Høyskolen Kristiania

Final Exam

Programmering i C for Linux

Course code: PG3401

Candidate number: 6

Date: 30.04 – 14.05.2025

Semester: Spring 2025

Table of Contents

Task 1. Theory

A. Explain what the C programming language can be used for.

The C programming language is a general-purpose language, meaning it can be used in a wide range of applications without being restricted to any specific area of use. It was originally developed for system programming, but has since found use in several domains. Despite being a somewhat small and not as “high-level” as many modern programming languages, C’s minimalistic design and flexibility make it ideal for developing highly efficient and specialized applications where performance is a priority. This is why C is frequently used in scientific research, exploration, and defense sectors, particularly for programming embedded systems with limited resources, operating systems, device drivers, as well as some AAA games. C gives you the possibility to get very close to the hardware and gain full control of the platform it operates on, allowing developers to optimize performance and tailor the application to their specific needs.

A notable example of C’s application is the UNIX operating system, which, along with its compiler and various UNIX applications, is all written in C. This shows how robust and adaptable the language can be, as programs written in C can typically run on any machine that supports the language with little to no modifications. C has also been the foundation for the development of other operating systems, such as Linux and Windows, as well as other applications, game development, and high-performance computing. C’s ability to communicate directly with hardware and its efficient execution make it a preferred choice for developers who want to maximize performance and resource utilization. (Kernighan & Ritchie, 1988, p. 8) (PG3401_Lecture_01_Intro, slide. 37, 40)

B. Who is Dennis Ritchie, and what is he known for in the field of Information Technology?

Dennis Ritchie earned a bachelor's degree in physics from Harvard University in 1963. His interest in computer science was sparked by a lecture on the UNIVAC I, leading him to join Bell Labs, a research and development company, shortly after his graduation in 1967.

In the early 1970s, Ritchie developed the C programming language, and together with his colleague Ken Thompson, they created the UNIX operating system. Their goal was to design an operating system that would improve interaction and information sharing among programmers. Writing UNIX in C enhanced its portability, as C was intentionally designed as a minimalist language suitable for writing operating systems for minicomputers with limited memory. The language quickly gained popularity through the book "*The C Programming Language* ", co-written by Brian W. Kernighan and Dennis Ritchie. This publication made it possible for many programmers to learn and write in C.

The success of C also led to the development of object-oriented languages like Java and C++. Both C and UNIX remain widely used today, powering everything from supercomputers and smartphones to games and embedded systems. Dennis Ritchie received many honors throughout his career, including the Turing Award in 1983, the National Medal of Technology in 1998, and in 2019, he was added to the National Inventors Hall of Fame. These awards are some of the highest honors in computer science, and they reflect his impact on the field.

Ritchie's work still influences modern computing today, providing a foundation for many other technologies. (National Inventors Hall of Fame, 2019) (EBSCO, 2024)

C. List at least 5 different Linux "distros", explain briefly their background, if they have any specific purpose, and how they differ from each other.

In 1993, Debian was established by Ian Murdock with the goal of creating a universal, free, and open-source operating system. Debian has been the foundation for several other distros, such as Ubuntu. Debian is a preferred choice for servers and mission-critical systems, as it is very stable and reliable with a detailed testing process. Debian is developed and maintained by a dedicated community of volunteers, and it is therefore easy and quick to get help from active users. (Geeks for Geeks, *8 Most Popular Linux Distributions*, 2025) (Geeks for Geeks, *Introduction to Debian Linux*, 2024)

In 2002, Red Hat Enterprise Linux was developed by Red Hat and became a popular choice for businesses and organizations that need robust and reliable operating systems to support their mission-critical applications and large workloads. This, together with strong security and stability, makes companies willing to pay for the Red Hat Enterprise Linux distribution. In comparison, CentOS is another distro that is based on RHEL's source code, but is a free version. CentOS is therefore extremely popular due to it being free but having the same enterprise-class performance and level of security as RHEL. (Geeks for Geeks, *8 Most Popular Linux Distributions*, 2025)

In 2002, Judd Vinet established Arch Linux because he was dissatisfied with how other operating systems lacked package management. Today, it follows a rolling release model, which means that it constantly offers the latest updates. Arch Linux is customizable and is very popular among users who like to set up and maintain their own systems themselves. It is a highly technical distribution and is therefore used by people who can effectively use it and like flexibility and control. In comparison, there is Linux Mint, which is an easy-to-use operating system. It is visually very similar to Windows and, therefore, a good choice for people transitioning from Windows to Linux. (Geeks for Geeks, *8 Most Popular Linux Distributions*, 2025) (Geeks for Geeks, *What is Arch Linux?*, 2025)

