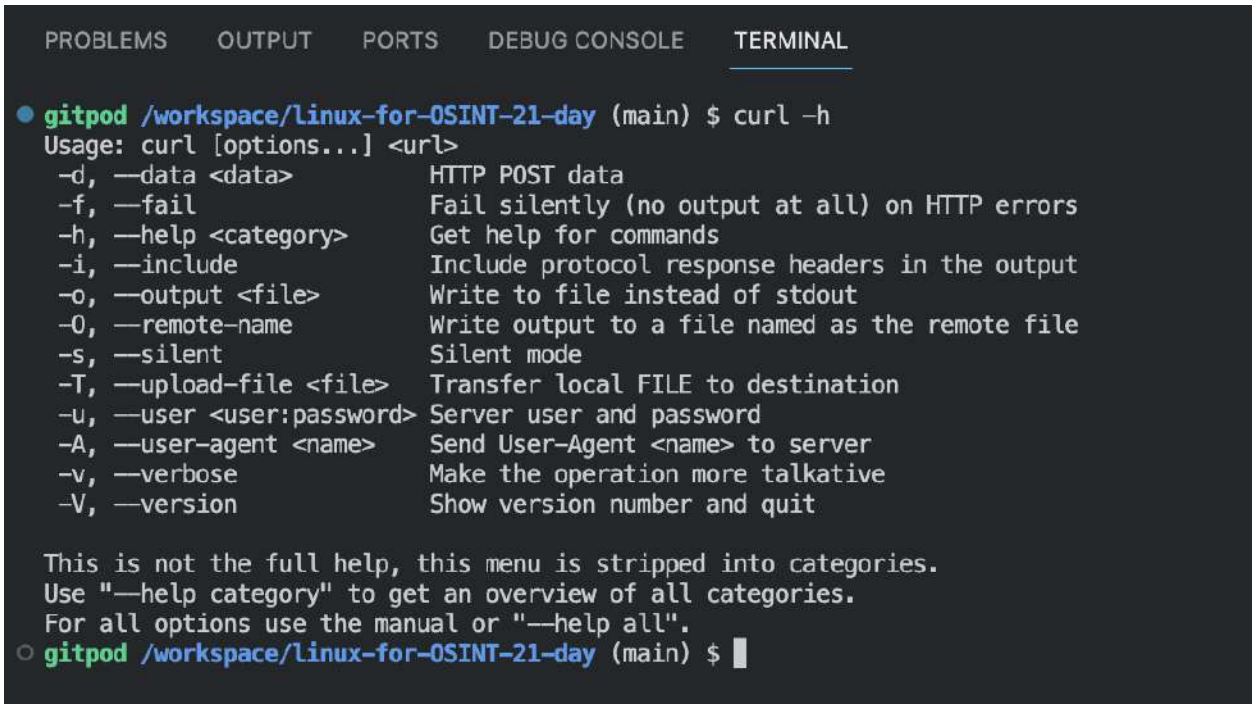
The material on this day can be combined with almost all of the chapters in our course. But I recommend that you pay special attention to some of them:

Day 5 - file downloading.
Day 7 and **Day 10** - text file editing.
Day 11 - working with video, audio and images.
Day 12 - working with PDF.
Day 13 - working with MS Office files.
Day 14 - working with JSON, CSV and XML files.

After reading the lesson in each of these days, I recommend running at least one command in combination with **Xargs**.

Day 5. Downloading files and working with APIs. Curl.

Today we're going to learn one of the most important skills for automating data collection in Linux - downloading files using the **Curl** utility. This comes in handy when processing API requests, scraping websites, downloading data from archives or other sources.



Curl (https://curl.se) Is a tool to transfer data from or to a server, using one of the a lot of supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP). It does not require installation. It is available in Linux by default.

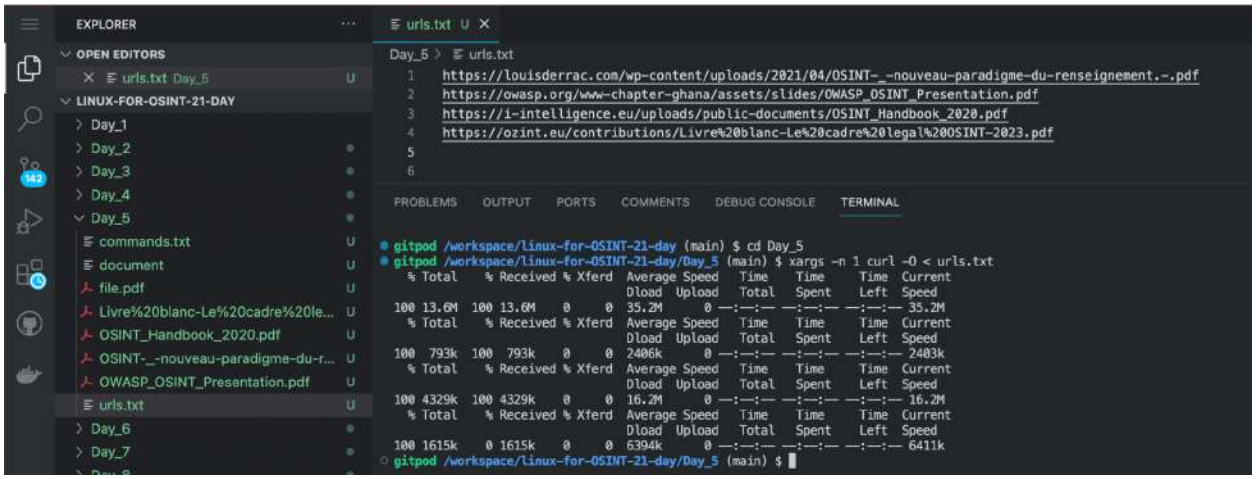
Let's try to download file. Change original name to file.pdf with **-o** flag:

Enter in command line:

Cd Day_5

curl https://eduscol.education.fr/document/3366/download -o file.pdf

Now let's try downloading files from the list of links using the **Xargs** command (we talked about it yesterday).

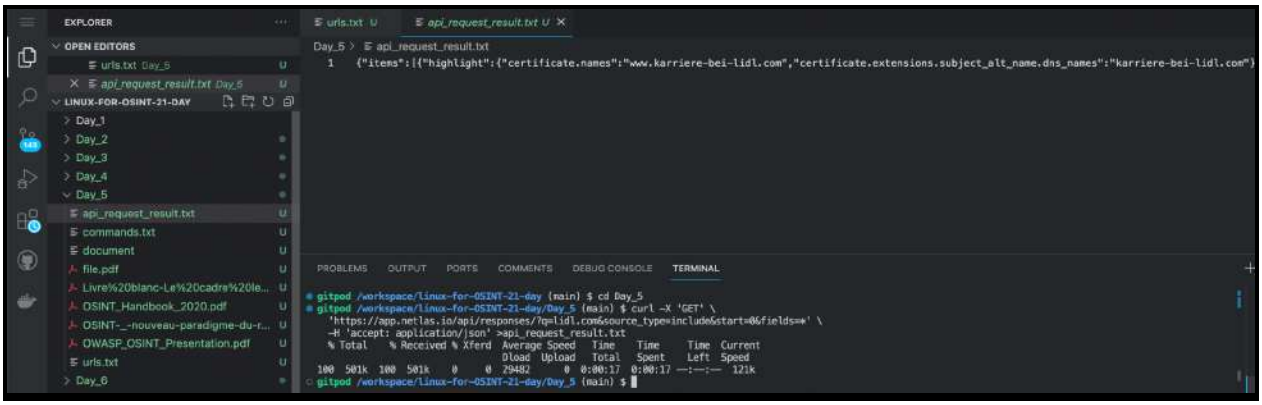


xargs -n 1 curl -O < urls.txt

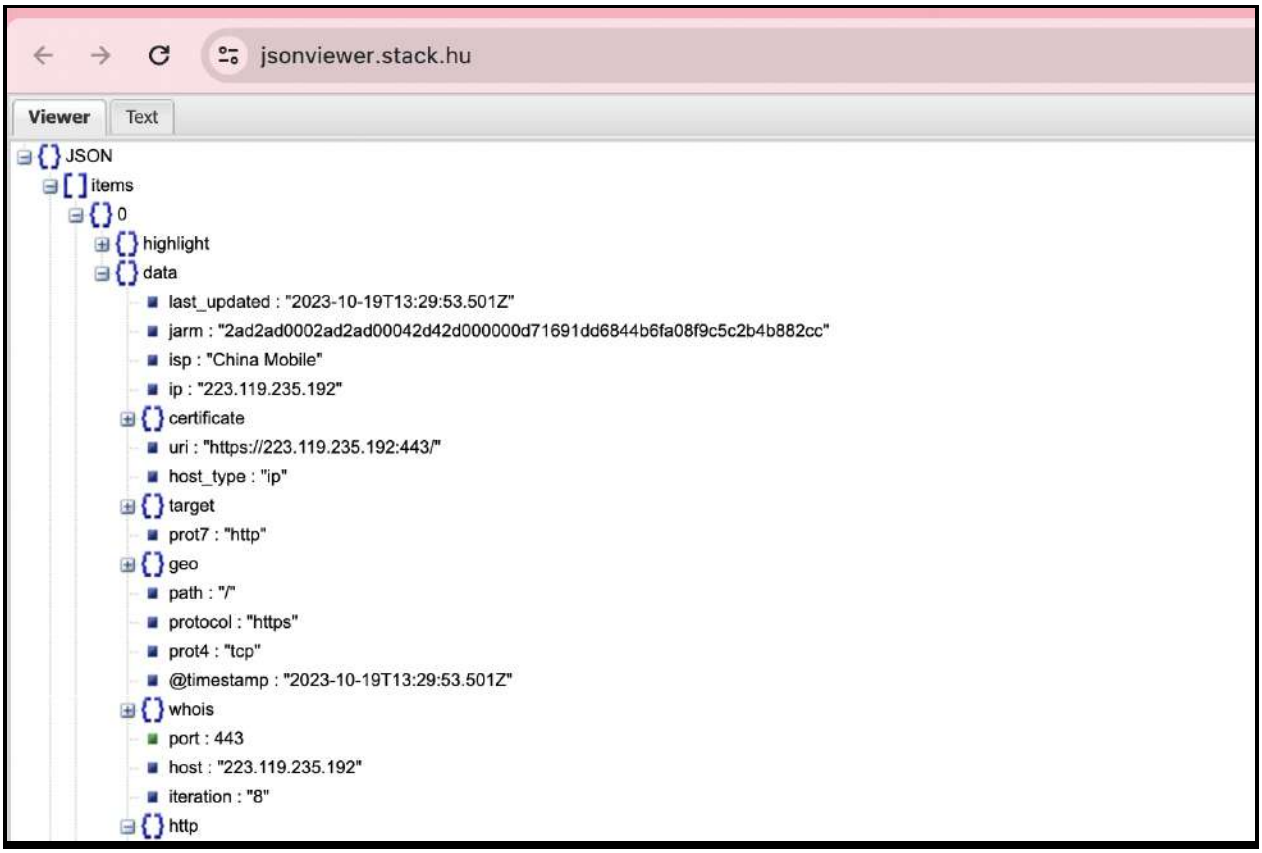
API requests

Using the **-H** flag, you can send requests with different http headers. This function is useful when automating the execution of requests to various APIs.

Simple example for Netlas API (https://github.com/netlas-io/netlas-cookbook):

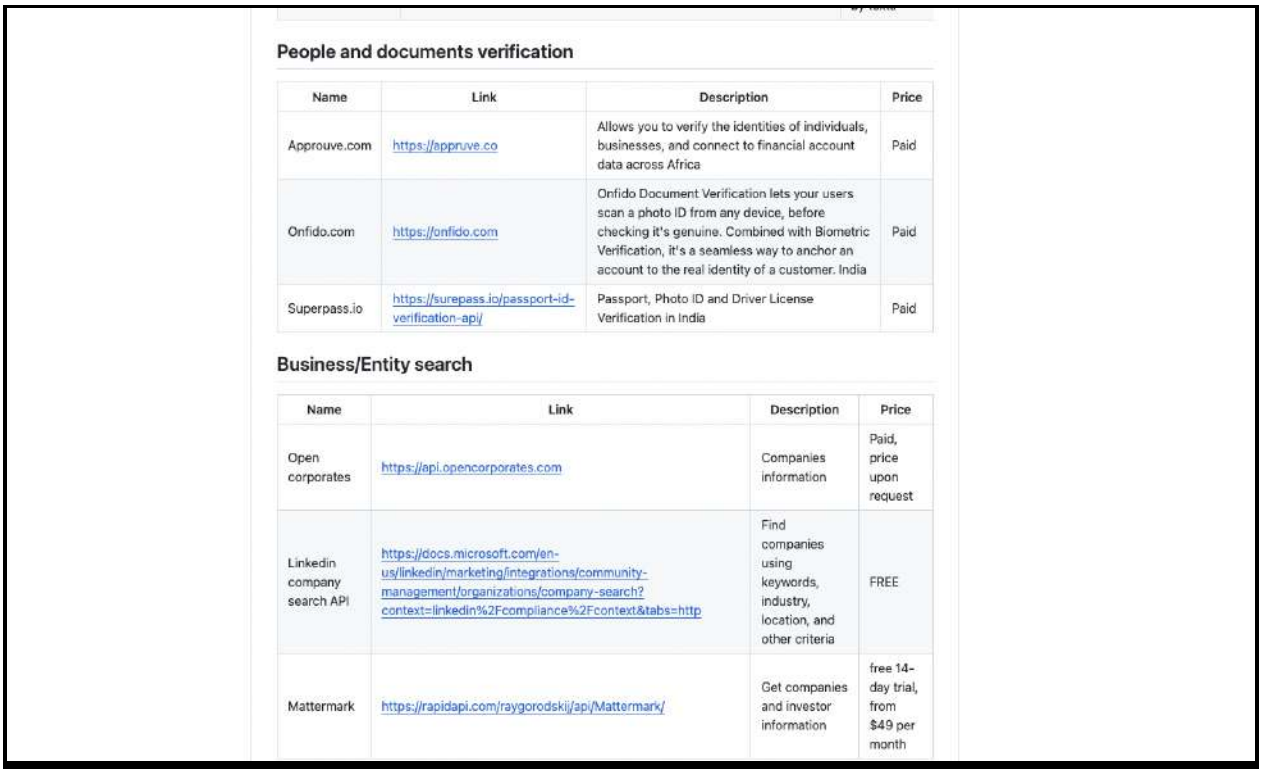


curl -X 'GET' \'https://app.netlas.io/api/responses/?q=lidl.com&source_type=include&start=0&fields=*' \ -H 'accept: application/json' >api_request_result.txt



Using this query, we retrieved a JSON file with detailed domain information found in Netlas IP search engine by query "lidl.com". You can view it in a convenient format using some online JSON file viewer tools (https://jsonviewer.stack.hu/).

On **Day 14**, we'll talk a little more about data in JSON format and learn how to process it using the **jq** utility.



People and documents verification

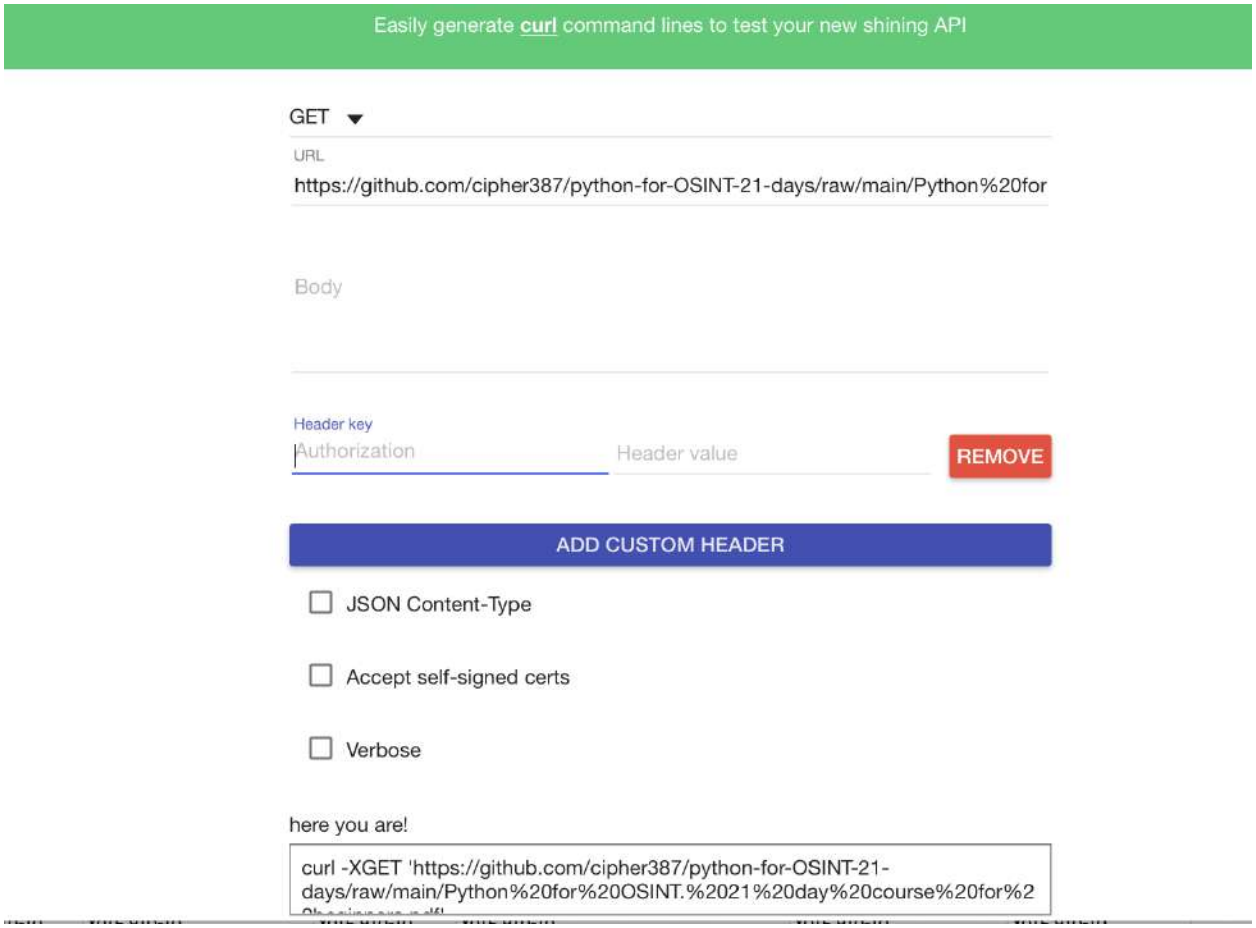
Name	Link	Description	Price
Approuve.com	https://approuve.co	Allows you to verify the identities of individuals, businesses, and connect to financial account data across Africa	Paid
Onfido.com	https://onfido.com	Onfido Document Verification lets your users scan a photo ID from any device, before checking it's genuine. Combined with Biometric Verification, it's a seamless way to anchor an account to the real identity of a customer. India	Paid
Superpass.io	https://surepass.io/passport-id-verification-api/	Passport, Photo ID and Driver License Verification in India	Paid

Business/Entity search

Name	Link	Description	Price
Open corporates	https://api.opencorporates.com	Companies information	Paid, price upon request
Linkedin company search API	https://docs.microsoft.com/en-us/linkedin/marketing/integrations/community-management/organizations/company-search?context=linkedin%2Fcompliance%2Fcontext&tabs=http	Find companies using keywords, industry, location, and other criteria	FREE
Mattermark	https://rapidapi.com/raygorodskij/api/Mattermark/	Get companies and investor information	free 14-day trial, from $49 per month

In addition to the Netlas API, which can be used to collect domain or IP address data, there are hundreds of other different APIs that are useful for OSINT.

They can help you collect information about emails, phone numbers, physical addresses, companies, cryptocurrency wallets, flights, and much more. You can find a list of them in the **APIs for OSINT** repository on Github (https://github.com/cipher387/API-s-for-OSINT).

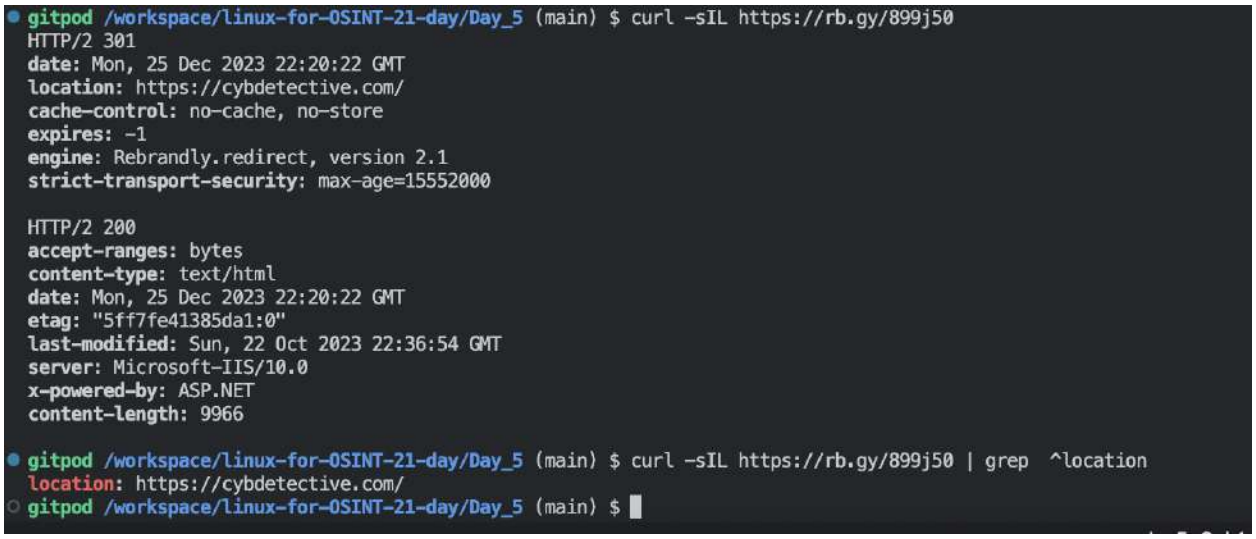


For writing complex queries with header modification, you can use a special online service to generate commands for the **Curl** utility - Curl builder (https://curlbuilder.com/).

Curl can also be used to view information about web pages. Use **-I** flag to send HEAD request and retrieve only the headers of the response (without the body):

curl -I https://sector035.nl

This can be used, for example, to safely open short links:



curl -sIL https://rb.gy/899j50 | grep ^location

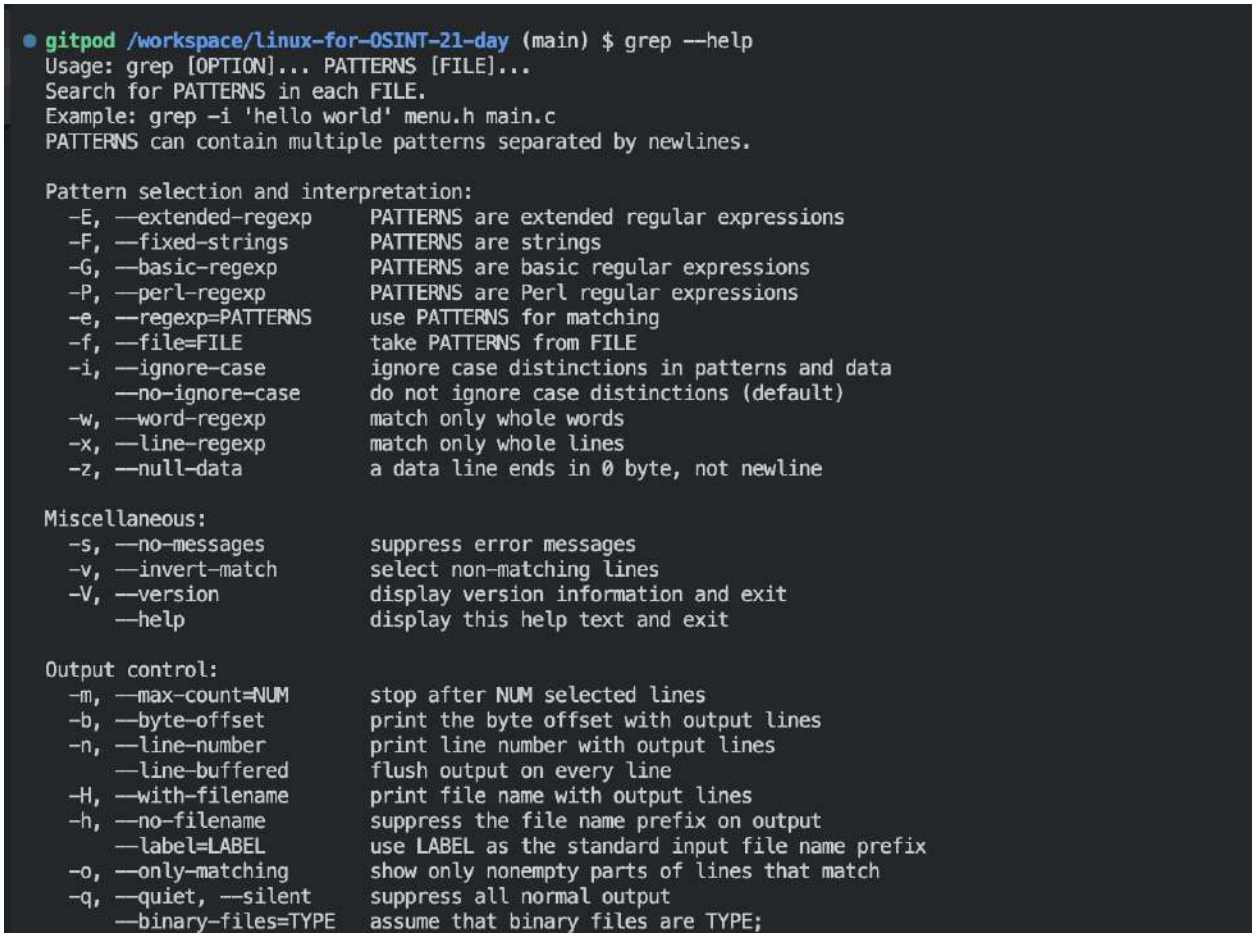
Note that here we are combining **Curl** with the **Grep** command, using the **|** symbol to cut out the string that starts with "location" from the result. Tomorrow we will talk more about this wonderful utility.

Day 6. Search in files. Grep

What can you do next with the downloaded files? Lots of different things. But the most important thing an OSINT specialist should be able to do is automatically search and extract different data.

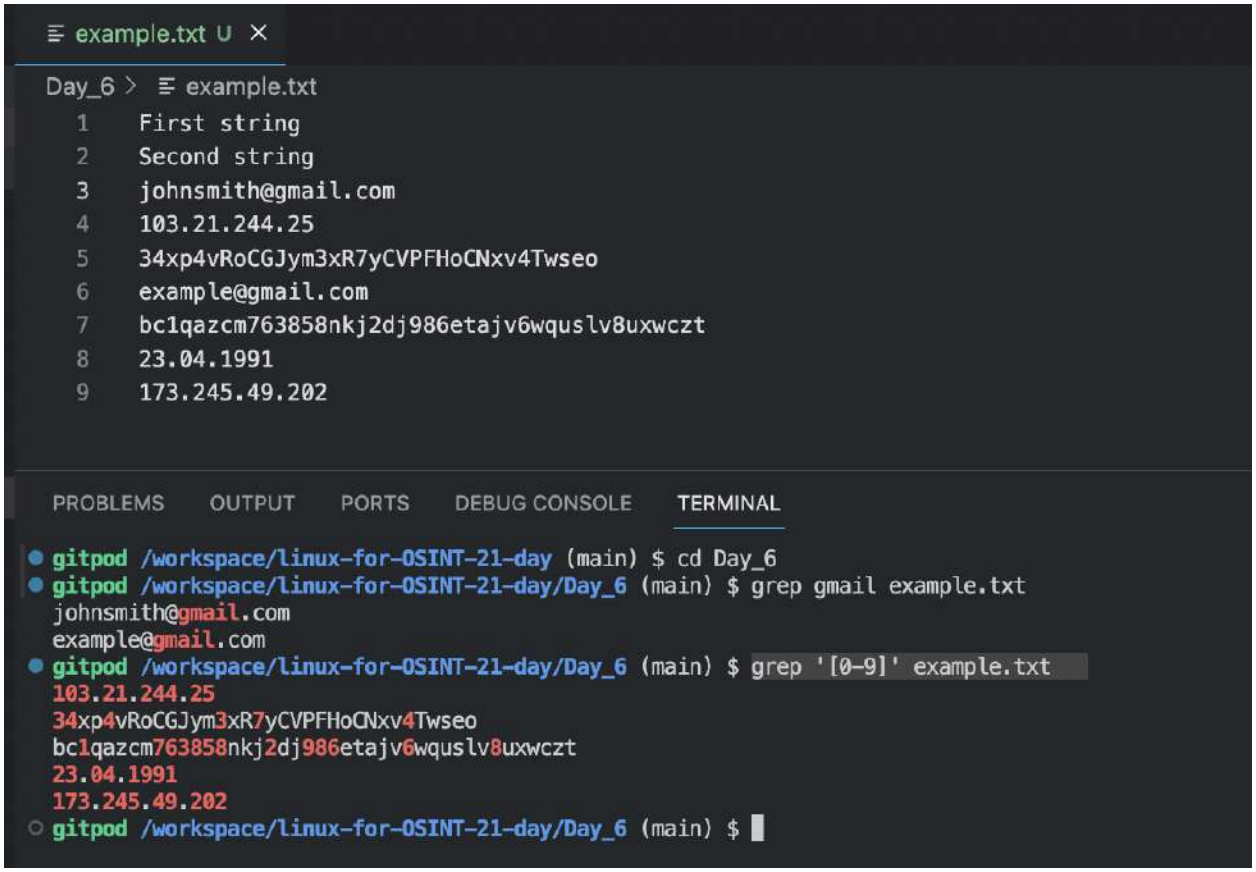
Grep (https://www.gnu.org/software/grep/) is one of the oldest UNIX utilities, released in November 1973 (50 years in 2023!). It allows searching for matches to certain text patterns (regular expressions) in files.

The **Grep** utility does not require installation. It is available in Linux by default.



Run the help command to appreciate just how many options this seemingly simple utility has.

Let's try to find lines in the text file that include "gmail". Use -F flag for search strings (it's optional, you can miss this flag):



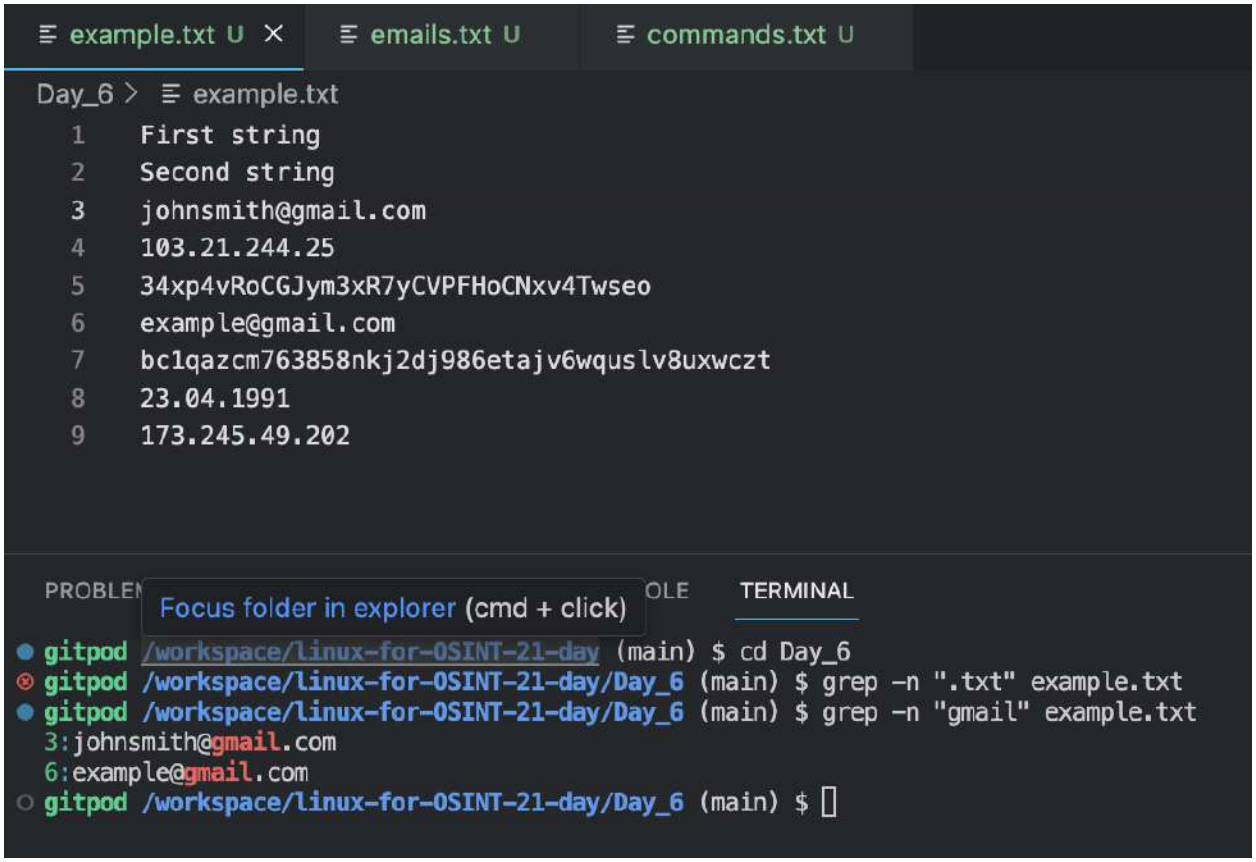
grep -F gmail Day_6/example.txt

By default, the **Grep** utility is case sensitive. To disable this option (search for both "gmail" and "GMAIL", "Gmail") add the -i flag.

grep -F -i gmail Day_6/example.txt

Now let's try to find the lines that contain at least one digit (use simple regular expression [0-9]):

grep '[0-9]' example.txt



If you want to add line number display to the results, use the -n flag:

grep -n "gmail" example.txt

Now the task is a little more difficult. Let's find the strings that contain emails:

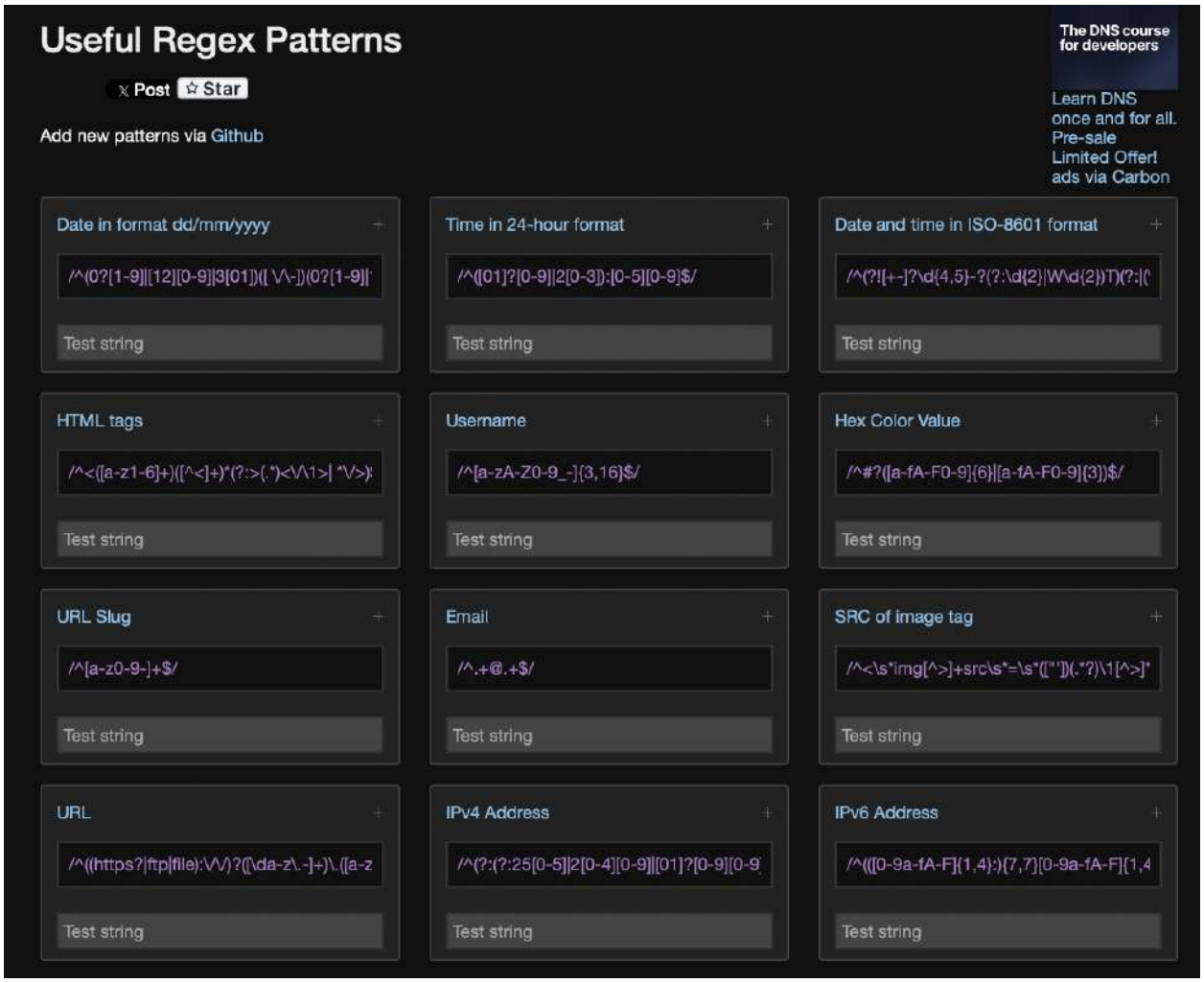
```
grep -E -o -r "[A-Za-z0-9][A-Za-z0-9._%+-]+@[A-Za-z0-9][A-Za-z0-9.-]+\.[A-Za-z]{2,6}"
```

We used a few other useful flags in this command:

E - use extended regular expressions syntax.

o - show only parts of lines that match (NOT all lines).

r - search for all files in the current directory, as well as for all files in subdirectories.



Patterns to search with **Grep** are called regular expressions. With their help you can search not only numbers and emails, but also URLs, IP addresses, crypto currency wallets, usernames, ZIP codes, code comments and much more.

We will not go into their syntax in this course. For most OSINT purposes, you will just need to copy a regular expressions from Stackoverflow or some online library:

RegexLib (https://regexlib.com/)

RegexHub (https://projects.lukehaas.me/regexhub/)

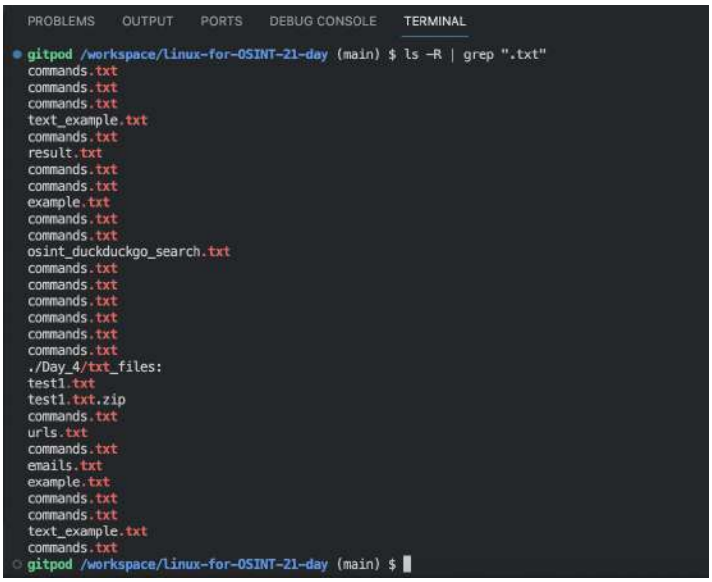
And, of course, you can always ask ChatGPT, Perplexity or Phind to compose a regular expression for you (we'll talk more about helper tools on **Day 20**).

If you are interested in this topic, my Medium blog has a very detailed and long article about it: **How regular expressions can be useful in OSINT. Theory and some practice using Google Sheets** (https://medium.com/osint-ambition/this-article-consists-of-three-short-parts-31d31efabd5).

You can also use regular expressions when working with other Linux utilities: **find**, **sed**, **awk**, **tr**, **vi** and others.

If you want to search for files strictly with a certain extension or for files whose name contains a certain string, replace part of the file name with asterisks:

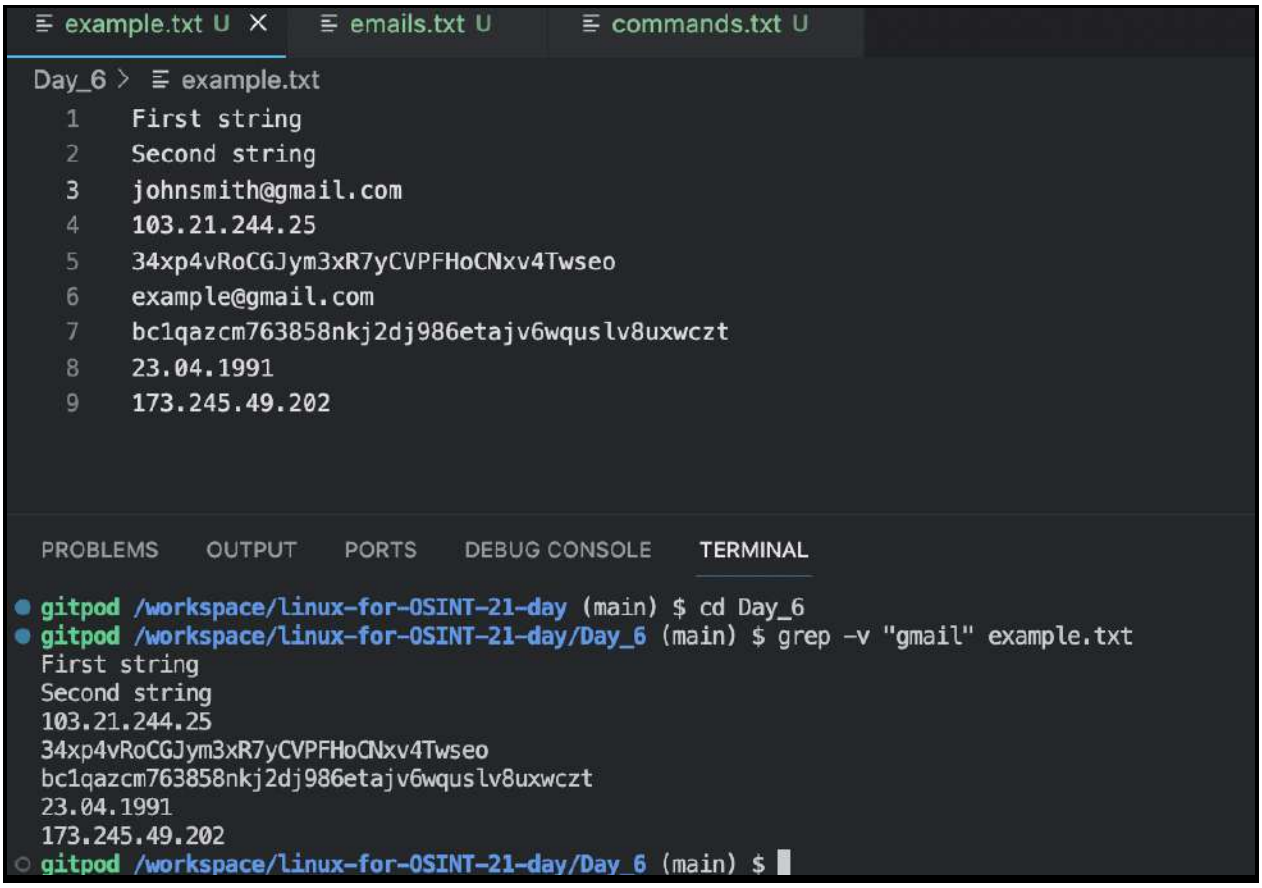
grep -n "gmail" *.txt



The real power of **Grep** comes in combination with other commands. For example, you can use it to filter the output of the **ls** command:

```
ls -R | grep ".txt"
```

Note that we used the **ls** command with the **-R** flag to display the list of files in the subdirectories too.



