VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

OPERATING SYSTEM

ATM MACHINE USING SYSCALLS

FACULTY: PADMA PRIYA

GROUP MEMBERS:

NAME	REG. NO.	EMAIL ID
ROHAN MITTAL	18BCE0503	rohan.mittal2018@vitstudent.ac.in
RITWIK SHARAN	18BCE2232	ritwik.sharan2018@vitstudent.ac.in
SHASHANK SHUKLA	18BCE2522	shashank.shukla2018@vitstudent.ac.in
ISHAN SOURAV	18BCI0059	ishan.sourav2018@vitstudent.ac.in

CONTENTS

ABSTRACT

This project is based on System Calls or sys calls. We know that in order to perform certain system related and app related tasks, we need large chunks of code, and need to execute them by running in a terminal or a compiler. So, in order to ease the process, we make syscalls for the specific code, and the user needs to only type a few letters or a word to access and run the code file immediately, hence easing the process.

Hence, in this project, we have demonstrated this process through the use of xv6, a learning OS, in which we have developed an ATM machine demo, where the user has to type whichever facility he wants i.e. options, withdraw, deposit, view_balance, and then the system will automatically ask the user for his/her details to modify the user’s record.

Introduction about Installation and Tools Used with relevant screen shots

We have used the following:

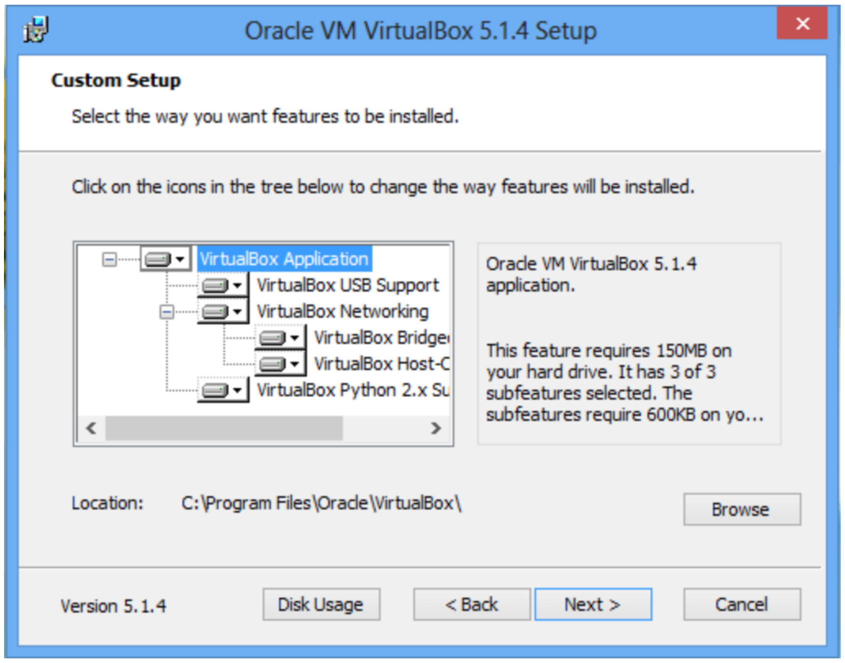
1 VirtualBox
2 xv6 OS

VIRTUALBOX

Step 1: Download VirtualBox installer for Windows

The installer can be found on its download page here https://www.virtualbox.org/wiki/Downloads

Go to the page above and download the binary version for **Windows hosts**
After downloading, run the executable to begin installing the software. When you start the installation, you should get a prompt like the one below.



Oracle VM VirtualBox 5.1.4 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridge
    - VirtualBox Host-C
  - VirtualBox Python 2.x Su

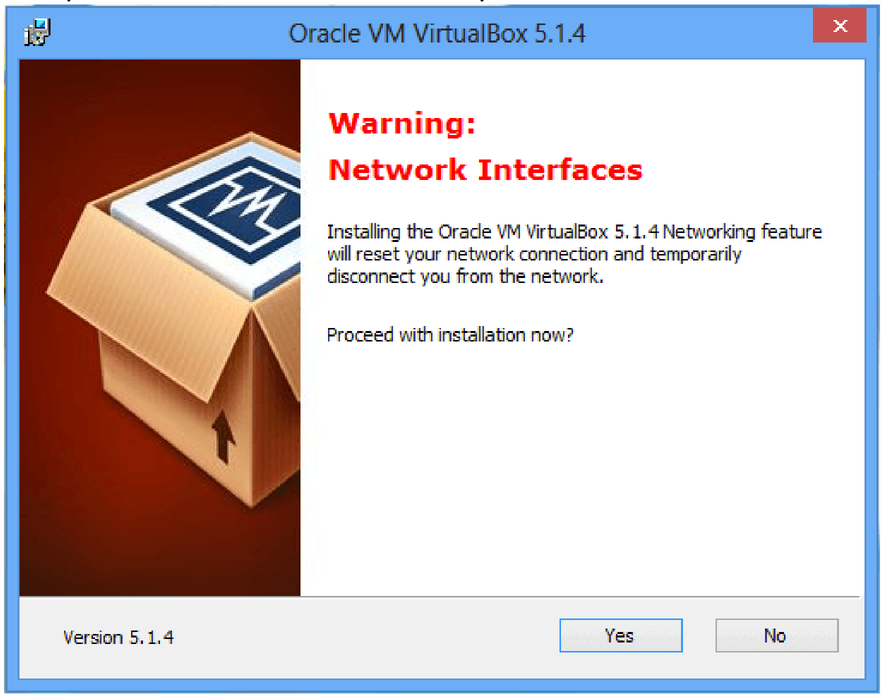
Oracle VM VirtualBox 5.1.4 application.

This feature requires 150MB on your hard drive. It has 3 of 3 subfeatures selected. The subfeatures require 600KB on yo...

Location: C:\Program Files\Oracle\VirtualBox\

Version 5.1.4

During the installation wizard, you shall obtain a network interfaces warning. Click yes and proceed with the installation process.



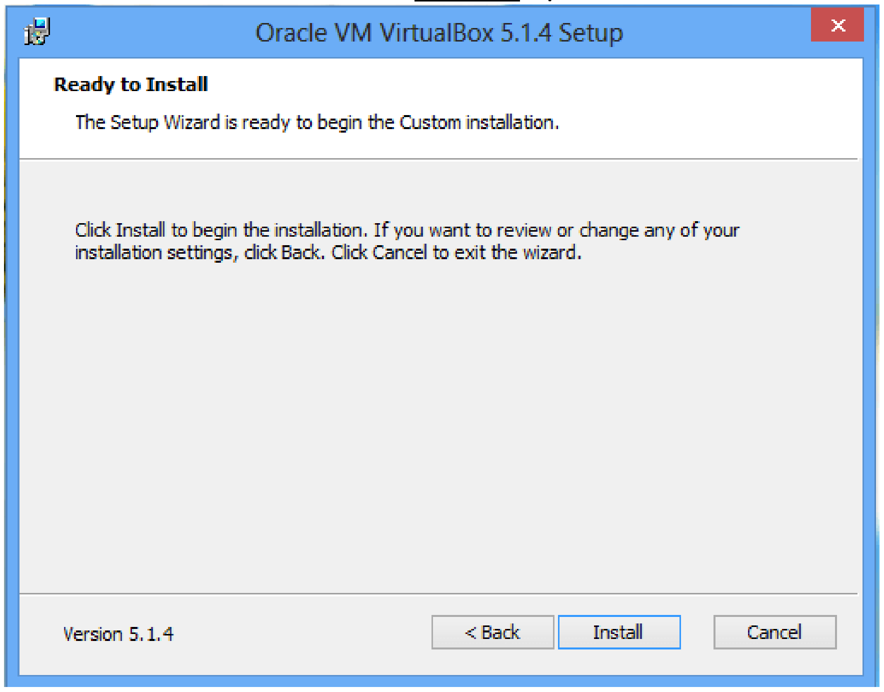
Warning:
Network Interfaces

Installing the Oracle VM VirtualBox 5.1.4 Networking feature will reset your network connection and temporarily disconnect you from the network.

Proceed with installation now?

Version 5.1.4

On the next screen, Click the **<u>'install'</u>** option.



Ready to Install

The Setup Wizard is ready to begin the Custom installation.

Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Version 5.1.4

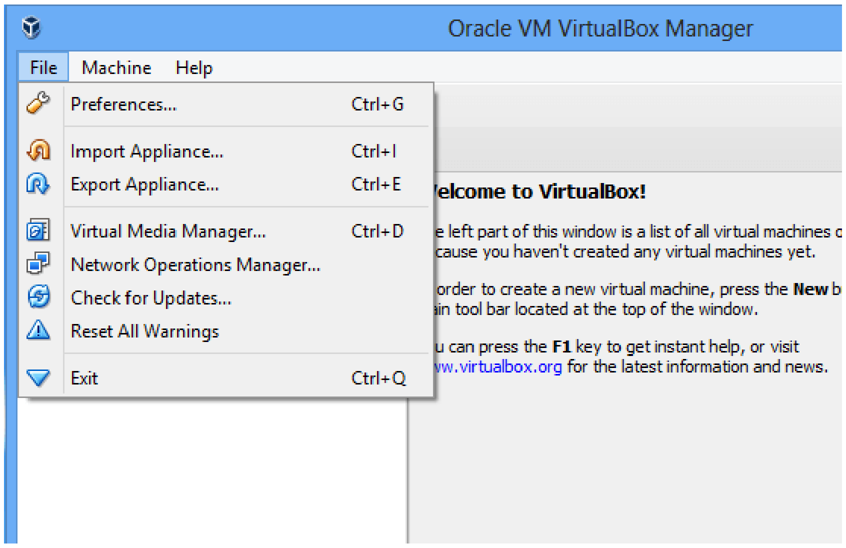
Continue with the wizard until you're done.
After that, VirtualBox should be installed. When you're done installing, open VirtualBox and install its extension pack.
The extension pack extends the functionality of VirtualBox base packages. It provides the following enhancements to VirtualBox:

a. Virtual USB 2.0 (EHCI) device
b. Virtual USB 3.0 (xHCI) device
c. VirtualBox Remote Desktop Protocol (VRDP) support
d. Host webcam passthrough
e. Intel PXE boot ROM.
f. Experimental support for PCI passthrough on Linux hosts
g. Disk image encryption with AES algorithm

To install the extension pack, go back to VirtualBox's download page , download and save the current pack for **all supported platforms**

Step 2: Install VirtualBox extension pack.