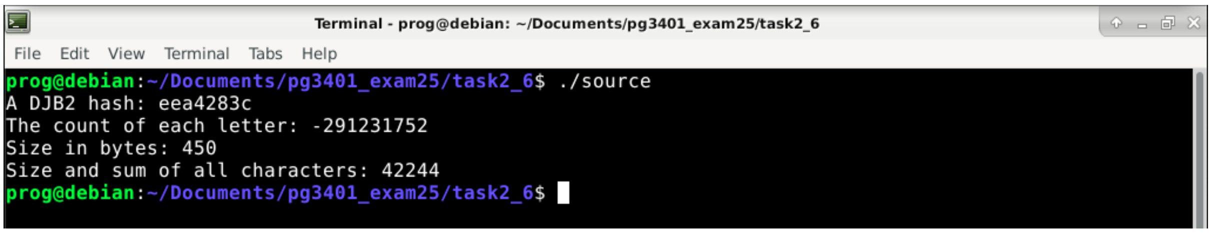
In 2004, the Ubuntu project was started by Mark Shuttleworth alongside a team of Debian developers. Ubuntu has gained popularity as a Linux distribution due to its user-friendly interface, together with robust security and stability that comes from regular updates and patches. Because of its popularity, the community support is large, resulting in detailed documentation, forums, and resources for troubleshooting and assistance. These are the reasons why Ubuntu is so popular and why it is a good choice for newcomers to Linux. (Geeks for Geeks, *8 Most Popular Linux Distributions*, 2025) (Geeks for Geeks, *What is Ubuntu?*, 2024)

Task 2. File management and functions

I started by including the necessary header files and defined the needed structure. I opened the provided text file for reading and called the three given functions on this file. A new output.bin file was then opened in write binary mode, and a metadata struct was initialized with the expected values. Four bytes from piHash were then copied into the byHash field, and by using fwrite, I saved the current state of the metadata struct to the output file. Finally, I printed the values to the terminal, freed all allocated memory, and closed all open files.

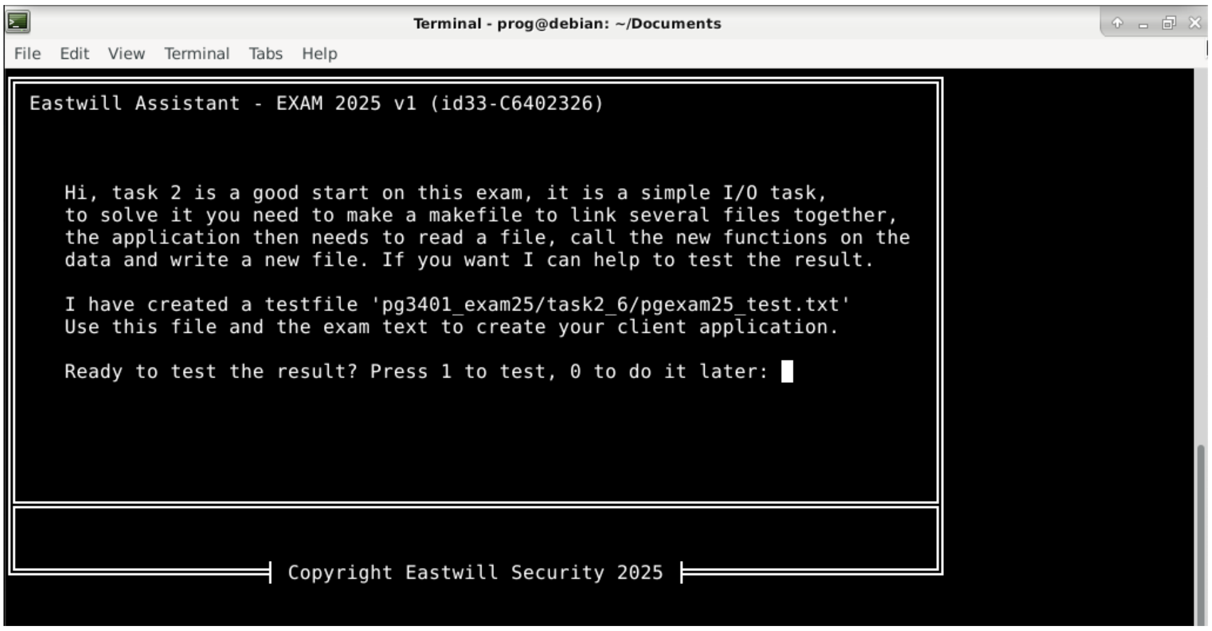
It should be noted that "The count of each letter" is not displayed in a human-readable format, as it's a list containing how many times each letter is present in the text file. This prevents overflowing the terminal. The value in the terminal output is not very informative, but it is written correctly in the .BIN file.

