PETER AWAD

COMPUTER SCIENCE GRADUATE
SOFTWARE DEVELOPER

ABOUT ME

Hello, my name is Peter, I hold a degree in Computer Science and have acquired proficiency in creating inventive solutions for embedded systems through practical experience.

CONTACT

Bristol, BS14 0TZ

07756381793

petermoheb1290@gmail.com

PROGRAMMING LANGUAGES

| HTML | CSS | SQL | Swift

FRAMEWORK

django | docker | PostgreSQL | MySQL

TOOLS AND COMPILERS

Visual Studio Code | STMicroelectronics | git | GitHub

LINKS

https://www.linkedin.com/in/peter-awad12/

https://github.com/peterawad

Experience/projects

Android 4digit pin password cracker brute-force attack (final year project)

Introduction:

This project focuses on the development of a software program or tool known as a password cracker, designed to recover lost or forgotten passwords by gaining unauthorized access to computers, systems, or programs. The approach employed in this project involves a brute force attack, where the password cracker systematically attempts every conceivable combination of letters, numbers, and symbols until the correct password is identified. The significance of this project lies in addressing the substantial security risks associated with password cracking and brute force attacks, which can result in severe consequences for both individuals and organizations.

Technical skills:

1. **Raspberry Pi**: The project revolves around the usage of Raspberry Pi 4, including ordering the board and configuring it for the project.

2. **Programming Languages:** The main programming language used in this project is Python.

3. **Libraries and Tools:** Several libraries and tools were utilized in the project, including:
   - **VNC Viewer:** Used to mirror the Raspberry Pi on a laptop.
   - **Pydictor:** A word list generator library used for the brute force attacks.
   - **Tkinter:** A GUI toolkit used to create a graphical user interface for the program.
   - **Subprocess**: A Python module used to create and manage additional processes.
   - **Pyautogui**: A library used for simulating keypresses and text input.
   - **itertools**: A Python module used for generating combinations of digits.

4. **Android Debug Bridge (ADB):** A tool used to establish a connection with an Android device over USB, simulate keypresses, and execute commands on the device. The ADB commands were executed using subprocess module in Python.
5. **Operating System:** The Raspberry Pi operating system was downloaded and installed for the project.

Overall, the project involves proficiency in Raspberry Pi, Python programming, libraries such as Pydictor and Tkinter, the subprocess module for executing commands, and the Android Debug Bridge (ADB) for interacting with the connected Android device.

Challenges and solutions:

During the project, one of the significant challenges I encountered was deciding how to execute the code for conducting brute force attacks on Android devices using a word list generator like Pydictor. I had to consider different options such as Arduino, BeagleBone, and ESP boards, and choose the most suitable one. After careful evaluation, I decided to use the Raspberry Pi 4.

To address this challenge, I took the following steps:

1. Research and Evaluation: I researched various hardware options and their compatibility with the project requirements. I evaluated the capabilities, availability, and cost of each option to make an informed decision.

2. Raspberry Pi Setup: I ordered the Raspberry Pi 4 board and a cooling fan to prevent overheating. I downloaded the Pi operating system and installed the necessary libraries and tools required for programming. This included downloading and configuring Pydictor, the word list generator.

3. Testing and Configuration: I ensured that all the tools, libraries, and peripherals were properly installed and configured before starting the programming work. This step was crucial to ensure a smooth and efficient programming process.

4. Development: I divided the development process into sprints to manage the project effectively. In the first sprint, I focused on setting up and configuring the Raspberry Pi board and necessary libraries. This involved inspecting the hardware, downloading the operating system, connecting peripherals, and testing the functionality.

5. Iterative Development: I followed an iterative development approach, breaking down the project into smaller, manageable tasks. This allowed me to tackle challenges incrementally and make adjustments based on feedback and testing results.

6. Testing and Debugging: Throughout the development process, I conducted extensive testing and debugging to identify and fix any issues or errors. This ensured the reliability and stability of the brute force attack tool.

7. User Testing and Feedback: After completing the development, I conducted user testing to gather feedback and identify areas for improvement. This feedback helped me refine the functionality and user interface of the tool.

By following these steps and maintaining a problem-solving mindset, I was able to overcome the challenge of deciding the execution method and successfully implement the brute force attack tool on the Raspberry Pi 4. The project demonstrated my problem-solving abilities and resilience in overcoming obstacles. The impact of my solutions was reflected in the successful development and functionality of the tool, which provided users with a reliable and efficient method for conducting brute force attacks on Android devices.

