**CONTACT MANAGER DOCUMENTATION (GROUP 4, SET 2)**

**INTRODUCTION**

Contact manager is a software or program that enables users to easily store contact information such as names and perform certain functionalities with the stored data.

This project is a reference guide that must be disseminated in our project to continuously improve and run the code. Right now, our strength and efforts were focused on password protection, view contact, speed-dial, add contacts, view call logs and deleting call logs. This documentation will also help people answer their own questions.

This project was based on the file management system although there might be other processes to achieve the same result.

The file management system was chosen because it has functionalities like update, create, delete, file open, file close which was comfortable for our project.

**ReadMe**: Due to frequently running of the program which recreates text files, text files (Phonebook.txt and Clog.txt) have being place manually in the directory of the executable file. Hence user should create these text files before running the program.

**EXECUTION**

Use nasm -f elf64 -o objectname.o objectname.asm

ld -o object object.o

./object

Password: 1234

**NB**: object in the case of our project is **contactmanager.** The code is written in 32bit.

**CONTENT**

Our project code is in three sections. The .data, .bss and .text(code)

**.data section**

This contains static variables or variables that do not change in the program execution. Variables used were RWRITE, msg, phonebook, Call\_log and len

The RWRITE takes a value of 2. The phonebook takes the text file ‘Phonebook.txt’. The Call\_log takes the text file ‘Clog.txt’.

**NB:** 0 is placed at the end of a text file declaration to show the offset value at the beginning of a file

**.bss section**

This section contains declared variables with reserved bytes for future use. In our project we have value, number, password, view\_contaact, option, fd\_out and fd\_in

**NB**: Any time we open a file to write, fd\_out is used in ebx and when we open to read a file, fd\_in is used in ebx.

**.text(Code) section**

This section contains our executable instructions/code. In this section, we used macro, labels, conditional and unconditional jumps.

**Macro**

Macro are sequential instructions assigned with names that are used more than once in a program execution. Macro starts with %macro and ends with %endmacro. Macro take in arguments that varies/changes in the program execution.

The macro used in our program are read, write, file open, file close, file update and file delete

**Read macro**

The read macro takes three arguments. Here in our read macro, the syscall for read and the kernel call is always constant and the others may vary within the code. We start listing them with a counting number. Refer to source code.

**FAQ:** How does the others vary

**ANS: eg.** In reading an input from the user, we use ebx, 2 but in reading from a file we use ebx, [fd\_in]. Hence it may vary in our program execution.

**Write macro**

The write macro takes three arguments. Refer to source code. Similar to the read macro.

**Fileopen macro**

The file open macro takes two arguments. Refer to source code. The 0777 in the ecx register are the base permissions given to the file.

**FAQ:** How does the others vary?

**ANS**: eg. In opening a file for writing/reading, the permission given to the file at that moment may vary within the code. It can be read only, write only or even read and write. Also, the file to be open is placed in the ecx register which may vary within the code.

**Filedelete macro**

The file delete macro take only one argument. The ebx register takes the name of the file to be deleted. Refer to source code

**Fileupdate macro**

The file update macro takes only one argument. The 0 in the ecx register show the offset value of the file to be updated and the 2 in the edx register show that the update should be done at the end of the file. Refer to source code

**NB**: In the edx register, 0 is for the beginning of the file and 1 is for the current position of the file.

**Fileclose macro**

The file close macro takes only one argument. If the file is open to be written to, when closing fd\_out is used and when the file is open to be read, when closing fd\_in is used. Refer to source code

