

|  |
| --- |
| **5004CEM - Operating Systems and Security - 2223JANMAY**  Student id 11782139 |
|  |
| April 6 |

# Index

|  |
| --- |
|  |
| [*Task 1*](#_task_1:_essay)  [*Task 2*](#_task_2)  [*Commented code*](#_COMMNEDT_code:)  [*Output*](#_Output:)  [*Explanation*](#_Explanation:)  [*Task 4*](#_task_4)  [*Commented code*](#_command_code:)  [*Output*](#_Output:_1)  [*Explanation*](#_Explanation) |
|  |

# task 1: essay

OS Kernal

Operating system(OS) developers are exploring two uses of two types of Operating system kernels. There are two main types of kernels micro and macro kernels. This essay will be describing the main features of both kernels compare them and as well as diss how the combination of these would improve Operating system performance.

# **Micro-kernel**

Microkernel is a minimalist approach to the design of the operating system, it only provides basic functions, such as process management, memory management, and inter-process communication and other services such as device driver, file system etc are implemented at the user level process, running in their own address spaces. Tanenbaum and Bos (2014)

This provides several benefits to this approach. First of all, the flexibility of the microkernel as the service is implemented at the user level can be added or removed depending on the stability. Secondly, the security of the microkernel as services run in their own address space which means they cannot access the kernel data structure directly the resulting risk of vulnerability in the security of the kernel is reduced. As well as it is more reliable as the kernel only provides basic functionality as the result it is less complex and therefore less likely to have bugs and crashes.

### **Macro-kernel**

Macro-kernel is a traditional approach to the design of the operating system. In this approach, the kernel provides a wide range of functions such as file systems, networking, etc with the basic functionality of a macro-kernel. According to Silberschatz et al. (2018)

The benefits are, Firstly, the microkernel is efficient as services are implemented in the kernel level, which can be optimized for performance, reducing overhead and improving overall system performance. Secondly, the simplicity of this kernel is that all services are provided by the kernel, reducing complexity and making the Operating system easier to understand and debug.

According to Liedtke (1996), a hybrid kernel provides the benefits of a micro-kernel and macro-kernel. This provides basic functionality, such as process management, and memory management, while more complex services, such as device drivers, and file systems, are implemented in user-level processes, running in their own address spaces. This approach provides several benefits, including performance, flexibility, security, and reliability.

### **Comparison**

There are several important differences between microkernel and macro kernel as below.

|  |  |
| --- | --- |
| Microkernel | Monolithic kernel |
| User and kernel services are stored in distinct address regions in the microkernel. | Both user and kernel services are retained in the same address space with a monolithic kernel. |
| OS design is difficult. | OS is simple to create and implement. |
| Microkernels are much smaller. | A monolithic kernel is significantly bigger than a microkernel. |
| It is simpler to add new features. | It is difficult to add new features. |
| More code is necessary to construct a microkernel. | When compared to the microkernel, there is less code. |
| The failure of one component does not affect the operation of the microkernel. | When one component of a monolithic kernel fails, the entire system fails. |
| The rate of execution is slow. | The execution rate is fast. |
| Microkernel is simple to expand. | The monolithic kernel is difficult to expand. |
| Communication microkernels employ messaging queues to implement IPC. | In monolithic kernels, IPC is implemented through signals and sockets. |
| Debugging is straightforward. | Debugging is a challenging task. |
| It is simple to keep. | Maintenance necessitates more time and resources. |
| The microkernel requires message routing and context switching. | When the kernel is running, message forwarding and context switching are not necessary. |
| The kernel solely provides IPC and low-level device management. | The Kernel includes all of the services provided by the operating system. |

According to Heiser (2008), the micro-kernel is secure and flexible, but less efficient compared macro-kernel and The macro-kernel approach is efficient and simpler, but less flexible and secure than the micro-kernel.

### **Combining Kernel Approaches**

One way to improve the performance of an operating system(OS) is to combine the micro-kernel and macro-kernel. This is known as a hybrid kernel and provides the best of both kernels. This provides basic functionality, such as process management, memory management, and inter-process communication, while more complex services, such as device drivers, file systems, and networking, are implemented in user-level processes, running in their own address spaces.

This approach provides several benefits. Firstly, the performance of the hybrid kernels is because complex services are implemented in user-level processes, which can be optimized for performance, improving operating system performance. Secondly, the flexibility, services are implemented as user-level processes, which can be easily added or removed without affecting the kernel's stability. Thirdly, the security, services run in their own address spaces, which cannot access kernel data structures directly, reducing the risk of security risk. Finally, a hybrid kernel is reliable since the kernel provides only basic functionality, which is less complex and therefore less likely have to bugs and crashes.

