Computing IV Sec 201: Project Portfolio

Anderson Torres

Fall 2022

Contents

Time to Complete Portfolio: **12 hours**

1 PS0: Hello SFML

1.1 Discussion

Hello World was our first Computing IV assignment. The main goal of this assignment was to setup our Linux build environment and test out the SFML library. This included getting Linux running – either through a Virtual box, native or Windows server for Linux(WSL). We started with the demo and expanded from there. Linux was a bit new for me at the moment, it was easy to get used to it and it didn't take very long to setup my environment – a few updates later i was able to get the demo working and starter my own expansion of it. There wasn't much new information to get the project done but it was a good start too make us more interested in graphical user interface (GUI).
the program consisted of basic programming such as variables, loops, conditionals and objects. The idea of this assignment was just to test out SFML's different classes, which included Images, Sprites, Text, Texture, and Keyboard. The Texture / Image / Sprite classes also interact with each other For this assignment, I created a program that uses SFML's sprite class to display a moving image on an SFML window. I added some basic controls to move it around – arrow keys for up/down/left/right and a Easter egg image. 1.

Figure 1: images used for this project

1.2 What I accomplished

I successfully implemented a window that dynamically changed the displayed image based on the position of a circle, creating a captivating effect reminiscent of seasonal transitions. Moreover, users could interact with the application by adjusting the circle's position using the arrow keys on the keyboard. Additionally, pressing the letter "C" centered the ball within the window, providing a convenient way to reset its position. This interactive functionality enhanced the user experience and allowed for greater engagement with the visual representation of changing seasons.

1.3 What I already knew

I had some prior knowledge of CPP from previous classes, but most of my experience was limited to command line assignments. While I understood certain concepts individually, I hadn't yet had the opportunity to integrate them into a cohesive project.

1.4 What I learned

- Learned the basics of utilizing SFML for multimedia tasks.
- Learned the process of displaying images in an SFML window and controlling sprites using SFML's Keyboard library.
- Set up a build environment tailored for Linux.
- Acquired proficiency in creating and utilizing makefiles for project compilation and management.

1.5 Challenges

Setting up the ball to trigger the appropriate picture and ensuring that the positioning of the images was accurate was a meticulous process. It involved meticulous coordination between the ball's movements and the activation of specific images, ensuring that they aligned seamlessly with the intended visual narrative or gameplay experience. Additionally, attention to detail was crucial to ensure that the images were positioned correctly within the game environment, enhancing the overall aesthetic appeal and gameplay immersion.

2 PS1: PhotoMagic

2.1 Discussion

This task demanded the implementation of a Linear Feedback Shift Register (LFSR). The register shifts all bits to the left by one position, then performs an XOR operation between the leftmost bit and a seed bit to occupy the empty space on the far right after the left shift. Our primary objective was to generate a pseudo-random and predictable output. Additionally, we aimed to gain proficiency in implementing unit tests using the Boost test framework.

Furthermore, left shifting was achieved by inputting a string, verifying the data, converting it into a C++ bit-set, and then appending the result of the XOR operation to the leftmost bit with the tap position.

We used the Boost test framework to check our LFSR classes thoroughly. With Boost's auto test cases, we made sure all our functions could handle edge cases.

Our LFSR class relies on a shift register to hold bits. It offers two methods, step and generate, to help shift all the bits to the left.

We also made good use of SFML objects like textures, images, and sprites. These helped us read files, encode them, and display the final encoded image both on the screen and in a directory. The image class was crucial for encoding the image.

To encode the image, we simply XORed each pixel with our pseudo-random generator LFSR, resulting in an unrecognizable image. 2.

