**שמות המגישים:**

**דימה ספקטור – 319286738**

**אור בן-סימון – 305605743**

**תכנית:**

#include <time.h>

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <pthread.h>

#include <fcntl.h>

#include <errno.h>

#include <unistd.h>

#include <string.h>

#include <semaphore.h>

#include <assert.h>

#define BUFFER\_SIZE 8192

typedef struct atm {

char nameFile[400];

int numATM;

char numATMChar[400];

char \*buffer;

int ret\_in;

int file;

int lengthBuffer;

struct ATM \*next;

}\*ATM;

typedef struct account {

sem\_t \*mutex;

int numAccount;

char numAccountChar[400];

int password;

char passwordChar[5];

int balance;

char balanceChar[400];

struct account \*next;

struct account \*prev;

}\*Account;

typedef struct bank {

Account bankAccounts;

int numOfAccounts;

}\*Bank;

Bank bankAllAccounts;

int logText;

int count;

void \*checkATM(void \*ATM);//Create ATM details from file input

void \*print\_bank(void \*Bank);//Print bank accounts details

void \*commandATM(void \*ATM);//All the commands of ATM

void createNewAccount(char\* fileInput, ATM itemAtm);//Create new account

void withdrewFromAccount(char\* fileInput, ATM itemAtm);//Withdrew amount from account

void depositToAccount(char\* fileInput, ATM itemAtm);//Deposit amount to account

void checkBalance(char\* fileInput, ATM itemAtm);//Check balance for account

void closeAccount(char\* fileInput, ATM itemAtm);//Close account

void transferAccounts(char\* fileInput, ATM itemAtm);//Transfer amount from account to target account

int main(int argc, char\* argv[]) {

ATM Atm;

Account deleteAccount, tempDeletePrev, tempDeleteNext;

pthread\_t bankThread[1], \*threadArr;

int checkCloseFile2, iseekBits, iseekBitsForZero, checkCloseFile;

char \*bufferFile, \*inputATM, \*checkNumATMChar, ch, \*charAmount, \*nameFile;

int ret\_in, ret\_out, checkError, lenFileATM, numATM = 0, k = 1;

int n, i = 0, j;

printf("Please enter numbers of ATM:");

scanf("%d", &n);

if (argc > n + 1) {

printf("More than %d of files of the ATM\n", &n);

exit(1);

}

if (argc < n + 1) {

printf("less than %d of files of the ATM\n", &n);

exit(1);

}

logText = open("log.txt", O\_WRONLY | O\_CREAT, 0644);

if (logText == -1) {

perror("failed open log");

exit(100);

}

count = n;

Atm = (ATM)malloc(sizeof(struct atm)\*n);

threadArr = (pthread\_t\*)malloc(sizeof(pthread\_t)\*n);

bankAllAccounts = (Bank)malloc(sizeof(struct bank));

bankAllAccounts->numOfAccounts = 0;

bankAllAccounts->bankAccounts = NULL;

for (i = 0; i < n; i++){

strcpy(Atm[i].nameFile, argv[i + 1]);

}

for (i = 0; i < n; i++) {

if (pthread\_create(&threadArr[i], NULL, checkATM, (void\*)&Atm[i]) != 0)

exit(2);

}

for (i = 0; i < n; i++) {

if (pthread\_join(threadArr[i], NULL) != 0)

exit(4);

}

if (pthread\_create(&threadArr[i], NULL, print\_bank, (void\*)bankAllAccounts) != 0)

exit(5);

while(count > 0){

if (pthread\_join(threadArr[i], NULL) != 0)

exit(6);

sleep(2);