task2_scrn1.png


Terminal - prog@debian: ~/Documents/pg3401_exam25/task2_6
File Edit View Terminal Tabs Help

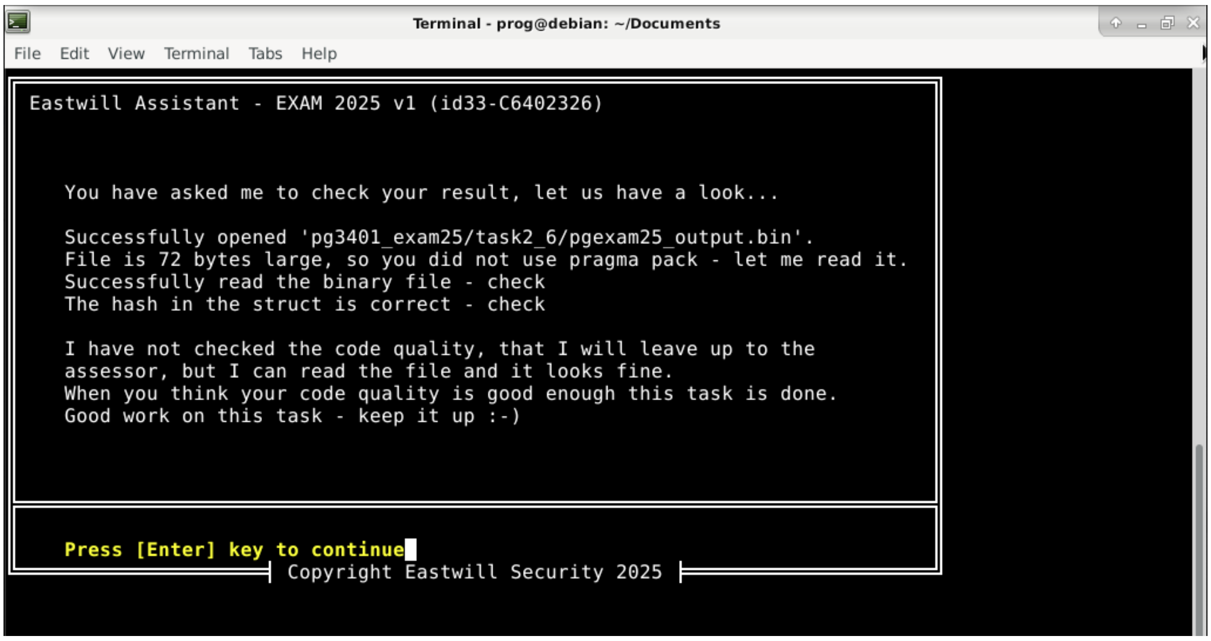
```
prog@debian:~/Documents/pg3401_exam25/task2_6$ ./source
A DJB2 hash: eea4283c
The count of each letter: -291231752
Size in bytes: 450
Size and sum of all characters: 42244
prog@debian:~/Documents/pg3401_exam25/task2_6$
```

task2_scrn2.png


Terminal - prog@debian: ~/Documents
File Edit View Terminal Tabs Help

```
Eastwill Assistant - EXAM 2025 v1 (id33-C6402326)


   Hi, task 2 is a good start on this exam, it is a simple I/O task,
   to solve it you need to make a makefile to link several files together,
   the application then needs to read a file, call the new functions on the
   data and write a new file. If you want I can help to test the result.

   I have created a testfile 'pg3401_exam25/task2_6/pgexam25_test.txt'
   Use this file and the exam text to create your client application.

   Ready to test the result? Press 1 to test, 0 to do it later:


                     Copyright Eastwill Security 2025
```

