A simple record system in x86 assembler

Student number: 19131287

MSc in Computing Science for Cyber Security

Module: NETW7006
Word count: 1627

---

1. Introduction

The goal of this task was to develop a software in x86 64bit assembly, that would store user and computer information.

As a help, we received an IO library for the use with SASM, written by Joe Norton. It includes following IO functions we could make use of:

Function	Explanation
print_string_new	Prints a string to the command line. A pointer to the string is expected in rdi
print_uint_new	Prints an unsigned int to the command line. The uint is expected in rdi
print_int_new	*Has not been used* Prints a signed int to the command line. The int is expected in rdi
print_char_new	Prints a char to the command line. The char is expected in rdi
print_nl_new	Prints a new line to the command line
read_char_new	*Has not been used* Reads a char from command line and puts in rax
read_string_new	Reads a string from the command line and puts a pointer in rax
atoi	Reads a string pointer in rdi and converts to an int and puts in rax
read_uint_new	Reads an unsigned int from the command line and puts in rax
read_int_new	*Has not been used* Reads a signed int from the command line and puts in rax
copy_string	*Has not been used* copies a string pointed to in rsi to pointer in rdi
strings_are_equal	compares strings in rsi and rdi. Return 1 in rax if equal 0 if not
print_adress_new	*Has not been used* prints pointer address in rdi

2. The Setup

The IDE SASM on lUbuntu has been used to write, debug and execute the software. The version control git has been used, with the remote being located on GitHub.

Name	Notes	Type	64-bit long	32-bit int	16-bit short	8-bit char
rax	Values are returned from functions in this register.	scratch	rax	eax	ax	ah and al
rcx	Typical scratch register. Some instructions also use it as a counter.	scratch	rcx	ecx	cx	ch and cl
rdx	Scratch register.	scratch	rdx	edx	dx	dh and dl
rbx	*Preserved register: don't use it without saving it!*	*preserved*	*rbx*	*ebx*	*bx*	*bh and bl*
rsp	*The stack pointer. Points to the top of the stack (details coming soon!)*	*preserved*	*rsp*	*esp*	*sp*	*spl*
rbp	*Preserved register. Sometimes used to store the old value of the stack pointer, or the "base".*	*preserved*	*rbp*	*ebp*	*bp*	*bpl*
rsi	Scratch register used to pass function argument #2 in 64-bit Linux. In 64-bit Windows, a preserved register.	scratch	rsi	esi	si	sil
rdi	Scratch register and function argument #1 in 64-bit Linux. In 64-bit Windows, a preserved register.	scratch	rdi	edi	di	dil
r8	Scratch register. These were added in 64-bit mode, so they have numbers, not names.	scratch	r8	r8d	r8w	r8b
r9	Scratch register.	scratch	r9	r9d	r9w	r9b
r10	Scratch register.	scratch	r10	r10d	r10w	r10b
r11	Scratch register.	scratch	r11	r11d	r11w	r11b
r12	*Preserved register. You can use it, but you need to save and restore it.*	*preserved*	*r12*	*r12d*	*r12w*	*r12b*
r13	*Preserved register.*	*preserved*	*r13*	*r13d*	*r13w*	*r13b*
r14	*Preserved register.*	*preserved*	*r14*	*r14d*	*r14w*	*r14b*
r15	*Preserved register.*	*preserved*	*r15*	*r15d*	*r15w*	*r15b*

Figure 1: x64 Registers (Lawlor)

3. Software Design

3.1 Memory

The software contains two arrays, one holding the users and the other the computers. Both are placed in the .data section and are allocated with 0 at the beginning. The data written in the arrays during execution is therefore being dropped as soon as the program finishes. A possible implementation for saving the data before quitting the program has been developed and can be found in Appendix 2.

3.1.1 Users

Each user consists out of 5 variables, which need to be stored:

- Surname name
- First Name
- Dept (Development, IT Support, Finance, or HR)
- User ID
- Email address

Surname, name and email are all represented as a String of 64 characters. The department is represented as an enumeration INT8. The user ID consists of a character p followed by 7 numbers. This could either be stored as an 8-char string or as an unsigned INT32.

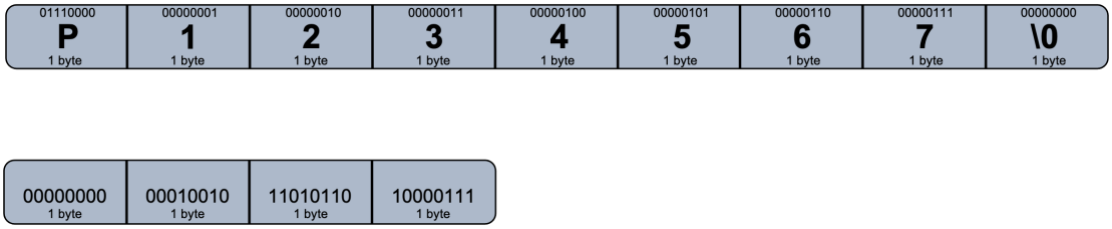

Figure 2: User ID as string (top) vs User ID as INT32 (bottom) memory usage

As clearly visible in Figure 2, the memory needed decrements by 5byte for each User ID, if stored as INT32. For this reason, the Integer representation for the User ID and the very similar Computer ID have been chosen. This decision decreases the size of the space needed by 500byte for the user array and 5kbyte for the computer array.

Important to notice is that this representation does not make comparisons between IDs easier and quicker. The first 8 byte (64bit) of the string representation would be enough for a unique identification, as the null terminator in the 9th byte does not add any information to the comparison. The first 8 byte can be loaded all in once in a 64bit register and then compared with another ID in a different 64bit register. If the user ID was longer then 8 characters, this would not be possible. The string-comparison needed for an ID longer then 8 characters would not result in being an atomic operation, as the register comparison is. Therefore, the integer representation would be more efficient for comparisons of longer IDs.

The final structure for one user can be seen in Figure 3. And the total memory needed for 100 users is 20kbyte.

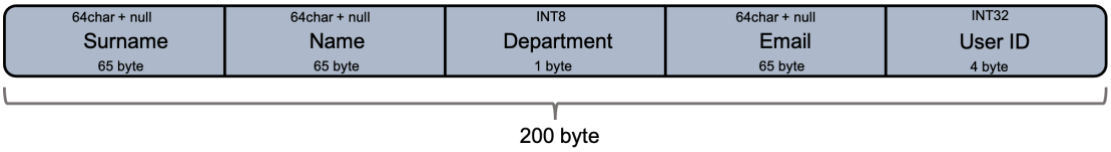

Figure 3: User memory structure

3.1.2 Computers

Similar to the user, the Computer also has 5 variables.

- Computer ID
- IP address
- OS (Linux, Windows, or Mac OSX)
- User ID of main user
- Date of purchase

As already discussed for the User ID, the Computer ID which consist out of a c followed by 7 numbers, will be stored as an INT32 value. The Main User ID corresponds to a User ID and therefore is also a INT32 value. The OS will be represented as an enumeration INT8. The IP address will be represented by 4 INT8. The date of purchase will be represented as two INT8, which represent respectively the day and month and one INT16, which represents the year. Therefore, one computer needs 17byte and the memory structure can be seen in Figure 4. The computer array needs a total storage of 8,5kbyte.