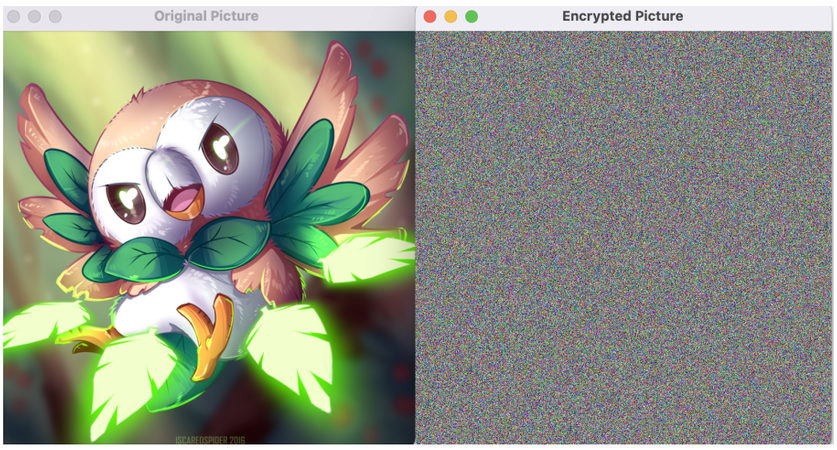

Figure 2: this is our input image

2.2 What I accomplished

I successfully implemented a Linear Feedback Shift Register (LFSR) to generate pseudo-random numbers. This technique allowed me to encode an image and decode it using the same key. Moreover, I could use both text and binary as keys. Ultimately, the encoded image remained unrecognizable unless the correct key was provided.

2.3 What I already knew

During my time at NECC, I took a cybersecurity class that gave me some great insights into how projects like this one should be handled. It was eye-opening to learn about encryption methods, security protocols, and ways to keep data safe from cyber threats. These lessons are definitely coming in handy as I work on this project.

My journey into programming started with my first Java class, where one of the assignments involved encrypting data within an image. It was a cool introduction to the world of encryption and got me interested in exploring more about digital security.

Also, I've dabbled in the SFML (Simple and Fast Multimedia Library) during previous projects. It's been a useful tool for handling multimedia tasks, and I've picked up some basics along the way. I'm hoping to leverage that knowledge to make the encryption process smoother and maybe even improve the user experience.

So, between my cybersecurity insights from NECC, my Java encryption experiments, and my experience with SFML, I feel like I've got a pretty good foundation for tackling this project. It's exciting to see how everything I've learned so far can come together to create something cool and useful.

2.4 What I learned

- Strengthened understanding of encryption principles and its significance in data security.
- Acquired proficiency in overloading operators in C++ for enhanced functionality.
- Mastered the technique of reading input from command line-provided files in C++.
- Familiarized myself with various C++ standard libraries, broadening my programming toolkit.
- Gained expertise in utilizing Boost to generate comprehensive unit tests, ensuring code reliability and robustness.

2.5 Challenges

There were many challenges to get this program to work.

- Determined the appropriate data structure for storing the data and implemented it effectively.
- Selected the relevant SFML class to utilize for the project's requirements.
- Developed a strategy for combining pixels with random numbers to achieve desired encryption.
- Implemented measures to prevent user input errors and ensure data integrity.

3 PS3: StringSound

3.1 Discussion

We successfully implemented a Circular buffer, incorporating comprehensive unit tests and exception handling. The Circular buffer played a pivotal role in our project, operating by wrapping around like a circular array to efficiently store values. Boost unit tests were instrumental in validating the functionality of the Circular buffer, ensuring its reliability and accuracy. Various exceptions were incorporated to handle specific errors, such as throwing an invalid argument exception if attempting to create a Circular buffer with a capacity of 0 or less, or a runtime error if attempting to enqueue a full Circular buffer, or dequeue or peek at an empty Circular buffer.

In the subsequent phase of our project, we utilized the Circular buffer from PS5a to construct a model of a Guitar, employing the Karplus-Strong algorithm to simulate the plucking of guitar strings. Our task involved implementing key methods to emulate guitar behavior, including plucking, tic, and sample functionalities. Furthermore, we enhanced user interaction by enabling the main program, KSguitarSim, to respond to keyboard presses, with each key generating a different note. The Karplus-Strong algorithm, which models frequencies, operates by averaging the first two values and then multiplying the result by an energy decay factor, typically set to 0.996. Leveraging the Circular buffer in conjunction with this algorithm allowed us to simulate sound to a considerable extent, creating the illusion of plucked guitar strings.