As stated by Liedtke (1996), the hybrid kernel blends elements of both micro and macro kernels. It allows for fundamental features like managing processes and memory, but concurrently grants more intricate services - such as device drivers or file systems- to be executed in individual user-level procedures within their own address spaces. The advantages generated from this method ring true across improved functionality, adaptability, safety measures and dependability."

## **References**

1. Tanenbaum, A. S., & Bos, H. (2014). Modern operating systems. Pearson Education Limited.
2. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating system concepts. Wiley.
3. Heiser, G. (2008). The role of micro-kernels in modern operating systems. ACM SIGOPS Operating Systems Review, 42(1), 34-38.
4. Liedtke, J. (1996). On µ-kernel construction. Proceedings of the 15th ACM Symposium on Operating Systems Principles, 237-250.

# task 2

# COMMNEDT code:

section  .datans10

quiz\_Q1 dq "What is the smallest country in the world by land area? ", 10 ; Ask the user for a quess

quiz\_Q1Len equ $- quiz\_Q1 ; calculate the length of the guess message in bytes

quiz\_Q2 dq "Who is the current Prime Minister of Canada? ", 10;

quiz\_Q2Len equ $- quiz\_Q2 ;

quiz\_Q3 dq "Which planet in our solar system is the hottest? ", 10;

quiz\_Q3Len equ $- quiz\_Q3 ;

quiz\_Q4 dq "What is the capital city of South Korea? ", 10;

quiz\_Q4Len equ $- quiz\_Q4 ;

quiz\_Q5 dq "What is the highest mountain in Africa?", 10;

quiz\_Q5Len equ $- quiz\_Q5 ;

quiz\_Q6 dq "Which American state is known as the unshine State?", 10;

quiz\_Q6Len equ $- quiz\_Q6 ;

quiz\_Q7 dq "Who wrote the novel To Kill a Mockingbird ?", 10;

quiz\_Q7Len equ $- quiz\_Q7 ;

quiz\_Q8 dq "What is the chemical symbol for gold?", 10;

quiz\_Q8Len equ $- quiz\_Q8 ;

quiz\_Q9 dq "Which country is home to the world's largest rainforest?", 10;

quiz\_Q9Len equ $- quiz\_Q9 ;

quiz\_Q10 dq "What is the name of the famous painting by Leonardo da Vinci that depicts a woman with a mysterious smile? ", 10;

quiz\_Q10Len equ $- quiz\_Q10 ;

correct\_aswer db "you are correct ", 10 ; Message that the guess and the right answer are same

correct\_aswerLen equ $- correct\_aswer ; calculate the length of the message

ans1 dq "vatican city" ; variable that stores the correct answer

ans2 dq "justin trudeau"

ans3 dq "venus"

ans4 dq "seoul"

ans5 dq "mount kilimanjaro"

ans6 dq "florida"

ans7 dq "harper lee"

ans8 dq "au"

ans9 dq "brazil"

ans10 dq "mona lisa"

point\_message db "The score is :" ; Message to present the inc value

point\_messageLen equ $-point\_message ; Getting the length of the inc message

Userscore1 db 0

Userscore2 db 0

newline db 10;

notcorrect\_answer db "Your wrong", 10 ; Message that the guess and the right answer are same

notcorrect\_answerLen equ $- notcorrect\_answer ; calculate the length of the message

notcorrect\_answer2 db "Your incorrect ", 10 ; message user guess is incorrect

samenotlen equ $-notcorrect\_answer ;Calculate the length of the message

section .bss

   user1\_quiz1 resq 1 ; variable containing letter entered by the user 8 bytes

   clear resq 1

   user1\_quiz2 resq 1

   user1\_quiz3 resq 1

   user1\_quiz4 resq 1

   user1\_quiz5 resq 1

   user1\_quiz6 resq 1

   user1\_quiz7 resq 1

   user1\_quiz8 resq 1

   user1\_quiz9 resq 1

   user1\_quiz10 resq 1

   user2\_quiz1 resq 1

   user2\_quiz2 resq 1

   user2\_quiz3 resq 1

   user2\_quiz4 resq 1

   user2\_quiz5 resq 1

   user2\_quiz6 resq 1

   user2\_quiz7 resq 1

   user2\_quiz8 resq 1

   user2\_quiz9 resq 1

   user2\_quiz10 resq 1

   printScore resb 1;

   printScore1 resb 1 ;

