Operating Systems

Bank OS

Team 2 :

Sripranav Mannepalli	cs18b036
Maheswara Reddy	cs18b032
Chirag gupta	cs18b006
A V S Hrudai	cs18b001

Introduction

The Operating System we developed can be used by the Bank's cashier, manager and account holder. We made it in a way that it covers all the basic tasks required by the banks.
OS concepts that we covered in this project:

- Process Scheduling.
- InterProcess Communication.
  - Message Passing.
  - Shared memory.
- Process Synchronization.
- High level memory management.
- Deadlock handling.

Features :

Our Operating System has the following features:

- Creating a bank account. We can create as many bank accounts as we need.
- Login to your Bank account.
- Deposit money.
- Withdraw money.
- Transfer money to a different account.
- Advertisements of the bank.
- Show all accounts.
- Add review.
- Add comment.
- Security Feature (Password Authorization)
- Screen device (To display output)

NOTE : *We are using the Backend Screen (of VirtualBox) as our SCREEN. We use both "SCREEN' and shell to show outputs. The Inputs are taken from the shell.*

Revised bottom-up details :

OS Concept	**Files involved**
Input/Output , Device Management.	We take input from the user through the shell. Both the shell and a screen, which is a Video Graphic Array(vga) is used to display output.
Process Scheduling.	Our OS allows the bank to display ads in regular intervals of time on the Screen.We have a deadlock process that checks for deadlocks, ring process that acts as an error alarm running continuously. We use priority scheduling to schedule between different processes.
Message Passing(IPC).	We created a process called Ring to display error messages. If a user entered an incorrect password or the likes, a message is passed from user process to this Ring process and the error message is displayed. We established a message passing route between the user process and deposit, withdraw, transfer and other processes. We pass the amount as the message from the user process to other processes and these perform their tasks using this message.
Shared memory(IPC).	We also used shared memory between different processes through buffer pools. Processes like deposit, withdraw and transfer can access the amount of money in an account and change it.
Process Synchronization.	We used Semaphores to prevent context switching from happening when we inside the deposit, withdraw or transfer processes.
High level memory management.	Buffer pools are used to store all the details of the user accounts.
Deadlock handling	We have created a deadlock.c file in which we have used an algorithm to find if a deadlock situation occurs and a deadlock recovery algorithm to recover from this situation. **Example situation**:There are two processes: comment and review. The resources are keyboard and screen. Both the processes need both the resources.First, the comment process acquires the keyboard and the review process acquires the screen.To get out of this deadlock, we kill a process, in this case the comment process.

Revised top- down details :

Features	OS concept involved
Creating a bank account	I/O & D management, memory management, message passing
Login to your Bank account	I/O & D management, memory management, message passing.
Deposit money.	Message passing, shared memory, process synchronization, memory management.
Withdraw money	Message passing, shared memory, process synchronization, memory management.
Transfer money to a different account	Message passing, shared memory, process synchronization, memory management.
Advertisements of the bank	Process scheduling, I/O & D management and deadlock handling(detection and recovery).
Show all accounts.	I/O & D management and memory management.
Add review	I/O & D management, memory management and deadlock handling(detection and recovery).
Add comment	I/O & D management, memory management and deadlock handling(detection and recovery).

* I/O & D = Input, Output and Device.

Implementation :