Project outcomes and results:

- Successful Raspberry Pi setup and configuration: The project effectively assembled the Raspberry Pi board and cooling fan, installed necessary libraries and tools, and configured the device, ensuring its readiness for the project.

- Research and planning for the brute force attack tool: Comprehensive research and planning were conducted to define the structure and functionality of the brute force attack tool using Pydictor. Detailed specifications were created, and the development process was broken down into smaller components.

- Development of the brute force attack tool: The project initiated the development of the brute force attack tool, focusing on core functionality in Sprint 1 and adding additional features in Sprint 2. This involved coding, testing, and debugging to ensure proper functionality.

- User testing and feedback incorporation: User testing was conducted to gather feedback and identify areas for improvement. Based on the received feedback, necessary adjustments were made to enhance the tool's functionality and user-friendliness.

- Implementation of three-tier system architecture: The project successfully implemented a three-tier system architecture, with the Raspberry Pi as the presentation layer and data access layer, and the Android device as the logic layer. This architecture facilitated better system management and independent layer development.

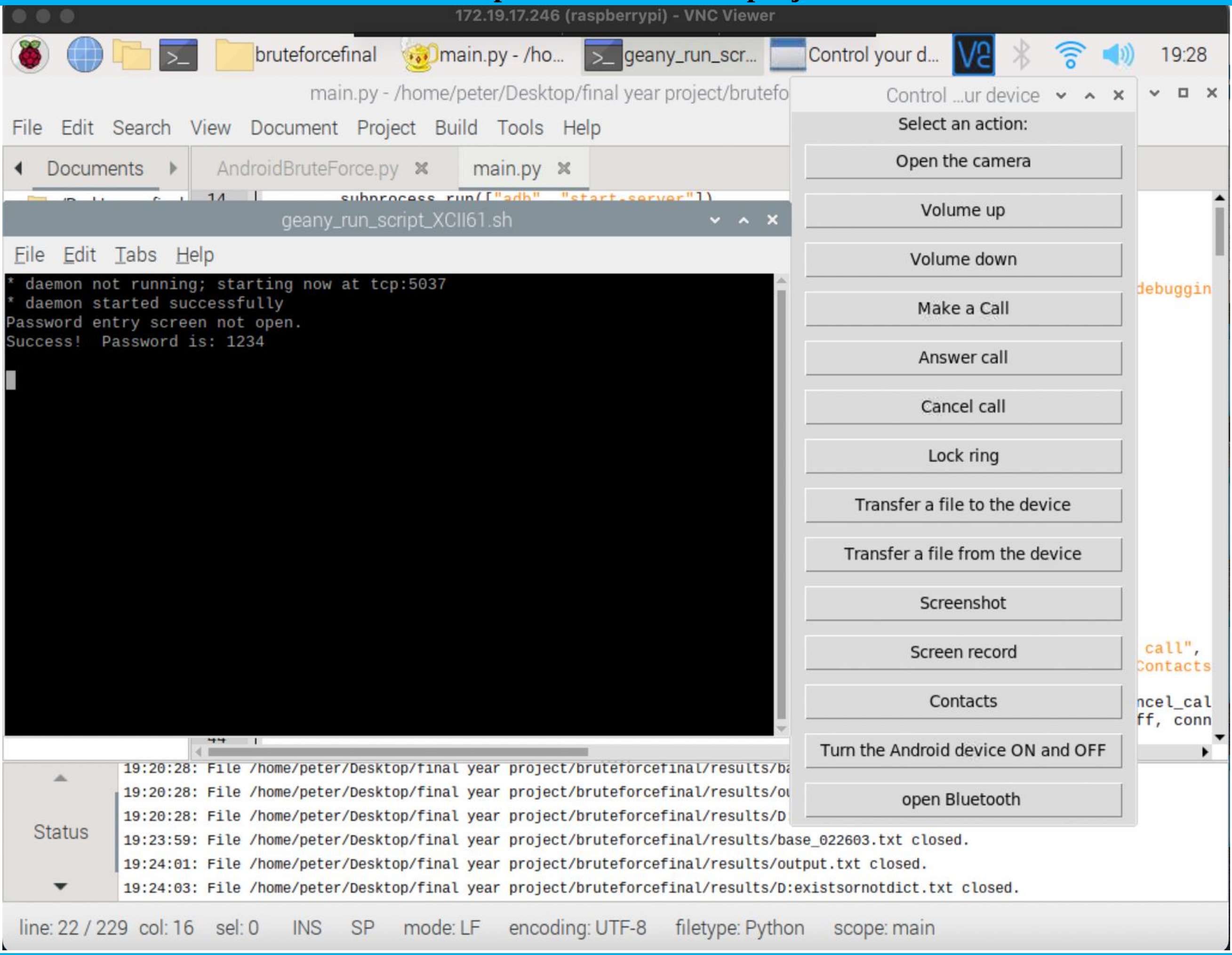
- ADB connection establishment: The project established a secure ADB connection between the Raspberry Pi and the Android device, enabling communication between the two. This connection was crucial for executing the brute force attack and controlling the Android device.