**PROJECT PSEUDOCODE**

1. Display appropriate messages

2. Ask user to enter password

If password mismatched end code

Else display appropriate messages

3. Take user input(option) for functionality

5. Move option into a register for comparison

If comparison is false end code

Else Jump to appropriate labels

6. If label 1 is true (view)

Display appropriate messages

Open file and read content into appropriate variable to be displayed

Close file and display content to user

End code

7. Else if label 2 is true (dial)

Display appropriate messages

Take user input

Open file and write input to appropriate file

Close file and display appropriate messages

End code

8. Else if label 3 is true (create)

Display appropriate messages

Take user input

Open file and write input to appropriate file

Close file and display appropriate messages

End code

9. Else if label 4 is true (call log)

Display appropriate messages

Open file and read content into a variable to be displayed

Close file and display content to user

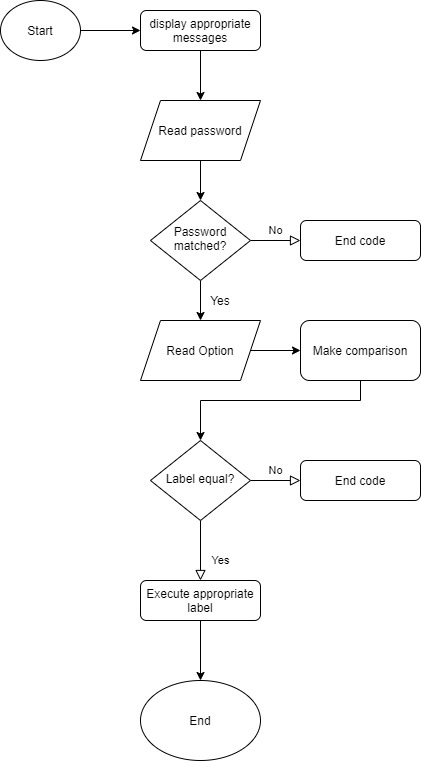
End code

10. Else if label 5 is true (clear call log)

Clear call log by deleting the file

End code

**PROJECT FLOWCHART**



**PROJECT SOURCE CODE**

section .data

RWRITE equ 2

phonebook db 'Phonebook.txt', 0

call\_log db 'Clog.txt', 0

msg1 db 'Contact created...contact saved', 0xa, 0xd

lenmsg1 equ $ - msg1

msg3 db 'Call log Cleared', 0xa, 0xd

lenmsg3 equ $ - msg3

msg7 db ' OPTIONS', 0xa, 0xd

lenmsg7 equ $ - msg7

msg8 db 'Calling...', 0xa, 0xd

lenmsg8 equ $ - msg8

msg10 db 'Enter 1 to view contact',0xa, 0xd

lenmsg10 equ $ - msg10

msg11 db 'Enter 2 to call',0xa, 0xd

lenmsg11 equ $ - msg11

msg12 db 'Enter 3 to Add contact', 0xa, 0xd

lenmsg12 equ $ - msg12

msg13 db 'Enter 4 to View Call log', 0xa, 0xd

lenmsg13 equ $ - msg13

msg14 db 'Enter 5 to Delete Call log', 0xa, 0xd

lenmsg14 equ $ - msg14

msg15 db '------------------------------PHONEBOOK-----------------------', 0xa,0xd

lenmsg15 equ $ - msg15

msg16 db 'Enter the Number/Name', 0xa, 0xd

lenmsg16 equ $ - msg16

msg17 db 'Dailed numbers Missed calls Recieved Calls', 0xa, 0xd

lenmsg17 equ $ - msg17

msg18 db '---------Saved Contacts----------', 0xa, 0xd

lenmsg18 equ $ - msg18

msg19 db 'Enter password', 0xa, 0xd

lenmsg19 equ $ - msg19

msg20 db 'Password matched', 0xa, 0xd

lenmsg20 equ $ - msg20

msg21 db 'Password Mismatched', 0xa, 0xd

lenmsg21 equ $ - msg21

msg22 db 'Contact manager is password protected', 0xa, 0xd

lenmsg22 equ $ - msg22

section .bss

option resb 2

fd\_out resb 1

fd\_in resb 1

value resb 20

number resb 100

view\_contact resw 100

password resb 10

section .text

global \_start

%macro read 3

mov eax, 3 ;syscall for read

mov ebx, %1

mov ecx, %2

mov edx, %3

int 0x80

%endmacro

%macro write 3

mov eax, 4 ;syscall for write

mov ebx, %1

mov ecx, %2

mov edx, %3

int 0x80

%endmacro

%macro fileopen 2

mov eax, 5 ;syscall for file open

mov ebx, %1

mov ecx, %2

mov edx, 0777 ;file permissions

int 0x80

%endmacro

%macro filedelete 1

mov eax, 10 ;syscall for file delete

mov ebx, %1

mov ecx, 0777

int 0x80

%endmacro

%macro fileupdate 1

mov eax, 19 ;syscall for iseek(update)