S.No	Features	Xinu files modified	Xinu files used	Xinu files added
1	Creating a bank account	prototypes.h, initialize.c, bufpool.h	kprintf.c, fgetc.c, send.c, exit.c,	xsh_user.c, xsh_bank.c, Ring.c
2	Login to your Bank account	prototypes.h, initialize.c	kprintf.c, send.c, fgetc.c, exit.c	xsh_user.c, Ring.c
3	Deposit money	prototypes.h, initialize.c	semcreate.c, wait.c, signal.c, kprintf.c, create.c, fputc.c, send.c, resume.c	deposit.c, xsh_user.c, Ring.c
4	Withdraw money	prototypes.h, initialize.c	semcreate.c, wait.c, signal.c, kprintf.c, create.c, fputc.c, send.c, resume.c	withdraw.c, xsh_user.c, Ring.c
5	Transfer money to a different account	prototypes.h, initialize.c	semcreate.c, wait.c, signal.c, kprintf.c, create.c, resume.c, send.c, fputc.c, sleep.c	transfer.c, xsh_user.c, Ring.c
6	Advertisements of the bank	prototypes.h, initialize.c	wait.c, fputc.c, resched.c, create.c, resume.c, signal.c, sleep.c	ads.c, Ring.c
7	Show all accounts.	prototypes.h, initialize.c	fputc.c	xsh_bank.c, Ring.c
8	Add review	prototypes.h, initialize.c	kprint.c, create.c, resume.c ,fputc.c,fgetc.c, wait.c, clearScreen.c, sleep.c, wait.c, kill.c, resume.c, signal.c	xsh_reviewcomme nt.c, review_comment.c, Ring.c
9	Add comment	prototypes.h,	kprint.c, create.c,	xsh_reviewcomme

		initialize.c	resume.c ,fputc.c,fgetc.c, wait.c, clearScreen.c, sleep.c, wait.c, kill.c, resume.c, signal.c	nt.c, review_comment.c, Ring.c
10.	Initial Login to our OS (As an additional Security Feature)	main.c	wait.c, signal.c, kill.c, kprintf.c	
11	Added SCREEN(vga)	conf.c configuration conf.h compile/makefile	xinu.h	device/vga/vga.c (Added vga directory in device and added vga.c file)

Our Submission Files :

TEAM2

unmodified-xinu	Original XINU
xinu	Our Bank OS
Makefile	Makefile to run our BANK OS from the ubuntu host machine

Commands to run the Bank OS from the host machine (Linux):

1) Put the directory in the home/root folder (~).
2) 'make dev-start-headless' to start the development system in headless mode.
3) 'make' or 'make execute' to run the OS. Enter 'xinurocks' when required as this command also calls the backend system in UI Mode as our SCREEN (VGA) is shown on the Virtual Box Backend UI.

A detailed explanation of the features in our Bank OS :

Creating a Screen(VGA):

In configuration add a device type to the file: vga

vga support initialization and output according to the device types, define the device as SCREEN

For display devices, the current cursor position needs to be maintained. Since there is only one display, global variables are used directly here to store, without using the device data structure.

Set the current cursor position by the following function. This function will not only modify the saved cursor position, but also the cursor position.

When outputting characters, it is judged if special characters (newline, tab, backspace) are to be output, then special operations are performed, otherwise the characters are written and shifted

Move the cursor to the next position:

When moving to the next line (because of a newline or full line), determine whether you need to scroll.

The way to scroll the screen is to start from the first one. The beginning of the line, until the last line of data, write to the first 0 In the buffer at the beginning of the line, and empty the original at the end

Buffer data for one line:

Creating a bank account :

The command is used to create a bank account.

The shell command is ‘addaccount’ .

The user is asked to provide a username and password. Then the OS assigns an account number to the user.

In case the user does not follow the rules, an error message is generated by passing a message to the always running ‘ring’ process(This ring process prints the beeps that are received through message).

Login to your bank account :

If the user wants to access his bank account,he needs to login in their bank account first.

The shell command is ‘login’.

The user needs to provide the account number and password to login to his account.

After logging to his account the user can deposit, withdraw and transfer the money. He can also give comments and reviews.

In case the user enters the wrong account number or the user enters the wrong password , an error message is generated by passing a message to the always running ‘ring’ process(This ring process prints “beep” a number of times that are received through message).

Deposit Money :

After logging into his account the user can deposit the money he wants into his account. No need to provide any password.

The shell command is ‘deposit’.

A message is generated by the user process and this message is received by the deposit process. This message is the amount of money the user wants to deposit. The deposit process deposits the money into the account.

Withdraw Money :

The user needs to have some initial amount of cash in his account to withdraw money. He cannot withdraw more than the amount he has in his account. He needs to maintain at least 1Rupee in his account.

The shell command is ‘withdraw’.

A message is generated by the user process and this message is received by the withdraw process. This message is the amount of money the user wants to withdraw. The withdrawal process withdraws the money from the account.