Moreover, we explored the versatility of our implementation by modifying the data to produce sounds akin to those found in horror movies, showcasing the adaptability and creativity inherent in our project. 3.

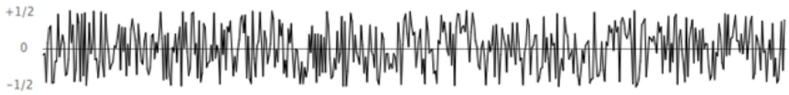

Figure 3: Frequency wavelength

3.2 What I accomplished

I successfully developed a circular buffer and harnessed the power of the Karplus-Strong algorithm to generate sound. Beginning with the creation of the buffer itself, which served as the cornerstone of the entire program, I prioritized its implementation as the most crucial aspect. Once established, the implementation process was relatively straightforward, allowing me to progress efficiently.

3.3 What I already knew

- Pointers: I had a basic understanding of pointers, which are variables that store memory addresses. They are powerful tools in C++ for dynamic memory allocation and manipulation.
- Exceptions: I had some knowledge of exceptions, which are mechanisms in C++ for handling errors and exceptional situations during program execution.
- SFML Keyboard Basics: I had a basic understanding of handling keyboard input in SFML, a multimedia library for C++. This knowledge enabled me to incorporate user interaction into my projects.
- Basic Algorithms in C++: I was acquainted with some fundamental algorithms in C++, which are essential for performing various tasks efficiently, such as sorting and searching data structures.

3.4 What I learned

- CPP Templates and Smart Pointers: I gained proficiency in utilizing C++ templates and smart pointers, which are powerful features for generic programming and memory management, respectively.
- Creating Custom Data Structures: I learned how to design and implement data structures tailored to specific project requirements, enhancing the efficiency and organization of my code.
- Exception Handling: I deepened my understanding of exception handling in C++, mastering techniques for gracefully managing errors and exceptional situations in my programs.
- Advanced Knowledge of CPP Standard Libraries: I expanded my knowledge of the C++ standard libraries, exploring additional functionalities and leveraging them to enhance my projects.
- Advanced Algorithms in C++: I delved into more advanced algorithms in C++, equipping myself with a broader toolkit for solving complex computational problems.
- Representing Data as Sound: I acquired the ability to represent data as sound, exploring methods for converting numerical data into audible signals for creative and practical applications.
- CPPLint: I familiarized myself with CPPLint, a style guide for C++, which helped me maintain consistent and readable code throughout my projects
- CPP Random Library, Especially Uniform Distribution: I learned about the CPP random library and its capabilities, with a particular focus on generating random numbers following a uniform distribution, facilitating probabilistic simulations and randomness-based functionalities in my programs.

3.5 Challenges

There were many challenges to get this program to work.

- Determining the most suitable data structure to hold the data and implementing it effectively posed a significant challenge, requiring careful consideration of the project's requirements and performance considerations.
- Understanding the intricacies of SFML sound libraries presented a steep learning curve, as I grappled with their documentation and functionality to achieve the desired audio effects.
- Accurately interpreting the Karplus-Strong algorithm proved challenging, as its implementation involved complex mathematical concepts and acoustic principles that required thorough comprehension for effective utilization.
- Testing the program effectively posed a challenge, as I had to develop comprehensive test cases to ensure the correctness and reliability of the code, requiring meticulous attention to detail and thorough testing procedures.
- Ensuring a perfect Circular Buffer and testing its functionality rigorously presented challenges, as I had to account for various edge cases and handle potential errors effectively to guarantee the buffer's stability and correctness.

4 PS4: Sokoban

4.1 Discussion

In this assignment, several key concepts were explored. Initially, we utilized the command line operator "<" to efficiently read a file into standard input/output. Within the main program, file contents were seamlessly read using the istream, simplifying the data input process. Additionally, to streamline data input, we implemented the overload of the "»" operator, enabling a concise syntax such as "istream » sokoban" for reading data.
Another pivotal aspect of the assignment involved implementing the draw method within our Sokoban class. Although unfamiliar initially, experimenting with this method proved insightful as we navigated through its functionalities to achieve desired outcomes. Overloading the "»" operator provided valuable insights, expanding our understanding of operator overloading in C++.
Fortunately, prior experience with SFML's textures, images, and sprite objects facilitated the display of tiles without significant hurdles. However, mastering collision detection and ensuring proper positioning of tiles presented notable challenges. It required iterative adjustments and troubleshooting, as tiles intermittently failed to appear or appeared incorrectly. Nonetheless, upon resolving these issues, the gratification of observing neatly aligned tiles was immensely satisfying, highlighting the culmination of our efforts in refining the Sokoban game implementation. 4.