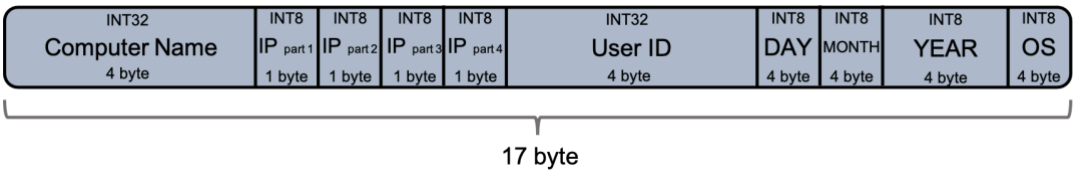

Figure 4: Computer Memory structure

3.2 Getting the Date

A Computer holds a variable, which represents its purchase date. When the user inputs the purchase date, 3 checks will be made about the input. One concerns the input format, which should be dd.mm.yyyy. The second check makes sure a day can only be between 1-31, a month only between 1-12 and the year only between 1901 and 2100. The third check makes sure that the date entered is not in the future. For this check, the System call 201 (Valsorda) was called to get the second since epoch (1.1.1970). A custom function has been created to translate the second into the current date and make a comparison.

3.3 Call Diagram

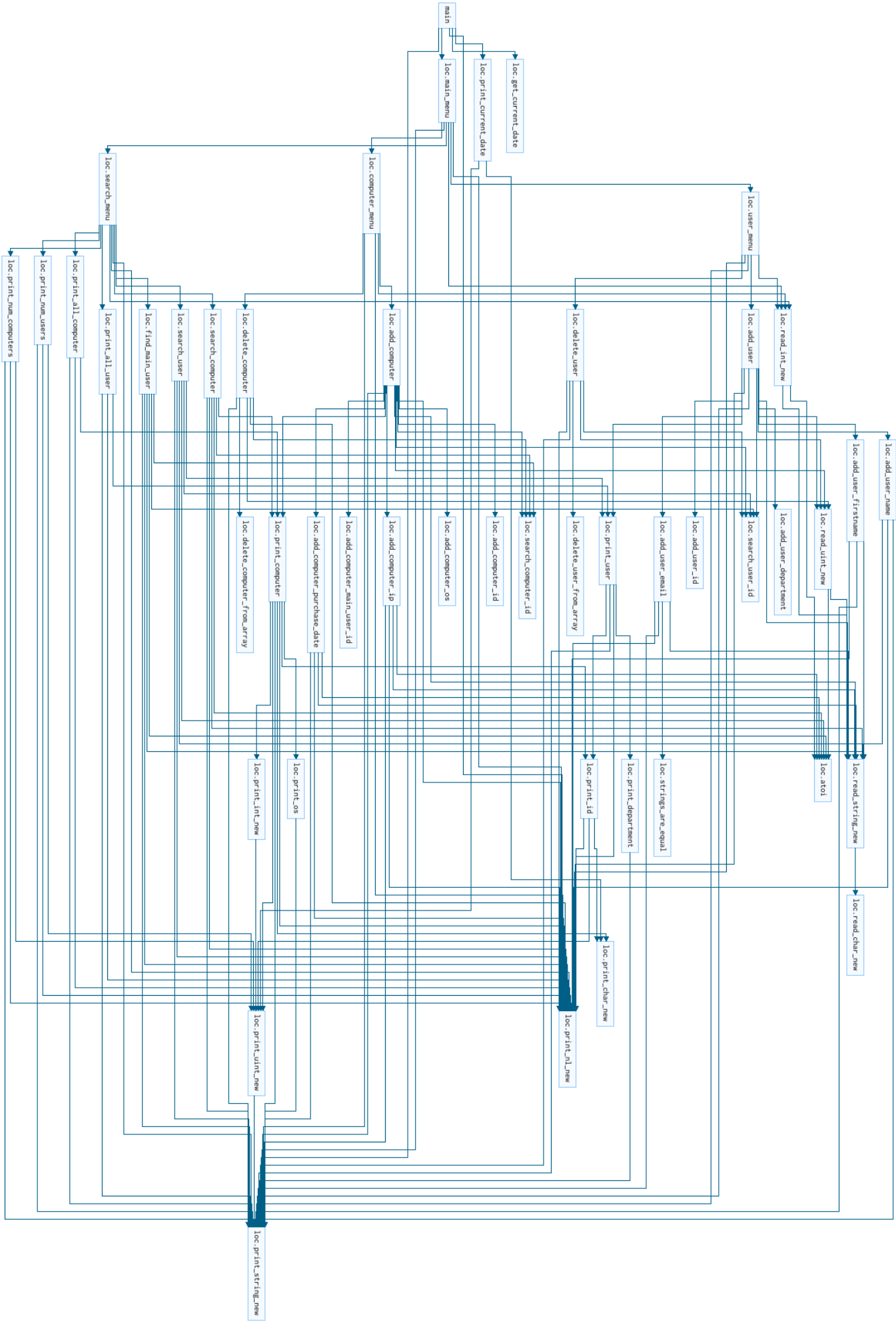

Figure 5: Global Call Graph generated by Cutter

A more detailed explanation of each of the functions shown in Figure 5 can be found in Appendix 1 which represents the Code Documentation.

3.4 CLI

The Command Line Interface (CLI) has been structured in 3 sub menus and one main menu. The 3 menus and the program functions availiabe in each menu are represented in Figure 5.