task2_scrn3.png

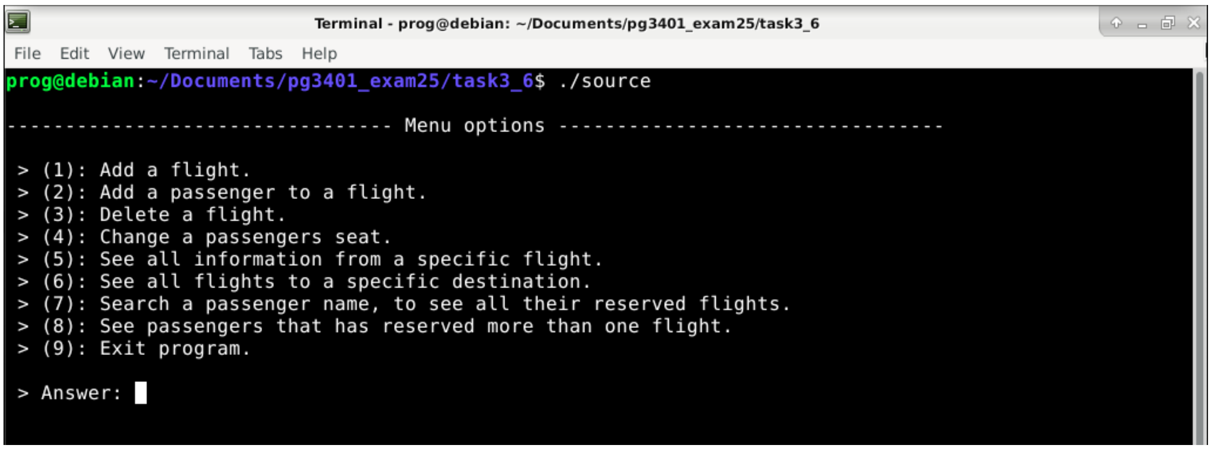

Terminal - prog@debian: ~/Documents
File Edit View Terminal Tabs Help

```
Eastwill Assistant - EXAM 2025 v1 (id33-C6402326)



   You have asked me to check your result, let us have a look...

   Successfully opened 'pg3401_exam25/task2_6/pgexam25_output.bin'.
   File is 72 bytes large, so you did not use pragma pack - let me read it.
   Successfully read the binary file - check
   The hash in the struct is correct - check

   I have not checked the code quality, that I will leave up to the
   assessor, but I can read the file and it looks fine.
   When you think your code quality is good enough this task is done.
   Good work on this task - keep it up :-)



   Press [Enter] key to continue
                         Copyright Eastwill Security 2025
```

Task 3. List handling

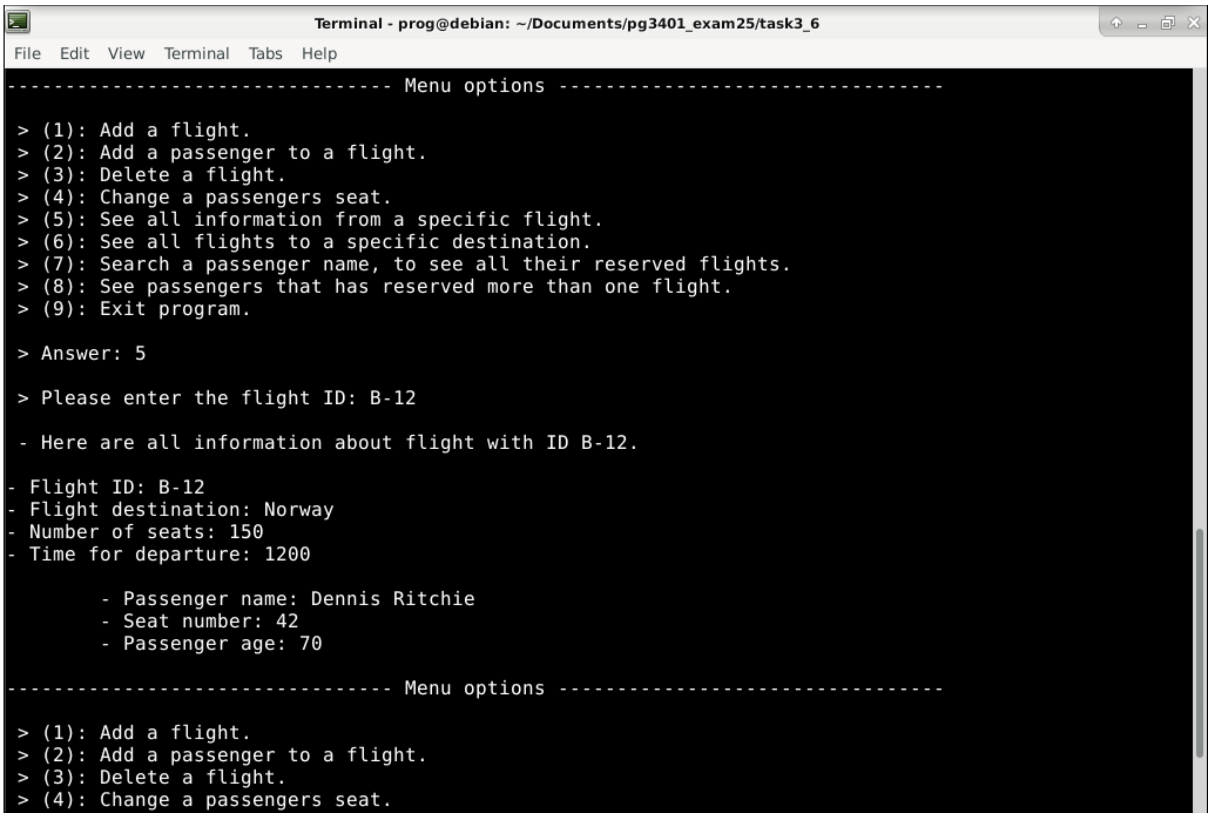
I have successfully implemented all the features asked for, while ensuring clean and readable code, by splitting the structures and their associated functions into separate .c and .h files. I went for a nested structure for passengers in planes, as it made the most sense to me during development. Still, you could also have two separate linked lists with references to each other, either a plane with a list of passenger IDs, or a passenger has a list of plane IDs.