After downloading the saving he extension pack. open VirtualBox host software click **File –> Preferences**


Oracle VM VirtualBox Manager
File
Machine
Help
Preferences... Ctrl+G
Import Appliance... Ctrl+I
Export Appliance... Ctrl+E
Virtual Media Manager... Ctrl+D
Network Operations Manager...
Check for Updates...
Reset All Warnings
Exit Ctrl+Q

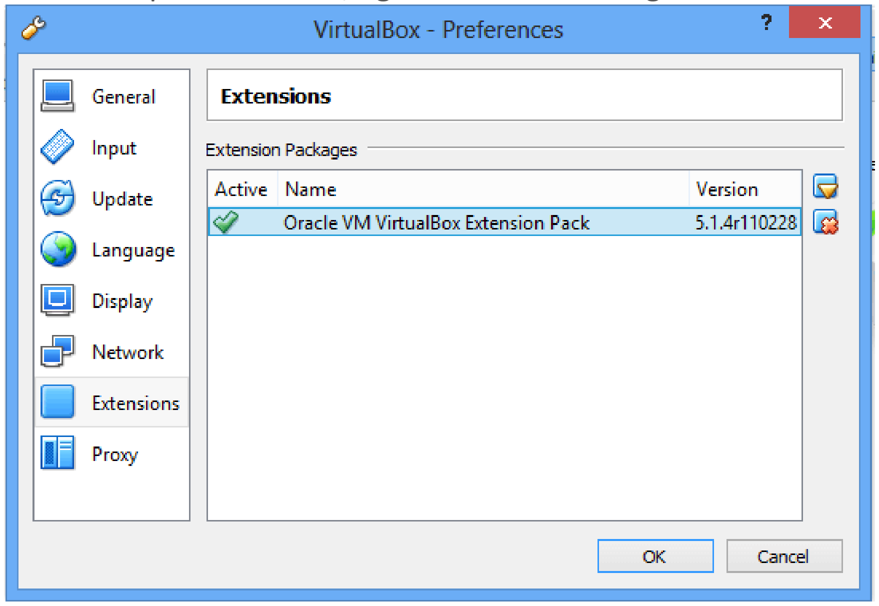
'elcome to VirtualBox!

e left part of this window is a list of all virtual machines c
cause you haven't created any virtual machines yet.

order to create a new virtual machine, press the **New** b
in tool bar located at the top of the window.

u can press the **F1** key to get instant help, or visit
ww.virtualbox.org for the latest information and news.

Then select extension, click the browse button to the right to find the download extension pack. Select it, agree to the licensing terms and install.



Active	Name	Version
✓	Oracle VM VirtualBox Extension Pack	5.1.4r110228

XV6 INSTALLATION:

Step 1 – Install qemu:

```
$ sudo apt install qemuCopy
```

If you have 64 bit OS there is a chance Makefile will not be able to find qemu. In that case you should edit the Makefile at line 54 and add the following code:

```
QEMU = qemu-system-x86_64Copy
```

Step 2 – Install xv6

Create a directory, and clone xv6 to that directory:

```
$ git clone git://github.com/mit-pdos/xv6-public.gitCopy
```

Step 3 – Compile xv6

```
$ makeCopy
```

Step 4 – Compile and run the emulator qemu:

```
$ make qemuCopy
```

The emulator will start but it will echo on the terminal too. Better to run it with the following frag (preferred):

```
$ make qemu-noxCopy
```

Also, for gdb enabled execution run:

```
$ make qemu-nox-gdbCopy
```

To enter the qemu console at any time, press ctrl+a, then c. If you want to exit, simply type "quit" to exit the emulator

Step 5 – Write an user application (eg: bla)

5.1) Create a copy of an existent program like wc and name it as your new user program (eg bla.c):

```
$ cp wc.c bla.cCopy
```

5.2) Edit bla.c with your favorite editor. Take a look at the source code examples of existent commands, in order to see the syntax of the available functions of xv6 (they are slightly different).
5.3) Modify the makefile of xv6 to include bla.c:
The simplest way is to open the makefile, search for wc and add bla next to it whenever it appears on the makefile
5.4) Clean and recompile xv6.
The new command should be available.

INTRODUCTION ABOUT THE CONCEPT/METHODOLOGY TAKEN

Through the mode of this project we aim at explaining how to make syscalls in xv6 and developing a few syscalls of our own.

What is xv6?

xv6 is a modern reimplementation of Sixth Edition Unix in ANSI C for multiprocessor x86 and RISC-V systems. It contains all the important concepts and organization of Unix. It can be used by a user to develop new syscalls to suit their purpose.

What is a Syscall?

In computing, a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system's kernel. System call provides the services of the operating system to the user programs via Application Program Interface(API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.
Using the syscalls, that have been made and modified in xv6, the learning OS. We are capable of making applications.

Adding a system call in XV6 OS

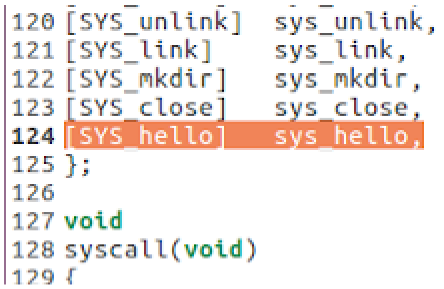
We will show you how to add a system call to XV6 OS. We will just add a simple HelloWorld system call which will print hello world and the argument passed to the system call.

Steps:

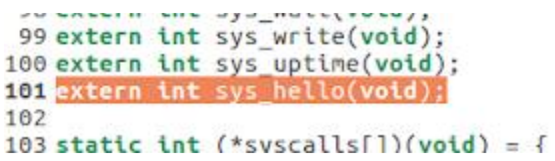
For adding the system call we need to make changes in the following files:

1. syscall.c
2. syscall.h
3. user.h
4. usys.S
5. sysproc.c

First, we add the call to the list in syscall.c

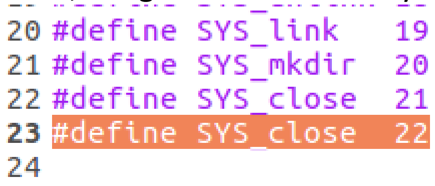


```
[SYS_unlink]  sys_unlink,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
[SYS_hello]   sys_hello,
};

void
syscall(void)
{
```



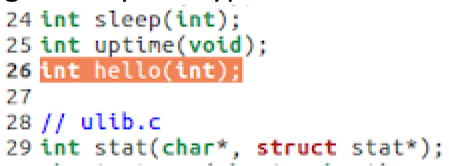
```
extern int sys_write(void);
extern int sys_uptime(void);
extern int sys_hello(void);

static int (*syscalls[])(void) = {
```

Next, assign it a number in syscall.h



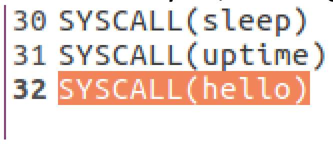
```
#define SYS_link    19
#define SYS_mkdir   20
#define SYS_close   21
#define SYS_close   22

```

give it a prototype in user.h:



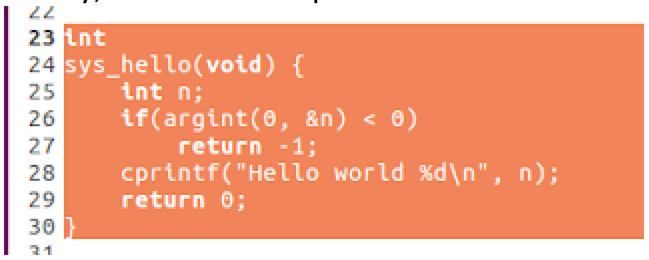
```
int sleep(int);
int uptime(void);
int hello(int);

// ulib.c
int stat(char*, struct stat*);
```

Add it to usys.S, which generates the user-space assembly code for it



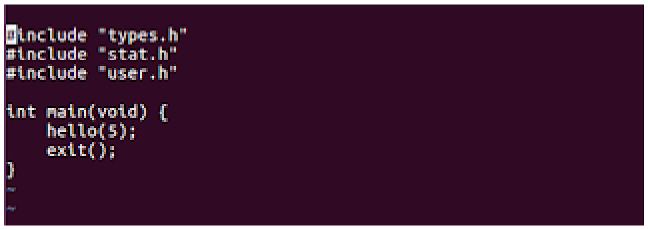
```
SYSCALL(sleep)
SYSCALL(uptime)
SYSCALL(hello)
```

Finally, we add the implementation somewhere (e.g. sysproc.c)



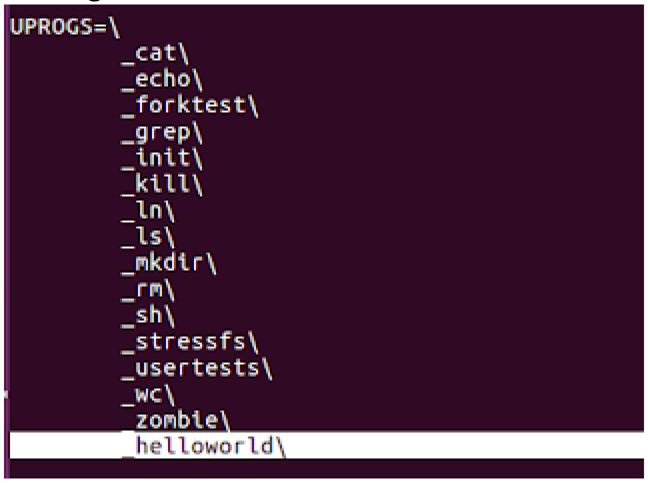
```
int
sys_hello(void) {
    int n;
    if(argint(0, &n) < 0)
        return -1;
    cprintf("Hello world %d\n", n);
    return 0;
}
```

Testing
To test if the system call works, create a c file and use the system call in it. Remember to add the c file in Makefile so that you can use it.



```
#include "types.h"
#include "stat.h"
#include "user.h"

int main(void) {
    hello(5);
    exit();
}
~
~
```