Also, grep allows you to search for strings that do NOT contain the word you are looking for or match a regular expression. Use the **-v** flag:

```
grep -v "gmail" example.txt
```

One last, very important note. You can run **Grep** multiple times at the one line. Let's try to find strings that contain "johnsmith" but do not contain "gmail":

```
grep "johnsmith" example.txt | grep -v "gmail"
```

Again, notice how we combine two commands on the same line using the **|**. This is a simple but very important skill that will allow you to get the most out of Linux.

Alternatives

Name	Description	Link.
RZgrep	"Grep utility that searches through zip,jar,ear,tgz,bz2 in any form of nesting"	https://github.com/MoserMichae
XLSgrep	"CLI tool to search text in **XLSX, XLS and ODS** files. It works similarly to Unix/GNU Linux grep"	https://github.com/zazuum/xlsx
Hgrep	"Grep" for searching/replacement in html code	https://github.com/TUVIMEN/hg
BINgrep	Grep for binaries files	https://github.com/m4b/bingrep
Cgrep	Grep for source codes in C/C++	https://github.com/awgn/cgrep
Ngrep	PCAP-based tool that allows you to specify an extended regular or hexadecimal expression to match against data payloads of packets. It understands many kinds of protocols, including IPv4/6, TCP, UDP, ICMPv4/6, IGMP and Raw	https://github.com/jpr5/ngrep
PHPgrep	Grep for PHP source code	https://github.com/quasilyte/php
UCG (Universal Code Grep)	Extremely fast grep-like tool specialized for searching large bodies of source code	https://github.com/gvansickle/uc

MP4grep	CLI for transcribing and searching audio/video files	https://github.com/o-oconnell/m
minigrep	A minimalistic regex search and print out tool implementation as per Ch.12 of The Rust Programming Language	https://github.com/paul-michell
Hackers Grep	Utility to search for strings in PE executables including imports, exports, and debug symbols	https://github.com/codypierce/h
JVgrep	Grep for Japanese vimmer. You can find text from files that written in another Japanese encodings	https://github.com/mattn/jvgrep
LiveGrep	Online grep search in Github repositories	https://livegrep.com/
DNgrep	Graphical grep tool for Windows	https://github.com/dnGrep/dnG
Greplace.vim	Global search and replace for VI	https://github.com/skwp/greplac
SearchInProject_SublimeText	Use ag, ack, grep and git grep directly from Sublime Text 2 and 3	https://github.com/leonid-shevtsov/SearchInProject_Subli
ACK3	grep-like Perl search tool optimized for source code	https://github.com/beyondgrep/
The Silver Searcher	Extreme grep-like C+Perl search tool for source code	https://github.com/ggreer/the_s
Sift	GO fast and powerfull alternative for grep	https://github.com/svent/sift

The **Grep** utility is designed to search through text files. But it has many analogs that work with other formats: XLSX, ZIP, JAR, MP4, PDF etc. You can find links to them in the Awesome Grep (https://github.com/cipher387/awesome-grep) repository.

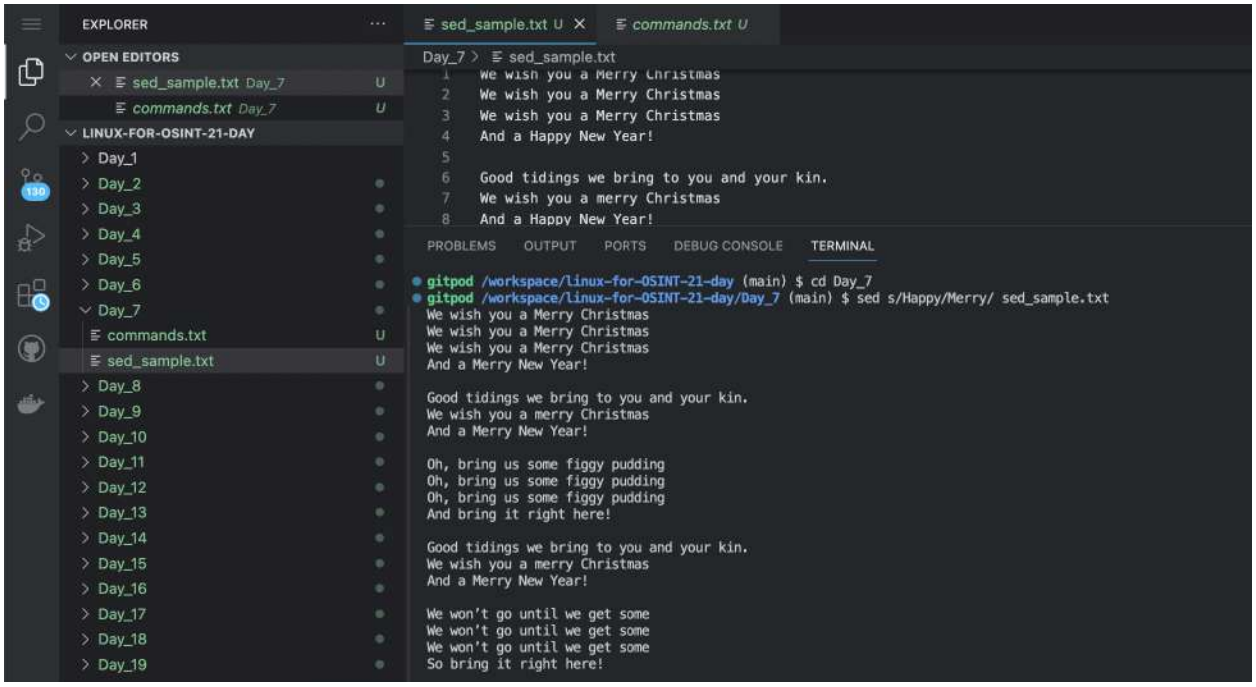
Day 7. Sed and Awk

Today is one of the longest lessons with the most examples. Some people will find it the most boring in the whole course. But believe me, from the point of view of learning to save time, it is the most useful day. It is even more useful than the day in which we will talk about AI assistants.

Sed (https://www.gnu.org/software/sed/) is a non-interactive command-line text editor (it would be more accurate to say stream editor). In other words, it allows you to make changes to a text file without opening it.

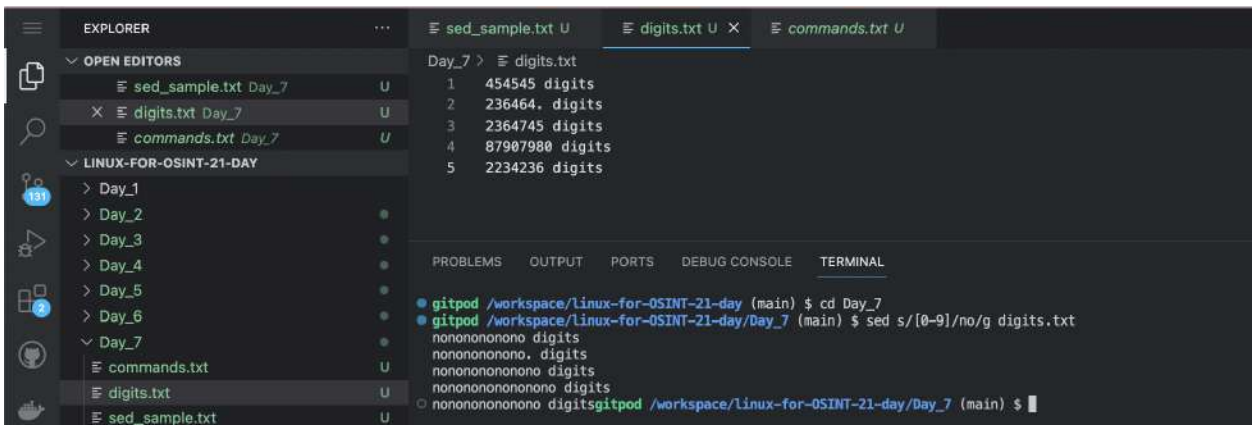
```
gitpod /workspace/linux-for-OSINT-21-day (main) $ sed -h
sed: invalid option -- 'h'
Usage: sed [OPTION]... {script-only-if-no-other-script} [input-file]...

  -n, --quiet, --silent
                 suppress automatic printing of pattern space
      --debug
                 annotate program execution
  -e script, --expression=script
                 add the script to the commands to be executed
  -f script-file, --file=script-file
                 add the contents of script-file to the commands to be executed
  --follow-symlinks
                 follow symlinks when processing in place
  -i[SUFFIX], --in-place[=SUFFIX]
                 edit files in place (makes backup if SUFFIX supplied)
  -l N, --line-length=N
                 specify the desired line-wrap length for the `l' command
  --posix
                 disable all GNU extensions.
  -E, -r, --regexp-extended
                 use extended regular expressions in the script
                 (for portability use POSIX -E).
  -s, --separate
                 consider files as separate rather than as a single,
                 continuous long stream.
      --sandbox
                 operate in sandbox mode (disable e/r/w commands).
  -u, --unbuffered
                 load minimal amounts of data from the input files and flush
                 the output buffers more often
  -z, --null-data
                 separate lines by NUL characters
      --help     display this help and exit
      --version  output version information and exit
```

First, let's just try to replace one piece of text with another in the file:



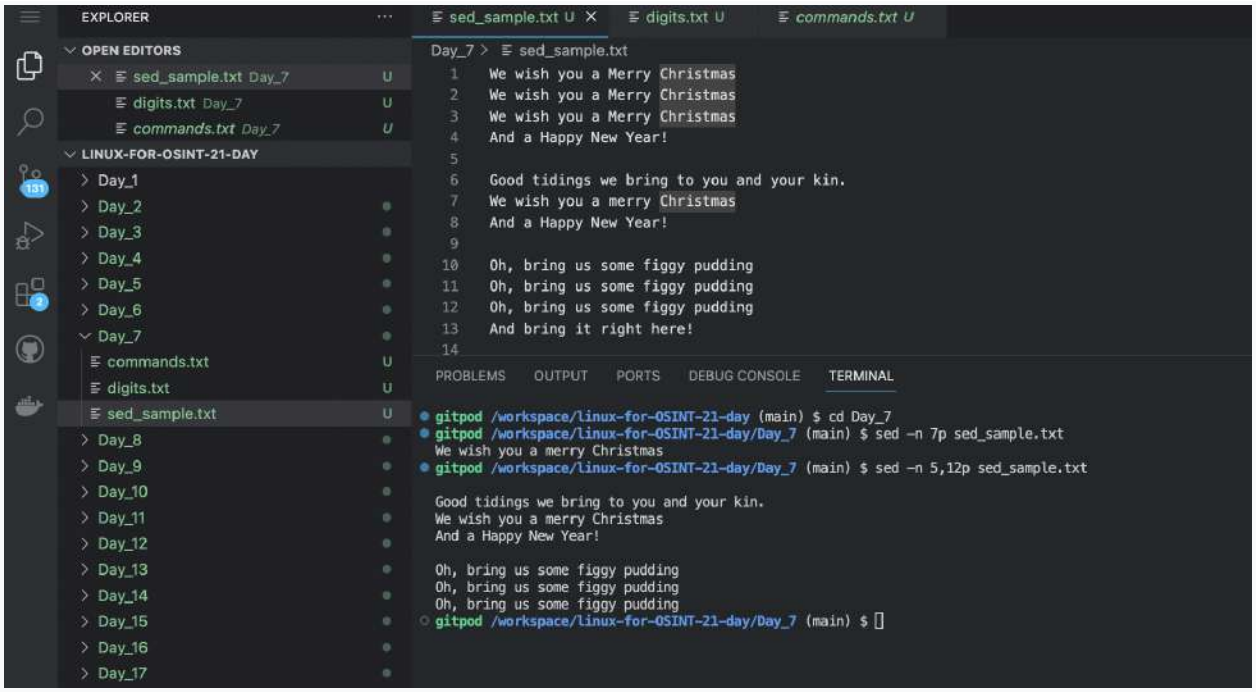
cd Day_7
sed s/Happy/Merry/ sed_sample.txt

Now let's try to replace the text in the file using a regular expression (replace each digit to word “no”):



sed s/[0-9]/no/g digits.txt

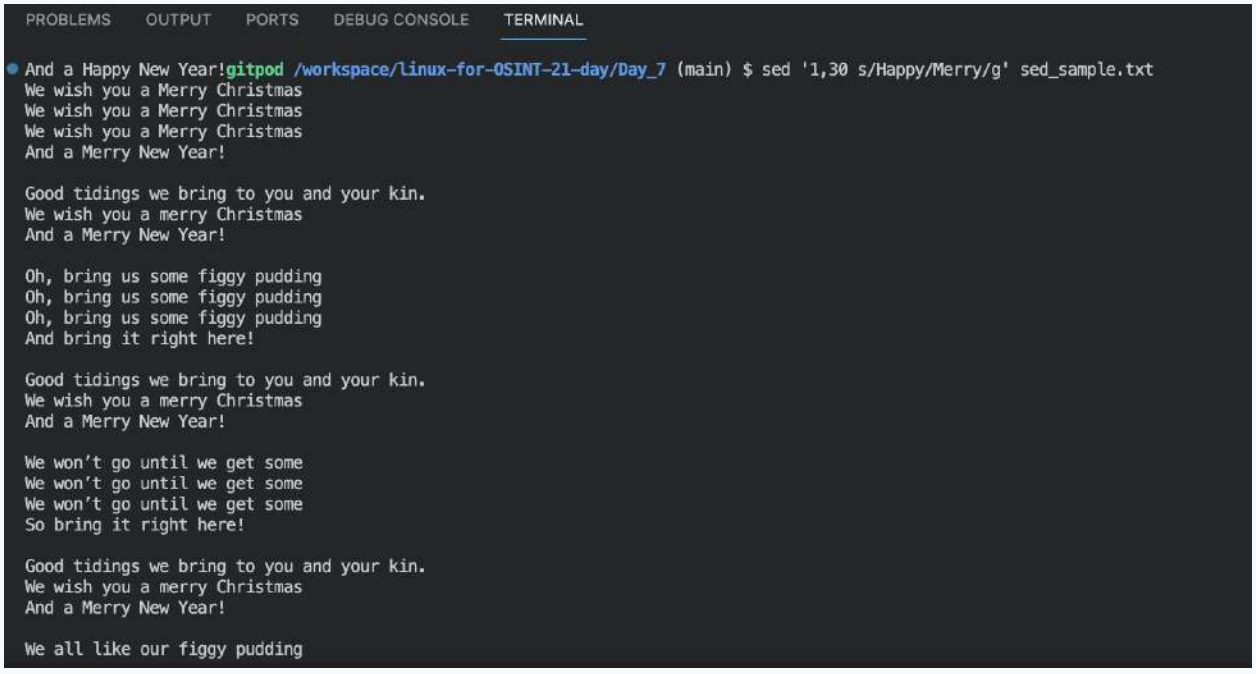
The **-n** flag can be used to extract a single line or a range of lines from a file:



sed -n 7p sed_sample.txt
sed -n 5,12p sed_sample.txt

For a file of a few dozen lines, this command seems pointless. Its true power comes out when you have, for example, a CSV file with tens of millions of lines that Excel refuses to open and you need to see what is on lines number 108100, 999588, 10009898 to make sure the data is in the right format.

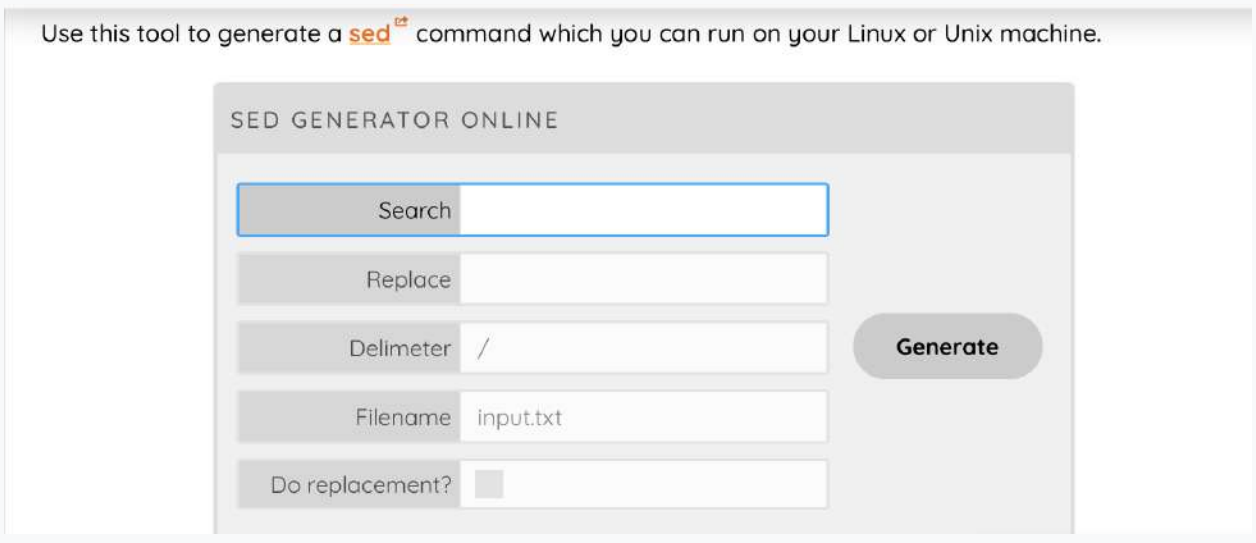
It is also possible to combine this to function and replace a word or a regular expression in a given range of strings:



sed '1,30 s/Happy/Merry/g' sed_sample.txt

More examples can find in **"Useful sed"** Github repository (https://github.com/adrianlarion/useful-sed).

.



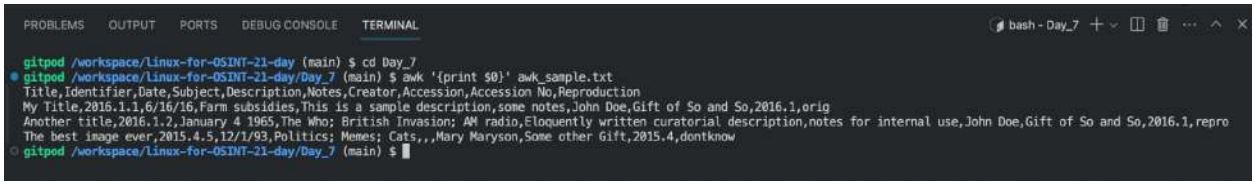
Do not try to memorize the syntactic rules for composing **Sed** commands. You can always find an example on StackOverflow, ask some AI code assistant for help, or use a special online service like **Sed command generator** (https://www.jamiebalfour.scot/devkit/sed/).

AWK

Awk (https://www.gnu.org/software/gawk/manual/gawk.html) is a scripting language for text processing. Interesting fact. Awk is not a technical acronym, but simply the initials of the three developers of this language: Alfred V. **A**ho, Peter J. **W**einberger и Brian W. **K**ernighan.

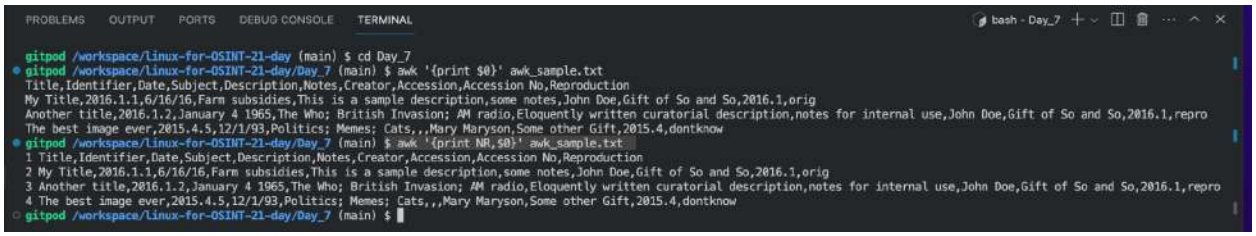
Awk, like **Sed** does not require installation as it is included in Linux by default.

First, let's try to display the entire text of the file. It is easier to do this with **Cat**, but this example is necessary to better understand awk syntax:



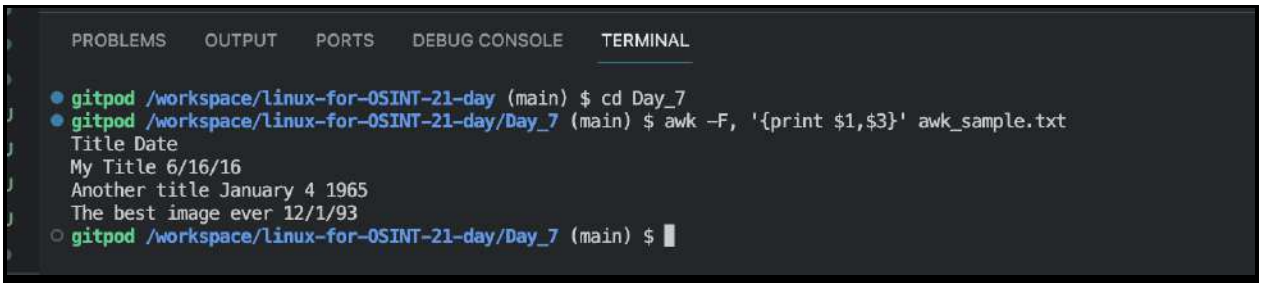
awk '{print $0}' awk_sample.txt

Now let's do the same thing, but with line numbering:



awk '{print NR,$0}' awk_sample.txt

Let's display only the first and third columns of the document (using flag **-F** to indicate that the separator is a comma):

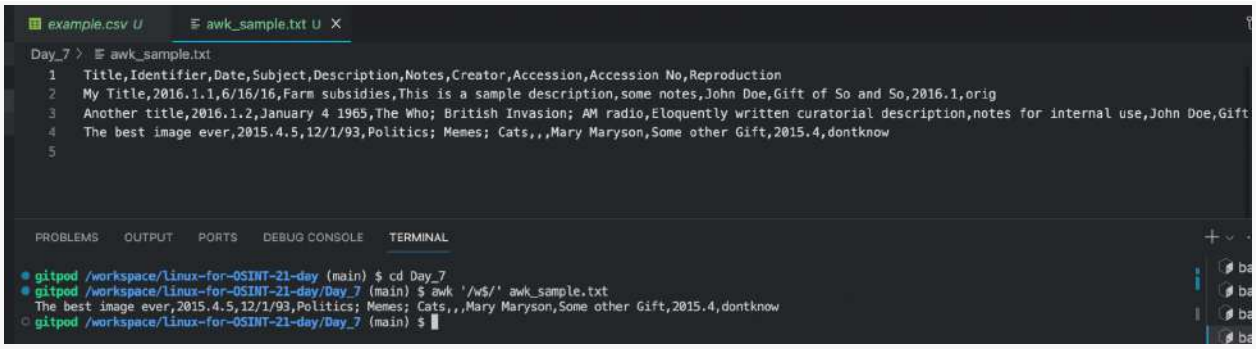


awk -F, '{print $1,$3}' awk_sample.txt

Now output only the strings that start with a certain character:

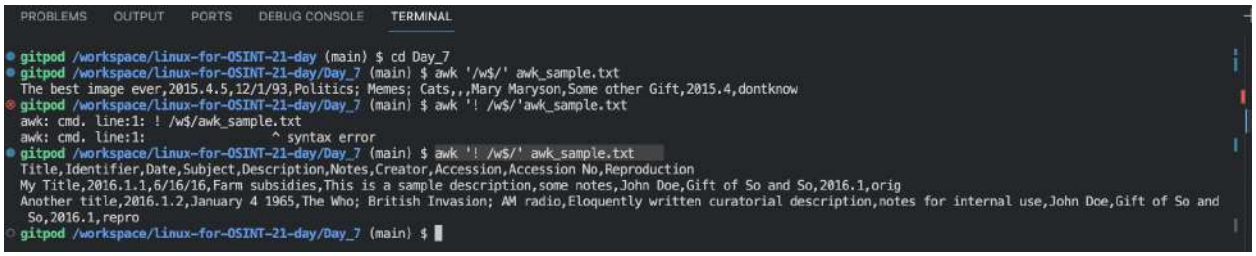
awk '/^O/' awk_sample.txt

Conversely, strings that end with a specific character:



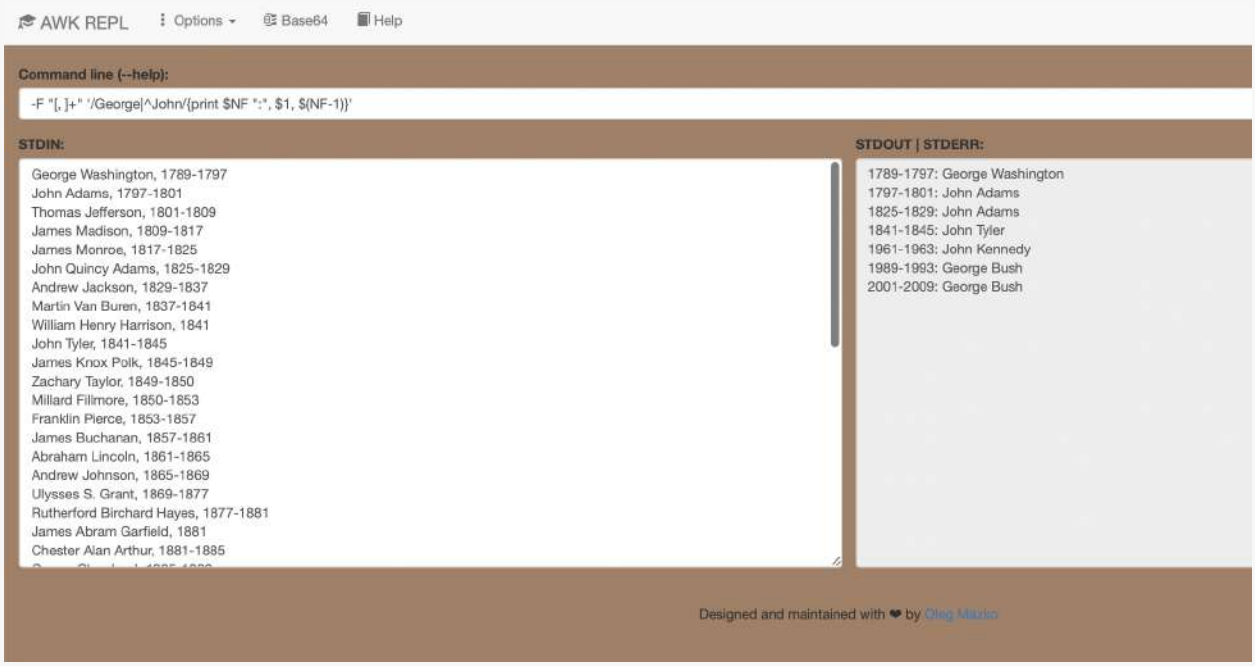
awk '/w$/' awk_sample.txt

And now all the strings that do NOT end with a certain character:



awk '! /w$/' awk_sample.txt

More examples can find in **Simple AWK** Github repository (https://github.com/adrianlarion/simple-awk). As with **Sed**, you can ask the AI assistant for help writing commands or use special services.



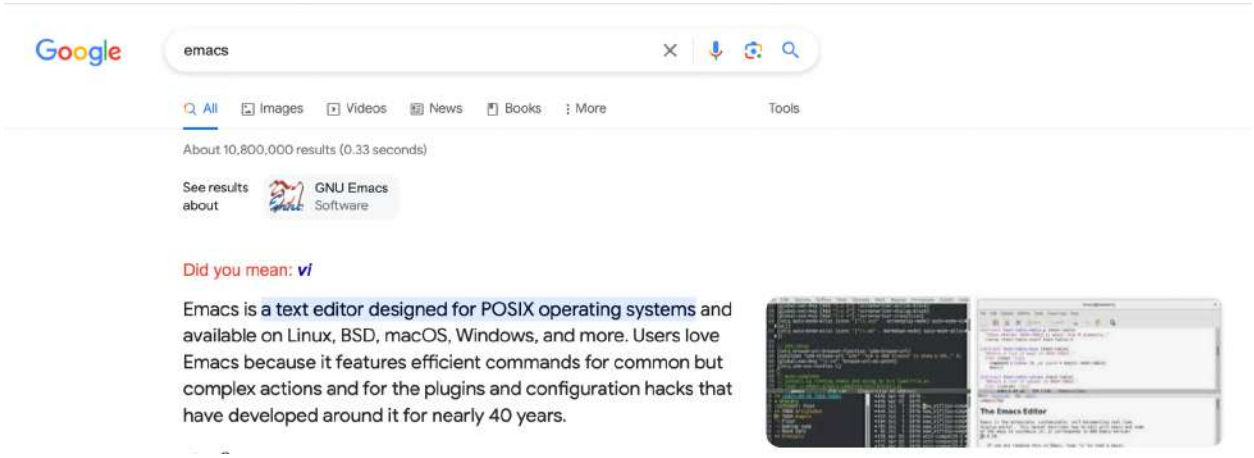
AWK online (https://awk.js.org) is an online tool on which it is very convenient to test how various commands are executed on small text fragments.

When reading the additional documentation, you may notice that **Grep**, **Sed** and **Awk** may have common functions and it is not always clear which command is best to use. And more often than not, it doesn't matter.

Yes, 20-30 years ago when computers had modest capabilities it made sense to think deeply about this when working with large files. Today, however, in most situations it is not an important issue at all.

Day 8. Vim text editor

Interesting fact, if you type "emacs" into Google, the search engine will ask you, "Maybe you mean vi?" (vim is an improved version of the vi text editor, which was introduced first time in 1976). Many developers around the world really love Vim, have been using it for decades and are constantly discovering some amazing new features of this text editor.



Vim (https://www.vim.org) - is a universal and highly configurable command line text editor. The first version of it appeared in 1991 and since then it has continued to be popular, despite many competitors in the form of other text editors and IDEs (Integrated Development Environment). Let's run it and see what it's good for.

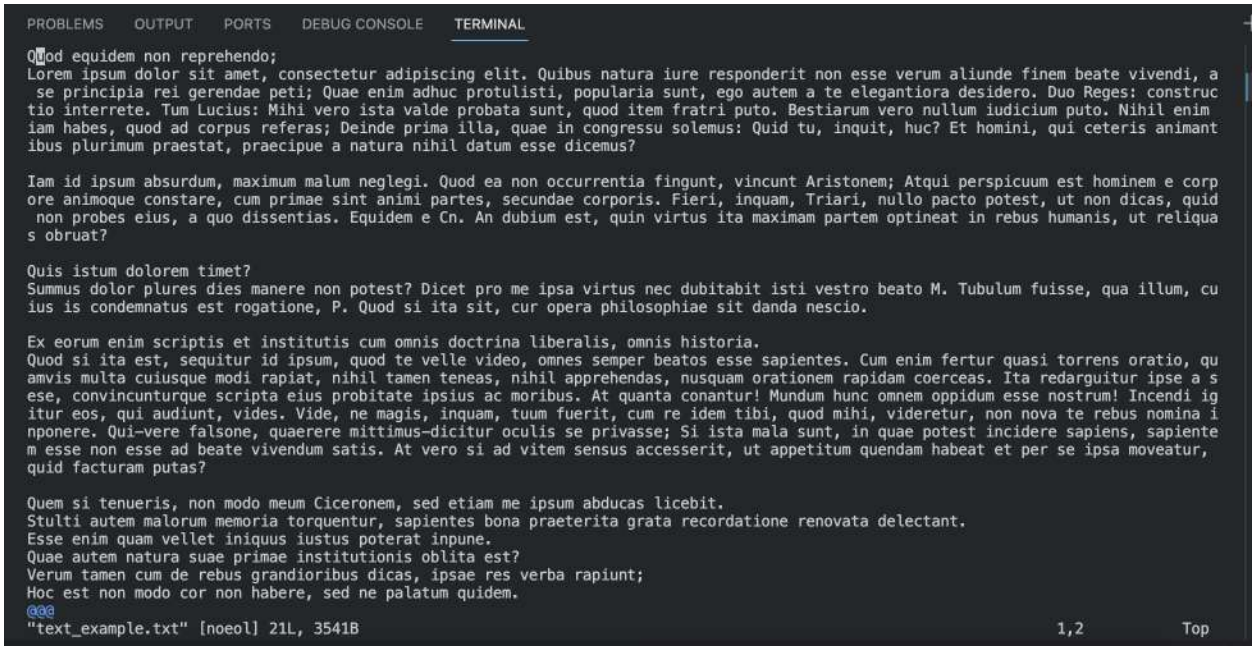
```
gitpod /workspace/linux-for-OSINT-21-day/Day_14 (main) $ vim --version
VIM - Vi IMproved 8.2 (2019 Dec 12, compiled Oct 16 2023 18:15:38)
Included patches: 1-579, 1969, 580-1854, 1857, 1855-1857, 1331, 1858, 1858-1859, 1873, 1860-1969, 1992, 1970-1992, 2010, 1993-3995, 4563, 4646, 4
774, 4895, 4899, 4901, 4919, 213, 1840, 1846-1847
Modified by team+vim@tracker.debian.org
Compiled by team+vim@tracker.debian.org
Huge version without GUI.  Features included (+) or not (-):
+acl               +file_in_path      +mouse_urxvt       -tag_any_white
+arabic            +find_in_path      +mouse_xterm       -tcl
+autocmd           +float             +multi_byte        +termguicolors
+autochdir         +folding           +multi_lang        +terminal
-autoservername    -footer            -mzscheme          +terminfo
-balloon_eval      +fork()            +netbeans_intg     +termresponse
+balloon_eval_term +gettext           +num64             +textobjects
-browse            -hangul_input      +packages          +textprop
++builtin_terms    +iconv             +path_extra        +timers
+byte_offset       +insert_expand     -perl              +title
+channel           +ipv6              +persistent_undo   -toolbar
+cindent           +job               +popupwin          +user_commands
-clientserver      +jumplist          +postscript        +vartabs
-clipboard         +keymap            +printer           +vertsplit
+cmdline_compl     +lambda            +profile           +vim9script
+cmdline_hist      +langmap           -python            +viminfo
+cmdline_info      +libcall           +python3           +virtualedit
+comments          +linebreak         +quickfix          +visual
+conceal           +lispindent        +reltime           +visualextra
+cryptv            +listcmds          +rightleft         +vreplace
+cscope            +localmap          -ruby              +wildignore
+cursorbind        -lua               +scrollbind        +wildmenu
+cursorshape       +menu              +signs             +windows
```

Let’s install:

sudo apt update
sudo apt install vim

```
PROBLEMS  OUTPUT  PORTS  DEBUG CONSOLE  TERMINAL                                  bash - Day_14
gitpod /workspace/linux-for-OSINT-21-day/Day_14 (main) $ sudo apt install vim
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  vim-common vim-runtime
Suggested packages:
  ctags vim-doc vim-scripts
The following packages will be upgraded:
  vim vim-common vim-runtime
3 upgraded, 0 newly installed, 0 to remove and 80 not upgraded.
Need to get 8,650 kB of archives.
After this operation, 7,168 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim amd64 2:8.2.3995-1ubuntu2.13 [1,734 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-runtime all 2:8.2.3995-1ubuntu2.13 [6,834 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-common all 2:8.2.3995-1ubuntu2.13 [81.5 kB]
Fetched 8,650 kB in 1s (7,913 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
(Reading database ... 35595 files and directories currently installed.)
Preparing to unpack .../vim_2%3a8.2.3995-1ubuntu2.13_amd64.deb ...
Unpacking vim (2:8.2.3995-1ubuntu2.13) over (2:8.2.3995-1ubuntu2.12) ...
Preparing to unpack .../vim-runtime_2%3a8.2.3995-1ubuntu2.13_all.deb ...
Unpacking vim-runtime (2:8.2.3995-1ubuntu2.13) over (2:8.2.3995-1ubuntu2.12) ...
Preparing to unpack .../vim-common_2%3a8.2.3995-1ubuntu2.13_all.deb ...
Unpacking vim-common (2:8.2.3995-1ubuntu2.13) over (2:8.2.3995-1ubuntu2.12) ...
Setting up vim-common (2:8.2.3995-1ubuntu2.13) ...
Setting up vim-runtime (2:8.2.3995-1ubuntu2.13) ...
Setting up vim (2:8.2.3995-1ubuntu2.13) ...
Processing triggers for man-db (2.10.2-1) ...
```

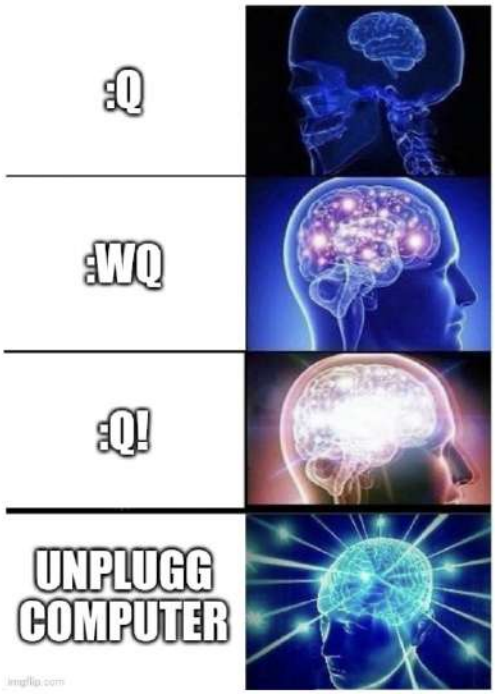
And run:



cd Day_8
vim text_example.txt

We will do nothing more with this text for now, but go straight to the most important question.

Exit Vim



This is a very important question that has inspired hundreds of people to create memes (google vim exit memes). Save these instructions somewhere.

1. Press the **Esc** key to switch to command mode.
2. Type **:q!** and press Enter (or type **:wq!** and press Enter, if you want to save the changes)

In case you forget these instructions remember that you can always just click the right cross and close the terminal (in Gitpod too).

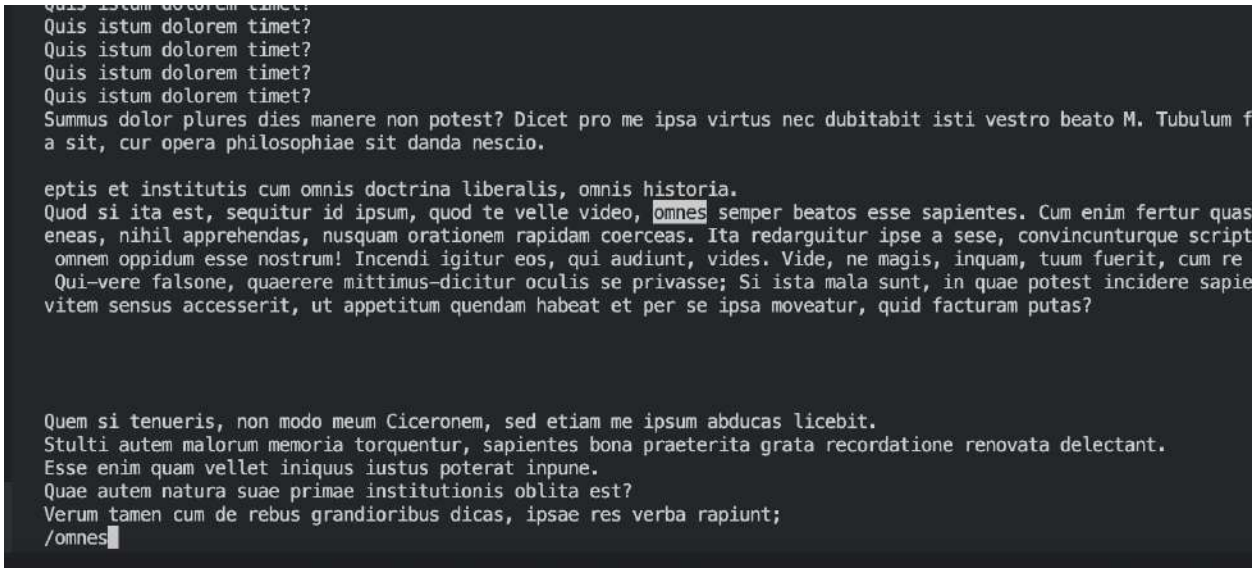
If suddenly, for some reason, the instructions above seem too simple for you, you can try a few dozen other ways to quit vim (https://github.com/hakluke/how-to-exit-vim).

Using Vim

Let's open the file again:

vim text_example.txt

And let's try to navigate through the text:

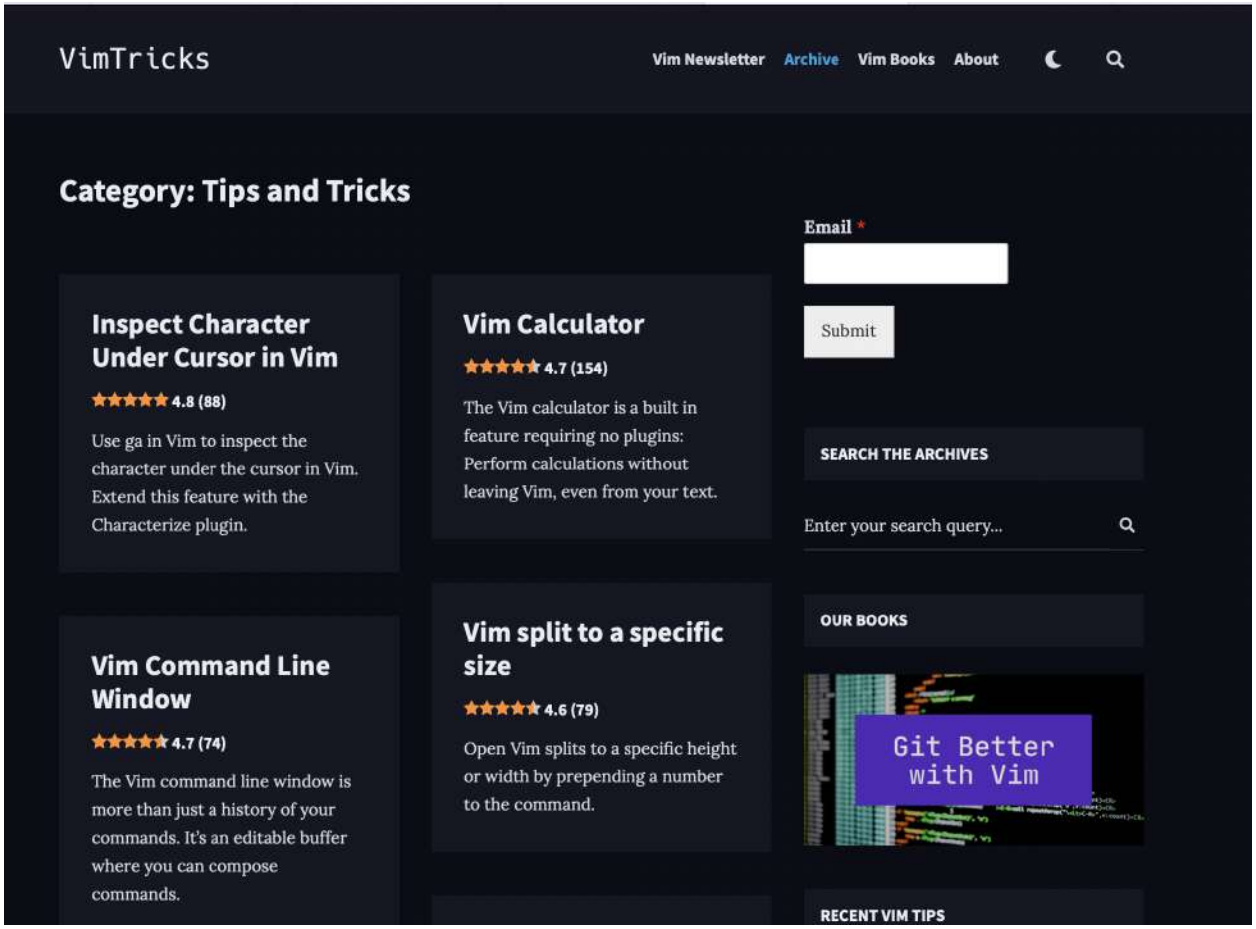


G - move to the last line.
gg - move to the first line
^ - move to the first character of current line
+ - move to the first character of next line
yyp - duplicate current line
dd - delete current line
g; - go back to the last modified location
x - delete character or selected
y - delete character or selected text
/ - find text

(If something suddenly doesn't work, press Esc to enter command entry mode).

And the bottom line is that there are over a thousand such quick commands. Yes, over a thousand commands for working with text files. Here's a list of 1,349, but it's probably not complete, since Vim is infinite (https://pastebin.com/xBRYzAw0).