section  .text

global \_start      ;must be declared for using gcc

\_start:   ;tell linker entry point

  call printquiz\_Q1

  call compareValue1user1\_quiz1

  call compareValue1User2Q1

  call printUserscore1

  call printUserscore2

  call printquiz\_Q2

  call compareValue1user1\_quiz2

  call compareValue1user2\_quiz2

  call printUserscore1

  call printUserscore2

  call printquiz\_Q3

  call compareValue1user1\_quiz3

  call compareValue1user2\_quiz3

  call printUserscore1

  call printUserscore2

  call printquiz\_Q4

  call compareValue1user1\_quiz4

  call compareValue1user2\_quiz4

  call printUserscore1

  call printUserscore2

  call printquiz\_Q5

  call comparevalue1user1\_quiz5

  call comparevalue1user2\_quiz5

  call printUserscore1

  call printUserscore2

  call printquiz\_Q6

  call comparevalue1user1\_quiz6

  call comparevalue1user2\_quiz6

  call printUserscore1

  call printUserscore2

  call printquiz\_Q7

  call comparevalue1user1\_quiz7

  call comparevalue1user2\_quiz7

  call printUserscore1

  call printUserscore2

  call printquiz\_Q8

  call comparevalue1user1\_quiz8

  call comparevalue1user2\_quiz8

  call printUserscore1

  call printUserscore2

  call printquiz\_Q9

  call comparevalue1user1\_quiz9

  call comparevalue1user2\_quiz9

  call printUserscore1

  call printUserscore2

  call printquiz\_Q10

  call comparevalue1user1\_quiz10

  call comparevalue1user2\_quiz10

  call printUserscore1

  call printUserscore2

  call end ; call subroutine to end the program

; subroutine to get user input and compare with correct answer

compareValue1user1\_quiz1:

  mov eax, 3 ; read from keyboard using system call SYS\_READ

  mov ebx, 2 ; stdin

  mov ecx, user1\_quiz1 ; Move the character entered by the user into variable guess

  mov edx, 1 ;  As single letter input use 1 byte

  int 80h   ; Invoke the kernel to get the user input

  mov   rax, [user1\_quiz1] ; move the value in variable guess into register eax

  cmp   rax, [ans1]  ; compare correct answer with what in eax (user guess)

  je    user1 ; if value in eax and variable answer are the same jump to same

  call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

compareValue1User2Q1:

  mov eax, 3 ; read from keyboard using system call SYS\_READ

  mov ebx, 2 ; stdin

  mov ecx,  user2\_quiz1 ; Move the character entered by the user into variable guess

  mov edx, 1 ;  As single letter input use 1 byte

  int 80h   ; Invoke the kernel to get the user input

  mov   rax, [ user2\_quiz1] ; move the value in variable guess into register eax

  cmp   rax, [ans1]  ; compare correct answer with what in eax (user guess)

  je    user2 ; if value in eax and variable answer are the same jump to same

  call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

compareValue1user1\_quiz2:

  mov eax, 3 ; read from keyboard using system call SYS\_READ

  mov ebx, 2 ; stdin

  mov ecx, user1\_quiz2 ; Move the character entered by the user into variable guess

  mov edx, 1 ;  As single letter input use 1 byte

  int 80h   ; Invoke the kernel to get the user input

  xor rax, rax

  mov   rax, [user1\_quiz2] ; move the value in variable guess into register eax

  cmp   rax, [ans2]  ; compare correct answer with what in eax (user guess)

  je    user1 ; if value in eax and variable answer are the same jump to same

  call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

compareValue1user2\_quiz2:

  mov eax, 3 ; read from keyboard using system call SYS\_READ

  mov ebx, 2 ; stdin

  mov ecx, user2\_quiz2 ; Move the character entered by the user into variable guess

  mov edx, 1 ;  As single letter input use 1 byte

  int 80h   ; Invoke the kernel to get the user input

  xor rax, rax

  mov   rax, [user2\_quiz2] ; move the value in variable guess into register eax

  cmp   rax, [ans2]  ; compare correct answer with what in eax (user guess)

  je    user2 ; if value in eax and variable answer are the same jump to same

  call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

compareValue1user1\_quiz3:

  mov eax, 3 ; read from keyboard using system call SYS\_READ

  mov ebx, 2 ; stdin

  mov ecx, user1\_quiz3 ; Move the character entered by the user into variable guess

  mov edx, 1 ;  As single letter input use 1 byte

  int 80h   ; Invoke the kernel to get the user input

  xor rax, rax

  mov   rax, [user1\_quiz3] ; move the value in variable guess into register eax

  cmp   rax, [ans3]  ; compare correct answer with what in eax (user guess)

  je    user1 ; if value in eax and variable answer are the same jump to same

  call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

compareValue1user2\_quiz3:

  mov eax, 3 ; read from keyboard using system call SYS\_READ

  mov ebx, 2 ; stdin

  mov ecx, user2\_quiz3 ; Move the character entered by the user into variable guess

  mov edx, 1 ;  As single letter input use 1 byte

  int 80h   ; Invoke the kernel to get the user input

  xor rax, rax

  mov   rax, [user2\_quiz3] ; move the value in variable guess into register eax

  cmp   rax, [ans3]  ; compare correct answer with what in eax (user guess)

  je    user2 ; if value in eax and variable answer are the same jump to same

  call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

compareValue1user1\_quiz4:

  mov eax, 3 ; read from keyboard using system call SYS\_READ

  mov ebx, 2 ; stdin

  mov ecx, user1\_quiz4 ; Move the character entered by the user into variable guess

  mov edx, 1 ;  As single letter input use 1 byte

  int 80h   ; Invoke the kernel to get the user input

  xor rax, rax

  mov   rax, [user1\_quiz4] ; move the value in variable guess into register eax

  cmp   rax, [ans4]  ; compare correct answer with what in eax (user guess)

  je    user1 ; if value in eax and variable answer are the same jump to same

  call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

compareValue1user2\_quiz4:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user2\_quiz4 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user2\_quiz4] ; move the value in variable guess into register eax

    cmp   rax, [ans4]  ; compare correct answer with what in eax (user guess)

    je    user2; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user1\_quiz5:

  mov eax, 3 ; read from keyboard using system call SYS\_READ

  mov ebx, 2 ; stdin

  mov ecx, user1\_quiz5 ; Move the character entered by the user into variable guess

  mov edx, 1 ;  As single letter input use 1 byte

  int 80h   ; Invoke the kernel to get the user input

  xor rax, rax

  mov   rax, [user1\_quiz5] ; move the value in variable guess into register eax

  cmp   rax, [ans5]  ; compare correct answer with what in eax (user guess)

  je    user1 ; if value in eax and variable answer are the same jump to same

  call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user2\_quiz5:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user2\_quiz5 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user2\_quiz5] ; move the value in variable guess into register eax

    cmp   rax, [ans5]  ; compare correct answer with what in eax (user guess)

    je    user2; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user1\_quiz6:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user1\_quiz6 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user1\_quiz6] ; move the value in variable guess into register eax

    cmp   rax, [ans6]  ; compare correct answer with what in eax (user guess)

    je    user1 ; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user2\_quiz6:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user2\_quiz6 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user2\_quiz6] ; move the value in variable guess into register eax

    cmp   rax, [ans6]  ; compare correct answer with what in eax (user guess)

    je    user2; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user1\_quiz7:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user1\_quiz7 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user1\_quiz7] ; move the value in variable guess into register eax

    cmp   rax, [ans7]  ; compare correct answer with what in eax (user guess)

    je    user1; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user2\_quiz7:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user2\_quiz7 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user2\_quiz7] ; move the value in variable guess into register eax

    cmp   rax, [ans7]  ; compare correct answer with what in eax (user guess)

    je    user2; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user1\_quiz8:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user1\_quiz8 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user1\_quiz8] ; move the value in variable guess into register eax

    cmp   rax, [ans8]  ; compare correct answer with what in eax (user guess)

    je    user1; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user2\_quiz8:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user2\_quiz8 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user2\_quiz8] ; move the value in variable guess into register eax

    cmp   rax, [ans8]  ; compare correct answer with what in eax (user guess)

    je    user2; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user1\_quiz9:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user1\_quiz9 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user1\_quiz9] ; move the value in variable guess into register eax

    cmp   rax, [ans9]  ; compare correct answer with what in eax (user guess)

    je    user1; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user2\_quiz9:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user2\_quiz9 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user2\_quiz9] ; move the value in variable guess into register eax

    cmp   rax, [ans9]  ; compare correct answer with what in eax (user guess)

    je    user2; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user1\_quiz10:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user1\_quiz10 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user1\_quiz10] ; move the value in variable guess into register eax

    cmp   rax, [ans10]  ; compare correct answer with what in eax (user guess)

    je    user1; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

comparevalue1user2\_quiz10:

    mov eax, 3 ; read from keyboard using system call SYS\_READ

    mov ebx, 2 ; stdin

    mov ecx, user2\_quiz10 ; Move the character entered by the user into variable guess

    mov edx, 1 ;  As single letter input use 1 byte

    int 80h ; Invoke the kernel to get the user input

    xor rax, rax

    mov   rax, [user2\_quiz10] ; move the value in variable guess into register eax

    cmp   rax, [ans10]  ; compare correct answer with what in eax (user guess)

    je    user2; if value in eax and variable answer are the same jump to same

    call samenot ; go to samenot subroutine if answer and guess are not same

ret ; return to the main code

printquiz\_Q1:

   mov   edx, quiz\_Q1Len ; length of the guess request message

   mov   ecx,quiz\_Q1 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

printquiz\_Q2:

   mov   edx, quiz\_Q2Len ; length of the guess request message

   mov   ecx,quiz\_Q2 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

printquiz\_Q3:

   mov   edx, quiz\_Q3Len ; length of the guess request message

   mov   ecx,quiz\_Q3 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

; subroutine to deal with when user input and and real answer are the same

printquiz\_Q4:

   mov   edx, quiz\_Q4Len ; length of the guess request message

   mov   ecx,quiz\_Q4 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

; subroutine to deal with when user input and and real answer are the same

printquiz\_Q5:

   mov   edx, quiz\_Q5Len ; length of the guess request message

   mov   ecx,quiz\_Q5 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

; subroutine to deal with when user input and and real answer are the same

printquiz\_Q6:

   mov   edx, quiz\_Q6Len ; length of the guess request message

   mov   ecx,quiz\_Q6 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

; subroutine to deal with when user input and and real answer are the same

printquiz\_Q7:

   mov   edx, quiz\_Q7Len ; length of the guess request message

   mov   ecx,quiz\_Q7 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

; subroutine to deal with when user input and and real answer are the same

printquiz\_Q8:

   mov   edx, quiz\_Q8Len ; length of the guess request message

   mov   ecx,quiz\_Q8 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

; subroutine to deal with when user input and and real answer are the same

printquiz\_Q9:

   mov   edx, quiz\_Q9Len ; length of the guess request message

   mov   ecx,quiz\_Q9 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

; subroutine to deal with when user input and and real answer are the same

printquiz\_Q10:

   mov   edx, quiz\_Q8Len ; length of the guess request message

   mov   ecx,quiz\_Q10 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret ; return to main code

; subroutine to deal with when user input and and real answer are the same

user1:

mov eax, 3 ; read from keyboard using system call SYS\_READ

   mov ebx, 2 ; stdin

   mov ecx, clear ; Move the character entered by the user into variable guess

   mov edx, 1 ;  As single letter input use 1 byte

   int 80h  ; Invoke the kernel to get the user input

   mov   edx, correct\_aswerLen ; length of the same message

   mov   ecx,correct\_aswer ; same message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   inc byte [Userscore1]

   ret ; return to main code

user2:

mov eax, 3 ; read from keyboard using system call SYS\_READ

   mov ebx, 2 ; stdin

   mov ecx, clear ; Move the character entered by the user into variable guess

   mov edx, 1 ;  As single letter input use 1 byte

   int 80h  ; Invoke the kernel to get the user input

   mov   edx, correct\_aswerLen ; length of the same message

   mov   ecx,correct\_aswer ; same message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

 inc byte [Userscore2]

   ret ; return to main code

; subroutine to deal with when user input and and real answer are not the same

samenot:

   mov   edx, notcorrect\_answerLen ; not same message length

   mov   ecx,notcorrect\_answer ; not same message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the kernel to print the message

   mov eax, 3 ; read from keyboard using system call SYS\_READ

   mov ebx, 2 ; stdin

   mov ecx, clear ; Move the character entered by the user into variable guess

   mov edx, 1 ;  As single letter input use 1 byte

   int 80h  ; Invoke the kernel to get the user input

   ret ; return to main code

printUserscore1:

   mov eax, [Userscore1]

   add eax, 48

   mov [printScore], eax ;

   mov   edx, point\_messageLen ; length of the guess request message

   mov   ecx, point\_message ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   mov   edx, 1 ; length of the guess request message

   mov   ecx, printScore ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   mov   edx, 1 ; length of the guess request message

   mov   ecx, newline ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret

printUserscore2:

   mov eax, [Userscore2]

   add eax, 48

   mov [printScore1], eax ;

   mov   edx, point\_messageLen ; length of the guess request message

   mov   ecx, point\_message ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   mov   edx, 1 ; length of the guess request message

   mov   ecx, printScore1 ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   mov   edx, 1 ; length of the guess request message

   mov   ecx, newline ; guess request message

   mov   ebx,1 ;file descriptor (stdout)

   mov   eax,4 ;system call number (sys\_write)

   int   0x80  ;invoke the  kernel to display the message

   ret

;subroutine to end the program

end:

   mov eax, 1 ; system call number (sys\_exit)

   int 80h ; invoke the kernel to end the program

# Output:

Text

Description automatically generated

# Explanation:

The first part of the code where I define and store question and answers in variables and then the length of each guess question, as well as the message for a correct and wrong answer, has to function known as user and user where each user data will be stored and 10 quiz function that for each player so each player score can be calculated

# task 4

## command code:

server.py

# Import necessary modules

import sqlite3

import socket

import sys

import time

# Define host and port

host = 'localhost'

port = 9900

# Define data payload size and maximum number of queued connections

data\_payload = 2048

backlog = 5

# Create a TCP socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Enable reuse address/port

sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

# Bind the socket to the port

server\_address = ('localhost', port)

print("Starting up echo server on %s port %s" % server\_address)

sock.bind(server\_address)

# Listen to clients, backlog argument specifies the max no. of queued connections

sock.listen(backlog)

# Accept a connection

client, address = sock.accept()

# Receive data from the client and split it into a list of strings

data = client.recv(data\_payload).decode("utf-8").split(',')

# Extract the name of the client from the received data

name = data[4]

# Remove the name from the list of received data

delnme=data.pop(4)