Adding it to make file



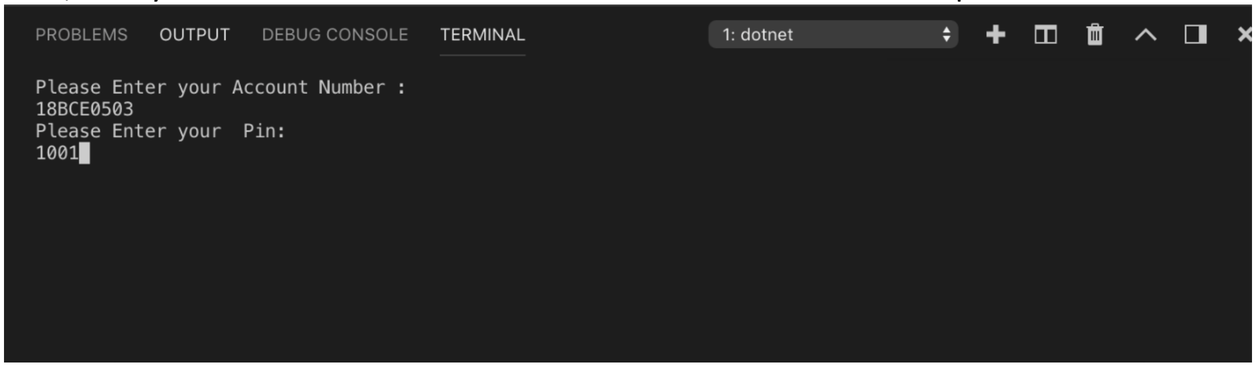
```
UPROGS=\
        _cat\
        _echo\
        _forktest\
        _grep\
        _init\
        _kill\
        _ln\
        _ls\
        _mkdir\
        _rm\
        _sh\
        _stressfs\
        _usertests\
        _wc\
        _zombie\
        _helloworld\
```

MODULES SHOWN IN PREVIOUS REVIEWS:

During the previous review, we had tried to demonstrate the working of our own OS which was basically an ATM facility using the COSMOS software. The code had been formed in c-sharp language i.e. '.cs' extension. The .cs file had to be loaded into the COSMOS software to be run. But the COSMOS software has been depreciated, hence we were not able to deploy it successfully. The code contained the provision to make an ATM in the terminal or COSMOS software. The code contained the following features:

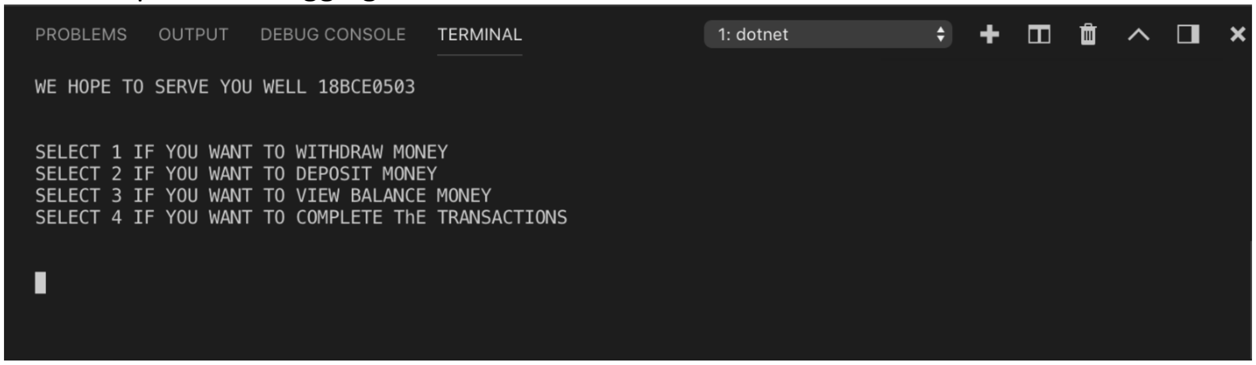
1. withdraw
2. deposit
3. view_balance
4. an exit() function to terminate all ongoing activities i.e. Shutdown the ATM

First, the system will ask the user to enter the account number and pin.



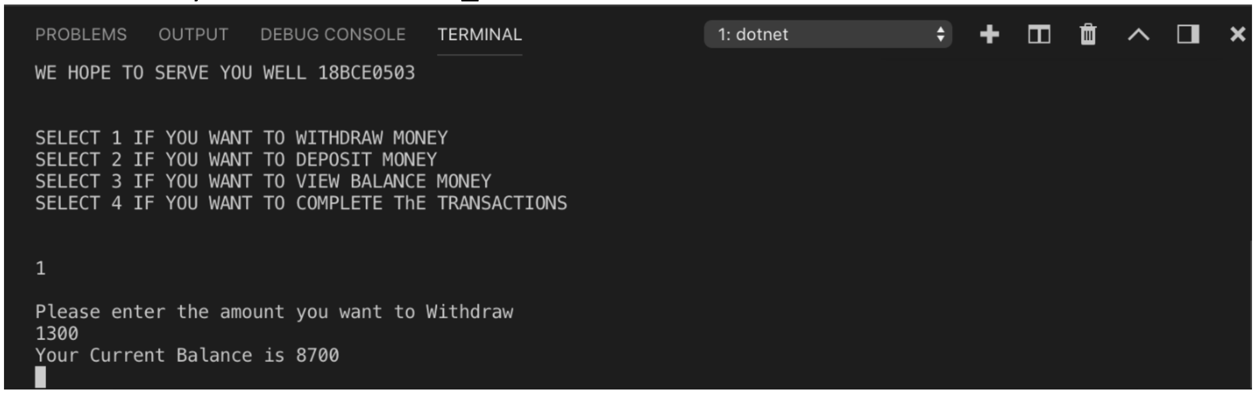
```
Please Enter your Account Number :
18BCE0503
Please Enter your  Pin:
1001
```

Various options on logging into the ATM



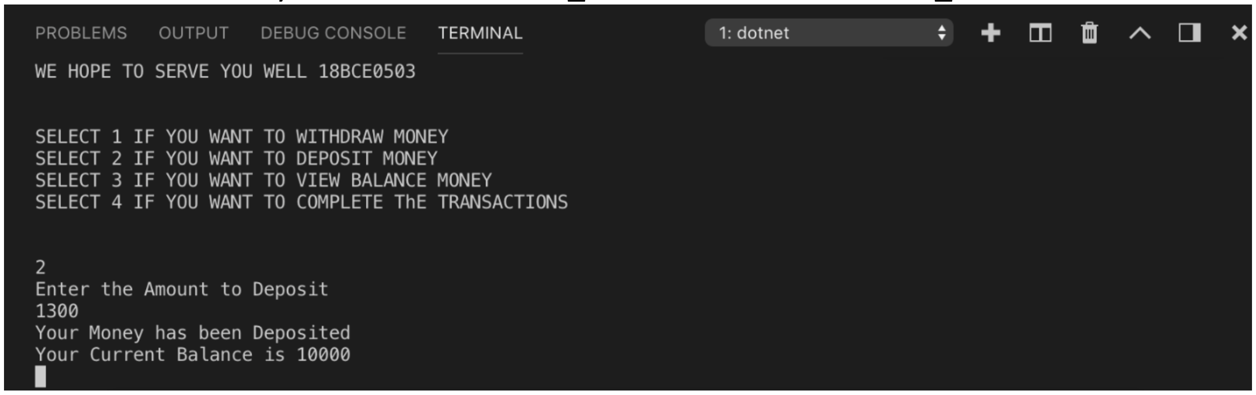
```
WE HOPE TO SERVE YOU WELL 18BCE0503


SELECT 1 IF YOU WANT TO WITHDRAW MONEY
SELECT 2 IF YOU WANT TO DEPOSIT MONEY
SELECT 3 IF YOU WANT TO VIEW BALANCE MONEY
SELECT 4 IF YOU WANT TO COMPLETE ThE TRANSACTIONS
```

1. **<u>The withdraw option</u>**
   This option allows the user to enter the amount to withdraw(). Then the code would run and deduct the amount from the account balance. But the user can withdraw only if his/her account_balance>1000.



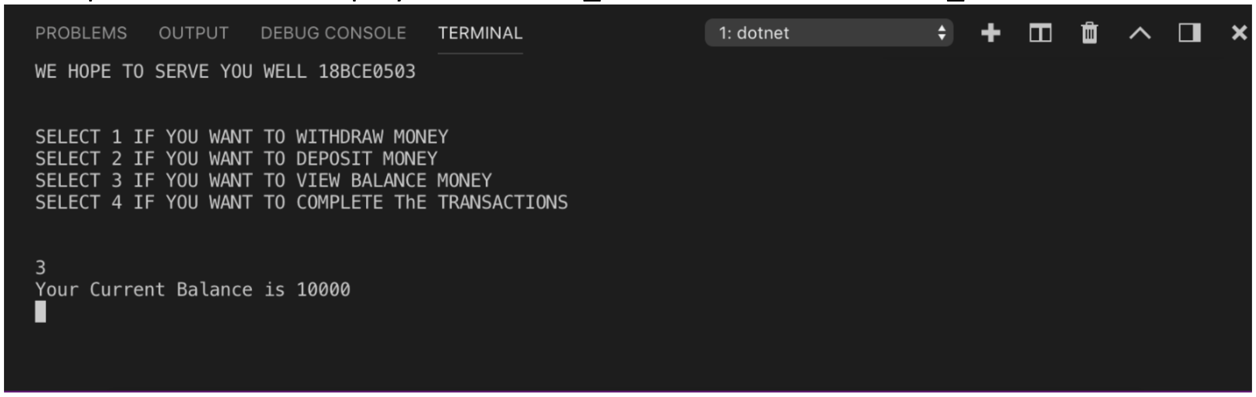
```
WE HOPE TO SERVE YOU WELL 18BCE0503


SELECT 1 IF YOU WANT TO WITHDRAW MONEY
SELECT 2 IF YOU WANT TO DEPOSIT MONEY
SELECT 3 IF YOU WANT TO VIEW BALANCE MONEY
SELECT 4 IF YOU WANT TO COMPLETE ThE TRANSACTIONS


1

Please enter the amount you want to Withdraw
1300
Your Current Balance is 8700
```

2. **<u>The deposit option</u>**
   This option allows the user to enter the amount to deposit(). Then the code would run and add the amount to the user's existing account_balance. Then the code would automatically inform the account_holder about his account_balance.



```
WE HOPE TO SERVE YOU WELL 18BCE0503


SELECT 1 IF YOU WANT TO WITHDRAW MONEY
SELECT 2 IF YOU WANT TO DEPOSIT MONEY
SELECT 3 IF YOU WANT TO VIEW BALANCE MONEY
SELECT 4 IF YOU WANT TO COMPLETE ThE TRANSACTIONS


2
Enter the Amount to Deposit
1300
Your Money has been Deposited
Your Current Balance is 10000
```