task3_scrn1.png


Terminal - prog@debian: ~/Documents/pg3401_exam25/task3_6
File Edit View Terminal Tabs Help

```
prog@debian:~/Documents/pg3401_exam25/task3_6$ ./source

-------------------------------- Menu options --------------------------------

 > (1): Add a flight.
 > (2): Add a passenger to a flight.
 > (3): Delete a flight.
 > (4): Change a passengers seat.
 > (5): See all information from a specific flight.
 > (6): See all flights to a specific destination.
 > (7): Search a passenger name, to see all their reserved flights.
 > (8): See passengers that has reserved more than one flight.
 > (9): Exit program.

 > Answer:
```

task3_scrn2.png


Terminal - prog@debian: ~/Documents/pg3401_exam25/task3_6
File Edit View Terminal Tabs Help

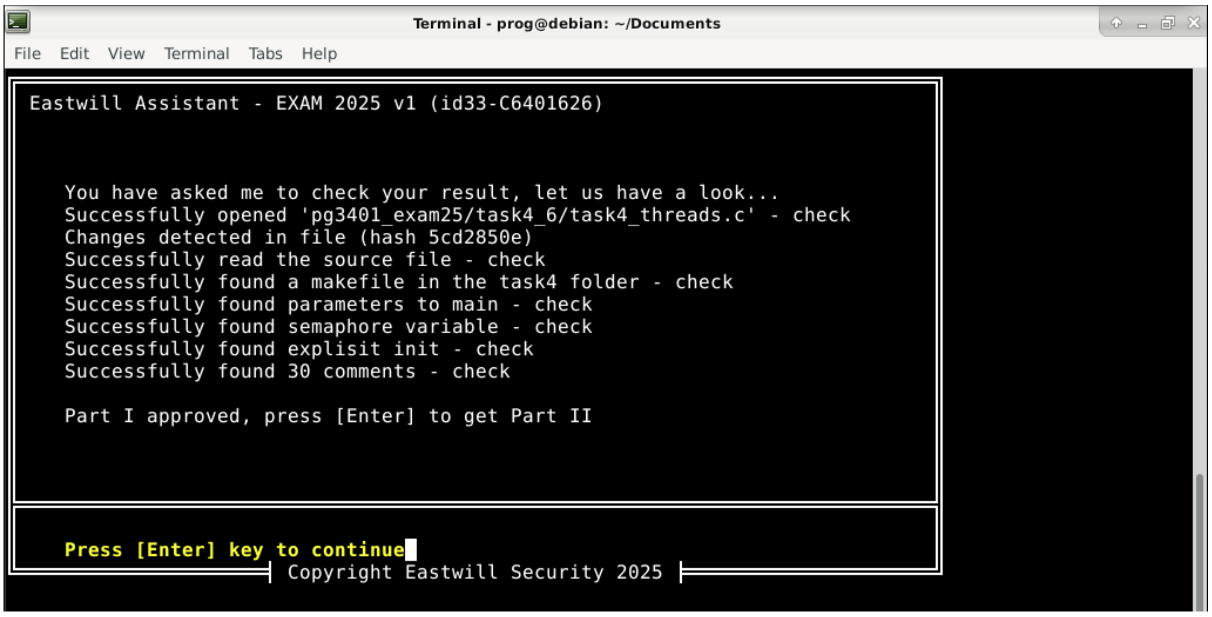
```
-------------------------------- Menu options --------------------------------

 > (1): Add a flight.
 > (2): Add a passenger to a flight.
 > (3): Delete a flight.
 > (4): Change a passengers seat.
 > (5): See all information from a specific flight.
 > (6): See all flights to a specific destination.
 > (7): Search a passenger name, to see all their reserved flights.
 > (8): See passengers that has reserved more than one flight.
 > (9): Exit program.

 > Answer: 5

 > Please enter the flight ID: B-12

 - Here are all information about flight with ID B-12.

- Flight ID: B-12
- Flight destination: Norway
- Number of seats: 150
- Time for departure: 1200

        - Passenger name: Dennis Ritchie
        - Seat number: 42
        - Passenger age: 70

-------------------------------- Menu options --------------------------------

 > (1): Add a flight.
 > (2): Add a passenger to a flight.
 > (3): Delete a flight.
 > (4): Change a passengers seat.
```