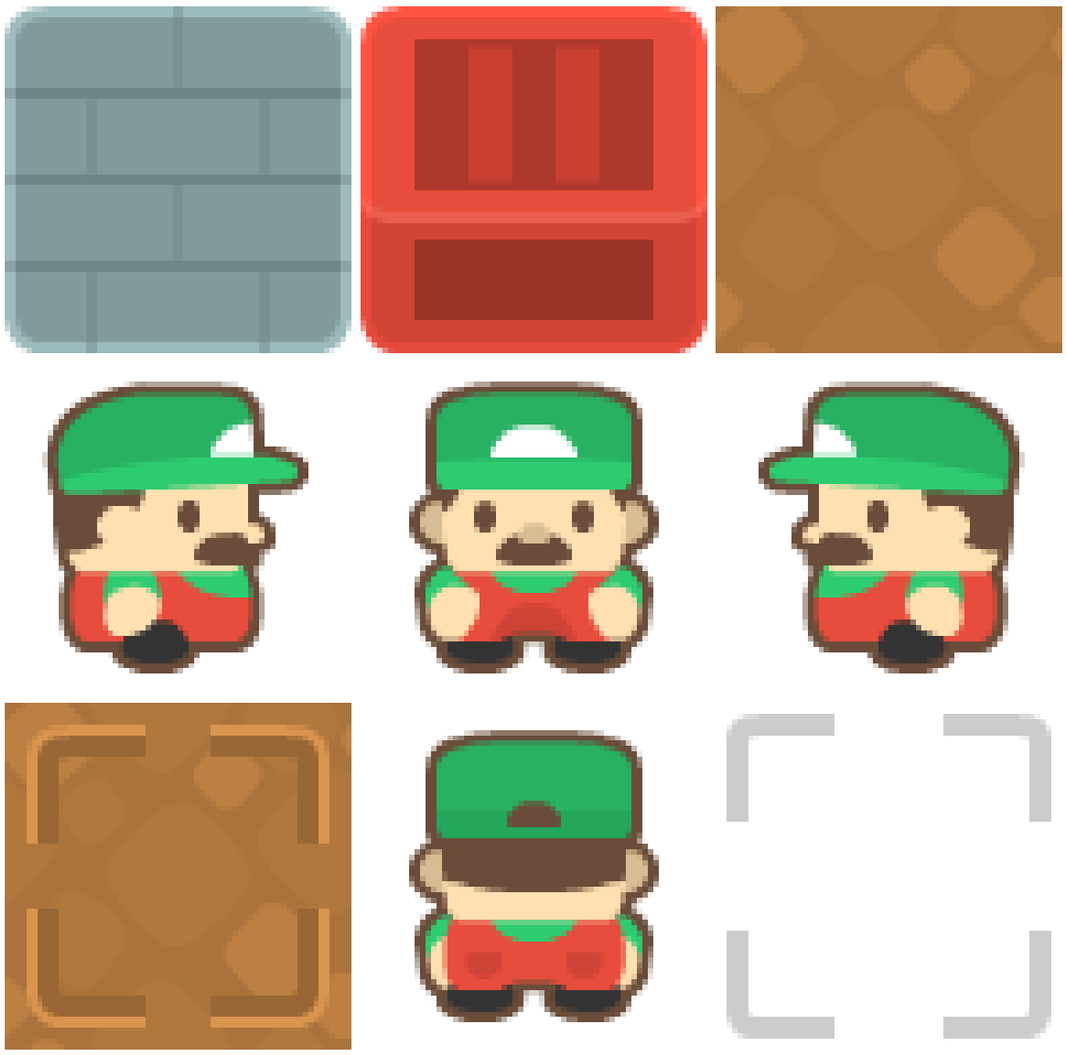

Figure 4: game tiles

4.2 What I accomplished

In this project, I successfully developed the Sokoban game, also known as "warehouse man". Initially, I began by creating a basic representation of the game using ASCII