# Convert the remaining items in the list to integers

list\_int = [int(i) for i in data ]

#Determine the chosen area based on user input

if list\_int[0] == 1:

    chosen\_area = "PeakDistrict"

elif list\_int[0] == 2:

    chosen\_area = "Lincolnshire"

elif list\_int[0] == 3:

    chosen\_area = "York"

elif list\_int[0] == 4:

    chosen\_area = "NorthWales"

elif list\_int[0] == 5:

    chosen\_area = "Warwickshire"

elif list\_int[0] == 6:

    chosen\_area = "Cheshire"

# Determine the second  chosen area based on user input

if list\_int[4] == 1:

    chosen\_area1 = "PeakDistrict"

elif list\_int[4] == 2:

    chosen\_area1 = "Lincolnshire"

elif list\_int[4] == 3:

    chosen\_area1 = "York"

elif list\_int[4] == 4:

    chosen\_area1 = "NorthWales"

elif list\_int[4] == 5:

    chosen\_area1 = "Warwickshire"

elif list\_int[4] == 6:

    chosen\_area1 = "Cheshire"

# Determine the chosen minimum area based on user input

if list\_int[1] >= 0:

    chosen\_minimum\_area = list\_int[1]

# Determine thesedn  chosen minimum area based on user input

if list\_int[5] >= 0:

    chosen\_minimum\_area1 = list\_int[5]

# Determine the chosen maximum area based on user input

if list\_int[2] >= 0:

    chosen\_maximum\_area = list\_int[2]

# Determine the second  chosen maximum area based on user input

if list\_int[6] >= 0:

    chosen\_maximum\_area1 = list\_int[6]

# Determine the chosen difficulty based on user input

if list\_int[3] == 1:

    chosen\_difficulty = "Easy"

if list\_int[3] == 2:

    chosen\_difficulty = "Medium"

if list\_int[3] == 3:

    chosen\_difficulty = "Hard"

# Determine the chosen difficulty based on user input

if list\_int[7] == 1:

    chosen\_difficulty1 = "Easy"

if list\_int[7] == 2:

    chosen\_difficulty1= "Medium"

if list\_int[7] == 3:

    chosen\_difficulty1 = "Hard"

# Call database\_query function to retrieve the relevant data from the database

orginal\_card=list\_int[8]

orginal\_expiry=list\_int[9]

orginal\_security=list\_int[10]

amount=list\_int[11]

# Connect to the SQLite database

conection = sqlite3.Connection("D:/New folder/database.db")

cursor = conection.cursor()

# Query the database for the walk names, book names, and page numbers that match the user's query parameters

cursor.execute("SELECT walk\_name, book, page FROM book\_detail WHERE area = ? AND distance BETWEEN ? AND ? AND dificulty = ?",(chosen\_area, chosen\_minimum\_area, chosen\_maximum\_area, chosen\_difficulty))

result1 = cursor.fetchall()

# Execute Query 2

cursor.execute("SELECT walk\_name, book, page FROM book\_detail WHERE area = ? AND distance BETWEEN ? AND ? AND dificulty = ?",(chosen\_area1, chosen\_minimum\_area1, chosen\_maximum\_area1, chosen\_difficulty1))

result2 = cursor.fetchall()

walkname\_book\_page = set(result1 +result2)

# Query the database for the book names that match the user's query parameters

cursor.execute("SELECT  book FROM book\_detail WHERE area = ? AND distance BETWEEN ? AND ? AND dificulty = ?",(chosen\_area, chosen\_minimum\_area, chosen\_maximum\_area, chosen\_difficulty))

booksearch1=cursor.fetchall()

cursor.execute("SELECT  book FROM book\_detail WHERE area = ? AND distance BETWEEN ? AND ? AND dificulty = ?",(chosen\_area1, chosen\_minimum\_area1, chosen\_maximum\_area1, chosen\_difficulty1))

booksearch2 = cursor.fetchall()

booksearch =set(booksearch1 +booksearch2)

ans=[]

resultingstring=[]

cost\_statement=[]

timesodres=0

for i in walkname\_book\_page:

    walkname = i[0]

    book = i[1]

    page = i[2]

    # Construct the answer string for each row and append it to ans

    cursor.execute("SELECT price FROM book\_price WHERE book =?", (book,))

    real=str(cursor.fetchone())

    answer = "walk name:" + str(walkname) + "--------book:" + str(book) + " ----------page:" + str(page)+" ----------price £"+real[1:6]

    ans.append(answer)

for book\_search in booksearch:

    quantity\_books = next(iter(book\_search))

    print(quantity\_books)

    # Query the database for the quantity of the selected book

    cursor.execute("SELECT quantity FROM book\_order WHERE book=?", (quantity\_books,))

    quantity = cursor.fetchone()

    quantity = quantity[0]