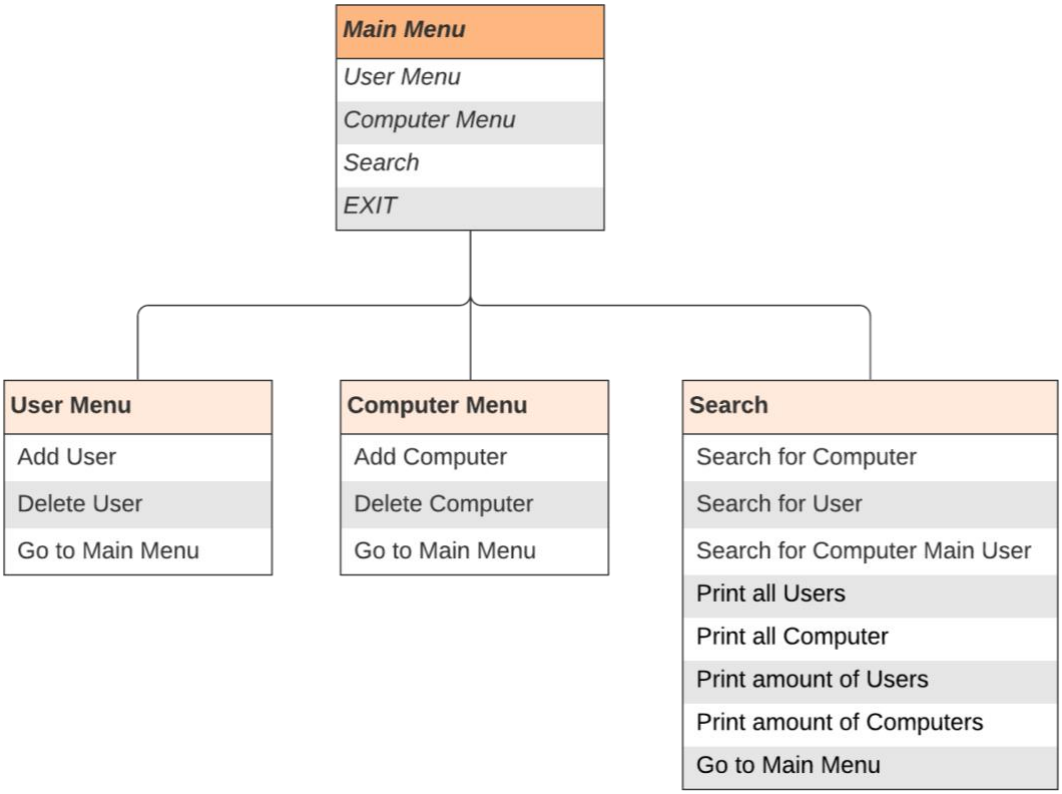

Figure 6: Menu Structure

As seen in the screenshots in Figure 7 to 10, the navigation between the menus is handled by reading the user input, which should correspond to a menu number.

```
------------------------------------------------
STUDENT AND COMPUTER ORGANISER 3000
------------------------------------------------
© Alexander Sworski 19131287

Today is the: 26.02.2021


Please select one of the following options:

1. Manage Users
2. Manage Computers
3. Search
4. EXIT

Enter the number of the menu you want to enter:
```

Figure 7: Main Menu screenshot

```
You are in the Search Menu
Please select one of the following options:

1. Search for a Computer
2. Search for a User
3. Search for Computer Main User
4. Print all Users
5. Print all Computers
6. Print amount of Users
7. Print amount of Computers
8. Go to Main Menu

Enter the number of the menu you want to enter:
```

Figure 8: Search Menu screenshot

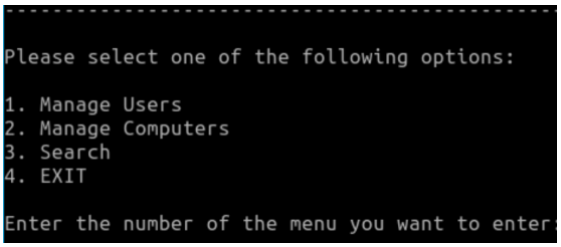

Figure 9: User Menu screenshot

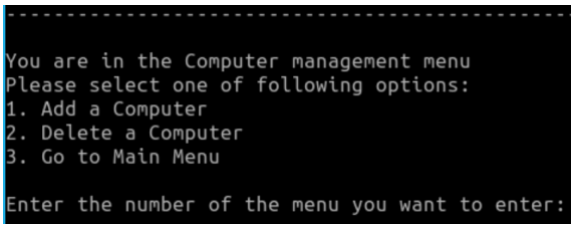

Figure 10: Computer Menu Screenshot

4. Testing

4.1 User Menu

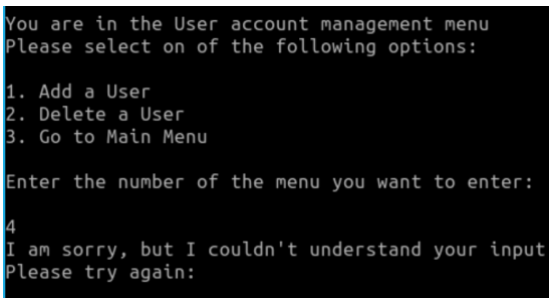

Figure 11: Wrong User Menu Input (positive)

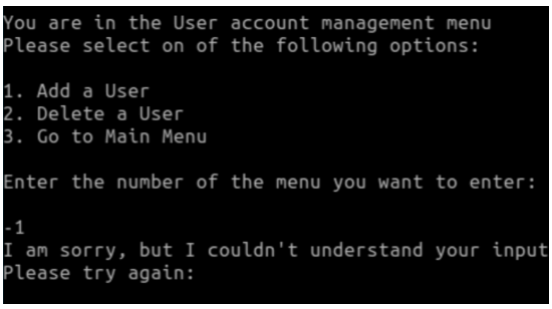

Figure 12: Wrong User Menu Input (negative)

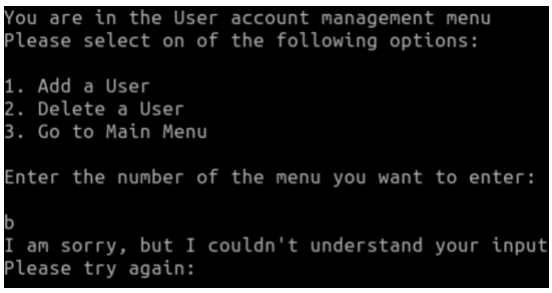

Figure 13: Wrong User Menu Input (alphabetic)

The program checks for a valid input and, if the input is not valid, asks the user to re-enter the selection.

4.1.1 Adding a User

Figure 14: Adding a User correctly

In Figure 13 we can see that, if the inputs are valid, everything works as expected. The program saves all the inputs and then prints the created user by reading from the user array.