Task 4. Threads

The program no longer uses global variables, and I made a struct for the thread data so that I could keep all variables containing the thread data in one place and simply add more if needed. This made it easy to make changes and improved readability. When I changed the name of the file from being hardcoded to being passed through the command line as an argument, I made a new variable in my struct so that I had easy access to it when I needed to use it. I am using the POSIX semaphores library to use semaphores instead of conditions. The library gives me all the functions needed to control access to shared resources by multiple threads. Additionally, both mutexes and semaphores are now explicitly initialized by using the respective init functions.

I started on part.2 of the task and got the .txt file hashed, but I was not able to solve the TEA encryption. I attempted to read the documentation, look at the code, and search the web, but found myself unsuccessful in solving this task. I did however understand that TEA is a "Tiny Encryption Algorithm" and by using it on the hashed value, it will give me an encrypted version of it. This adds a layer of security since the hashed value won't be readable without the decryption key.

task4_scrnshot.png


Terminal - prog@debian: ~/Documents
File Edit View Terminal Tabs Help

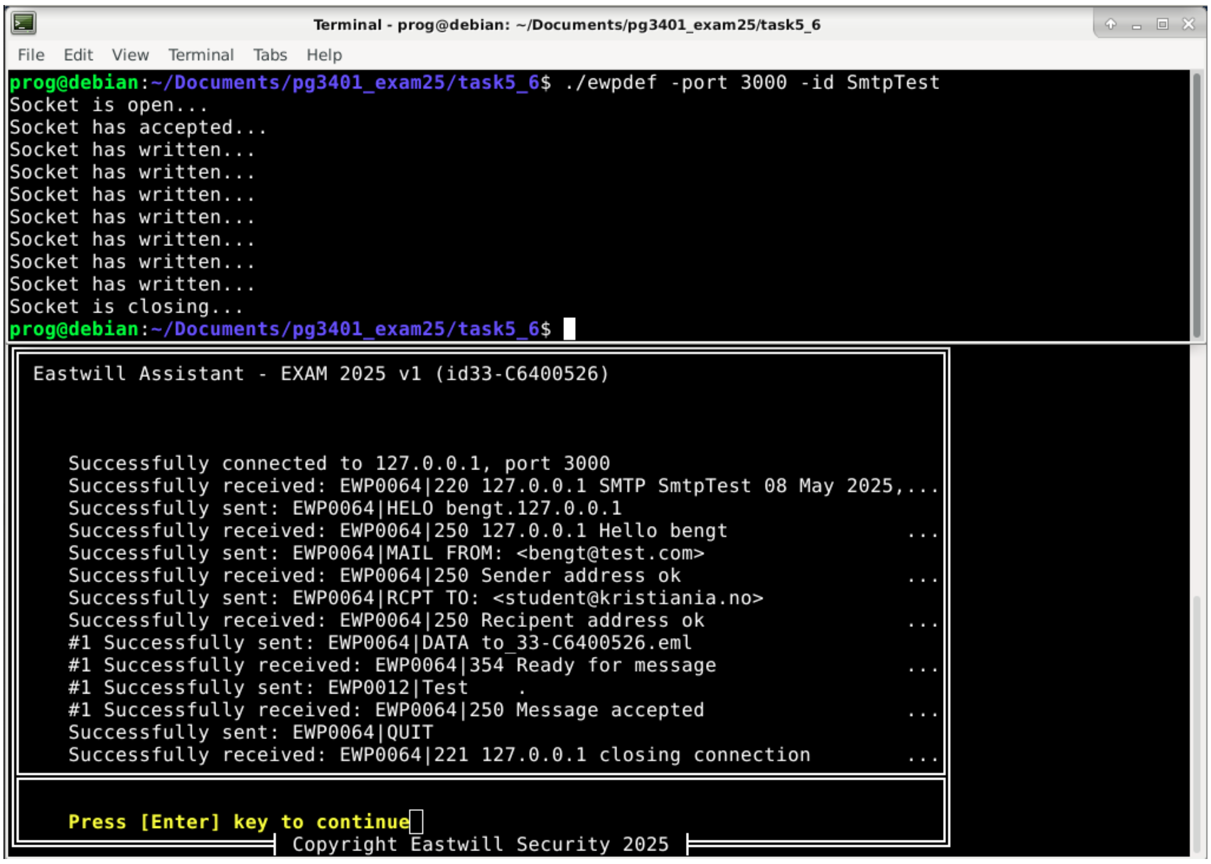
```
Eastwill Assistant - EXAM 2025 v1 (id33-C6401626)


   You have asked me to check your result, let us have a look...
   Successfully opened 'pg3401_exam25/task4_6/task4_threads.c' - check
   Changes detected in file (hash 5cd2850e)
   Successfully read the source file - check
   Successfully found a makefile in the task4 folder - check
   Successfully found parameters to main - check
   Successfully found semaphore variable - check
   Successfully found explisit init - check
   Successfully found 30 comments - check

   Part I approved, press [Enter] to get Part II



   Press [Enter] key to continue
                    Copyright Eastwill Security 2025
```