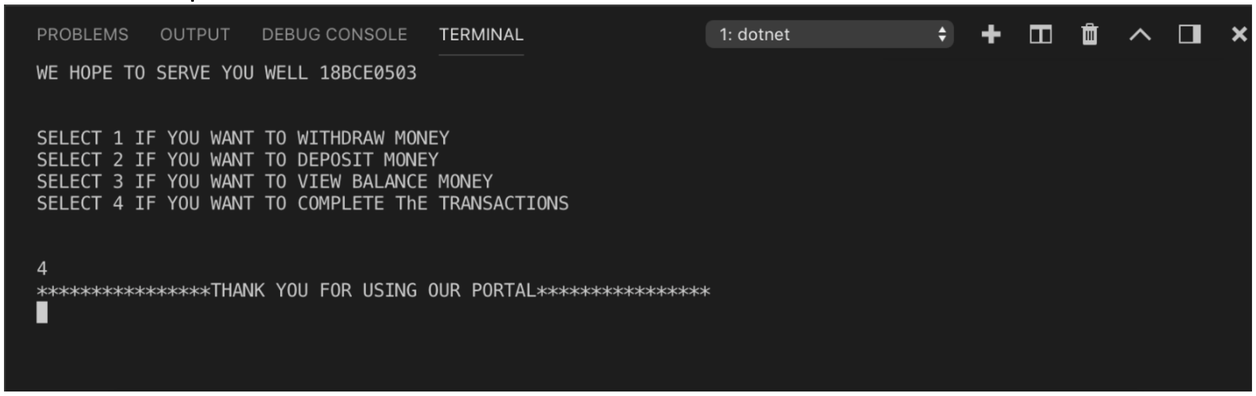
3. **<u>The view_balance option</u>**
   This option is used to display the account_holder's current account_balance



```
WE HOPE TO SERVE YOU WELL 18BCE0503


SELECT 1 IF YOU WANT TO WITHDRAW MONEY
SELECT 2 IF YOU WANT TO DEPOSIT MONEY
SELECT 3 IF YOU WANT TO VIEW BALANCE MONEY
SELECT 4 IF YOU WANT TO COMPLETE ThE TRANSACTIONS


3
Your Current Balance is 10000
```

4. The exit option

This option allows the user to shutdown the atm OS successfully, with the system displaying
****************THANK YOU FOR USING OUR PORTAL****************
For a certain period of time i.e. 10 seconds.



```
WE HOPE TO SERVE YOU WELL 18BCE0503


SELECT 1 IF YOU WANT TO WITHDRAW MONEY
SELECT 2 IF YOU WANT TO DEPOSIT MONEY
SELECT 3 IF YOU WANT TO VIEW BALANCE MONEY
SELECT 4 IF YOU WANT TO COMPLETE ThE TRANSACTIONS


4
****************THANK YOU FOR USING OUR PORTAL****************
```

CODE:

```
using System;

namespace consoleProject
{
  class Program
  {
    static void Main(string[] args)
    {
      string accountNumberAccepted;
      string[] accountNumber = {"18BCE0503", "18BCE2232", "18BCE2522","18BCI0059"};
      int[] pinArray = {1001,1002,1003,1004};
      int[] balance = {10000,20000,30000,3000};
      int flag=1;
      int z=-1;
      Console.Clear();
      Console.WriteLine("Please Enter your Account Number : ");



      while(true){

        accountNumberAccepted = Console.ReadLine();

        foreach (string x in accountNumber)
        {
          if(accountNumberAccepted.Contains(x))
          {
            Console.WriteLine("Please Enter your  Pin:");
            int pin = Convert.ToInt32(Console.ReadLine());

              if(pin == pinArray[z+1])
```

```
                {
                    flag=0;
                    Console.Clear();
                    break;
                }



        }
        z=z+1;
    }
    if(flag==1){
        Console.Clear();
        Console.WriteLine("\t\t Invalid User ID or Invalid Pin");
        Console.WriteLine("\t Please Enter A Valid Account Number");

    }
    else{
        break;
    }
    z=-1;
}

z=z+1;


while(true)
{
    Console.WriteLine("WE HOPE TO SERVE YOU WELL "+accountNumberAccepted);
    Console.WriteLine("\n\nSELECT 1 IF YOU WANT TO WITHDRAW MONEY ");
    Console.WriteLine("SELECT 2 IF YOU WANT TO DEPOSIT MONEY ");
    Console.WriteLine("SELECT 3 IF YOU WANT TO VIEW BALANCE MONEY ");
    Console.WriteLine("SELECT 4 IF YOU WANT TO COMPLETE ThE TRANSACTIONS\n\n");


    int choice=0;

    choice = Convert.ToInt32(Console.ReadLine());
    if(choice==1)
    {
        if(balance[z]<1000)
        {
            Console.WriteLine("Sorry, You do not have enough Balance to Withdraw");
        }
        else{
            Console.WriteLine("\nPlease enter the amount you want to Withdraw");
            int amountToWithdraw = Convert.ToInt32(Console.ReadLine());
            if((balance[z]-amountToWithdraw)>0)
            {
                balance[z]=balance[z]-amountToWithdraw;
                Console.WriteLine("Your Current Balance is "+balance[z]);
            }
            else
```

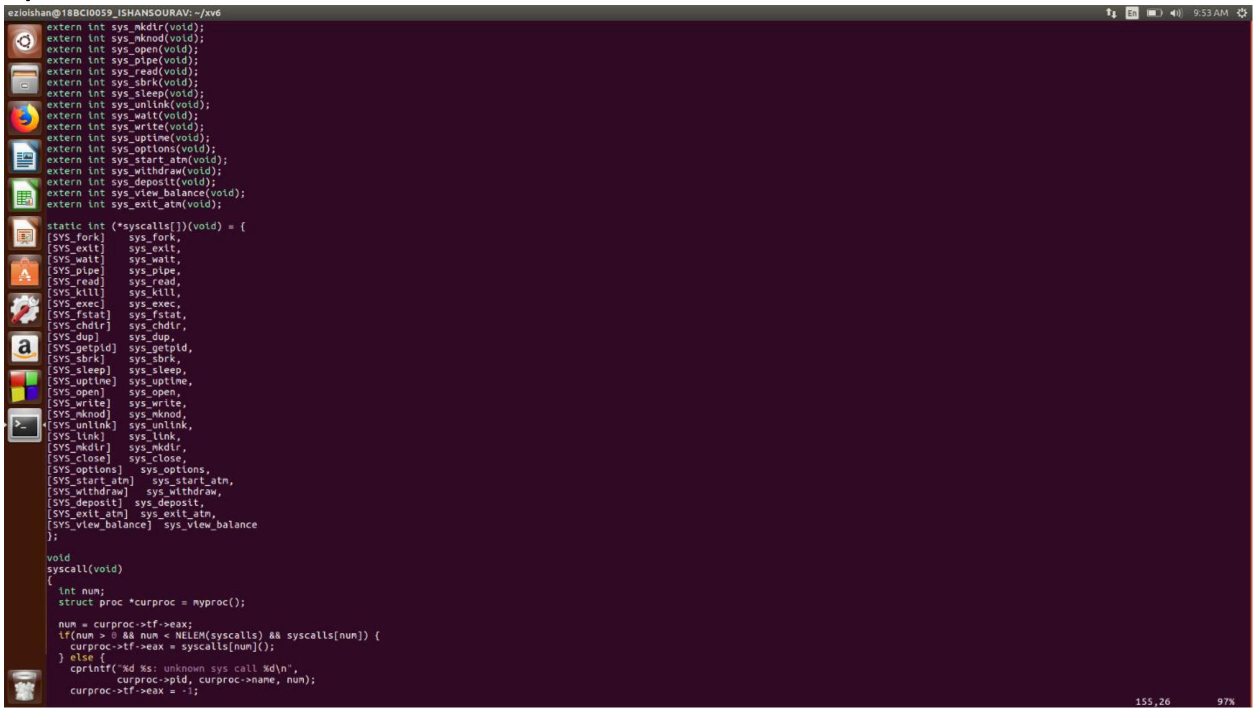
```
                        {
                            Console.WriteLine("Sorry, You do not have enough Balance to Withdraw");
                        }
                    }
                }
                else if(choice==2)
                {
                    Console.WriteLine("Enter the Amount to Deposit");
                    int amountToDeposit = Convert.ToInt32(Console.ReadLine());
                    balance[z]=balance[z]+amountToDeposit;
                    Console.WriteLine("Your Money has been Deposited");
                    Console.WriteLine("Your Current Balance is "+balance[z]);
                }
                else if(choice==3)
                {
                    Console.WriteLine("Your Current Balance is "+balance[z]);
                }
                else if(choice==4)
                {
                    Console.WriteLine("***************THANK YOU FOR USING OUR PORTAL****************");
                    System.Threading.Thread.Sleep(10000);
                    break;
                }
                Console.ReadKey();
                Console.Clear();
            }

        }
    }
}
```

HOW DID WE MAKE SYSCALLS?

The syscalls were made by making changes in the following files:

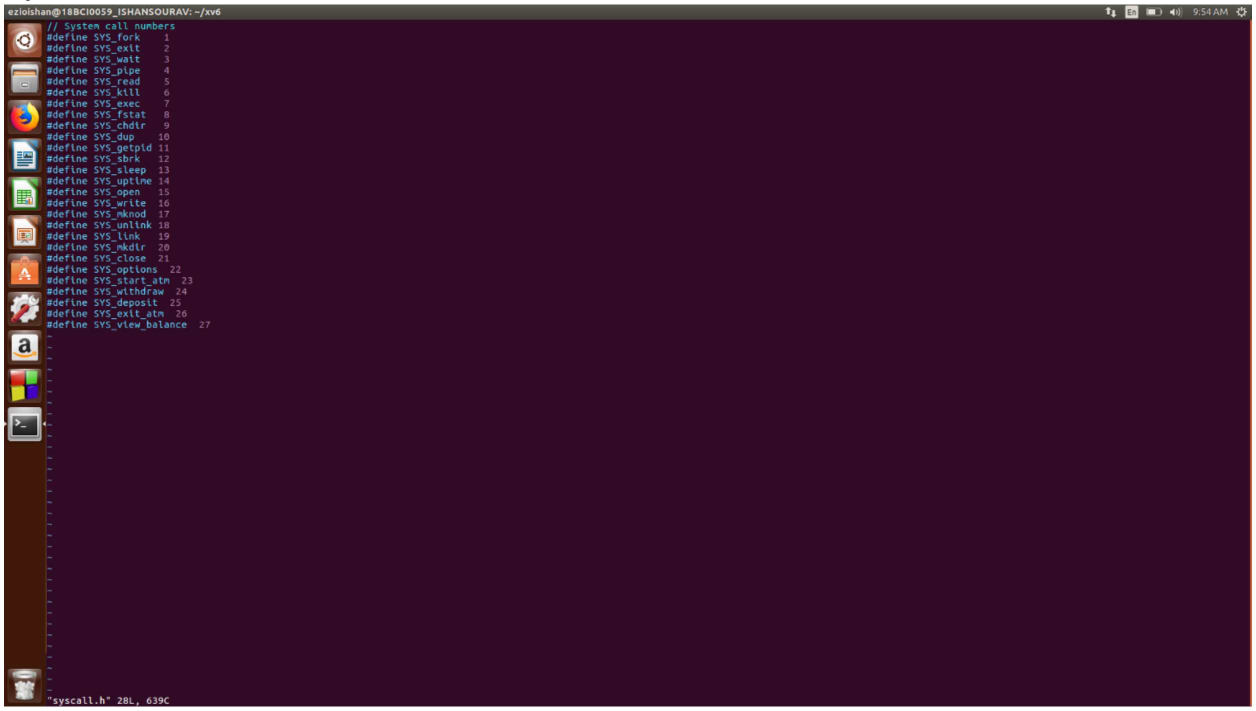
1. syscall.c
2. syscall.h
3. user.h
4. usys.S
5. sysproc.c