If you pick at least a few dozen commands that best suit your goals, you can take your productivity to a whole new level (as well as improve memory and concentration tremendously). This is what hundreds of thousands of **Vim** fans around the world have already done (but I'm just about to do it myself, to be honest).



Learning tricks for **Vim** is a never-ending process. If you want, you can continue it on **Vim tricks newsletter** website (https://vimtricks.com/).

If you want to use VIM to its full potential, then install additional plugins suitable for your daily tasks. You can find them at **VimAwesome** (https://vimawesome.com).

Alternatives

Vim is damn good, but I still advise you not to stop at **Vim** and try different good text editors for Linux, which allow you to speed up your work and automate different tasks:

Emacs (https://www.gnu.org/software/emacs/)
Neovim (https://neovim.io)
Atom (https://github.com/atom/atom)
Notepad++ (https://notepad-plus-plus.org)
Sublime Text (https://www.sublimetext.com)

Many of them have many more plugins and integrations as Vim and can be a complete replacement for the IDE in everyday work.

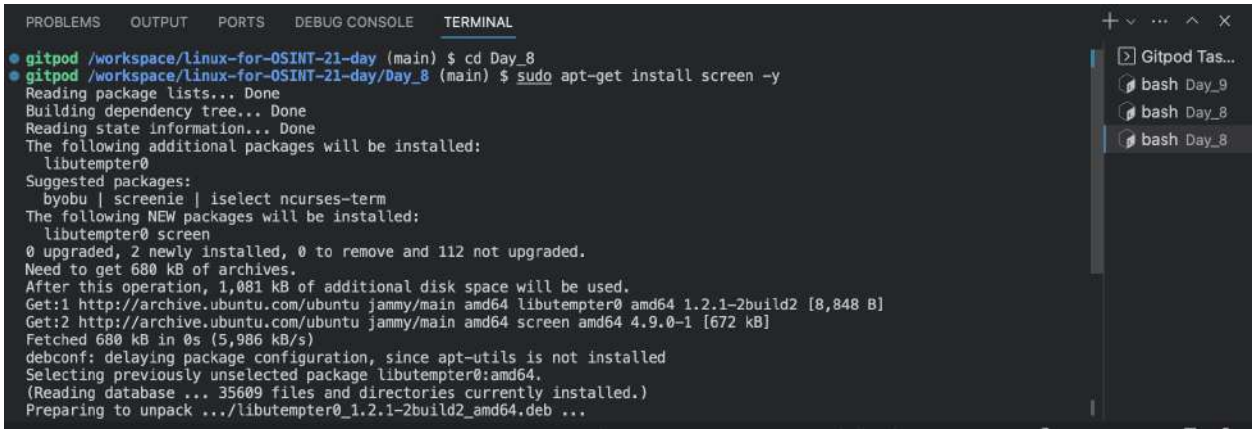
Day 9. Screen and Crone

Some commands can take a very long time to execute. Sometimes even for several days. For example, if you want to process hundreds of gigabytes of text data on a server with modest technical capacity.

Screen (https://www.gnu.org/software/screen/) allows you to perform multiple tasks simultaneously in the background. Roughly speaking, it allows you to make one terminal into several terminals working independently of each other.

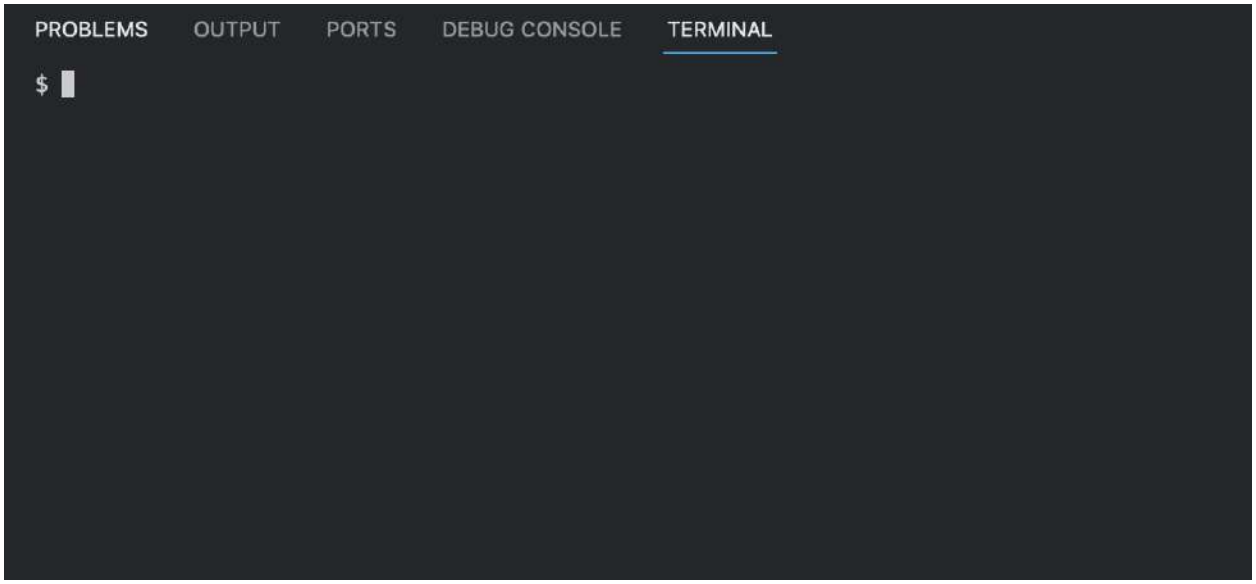
Let's start with the installation:

sudo apt-get install screen

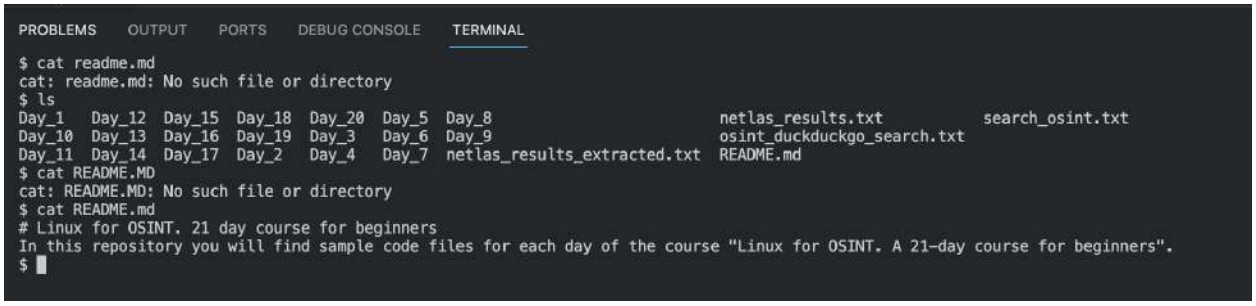


Create new session with name “demo-screen”:

screen -S demo-screen

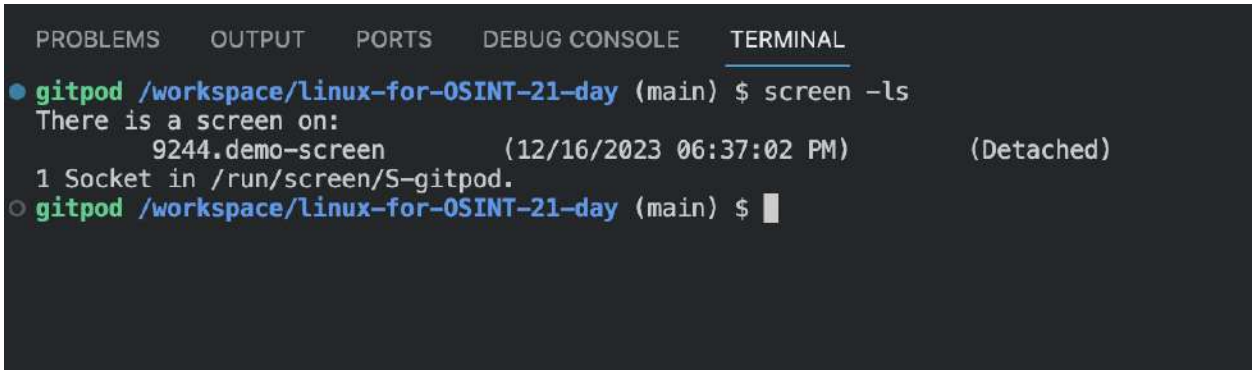


Run some commands in it and press **Ctrl+A+D** to exit.



Now let's get the list of active sessions to see if there is a new session called “demo-screen”:

```
screen -ls
```

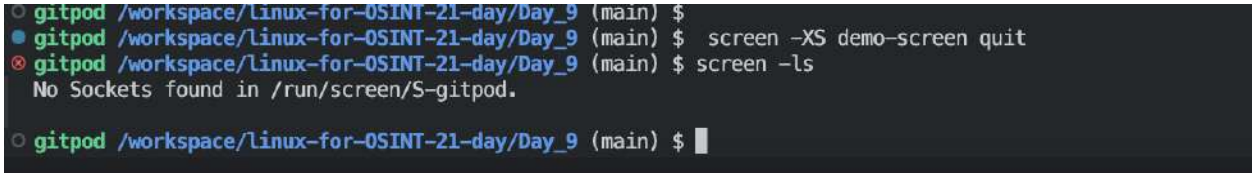


To return to a session named "demo-screen" simply type in the command line:

screen -r demo-screen

And press **Ctrl+A+D** to exit again.

And if you want to delete a session named "demo-screen", type:



screen -XS demo-screen quit

After that, check the list of sessions to make sure the deletion was successful.

screen -ls

Unfortunately, Gitpod is not very suitable for demonstrating the capabilities of **Screen**, because after closing the workspace tab and after the user is inactive for more than half an hour, all commands are stopped.

If you will be using the **Screen** on your own VM, make sure it is running until all the commands you need are executed in all sessions.

The ideal option for using the **Screen** is a VPS (Virtual Private Server) that runs 24/7 and you can just connect via SSH to it from time to time.

Crontab

Another important advantage of automation is the ability to schedule a certain sequence of actions for a certain time.

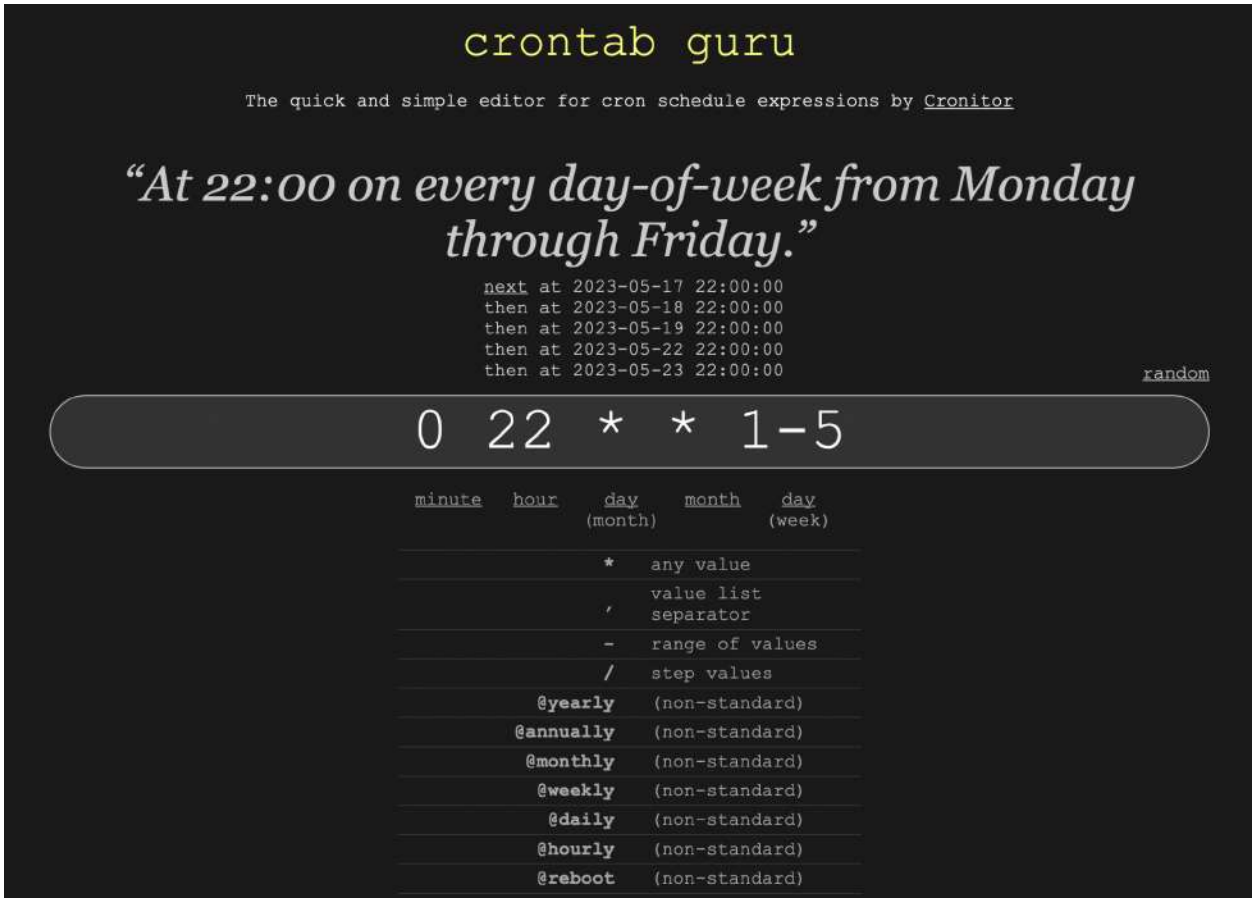
Crontab (https://ss64.com/bash/crontab.html) is a utility for editing the crontab file, which stores a schedule of commands to be executed by the **cron** daemon (an old UNIX term).

Here is an example from Wikipedia of what a crontab file might look like:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
 17 *  *   *   *  root   cd / && run-parts --report /etc/cron.hourly
 25 6  *   *   *  root   test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
 47 6  *   *   7  root   test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
 52 6  1   *   *  root   test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Gitpod prohibits the use of Crontab utility at all. And we're not going to try any more commands today.

But lastly, I want to tell you about one very good service that will come in handy when you work with Crontab on a VM or VPS.



Crontab Guru (https://crontab.guru/) - is an online service that allows you to quickly compose crone schedule expressions.

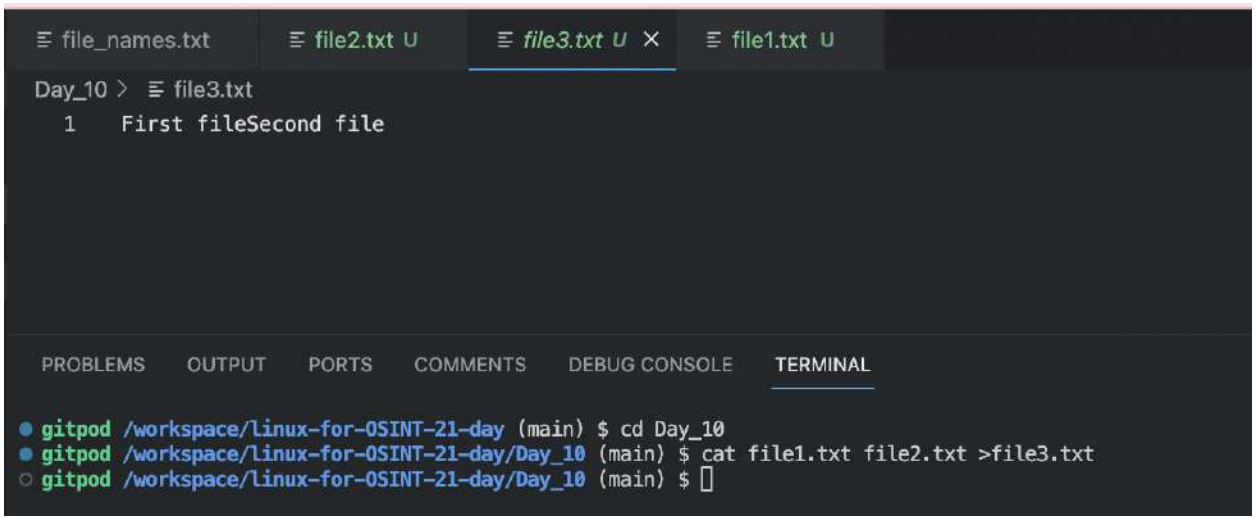
Day 10. Text analyzing and editing utilities

A significant part of this course is devoted to working with various text data formats. After all, the most important advantage of Linux is the significant reduction of routine work. Here are some more universal commands that are useful to know. All of them are available in Linux by default.

Cat

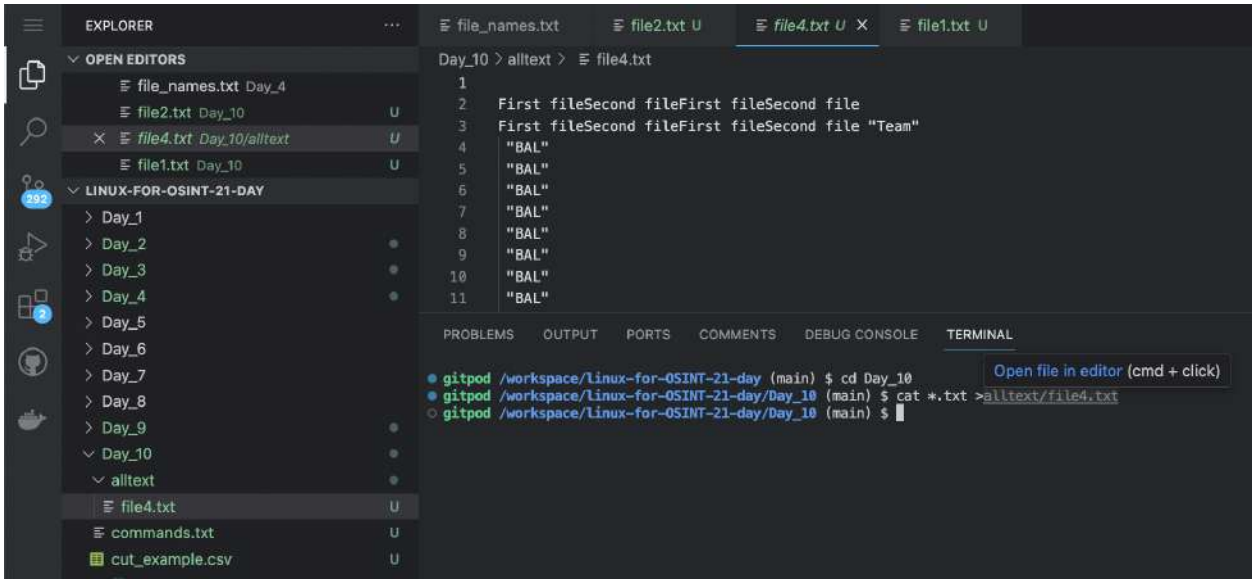
Cat (https://pubs.opengroup.org/onlinepubs/9699919799/utilities/cat.html) - utility that allows to view the contents of a text file. We have already used it in previous lessons and it may seem very simple, but it is not quite so. Two second examples.

It allows to view the contents of multiple files at once and save them all at once to a single file:



cd Day_10
cat file1.txt file2.txt >file3.txt

You can even use it to collect the content of all files with a certain extension (or containing a certain word in the name) into one file:

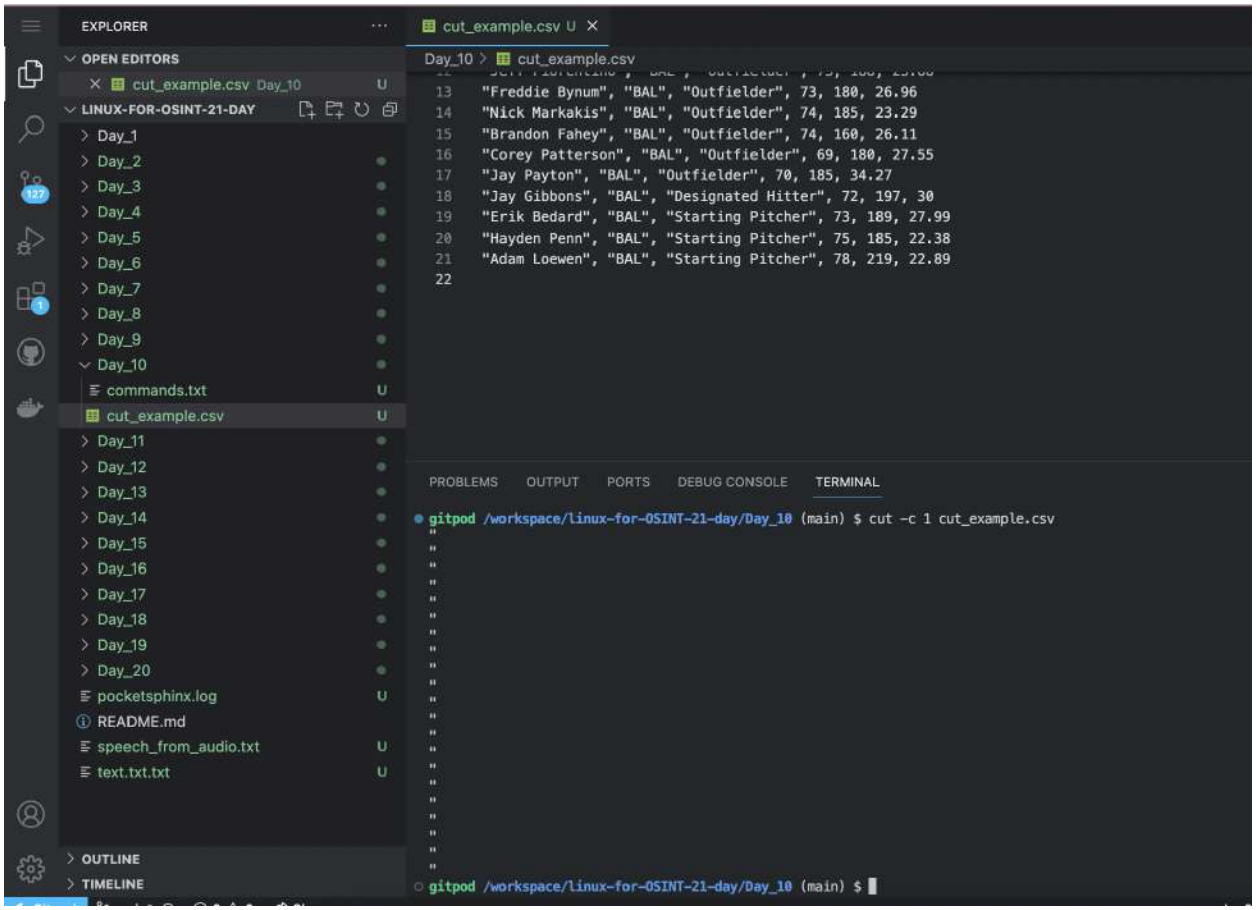


cat *.txt >alltext/file4.txt

Cut

Cut (https://pubs.opengroup.org/onlinepubs/9699919799/utilities/cut.html) is utility for cutting sections from each line of a file. Recall that we used the **Sed** utility to cut lines from the file, and the **Awk** utility to cut individual columns (**Day 7**).

Let's try to extract the first character from each line:



cut -c 1 cut_example.csv

And here's a command that cuts the first through tenth characters from each line:

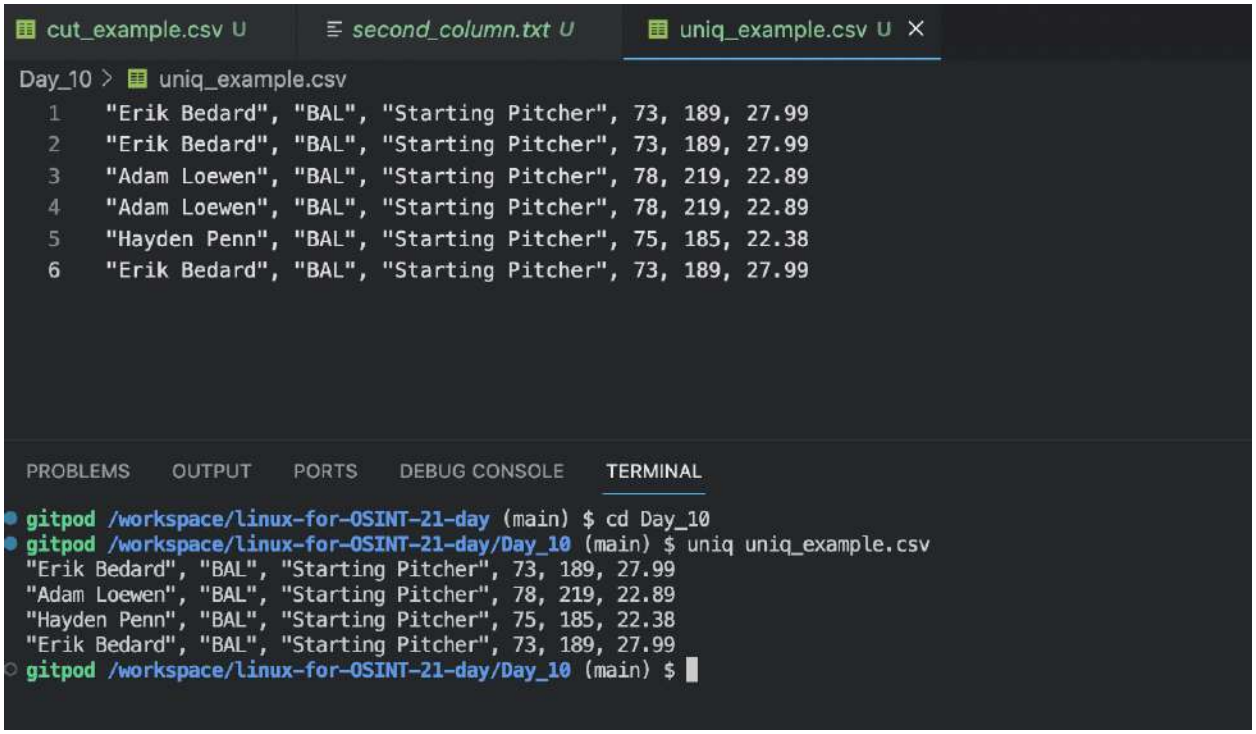
cut -c 5-10 cut_example.csv

Cut can also be used to cut individual bytes or fields with delimiter designation (another method for extracting columns from CSV files).

Uniq

Uniq (https://pubs.opengroup.org/onlinepubs/9699919799/utilities/uniq.html) - utility that allows you to remove all duplicate lines from a file.

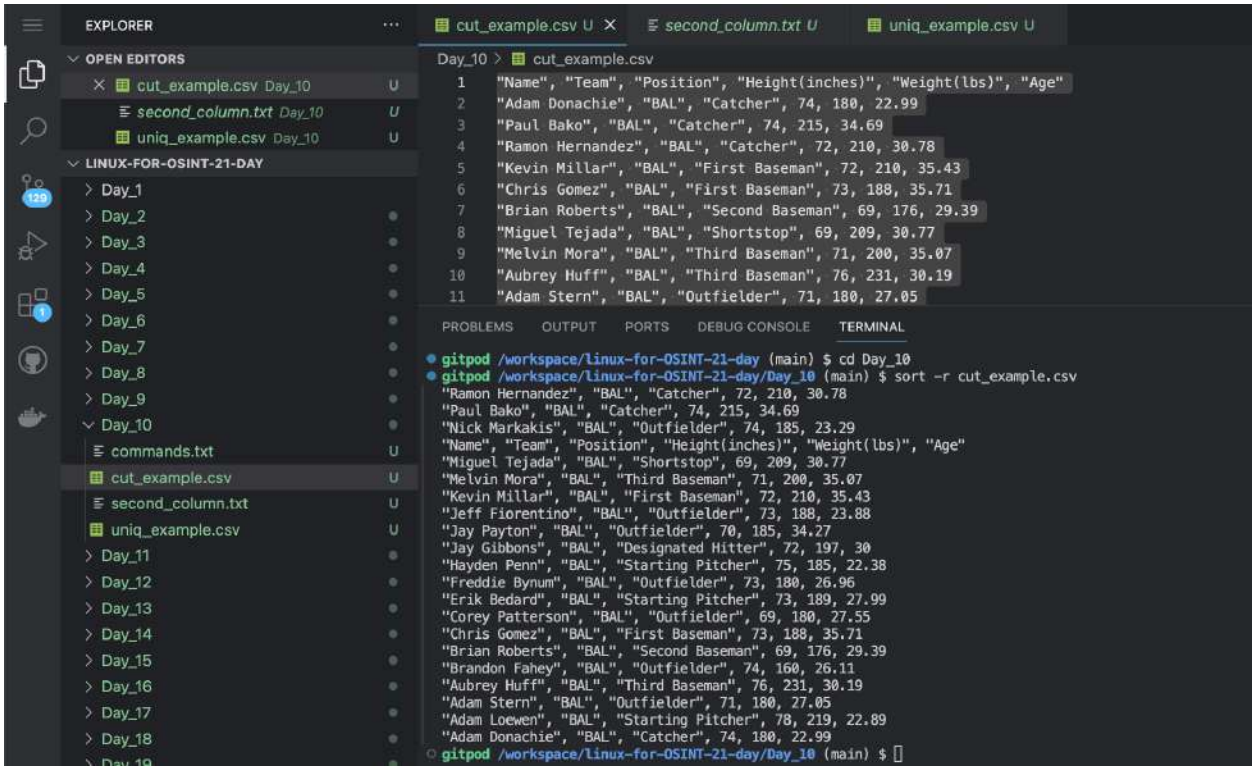
Let's try:



uniq uniq_example.csv

Sort

Sort (https://pubs.opengroup.org/onlinepubs/9699919799/utilities/sort.html) - utility that sorts files in alphabetical and reverse alphabetical order, as well as descending and ascending order.



sort -r cut_example.csv

Here we used the **-r** flag to sort the rows in reverse alphabetical order.

Iconv

Iconv (https://pubs.opengroup.org/onlinepubs/9699919799/utilities/iconv.html) utility that convert some text in one encoding into another encoding.

We won't run this command (I don't have a suitable test file at hand right now), but it will be useful for you to know about its existence. If some text suddenly displays obscure characters instead of letters, try saving the file in a different encoding using this command.

iconv -f UTF-8 -t ASCII//TRANSLIT utf8.txt acii.txt

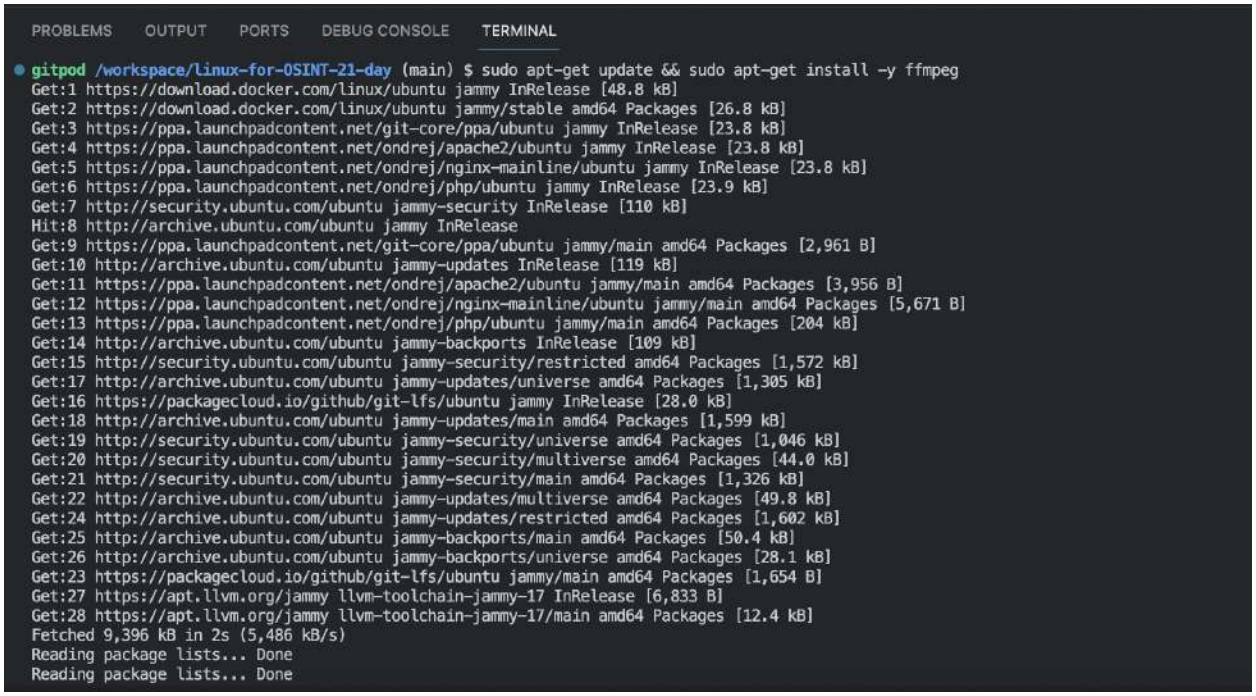
This is the last chapter in this book about automating text file manipulation in Linux. If you are interested in this topic, I recommend you to check out the repository T**ext processing Recipes Linux** (https://github.com/adrianlarion/text-processing-recipes-linux).

Day 11. Video, audio and images

Sometimes in investigations, you have to deal with a large volume of media files that are very difficult to analyze manually. Today I will show you some very simple tools that will help you save time while watching videos, listening to audio and translating text from pictures.

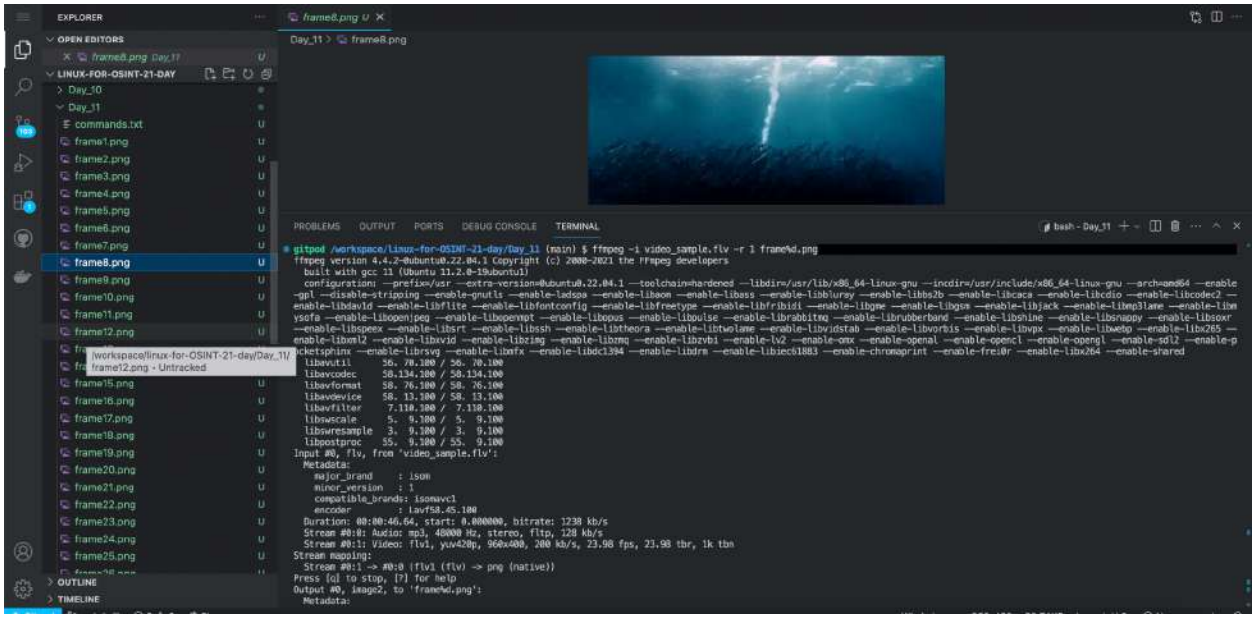
FFMPEG

FFMPEG (https://www.ffmpeg.org/) is a toolkit for handling video, audio, and other multimedia files, capable of hundreds of different operations.



Let's install:

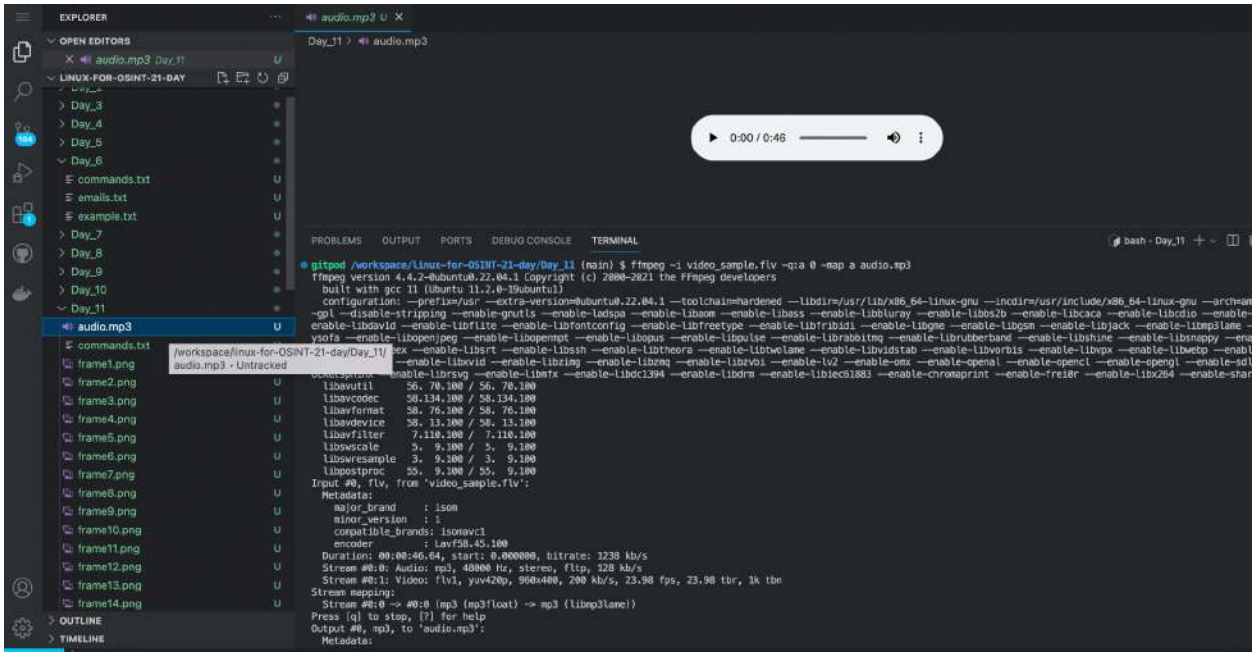
sudo apt-get update
sudo apt-get install -y ffmpeg



Now let's try to convert the video into a set of frames in png format:

```
cd Day_11
ffmpeg -i video_sample.flv -r 1 frame%d.png
```

Viewing a video frame by frame allows you to quickly understand what the video is about and highlight key episodes in it. A simple but very time-saving trick. Note that if you right-click on a file and folder in Gitpod, it can be downloaded from workspace.



Now let's try to extract the audio from the video with another simple command. You can listen to the result without leaving Gitpod:

```
ffmpeg -i video_sample.flv -q:a 0 -map a audio.mp3
```

As you can guess, audio needs to be extracted from video to turn it into text. This will help you quickly analyze it and find the key semantic parts.

Transcribe audio

PocketSphinx (https://github.com/cmusphinx/pocketsphinx) is a very simple speech recognizer.

```
gitpod /workspace/linux-for-OSINT-21-day/Day_11 (main) $ sudo apt install pocketsphinx pocketsphinx-en-us
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
pocketsphinx-en-us is already the newest version (0.8.0+real5prealpha+1-14ubuntu1).
pocketsphinx-en-us set to manually installed.
The following NEW packages will be installed:
  pocketsphinx
0 upgraded, 1 newly installed, 0 to remove and 87 not upgraded.
Need to get 32.6 kB of archives.
After this operation, 140 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pocketsphinx amd64 0.8.0+real5prealpha+1-14ubuntu1 [32.6 kB]
Fetched 32.6 kB in 0s (111 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package pocketsphinx.
(Reading database ... 36864 files and directories currently installed.)
Preparing to unpack .../pocketsphinx_0.8.0+real5prealpha+1-14ubuntu1_amd64.deb ...
Unpacking pocketsphinx (0.8.0+real5prealpha+1-14ubuntu1) ...
Setting up pocketsphinx (0.8.0+real5prealpha+1-14ubuntu1) ...
Processing triggers for man-db (2.10.2-1) ...
gitpod /workspace/linux-for-OSINT-21-day/Day_11 (main) $
```

Let's install:

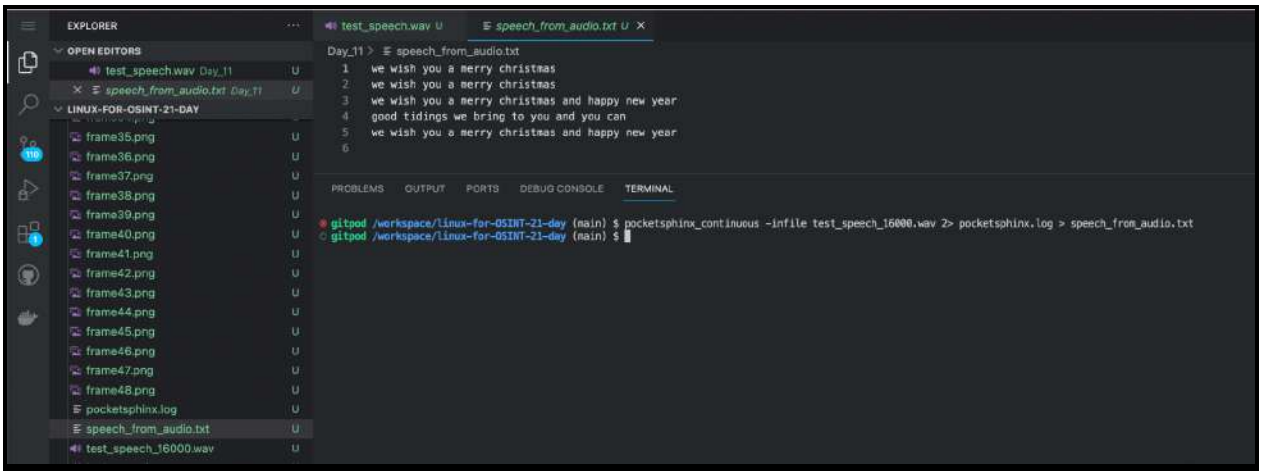
sudo apt install pocketsphinx pocketsphinx-en-us

If you want to work with audio recordings in other languages, install additional libraries. A complete list can be found here.

PocketSphinx is an easy to use utility that produces quality results. But, unfortunately, it does not work with different audio file formats, but only with wav 16000 Hz.