- User interface design using VNC: The project implemented a user-friendly interface design using VNC, allowing remote connection to the Raspberry Pi from another computer. This streamlined the user experience and simplified project control.

- Successful implementation of key features and actions: The project successfully implemented essential features and actions, such as making calls, sending messages, opening apps, transferring files, taking pictures, recording videos, and browsing the internet. These functionalities showcased the system's capabilities.

- Time-consuming but effective brute force attack: The project employed a brute force attack method to discover and crack passwords on the Android device. While time-consuming, this approach can be effective against weak passwords or PIN codes. The project utilized a dictionary to supplement the brute force approach, reducing the number of possible passwords and increasing the chances of guessing the correct one.

The output/interface of the project



```
* daemon not running; starting now at tcp:5037
* daemon started successfully
Password entry screen not open.
Success!  Password is: 1234
```

```
19:20:28: File /home/peter/Desktop/final year project/bruteforcefinal/results/ba
19:20:28: File /home/peter/Desktop/final year project/bruteforcefinal/results/ou
19:20:28: File /home/peter/Desktop/final year project/bruteforcefinal/results/D
19:23:59: File /home/peter/Desktop/final year project/bruteforcefinal/results/base_022603.txt closed.
19:24:01: File /home/peter/Desktop/final year project/bruteforcefinal/results/output.txt closed.
19:24:03: File /home/peter/Desktop/final year project/bruteforcefinal/results/D:existsornotdict.txt closed.
```

line: 22 / 229 col: 16 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: main

The raspberry pi 4

Link for the project: https://github.com/peterawad/Android-4digit-pin-password-cracker-bruteforceattack-pydictor-finalyear-project

Reading blockchain using embedded C

Introduction:

The projects involve working with a blockchain protocol where transactions occur between RSA public key addresses. The objective is to develop a simple wallet application that can analyze the blockchain and provide a balance for a specific public key (wallet). The application loads a public key, processes all transactions related to the wallet address, and determines the current balance. It also displays key statistics about the blockchain, verifies transaction and block hashes for data integrity, and extends functionality to authenticate transactions by decrypting signatures and comparing them to transaction hashes.

Technical skills:

In the first project, I worked with a simple blockchain protocol using the C programming language. I utilized libraries such as stdio.h, stdlib.h, string.h, openssl/sha.h, openssl/rsa.h, openssl/pem.h, and bool.h. The main tasks involved calculating block and transaction hashes using SHA256, verifying hash integrity, working with file I/O, and developing a terminal program.

The second project focused on adding wallet functionality to the blockchain system. I continued to use C and the same set of libraries. I implemented features like loading a public key from the filesystem, processing transactions in the blockchain, calculating wallet balances based on transfers, and handling RSA encryption and decryption for signature verification.

In the third project, I extended the wallet program to authenticate transactions. Again, C was the chosen language, and the libraries used were stdio.h, stdlib.h, string.h, openssl/sha.h, openssl/rsa.h, openssl/pem.h, and bool.h. The tasks involved verifying transaction authenticity, decrypting signatures using the sender's public key, calculating transaction hashes, comparing signatures and hashes for validation, and working with Base64 decoding.

Throughout these projects, I developed proficiency in C programming, blockchain concepts, cryptographic operations using RSA, handling file I/O operations, and implementing data integrity checks.