syscall.c



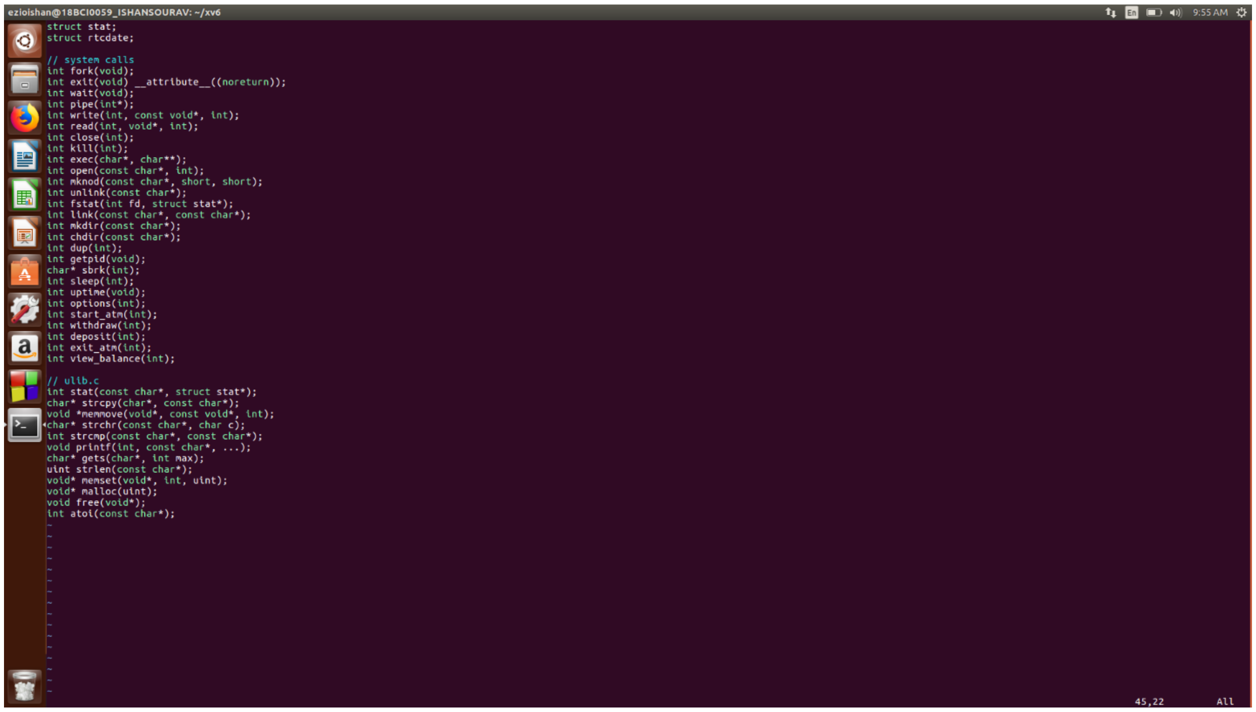
```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6                    9:53 AM
extern int sys_mkdir(void);
extern int sys_mknod(void);
extern int sys_open(void);
extern int sys_pipe(void);
extern int sys_read(void);
extern int sys_sbrk(void);
extern int sys_sleep(void);
extern int sys_unlink(void);
extern int sys_wait(void);
extern int sys_write(void);
extern int sys_uptime(void);
extern int sys_options(void);
extern int sys_start_atm(void);
extern int sys_withdraw(void);
extern int sys_deposit(void);
extern int sys_view_balance(void);
extern int sys_exit_atm(void);

static int (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_exit]    sys_exit,
[SYS_wait]    sys_wait,
[SYS_pipe]    sys_pipe,
[SYS_read]    sys_read,
[SYS_kill]    sys_kill,
[SYS_exec]    sys_exec,
[SYS_fstat]   sys_fstat,
[SYS_chdir]   sys_chdir,
[SYS_dup]     sys_dup,
[SYS_getpid]  sys_getpid,
[SYS_sbrk]    sys_sbrk,
[SYS_sleep]   sys_sleep,
[SYS_uptime]  sys_uptime,
[SYS_open]    sys_open,
[SYS_write]   sys_write,
[SYS_mknod]   sys_mknod,
[SYS_unlink]  sys_unlink,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
[SYS_options]   sys_options,
[SYS_start_atm]   sys_start_atm,
[SYS_withdraw]   sys_withdraw,
[SYS_deposit]  sys_deposit,
[SYS_exit_atm]  sys_exit_atm,
[SYS_view_balance]  sys_view_balance
};

void
syscall(void)
{
  int num;
  struct proc *curproc = myproc();

  num = curproc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
    curproc->tf->eax = syscalls[num]();
  } else {
    cprintf("%d %s: unknown sys call %d\n",
            curproc->pid, curproc->name, num);
    curproc->tf->eax = -1;
                                                              155,26        97%
```

syscall.h



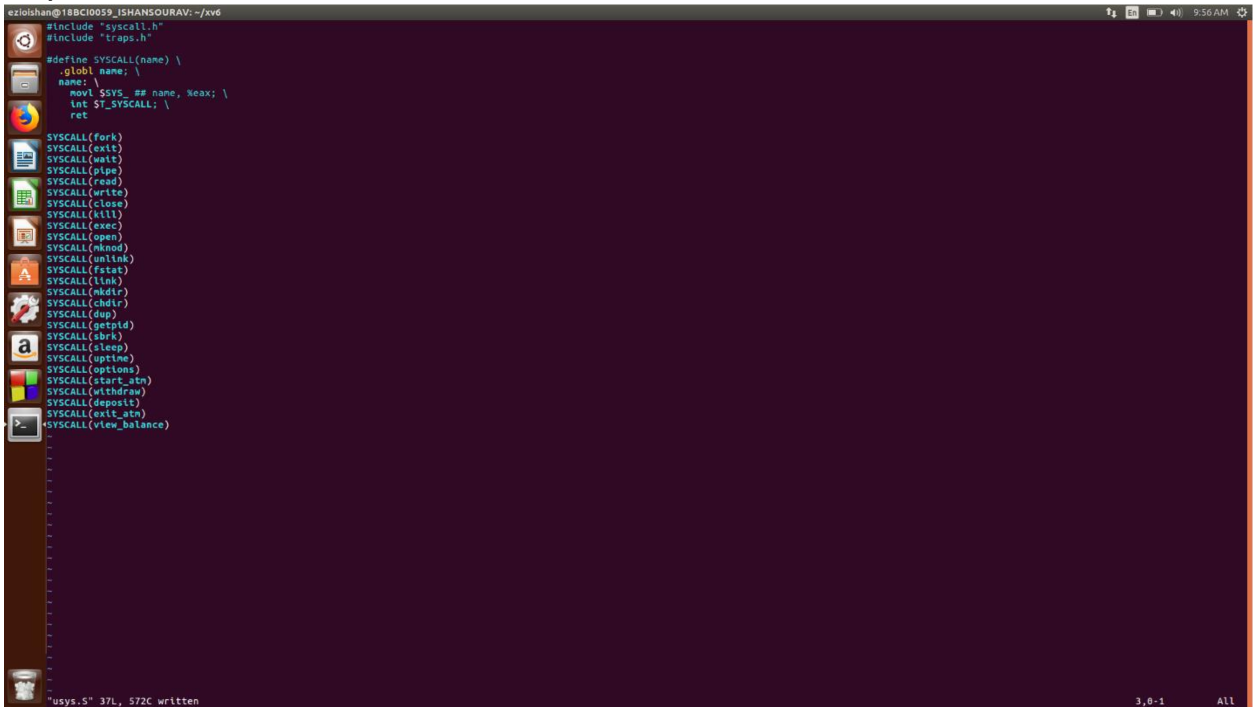
```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
// System call numbers
#define SYS_fork    1
#define SYS_exit    2
#define SYS_wait    3
#define SYS_pipe    4
#define SYS_read    5
#define SYS_kill    6
#define SYS_exec    7
#define SYS_fstat   8
#define SYS_chdir   9
#define SYS_dup    10
#define SYS_getpid 11
#define SYS_sbrk   12
#define SYS_sleep  13
#define SYS_uptime 14
#define SYS_open   15
#define SYS_write  16
#define SYS_mknod  17
#define SYS_unlink 18
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
#define SYS_options  22
#define SYS_start_atm  23
#define SYS_withdraw  24
#define SYS_deposit  25
#define SYS_exit_atm  26
#define SYS_view_balance  27
"syscall.h" 28L, 639C
```