characters. With guidance from the professor, I adopted a matrix implementation for the game grid, enabling the recreation of the Sokoban game's ASCII representation effectively. Key components pivotal to the success of this project included the implementation of insertion/extraction overloads, which facilitated seamless input/output operations with the game data. Additionally, leveraging matrix mathematics allowed for precise positioning of the game tiles within the grid.

The process of displaying the tiles involved looping within another loop, systematically iterating over the grid to showcase all the game elements accurately. This iterative approach ensured that each tile was properly visualized within the game environment.

Furthermore, the visualization of the matrix played a crucial role in shaping the development of this project. By visualizing the matrix structure, I gained valuable insights into how to effectively implement and manipulate game elements within the Sokoban game grid. Overall, the combination of these elements contributed to the successful creation of the Sokoban game, providing a rewarding experience in game development and matrix visualization techniques.

4.3 What I already knew

From prior programming projects, I gained familiarity with certain algorithms and had some exposure to SFML. While I learned how to handle spaces effectively, collisions remained a challenge that I hadn't fully addressed.

4.4 What I learned

- I gained proficiency in generating grids, a fundamental skill for various programming tasks.
- My understanding of C++ was broadened, particularly in algorithms and general programming concepts.
- I delved deeper into SFML, mastering sprite and other classes to enhance visual elements in my projects.
- Understanding and implementing collision detection was a significant learning experience, requiring problem-solving skills and attention to detail.
- I learned about managing program status effectively, which is crucial for maintaining program flow and user experience.
- This project gave me insight into the challenges and rewards of game development, providing a glimpse into the world of game developers.
- I honed my skills in overloading input and output in a more advanced manner, enabling more streamlined data handling in my programs.

4.5 Challenges

There were many challenges to get this program to work.

- Establishing a reliable method to track the player's movements throughout the game proved to be challenging.
- Implementing a draw function capable of accurately displaying all game elements posed a significant hurdle.
- Grappling with matrix positioning complexities presented a notable challenge during development.
- Generating a grid that effectively represented the game environment required careful consideration and implementation.
- Determining the conditions to identify when the game had been won proved to be a challenging aspect of the project.

5 PS5: DNA Alignment

5.1 Discussion

In our project, we implemented a program aimed at determining the optimal alignment of two strings. A pivotal aspect of this program was the utilization of dynamic programming to efficiently compute the edit distance.

A central concept introduced in this program is the Needleman-Wunsch method, which leverages dynamic programming to solve subproblems and subsequently derive the main solution. This method involves utilizing an NxM matrix, where each cell corresponds to a particular edit distance calculation. The process begins by computing the simplest edit distances and progressively iterates through the matrix until reaching the solution at the [0][0] cell.

Moreover, we successfully reconstructed the path taken by our algorithm by retracing our steps through the matrix. This was achieved by adhering to specific rules:

When the optimal alignment matches x[i] with y[j], opt[i][j] equals opt[i+1][j+1] if x[i] equals y[j], or opt[i][j] equals opt[i+1][j+1] + 1 otherwise. When the optimal alignment matches x[i] with a gap, opt[i][j] equals opt[i+1][j] + 2. When the optimal alignment matches y[j] with a gap, opt[i][j] equals opt[i][j+1] + 2. By following these rules, we were able to traverse from the top-leftmost cell of the matrix, where the final edit distance was found, and backtrack to the bottom-rightmost cell ([N][M]). This process facilitated the identification of the optimal alignment path between the two strings.

5.2 What I accomplished

I accomplished the task of recreating alignments of letters, such as those found in DNA sequences. By employing the Needleman-Wunsch approach, I successfully determined the shortest and optimal alignment between two strings. This involved generating alignments of strings in a manner akin to the Needleman-Wunsch algorithm.