Fortunately, most audio formats can be easily converted to it with ffmpeg. Example:

```
ffmpeg -i test_speech.wav -ar 16000 -ac 1 test_speech_16000.wav
```



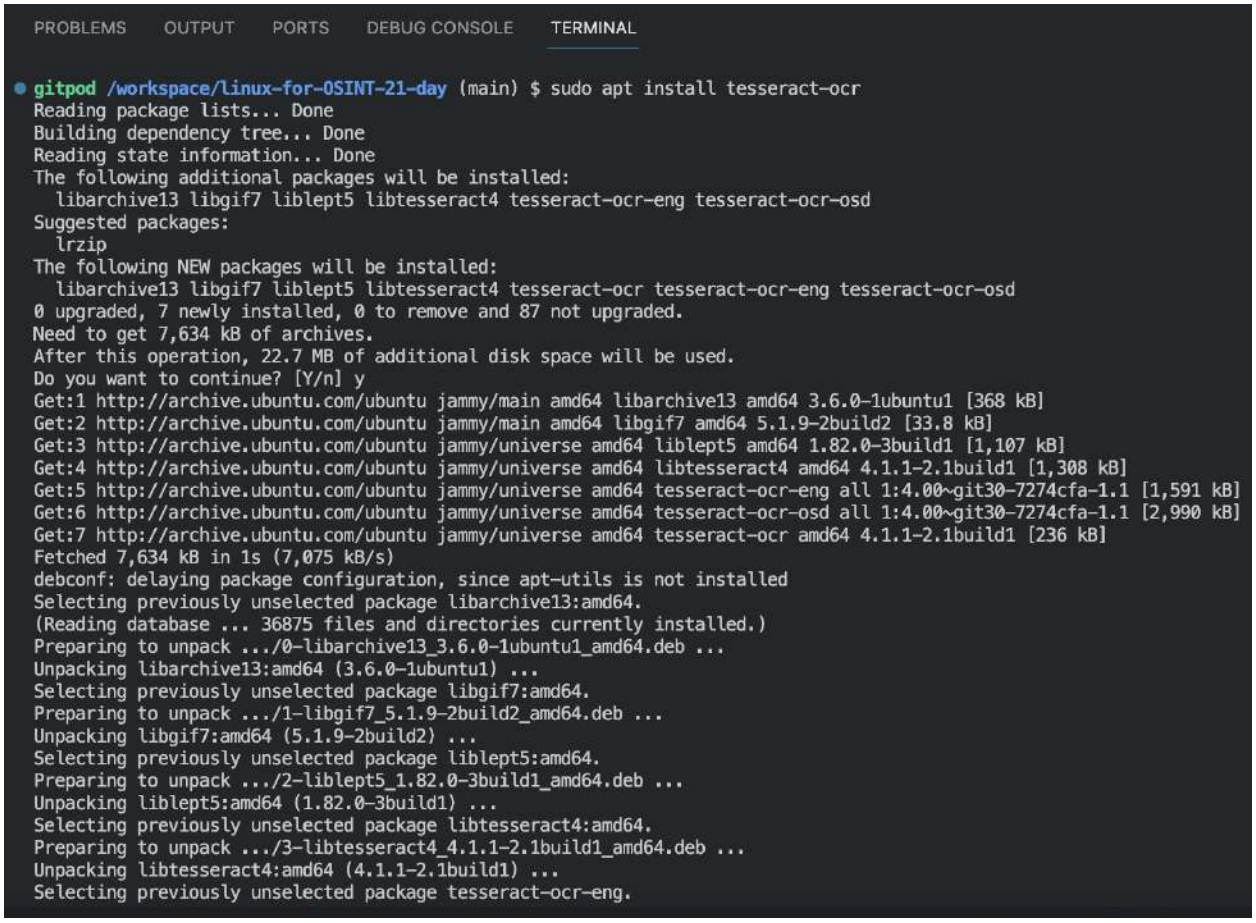
And now try convert speech audio to text:

```
pocketsphinx_continuous -infile test_speech_16000.wav 2> pocketsphinx.log > speech_from_audio.txt
```

If you need very high quality speech recognition for different rare languages, I suggest you take a look at Open AI Whisper API (https://openai.com/research/whisper). It is a paid tool, but it is really good.

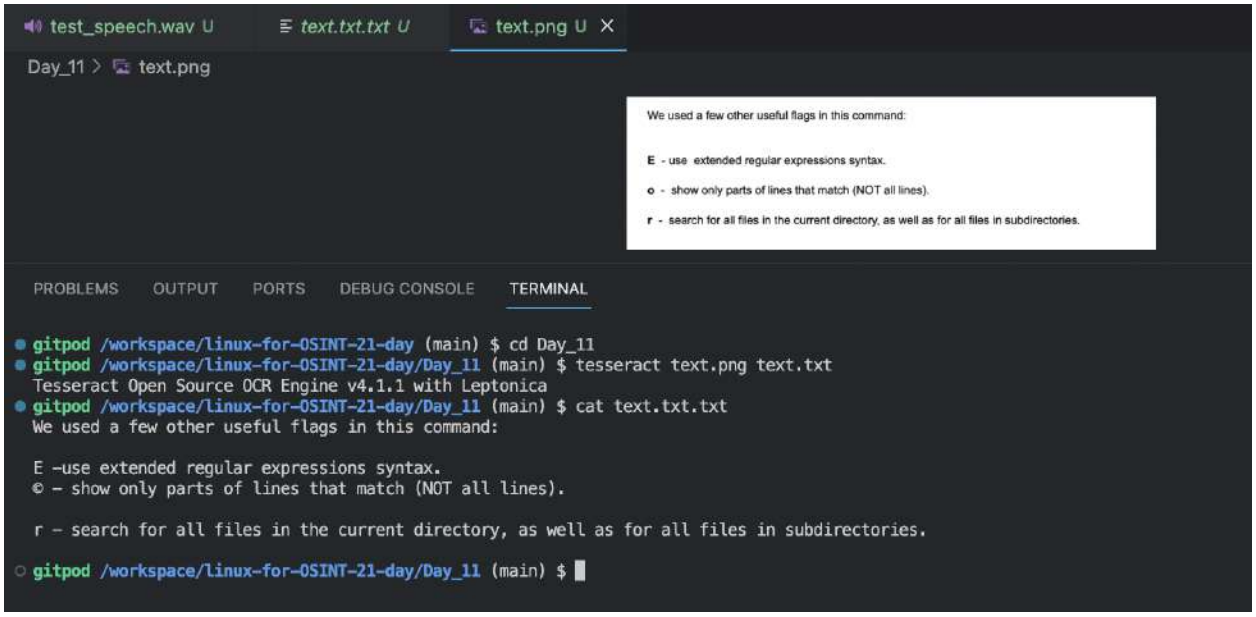
Extract text from images

Tesseract OCR (https://github.com/tesseract-ocr/tesseract) is a simple Optical Text Recognition utility.



sudo apt install tesseract-ocr

Try to OCR text in image and save to text.txt:



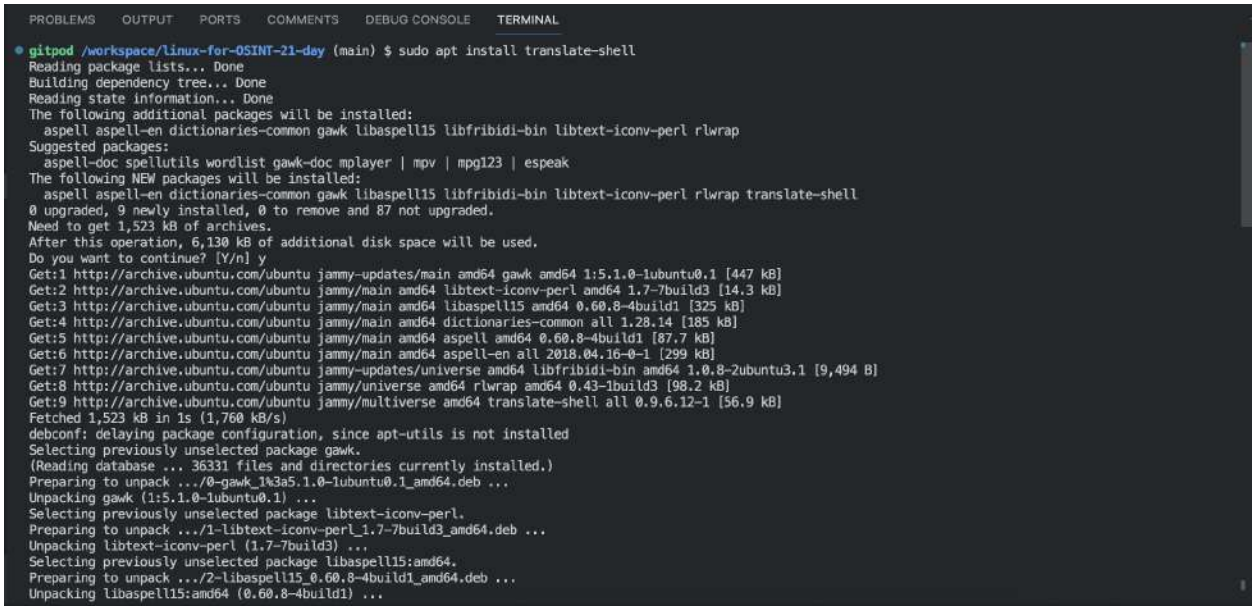
tesseract text.png text.txt

Once we've turned video, audio, and pictures into text, we can translate it! Here is one of the many ways to automate this process.

Translate text

Translate Shell (https://github.com/soimort/translate-shell) is command line translator based on Google Translate, Bing Translator, Yandex.Translate and other translate services.

Let’s install:



sudo apt install translate-shell

And translate text from file in French:

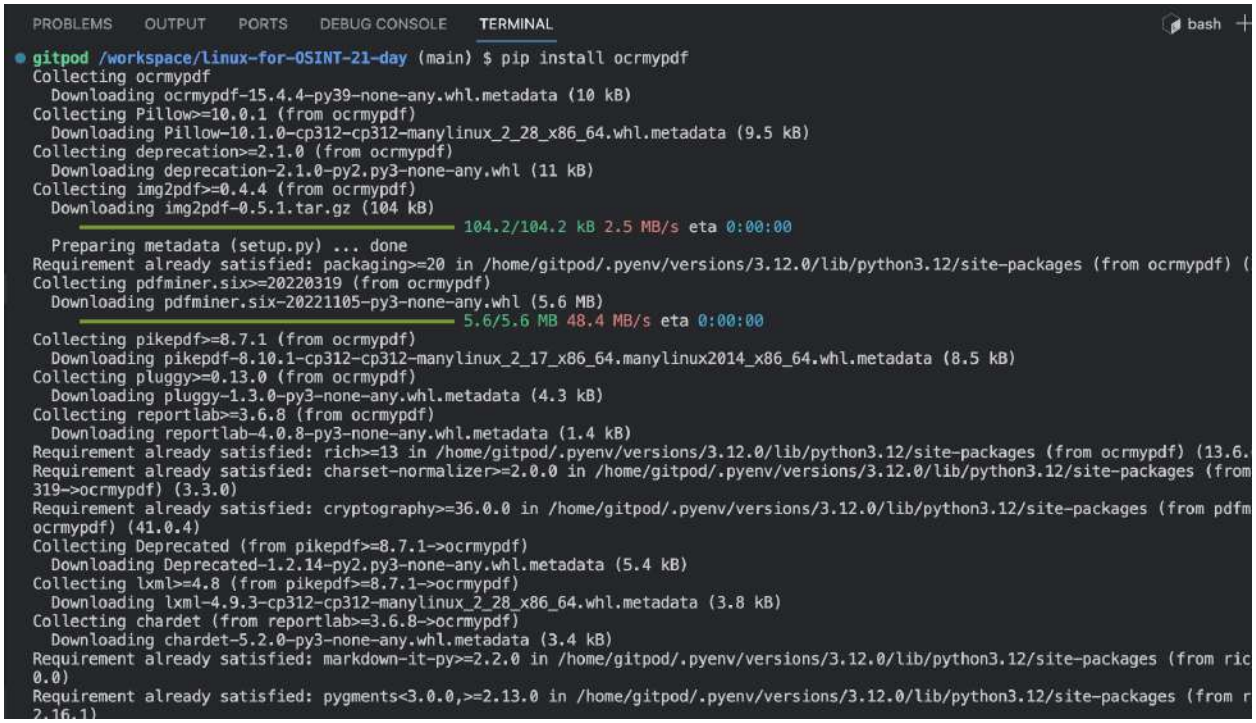
cat text.txt.txt | trans :fr

Codes for different languages can be found on this page: https://github.com/soimort/translate-shell/wiki/Languages.

Today, we explored the easiest-to-use methods for automating media content analysis. But if you have to do it often, I recommend you to learn more about AI tools for image recognition that have appeared in the last six months to a year. Many of them have APIs that can be used to create command line tools.

Day 12. Analyze PDF

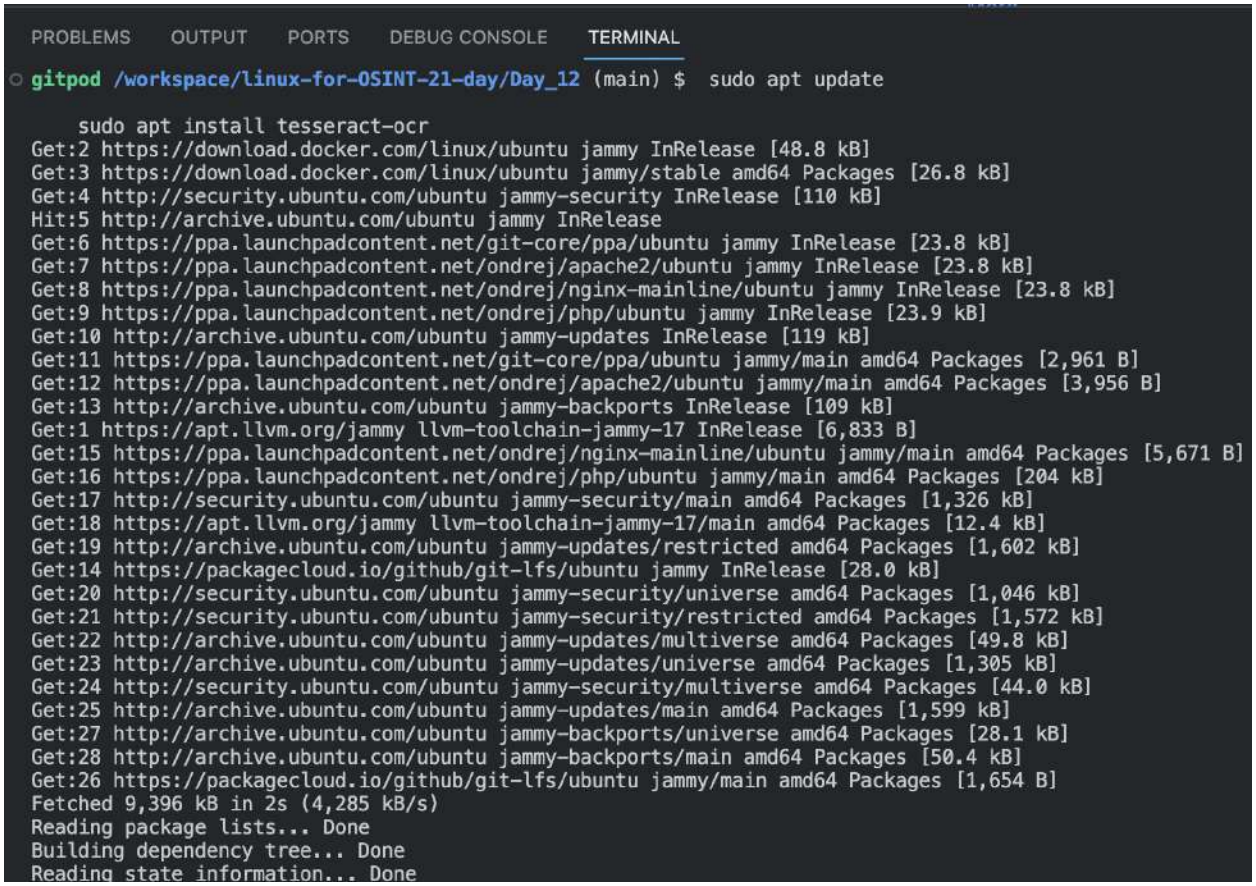
Sometimes investigations have to deal with hundreds of documents and only one of them may contain really important information. Therefore, it is worth trying to automate the analysis of text, images and metadata in PDF.



OCRmyPDF (https://github.com/ocrmypdf/OCRmyPDF) - tool that adds an optical character recognition (OCR) text to scanned PDF files. It makes the text searchable and processable.

Let's install:

pip install ocrmypdf

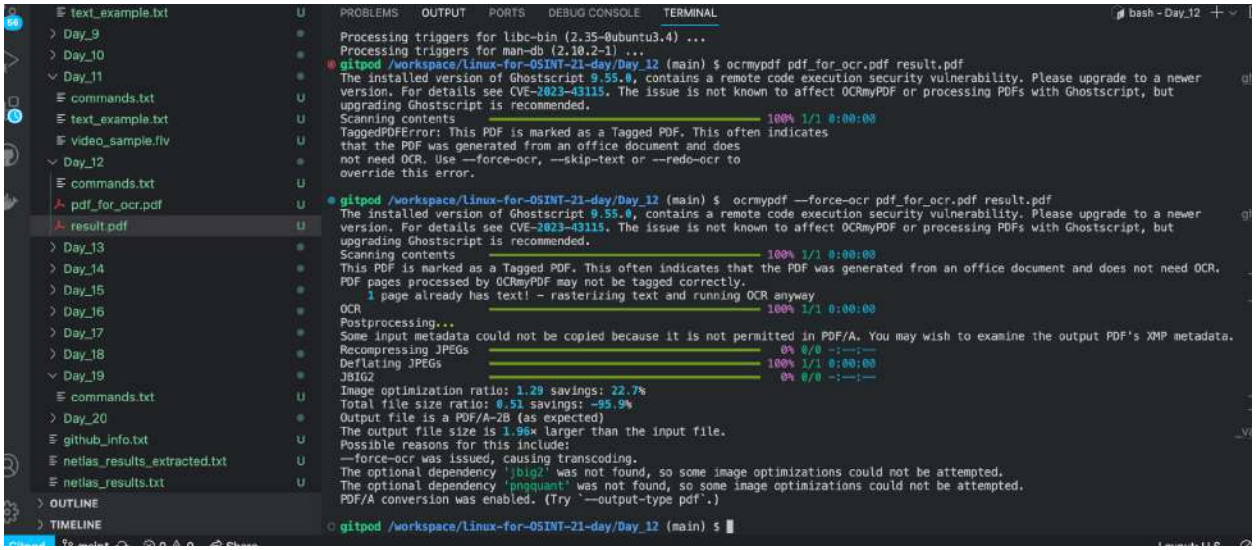


Install Tesseract OCR (https://github.com/tesseract-ocr/tesseract), the text recognition utility that OCRMyPDF is based on.

sudo apt update
sudo apt install tesseract-ocr

Install Ghostscript (https://www.ghostscript.com) is an interpreter for the PostScript® language and PDF files:

sudo apt install ghostscript



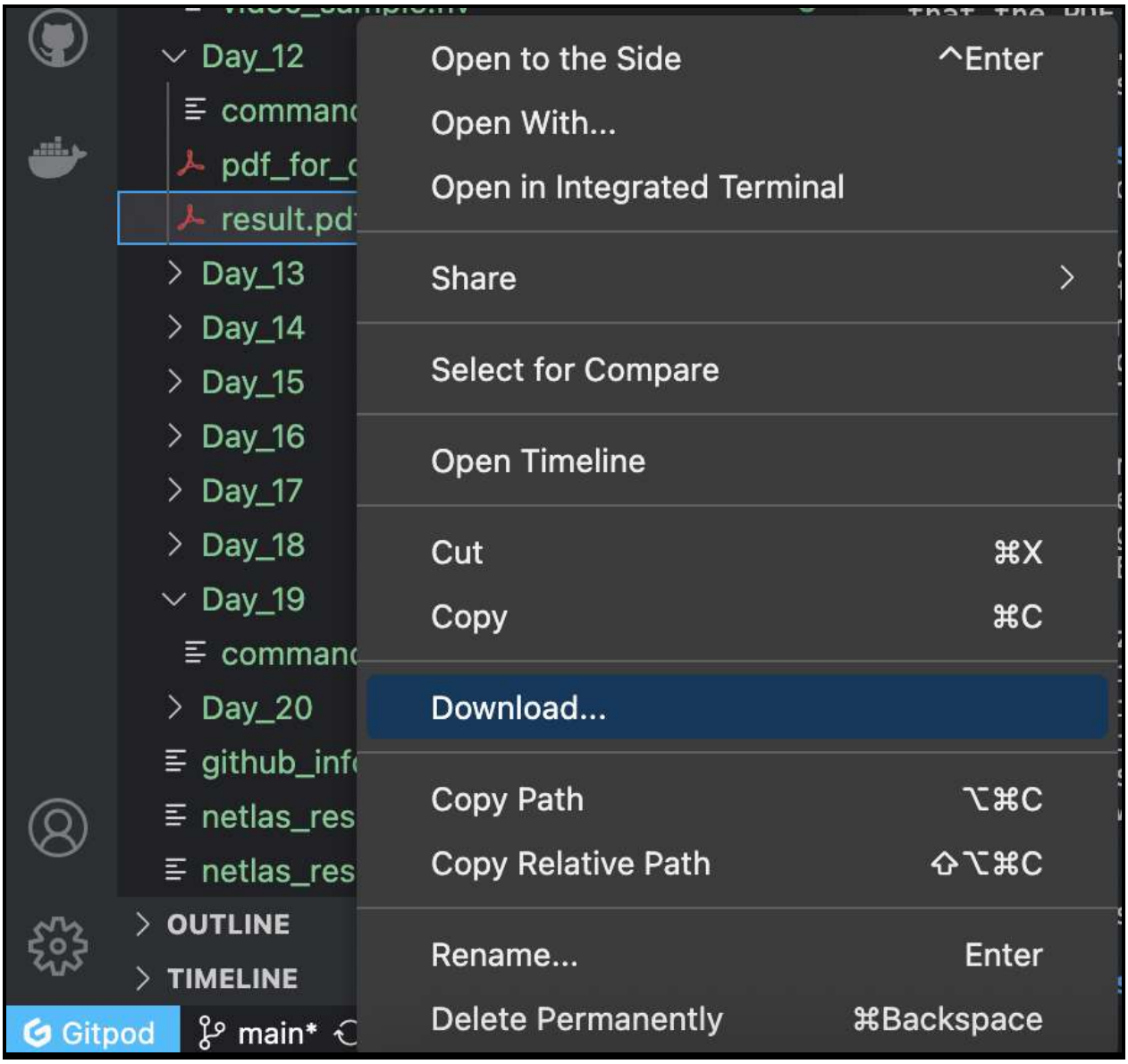
Let's run:

```
cd Day_12
ocrmypdf pdf_for_ocr.pdf result.pdf
```

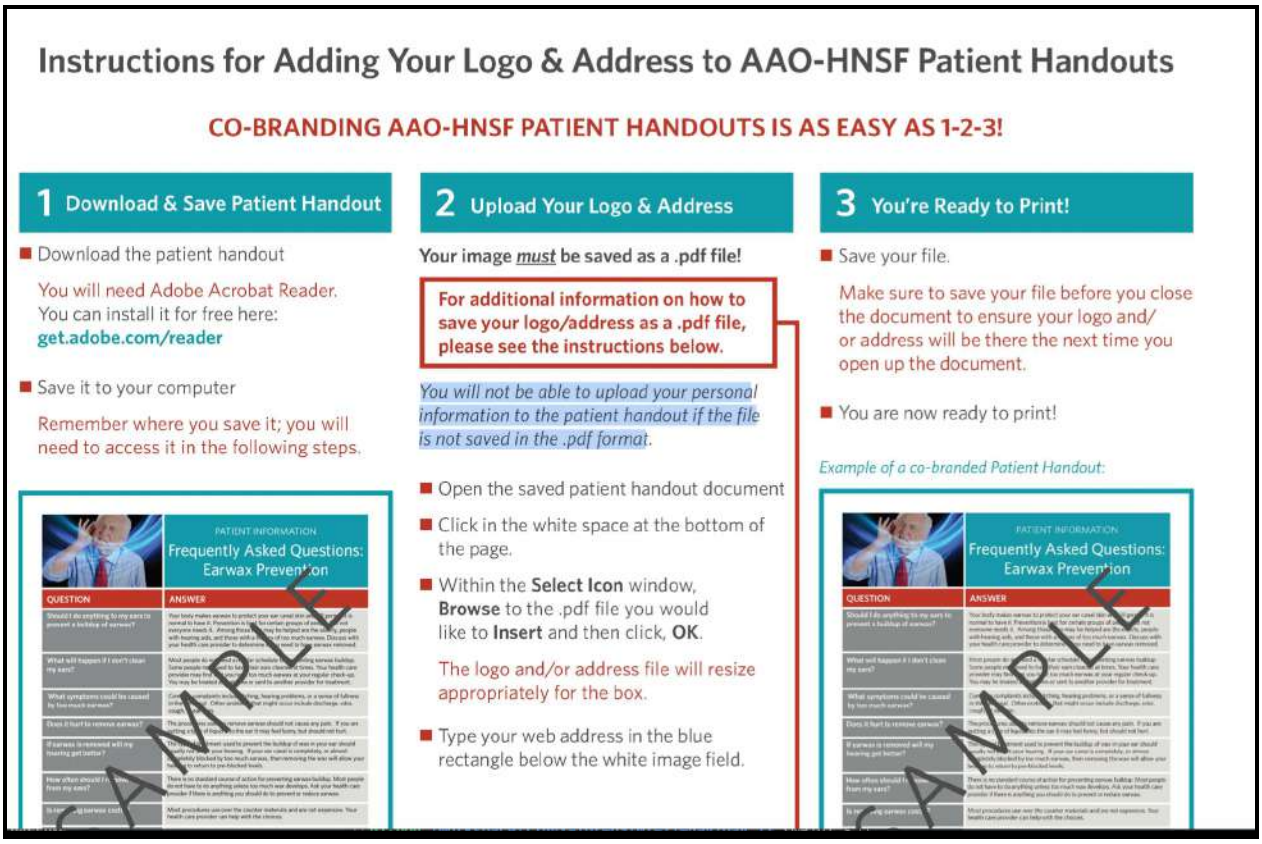
As it happens, I picked up a not-so-great PDF that already has a text layer. So the **--force-ocr** flag is used to create another layer.

```
ocrmypdf –force-ocr pdf_for_ocr.pdf result.pdf
```

The command may take some time to execute, depending on the number of pages in the document.



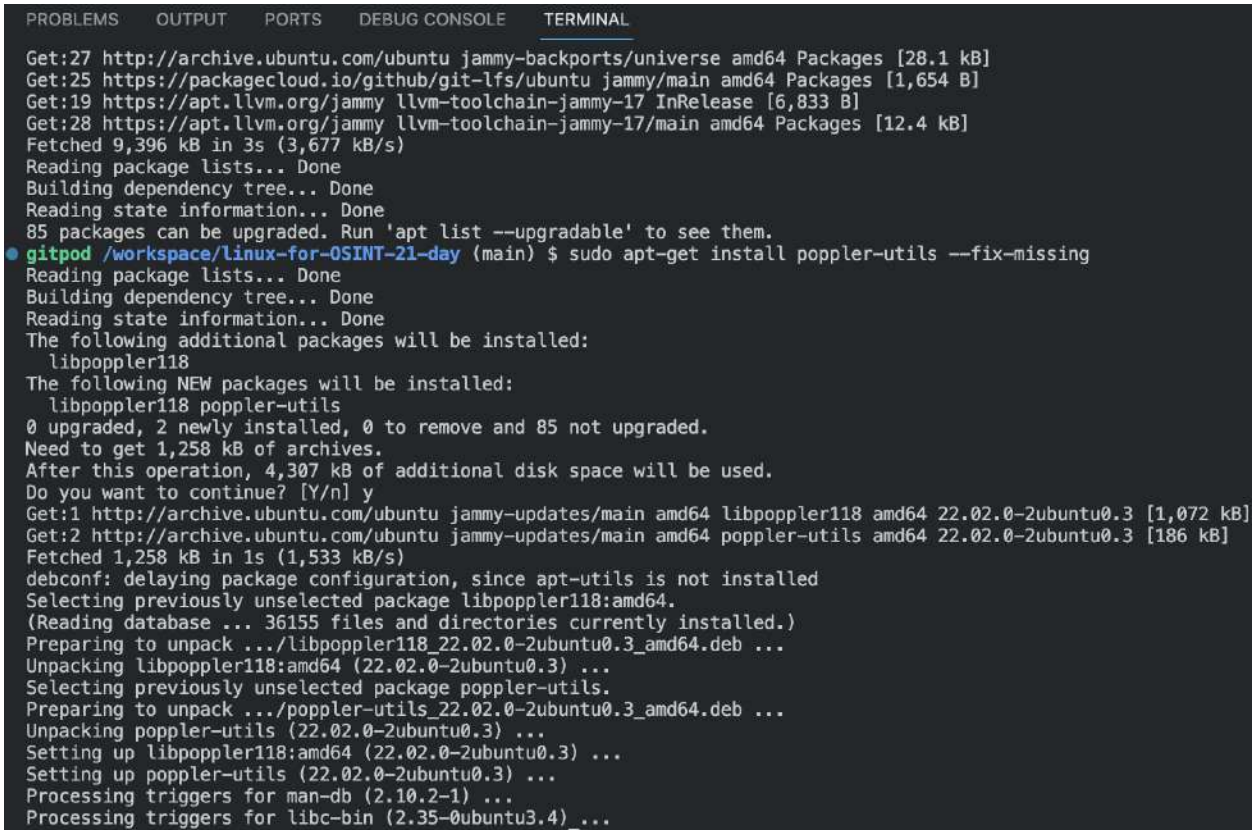
If you use Gitpod, you need to download the PDF document to your computer to view it. Right-click on it and select **Download**.



When viewing a file, the text should be available for selection, copying and pasting.

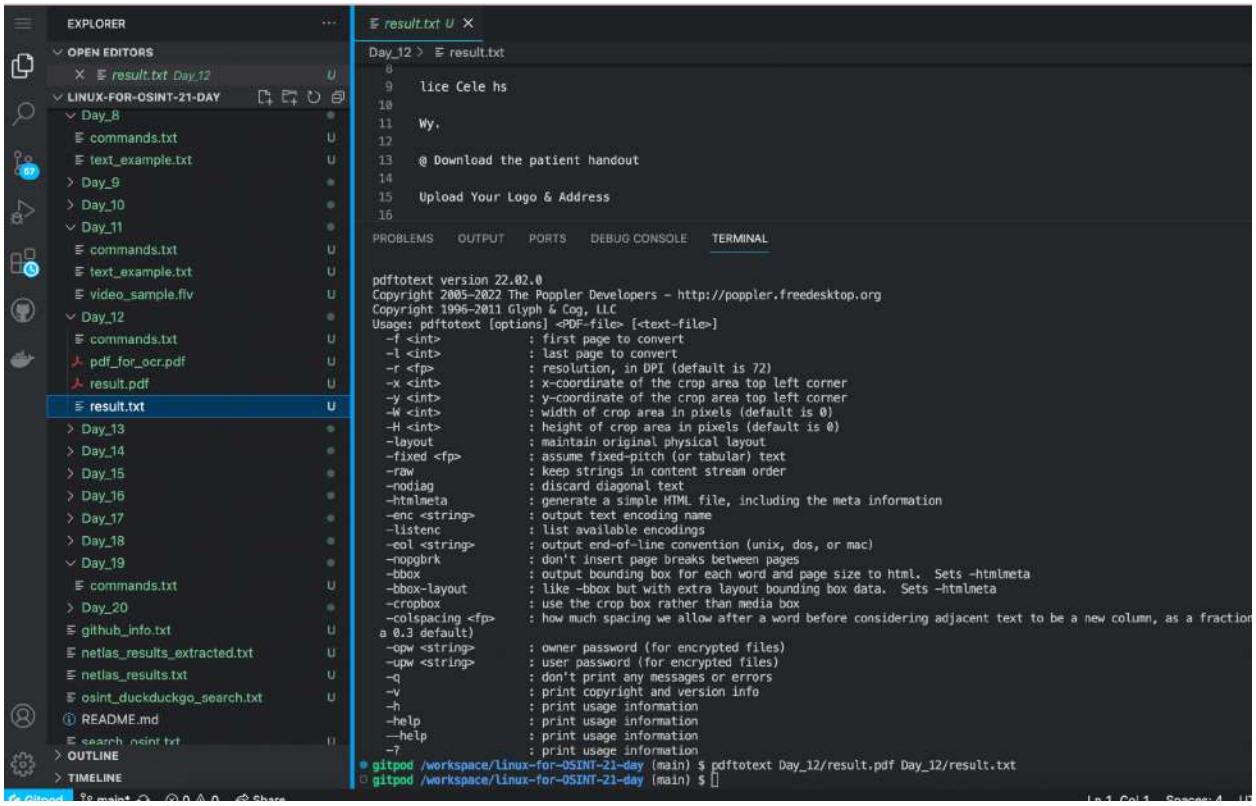
Now let's try to extract the text from the PDF file and save it to a separate file. We will use **Pdftotext** command, part of poppler-utils (https://popper.js.org/docs/v2/utils/).

Extract text from PDF



First, Let's install popper-utils:

sudo apt clean
sudo apt autoclean
sudo apt update
sudo apt-get install poppler-utils --fix-missing

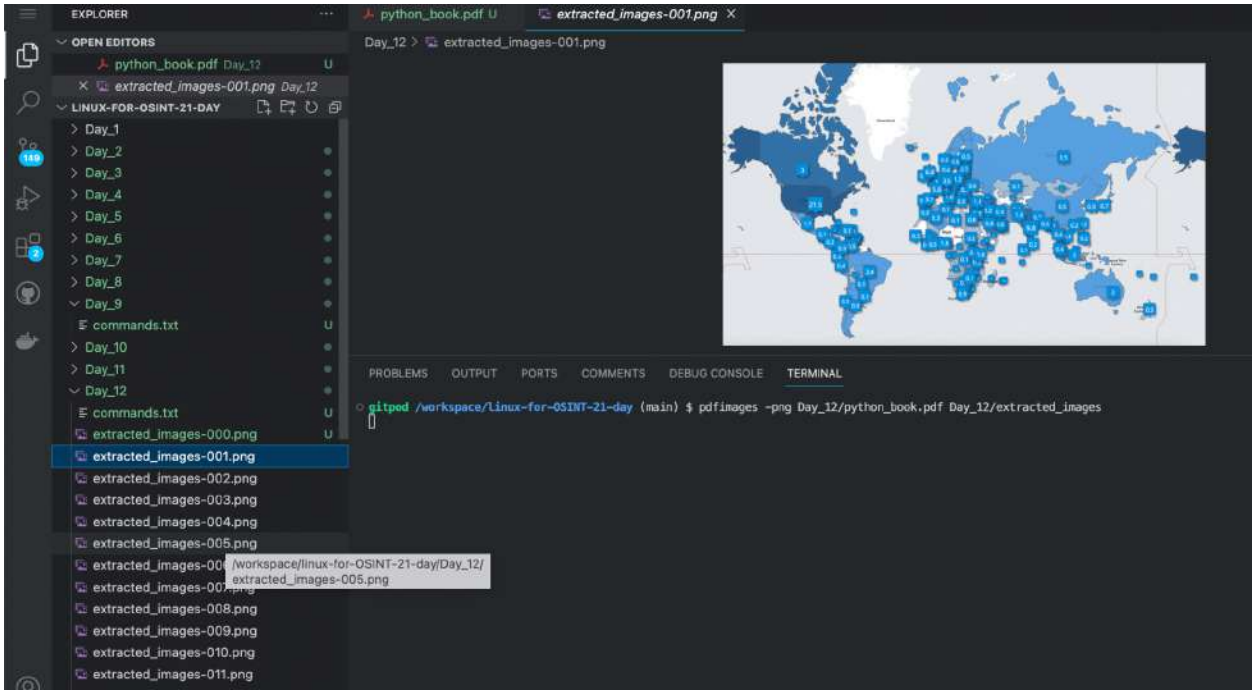


pdftotext result.pdf result.txt

You can now return to **Day 6** and **Day 7** to use the commands described there to work with text extracted from PDF files. For some people, it may be more convenient to skip text extraction and search directly in the PDF using PDFGrep (https://pdfgrep.org/) utility.

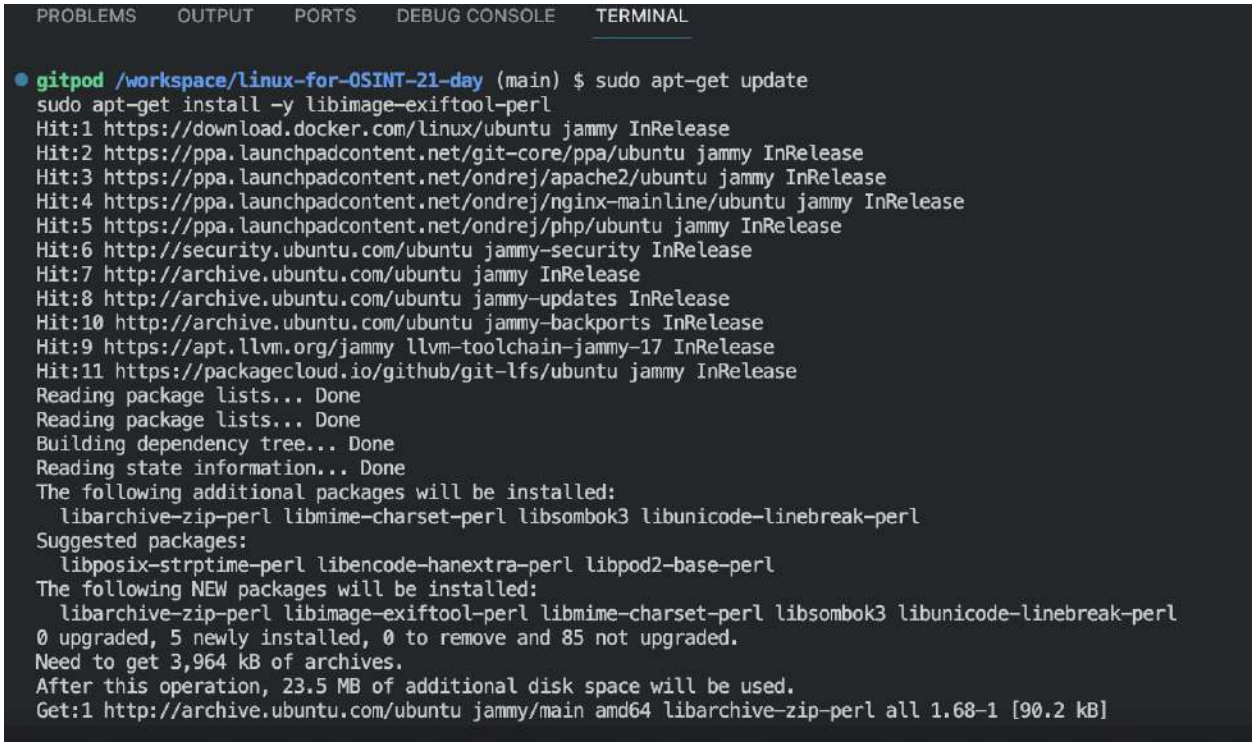
Extract images from PDF

Also **PopperUtils** include **Pdfimages**, which allows you to extract images from PDF documents. Unfortunately, it works far from perfect, but let's give it a try:



pdfimages -png python_book.pdf extracted_images

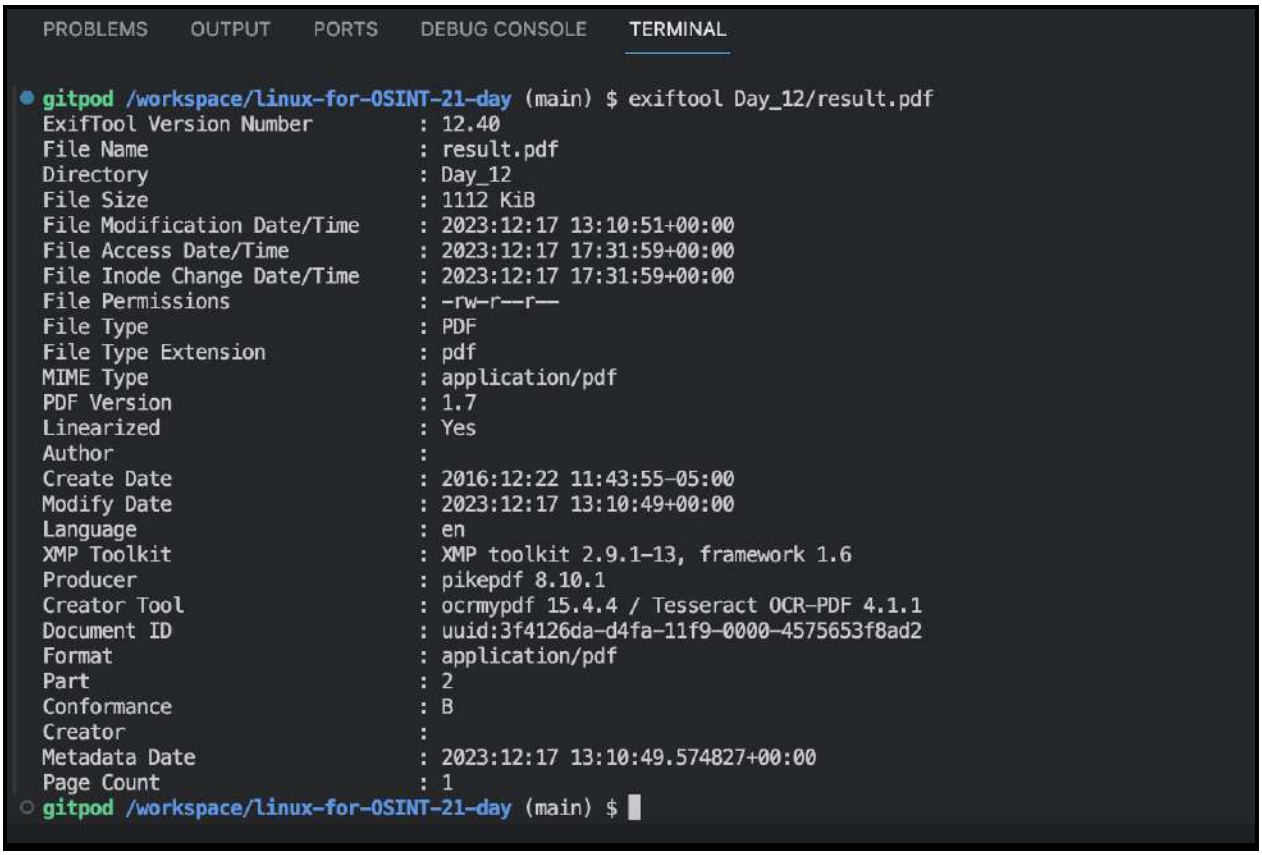
Metadata analyze



ExifTool (https://exiftool.org/) is a Perl library for reading, editing and deleting metadata. According to the list on the official website, it supports 209 file formats!

Let's install:

sudo apt-get update
sudo apt-get install -y libimage-exiftool-perl



And run :

exiftool result.pdf

If you only need certain metadata fields, you can filter them using other Linux utilities such as Grep:

exiftool result.pdf | grep ^Producer

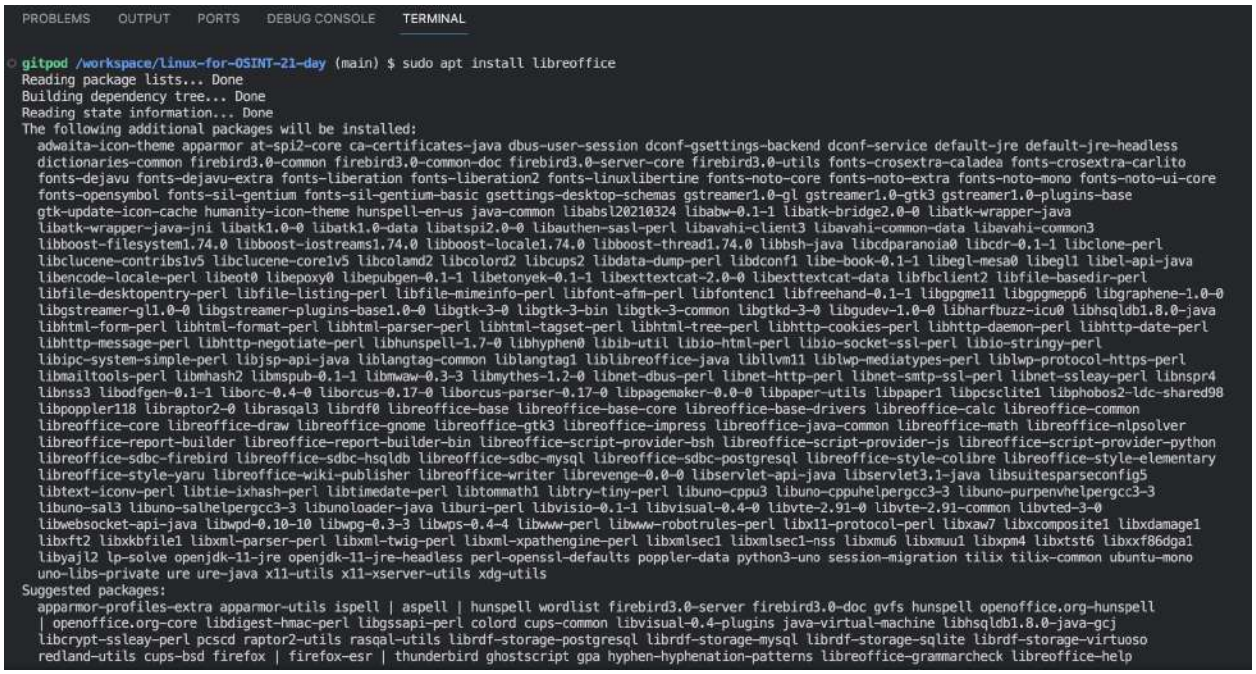
You can also try many other tools to automate work with files (PDF and others formats) metadata:

Metadetective (https://github.com/franckferman/MetaDetective)
ExifLooter (https://github.com/aydinnyunus/exifLooter)
MetaFinder (https://github.com/Josue87/MetaFinder)
PyMeta (https://github.com/m8sec/pymeta)
Hachoir (https://github.com/vstinner/hachoir)

Day 13. MS Office files

LibreOffice (https://www.libreoffice.org) is an open source analog of Microsoft Office. Yes, it is far from perfect and lacks some features, but still it allows you to work very successfully with text documents, spreadsheets, presentations and databases.