user.h



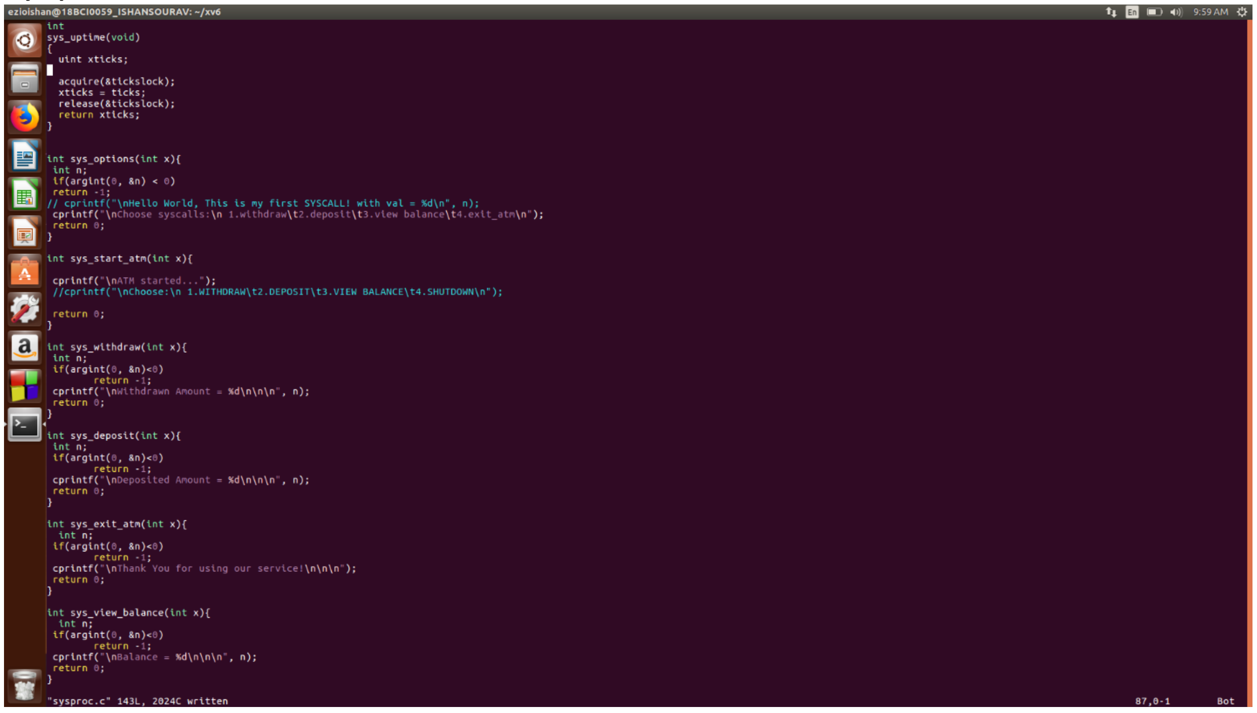
```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
struct stat;
struct rtcdate;

// system calls
int fork(void);
int exit(void) __attribute__((noreturn));
int wait(void);
int pipe(int*);
int write(int, const void*, int);
int read(int, void*, int);
int close(int);
int kill(int);
int exec(char*, char**);
int open(const char*, int);
int mknod(const char*, short, short);
int unlink(const char*);
int fstat(int fd, struct stat*);
int link(const char*, const char*);
int mkdir(const char*);
int chdir(const char*);
int dup(int);
int getpid(void);
char* sbrk(int);
int sleep(int);
int uptime(void);
int options(int);
int start_atm(int);
int withdraw(int);
int deposit(int);
int exit_atm(int);
int view_balance(int);

// ulib.c
int stat(const char*, struct stat*);
char* strcpy(char*, const char*);
void *memmove(void*, const void*, int);
char* strchr(const char*, char c);
int strcmp(const char*, const char*);
void printf(int, const char*, ...);
char* gets(char*, int max);
uint strlen(const char*);
void* memset(void*, int, uint);
void* malloc(uint);
void free(void*);
int atoi(const char*);
                                                            45,22         All
```

usys.s



```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
#include "syscall.h"
#include "traps.h"

#define SYSCALL(name) \
  .globl name; \
  name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret

SYSCALL(fork)
SYSCALL(exit)
SYSCALL(wait)
SYSCALL(pipe)
SYSCALL(read)
SYSCALL(write)
SYSCALL(close)
SYSCALL(kill)
SYSCALL(exec)
SYSCALL(open)
SYSCALL(mknod)
SYSCALL(unlink)
SYSCALL(fstat)
SYSCALL(link)
SYSCALL(mkdir)
SYSCALL(chdir)
SYSCALL(dup)
SYSCALL(getpid)
SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
SYSCALL(options)
SYSCALL(start_atm)
SYSCALL(withdraw)
SYSCALL(deposit)
SYSCALL(exit_atm)
SYSCALL(view_balance)
~
"usys.S" 37L, 572C written                                        3,0-1         All
```

sysproc.c



```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
int
sys_uptime(void)
{
  uint xticks;

  acquire(&tickslock);
  xticks = ticks;
  release(&tickslock);
  return xticks;
}


int sys_options(int x){
 int n;
 if(argint(0, &n) < 0)
 return -1;
// cprintf("\nHello World, This is my first SYSCALL! with val = %d\n", n);
 cprintf("\nChoose syscalls:\n 1.withdraw\t2.deposit\t3.view balance\t4.exit_atm\n");
 return 0;
}

int sys_start_atm(int x){

 cprintf("\nATM started...");
 //cprintf("\nChoose:\n 1.WITHDRAW\t2.DEPOSIT\t3.VIEW BALANCE\t4.SHUTDOWN\n");

 return 0;
}

int sys_withdraw(int x){
 int n;
 if(argint(0, &n)<0)
        return -1;
 cprintf("\nWithdrawn Amount = %d\n\n\n", n);
 return 0;
}

int sys_deposit(int x){
 int n;
 if(argint(0, &n)<0)
        return -1;
 cprintf("\nDeposited Amount = %d\n\n\n", n);
 return 0;
}

int sys_exit_atm(int x){
  int n;
 if(argint(0, &n)<0)
        return -1;
 cprintf("\nThank You for using our service!\n\n\n");
 return 0;
}

int sys_view_balance(int x){
  int n;
 if(argint(0, &n)<0)
        return -1;
 cprintf("\nBalance = %d\n\n\n", n);
 return 0;
}
"sysproc.c" 143L, 2024C written                                   87,0-1        Bot
```

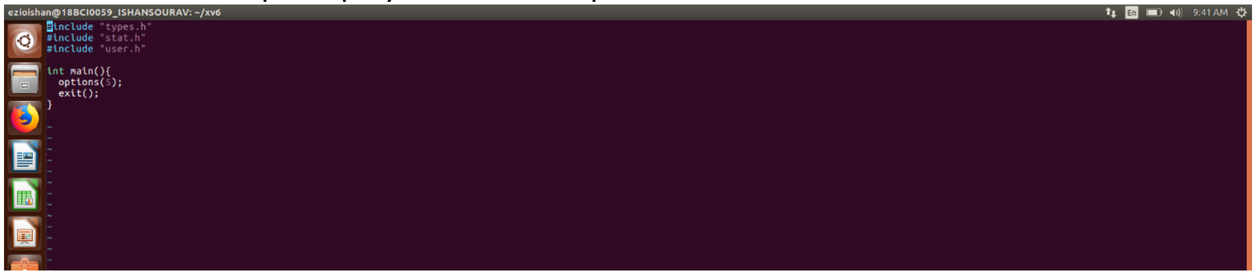
Module Implemented for the final review

For the final review, we have rewamped our complete program, and shifted to the xv6 OS. In xv6, we created an ATM facility, which offered the following features:

1. **start_atm** to start our ATM application.
2. **withdraw** to withdraw a certain amount
3. **deposit** to deposit a certain amount
4. **exit_atm** to exit the atm
5. **options** to view the various ATM options i.e. syscalls available for the ATM transaction
6. **view_balance** to view the balance at any given point.
7. Also **Authorization** facility but it did not work successfully

Options

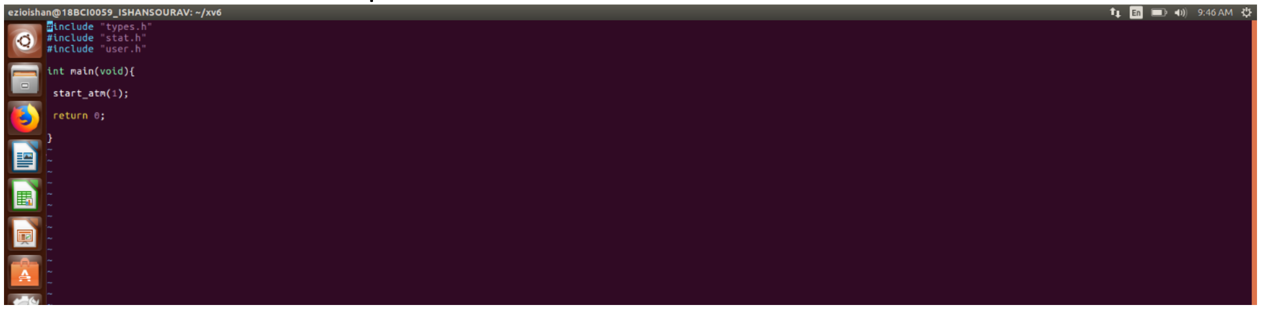
This command helps display the various options that our ATM offers to the user.



```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
#include "types.h"
#include "stat.h"
#include "user.h"

int main(){
  options(5);
  exit();
}
```

Start_atm

This system call was used to start the atm and was called from inside the other system calls like withdraw and deposit.

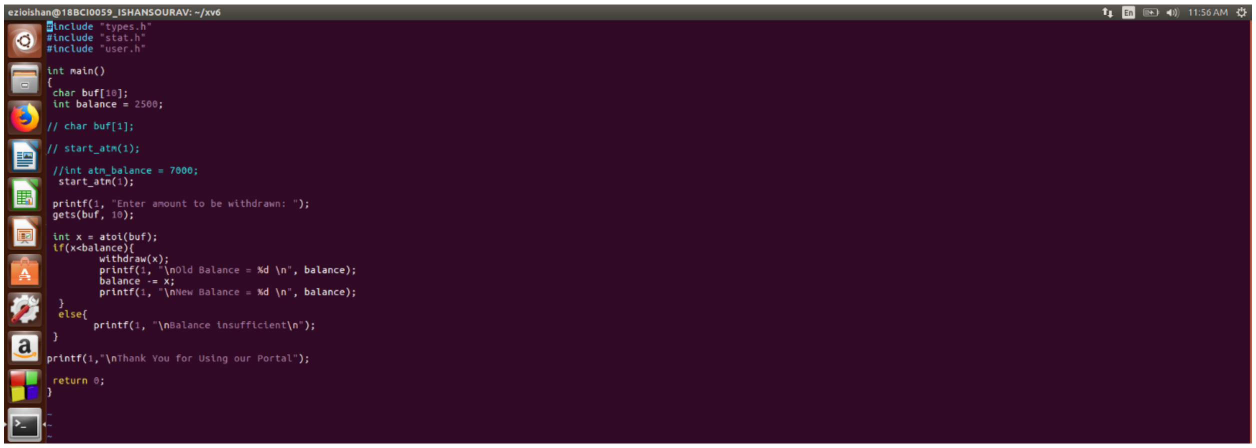


```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
#include "types.h"
#include "stat.h"
#include "user.h"

int main(void){

 start_atm(1);

 return 0;

}
```

Withdraw

This syscall allows the user to enter the amount that he/she needs to withdraw from their bank account.

After asking for the amount to withdraw, the system call shall process the data and display the Old balance and the New balance in the console.