5.3 What I already knew

Prior to this project, my familiarity with using matrices was the extent of my knowledge. I had never encountered dynamic programming before, making it a completely new concept for me. Despite being unfamiliar, diving into dynamic programming was incredibly enjoyable. I found the problem-solving process intriguing, as there were numerous intricate ways to describe and approach the problem, which added to the excitement of tackling new challenges.

5.4 What I learned

- I discovered that vectors can be inefficient compared to C arrays, especially when dealing with large datasets. This understanding was gained by comparing the outputs of both data structures.
- I learned how to use Valgrind, a powerful tool for detecting memory leaks and other memory-related errors in C and C++ programs.
- The Needleman-Wunsch algorithm was a new concept for me, and I gained a thorough understanding of its application in sequence alignment problems
- I gained a basic understanding of dynamic programming principles, which proved invaluable in solving complex optimization problems efficiently.
- Exploring and learning other programming languages helped me become a more versatile and adaptable programmer, enhancing my problem-solving skills and widening my perspective on software development.
- I gained insight into the importance of optimization and efficient data management practices. For instance, identifying and fixing memory leaks in my code helped me improve its efficiency and performance.

5.5 Challenges

There were many challenges to get this program to work.

- Determining the appropriate data structure to hold the data and implementing it effectively posed a significant hurdle.
- Calculating the runtime of the program proved challenging, requiring a deep understanding of timing mechanisms and profiling techniques.
- Thinking in two dimensions, especially when dealing with complex algorithms or data structures, presented a notable challenge, as it required visualizing and managing data in multiple dimensions effectively.

6 PS6: RandWriter

6.1 Discussion

In this project, we developed a class that implements a model of natural language based on English text using Markov chains. Markov chains are statistical models of text that count the occurrences and sequences of characters in English words or sentences. Our MarkovModel class includes several functions to facilitate this process.

The freq() function calculates the frequency of a character in a k-gram, offering two versions—one for finding the frequency of k-grams and another for finding the frequency of a character following a given k-gram.

The randk() function generates a random character that follows the given k-gram, while the gen() function generates a string of random characters following a given k-gram, simulating a trajectory through the model.

Key concepts of the program include generating statistical models of English text and utilizing various functions to achieve this goal. These functions include freq(), randk(), and gen(), each contributing to the overall functionality and utility of the MarkovModel class.

6.2 What I accomplished

I accomplished the task of generating text using the model, often surprising myself with how closely it resembled the original text. By leveraging random numbers and k-gram information, I successfully generated text that maintained the same style as the original text. This process allowed me to simulate the natural flow and structure of the original text, showcasing the effectiveness of the model in capturing linguistic patterns and styles.

6.3 What I already knew

About the project nothing similar.

6.4 What I learned

- discovered that text can be predicted to some extent based on the occurrences of characters and sequences of characters. This insight into linguistic patterns deepened my understanding of natural language processing.
- I gained a greater appreciation for the role of mathematics in programming, particularly in algorithms such as Markov chains. Understanding mathematical concepts allowed me to implement more efficient and effective solutions.
- I expanded my knowledge of C++ algorithms, particularly those related to text generation and analysis. This project provided a hands-on opportunity to explore and apply various algorithms in a practical context.

6.5 Challenges

- Managing the data and ensuring the correct structure proved to be a significant hurdle. It required careful organization and manipulation of the data to fit the desired format and effectively utilize it within the model.
- Recreating a K-gram presented its own set of challenges. It required understanding the underlying principles of K-grams and implementing them accurately within the context of the project.

7 PS7: Kronos Log Parsing

7.1 Discussion

This assignment primarily focused on utilizing regular expressions, particularly leveraging Boost's regex library to locate matches within a string. Additionally, it provided an opportunity to become acquainted with date and time handling. I found Boost's regex library to be straightforward to use; simply employing 'boost::regex_match' or 'boost::regex_search' sufficed to identify matches against the regular expressions I formulated. The following expressions were pivotal for the success of the program:

bootBegin:

```
"([\\d]+)-([\\d]+)-([\\d]+) "
"([\\d]+):([\\d]+):([\\d]+): "
"\\(log.c.166\\) server started.*"
```