And all this can be done not only with the GUI, but also with the command line.



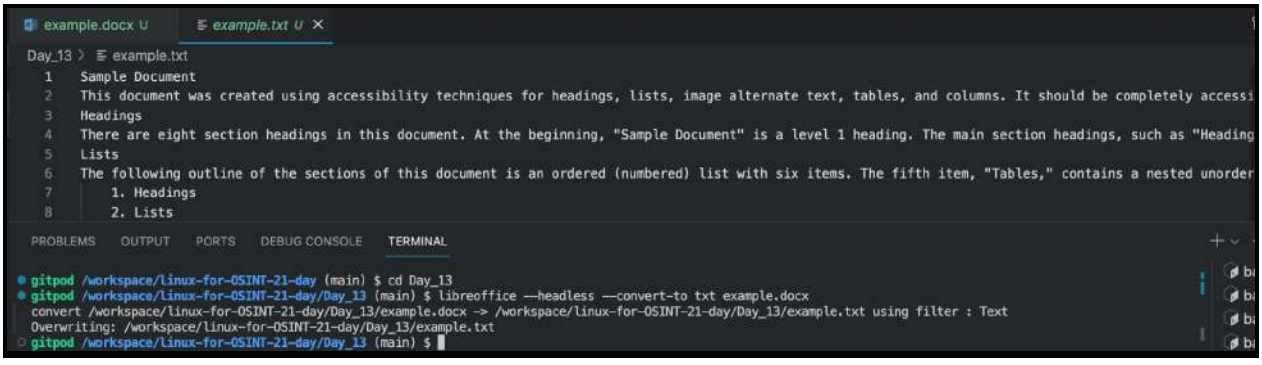
Let's install:

sudo apt install libreoffice

```
PROBLEMS   OUTPUT   PORTS   DEBUG CONSOLE   TERMINAL

gitpod /workspace/linux-for-OSINT-21-day (main) $ libreoffice -h
LibreOffice 7.3.7.2 30(Build:2)

Usage: soffice [argument...]
       argument - switches, switch parameters and document URIs (filenames).

Using without special arguments:
Opens the start center, if it is used without any arguments.
   {file}              Tries to open the file (files) in the components
                       suitable for them.
   {file} {macro:///Library.Module.MacroName}
                       Opens the file and runs specified macros from
                       the file.

Getting help and information:
   --help | -h | -?    Shows this help and quits.
   --helpwriter        Opens built-in or online Help on Writer.
   --helpcalc          Opens built-in or online Help on Calc.
   --helpdraw          Opens built-in or online Help on Draw.
   --helpimpress       Opens built-in or online Help on Impress.
   --helpbase          Opens built-in or online Help on Base.
   --helpbasic         Opens built-in or online Help on Basic scripting
                       language.
   --helpmath          Opens built-in or online Help on Math.
   --version           Shows the version and quits.
   --nstemporarydirectory
                       (MacOS X sandbox only) Returns path of the temporary
                       directory for the current user and exits. Overrides
                       all other arguments.

General arguments:
   --quickstart[=no]   Activates[Deactivates] the Quickstarter service.
   --nolockcheck       Disables check for remote instances using one
                       installation.
   --infilter={filter} Force an input filter type if possible. For example:
                       --infilter="Calc Office Open XML"
                       --infilter="Text (encoded):UTF8,LF,,,"
```

Run the help command to verify that the installation is correct. You will appreciate the huge number of functions this utility has.

libreoffice -h



Now let's convert the docx file to a txt file:

libreoffice --headless --convert-to txt example.docx

libreoffice --headless --convert-to csv example.xlsx

Libre Office includes applications for working with a wide variety of file formats. For example:

- swf, gif
- bmp, jpeg, png, tigg
- svg
- zml, html
- epub, fb2
- wks, wps, wdb
- vsd

And working with all these formats can most likely be automated using the command line. In addition, just as Microsoft allows you to automate documents using VBA or JavaScript, LibreOffice allows you to create automations in LibreOffice Basic (as well as Python).

You can read more about it here:

LibreOffice Basic Programming Guide (https://wiki.documentfoundation.org/Documentation/BASIC_Guide)
LibreOffice Python Design Guide (https://wiki.documentfoundation.org/Macros/Python_Design_Guide/en)

If your job involves creating more reporting documentation, it makes sense for you to read the links above in more detail. With the help of document templates and macros, you can simplify many tasks considerably. But there are other tools besides LibreOffice for this, such as Pandoc.

Creating .docx files

Pandoc (https://pandoc.org/MANUAL.html) is a universal utility for analyzing, converting and generating text documents (web pages, ebooks and more).

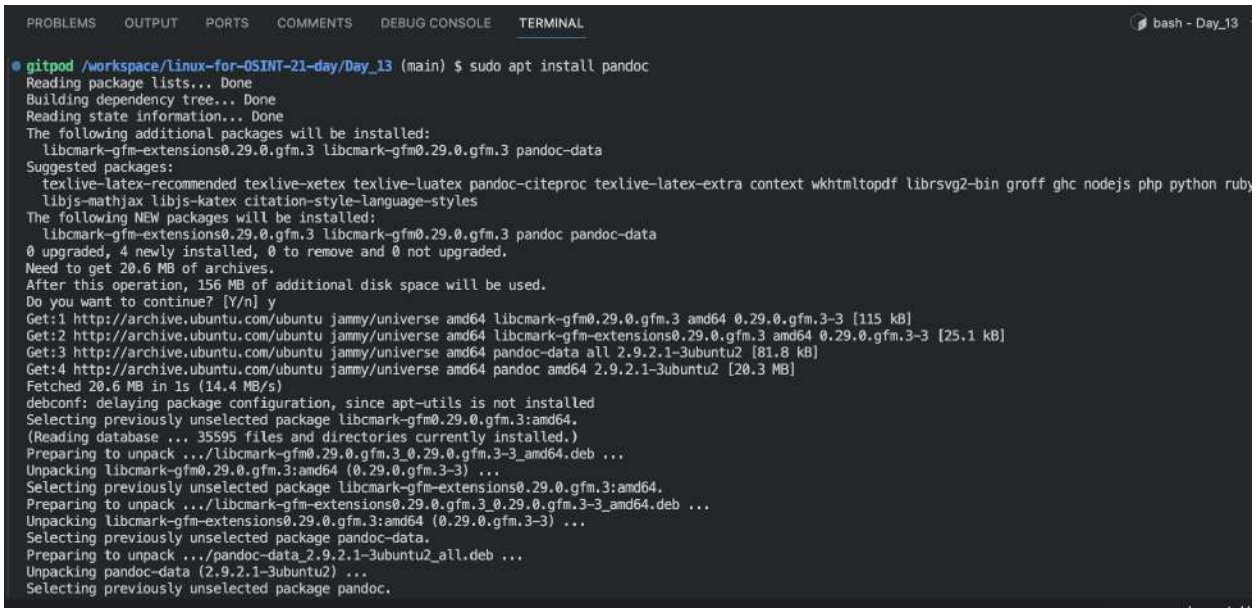
`-f` *FORMAT*, `-r` *FORMAT*, `--from=`*FORMAT*, `--read=`*FORMAT*

Specify input format. *FORMAT* can be:

- `bibtex` (BibTeX bibliography)
- `biblatex` (BibLaTeX bibliography)
- `bits` (BITS XML, alias for `jats`)
- `commonmark` (CommonMark Markdown)
- `commonmark_x` (CommonMark Markdown with extensions)
- `creole` (Creole 1.0)
- `csljson` (CSL JSON bibliography)
- `csv` (CSV table)
- `tsv` (TSV table)
- `docbook` (DocBook)
- `docx` (Word docx)
- `dokuwiki` (DokuWiki markup)
- `endnotexml` (EndNote XML bibliography)
- `epub` (EPUB)
- `fb2` (FictionBook2 e-book)
- `gfm` (GitHub-Flavored Markdown), or the deprecated and less accurate `markdown_github`; use `markdown_github` only if you need extensions not supported in `gfm`.
- `haddock` (Haddock markup)
- `html` (HTML)
- `ipynb` (Jupyter notebook)
- `jats` (JATS XML)
- `jira` (Jira/Confluence wiki markup)
- `json` (JSON version of native AST)
- `latex` (LaTeX)
- `markdown` (Pandoc's Markdown)
- `markdown_mmd` (MultiMarkdown)
- `markdown_phpextra` (PHP Markdown Extra)
- `markdown_strict` (original unextended Markdown)
- `mediawiki` (MediaWiki markup)
- `man` (roff man)
- `muse` (Muse)
- `native` (native Haskell)
- `odt` (ODT)
- `opml` (OPML)
- `org` (Emacs Org mode)
- `ris` (RIS bibliography)
- `rtf` (Rich Text Format)
- `rst` (reStructuredText)
- `t2t` (txt2tags)
- `textile` (Textile)
- `tikiwiki` (TikiWiki markup)
- `twiki` (TWiki markup)
- `typst` (typst)
- `vimwiki` (Vimwiki)
- the path of a custom Lua reader, see Custom readers and writers below

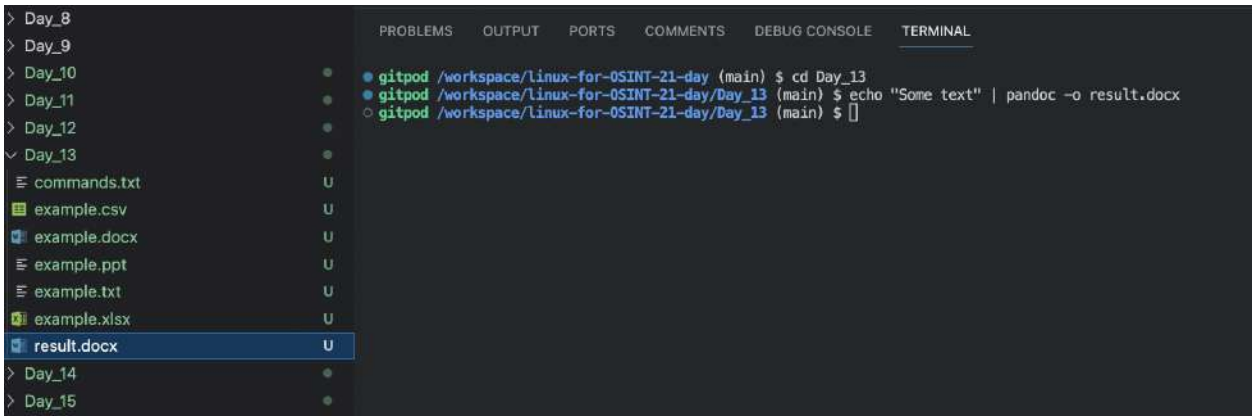
It can work with over 40 different file formats!

Let’s install:

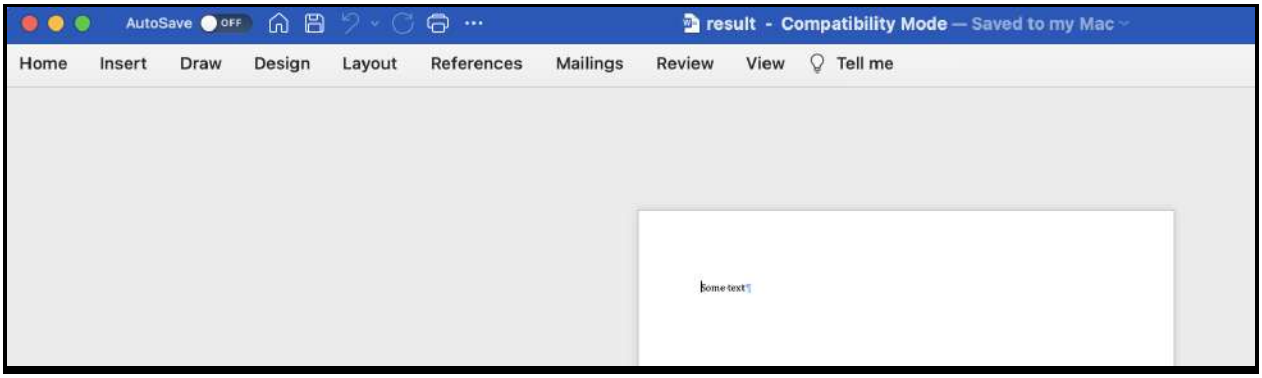


sudo apt install pandoc

Now let's write some text into a docx file:

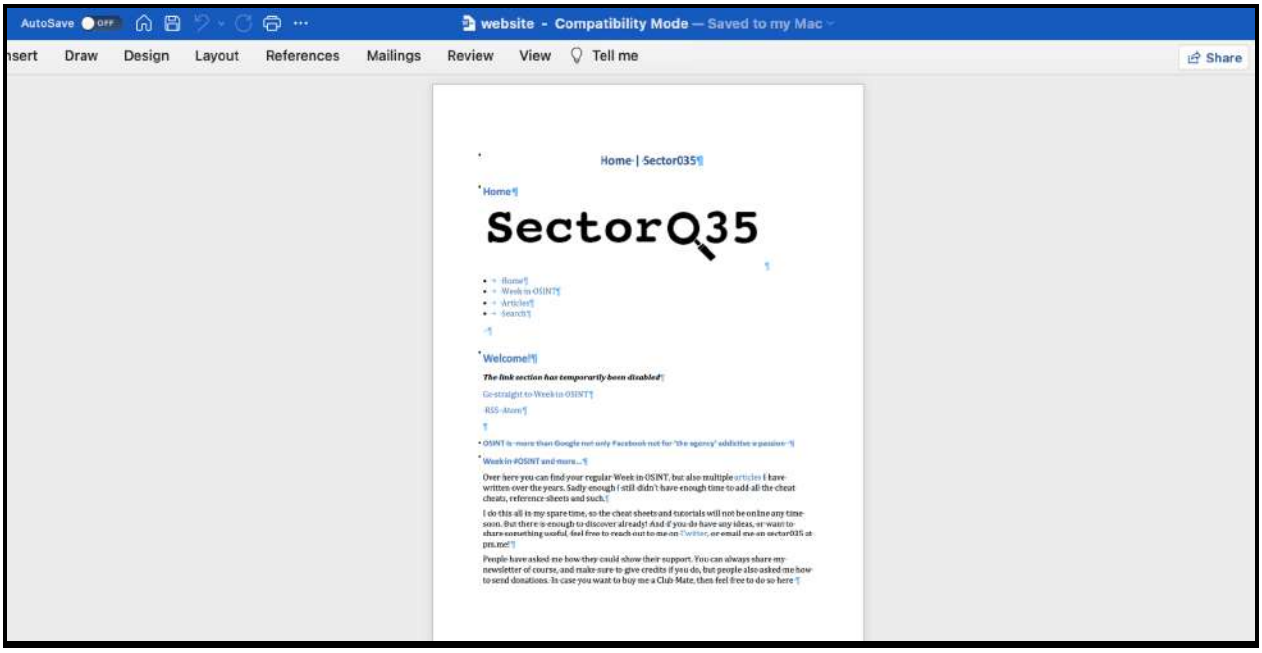


echo "Some text" | pandoc -o result.docx



Similarly, you can save the result of any command to docx.

Another interesting thing is that Pandoc can directly download files from the internet:



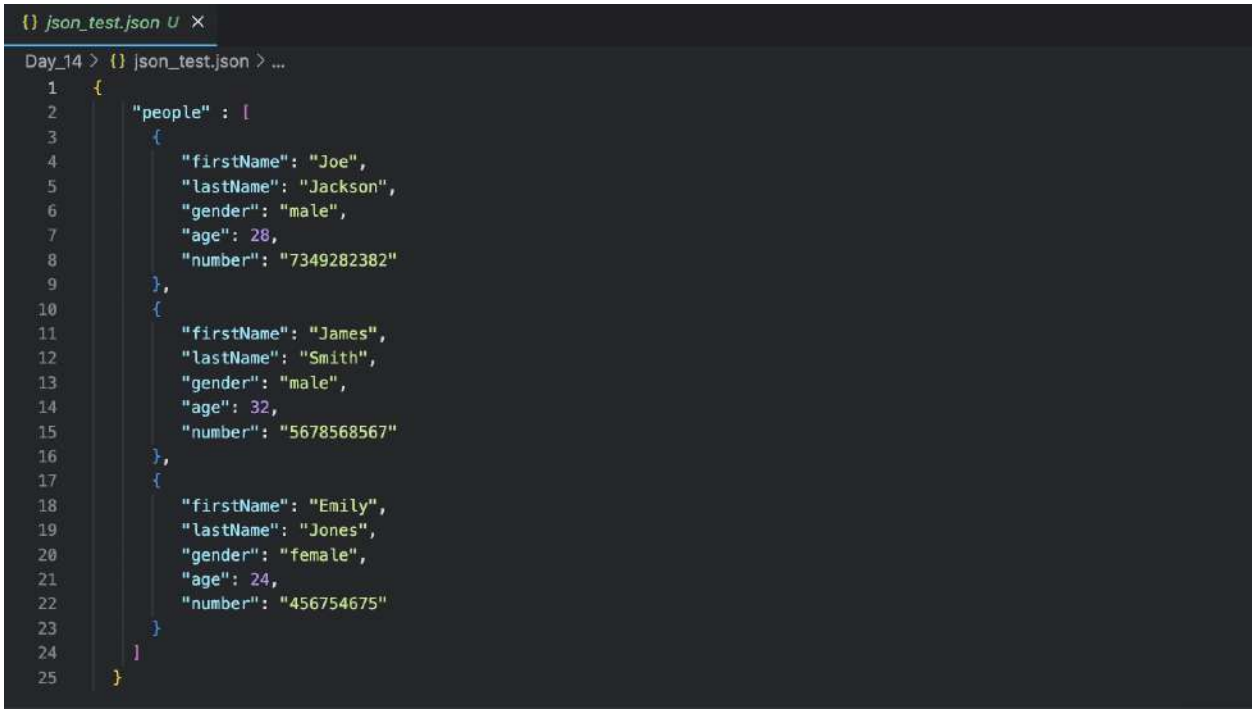
pandoc -f html -t docx https://sector035.nl -o website.docx

The above shows the simplest examples, but Pandoc also allows you to edit document formatting and structure, read metadata, add tables to documents, and more. Like LibreOffice, it is an incredibly versatile tool.

Day 14. JSON, XML, CSV

Today we'll talk about the three popular file formats, which are used in various REST-APIs (you can read more about how to make http requests in Day 5) and as formats for exporting/importing data in various tools.

JSON

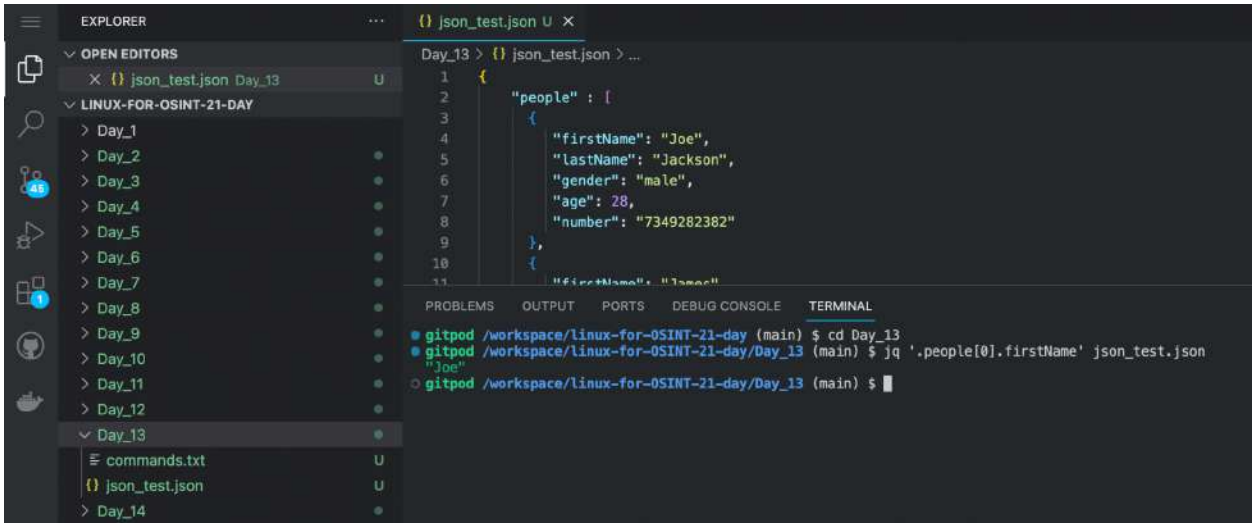


JSON is format for representing structured data based on JavaScript object syntax. That is, data in it are described in the form of objects (example on the picture - **people**), each of which has a set of certain properties (example on the picture - **firstName**, **lastName**, **gender**) and each property can have a certain value (example on the picture - "**Joe**", "**Jackson**", "**male**").

JQ (https://jqlang.github.io/jq/) is one of most popular command line JSON processor. It is installed by default in most Linux distributions.

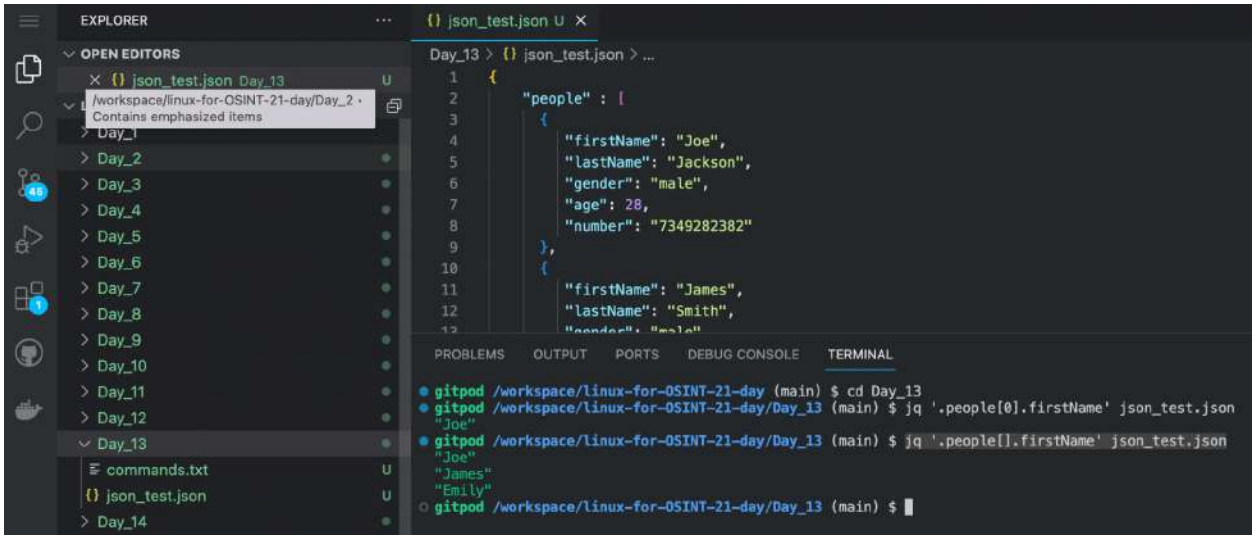
```
PROBLEMS  OUTPUT  PORTS  DEBUG CONSOLE  TERMINAL

gitpod /workspace/linux-for-OSINT-21-day/Day_13 (main) $ jq -h
jq - commandline JSON processor [version 1.6]

Usage:  jq [options] <jq filter> [file...]
        jq [options] --args <jq filter> [strings...]
        jq [options] --jsonargs <jq filter> [JSON_TEXTS...]

jq is a tool for processing JSON inputs, applying the given filter to
its JSON text inputs and producing the filter's results as JSON on
standard output.

The simplest filter is ., which copies jq's input to its output
unmodified (except for formatting, but note that IEEE754 is used
for number representation internally, with all that that implies).

For more advanced filters see the jq(1) manpage ("man jq")
and/or https://stedolan.github.io/jq

Example:

        $ echo '{"foo": 0}' | jq .
        {
                "foo": 0
        }

Some of the options include:
  -c               compact instead of pretty-printed output;
  -n               use `null` as the single input value;
  -e               set the exit status code based on the output;
  -s               read (slurp) all inputs into an array; apply filter to it;
  -r               output raw strings, not JSON texts;
  -R               read raw strings, not JSON texts;
  -C               colorize JSON;
  -M               monochrome (don't colorize JSON);
  -S               sort keys of objects on output;
  --tab            use tabs for indentation;
  --arg a v        set variable $a to value <v>;
```

Let's try to extract the firstName value of the first people object (key):



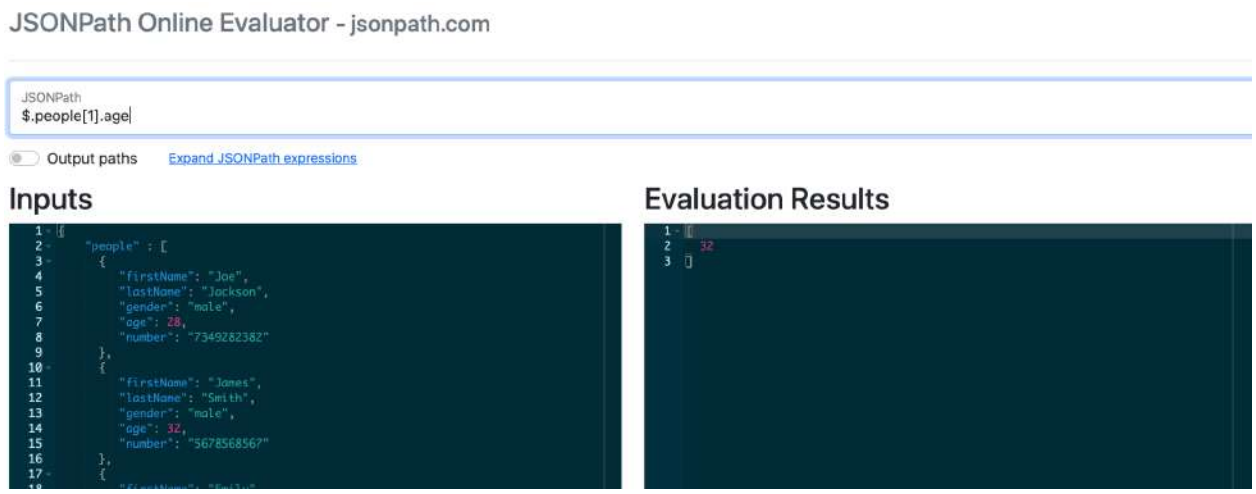
cd Day_14
jq '.people[0].firstName' json_test.json

Now let's display the values of the firstName for all “people” keys:



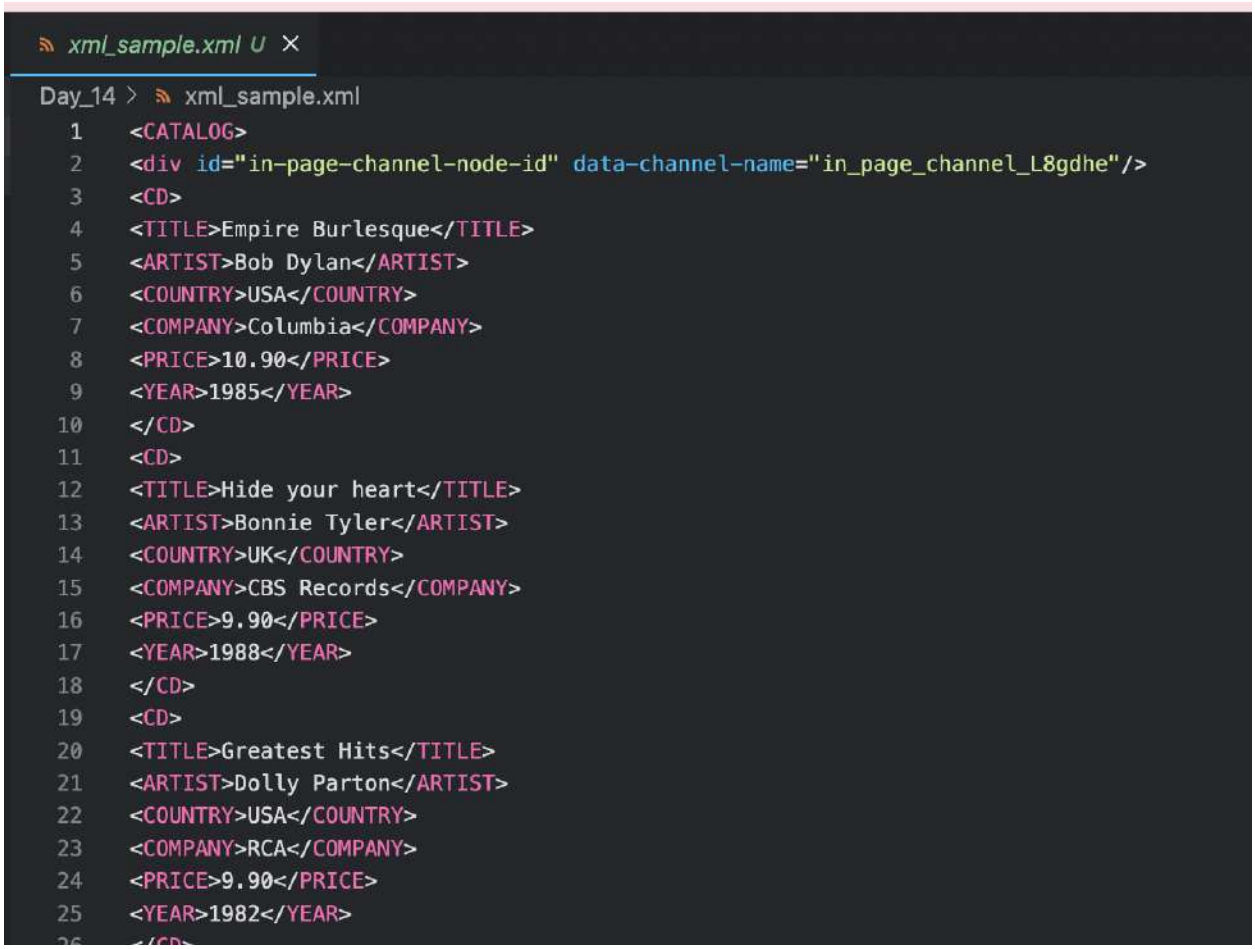
jq '.people[].firstName' json_test.json

Similarly, you can access any values of any properties for all keys.



We used a JSON file with a very simple structure as an example, but in the course of your investigation you may well have to deal with files whose structure will include dozens of different levels. Special online services such as **JSONPath Online Evaluator** (https://jsonpath.com).

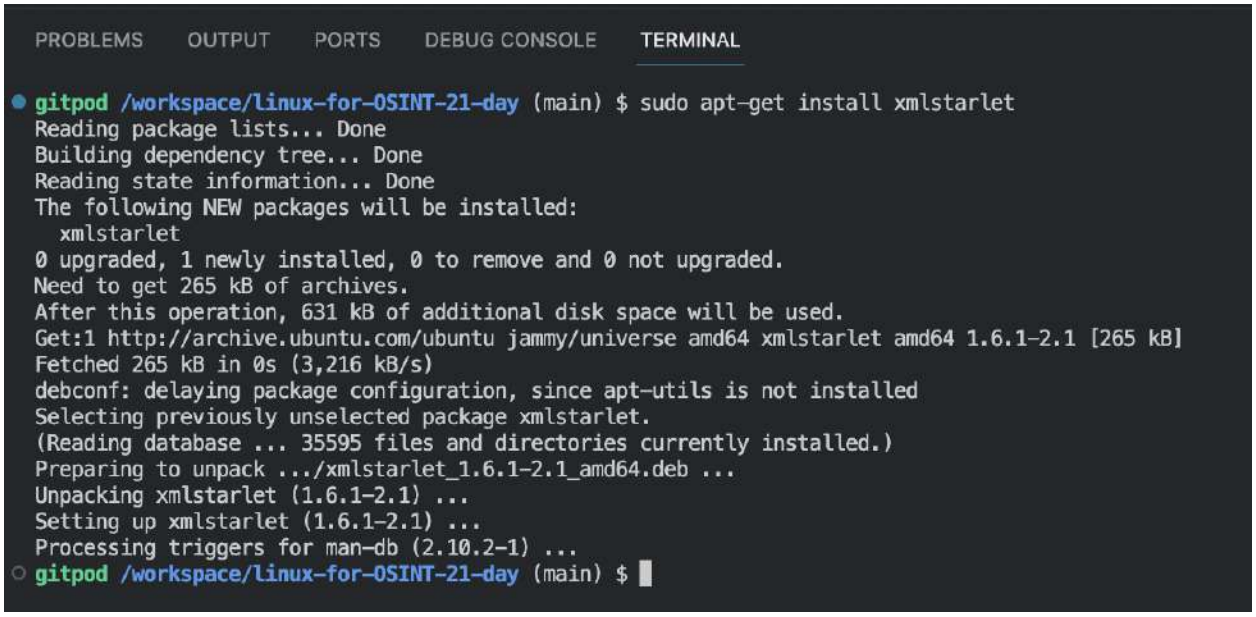
XML



XML (eXtensible Markup Language) is a markup language similar to HTML, but without predefined tags to use (the user can define their own tags).

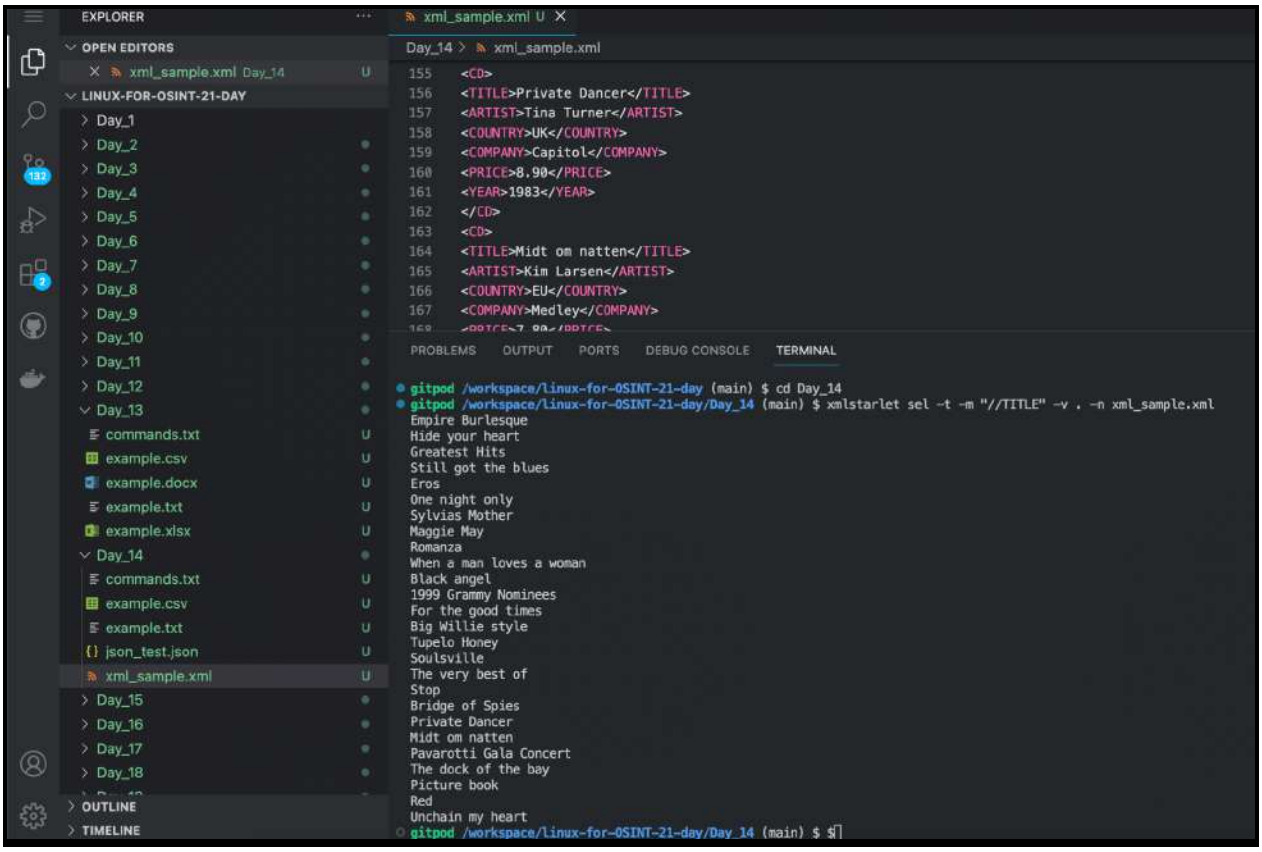
XMLStarlet (https://xmlstar.sourceforge.net/doc/UG/xmlstarlet-ug.html) is a simple cross-platform utility for extracting data from XML files.

Let's install:

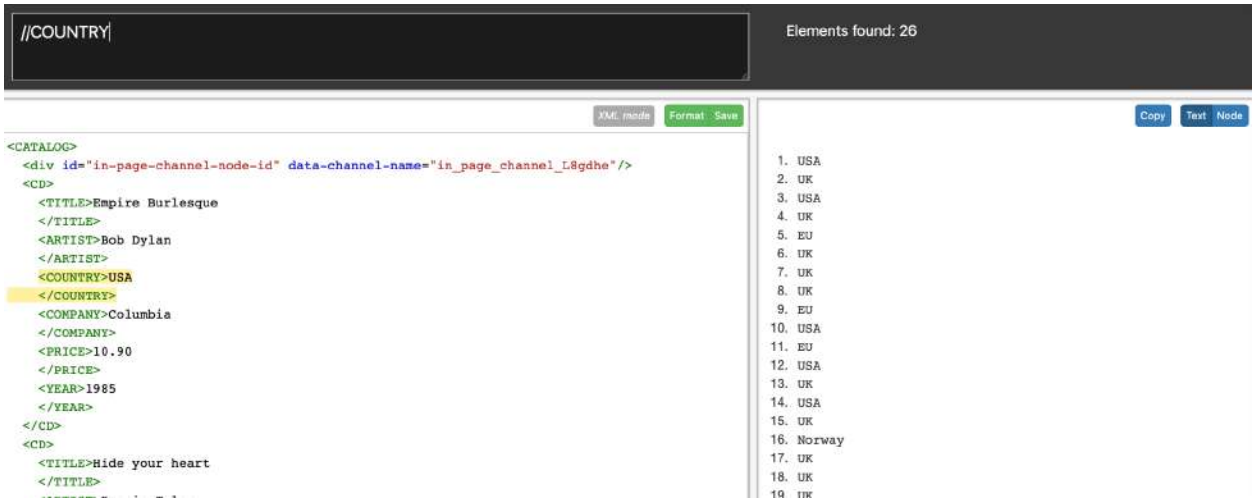


sudo apt-get install xmlstarlet

Now let's display the contents of all **<title>** tags:



xmlstarlet sel -t -m "//TITLE" -v . -n xml_sample.xml

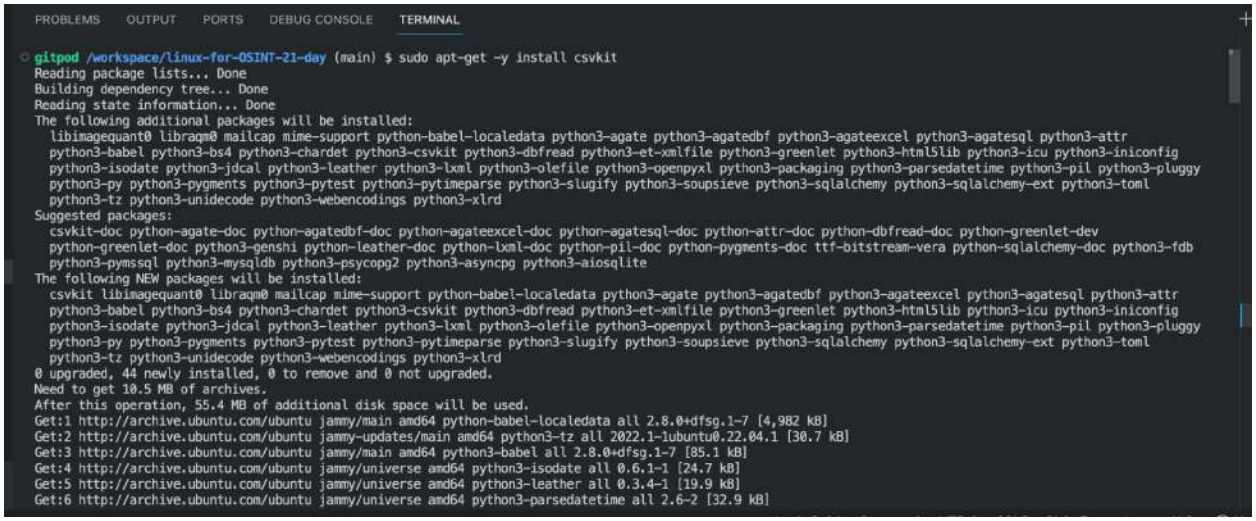


In order to find the path to specific data in an XML document you can use special online services such as XPather (http://xpather.com).

CSV

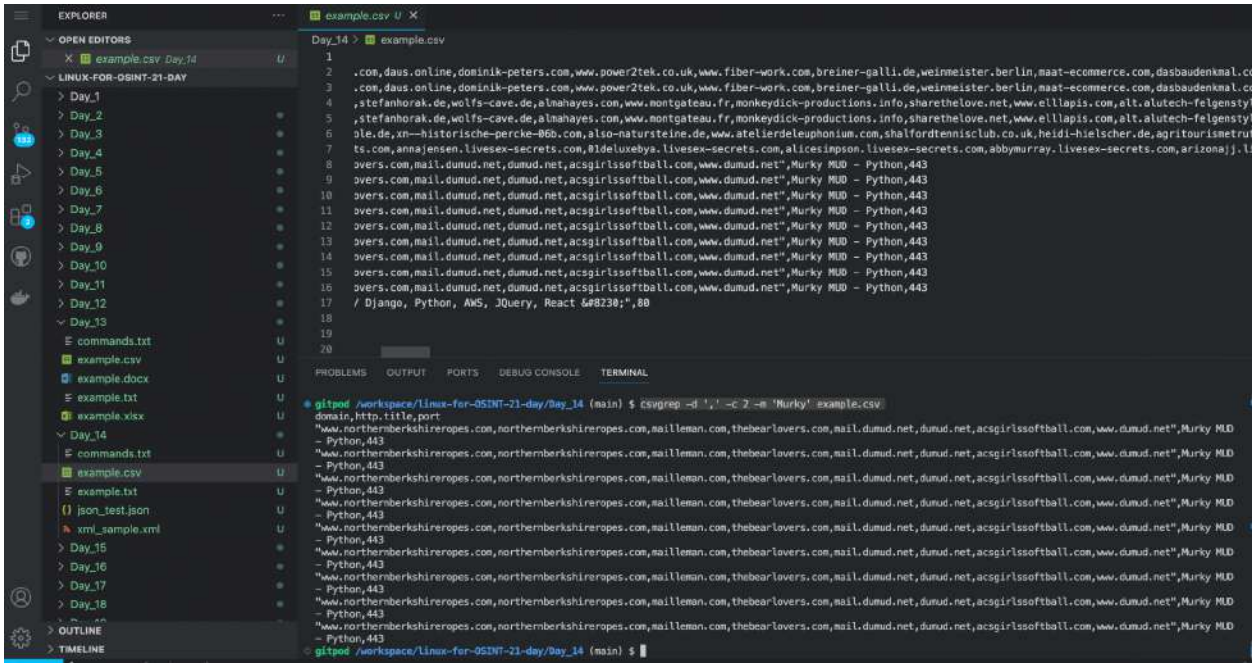
If you happened to miss **Day 6** and **Day 7**, I suggest you get back to it. Knowledge of **Grep**, **Sed** and **Awk** is enough to work efficiently with CSV files. Some users prefer to replace grep with CSVGrep (https://csvkit.readthedocs.io/en/1.0.2/scripts/csvgrep.html). It's a little more convenient for searching specific columns.