Transfer Money:

The user needs to have some initial amount of cash in his account to transfer the money to another account. He cannot transfer more than the amount he has in his account. The user needs to provide the account number of the recipient.

The shell command is ‘transfer’.

A message is generated by the user process and this message is received by the transfer process. This message is the amount of money the user wants to transfer. The transfer process transfers the money from the user's account to the recipient's account.

Advertisements:

An Advertisement process is always running in the background.

Advertisement process waits on the screen semaphore.

Advertisement process then shows the advertisement on screen and then signals back the screen semaphore and then sleeps for 30 seconds.

Show all accounts :

This process accesses the memory (bufferpool) and iterates over it to show all the account holders' names, account numbers and balances.

Initial Login:

Whenever our XINU boots up, we ask the user for the password (which is **team2rocks**).

The user can enter our OS only if he gives the correct password. We added this to increase security of our OS.

Add review :

This process is activated when a logged in user wants to give a review to the bank.

The command is 'review'.

When we activate this command,
Initially, wait for screen semaphore.

Then we display text on screen

We sleep the process for 7 seconds to give time to think for the user and then activate the input.

Then we wait for keyboard semaphore.

Then we take input from the user.

After the input process is completed, we show the entered review on screen and we finish the process.

Add comment :

This process is activated when a logged in user wants to give a comment to the bank.

The command is ‘comment’.

When we activate this command,
Initially we wait for keyboard semaphore(keyboard resource).

Then we take user input.

Then we wait for the screen semaphore(screen resource).

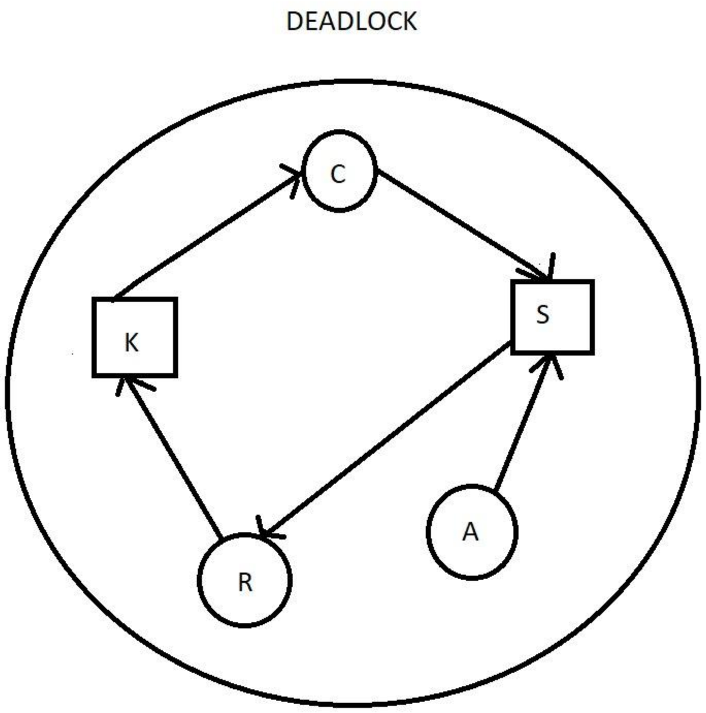
Then we display text on screen along with the user’s comment

Deadlock management:

Whenever we enter the shell command ‘comment’ after entering the shell command ‘review’ , (before asking input in review , that is within 7 seconds after giving the ‘review’ input), There occurs a deadlock situation, which is also shown pictorially in the below image. We are using a deadlock detection algorithm to detect deadlocks and we are doing deadlock recovery by killing the processes involved in the deadlock till we get a deadlock free state.

K: Keyboard Resource S: Screen Resource

A: Advertisement Process R: Review Process C: Comment Process



We can see a cycle in this graph. A cycle in a single instance resource graph means a deadlock occurred. Here the Review process has a resource - screen and wants the resource - keyboard. Comment process has the resource - keyboard and wants the resource - screen.

We recover from this deadlock by killing processes involved in deadlock till there is no deadlock.

Contributions:

We four members contributed equality in doing the OS project. We all met regularly and discussed what needs to be done on a regular basis and all of us implemented every OS concept.