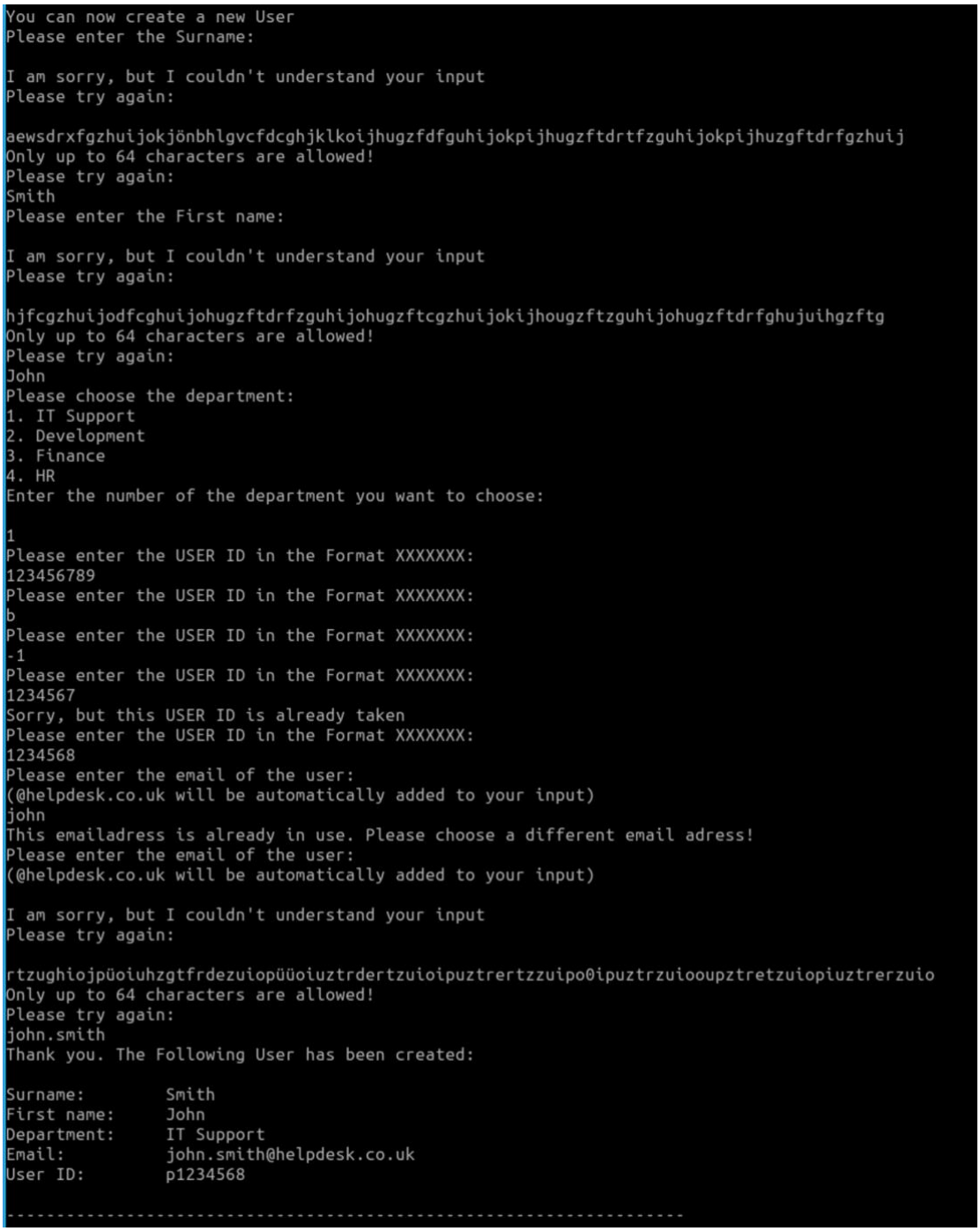

Figure 15: Adding a User with wrong input

In Figure 14 we are deliberately entering wrong inputs to the program to check error handling. As expected, the program detects the wrong inputs and outputs error messages. If the user tries to enter a User ID that has already been used, it detects the duplicate and outputs an error, asking the user to enter a different ID. The same applies to the email address, as the email address should be unique for every user. If the entered email address is already used by a different user in the system, the program will ask for a different input. If a string input is longer then 64characters (name, first name & email), the program asks the user to change the input, according to the 64character limitation.

4.1.2 Deleting a User

```
You selected: Delte a user
Please enter the User ID you want to delete in following format XXXXXXX:
12345678
I am sorry, but I couldn't understand your input
Please try again:
```

Figure 16: delete user too long user id input

```
You selected: Delte a user
Please enter the User ID you want to delete in following format XXXXXXX:
fewds
I am sorry, but I couldn't understand your input
Please try again:
```

Figure 17: delete user alphabetical id input

```
You selected: Delte a user
Please enter the User ID you want to delete in following format XXXXXXX:
-1
I am sorry, but I couldn't understand your input
Please try again:
```

Figure 18: delete user negative input

```
You selected: Delte a user
Please enter the User ID you want to delete in following format XXXXXXX:
1234566
The User could not be found
```

Figure 19: delete user input of id, that doesn't exist in the user array

```
You selected: Delte a user
Please enter the User ID you want to delete in following format XXXXXXX:
1234568
The User has been delteted
```

Figure 20: delete user with id 1234568, successful

The program checks if it is a valid ID, and if it is, it will check if the ID exists. If it is not a valid ID or user does not exist, an error will be displayed as seen in Figure 16 to 19. If the user ID entered exists, the User will be deleted (Figure 20).

4.2 Computer Menu

```
You are in the Computer management menu
Please select one of following options:
1. Add a Computer
2. Delete a Computer
3. Go to Main Menu

Enter the number of the menu you want to enter:

4
I am sorry, but I couldn't understand your input
Please try again:
```

Figure 21: Wrong Computer Menu Input (positive)

```
You are in the Computer management menu
Please select one of following options:
1. Add a Computer
2. Delete a Computer
3. Go to Main Menu

Enter the number of the menu you want to enter:

b
I am sorry, but I couldn't understand your input
Please try again:
```

Figure 22: Wrong Computer Menu Input (alphabetical)

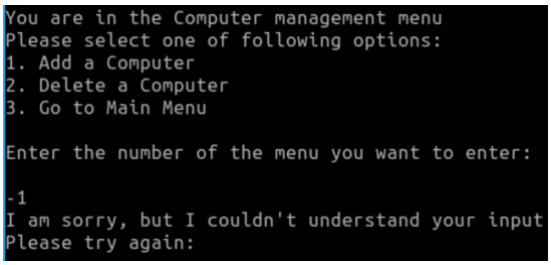

Figure 23: Wrong Computer Menu Input (negative)

4.2.1 Add Computer

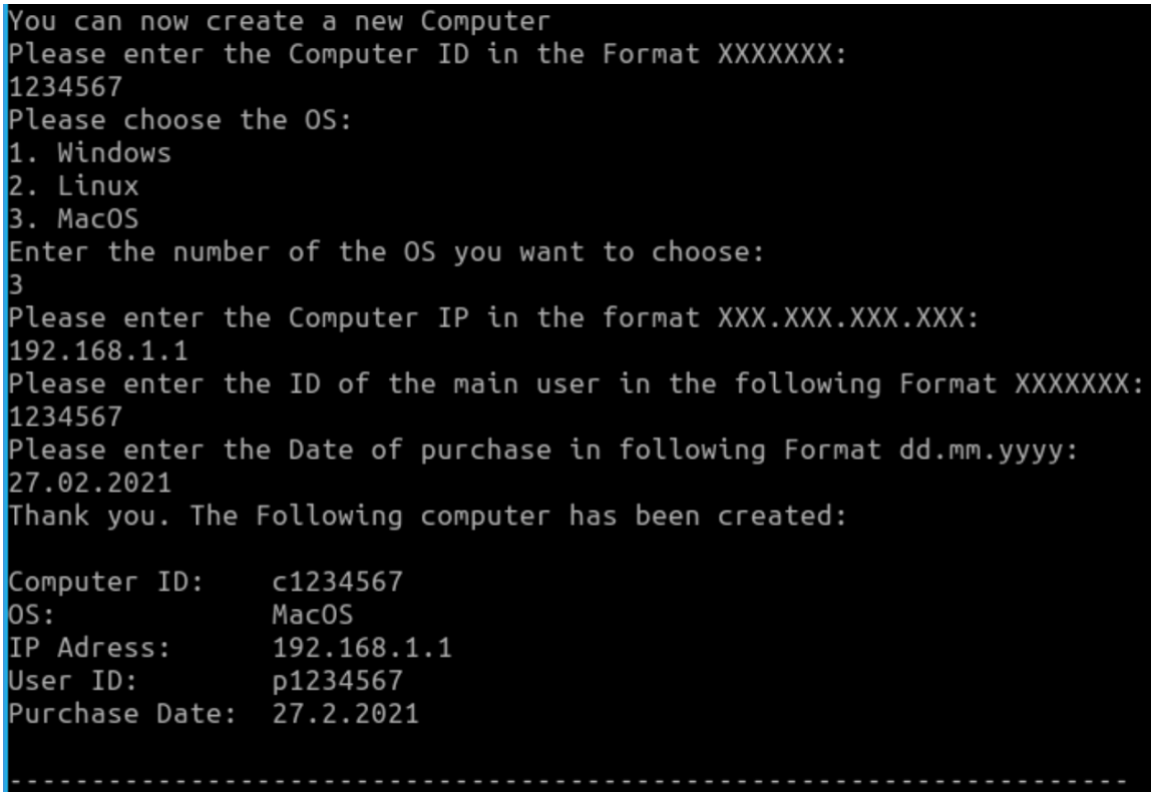

Figure 24: Add Computer with valid inputs

In Figure 20, a Computer has been added without any error, and was saved as expected.