This regex pattern facilitated the detection of the boot statement. It identifies a matching line containing a timestamp followed by the phrase "(log.c.166) server started." Employing this regex against the entire statement facilitated the extraction of the date and time from the current line.

bootEnd:

```
"([\\d]+)-([\\d]+)-([\\d]+) "
"([\\d]+):([\\d]+):([\\d]+).([\\d]+):INFO:"
"oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080.*"
```

This regex pattern was instrumental in identifying a successful boot. A successful boot is marked by a timestamp followed by the string: "INFO:oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080." Similar to the boot statement, I extracted the date/time by passing a variable to the regex search/match methods.

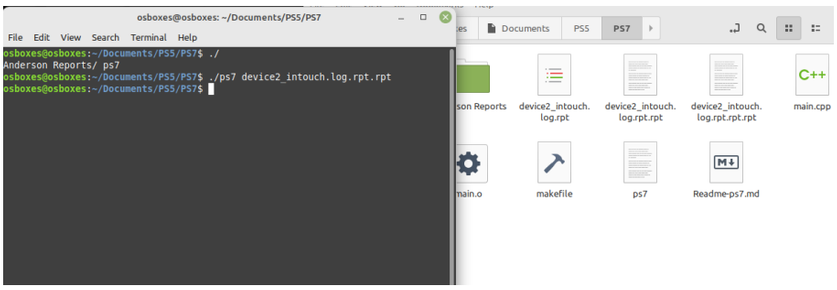

Figure 5: running the program

7.2 What I accomplished

I successfully generated a report detailing complete boots by utilizing date-based pattern matching. Additionally, I accomplished the extra credit task of extracting service names. This enhancement provided additional insights and enriched the generated report with comprehensive information.

7.3 What I already knew

I had limited knowledge about regex, primarily focusing on basic matching concepts.

7.4 What I learned

- I gained an understanding of how regular expressions work and their role in text processing.
- I learned the principles of pattern matching and how to apply them effectively.
- Through the implementation of regular expressions in this program, I feel confident in my ability to parse files and simplify data processing tasks in the future.
- I discovered the versatility and numerous applications of pattern matching, which can be utilized in various contexts beyond this project.

7.5 Challenges

- The only challenge I encountered was initially identifying a suitable pattern to match.

8 Sources

8.1 List of sources

- SpringPS-0: `https://images4.alphacoders.com/699/thumb-1920-699635.jpg`
- SummerPS-0: `https://wallpapercave.com/1920x1200-summer-wallpapers`
- FallPS-0: `https://wallpaperaccess.com/full/8082035.jpg` `https://wallpapercave.com/wp/wp2369327.jpg`
- EasterPS-0: `https://displate.com/displate/2638285?art=5d1e58186a8d0` `https://wallpapercave.com/wp/wp2369327.jpg`
- Rowlet-PS1 : `https://tmnt-x-pokemon.fandom.com/wiki/Rowlet`
- TrianglePS-2: was provided on class PDF for the assignment itself
- frequency and Piano PS-3: Provided on the project PDF
- SokobanPS-4: provided by professor but under license **License**

```
1
2
3     Sokoban (pack)
4
5     by  Kenney Vleugels (Kenney.nl)
6
7             ------------------------------
8
9     License (Creative Commons Zero, CC0)
10    http://creativecommons.org/publicdomain/zero/1.0/
11
12    You may use these assets in personal and commercial projects.
13    Credit (Kenney or www.kenney.nl) would be nice but is not mandatory
      .
14
15            ------------------------------
16
17    Donate:   http://support.kenney.nl
18    Request:  http://request.kenney.nl
19
20    Follow on Twitter for updates:
21    @KenneyNL
```

- algorithmPS-5: Provided on the project PDF Needleman-Wunsch
- algorithmPS-6: Provided on the project PDF Marvok-model
- algorithmPS-7: Provided on the project PDF Kronos Company based on Lowell MA
- audio for KSguitarSim: `https://mega.nz/file/Rt8zHA6Y#Gz2miMvaP5hCgxe-OwvtnewuiTZESikwvtDXJdOBqeI`