mov ebx, %1

mov ecx, 0

mov edx, 2 ;referencing the end of the file

int 0x80

%endmacro

%macro fileclose 1

mov eax, 6 ;syscall for file close

mov ebx, %1

int 0x80

%endmacro

\_start:

write 1, msg15, lenmsg15 ;print out the msg title, PHONEBOOK

write 1, msg22, lenmsg22 ;displaying contact is password protected

write 1, msg19, lenmsg19 ;displaying enter password

read 2, password, 10

mov ecx, [password]

cmp ecx, '1234'

jne \_wrongpassword

write 1, msg20, lenmsg20

;displaying phonebook options

write 1, msg7, lenmsg7

write 1, msg10, lenmsg10

write 1, msg11, lenmsg11

write 1, msg12, lenmsg12

write 1, msg13, lenmsg13

write 1, msg14, lenmsg14

read 2, option, 1 ;taking the users option

mov ecx, [option] ;putting the option into the ecx register

;for comparison

cmp ecx, '1'

je \_view

cmp ecx, '2'

je \_dial

cmp ecx, '3'

je \_create

cmp ecx, '4'

je \_clogs

cmp ecx, '5'

je \_delete

\_view:

write 1, msg18, lenmsg18 ;displaying a title msg

fileopen phonebook, RWRITE ;open the file for reading

mov [fd\_in], eax ;in opening a file for reading, fd\_in is used

;read from file

read [fd\_in], view\_contact, 100 ;reading the content of the file

;into the view\_contact variable to be displayed

fileclose [fd\_in] ;closing the file

;print the content of the file to the user

write 1, view\_contact, 100

jmp \_end

\_dial:

write 1, msg16, lenmsg16 ;displaying a msg enter name/number

read 2, number, 100 ;reading input

read 2, number, 100

fileopen call\_log, RWRITE ;opening the call log file

mov [fd\_out], eax ;in writing to file fd\_out is used

fileupdate [fd\_out] ;allowing the file to the updated

write [fd\_out], number, 100 ;writing to the Call log file

fileclose [fd\_out]

write 1, msg8, lenmsg8 ;displaying msg calling

write 1, number, 100 ;appending the name/number to the msg calling

jmp \_end

\_create:

write 1, msg16, lenmsg16 ;displaying msg enter name/number

read 2, number, 100 ;reading input from user

read 2, number, 100

fileopen phonebook, 2 ;phonebook opened with read and write permissions

mov [fd\_out], eax

fileupdate [fd\_out] ;allowing file to be updated

write [fd\_out], number, 100 ;writing the name/number to file

fileclose [fd\_out]

write 1, msg1, lenmsg1 ;msg contact created...contact saved

jmp \_end

\_clogs:

write 1, msg17, lenmsg17 ;msg title

fileopen call\_log, 2

mov [fd\_in], eax

read [fd\_in], view\_contact, 100 ;read from file

fileclose [fd\_in] ;closing the file

write 1, view\_contact, 100

jmp \_end

\_delete:

filedelete call\_log

write 1, msg3, lenmsg3 ;call log cleared

jmp \_end

\_wrongpassword:

write 1, msg21, lenmsg21

\_end:

mov eax, 1

int 0x80

**ISSUES ENCOUNTERED**

1. In the define byte of the text files, 0 was not place at the end of the string initially. Example:

Phonebook db ‘Phonebook.txt’. So anytime we run the code, the inputs were not written to the file. Also, when we view the content in the text files, nothing shows. But when we added 0 to the end of the define byte of the file, viewing and writing to the file works. Example:

Phonebook db ‘Phonebook.txt’ 0.

1. Initially when we where reading inputs from the user to be written to a file, we read it once. Example: read 2, number, 10. So anytime we run the code, the program skips the input and performs the next instruction.