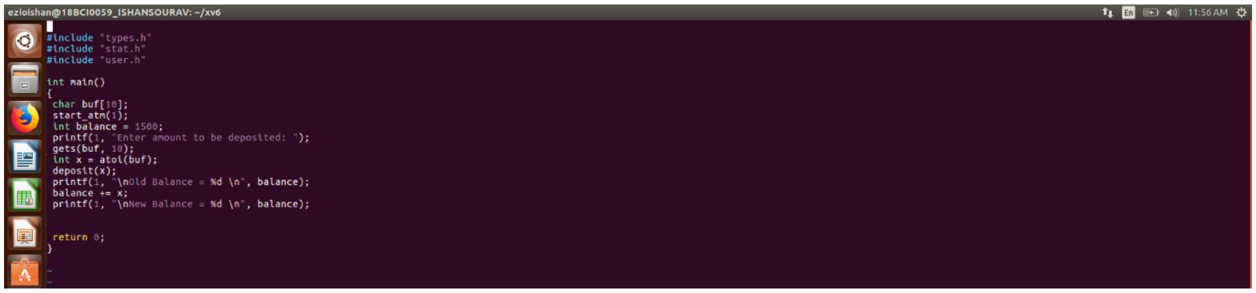


```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
#include "types.h"
#include "stat.h"
#include "user.h"

int main()
{
 char buf[10];
 int balance = 2500;

// char buf[1];

// start_atm(1);

 //int atm_balance = 7000;
  start_atm(1);

 printf(1, "Enter amount to be withdrawn: ");
 gets(buf, 10);

 int x = atoi(buf);
 if(x<balance){
         withdraw(x);
         printf(1, "\nOld Balance = %d \n", balance);
         balance -= x;
         printf(1, "\nNew Balance = %d \n", balance);
  }
  else{
        printf(1, "\nBalance insufficient\n");
 }

printf(1,"\nThank You for Using our Portal");

 return 0;
}
~
~
~
```

Deposit

This system call allows the user to enter the amount that he/she needs to deposit into their bank account.

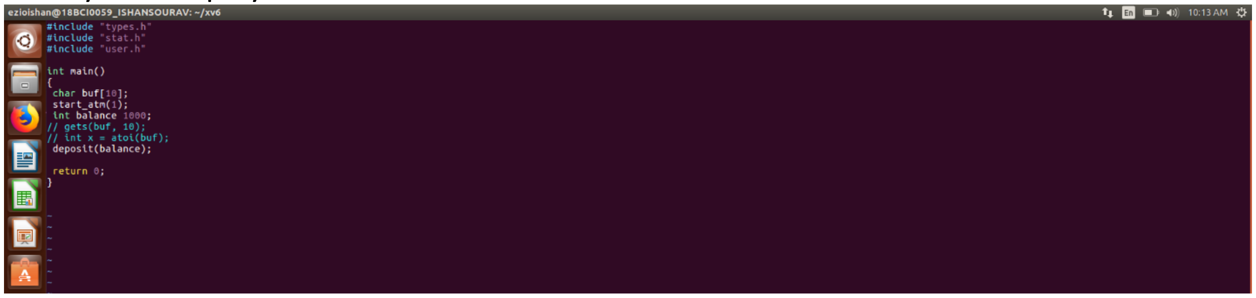
After asking for the amount to deposit, the system call shall process the data and display the Old balance and the New balance in the console.



```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6

#include "types.h"
#include "stat.h"
#include "user.h"

int main()
{
 char buf[10];
 start_atm(1);
 int balance = 1500;
 printf(1, "Enter amount to be deposited: ");
 gets(buf, 10);
 int x = atoi(buf);
 deposit(x);
 printf(1, "\nOld Balance = %d \n", balance);
 balance += x;
 printf(1, "\nNew Balance = %d \n", balance);


 return 0;
}
~
~
```

View_balance

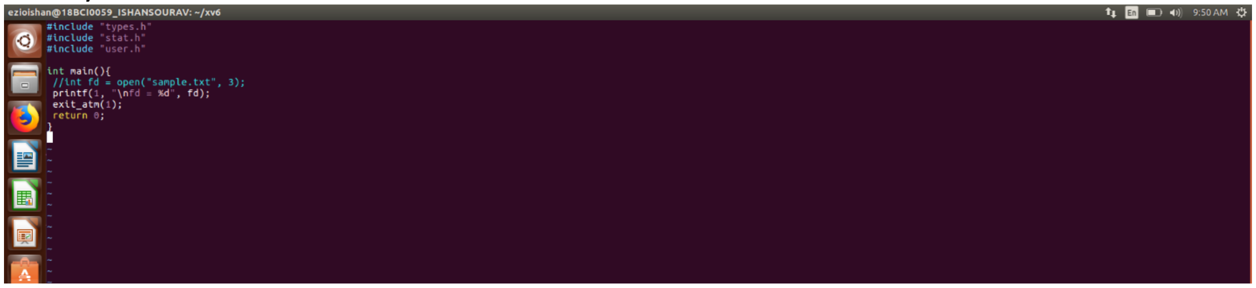
This syscall displays to the user their current account balance.



```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
#include "types.h"
#include "stat.h"
#include "user.h"

int main()
{
 char buf[10];
 start_atm(1);
 int balance 1000;
// gets(buf, 10);
// int x = atoi(buf);
 deposit(balance);

 return 0;
}



~
~
~
~
~
~
~
```

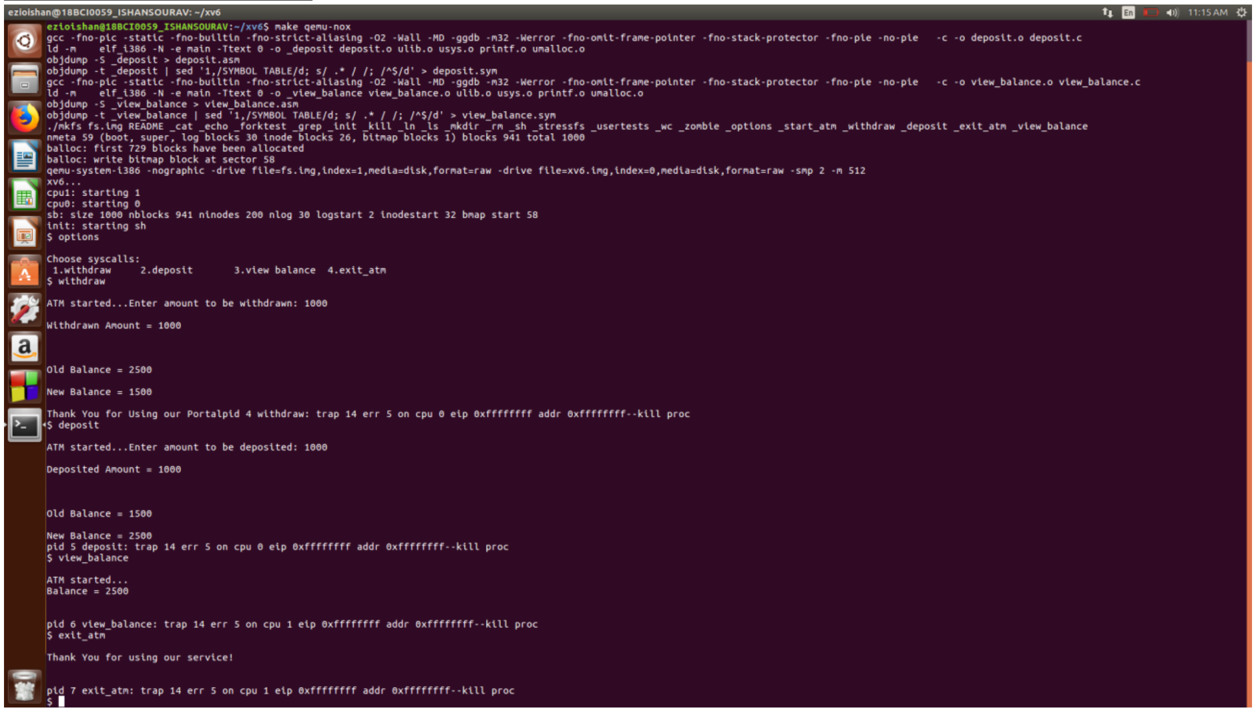
Exit_atm

This syscall is used to exit or shutdown the atm.



```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
#include "types.h"
#include "stat.h"
#include "user.h"

int main(){
 //int fd = open("sample.txt", 3);
 printf(1, "\nfd = %d", fd);
 exit_atm(1);
 return 0;
}
```