Let's install CSVKit (https://csvkit.readthedocs.io/en/1.0.2/index.html):



sudo apt-get -y install csvkit

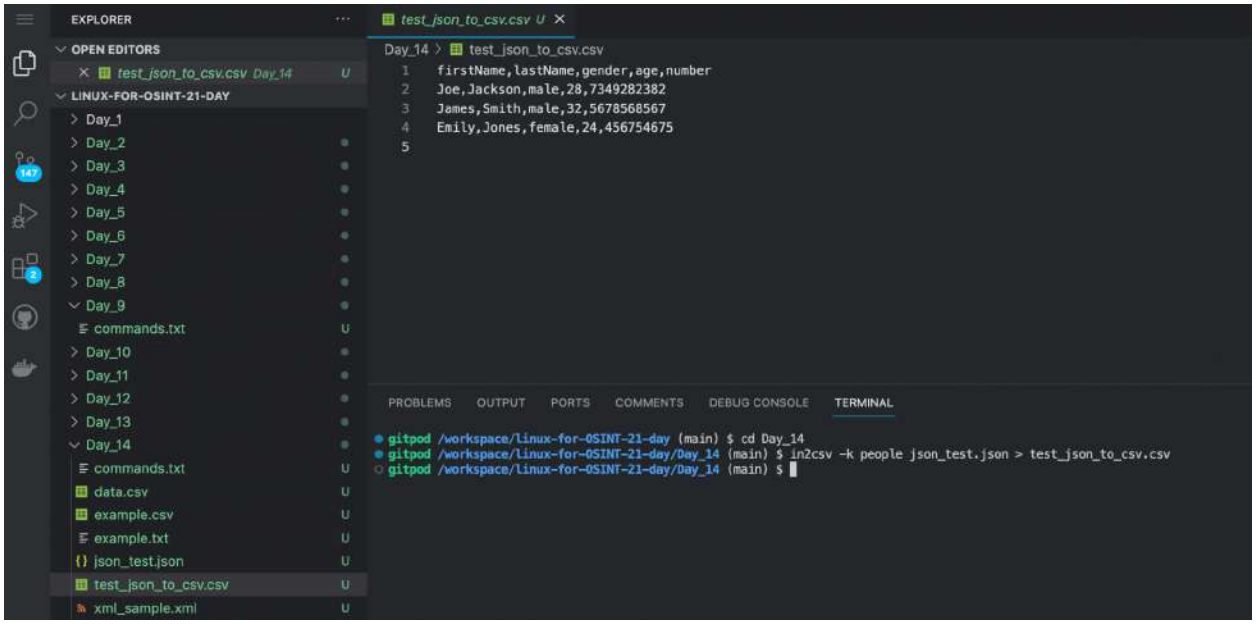
Now let's try to display all rows that contain a certain word in column 2:



csvgrep -d ',' -c 2 -m 'Murky' example.csv

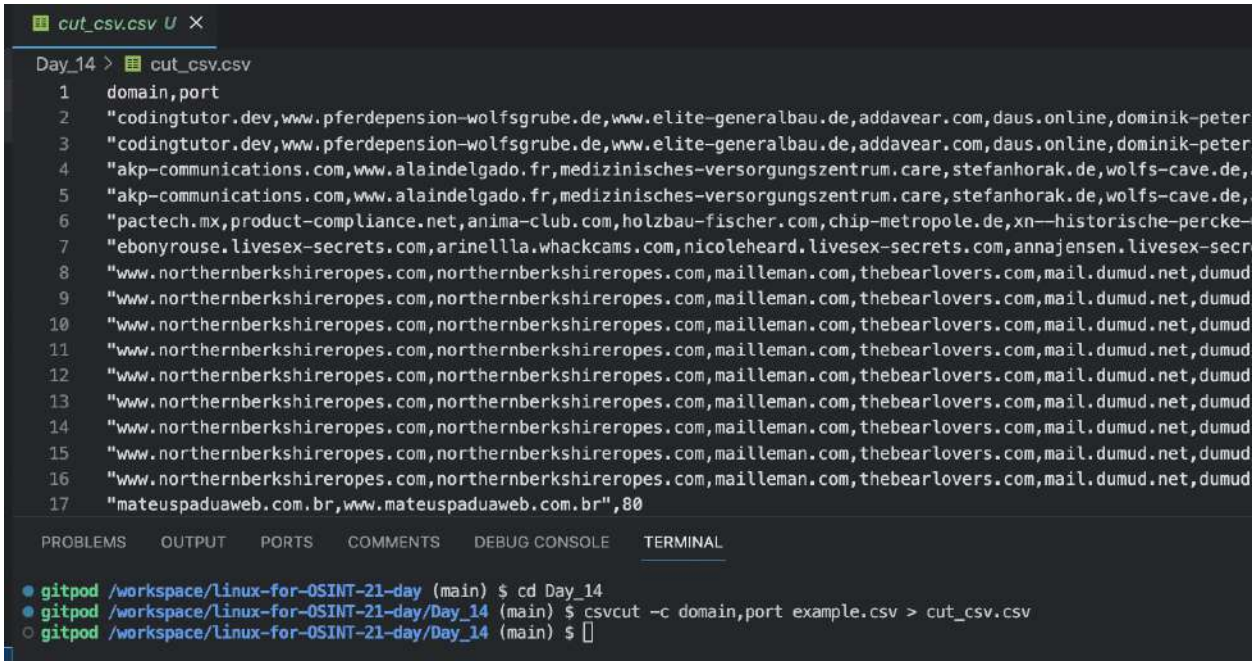
CSVKit also includes many other useful utilities.

In2csv (https://csvkit.readthedocs.io/en/latest/scripts/in2csv.html) - convert any files to CSV.



in2csv -k people json_test.json > test_json_to_csv.csv

Csvcut (https://csvkit.readthedocs.io/en/latest/scripts/csvcut.html) is a tool for filter and truncate CSV files.



```
csvcut -c domain,port example.csv > cut_csv.csv
```

It is also worth paying attention to:

Csvjoin (https://csvkit.readthedocs.io/en/latest/scripts/csvjoin.html)

Csvsort (https://csvkit.readthedocs.io/en/latest/scripts/csvsort.html)

Csvstat (https://csvkit.readthedocs.io/en/latest/scripts/csvstat.html).

When working with data in XML or JSON format, I recommend that you consider converting it to CSV. In many cases, this format is more convenient to understand and work in the command line (as mentioned above, you can easily do it with **in2csv**).

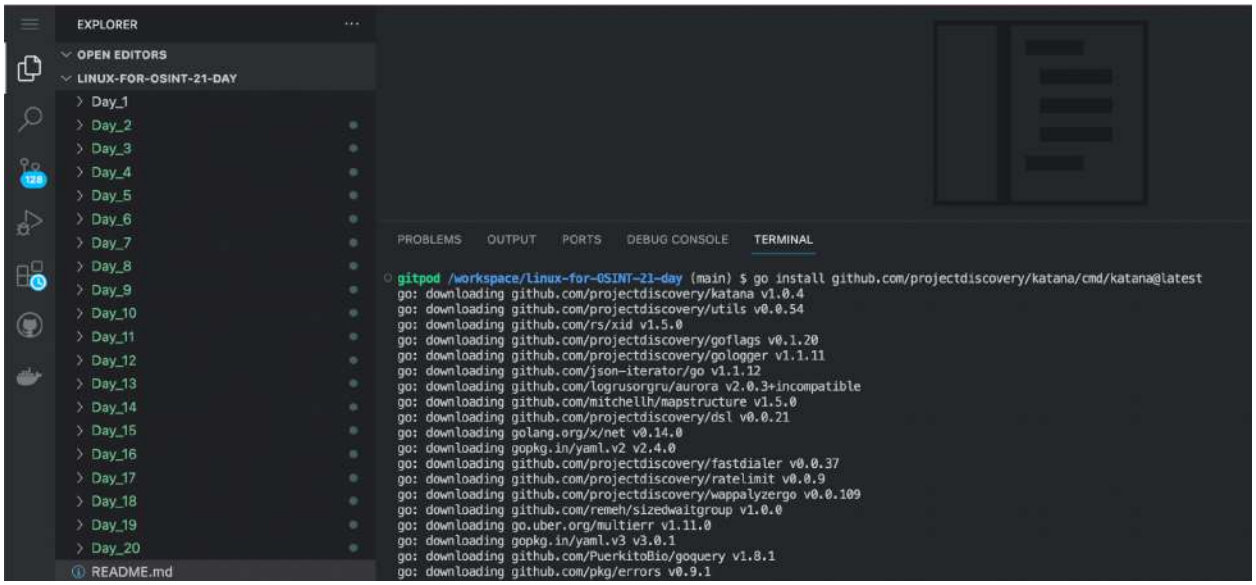
Day 15. Scraping

Scraping is the process of automatically extracting different data from web pages. If you are good at using **curl**, **grep**, **sed** and **awk** mentioned in the previous days, you can already successfully scrape web pages on Linux. But it's worth exploring a few more tools to be more effective.

Katana

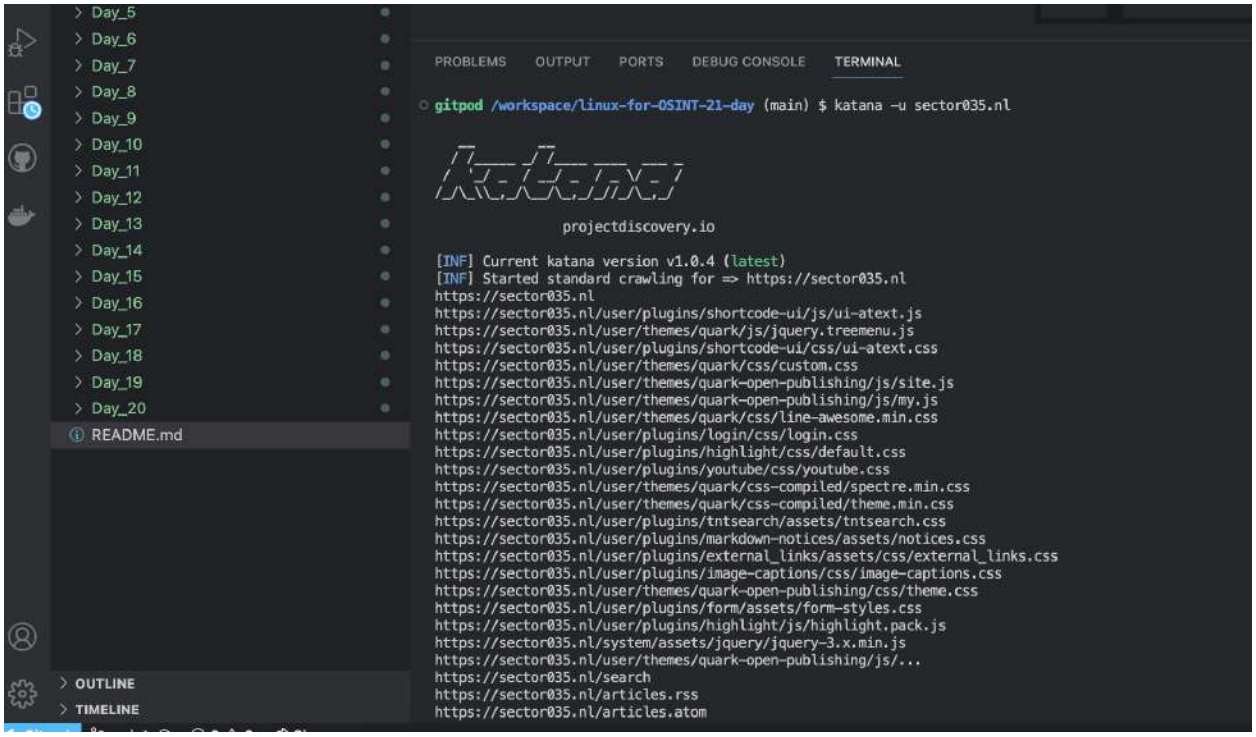
First of all, let's learn how to make a list of URLs of all pages of the site (or rather, all pages that are linked from other pages of the site, starting with the main page).

This is very easy if you have the wonderful Katana (https://github.com/projectdiscovery/katana) tool from Project Discovery (https://github.com/projectdiscovery).



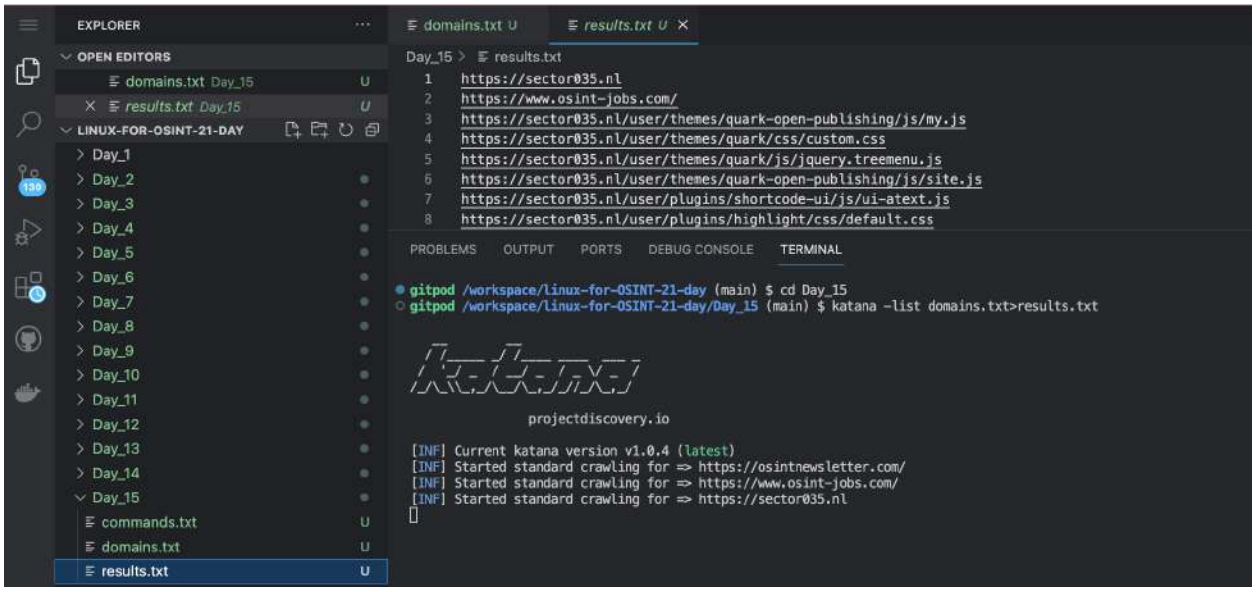
Let's install:

go install github.com/projectdiscovery/katana/cmd/katana@latest



And run:

katana -u sector035.nl



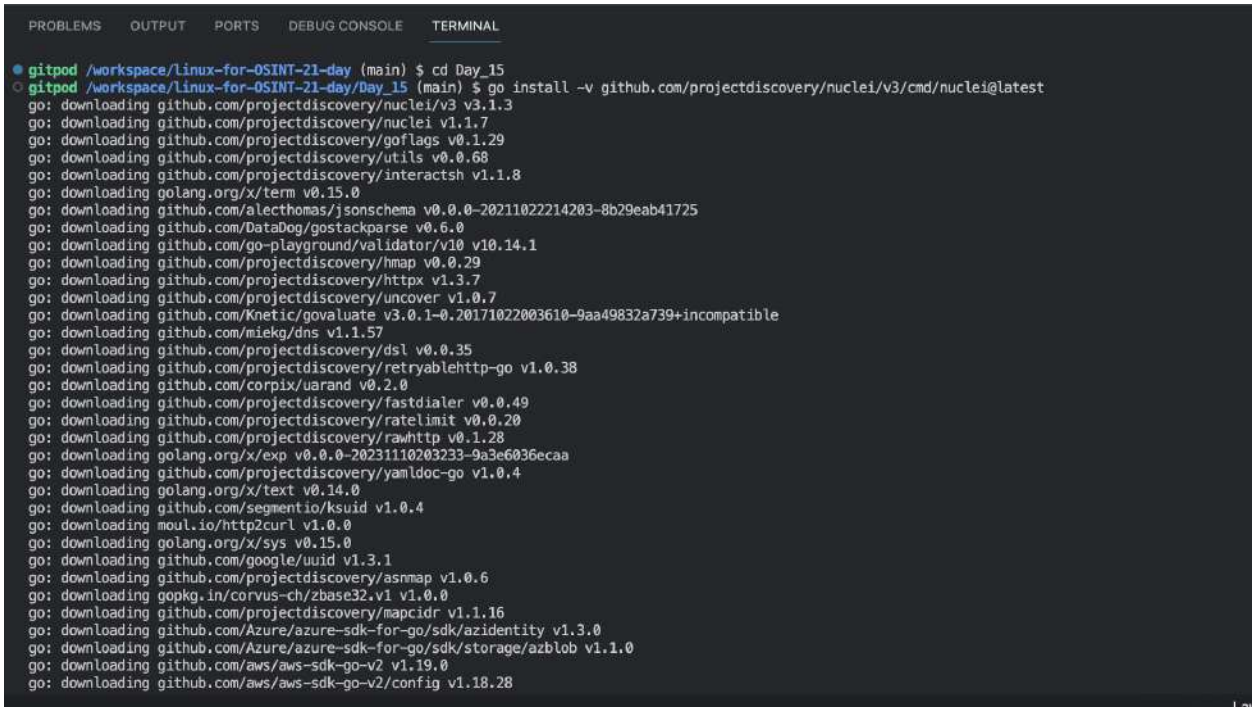
If you want to collect links from a whole list of domains, use the -list flag. And let's save the result this time, to a text file, so that we can use it when we run the next command:

```
cd Day_15
katana -list domains.txt>results.txt
```

NUCLEI

Nuclei (https://github.com/projectdiscovery/nuclei) is another great tool from Project Discovery. It is primarily designed to scan sites for vulnerabilities, but it also does an excellent job of collecting various investigative data from web pages.

Let's install:



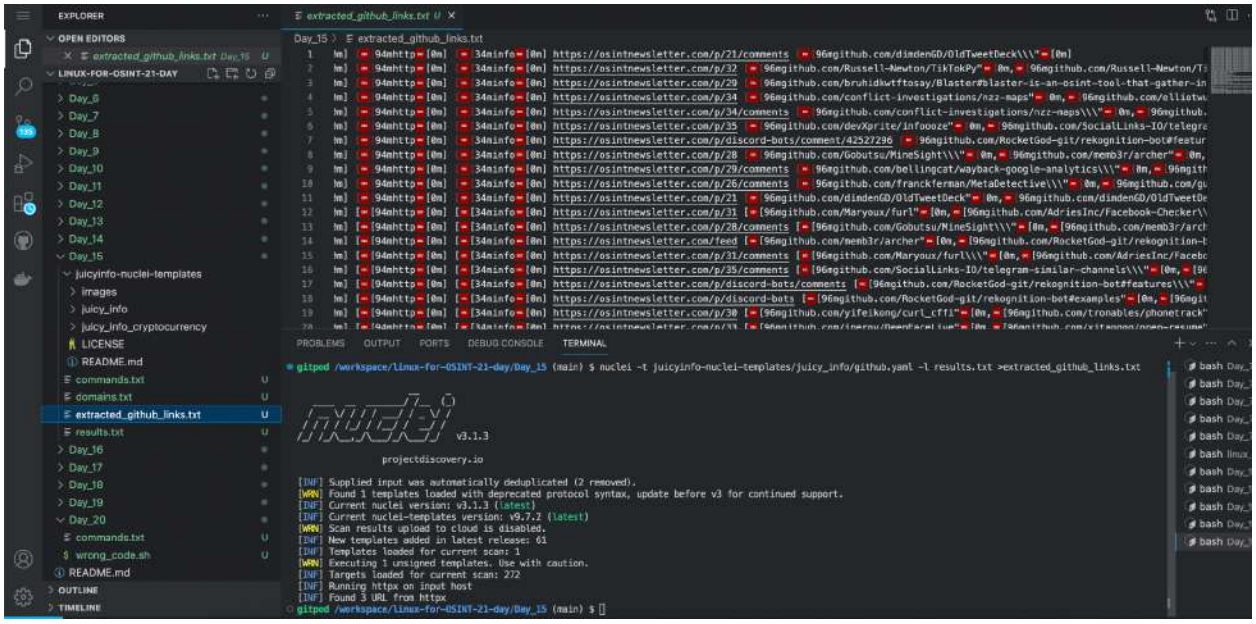

```
go install -v github.com/projectdiscovery/nuclei/v3/cmd/nuclei@latest
```

Clone repo "Juicy info Nuclei templates" (we'll talk more about the git command on day 19):

```
gitpod /workspace/linux-for-OSINT-21-day/Day_15 (main) $ git clone https://github.com/cipher387/juicyinfo-nuclei-templates
Cloning into 'juicyinfo-nuclei-templates'...
remote: Enumerating objects: 173, done.
remote: Counting objects: 100% (73/73), done.
remote: Compressing objects: 100% (71/71), done.
remote: Total 173 (delta 41), reused 0 (delta 0), pack-reused 100
Receiving objects: 100% (173/173), 14.55 MiB | 20.96 MiB/s, done.
Resolving deltas: 100% (85/85), done.
gitpod /workspace/linux-for-OSINT-21-day/Day_15 (main) $
```

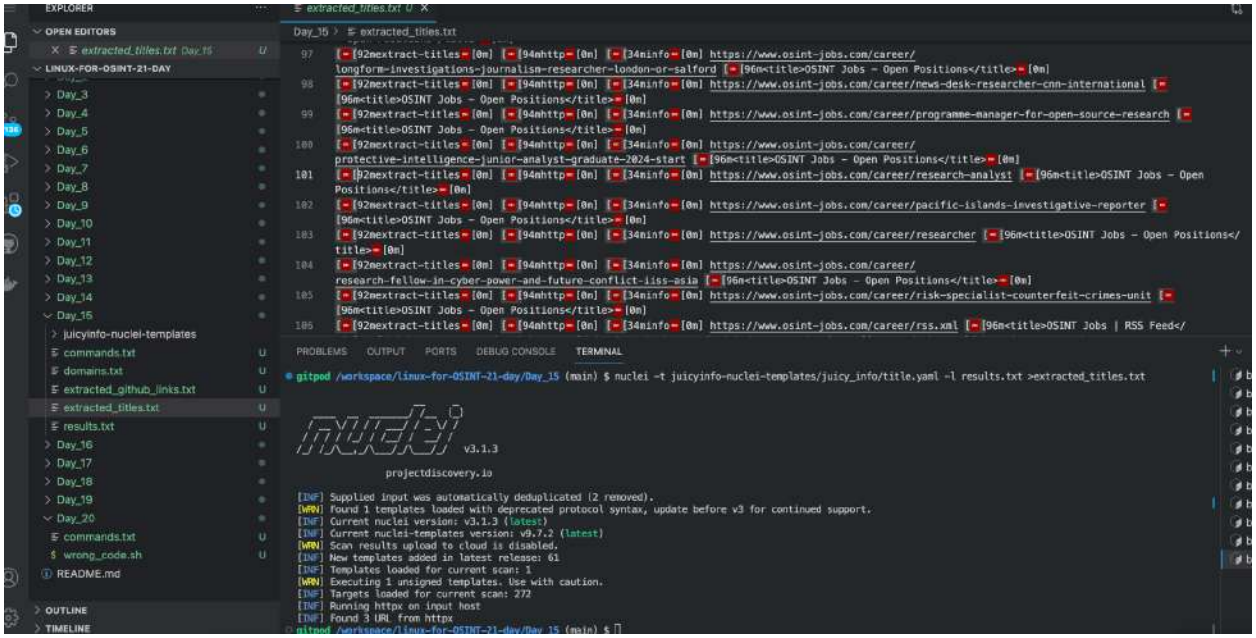
git clone https://github.com/cipher387/juicyinfo-nuclei-templates

Now, let's scan through all the links we've collected using Katana and extract links that contain the word Github in them from the pages (we'll use the github.yaml template to do this):



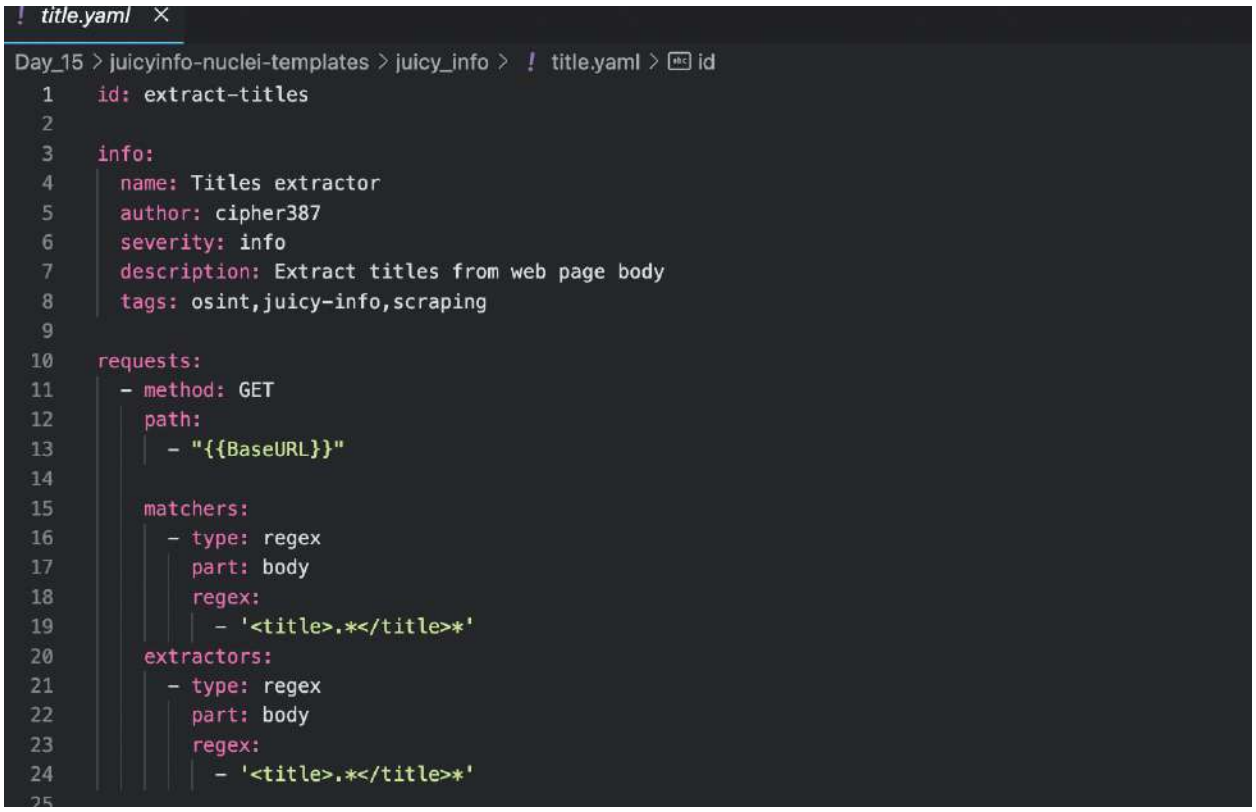
nuclei -t juicyinfo-nuclei-templates/juicy_info/github.yaml -l results.txt >extracted_github_links.txt

Similarly, you can collect links to other social networks, phone numbers, IP addresses, emails, cryptocurrency wallet addresses, etc. (template names are intuitive).
Nuclei also allows you to extract information from html tags. Let's collect the titles (<title> tag) of the same pages:



nuclei -t juicyinfo-nuclei-templates/juicy_info/title.yaml -l results.txt >extracted_titles.txt

What if you want to scrape some other tag instead of the <title> tag?



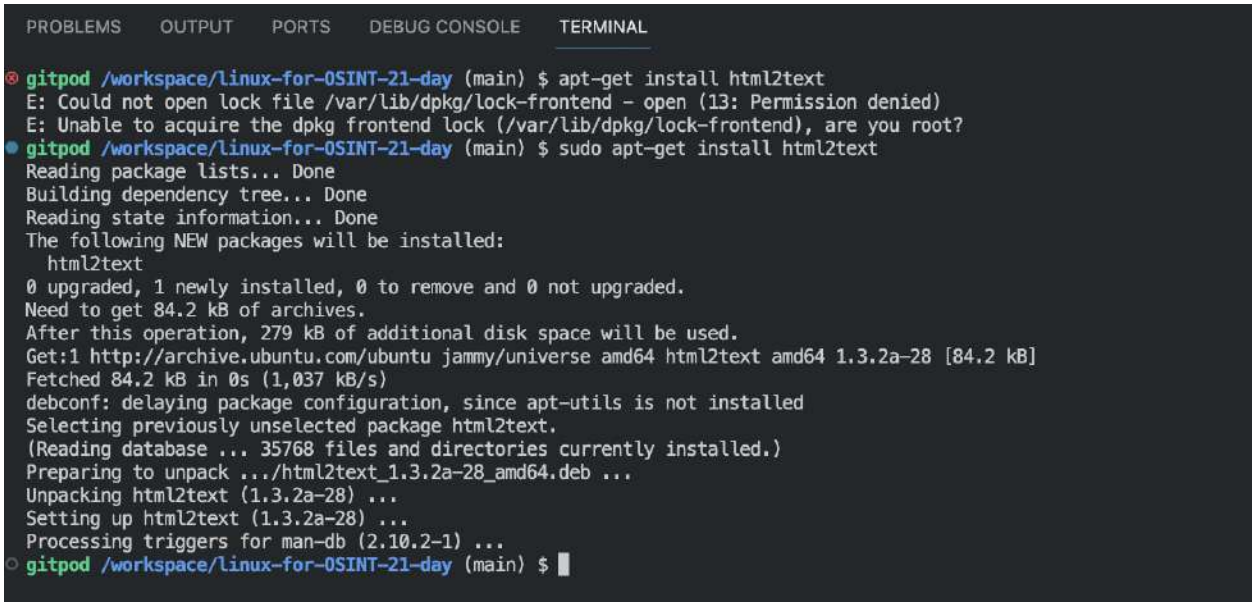
Open the **title.yaml** file and simply replace the title with some other title. If you wish, save the template under a different name and run Nuclei in the above way.

And, of course, you can use whatever regular expressions you can think of for your tasks in the templates.

You could do the same thing with the curl and grep command combinations we talked about on **Days 5** and **6**, but Nuclei is much faster and easier to set up.

HTML2TEXT

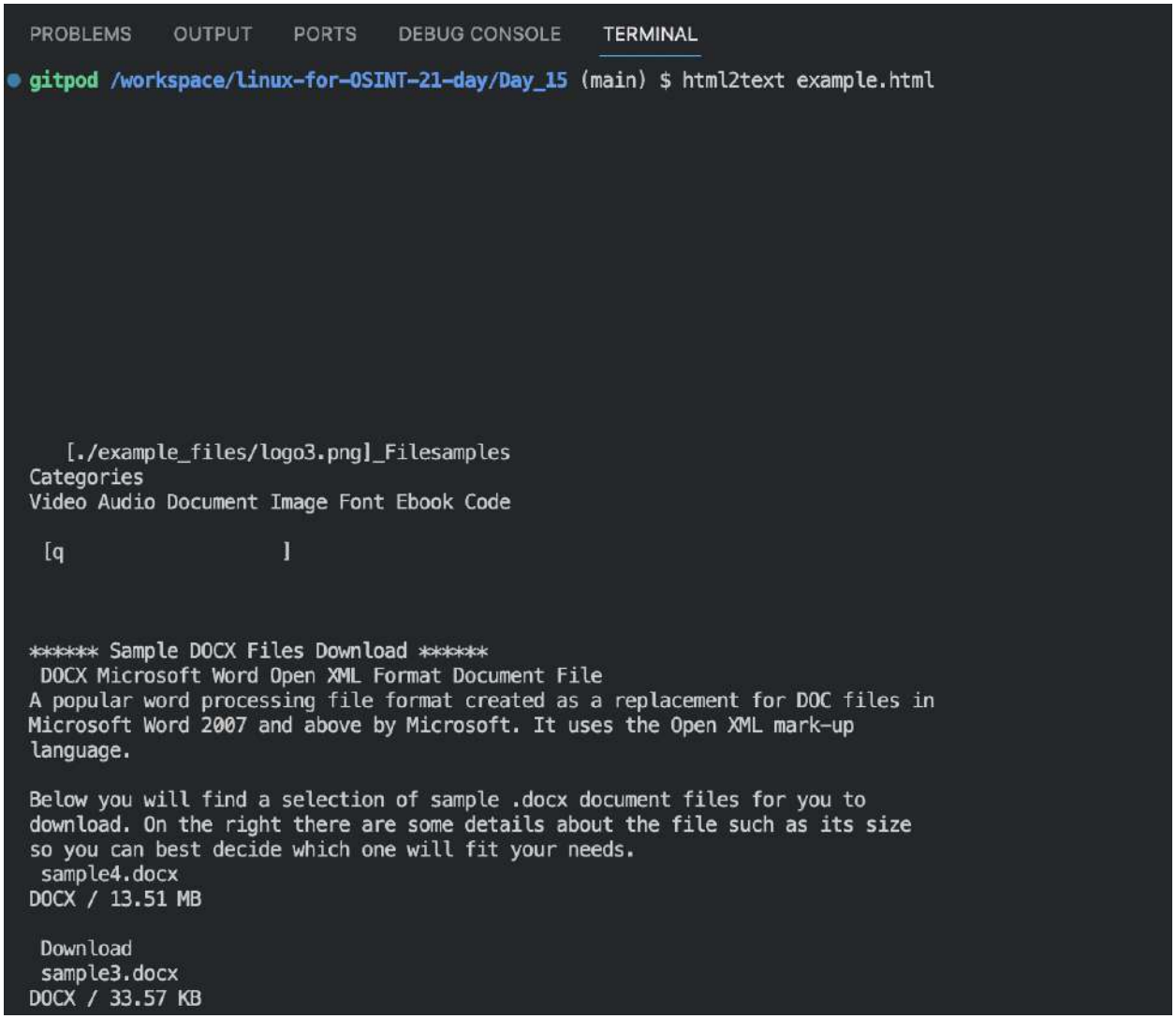
Html2text (https://alir3z4.github.io/html2text/) is a very simple Python script that cleans up the text of web pages from html tags and makes it easy to read and automated information retrieval.



Let’s install:

sudo apt-get install html2text

Let's try to extract the text from the html file contained in the **Day 15** folder:



html2text example.html

Other tools

Scraping is a huge topic with many different nuances that requires a book of its own. Today we just learned how to extract information from html code.

But sometimes collecting data from websites requires automating the opening of the browser and simulating different user actions (because the necessary data is loaded only after executing JavaScript code). In this case, you need to use tools that run browsers (for example, Firefox or Chromium) in the command line.

It should also be said that Linux and Bash are not always the right tools for scraping. For many OSINT-related tasks, any of the many scraping browser plugins that support exporting data in CSV format (pay special attention to AI scraping), which, as you know, is very convenient to work with in Linux, may be sufficient. Few examples:

Webscraper.io (https://webscraper.io)
DataMiner (https://dataminer.io)
Instant Data Scraper (https://chromewebstore.google.com/detail/instant-data-scraper/ofaokhiedipichpaobibbnahnkdoiiah)

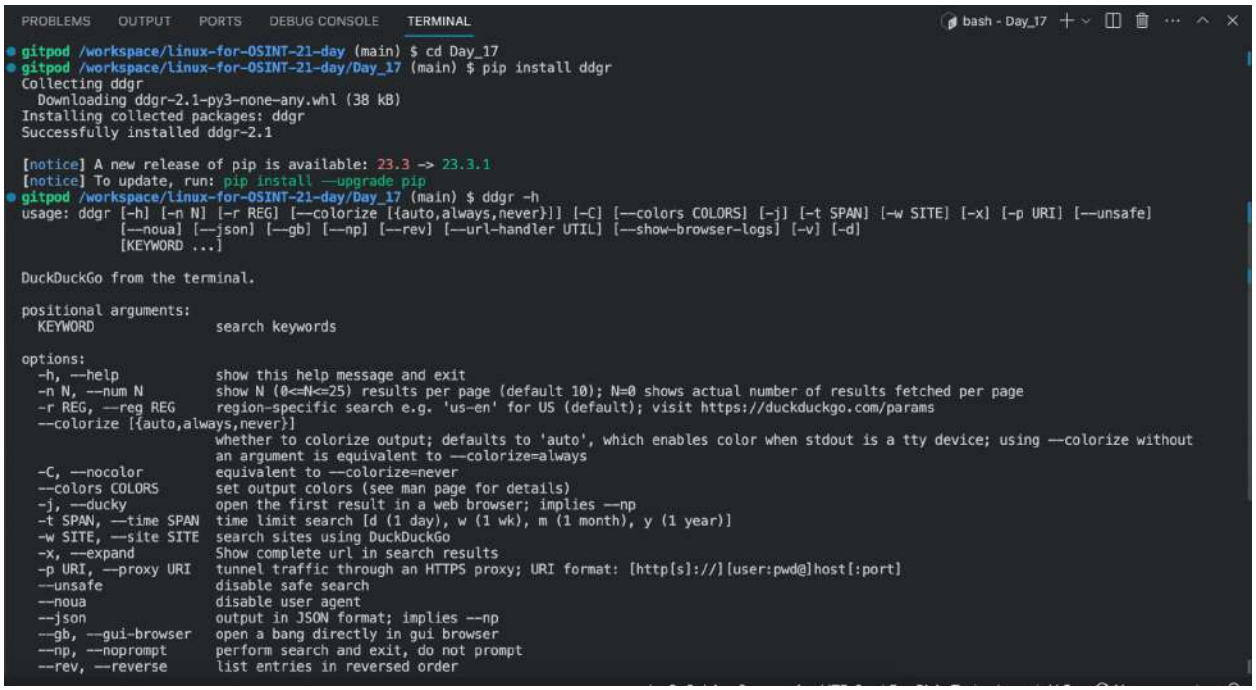
and many many others.

Day 16. Web search automation tools

Searching for information on Google or another search engine can take up most of an OSINT professional's time. Therefore, even if you forget about all the other chapters in this book and automate only this process, this would already be enough to reduce lost work time through automation.

There are many tools for automating your work with different search engines, but they are all pretty similar to each other. Today, I will show you examples of two types of tools - one that automates a traditional search engine (searches web pages) and one that automates an IP search engine (searches through all the devices connected to the internet).

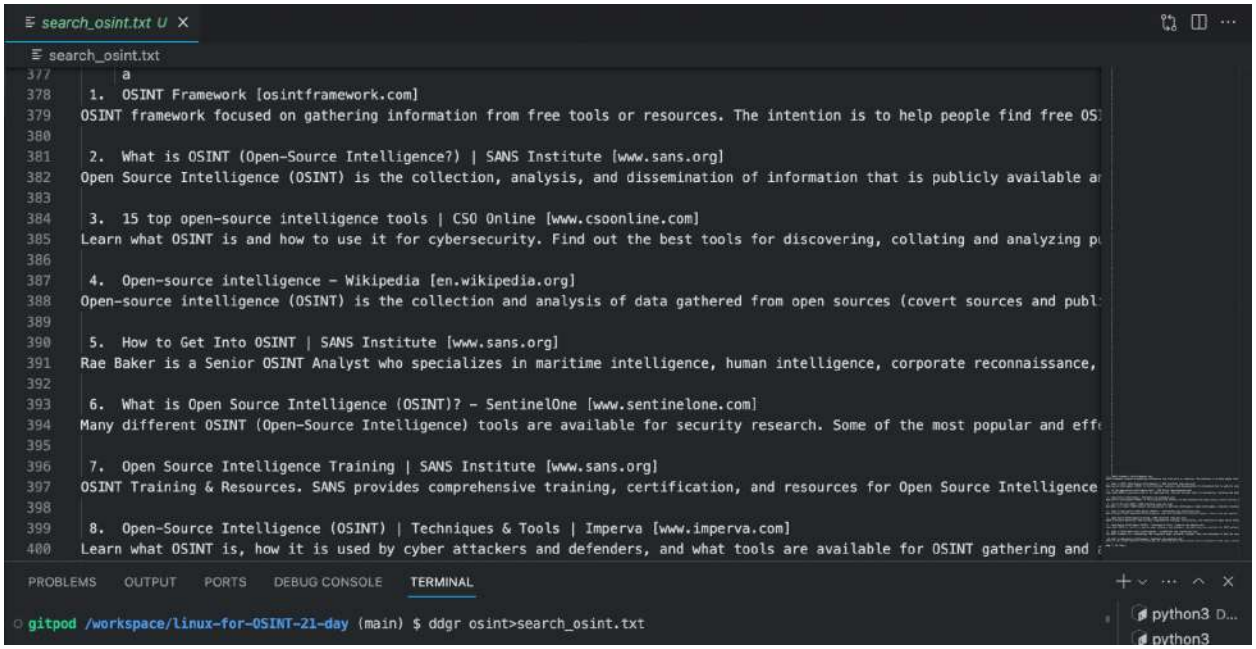
DuckDuckGo search automation



DuckDuckGo search (https://github.com/jarun/ddgr) is a command-line utility written in Python that allows you to automate DuckDuckGo searches.

Lets' install from PyPi package manager:

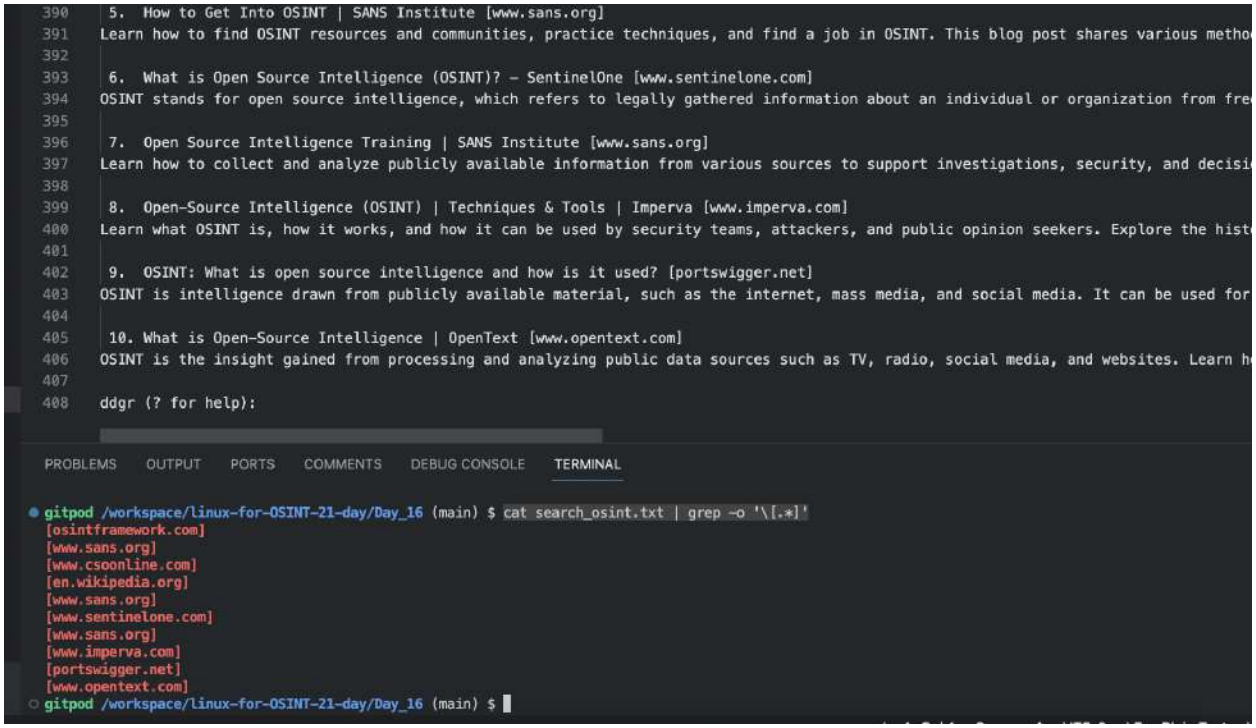
```
pip install ddgr
```



Now let's try searching for the word "osint" in DuckDuckGo and write the results to a text file:

```
cd Day_16
ddgr osint>search_osint.txt
```

If you want to extract only references from the results file (for example, for automatic scraping), you can use a combination of **Cat** and **Grep** commands:



cat search_osint.txt | grep -o '\[.*]'

Yes, is not very good that this tool does not collect full URLs but only site names. But the reader will be motivated to try other tools for collecting search results, which I'll talk about below.