    # If the book is in stock

    if quantity is not None:

        # If the requested amount is greater than the stock quantity

        if amount > quantity:

            # If there is no stock available

            if quantity == 0:

                print(0.5)

                result = f"{quantity\_books} is out of stock item come back tomorrow we will have it"

                cursor.execute("UPDATE book\_order SET  quantity = quantity + ? WHERE book=?", (amount, quantity\_books))

                resultingstring.append(result)

                conection.commit()

            # If there is some stock available

            if quantity != 0 and amount > quantity:

                print(1)

                left = amount - quantity

                result = f"{quantity\_books} is out of stock item come back tomorrow we will have it"

                cursor.execute("UPDATE book\_order SET  quantity = quantity + ? WHERE book=?", (left, quantity\_books))

                resultingstring.append(result)

                conection.commit()

        # If the requested amount is less than or equal to the stock quantity

        if amount <= quantity:

            print(2)

            # Point c: the user's payment is successful

            results = f"there are  {quantity} {quantity\_books} "

            cost\_statement = []

            for item in walkname\_book\_page:

                print(3)

                walkname = item[0]

                book = item[1]

                page = item[2]

                cursor.execute("SELECT price FROM book\_price WHERE book=?", (book,))

                prices = cursor.fetchone()

                if prices is not None:

                    print(4)

                    cursor.execute("SELECT totalorder FROM scamcheck WHERE name=?", (name,))

                    orders = cursor.fetchone()

                    orders = list(orders)

                    # If the user has previously ordered more than 100 books

                    if orders[0] > 100:

                        total\_price = float(amount) \* prices[0]

                        cursor.execute("UPDATE book\_order SET quantity = quantity - ? WHERE book=?", (amount, quantity\_books))

                        book\_price = f"payment successful walk name:{walkname}book name :{book}  page:{page} quantity:{amount} prices:£{total\_price}"

                        timesodres += amount

                        cost\_statement.append(book\_price)

                        conection.commit()

                        break

                    else:

                        # If the user has previously ordered less than or equal to 100 books

                        total\_price = float(amount) \* prices[0]

                        cursor.execute("UPDATE book\_order SET quantity = quantity - ? WHERE book=?", (amount, quantity\_books))

                        book\_price = f"payment successful walk name:{walkname}book name :{book}  page:{page} quantity:{amount} prices:£{total\_price}"

                        timesodres += amount

                        cost\_statement.append(book\_price)

                        conection.commit()

            resultingstring.append(results)

    # If the book is not in stock

    else:

        print(6)

        # Point e: the book cannot be found in the inventory

        result = f"no match  for {quantity\_books}"

        resultingstring.append(result)

# Update the user's order history

cursor.execute("UPDATE scamcheck SET totalorder = totalorder + ? ,name =? WHERE name =?", (timesodres, name, name))

conection .commit()

# Construct the response string

response = "{}\n{}\n{}\n{}\n\n".format(name, "\n".join(ans), "\n".join(resultingstring), "\n".join(cost\_statement)).encode("utf-8")

print("Sending response to client:", response)

time.sleep(120)

client.send(response)

cursor.close()

conection.close()

client.py

#!/usr/bin/env python

# Source Python Network Programming Cookbook, Second Edition -- Chapter - 1

# Import necessary modules

import socket

import sys

""" A simple echo client """

# Define constants

host = 'localhost'

port = 9900

# Create a TCP/IP socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Connect the socket to the server

server\_address = (host, port)

print("--------------------")

print("Connecting to %s port %s" % server\_address)

sock.connect(server\_address)

# Define the echo client function

print ("""

More Peak District-£12.99

Lincolnshire Wolds-£10.99

Vale Of York-£11.99

Peak District-£12.99

Snowdonia-£13.99

Malvern And Warwickshire-£10.99

Cheshire-£12.99""")

# Get search criteria from user

# Ask the user for the area they want to go to

while True:

    area = input("""What area do you want to go to?

    1. Peak District

    2. Lincolnshire

    3. York

    4. North Wales

    5. Warwickshire

    6.Cheshire

    Enter number 1-6: """)

    # Validate the user's input

    if area == "" or area == " ":

        print("try again")

    elif area not in ["1", "2", "3", "4", "5", "6"]:

        print("try again")

    else:

        str(area)

        break

# Ask the user for the minimum and maximum miles

while True:

    min\_mile = input("Minimum miles: ")

    max\_mile = input("Maximum miles: ")

    if min\_mile == "" or min\_mile == " " or max\_mile == " " or max\_mile == "":

        print("try again")

    else:

        str(min\_mile)

        str(max\_mile)

        break

# Ask the user for the level of difficulty

while True:

    difficulty = input("""Enter level of difficulty:

    1. Easy

    2. Medium

    3. Hard

    Enter number 1-3: """)

    # Validate the user's input

    if difficulty == "" or difficulty == " ":

        print("try again")

    if difficulty not in ["1", "2", "3"]:

        print("try again")

    else:

        str(difficulty)

        break

# Ask the user for the area they want to go to

while True:

    area1 = input("""What area1 do you want to go to?