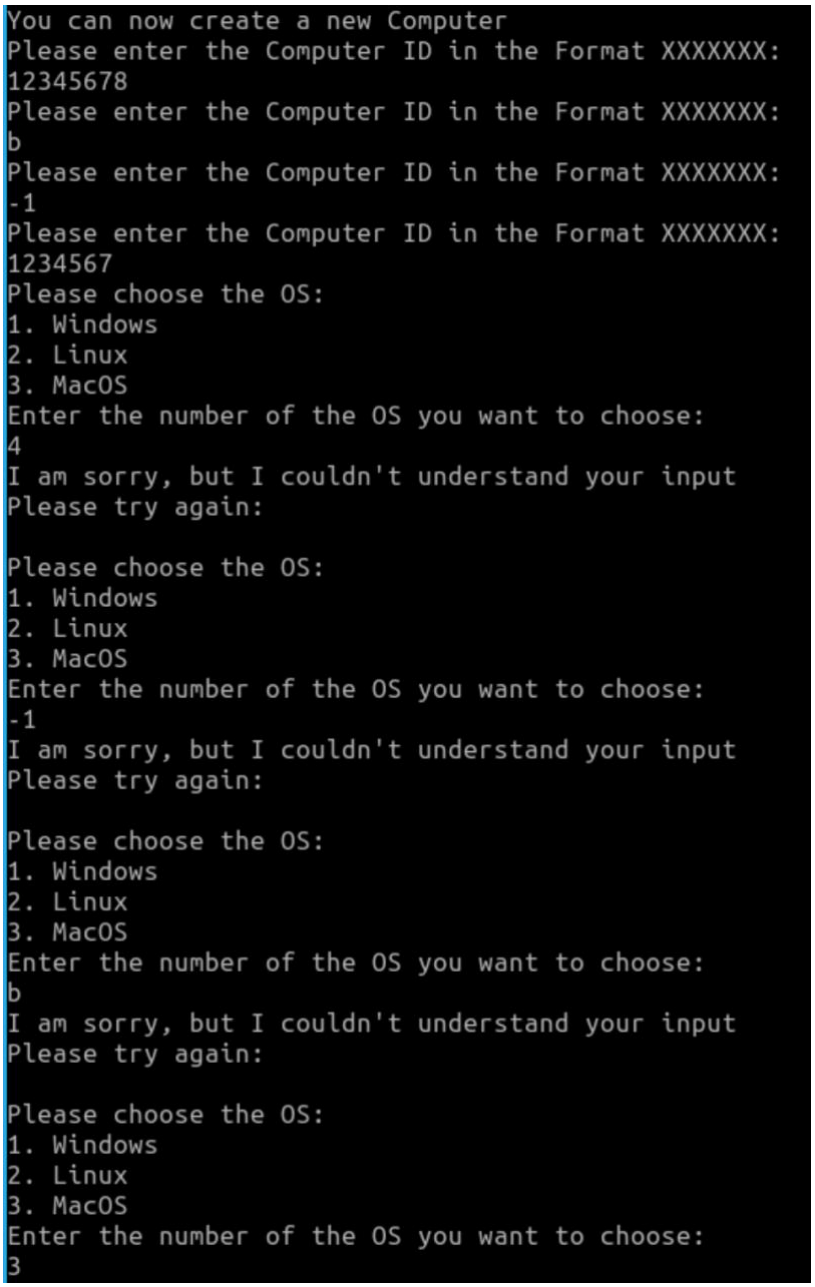

Figure 25: Adding a Computer with wrong input (ID, OS)

In figure 25 and 26 the user tried to enter an invalid input into the program and received error messages. In Figure 27 one can see that the program is checking the date entered by the user against the current date to make sure no future dates can be entered. Furthermore, the program checks if the IP entered is already used by another computer in the program and, if so, asks the user to enter a different IP. This makes sure that not only the Computer ID, but also the Computer IP is unique, as it would be within a Network.

```
You can now create a new Computer
Please enter the Computer ID in the Format XXXXXXX:
1234567
Sorry, but this Computer ID is allready taken
Please enter a differnt Computer ID:
Please enter the Computer ID in the Format XXXXXXX:
12345678
Please enter the Computer ID in the Format XXXXXXX:
1234568
Please choose the OS:
1. Windows
2. Linux
3. MacOS
Enter the number of the OS you want to choose:
2
Please enter the Computer IP in the format XXX.XXX.XXX.XXX:
192.168.1.1
Sorry, but this IP is already taken. IPs are unique in a Network. Pleas
XX:
192.168.1.2
Please enter the ID of the main user in the following Format XXXXXXX:
12345678
Please enter the ID of the main user in the following Format XXXXXXX:
b
Please enter the ID of the main user in the following Format XXXXXXX:
-1
Please enter the ID of the main user in the following Format XXXXXXX:
213
Sorry, but this User ID does not exists.
Do you want to go back to the main menu? (y/n)
n
Please enter the ID of the main user in the following Format XXXXXXX:
1234567
Please enter the Date of purchase in following Format dd.mm.yyyy:
25.02.2021
I am sorry, but the date you entered is in the future
Please enter the Date of purchase in following Format dd.mm.yyyy:
24.02.2021
Thank you. The Following computer has been created:

Computer ID:    c1234568
OS:             Linux
IP Adress:      192.168.1.2
User ID:        p1234567
Purchase Date:  24.2.2021

----------------------------------------------------------------------
```

Figure 26: Adding a Computer with wrong input values

4.2.2 Delete Computer

```
You selected: Delte a Computer
Please enter the Computer ID in the Format XXXXXXX:
12345678
I am sorry, but I couldn't understand your input
Please try again:
```

Figure 27: delete computer too long id input

```
You selected: Delte a Computer
Please enter the Computer ID in the Format XXXXXXX:
asdfs
I am sorry, but I couldn't understand your input
Please try again:
```

Figure 28: delete computer alphabetic input

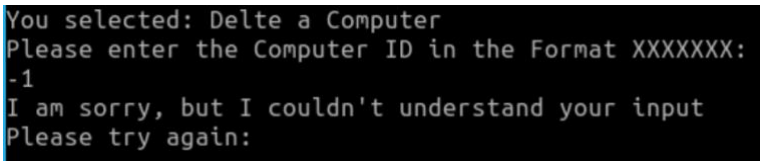

Figure 29: delete computer negative input

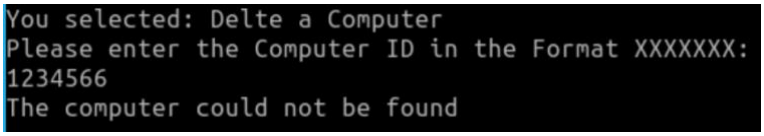

Figure 30: delete computer non existing id

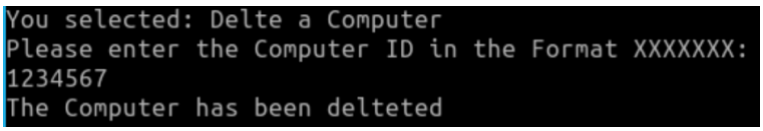

Figure 31: delete computer success

The program checks if it is a valid ID, and if it is, it will check if the ID exists. If it is not a valid ID or computer doesn't exist, an error will be displayed as seen in Figure 27 to 30. If the input is valid and a computer with this ID exist, the computer will be deleted as expected (Figure 31).