Also a good option is to export the JSON results immediately by running the **Ddgr** command with the **--json** flag.

Remember that you can use not only words and phrases in your queries, but also advanced search operators. For example:

filetype:xlsx
site:facebook.com
intitle:osint
inurl:osint

If you need to gather search results for a larger list of queries, use proxy servers to avoid having your IP address blocked with **--proxy** flag. These can be found in numerous lists of free proxy servers. For example, https://github.com/TheSpeedX/PROXY-List.

To collect the maximum amount of data, I recommend that you collect results not only from DuckDuckGo, but also from other search engines using the tools like:

PyDork (https://github.com/blacknon/pydork) - search in Google, Bing, DuckDuckGo, **Baidu** and Yahoo Japan.
Fast Google Dorks Scan (https://github.com/IvanGlinkin/Fast-Google-Dorks-Scan)
0xDork (https://github.com/rly0nheart/oxdork) - search in Google.

and many others.

Tools for automating the collection of search results for different search engines are plentiful. You can find a list of them in the Dorks collection list repository (https://github.com/cipher387/Dorks-collections-list).

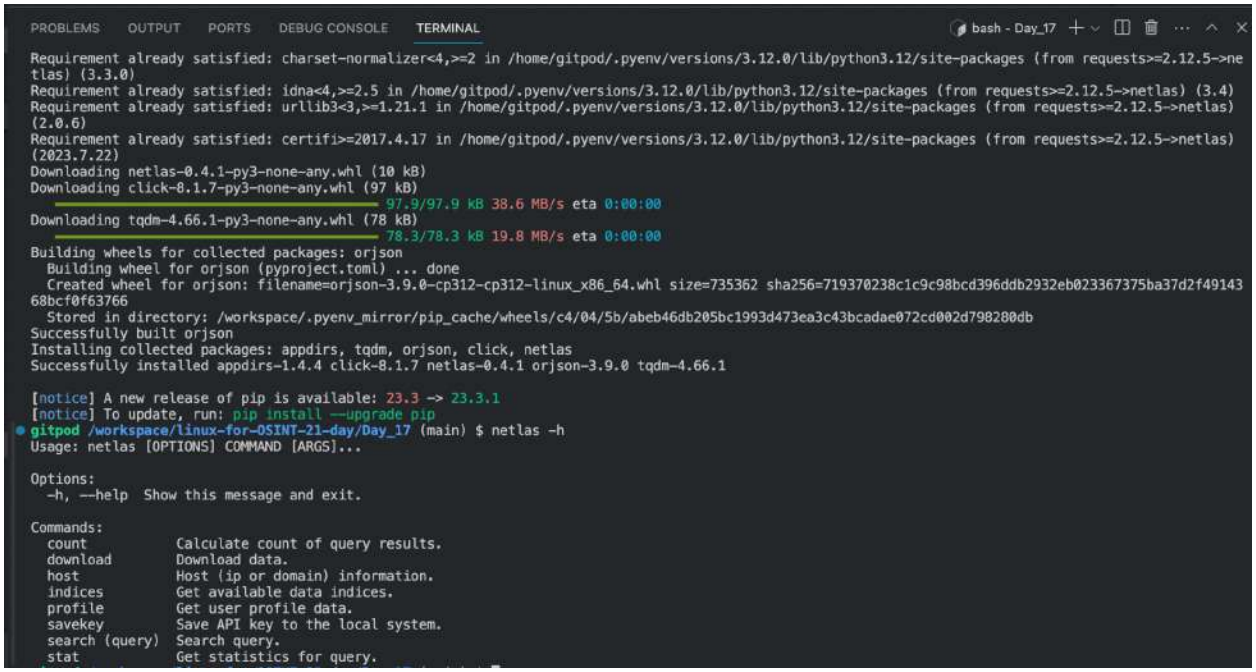
In addition, these tools are quite simply and you can easily make your own using APIs, a list of which can be found at “Search engines API” section of API for OSINT repository (https://github.com/cipher387/API-s-for-OSINT).

IP search engines

It is worth talking separately about automating the collection of IP search engines (Shodan, Censys, Fofa, Netlas) search results. This is a bit more complex process, as search results contain much more information than Google or DuckDuckGo search results.

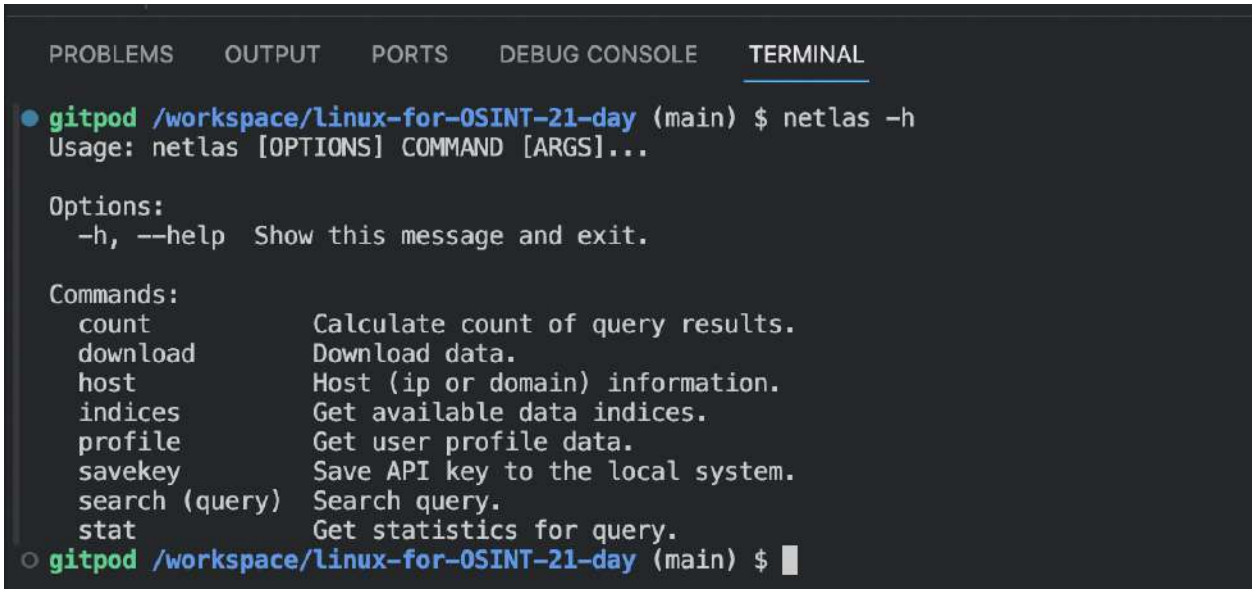
IP search engines differ from “normal” search engines in that they do not index web pages to which they can find links, but all IP addresses connected to the Internet. That is, not only websites, but all other servers and devices.

As usual, I'll use Netlas for an example.



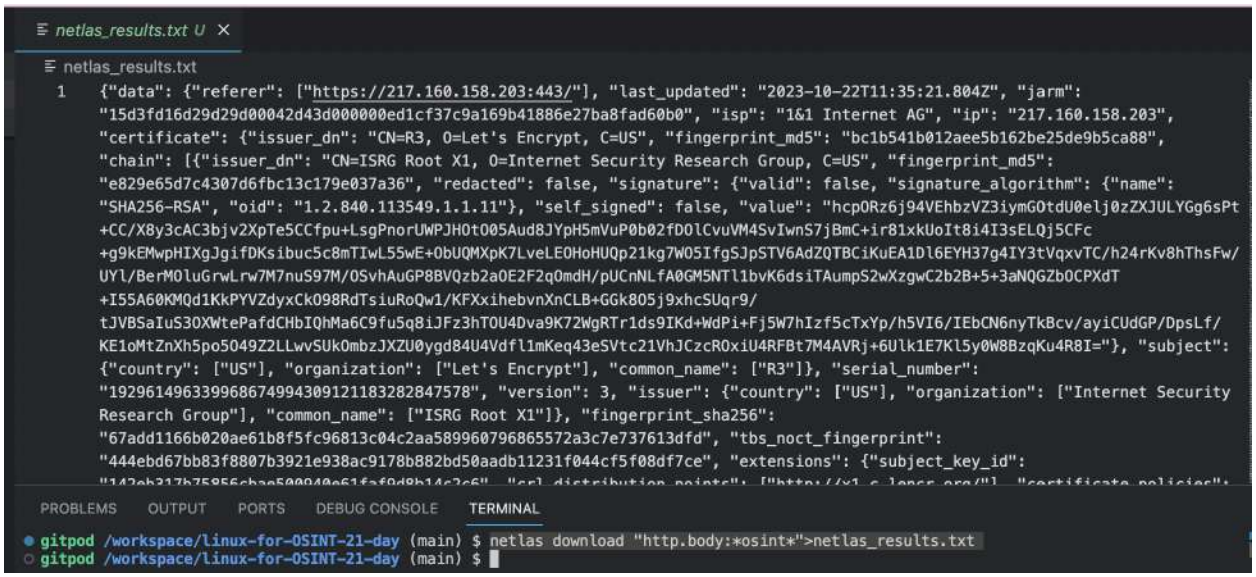
First, let's install the Netlas Python package (https://readthedocs.org/projects/netlas-python/):

pip install netlas
netlas -h



Before sending first request, register on Netlas and copy the API key from your personal account (50 request/day FREE):

```
netlas savekey YOUR_API_KEY
```



Let's try to find servers that have the word "osint" in the body tag of their http response html code:

```
netlas download "http.body:*osint*">netlas_results.txt
```

Note that in the query, we use the filter for advanced search "http.body" to search only the text of web pages. But Netlas has dozens of other advanced search operators. For example:

http.title:
http.meta:
uri:
domain:
port:
geo.city:
ip:
geo.country:

You can extract the any data from the results file using the JQ utility, which is described in detail in **Day 14**:

```
jq .data.http.title netlas_results.txt >netlas_results_extracted.txt
```

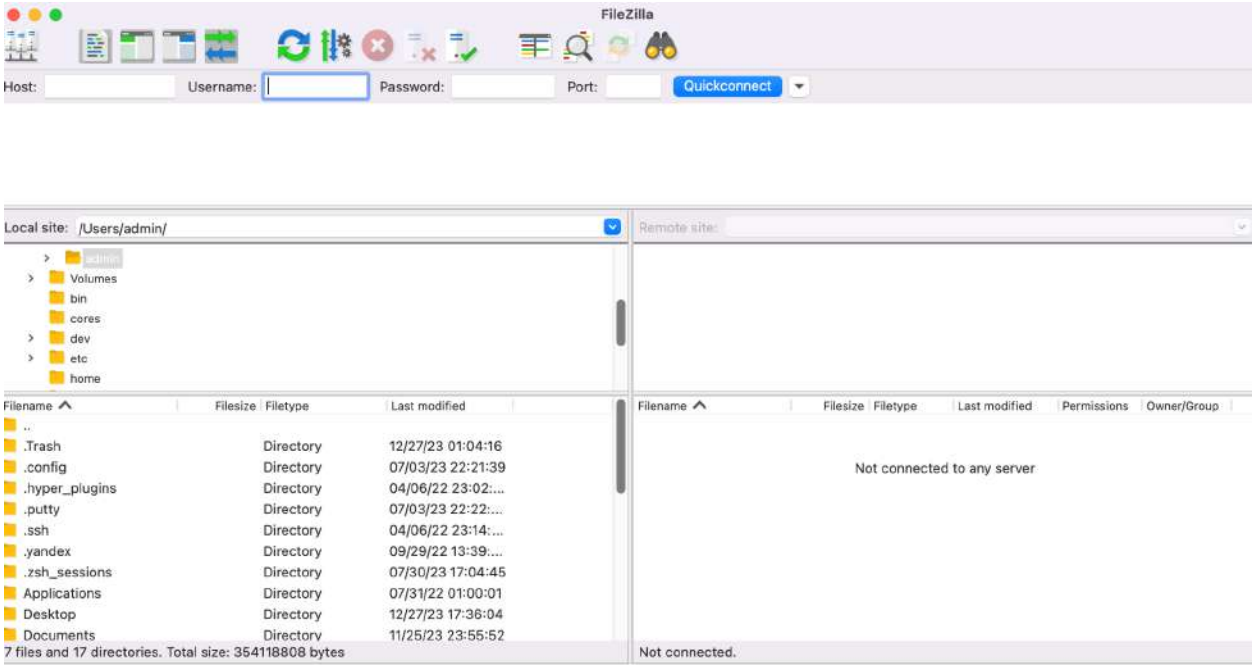
To learn more about how to automate dozens of different OSINT and pentest tasks with Netlas Python library, check out Netlas CookBook (https://github.com/netlas-io/netlas-cookbook).

Day 17. File sharing sites, torrents, FTP

Today we'll return to the topic we started talking about in **Day 5**. It's downloading content (for later automatic analysis). This time we will talk about slightly more complex things, but in a more superficial way: FTP servers, torrents, file sharing and social media.

Today we're not going to run examples of commands, we're just going to talk about different tools that are worth knowing about.

FTP



On the fifth day, we already talked about the **curl** utility, which supports uploading files through different protocols, including the FTP protocol.

But still, the FTP utility (https://www.gnu.org/software/inetutils/manual/html_node/ftp-invocation.html) is a bit more convenient for this task.

If you just need to transfer files, you will probably be more comfortable using a GUI FTP client such as FileZilla (https://wiki.filezilla-project.org/Main_Page) rather than the command line.

But FileZilla (like many other FTP clients) allows you to automate various actions using the command line (https://wiki.filezilla-project.org/Command-line_arguments_(Client)).

Torrents

It is illegal to use leaked data files, but many OSINT professionals use them anyway to gather more information. Often the leaked data is stored as huge files on torrents.

For example, I had to deal with one major social network leak that weighed hundreds of gigabytes. Unpacking the archives and extracting the necessary data from it using the JQ utility took more than two days! But thanks to the screen (**Day 9**) I only had to spend about an hour writing commands and check screen sessions a few times to see if everything's going well.

Seedbox is a high-bandwidth remote server used to upload and download digital files from a peer-to-peer (P2P) network (tech slang).

Here are some examples of popular services for monthly seedbox rentals:

RapidSeedbox (https://www.rapidseedbox.com)
CloudBoxes (https://cloudboxes.io)
DediSeedBox (https://dediseedbox.com/)

Seedboxes is a very handy thing, but you can run command line torrent clients on your VPS as well if you want:

Aria (https://aria2.github.io)
Transmission-cli (https://cli-ck.io/transmission-cli-user-guide/)
WebTorrent-cli (https://github.com/webtorrent/webtorrent-cli)
Stig (https://github.com/rndusr/stig)

File sharing services

When using files that are uploaded to various file sharing sites in investigations, you should also keep in mind that their downloading can be automated. Here are some examples of command line utilities for different services that will help you do this:

Google Drive (https://github.com/glotlabs/gdrive)

DropBox (https://github.com/dropbox/dbxcli)

Mega.nz (https://github.com/meganz/MEGAcmd)

OneDrive (https://github.com/skilion/onedrive)

But be very careful when using them and never use accounts where you have important information stored for authorization. **After all, these are unofficial utilities and accounts can be blocked for using them.**

Social media

When you work with a lot of content that is published on some popular social network, check to see if someone else has made a utility to automate its uploading. For example:

YouTube-dl (https://youtube-dl.org)

SlideShare (https://github.com/mohan3d/slideshare-go)

Spotify (https://github.com/spotDL/spotify-downloader)

You Get (https://github.com/soimort/you-get)

And I'll say it again. We are talking about unofficial clients, the use of which can lead to blocked accounts. Be very careful.

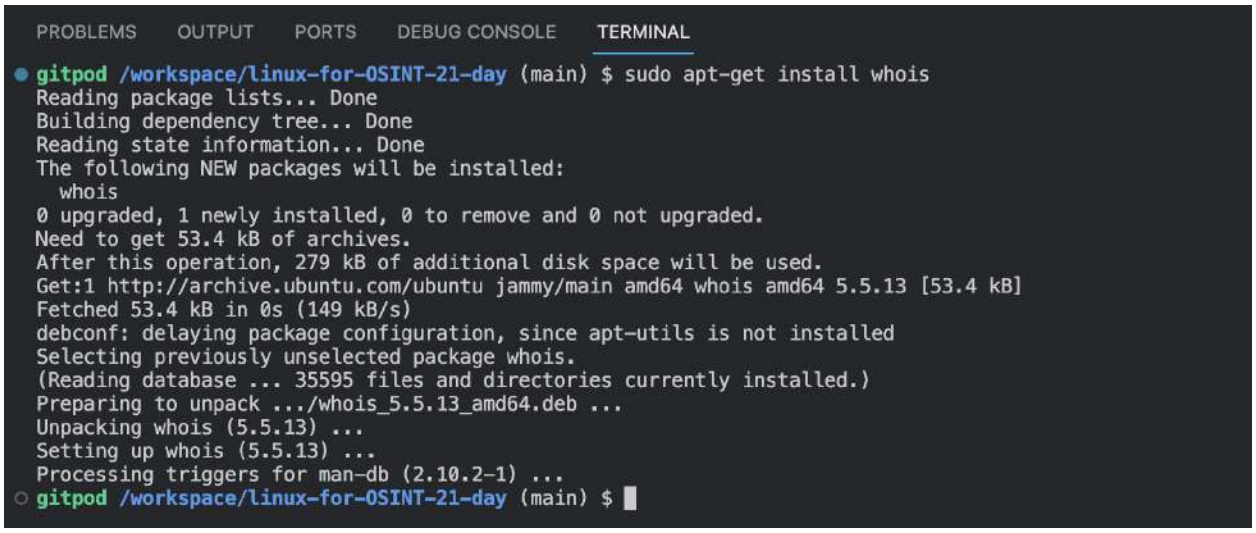
Day 18. Domain investigation

Perhaps when you first saw this book, you thought of it as being mostly about automating the network information gathering. But this book has the goal of teaching OSINT technicians minimal Linux skills.

Therefore, I will cover only the most basic techniques.

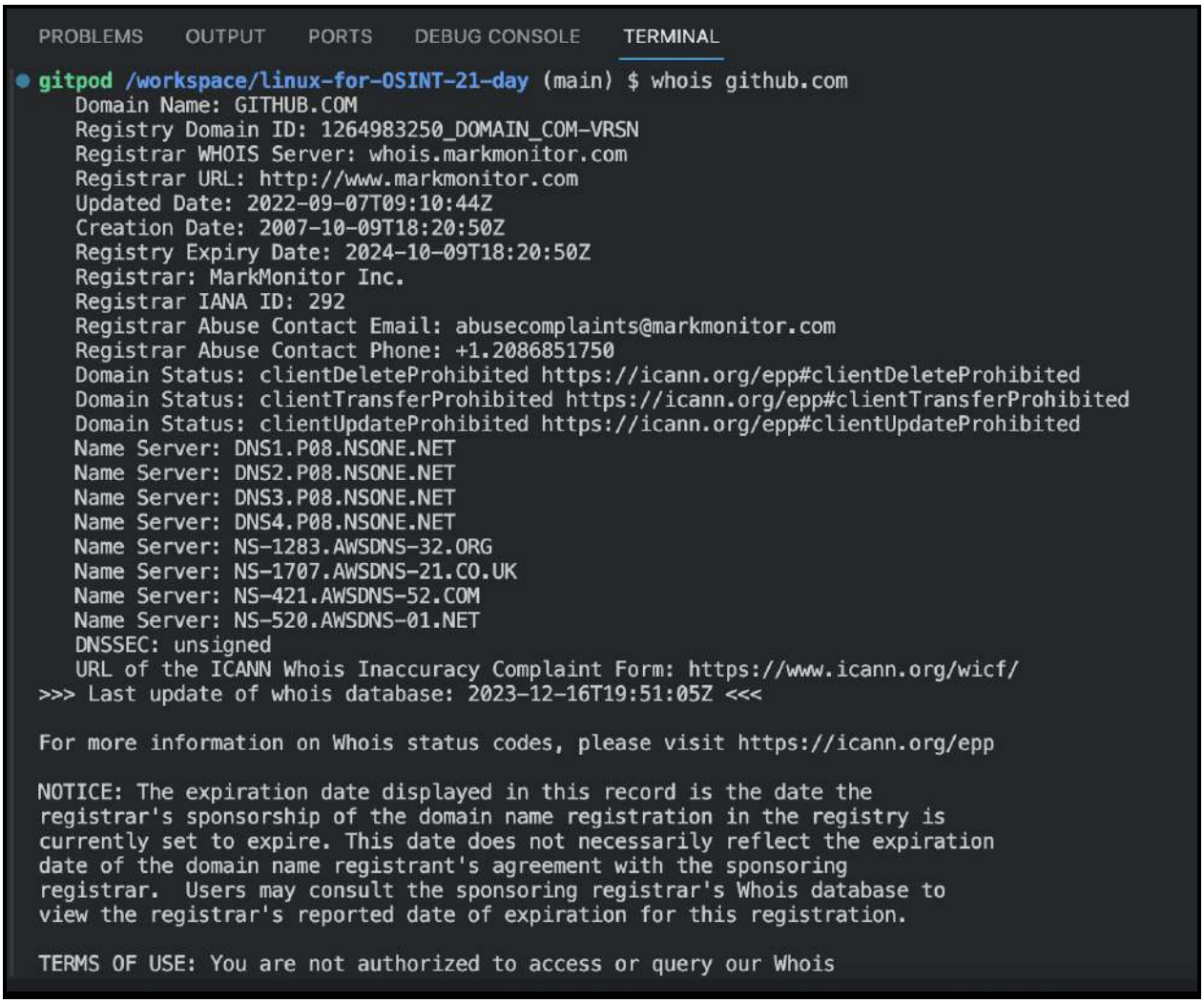
Whois

Whois is a simple utility that displays basic information about a domain: whether it is registered or not, date of registration, end date of the paid holding period, contact details of the owner, etc.



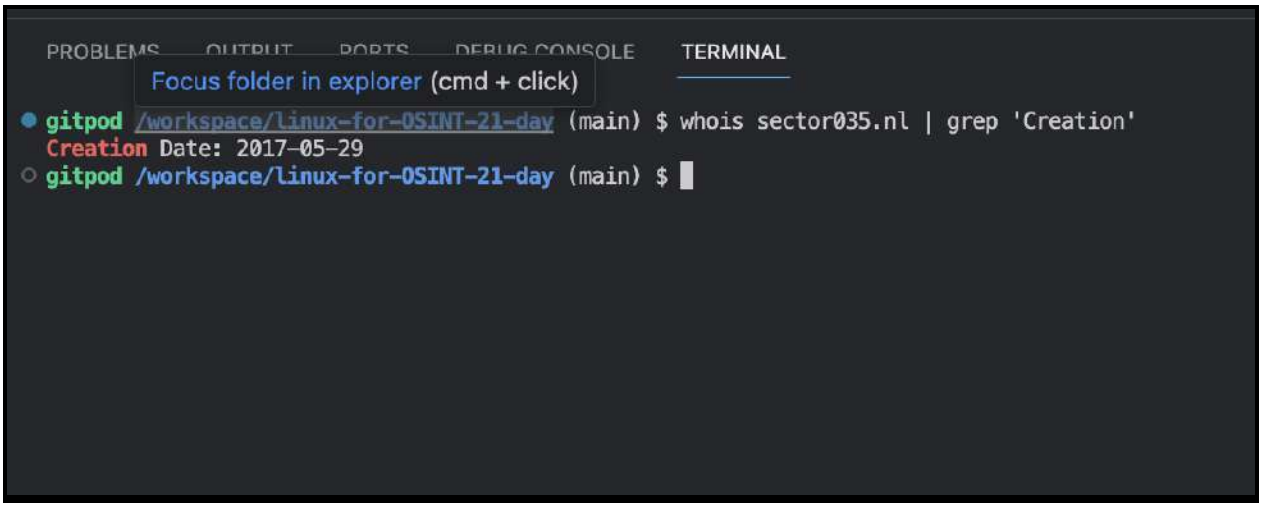
Let’s install:

sudo apt-get install whois



And let's gather information about the GitHub.com domain:

whois github.com



If you only need certain information from whois, you can extract the necessary strings with grep (remember **Day 6**):

whois sector035.nl | grep 'Creation'

DNS

Dnsutils Debian package (https://packages.debian.org/en/buster/dnsutils) includes three commands:

dig - query the DNS (Domain Name System) info;
nslookup - the older way to do it;
nsupdate - perform dynamic updates (See RFC2136).

For the purposes of this course, we will consider only the first.

```
PROBLEMS   OUTPUT   PORTS   DEBUG CONSOLE   TERMINAL
gitpod /workspace/linux-for-OSINT-21-day (main) $ sudo apt-get install dnsutils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bind9-dnsutils bind9-host bind9-libs liblmdb0 libuv1
The following NEW packages will be installed:
  bind9-dnsutils bind9-host bind9-libs dnsutils liblmdb0 libuv1
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,594 kB of archives.
After this operation, 9,892 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 liblmdb0 amd64 0.9.24-1build2 [47.6 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libuv1 amd64 1.43.0-1 [93.1 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 bind9-libs amd64 1:9.18.12-0ubuntu0.22.04.3 [1,240 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 bind9-host amd64 1:9.18.12-0ubuntu0.22.04.3 [52.3 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 bind9-dnsutils amd64 1:9.18.12-0ubuntu0.22.04.3 [157 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 dnsutils all 1:9.18.12-0ubuntu0.22.04.3 [3,924 B]
Fetched 1,594 kB in 0s (13.2 MB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package liblmdb0:amd64.
(Reading database ... 35604 files and directories currently installed.)
Preparing to unpack .../0-liblmdb0_0.9.24-1build2_amd64.deb ...
Unpacking liblmdb0:amd64 (0.9.24-1build2) ...
Selecting previously unselected package libuv1:amd64.
Preparing to unpack .../1-libuv1_1.43.0-1_amd64.deb ...
Unpacking libuv1:amd64 (1.43.0-1) ...
Selecting previously unselected package bind9-libs:amd64.
Preparing to unpack .../2-bind9-libs_1%3a9.18.12-0ubuntu0.22.04.3_amd64.deb ...
Unpacking bind9-libs:amd64 (1:9.18.12-0ubuntu0.22.04.3) ...
Selecting previously unselected package bind9-host.
Preparing to unpack .../3-bind9-host_1%3a9.18.12-0ubuntu0.22.04.3_amd64.deb ...
Unpacking bind9-host (1:9.18.12-0ubuntu0.22.04.3) ...
Selecting previously unselected package bind9-dnsutils.
```

Let's install dnsutils:

sudo apt-get install dnsutils

```
PROBLEMS   OUTPUT   PORTS   DEBUG CONSOLE   TERMINAL
gitpod /workspace/linux-for-OSINT-21-day (main) $ dig google.com

; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39629
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             122     IN      A       142.250.201.174

;; Query time: 3 msec
;; SERVER: 1.1.1.1#53(1.1.1.1) (UDP)
;; WHEN: Sat Dec 16 19:55:35 UTC 2023
;; MSG SIZE  rcvd: 55

gitpod /workspace/linux-for-OSINT-21-day (main) $ 4
```

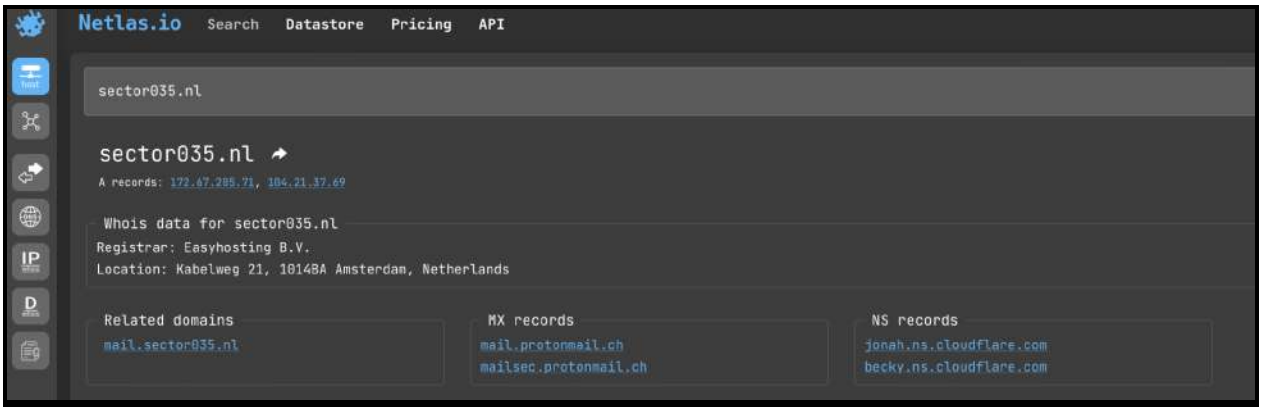
And get information from Domain Name System about google.com:

```
dig google.com
```

Netlas

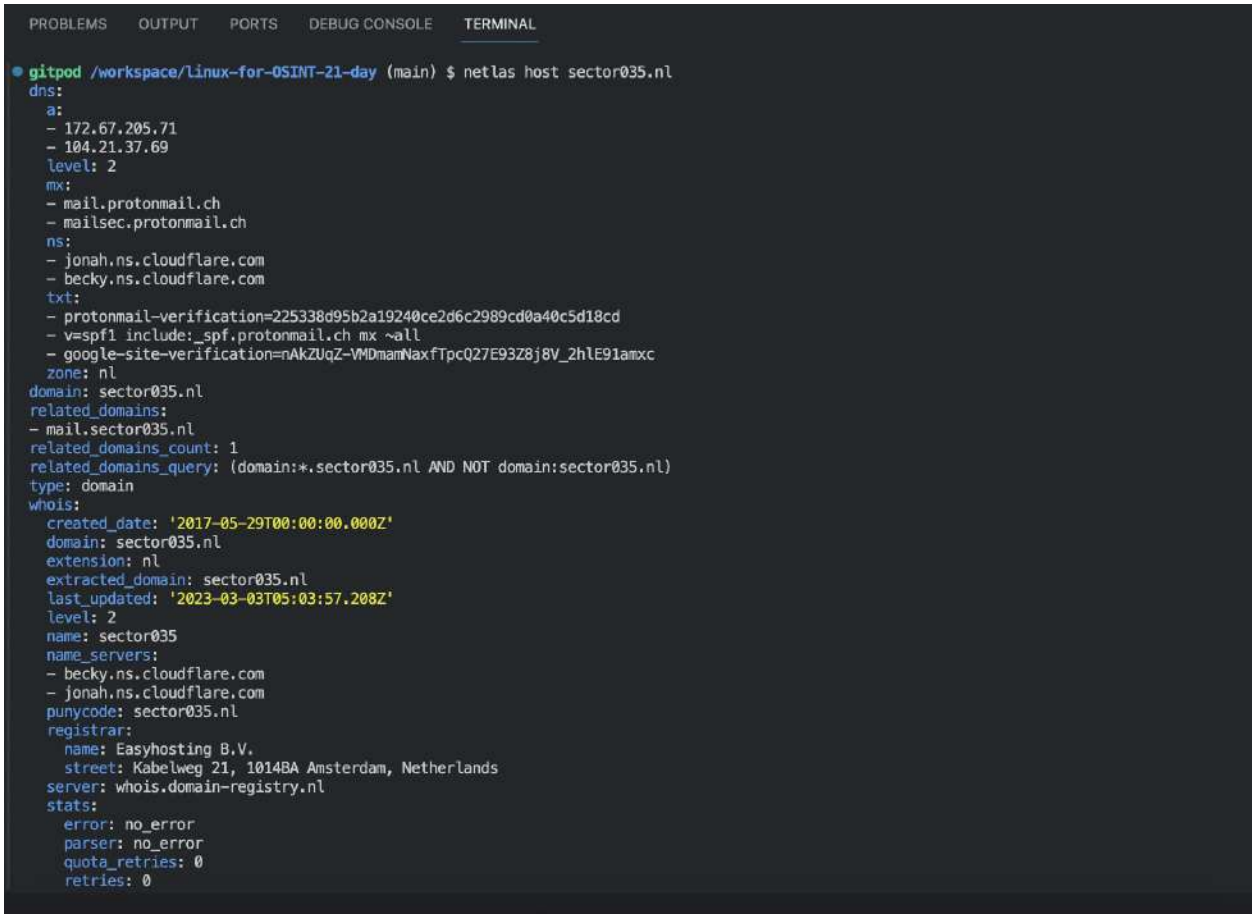
As you already know from past lessons, Netlas is a search engine that indexes not only websites, but all IP addresses connected to the Internet. And there is a lot of different information about each address or domain stored there, which can be retrieved with the **host** command.

Let's try to do it:



If you haven't already done so on **Day 16**, install the Netlas package from PyPi:

```
pip install netlas
```



And run:

netlas host sector035.nl

As you can see, it displays both whois information, dns information, addresses of various servers that are associated with the domain that can be used for further investigation and many other info.

As I warned in the preface to this book, it is not written for pentesters and bug hunters at all, but for OSINT specialists. If you plan to do a job that requires a more in-depth knowledge of computer networks, you should explore many other sources of information.

A good place to start is with the manual for Nmap (https://nmap.org), one of the most famous utility for network exploration and security auditing.

All the commands described above can be combined with Xargs (remember **Day 4**) and used to work with domain/IP address lists.

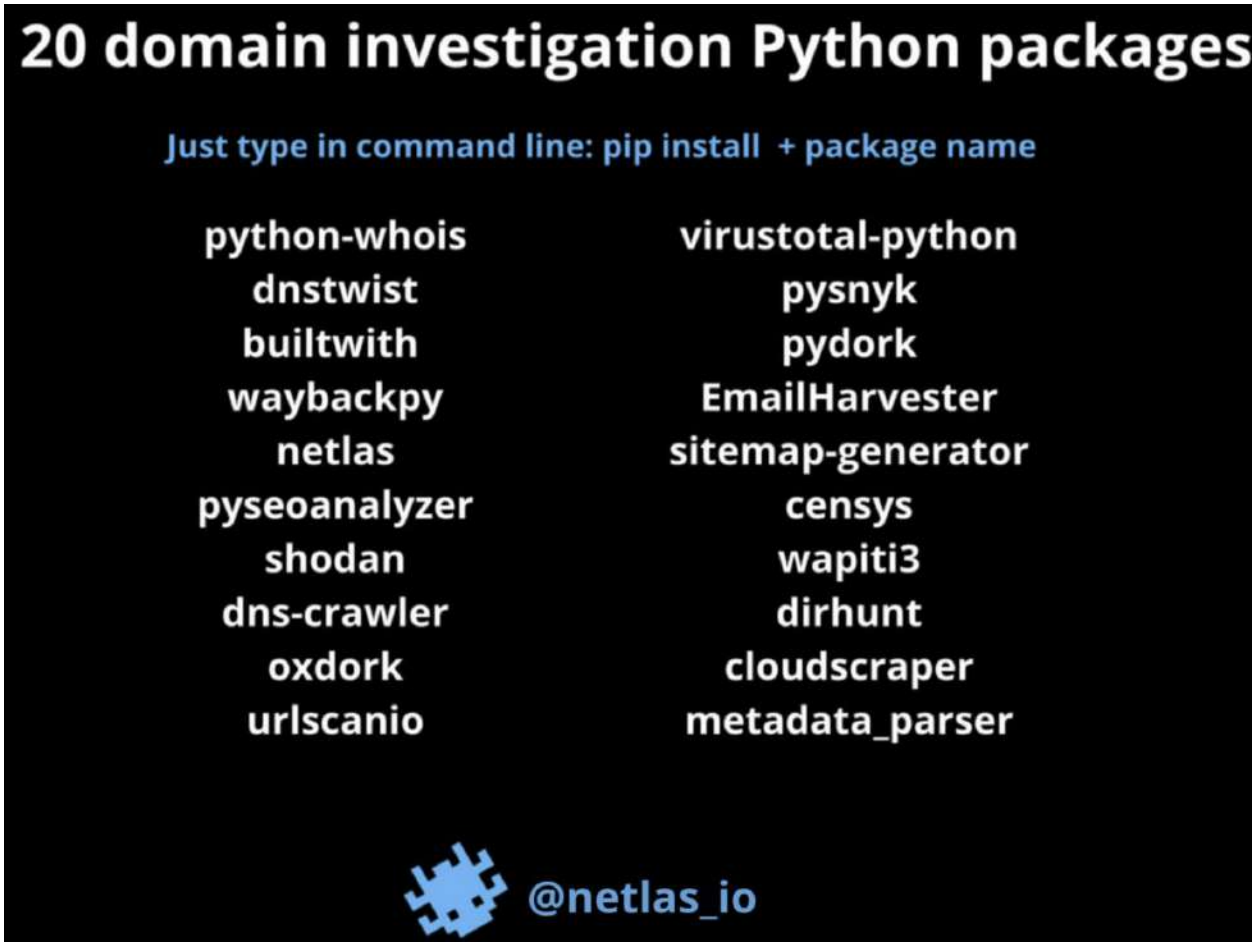


Image from Twitter.

In the picture is a small list with Python packages to gather information about the websites. To install each of them, type **pip install** (package name) at the command line, and then use the help command to understand what a particular package is for.

This is the largest homework in this book and you can spend several weeks completing it. We will try some more commands tomorrow, and next lessons will be without command examples, just reading and thinking.

Day 19. Git and Github

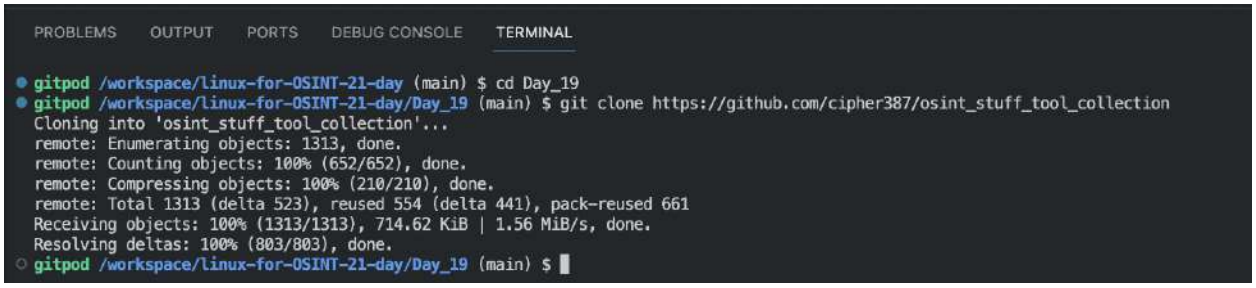
Git (https://git-scm.com) is a software version control system created by Linus Torvalds. According to StackOverflow's 2022 survey, more than 93% of developers use it as their primary version control system (https://survey.stackoverflow.co/2022/#section-version-control-version-control-systems).

The situation with Git and Linux is that exceptionally rare case where one person managed to invent two such game-changing software in one lifetime (Linux in 1991, Git in 2005).

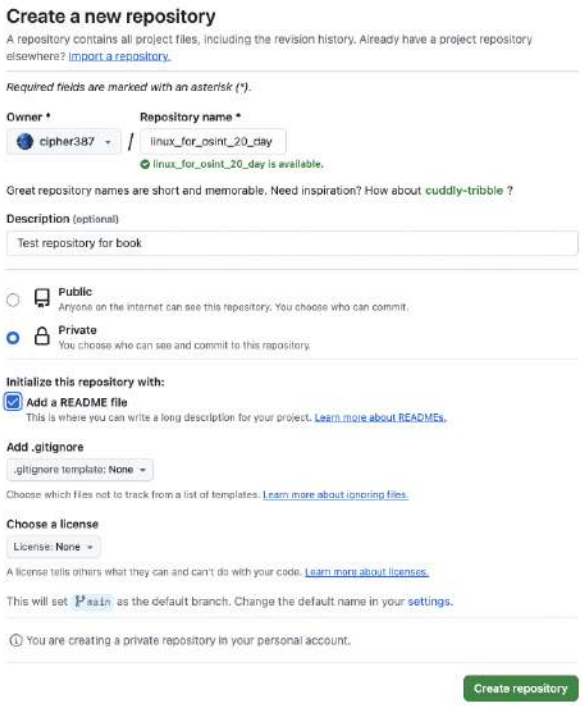
Even if you don't intend to do any development work, you can still benefit from basic Git skills for OSINT. This comes in handy for running different tools, as well as for analyzing Github repositories that store large amounts of text data (e.g., phishing domain lists or smart contract files).

Gitpod already has Git installed, but in other cases you may need to install it:

```
sudo apt-get update
sudo apt-get install git
```

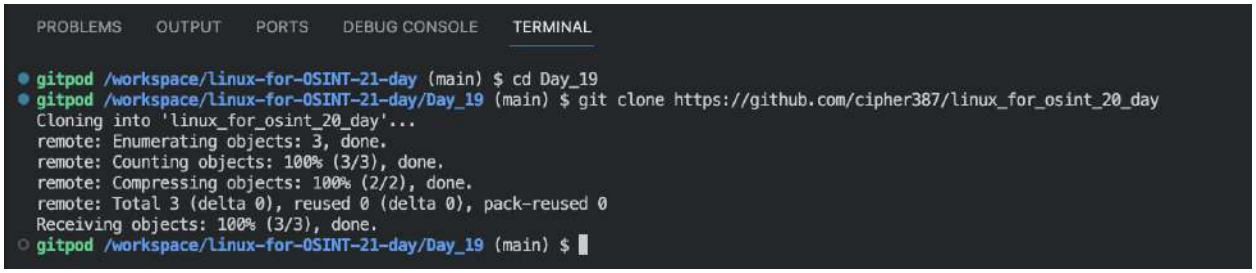


Let’s go to https://github.com/new and create new empty Github repository.



Look at the image above, fill in the repository name and description fields, make the repository private, create a README.MD file and click on **Create repository**.

Now go back to Gitpod and type in the command line:



git clone https://github.com/cipher387/linux_for_osint_20_day

Replace cipher387 with your Github login and if your repository name is different, replace that too.

You can only copy private repositories from your own account, or from the accounts of people who have given you access to one or more private repositories.

Public repositories can be cloned from any account.