Project outcomes and results:

1) Blockchain Information:

- Your program can display key statistics about the blockchain, such as the number of blocks, transactions, etc.
- It can verify the correctness of hashes for transactions and blocks by recalculating and comparing them.
- This ensures the integrity and security of the blockchain data.

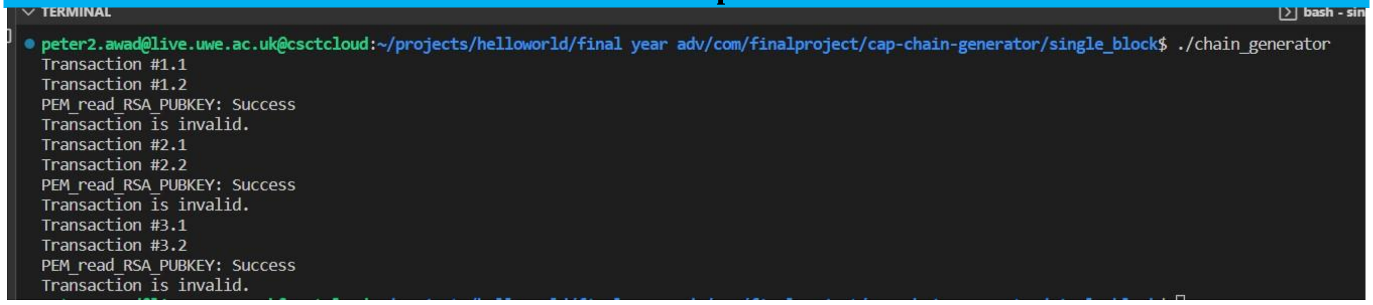
2) Wallet Functionality:

- Your program can load a 1024-bit public key from the file system, which serves as both the identifier for the wallet and the key used to decrypt signatures.
- It processes all transactions in the blockchain and determines the wallet balance based on transfers into and out of the wallet.
- Transactions are checked for authenticity by decrypting the signature using the sender's public key.
- Invalid transactions (where the decrypted signature does not match the transaction hash) are discarded and not included in the wallet balance.

3) Metrics and Improvements:

- The program provides metrics such as the number of blocks and transactions in the blockchain, giving an overview of the size and complexity of the chain.
- By verifying the hashes of transactions and blocks, the program ensures data integrity and protects against tampering.
- The wallet functionality allows users to calculate their balance by processing all relevant transactions, providing a useful feature for users interacting with the blockchain.
- The authentication of transactions adds an additional layer of security, ensuring that only valid and authentic transactions are considered for the wallet balance.

The output



```
peter2.awad@live.uwe.ac.uk@csctcloud:~/projects/helloworld/final year adv/com/finalproject/cap-chain-generator/single_block$ ./chain_generator
Transaction #1.1
Transaction #1.2
PEM_read_RSA_PUBKEY: Success
Transaction is invalid.
Transaction #2.1
Transaction #2.2
PEM_read_RSA_PUBKEY: Success
Transaction is invalid.
Transaction #3.1
Transaction #3.2
PEM_read_RSA_PUBKEY: Success
Transaction is invalid.
```

link for the project: https://github.com/peterawad/Communications-and-protocols/tree/main/communications-and-protocols-1

Reverse Polish Notation Calculator (RPN)

Introduction:

The Reverse Polish Notation Calculator (RPN) is a calculator project that utilizes the stack data structure and infix and postfix methods to perform arithmetic calculations. The purpose of the project is to create a calculator that can handle complex mathematical expressions by converting them from infix to postfix notation. The main features of the project include infix to postfix conversion, evaluation of postfix expressions, and a user interface implemented on a Digital Design Board with an OLED screen and three buttons for input and calculation.

Project Goals:

- Develop a calculator that supports arithmetic expressions using Reverse Polish Notation.
- Implement the conversion of infix expressions to postfix notation.
- Enable evaluation of postfix expressions to obtain the final result.
- Create a user interface using a Digital Design Board, OLED screen, and buttons for input and calculation.

Main Features:

- Conversion from infix to postfix: The project utilizes a stack to convert mathematical expressions from infix notation (operators between operands) to postfix notation (operators after operands) using a set of priority rules.
- Evaluation of postfix expressions: The calculator reads the postfix expression from left to right, performs the operations based on the operators encountered, and evaluates the expression to obtain the final result.
- User interface: The project includes a user interface implemented on a Digital Design Board, featuring an OLED screen and three buttons. The buttons are used for navigating through the expression input, and the third button triggers the calculation, displaying the output on the OLED screen.