4.3 Search Menu

4.3.1 Search for Computer

```
You are in the Computer Search menu

Enter the Computer ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
12345678
I am sorry, but I couldn't understand your input
Please try again:


Enter the Computer ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
vb
I am sorry, but I couldn't understand your input
Please try again:


Enter the Computer ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
-1
I am sorry, but I couldn't understand your input
Please try again:


Enter the Computer ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
1234567
Following Computer has been found:
Computer ID:    c1234567
OS:             MacOS
IP Adress:      192.168.1.1
User ID:        p1234567
Purchase Date:  24.2.2021

Enter the Computer ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
1
The computer could not be found
```

Figure 32: Searching for a Computer correct and incorrect input

4.3.2 Search User

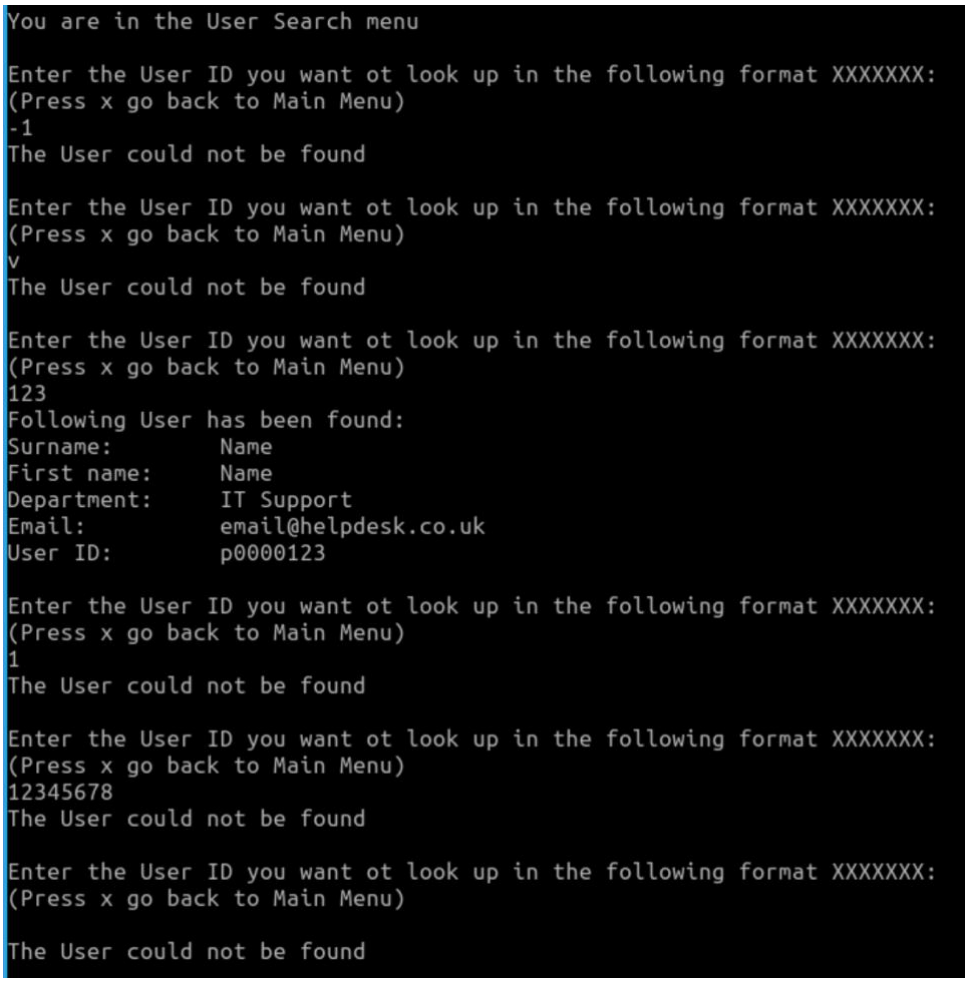

Figure 33: test search user with valid and invalid input

4.3.3 *Search from Computer Main User Email*

```
You are in the Email adress of main user Search menu

Enter the User ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
1
This is the email of the main user: m@helpdesk.co.uk

Enter the User ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
b
I am sorry, but I couldn't understand your input
Please try again:


Enter the User ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
-1
I am sorry, but I couldn't understand your input
Please try again:


Enter the User ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
123
The computer could not be found

Enter the User ID you want ot look up in the following format XXXXXXX:
(Press x go back to Main Menu)
23456789
I am sorry, but I couldn't understand your input
Please try again:
```

Figure 34: Search for Main User Email address test with valid and invalid input

4.3.4 *Print all Users*

```
------------------------------------------------------------------------
Printing all Users
------------------------------------------------------------------------



------------------------------------------------------------------------
------------------------------------------------------------------------
```