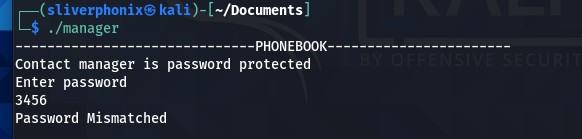
But when we doubled the read, it works normal. Example:

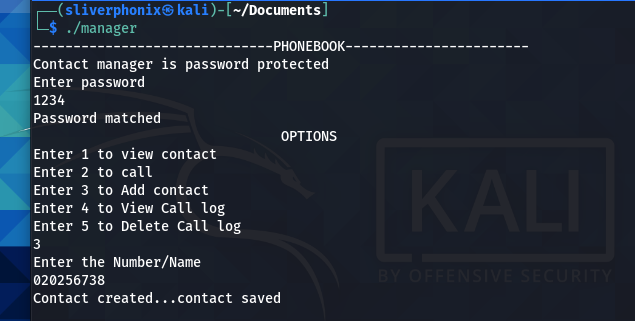
read 2, number, 10

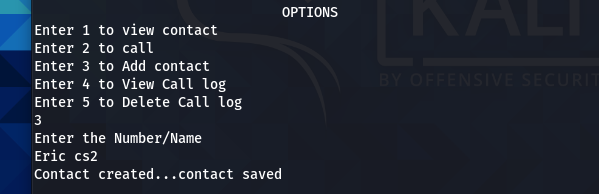
read 2, number, 10

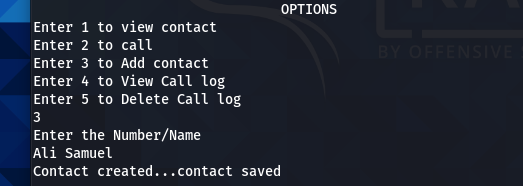
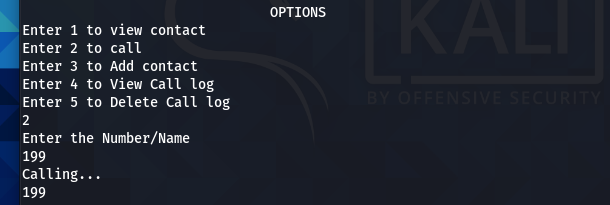
It always happens when we are taking user’s input to be written to file.

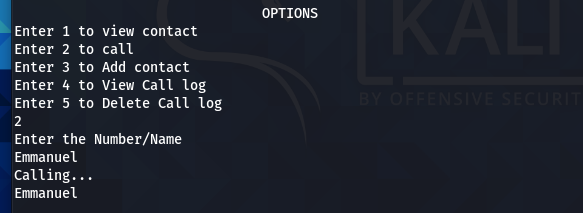
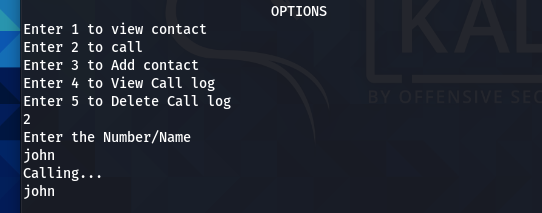
**EXAMPLE**

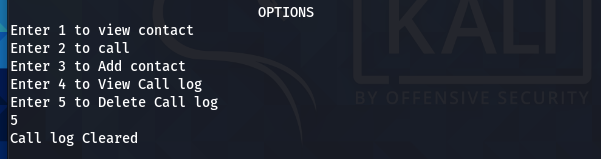
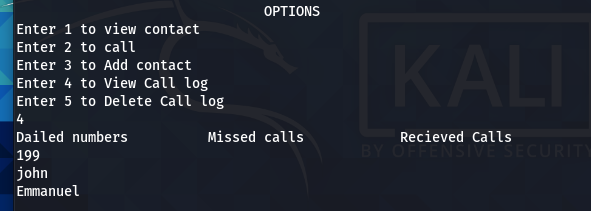
****

****

****

****

****

****