Task 5. Network

As requested, I have created a Server that expects a Client through a TCP Socket connection. Once the client does, the server receives and sends data as described in the task, using the supplied structures in the header file. I took heavy use of memcpy, as it allowed me to format strings easily, and omit the null-terminator, as it would cause the sent data to be terminated early.

The code that was reused multiple places was made into functions, and a separate .c and header file were made for this. This helps with the cleanliness and maintainability of the code.

task5_scrnshot.png


Terminal - prog@debian: ~/Documents/pg3401_exam25/task5_6
File Edit View Terminal Tabs Help

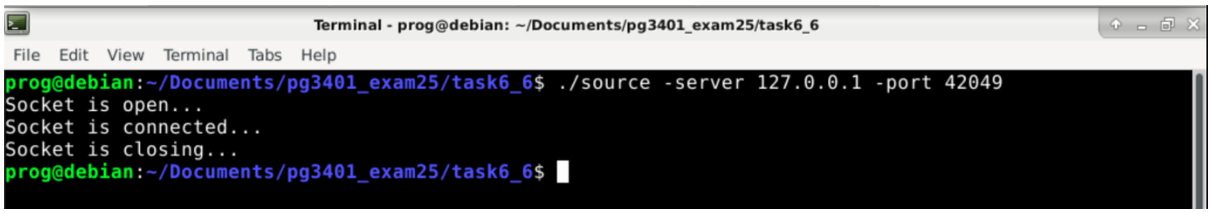
```
prog@debian:~/Documents/pg3401_exam25/task5_6$ ./ewpdef -port 3000 -id SmtpTest
Socket is open...
Socket has accepted...
Socket has written...
Socket has written...
Socket has written...
Socket has written...
Socket has written...
Socket has written...
Socket has written...
Socket is closing...
prog@debian:~/Documents/pg3401_exam25/task5_6$
```

```
 Eastwill Assistant - EXAM 2025 v1 (id33-C6400526)



    Successfully connected to 127.0.0.1, port 3000
    Successfully received: EWP0064|220 127.0.0.1 SMTP SmtpTest 08 May 2025,...
    Successfully sent: EWP0064|HELO bengt.127.0.0.1
    Successfully received: EWP0064|250 127.0.0.1 Hello bengt                 ...
    Successfully sent: EWP0064|MAIL FROM: <bengt@test.com>
    Successfully received: EWP0064|250 Sender address ok                     ...
    Successfully sent: EWP0064|RCPT TO: <student@kristiania.no>
    Successfully received: EWP0064|250 Recipent address ok                   ...
    #1 Successfully sent: EWP0064|DATA to_33-C6400526.eml
    #1 Successfully received: EWP0064|354 Ready for message                  ...
    #1 Successfully sent: EWP0012|Test     .
    #1 Successfully received: EWP0064|250 Message accepted                   ...
    Successfully sent: EWP0064|QUIT
    Successfully received: EWP0064|221 127.0.0.1 closing connection          ...


    Press [Enter] key to continue
                        Copyright Eastwill Security 2025
```

Task 6. Problem solving

I have made a Client that connects to a Server through a given port from EWA. I created a file so that I could save the data EWA was gonna send me, and I successfully received all the data over the TCP protocol. I now have the file in binary format, encrypted with the TEA encryption method. However, I did not get further than that, as I struggled to understand encryption with TEA, as well as decryption. I attempted to read the documentation and search the web, but was unsuccessful in solving the full task.