    1. Peak District

    2. Lincolnshire

    3. York

    4. North Wales

    5. Warwickshire

    6.Cheshire

    Enter number 1-6: """)

    # Validate the user's input

    if area1 == "" or area1 == " ":

        print("try again")

    elif area1 not in ["1", "2", "3", "4", "5", "6"]:

        print("try again")

    else:

        str(area1)

        break

# Ask the user for the minimum and maximum miles

while True:

    min\_mile1 = input("Minimum miles: ")

    max\_mile1 = input("Maximum miles: ")

    if min\_mile1 == "" or min\_mile1 == " " or max\_mile1 == " " or max\_mile1 == "":

        print("try again")

    else:

        str(min\_mile1)

        str(max\_mile1)

        break

# Ask the user for the level of difficulty

while True:

    difficulty1 = input("""Enter level of difficulty:

    1. Easy

    2. Medium

    3. Hard

    Enter number 1-3: """)

    # Validate the user's input

    if difficulty1 == "" or difficulty1 == " ":

        print("try again")

    if difficulty1 not in ["1", "2", "3"]:

        print("try again")

    else:

        str(difficulty1)

        break

while True:

    print("we will card inforamtion early on but wont use it  if u dont selct to use it ")

    print("are u buying books")

    buyornot=int(input("1 for yes 2 for no:"))

    if buyornot in [1,2]:

        if buyornot==1:

            name=input("enter name  on card/account:")

            orginal\_card=input("long card number:")

            orginal\_expiry=input("expiery code in tis foramt MMYY eg 0522:")

            orginal\_security=input("cvv:")

            card=len(str(orginal\_card))

            expiry=len(str(orginal\_expiry))

            security=len(str(orginal\_security))

            if card==16 and   expiry==4 and security==3 and orginal\_card.isdigit()   and orginal\_expiry.isdigit() and orginal\_security.isdigit():

                amount=int(input("ammount of books if your  are books are found:"))

                if amount>0:

                    print("procsiing info")

                    break

                else:

                    print("try again")

            else:

                print("try again")

        if buyornot ==2:

            name=input("enter name  on card/account")

            orginal\_card=0

            orginal\_expiry=0

            orginal\_security=0

            break

        else:

            print("try again")

    else:

        print("try again")

# Send the search criteria to the server

#              0      1         2           3                 4          5             6          7          8                9              10                 11

message = f"{area},{min\_mile},{max\_mile},{difficulty},{name},{area1},{min\_mile1},{max\_mile1},{difficulty1},{orginal\_card},{orginal\_expiry},{orginal\_security},{amount}"

sock.sendall(message.encode('utf-8'))

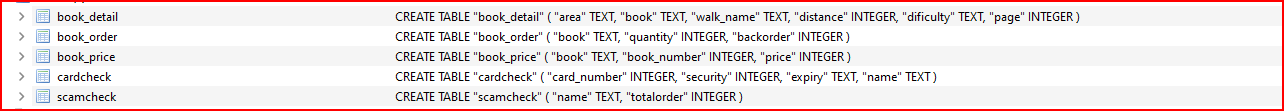
# Receive the response from the server

data = sock.recv(1024)

print(data.decode("utf-8"))

sock.close()

sql database



Graphical user interface, table

Description automatically generated with medium confidence

Table

Description automatically generatedTable

Description automatically generated with medium confidence

Table, calendar

Description automatically generated Table, calendar

Description automatically generated

# Output:

Text

Description automatically generated

# Explanation

When both client and server.py are running first code client.py will prompt the user to put hiking information needed to provide them with recommendations and ask them if they want to buy books if they want it will talk there card information and if not it will consider all card detail null and send details of hiking to server.

When the server receives all the information it covers them into a list and uses the information to recommend user books and allow them to buy them if they want to buy it if the user has over 100 books bought it will delay the process of ordering delaying process but 120s to try to stop users and books are finished or not enough it will take buy put for more books by order and I used a database to sr all information so it can be used when such as followed

"book\_detail" (

"area" TEXT,

"book\_order" (

"book" TEXT,

"quantity" INTEGER,

"backorder" INTEGER

);

"book" TEXT,

"walk\_name" TEXT,

"distance" INTEGER,

"dificulty" TEXT,

"page" INTEGER

);

"cardcheck" (

"card\_number" INTEGER,

"security" INTEGER,

"expiry" TEXT,

"name" TEXT

);

"scamcheck" (

"name" TEXT,

"totalorder" INTEGER

);

"book\_price" (

"book" TEXT,

"book\_number" INTEGER,

"price" INTEGER