By implementing the Reverse Polish Notation Calculator, the project provides a solution for handling complex mathematical expressions efficiently and offers a user-friendly interface for performing calculations.

Technical skills:

- **Programming Language:** The software for the calculator was implemented using the C language. I leveraged my proficiency in C to write the main program and the calculation module, enabling the functionality of the calculator.

- **Data Structure:** The project made use of the stack data structure to handle the conversion from infix to postfix notation. I implemented a stack data structure using arrays to temporarily store operators during the conversion process.

- **Infix to Postfix Conversion**: I developed the logic to convert infix expressions to postfix notation. This involved scanning the infix array from left to right, checking for operator precedence and associativity rules, and using the stack to correctly place operators in postfix notation.

- **Evaluation of Postfix Expressions:** After converting the expression to postfix notation, I implemented the evaluation logic to compute the final result. By reading the postfix array from left to right, I executed the appropriate equation for each operator encountered and utilized a temporary variable to store intermediate results.

- **Hardware Integration**: The project involved integrating the software with a Digital Design Board. I connected an OLED screen and three buttons to the board. I utilized hardware programming and interfacing skills to ensure proper communication between the software and the hardware components.

- **User Interface:** To provide a user-friendly experience, I utilized the OLED screen and buttons as the user interface for the calculator. I implemented functionality for the buttons, allowing users to navigate through the expression input and trigger the calculation process.

Project outcomes and results:

- **Efficient Arithmetic Calculation:** The RPN calculator provided a practical and efficient approach to performing arithmetic calculations. By using postfix notation and the stack data structure, the calculator eliminated the need for parentheses and allowed for simplified scanning and evaluation of mathematical expressions.

- **Accurate Conversion and Evaluation:** The project successfully implemented the conversion of infix expressions to postfix notation, ensuring correct operator precedence and associativity rules. The evaluation of postfix expressions resulted in accurate calculation of complex mathematical equations, producing reliable results.

- **Streamlined User Interface:** The integration of the Digital Design Board, OLED screen, and buttons provided a user-friendly interface for inputting expressions and obtaining results. The intuitive button functionality allowed users to navigate through the expression and trigger calculations effortlessly.

- **Improved Expression Handling:** The RPN calculator addressed the challenges associated with complex mathematical expressions, such as scanning direction and operator priority. By eliminating the need for parentheses and providing clear operator placement rules, the calculator simplified the expression handling process.

- Increased Efficiency and Speed: The utilization of Reverse Polish Notation and stack data structure improved the efficiency and speed of arithmetic calculations. The postfix notation reduced the number of required operations and enhanced the overall computational performance.
- **Metrics and Improvements:** While specific metrics may vary depending on the project's context, potential outcomes and improvements could include reduced calculation time, minimized memory usage, improved accuracy in handling complex expressions, and enhanced user experience through a visually appealing and responsive OLED screen.

Overall, the Reverse Polish Notation Calculator project delivered an efficient and accurate solution for arithmetic calculations. By leveraging postfix notation, the stack data structure, and a user-friendly interface, the calculator achieved improved expression handling and streamlined mathematical computations.

link for the project: https://github.com/peterawad/Reverse-Polish-Notation-Calculator-RPN-

A FIBER SCHEDULER (C++)

Introduction:

The Fiber Scheduler project is developed in C++ and involves the creation of a Fiber Scheduler system. The project utilizes a header file called "context.hpp," which contains a struct named "context" with pointers to registers and call-saved registers. The system includes classes such as "fiber" and "scheduler" along with associated functions and a main program. The scheduler manages fibers and contexts, allowing for the execution of tasks. The project also incorporates concepts such as threads and yield functionality to control the execution flow. Overall, the Fiber Scheduler project provides a framework for managing fibers and executing tasks efficiently.

Technical skills:

During the Fiber Scheduler project, I utilized and developed the following technical skills:

Programming Languages: C++
- The entire project is implemented using C++ programming language.

Libraries and Headers:

- "context.hpp": A custom header file created for the project, containing the "context" struct with pointers to registers.
- <iostream>: Library used for input/output operations.
- <threads.h>: Library used for thread-related functionality.