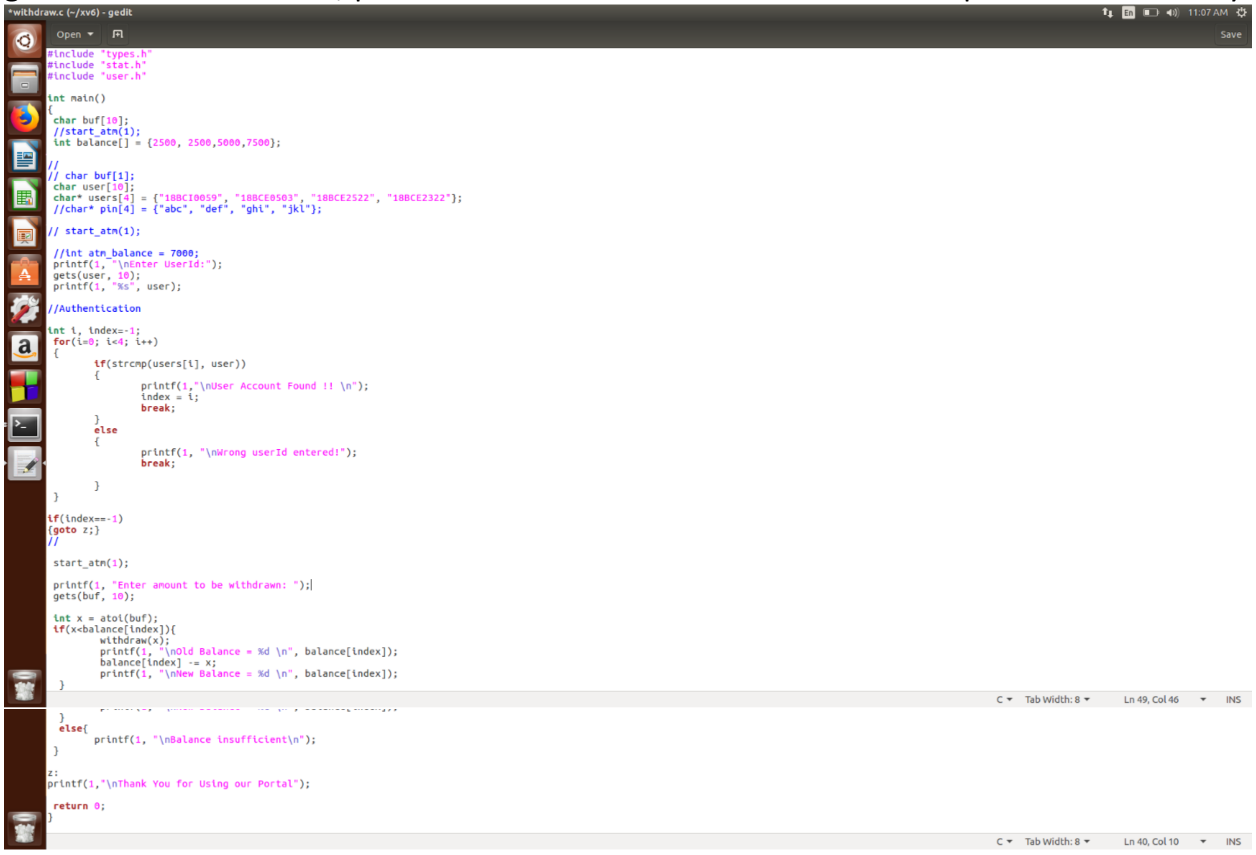
FINAL OUTPUT:



```
ezioishan@18BCI0059_ISHANSOURAV: ~/xv6
ezioishan@18BCI0059_ISHANSOURAV:~/xv6$ make qemu-nox
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie   -c -o deposit.o deposit.c
ld -m    elf_i386 -N -e main -Ttext 0 -o _deposit deposit.o ulib.o usys.o printf.o umalloc.o
objdump -S _deposit > deposit.asm
objdump -t _deposit | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > deposit.sym
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie   -c -o view_balance.o view_balance.c
ld -m    elf_i386 -N -e main -Ttext 0 -o _view_balance view_balance.o ulib.o usys.o printf.o umalloc.o
objdump -S _view_balance > view_balance.asm
objdump -t _view_balance | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > view_balance.sym
./mkfs fs.img README _cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm _sh _stressfs _usertests _wc _zombie _options _start_atm _withdraw _deposit _exit_atm _view_balance
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 941 total 1000
balloc: first 729 blocks have been allocated
balloc: write bitmap block at sector 58
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ options

Choose syscalls:
 1.withdraw     2.deposit       3.view balance  4.exit_atm
$ withdraw

ATM started...Enter amount to be withdrawn: 1000

Withdrawn Amount = 1000



Old Balance = 2500

New Balance = 1500

Thank You for Using our Portalpid 4 withdraw: trap 14 err 5 on cpu 0 eip 0xffffffff addr 0xffffffff--kill proc
$ deposit

ATM started...Enter amount to be deposited: 1000

Deposited Amount = 1000



Old Balance = 1500

New Balance = 2500
pid 5 deposit: trap 14 err 5 on cpu 0 eip 0xffffffff addr 0xffffffff--kill proc
$ view_balance

ATM started...
Balance = 2500


pid 6 view_balance: trap 14 err 5 on cpu 1 eip 0xffffffff addr 0xffffffff--kill proc
$ exit_atm

Thank You for using our service!


pid 7 exit_atm: trap 14 err 5 on cpu 1 eip 0xffffffff addr 0xffffffff--kill proc
$ 
```

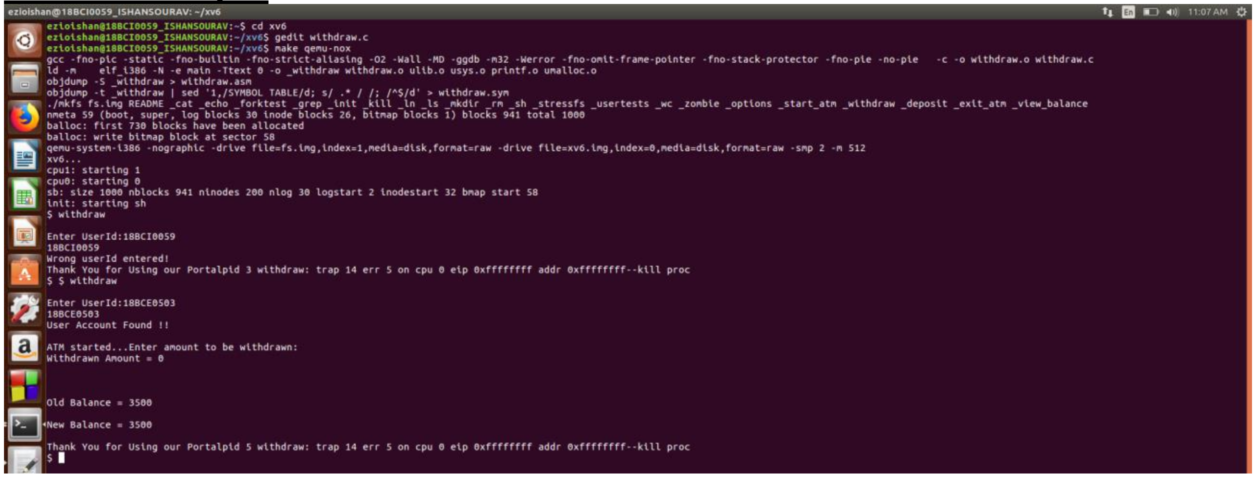
Authorization

Note: We tried to implement an Authorization functionality, but due to the inability to take multiple inputs from the user one after the other, we were not able to successfully get the values of userid, pin and the amount to be withdrawn or deposited successfully.



```c
#include "types.h"
#include "stat.h"
#include "user.h"

int main()
{
 char buf[10];
 //start_atm(1);
 int balance[] = {2500, 2500,5000,7500};

//
// char buf[1];
 char user[10];
 char* users[4] = {"18BCI0059", "18BCE0503", "18BCE2522", "18BCE2322"};
 //char* pin[4] = {"abc", "def", "ghi", "jkl"};

// start_atm(1);

 //int atm_balance = 7000;
 printf(1, "\nEnter UserId:");
 gets(user, 10);
 printf(1, "%s", user);

//Authentication

int i, index=-1;
 for(i=0; i<4; i++)
 {
        if(strcmp(users[i], user))
        {
                printf(1,"\nUser Account Found !! \n");
                index = i;
                break;
        }
        else
        {
                printf(1, "\nWrong userId entered!");
                break;

        }
 }

if(index==-1)
{goto z;}
//

 start_atm(1);

 printf(1, "Enter amount to be withdrawn: ");
 gets(buf, 10);

 int x = atoi(buf);
 if(x<balance[index]){
         withdraw(x);
         printf(1, "\nOld Balance = %d \n", balance[index]);
         balance[index] -= x;
         printf(1, "\nNew Balance = %d \n", balance[index]);
  }
  else{
        printf(1, "\nBalance insufficient\n");
 }

z:
printf(1,"\nThank You for Using our Portal");

 return 0;
}
```

Incorrect output



```
ezioishan@18BCI0059_ISHANSOURAV:~$ cd xv6
ezioishan@18BCI0059_ISHANSOURAV:~/xv6$ gedit withdraw.c
ezioishan@18BCI0059_ISHANSOURAV:~/xv6$ make qemu-nox
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie   -c -o withdraw.o withdraw.c
ld -m    elf_i386 -N -e main -Ttext 0 -o _withdraw withdraw.o ulib.o usys.o printf.o umalloc.o
objdump -S _withdraw > withdraw.asm
objdump -t _withdraw | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > withdraw.sym
./mkfs fs.img README _cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm _sh _stressfs _usertests _wc _zombie _options _start_atm _withdraw _deposit _exit_atm _view_balance
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 941 total 1000
balloc: first 730 blocks have been allocated
balloc: write bitmap block at sector 58
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ withdraw

Enter UserId:18BCI0059
18BCI0059
Wrong userId entered!
Thank You for Using our Portalpid 3 withdraw: trap 14 err 5 on cpu 0 eip 0xffffffff addr 0xffffffff--kill proc
$ $ withdraw

Enter UserId:18BCE0503
18BCE0503
User Account Found !!

ATM started...Enter amount to be withdrawn:
Withdrawn Amount = 0



Old Balance = 3500

New Balance = 3500

Thank You for Using our Portalpid 5 withdraw: trap 14 err 5 on cpu 0 eip 0xffffffff addr 0xffffffff--kill proc
$ 
```

ISSUES FACED:

We faced a lot of issues while making the final project. The main issues that we faced were Software issues and OS issues.

We started with the aim of making a Project using the COSMOS software, which is supposedly capable for letting the user to make his/her Mini OS. We tried installing the software in various number of devices, but it did not work. After a lot of efforts and attempts we found through a source online that the COSMOS software, which is available on GITHUB, has been depreciated, hence it didn't work. Then we were suggested by the faculty to use Kolibri OS, but we failed to make a project on that.

After that, we proposed that we make an ATM in xv6, which provides a real time user to Withdraw, Deposit and View his/her account balance using syscalls. We also decided to add a start_atm syscall which will be used to initiate the ATM. We installed xv6 software and started working on it, made a few syscalls, but then we lost all our progress, since the xv6 installed on the device crashed and had to be reinstalled. Hence, then we installed the xv6 software back into the device and made the Syscalls all over again once more. We faced a few issues while making the syscalls like the we were not able to use the scanf() function in the '.c' file which was used to make the syscall, we researched on the internet why it was not working and found that we had to use the gets() function as there was no provision for scanf() in xv6. Due to no proper documentation, we were not able to implement the functions used to read data from a file, also to read data from a file. Hence, we were not able to make the data to be stored permanently. Another reason for not being able to store the data in a file permanently is that there is no provision in xv6 to save a file, and the data gets restored once it is closed.

REFERENCES:

1. https://websiteforstudents.com/installing-virtualbox-windows-10/
2. https://www.assistedcoding.eu/2017/11/06/install-vx6-operating-system/
3. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/hello-world-your-first-program?tabs=macos
4. https://medium.com/@silvamatteus/adding-new-system-calls-to-xv6-217b7daefbe1
5. https://stackoverflow.com/questions/8021774/how-do-i-add-a-system-call-utility-in-xv6
6. https://01siddharth.blogspot.com/2018/04/adding-system-call-in-xv6-os.html
7. https://www.geeksforgeeks.org/introduction-of-system-call/

Inference from the project work:

During the duration of the Project, we learnt a lot of things like How to install a Virtual Machine, How to install an OS on a VirtualMachine. On the journey, we went through a lot of challenges like software failure, and software depreciation etc., but we learnt how to tackle them. We developed a skill to code in c-sharp programming language. We got to know how to install xv6 in Ubuntu. We learnt how to make system calls in xv6.

From the project we infer that xv6, a learning OS, can be used to make syscalls, which can help the user to directly access commands from the terminal by inserting the command as a syscall in xv6. These commands can then be used anytime to perform the specific purpose that they serve and were made for, like: we made the start_atm sycall to start our ATM application, which indeed had loads of syscalls within it.