Now let's make sure that the repository has been cloned correctly:

```
PROBLEMS  OUTPUT  PORTS  DEBUG CONSOLE  TERMINAL

gitpod /workspace/linux-for-OSINT-21-day (main) $ cd Day_19
gitpod /workspace/linux-for-OSINT-21-day/Day_19 (main) $ git clone https://github.com/cipher387/linux_for_osint_20_day
Cloning into 'linux_for_osint_20_day'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
gitpod /workspace/linux-for-OSINT-21-day/Day_19 (main) $ cd linux_for_osint_20_day
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $ ls
README.md
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $ cat README.md
# linux_for_osint_20_day
Test repository for book
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $
```

cd linux_for_osint_20_day
ls
cat README.md

Let's create a new file in the repository folder:

ls >test.txt
cat test.txt

```
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $ ls >test.txt
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $ cat ls
cat: ls: No such file or directory
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $ cat test.txt
README.md
test.txt
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $ git add test.txt
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $ git commit
[main 5fac56b] test commit
 1 file changed, 2 insertions(+)
 create mode 100644 test.txt
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 299 bytes | 299.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/cipher387/linux_for_osint_20_day
   37d73f3..5fac56b  main -> main
gitpod /workspace/linux-for-OSINT-21-day/Day_19/linux_for_osint_20_day (main) $
```

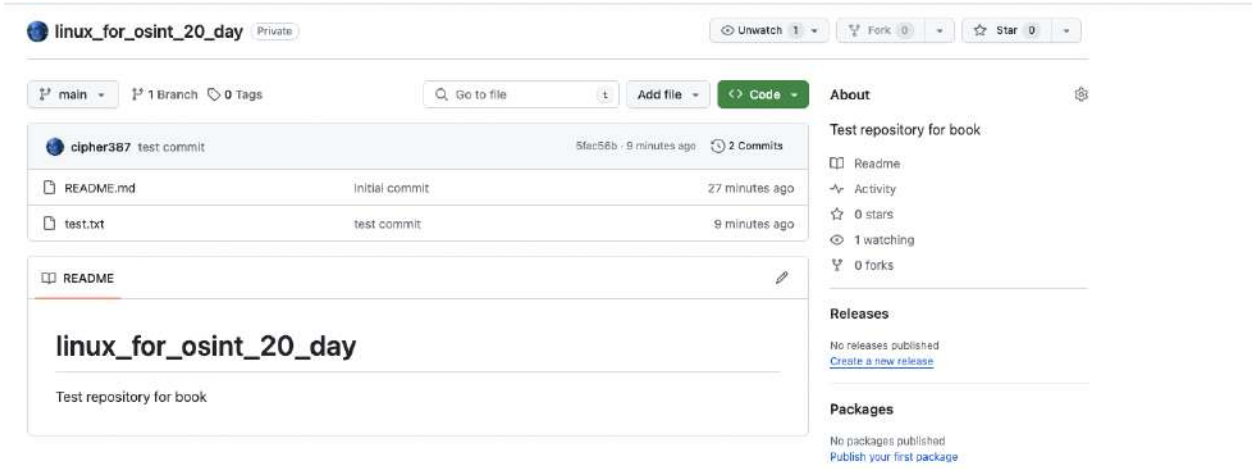
And add it to a new commit:

```
git add test.txt
git commit
```

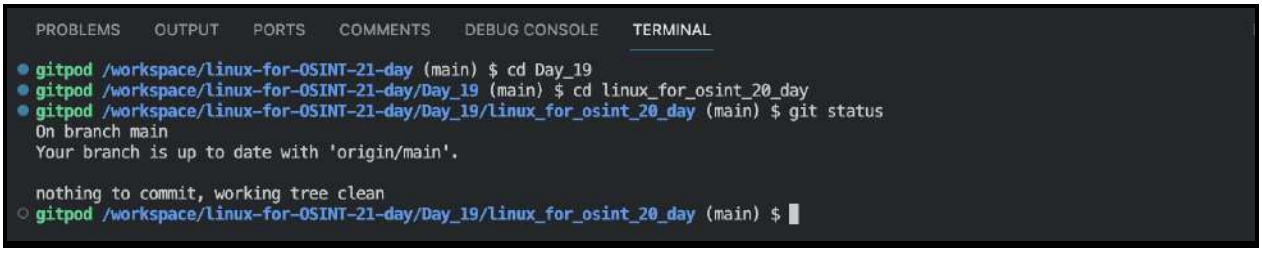
Write some comment on the commit, close the text editor and type.

```
git push
```

Now open Github and verify the changes you have made:

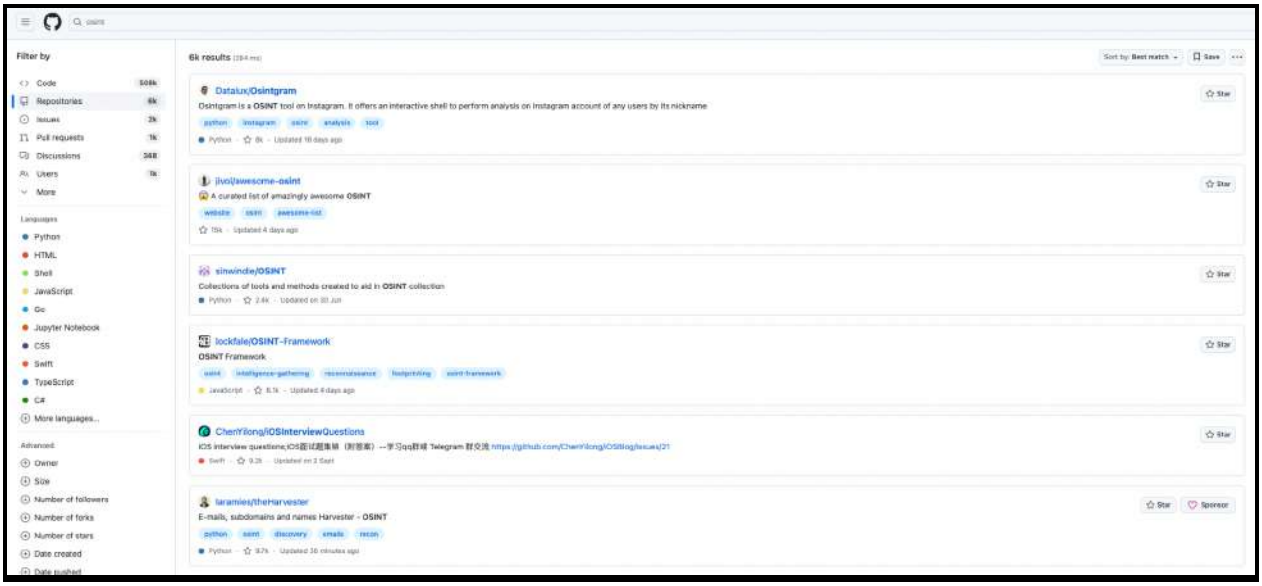


If you are in doubt about whether you have made the latest changes, check the status of the repository with this command:



git status

There are about a dozen different git commands. But the most important thing you should realize from this tutorial is that working with Github repositories using the command line is easy and convenient.



Github is not only a service for organizing collaboration on software development. It is also a huge platform for free distribution of open source software and technical knowledge.

You can also find an active OSINT-enthusiast community there, including thousands of users and repositories.

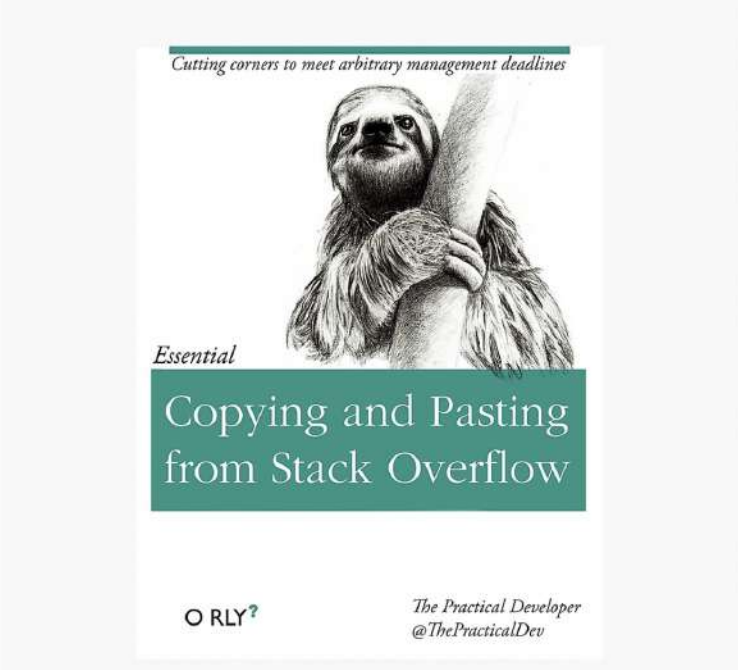
You've probably already created a Github profile to authorize Gitpod. Feel free to use it to interact with the global OSINT community:

- give stars to repositories you like. Pay special attention to good and little-known projects that have very few stars.
- If you encounter any errors while running any tools, feel free to write about them in Issues.
- If you have something useful, even if it's just a list of tools or a cheat sheet with useful commands, feel free to put it in the public repository.
- and, of course, don't forget to subscribe to my profile (https://github.com/cipher387).

Day 20. Tools to make Linux easier to use

In this book, I have already written about tools that can help you write commands in Linux. For example, in the chapters on **Sed** and **Awk**, JSON and XML. Today we will talk about more universal tools.

The most important thing you should know about the Linux command line is that you need to know almost nothing about it to successfully solve your problems using it.



If you have read at least 1-2 of the past lessons (days) and understand where to put files and where to enter commands, you can do a lot with the help of Google and StackOverflow. Really a lot.

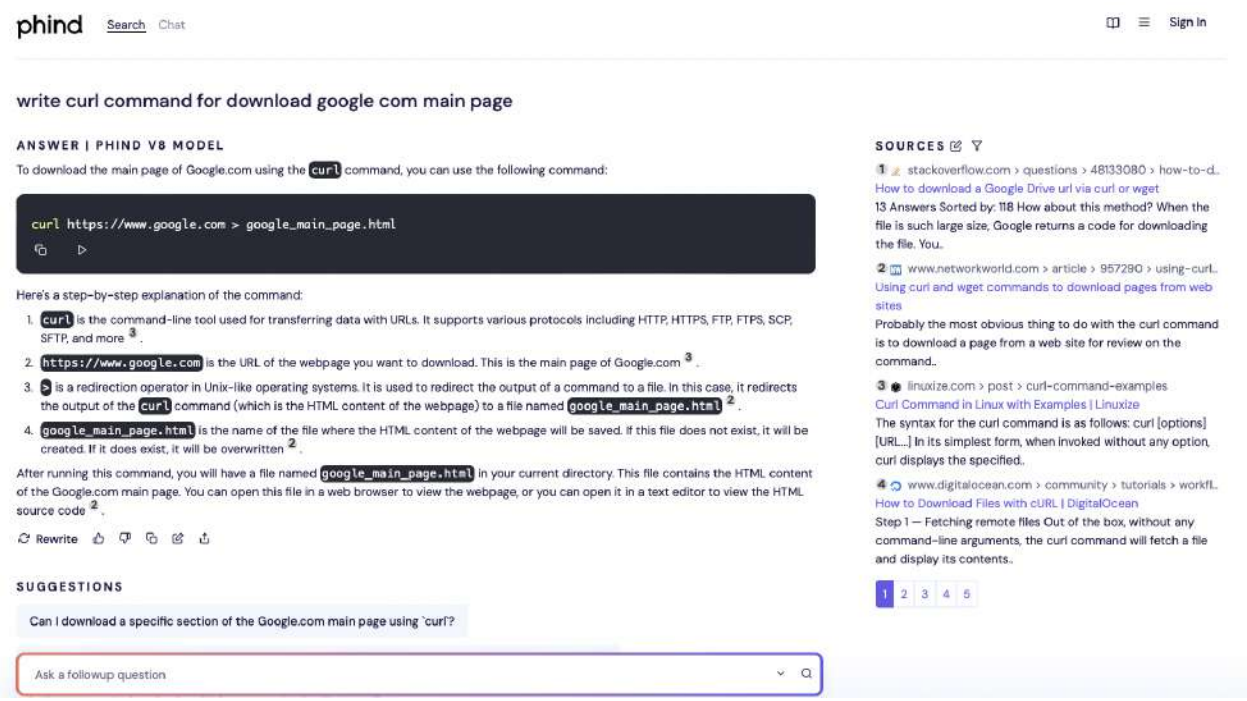
AI tools

Using AI assistants to write code has become very popular in the last year and a half. Many popular chatbots can generate both individual commands and full-fledged bash scripts:

ChatGPT (https://chat.openai.com)
Perplexity.ai (https://www.perplexity.ai)
Google Bard (https://bard.google.com)
You.com (https://you.com/)

They are exceptional at saving time on simple tasks, but when dealing with more complex issues, they can make mistakes and lead down the wrong path. Be careful.

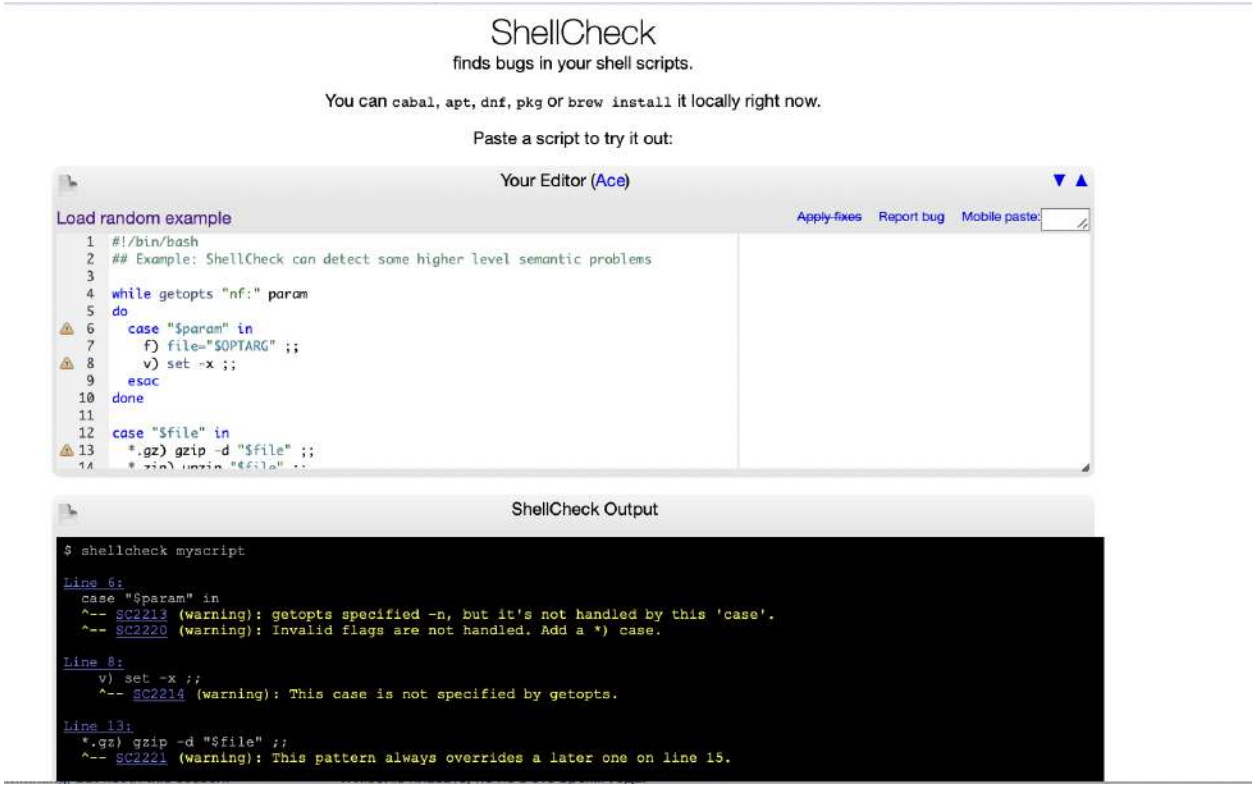
And don't forget that you can not only ask AI assistants to write code or a command from scratch, but explain what some command or script found on Stackoverflow does. You can also use the Explain Shell (https://explainshell.com/) for this purpose.



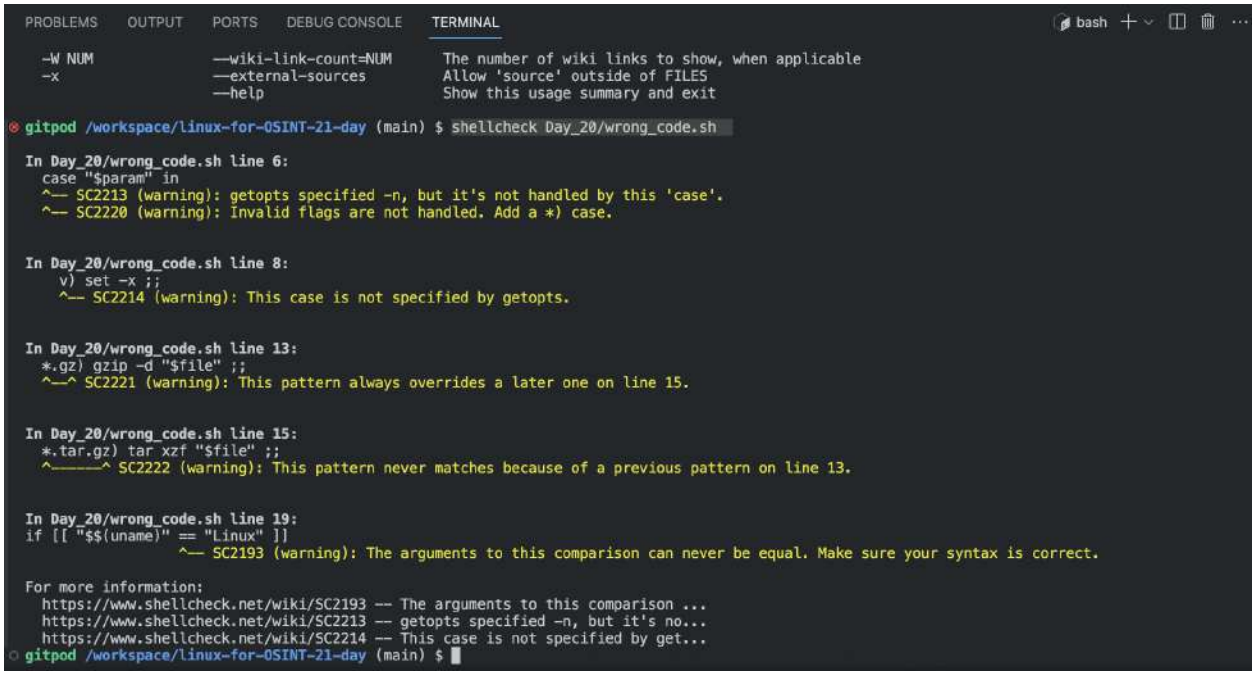
Separately, I would like to pay your attention to the AI assistant Phind (https://www.phind.com/), which not only write commands, but also provides explanations with sources on various reference sites like StackOverflow.

Code checkers

StackOverflow users, AI assistants and everyone who writes code sometimes make mistakes. Special tools will help you to understand where exactly the error is.



ShellCheck (https://www.shellcheck.net/#) - simple online tool that help to find bugs in shell scripts. You can also run it locally using the command line.



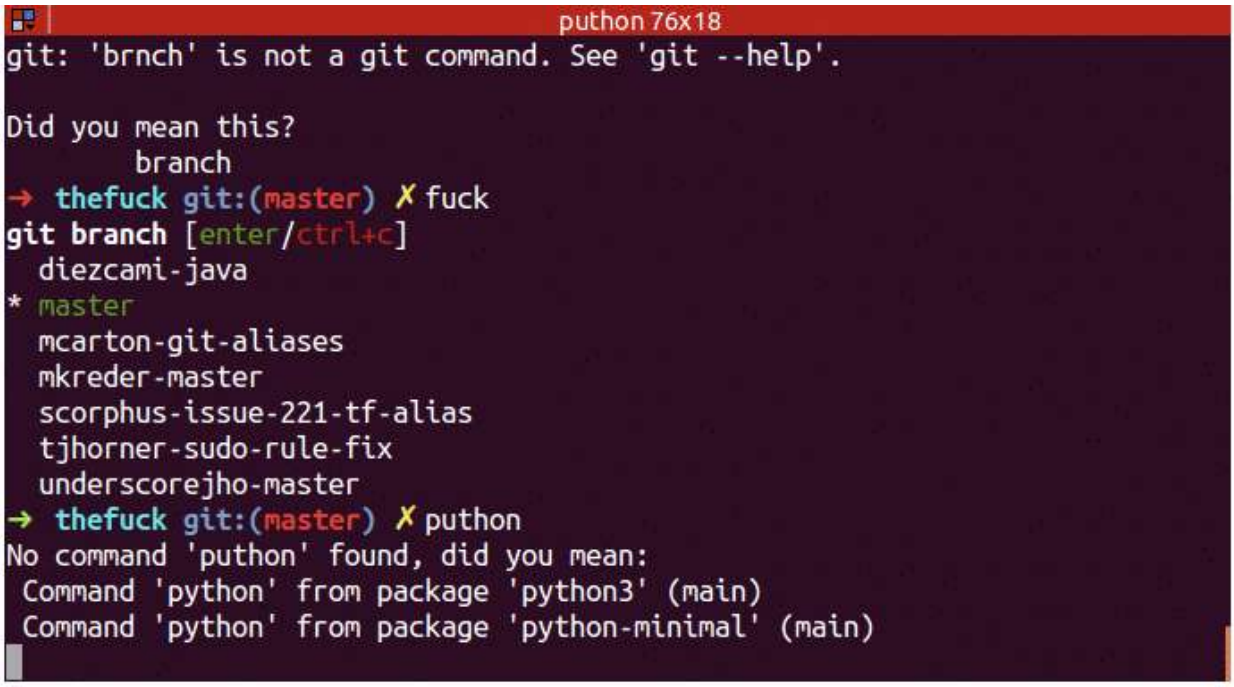
Install:

sudo apt-get install shellcheck

And run the test:

shellcheck Day_20/wrong_code.sh

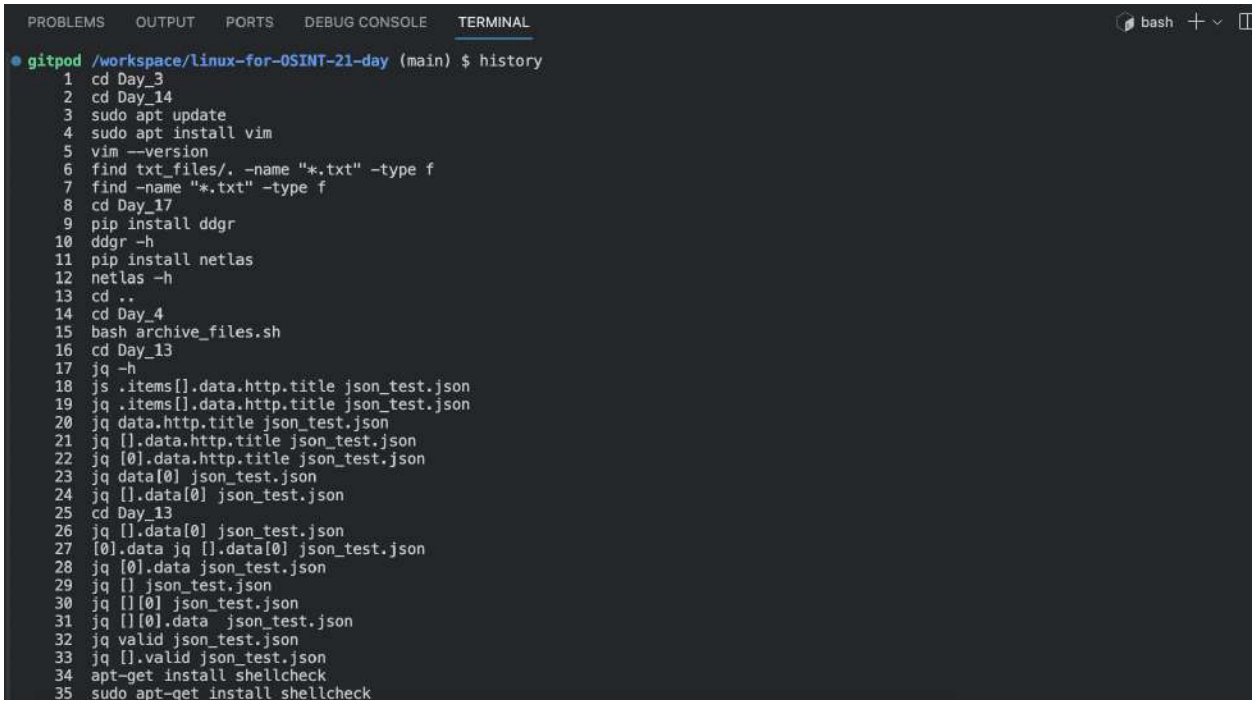
The most important reason for the loss of efficiency in Linux (especially among beginners) is syntax errors when entering commands. TheFuck (https://github.com/nvbn/thefuck) is a utility that corrects errors in the command you entered before. Let's install it and see how it works.



Unfortunately, at the moment this tool failed to install on Gitpod (due to certain issues with Python customizations) and I'm showing you a picture from the official documentation.

Bash history

Linux has an command that allows you to see a history of what you've typed in the terminal.



Just type at the command line:

history

Don't forget that you can save the results of running the command to a file and do text searches with Grep.

That's it for today, but there are many more different tools to make working with the Linux command line and writing bash scripts easier. But, unfortunately, the format of this book does not allow to cover all of them. In fact, such a conclusion can be written for all the previous chapters.

Day 21. Which Linux distribution is better to use?

The short answer is the one that you are most comfortable working with and the one that you put the most effort into setting up. Try different distributions. But I would still recommend that you start with the ones **that are actively developing at the moment and have a large community** of users around them.

Here are some examples of distributions that were created for cybersecurity researchers:

Tsurugi Linux Acquire (https://tsurugi-linux.org/tsurugi_acquire.php) - lightweight distribution (the installation file is just over 1 GB, there is a 32-bit version, very good for running from a USB device) with many pre-installed tools for OSINT.
Kali Linux (https://www.kali.org/get-kali/) - the world's most popular distribution for pentest and ethical hacking.
CSI LInux (https://csilinux.com) - complete “cyber forensic” platform based on Ubuntu.
BlackArch (https://blackarch.org) - lightweight (Slim ISO 5.5 Gb) expansion to Arch Linux for penetration testers.

Here are a couple more long lists of Linux distributions for cybersecurity professionals:

Distro Forensics Github repository (https://github.com/CScorza/DistroForensics)
Linux for OSINT Twitter list (https://twitter.com/cyb_detective/status/1441686068173033473)

But if you work in cybersecurity, that doesn't mean you really need to use distributions that are designed for cybersecurity professionals. It's not necessary at all.

And there is absolutely no way to say that one distribution is better than another. After all, the answer to that question is different for each person and changes over time.

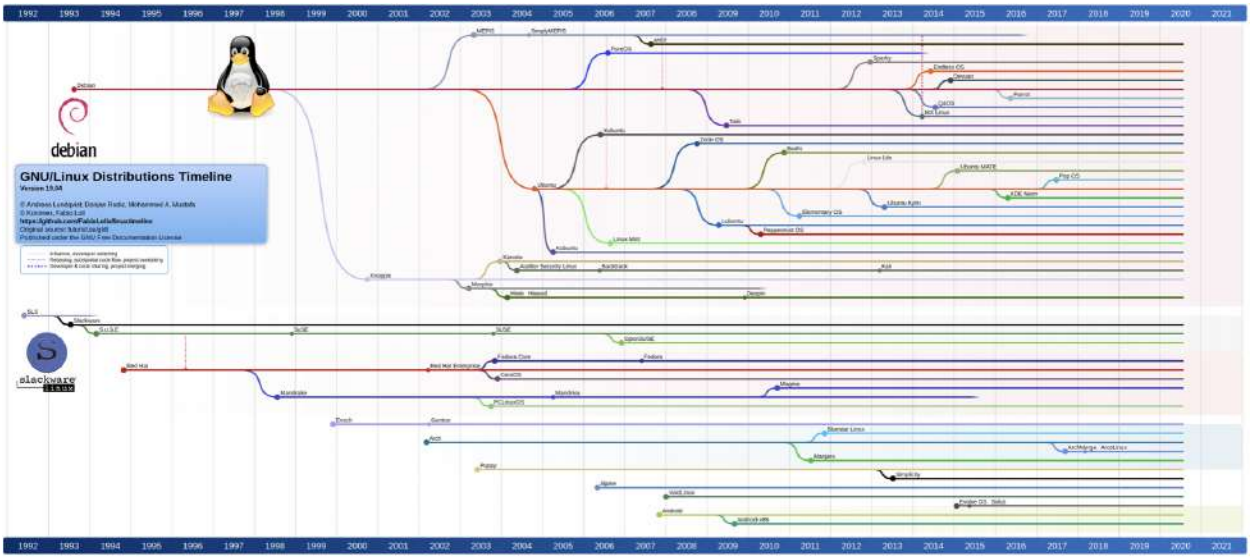


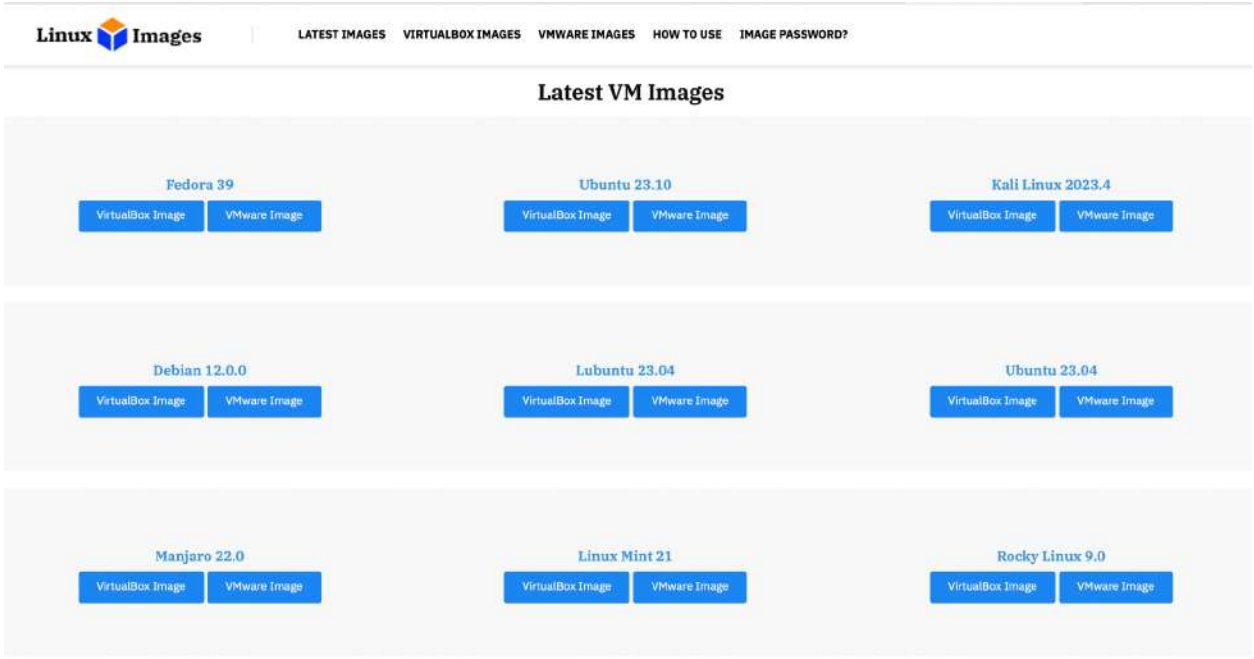
Image from Reddit.

There are hundreds of different Linux distributions and their variations (see picture above). Choose the one that:

- suits your computer's system requirements and runs quickly
- is convenient and understandable for you personally (you can only find out which one by practice)

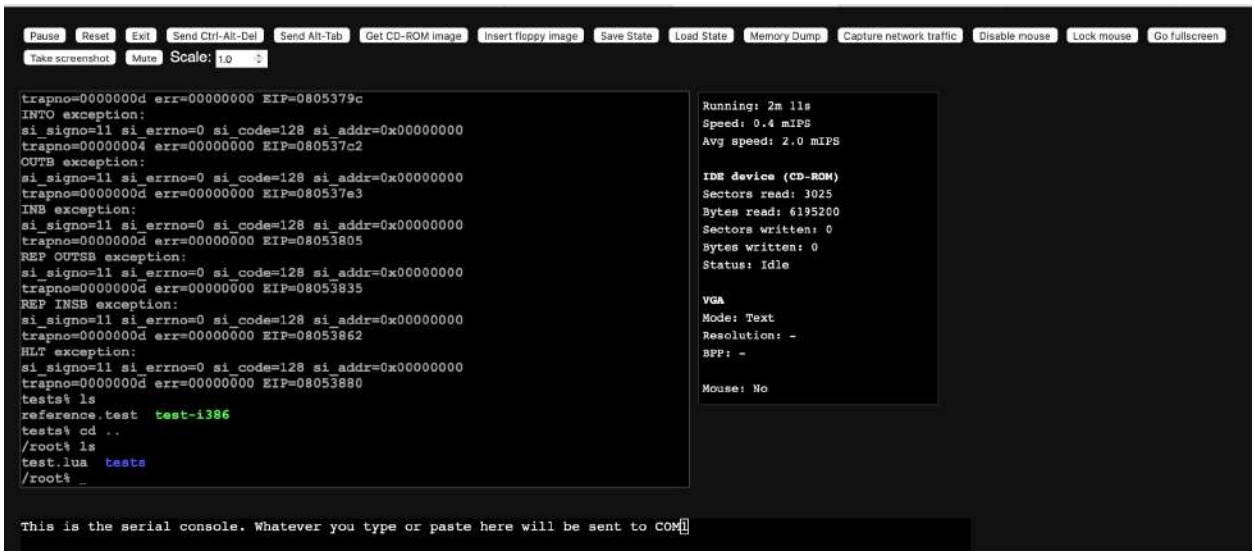
Your work performance depends on how much effort you put into customizing the distribution for your personal tasks. The toolkit is different for each researcher anyway and you'll still have to install a lot of the tools yourself.

And most importantly, remember that the Linux philosophy is about no rules, maximum flexibility and maximum attention for the user's individuality and goals.



You can use Linux as a virtual machine. Download images of different distributions in LinuxVMImages website (https://www.linuxvmimages.com/).

But think about other forms of usage such as VPS (virtual private server). The cost of renting them starts at a few dollars a month (few examples from $3.99 to $15) or bootable USB drive (example of instruction for Ubuntu and Windows https://ubuntu.com/tutorials/create-a-usb-stick-on-windows#1-overview).



Maybe after reading this book, Gitpod or even just browser-based versions of Linux will be enough for you:

https://copy.sh/v86/?profile=linux26
https://www.masswerk.at/jsuix/index.html
https://bellard.org/jslinux/vm.html?cpu=riscv64&url=fedora33-riscv.cfg&mem=

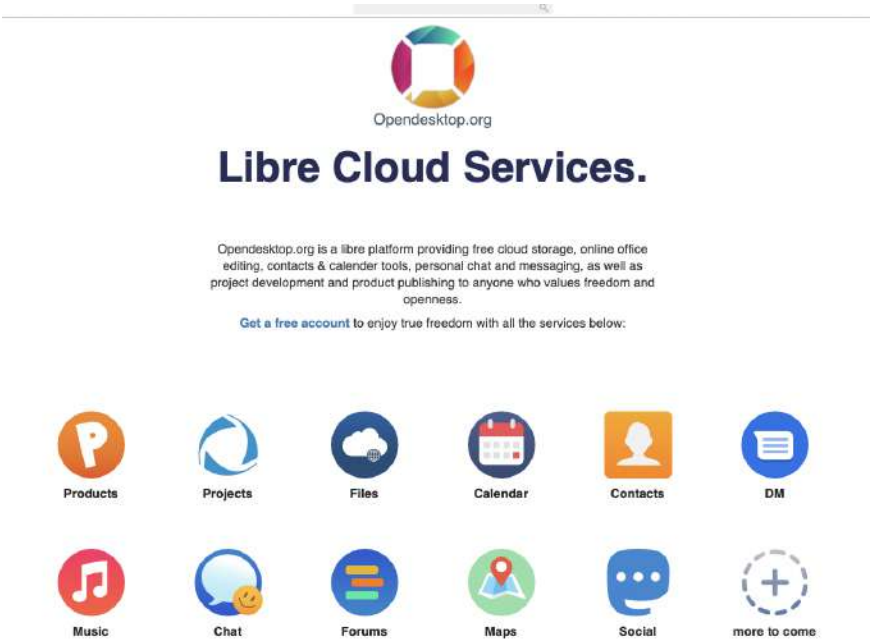
You might be comfortable using Linux on an Android smartphone using terminal emulators such as Termux (https://termux.dev/en/).

You will never know what's best for you until you try to solve your personal everyday problems.

Does it make sense to use Linux as your primary operating system

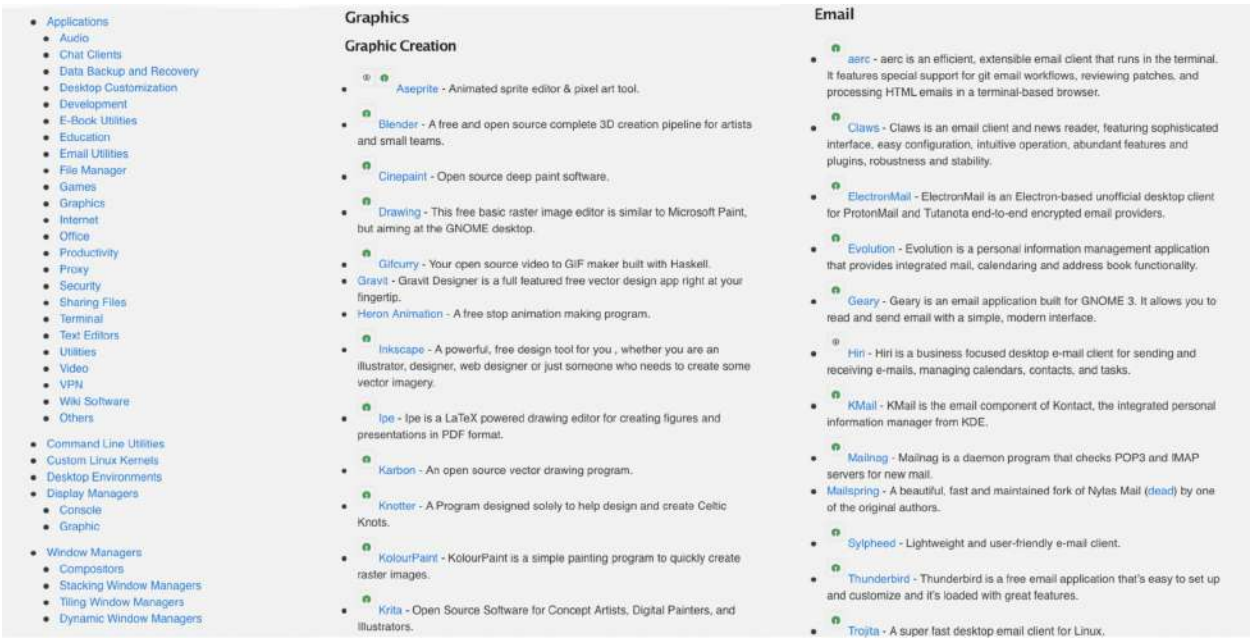
To be honest, I've only worked on a MacBook for 9 years now. And I do much of my work in macOS.

I use Linux several times a week to automate various tasks. But there are so many people who use it all the time and there are several reasons why this is a good practice.



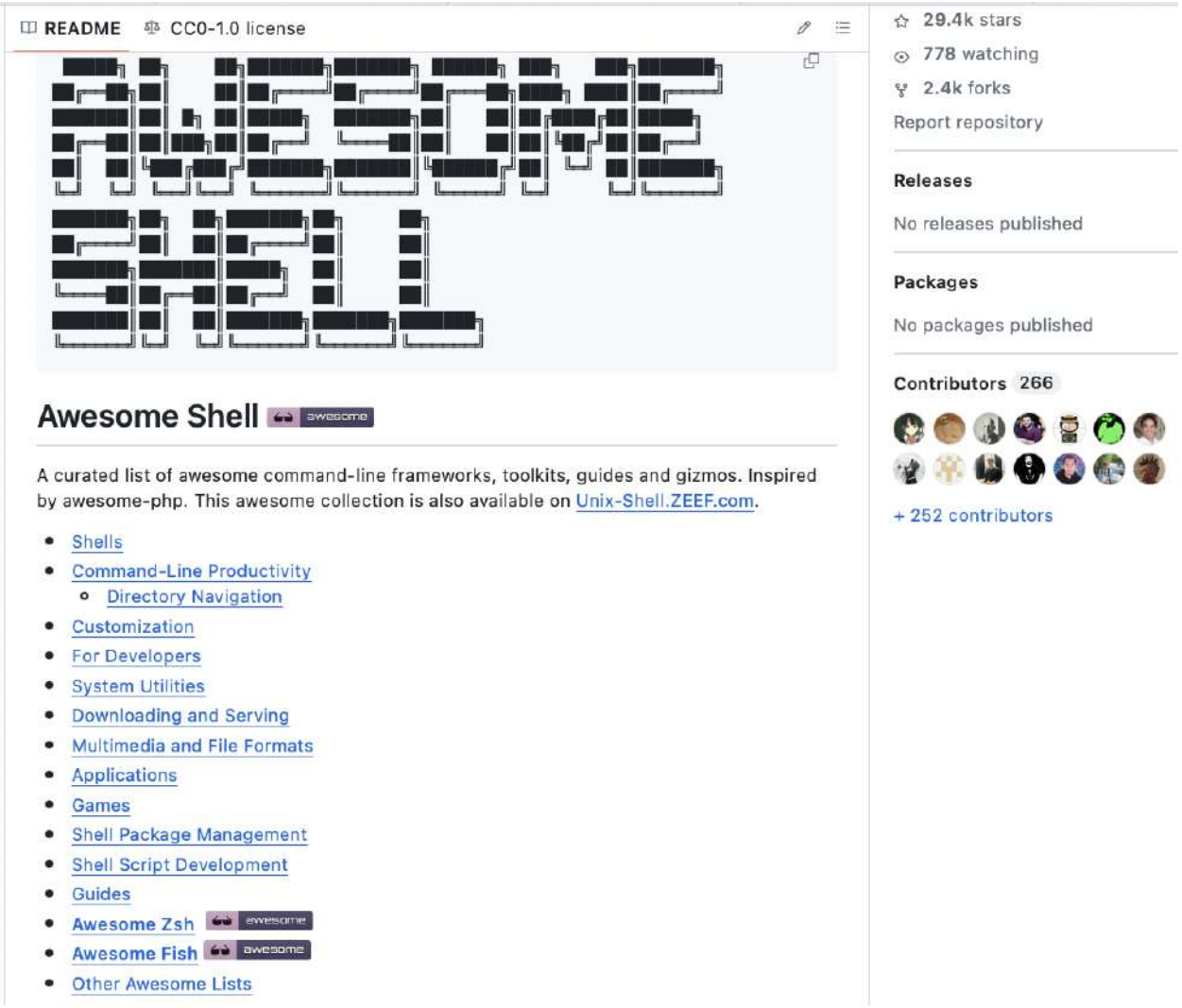
Firstly, it is an opportunity to significantly reduce the influence of global corporations on your life. If this idea inspires you, you can also opt out of Microsoft and Google online services like calendar, maps, cloud file storages, as well as streaming services and social networks. You can find decentralised open source analogues for all this on OpenDesktop (https://www.opendesktop.org).

(But keep in mind that you won't get rid of the influence of corporations on your life until the end, as the leading technology companies spend a lot of money to support Linux foundation and Open Source community).



Secondly, it's a huge selection of FREE, yet LEGAL software for a wide variety of purposes. Check out the **Awesome Linux** Software (https://github.com/luong-komorebi/Awesome-Linux-Software/) list, which has thousands of links.

By the way, many of these apps also have versions for Windows and macOS.



You can see what a huge number of command line utilities exist if you open the repository Awesome Shell (https://github.com/alebcay/awesome-shell). It includes almost 400 links (and these are only the most useful and well-known utilities).

Third, if you do need some applications that only exist on Windows, you can still run them on Linux using launchers such as:

Wine (http://winehq.org)
CrossOver (http://codeweavers.com/crossover)

Unfortunately, they are not always convenient and often produce errors. For example, I recently failed to run a programme for viewing dental images on Wine. But in many cases, they do work.

Fourth, Linux allows you maximum flexibility in customizing your desktop GUI (incomparable to Windows and Mac). You can start by trying one of the popular Linux Desktop Environments:

Gnome 3 (https://www.gnome.org/gnome-3/)
KDE (https://www.kde.org/)
Cinnamon (http://developer.linuxmint.com/projects.html)
Mate (http://mate-desktop.com/)
Xfce (http://www.xfce.org)

What to do next?

Practice as much as possible. Keep doing OSINT investigations and try to automate tasks with Linux utilities. Don't be afraid to make mistakes and don't hesitate to use Stackoverflow and AI tools described in **Day 20**.

Don't forget to subscribe to me on Twitter, Mastodon, Telegram, BlueSky, Substack, Medium to get regular news about new OSINT tools, and not to miss new books. All links can be found at cybdetective.com.

Once you've had at least a little bit of practice and you want to learn something new again, you can also read “Python for OSINT. A 21-day course for beginners” (https://github.com/cipher387/python-for-OSINT-21-days). It's also distributed freely (donate what you want and if you want).

Also remember that the global open source community is growing due to the small efforts of each of the tens of millions of enthusiasts around the world.

Don't forget to write about open source tools in your social networks, support developers with good reviews and donations, and if you like programming, don't hesitate to contribute to various projects on Github. Even if you don't know much yet, you can help with testing or writing documentation.

Application. Is it possible to do the same thing on Windows?

Yes, it is possible and plenty of people do it. But as mentioned above, it's easier to run Linux on VM or VPS than to think about how to do the same thing on Windows.

The most justifiable reason to run Linux utilities on Windows - it is to combine some complex automations created with PowerShell or VBA with automations based on Linux utilities. In that case, here are some instructions on how to do it:

How to install bash in Windows from HackerNoon
(https://hackernoon.com/how-to-install-bash-on-windows-10-lqb73yj3)

How to run Linux software on Windows from GeeksForGeeks
(https://www.geeksforgeeks.org/how-to-run-linux-software-on-windows/)