task6_scrnshot.png


Terminal - prog@debian: ~/Documents/pg3401_exam25/task6_6
File Edit View Terminal Tabs Help

```
prog@debian:~/Documents/pg3401_exam25/task6_6$ ./source -server 127.0.0.1 -port 42049
Socket is open...
Socket is connected...
Socket is closing...
prog@debian:~/Documents/pg3401_exam25/task6_6$
```

Bibliography

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language*. Prentice Hall. From PDF: https://colorcomputerarchive.com/repo/Documents/Books/The%20C%20Programming%20Language%20%28Kernighan%20Ritchie%29.pdf

Østby, B. (2025, January 8). *Canvas*. Accessed: 30.04.2025. From PG3401_Lecture_01_Intro: https://kristiania.instructure.com/courses/12804/files/1549106?module_item_id=531960

National Inventors Hall of Fame. (2019). *Dennis Ritchie - UNIX Operating System*. Accessed: 30.04.2025. From National Inventors Hall of Fame: https://www.invent.org/inductees/dennis-ritchie

EBSCO. (2024). *Dennis Ritchie*. Accessed: 6.05.2025. From EBSCO: https://www.ebsco.com/research-starters/biography/dennis-ritchie

Geeks for Geeks. (Last edited: 2025, March 20). *8 Most Popular Linux Distributions (2025)*. Accessed: 30.04.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/8-most-popular-linux-distributions/#5-arch-linux

Geeks for Geeks. (Last edited: 2024, April 18). *What is Ubuntu? (2024)*. Accessed: 30.04.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/what-is-ubuntu/

Geeks for Geeks (Last edited: 2024, February 1). *Introduction to Debian Linux (2024)*. Accessed: 30.04.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/introduction-to-debian-linux/

Geeks for Geeks (Last edited: 2025, January 30). *What is Arch Linux? (2025)*. Accessed: 30.04.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/what-is-arch-linux/

Geeks for Geeks (Last edited: 2024, March 27). *How to Write a Struct to a Binary File in C?*. Accessed: 30.04.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/how-to-write-struct-to-a-binary-file-in-c/?ref=ml_lbp

Geeks for Geeks (Last edited: 2024, June 19). *C Program to Implement Singly Linked List*. Accessed: 01.05.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/c-program-to-implement-singly-linked-list/

Geeks for Geeks (Last edited: 2024, October 25). *Doubly Linked List in C*. Accessed: 01.05.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/doubly-linked-list-in-c/

Educative (Last edited: n.d). *How to use the typedef struct in C*. Accessed: 03.05.2025. From Educative: https://www.educative.io/answers/how-to-use-the-typedef-struct-in-c

Lawrence Livermore National Laboratory (Last edited: n.d). *Passing Arguments to Threads*. Accessed: 03.05.2025. From Lawrence Livermore National Laboratory: https://hpc-tutorials.llnl.gov/posix/passing_args/

Geeks for Geeks (Last edited: 2024, May 24). *Mutex lock for Linux Thread Synchronization*. Accessed: 03.05.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/mutex-lock-for-linux-thread-synchronization/

The open group (Last edited: n.d). *pthread_mutex_init*. Accessed: 03.05.2025. From The open group: https://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread_mutex_init.html

Geeks for Geeks (Last edited: 2025, January 10). *How to use POSIX semaphores in C language*. Accessed: 03.05.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/use-posix-semaphores-c/

Østby, B. (2025, April 3). *Canvas*. Accessed: 05.05.2025. From PG3401_Lecture_10_Network: https://kristiania.instructure.com/courses/12804/files/1597520?module_item_id=547621

Østby, B. (2025, April 3). *Canvas*. Accessed: 05.05.2025. From PG3401_Lecture_09-12_exam_preparation: https://kristiania.instructure.com/courses/12804/files/1589250?module_item_id=543875

Geeks for Geeks (Last edited: 2025, April 23). *Socket Programming in C*. Accessed: 05.05.2025. From Geeks for Geeks: https://www.geeksforgeeks.org/socket-programming-cc/

Rosenfield, A. (Last edited: 2019, December 30). *How can i get the date and time values in a C program*. Accessed: 06.05.2025. From Stack Overflow: https://stackoverflow.com/questions/1442116/how-can-i-get-the-date-and-time-values-in-a-c-program

user207421 (Last edited: 2024, July 30). *How recv() function works when looping?*. Accessed: 07.05.2025. From Stack Overflow:

https://stackoverflow.com/questions/27205810/how-recv-function-works-when-looping