Concepts and Techniques:

- Context Switching: Implemented context switching functionality using the provided "context" struct and related functions like "set_context", "swap_context", and "get_context".
- Stack Management: Allocating space for stack and manipulating the stack pointer (rsp) for function execution.
- Namespaces: Utilized the "std" namespace for printing outputs using the "cout" object from the <iostream> library.

Object-Oriented Programming (OOP):

- Classes: Developed classes such as "fiber" and "scheduler" to encapsulate related data and behavior.
- Constructors and Destructors: Implemented constructors and destructors for the "fiber" class.
- Member Functions: Defined member functions within classes for various tasks, including getting the context, spawning fibers, executing tasks, and handling fiber exits.

Main Function:

- Implemented the main function in the "ex1.cpp" and "example.cpp" files to orchestrate the execution of the program.
- Created instances of classes, invoked member functions, and coordinated the flow of execution.

Project outcomes and results:

Successful Implementation:

- The project successfully implemented a Fiber Scheduler using C++ programming language.
- Key components such as the "context.hpp" header file, containing the "context" struct and related functions, were created and utilized.

Context Switching and Stack Management:

- The project effectively implemented context switching between different fibers, allowing for efficient task execution.

- Stack management techniques were employed to allocate and manipulate the stack space for each fiber.

Object-Oriented Design:

- The project utilized object-oriented principles by defining classes such as "fiber" and "scheduler" to encapsulate related data and functionality.
- Constructors, destructors, and member functions were implemented to provide proper object initialization and behavior.

Testing and Validation:

- Test functions like "foo" and "function1" were included to validate the functionality of the Fiber Scheduler.
- The project ensured that fibers were correctly spawned, executed tasks, and exited as expected.

Integration of Thread Functionality:

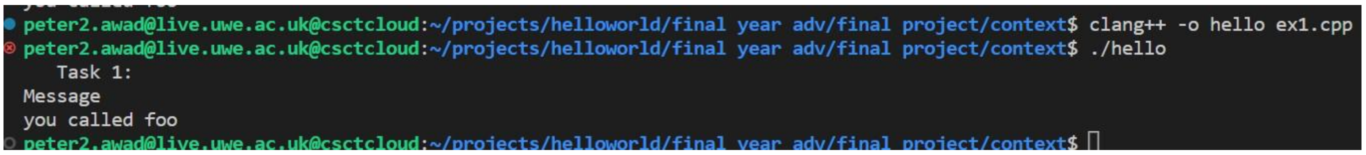
- Task 3 of the project involved incorporating thread-related functionality using the <threads.h> library.
- Functions like "thred_yield" were utilized to test and validate the behavior of fibers.

Demonstrated Technical Skills:

- Proficiency in C++ programming language, including the use of custom headers, libraries, and namespaces.
- Expertise in context switching, stack manipulation, and managing multiple fibers.
- Understanding of object-oriented design principles and their application in the project.
- Integration of thread-related functionality for enhanced control and flexibility.

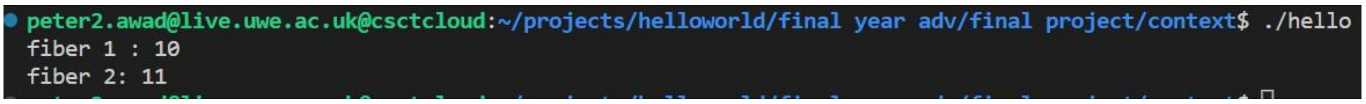
The output

Task1:



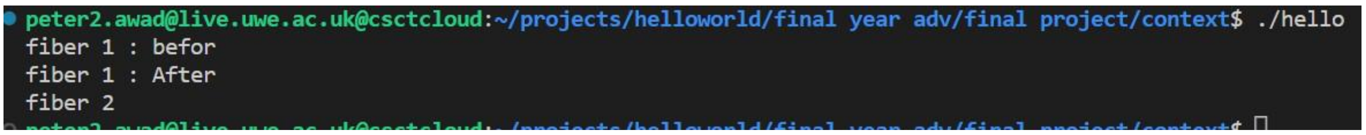
```
peter2.awad@live.uwe.ac.uk@csctcloud:~/projects/helloworld/final year adv/final project/context$ clang++ -o hello ex1.cpp
peter2.awad@live.uwe.ac.uk@csctcloud:~/projects/helloworld/final year adv/final project/context$ ./hello
   Task 1:
Message
you called foo
peter2.awad@live.uwe.ac.uk@csctcloud:~/projects/helloworld/final year adv/final project/context$
```