Figure 35: Print all users if user array empty

Figure 36: Print all users after adding 4 users

Figure 37: Print all users after deleting 1 user

4.3.5 Print all Computers

Figure 38: Print all computer if computer array empty

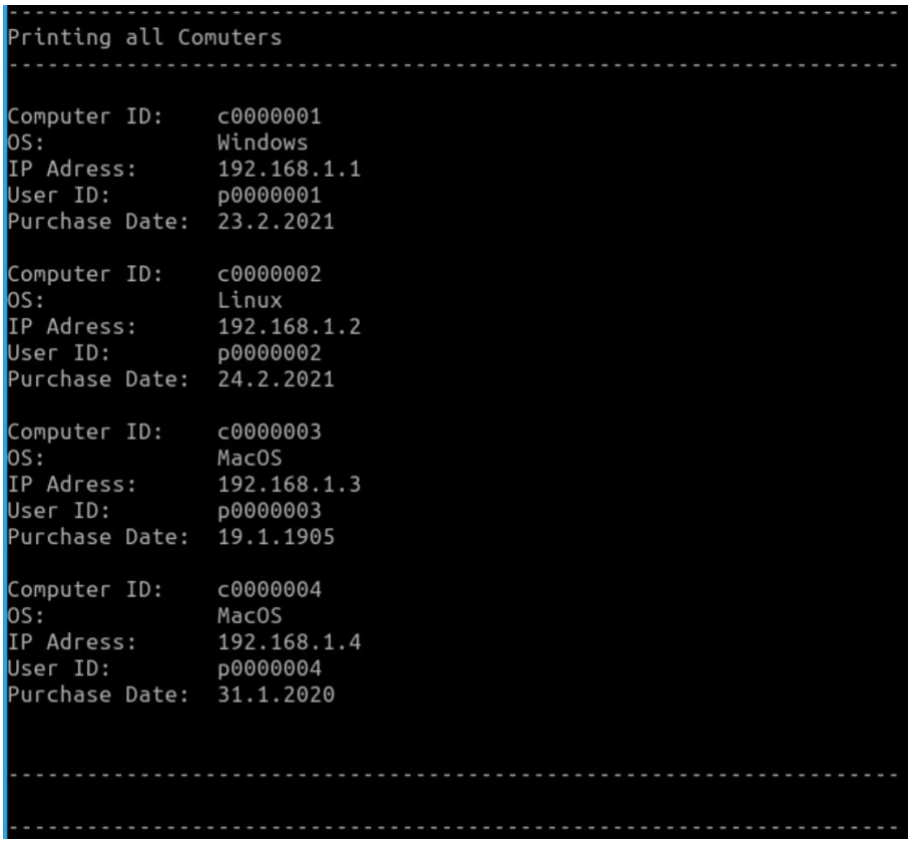

Figure 39: Print all Computer after adding 4 Computer

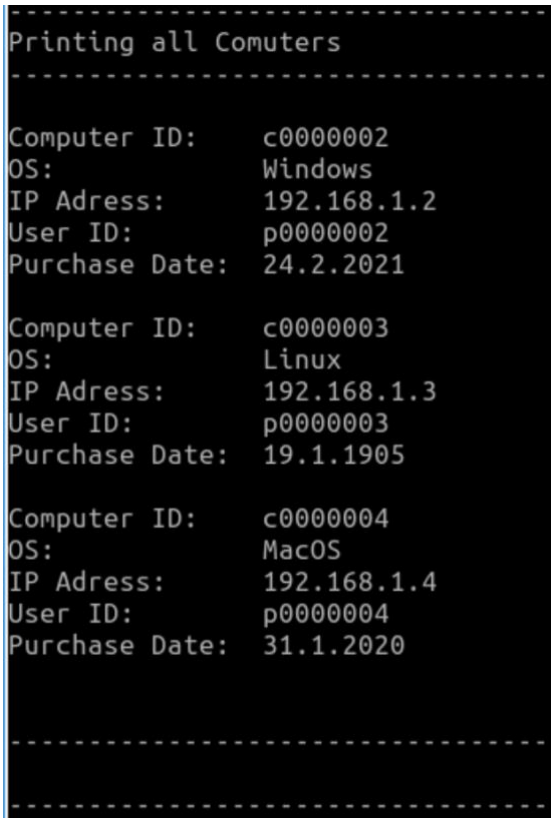

Figure 40: Print all Computer after deleting 1 Computer

4.3.6 Print Amount of Users

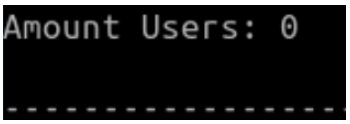

Figure 41: Amount User if array empty

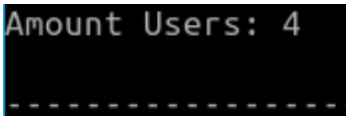

Figure 42: Amount Users after adding 4 users

Figure 43: Amount users after deleting 1 user

4.3.7 Print Amount of Computer

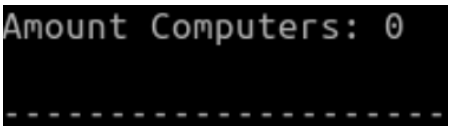

Figure 44: Amount Computer if array empty

Figure 45: Amount Computers after adding 4 computers

Figure 46: Amount Computers after deleting 1 user

5. Legal

Despite the Brexit, the GDPR is still the legislation to consider if personal data is being processed. In this case we are storing Name and Surname of the User. This ensures that the system knows who is using the computers. This is necessary for reliability, for example in case of damages. The name might also be necessary for other internal communications. Personal Information, stated by the GDPR, can only be stored as long as the data is required (Woollven, 2019). For this reason, the software would need to implement a policy (automated or manual by the admin) that deletes user information after they served their purpose and are no longer required.

6. Conclusion

The system has been implemented in NASM according to the requirements. Certain features, as insurances of a unique IP and email address, have been added. Also, other checks, as checking for dates in the future, have been implemented in order to improve the user experience. As part of this project, it has not been elaborated in detail how the data could be saved permanently, although a possible but not finalised solution has been added in Appendix 2. It has been worked out how the data could be stored in the least amount of space necessary and how to delete it again. However, a policy for the deletion process is still missing. This should be worked out in the future, as it is not clear what should happen to a Computer after its main user has been deleted. Nevertheless, this project can be considered a success, as the original goal was surpassed.

References

Lawlor, O. *x86_64 NASM Assembly Quick Reference ("Cheat Sheet").* cs.uaf.edu Available at: https://www.cs.uaf.edu/2017/fall/cs301/reference/x86_64.html.

Valsorda, F. *Searchable Linux Syscall Table for x86 and x86_64.* Available at: https://filippo.io/linux-syscall-table/ (Accessed: 26.02.2021).

Woollven, C. (2019) *List of mandatory documents required by the GDPR.* Available at: https://www.itgovernance.co.uk/blog/the-documents-you-need-to-comply-with-the-gdpr.

Code Documentation

A simple record system in x86 assembler

by Alexander Sworski

Version 1.0.0

A simple record system in 64 bit x86 assembler to keep track of computers and user.

global variables:

- **currentdate**:
  Contains the current date, will be filled by the function get_current_date
- **users**:
  A array of 100 users
- **computers**:
  A array of 500 computers
- **user_index**:
  The count of how many users are in the array
- **computer_index**:
  The count of how many computer are in the array

functions:

print_id

Prints the ID int he Fromat XXXXXXX\n

- parameters:
  - *rdi*: `INT32` ID

print_current_date

Prints the curent date in the Format d.m.yyyy
call get_current_date before

get_current_date

save the current date to currentdate

print_os

Prints the name of the OS

- parameters:
  - *rdx*: `INT8` index
- errors:
  - prints error to the screen if index out of bound

print_department

Prints the name of the department

- parameters:
  - *rdx*: `INT8` index
- errors:
  - prints error to the screen if index out of bound

delete_user_from_array

Deletes a User at index from usera

- parameters:
  - *rax*: `INT64` index
- comments:
  - moves all folowing users one place forward in the array
- ***WARNING:***
  - no check if index out of bound

delete_computer_from_array:

deletes a Computer at index from computers

- parameters:
  - *rax*: `INT64` index

270

- comments:
  - moves all folowing computer one place forward in the array
- ***WARNING:***
  - no check if index out of bound

add_computer_id

Adds a computer id to the new computer in computers

- parameters:
  - *rbx*: `INT32` computer id

add_computer_ip

Add a IP to the new computer in computers

- parameters:
  - *rbx*: `string*` ip in format `XXX.XXX.XXX.XXX`
- checks:
  - check 1: input format correct, asks to enter it again if not
  - check 2: checks if the IP is already used by a different computer in the array

add_computer_main_user_id

Adds a user id to the new computer in computers

- parameters:
  - *rbx*: `INT32` user id
- comments:
  - should check in advance, if the user id exists

add_computer_os

As the enumaration of the OS to the new computer in computers

- parameters:
  - *rbx*: `INT8` os
- comments:
  - check in advance if os enumaration is correct

add_computer_purchase_date

Adds the purchase date to the new computer in computers

- parameters:
  - *rbx*: `string*` date in format dd.mm.yyyy
- checks:
  - check 1: date in valid format, other asks to enter again
  - check 2: check of day between 1-31, month between 1-12 and year between 1900-2100, other asks to enter again
  - check 3: check if date equals today or past. If date in future, then asks to enter again

print_computer

Prints computer at index from computers

- parameters:
  - *rax*: `INT64` index
- **WARNING:**:
  - no check for index out of bound

search_computer_id

Searches for a Computer with the ID in computers

- parameters:
  - *R13*: `INT32` Computer ID
- return:
  - *rax*: `INT64` index
- error:
  - *rax* = 504 "computer not found"
- comments:
  - No check if ID in valid format. Invalid input will not crash but give error 504

search_user_id

Searches for a User with the ID in computers

- parameters:
  - *R13*: `INT32` User ID
- return:
  - *rax*: `INT64` index
- error:
  - *rax* = 504 "user not found"
- comments:
  - No check if ID in valid format. Invalid input will not crash but give error 504

add_user_name

Add User Name of new User in users

- parameters:
  - *rbx*: `string*` name
- checks:
  - check 1: if string lenght between 1 and 64, otherwise asks to enter name again

add_user_firstname

Add User First Name of new User in users

- parameters:
  - *rbx*: `string*` name
- checks:
  - check 1: if string lenght between 1 and 64, otherwise asks to enter frist name again

add_user_department

Adds the department enummaration to the new user in users

- parameters:
  - *rbx*: `INT8` department
- comments:
  - The department will not be checked and should be checked beforehand

add_user_email

Add the user email to the new user in users

- parameters:
  - *rbx*: `string*` email
- checks:
  - check 1: if string between 1 and 64 chars long, otherwise ask to reenter
  - check 2: check if email has not yet been taken by another user in the array, otherwise ask to reenter

add_user_id

Add the id to the new user in users

- parameters:

    - *rbx*: `INT32` ID
- **WARNING:**:
    - does not check if ID unique, needs to be checked before!

print_user

Prints the user at index in users

- parameters:
    - *rbx*: `INT64` index

main

Prints welcoem message

- commenst:
    - calls get_current_date
    - calls main_menu

main_menu

prints options in the main menu and waits for user selection
Options:

1. Manage Users
2. Manage Computers
3. Search
4. EXIT

- checks if user selection valid and then calls the selected menu, otherwise asks for reseletion

user_menu

shows user menu
prints options in the user menu and waits for user selection
Options:

1. Add Users
2. Delete User
3. Go back to main menu

- checks if user selection valid and then calls the selected option, otherwise asks for reseletion

computer_menu

shows computer menu
prints options in the computer menu and waits for user selection
Options:

1. Add Computer
2. Delete Computer
3. Go back to main menu

- checks if user selection valid and then calls the selected option, otherwise asks for reseletion

search_menu

shows search menu
prints options in the search menu and waits for user selection
Options:

1. Search Computer
2. Search Users
3. Search Main User
4. Print all User
5. Print all Computer
6. Amount of Users
7. Amount of Computers
8. Go back to main menu

- checks if user selection valid and then calls the selected option, otherwise asks for reseletion

add_user

Shows the Input Masks for adding a User

- checks:
  - check 1: department code betweem 1-3
  - check 2: user id valid format
  - check 3: user id unique
  - check 4: user array still has space

delete_user

Shows the Input Masks for deleting a User

- checks:
  - check 1: user id valid format

275

    - check 2: user id exists

add_computer

Shows the Input Masks for adding a Computer

- checks:
    - check 1: os code betweem 1-3
    - check 2: computer id valid format
    - check 3: computer id unique
    - check 4: computer array still has space
    - check 5: main user id exists
    - check 6: main user id valid format

delete_computer

Shows the Input Masks for deleting a Computer

- checks:
    - check 1: computer id valid format
    - check 2: compiter id exists

search_user

Shows the Input Masks for searching a User

- comment:
    - User can exit by enter x
- checks:
    - check 1: user id has valid format

search_computer

Shows the Input Masks for searching a computer

- comment:
    - computer can exit by enter x
- checks:
    - check 1: computer id has valid format

find_main_user

Shows the Input Masks for searching a main user of a computer

Appendix 2: (Read & Write to file)

```
%include "/home/malware/asm/joey_lib_io_v6_release.asm"

; This file represents a possible solution to save the user and computer arrays to a file, in order to not lose the saved date when the program finishes.
; This solution presents though is not elegant, as it needs 4 files, which would need to be saved in the save directory as the executable.
; A more elegant solution would be to save all the program data in one file
; or to save the 4 files in a directory hidden from the user

global main

section .data

error: db "error",0


user_file_location: db "users",0
computer_file_location: db "computers",0

user_index_file_location: db "users_index",0
computer_index_file_location: db "computers_index",0

;arrays
users: times 20000 db 0
computers: times 8500 db 0

;index
user_index: dq 0
computer_index: dq 0

section .bss
;File descriptors
user_file_descriptor: resq 1
computer_file_descriptor: resq 1
user_index_file_descriptor: resq 1
computer_index_file_descriptor: resq 1

section .text
```

```
read:
    push    rbp
    mov rbp, rsp
    sub rsp, 32
    mov rax, 2 ;open
    mov rdi, qword user_file_location
    mov rdx, 0666o
    mov rsi, 0102o
    syscall
    mov QWORD[user_file_descriptor], rax
    cmp rax, 0
    jl .error
    mov rdi, rax

    mov rax, 2 ;open
    mov rdi, qword computer_file_location
    mov rdx, 0666o
    mov rsi, 0102o
    syscall
    mov QWORD[computer_file_descriptor], rax
    cmp rax, 0
    jl .error
    mov rdi, rax

    mov rax, 2 ;open
    mov rdi, qword user_index_file_location
    mov rdx, 0666o
    mov rsi, 0102o
    syscall
    mov QWORD[user_index_file_descriptor], rax
    cmp rax, 0
    jl .error
    mov rdi, rax

    mov rax, 2 ;open
    mov rdi, qword computer_index_file_location
    mov rdx, 0666o
    mov rsi, 0102o
```

```
    syscall
    mov QWORD[computer_index_file_descriptor], rax
    cmp rax, 0
    jl .error
    mov rdi, rax

    mov rax, 0 ;read
    mov rdi, qword[user_file_descriptor]
    mov rsi, qword users
    mov rdx, 20000
    syscall
    cmp rax, 0
    jl .error
    mov rdi, rax

    mov rax, 0 ;read
    mov rdi, qword[computer_file_descriptor]
    mov rsi, qword computers
    mov rdx, 8500
    syscall
    cmp rax, 0
    jl .error
    mov rdi, rax

    mov rax, 0 ;read
    mov rdi, qword[user_index_file_descriptor]
    mov rsi, qword user_index
    mov rdx, 4
    syscall
    cmp rax, 0
    jl .error
    mov rdi, rax

    mov rax, 0 ;read
    mov rdi, qword[computer_index_file_descriptor]
    mov rsi, qword computer_index
    mov rdx, 4
    syscall
    cmp rax, 0
```

```
        jl .error
        mov rdi, rax
.end:
        add rsp, 32
        pop rbp
        ret
.error:
        mov rdi, qword error
        call print_string_new
        call print_nl_new
        jmp .end

save:
        push    rbp
        mov rbp, rsp
        sub rsp, 32
        mov rax, 1 ;write
        mov rdi, qword[user_file_descriptor]
        mov rsi, qword users
        mov rdx, 20000
        syscall
        cmp rax, 0
        jl .error
        mov rdi, rax

        mov rax, 1 ;write
        mov rdi, qword[computer_file_descriptor]
        mov rsi, qword computers
        mov rdx, 8500
        syscall
        cmp rax, 0
        jl .error
        mov rdi, rax

        mov rax, 1 ;write
        mov rdi, qword[computer_index_file_descriptor]
        mov rsi, qword computer_index
        mov rdx, 4
        syscall
```

```
        cmp rax, 0
        jl .error
        mov rdi, rax

        mov rax, 1 ;write
        mov rdi, qword[user_index_file_descriptor]
        mov rsi, qword user_index
        mov rdx, 4
        syscall
        cmp rax, 0
        jl .error
        mov rdi, rax
.end:
        add rsp, 32
        pop rbp
        ret
.error:
        mov rdi, qword error
        call print_string_new
        call print_nl_new
        jmp .end
```