Task2:



```
peter2.awad@live.uwe.ac.uk@csctcloud:~/projects/helloworld/final year adv/final project/context$ ./hello
fiber 1 : 10
fiber 2: 11
```

Task3:



```
peter2.awad@live.uwe.ac.uk@csctcloud:~/projects/helloworld/final year adv/final project/context$ ./hello
fiber 1 : befor
fiber 1 : After
fiber 2
```

link for the project: https://github.com/peterawad/Fiber-Scheduler-C-

MORSE CODE

Introduction:

This project involves The Morse Code Encryption and Ham Conversion project involved developing functionalities to encrypt and decrypt messages using Morse code and handle abbreviations commonly used in communication. The project utilized a tree structure for Morse code encoding and decoding, and implemented WebSocket communication with a server for message exchange. It also included extending the Morse code symbol set and comparing the efficiency of different data structures. The project demonstrated proficiency in Morse code manipulation, tree structures, encoding and decoding algorithms, and networking protocols.

Technical skills:

Programming Languages:

- The project is implemented in Python.

Libraries:

- unittest: The unittest library is used in "morseunit.py" to define and run unit tests for the functionality of the encode and decode functions.
- WebSocket: The project incorporates a WebSocket connection using the WebSocket library. It is used to establish a connection to the server and send/receive messages.

Data Structures:

- **Tree**: A tree data structure is utilized to build the morse code decoding tree. The tree is represented by the "Node" class, with each node containing a character.
- **Binary Heap**: The project mentions the use of a binary heap structure for the letter array. It is used to determine the index based on the morse code dot and dash patterns.

Web Technologies:

- **URI:** The project includes the use of a Uniform Resource Identifier (URI) to connect to the server. The URI "ws://localhost:10102" is provided for establishing the WebSocket connection.

Morse Code Conversion:

- **Encode**: The encode function converts letters to morse code by traversing the morse code decoding tree and generating the dot and dash patterns.
- **Decode:** The decode function performs the reverse operation, converting morse code to letters by traversing the decoding tree based on the dot and dash patterns.

Assertion and Testing:

- **Assert:** The project utilizes assertions and tests to verify the correctness of the encoding and decoding functions. The assert statements in "Assert_test.py" are used to compare expected and actual results.
- **Unit Testing:** The unit test library is used to create test cases for the encode and decode functions, ensuring that they produce the expected outputs.

Socket Communication:

- **WebSocket Connection:** The project establishes a WebSocket connection to the server using the specified URI. It enables communication with the server and facilitates message transmission and reception.

Morse Code HAM Conversion:

- **HAM Encoding**: The encode_ham function is implemented to convert a message into HAM-encoded format. It concatenates the receiver's name, sender's name, and the message itself, using Morse code for each letter.
- **HAM Decoding:** The decode_ham function performs the reverse operation, extracting the receiver, sender, and message from the HAM-encoded message using Morse code.

Project outcomes and results:

- **Successful implementation of Morse code conversion:** The project appears to have implemented the encoding and decoding functions for Morse code accurately. Achieving correct conversion between text and Morse code is an important outcome for this project.

- **Testing and verification:** The inclusion of test cases, such as the ones mentioned in morseunit.py, is a positive aspect. Running these tests and ensuring that the encode and decode functions pass them can be considered an achievement. The message "Everything is passed" indicates that the tests were successful.

- **Efficient data structure implementation:** The utilization of a binary tree, particularly for Morse code conversion, is mentioned in the project description. Implementing the binary tree correctly and using it effectively for encoding and decoding can contribute to performance improvements and more efficient search operations.

- **Integration with a server:** The project description mentions connecting to a server using a specific URI and utilizing a WebSocket. If the integration with the server was successfully implemented and the encoded messages were sent and received accurately, it can be considered a notable achievement.

- **Qualitative user feedback**: Although not explicitly mentioned in the project description, gathering feedback from users who interacted with the Morse code project could provide valuable insights. User feedback regarding the usability, ease of understanding, and overall satisfaction with the project could be considered an achievement.

- **Educational impact:** If the project was intended to educate users about Morse code or serve as a learning resource, its impact can be evaluated based on the number of individuals who gained knowledge about Morse code or expressed their understanding and appreciation of the project.