}

sleep(5);

deleteAccount = bankAllAccounts->bankAccounts;

while (bankAllAccounts->bankAccounts != NULL) {

bankAllAccounts->bankAccounts = bankAllAccounts->bankAccounts->next;

tempDeletePrev = deleteAccount->prev;

tempDeleteNext = deleteAccount->next;

if (tempDeletePrev != NULL)

tempDeletePrev->next = deleteAccount->next;

if (tempDeleteNext != NULL)

tempDeleteNext->prev = deleteAccount->prev;

free(deleteAccount);

deleteAccount = NULL;

deleteAccount = bankAllAccounts->bankAccounts;

}

free(bankAllAccounts);

bankAllAccounts = NULL;

if (close(logText) == -1) {

printf("error on closing file\n");

exit(10);

}

return (0);

}

//Create ATM details from file input

void \*checkATM(void\* fileATM) {

ATM Atm = ((ATM)(fileATM));

int checkCloseFile2, iseekBits, iseekBitsForZero;

pthread\_t t[1];

//ssize\_t ret\_in, ret\_out;

char \*bufferFile, \*checkNumATMChar, ch, \*charAmount, \*nameFile, \*errorMessage;

int ret\_in, ret\_out, checkError, lenFileATM, numATM = 0, k = 1;

int n = 1, i = 0, j;

char \*inputATM, arrayBuffer[BUFFER\_SIZE];

errorMessage = (char\*)malloc(sizeof(char)\*(strlen(Atm->nameFile) + 16));

inputATM = (char\*)malloc(sizeof(char)\*(strlen(Atm->nameFile) + 1));

strcpy(inputATM, Atm->nameFile);

lenFileATM = strlen(inputATM) - 19;

checkNumATMChar = (char\*)malloc(sizeof(char)\*(lenFileATM + 1));

for (j = 4; inputATM[j] != (char)95; j++) {

checkNumATMChar[j - 4] = inputATM[j];

}

checkNumATMChar[j - 4] = NULL;

for (j = lenFileATM - 1; j >= 0; j--) {

numATM = ((int)inputATM[j + 4] - 48)\*k + numATM;

k \*= 10;

}

Atm->buffer = arrayBuffer;

Atm->file = open(inputATM, O\_RDONLY);

if (!(Atm->file)) {

printf("Error in openning file\n");

exit(1);

}

Atm->numATM = numATM;

strcpy(Atm->numATMChar, checkNumATMChar);

while (Atm->ret\_in = read(Atm->file, &ch, 1) > 0) {

if (Atm->ret\_in < 0) {

strcpy(errorMessage, "Failed to open ");

strcpy(errorMessage, inputATM);

perror(errorMessage);

exit(Atm->numATM);

}

Atm->buffer[i] = ch;

i++;

}

Atm->buffer[i] = NULL;

Atm->lengthBuffer = i;

if (pthread\_create(&t[0], NULL, commandATM, (void\*)&Atm[0]) != 0)

exit(1);

if (pthread\_join(t[0], NULL) != 0)

exit(1);

usleep(100);

count--;

free(errorMessage);

free(inputATM);

free(checkNumATMChar);

if ((checkCloseFile2 = close(Atm->file) == -1)) {

printf("error on closing file\n");

exit(7);

}

errorMessage = NULL;

inputATM = NULL;

checkNumATMChar = NULL;

}

//All the commands of ATM

void\* commandATM(void\* fileATM) {

ATM itemAtm = ((ATM)(fileATM));

int w = 0, flagCheckMutex, conExit = 0, checkCloseFile,check = 1;

char \*fileInput, \*checkBuffer = itemAtm->buffer;

char charConditionATM;

long ret;

fileInput = (char\*)malloc(sizeof(char)\*(strlen(itemAtm->buffer) + 1));

while (w < itemAtm->lengthBuffer - 1) {

ret = strtol(checkBuffer, &fileInput, 36);

charConditionATM = checkBuffer[0];

switch (charConditionATM) {

case 'O':

createNewAccount(fileInput, itemAtm);

//sleep(1);

break;

case 'D':

depositToAccount(fileInput, itemAtm);

//sleep(1);

break;

case 'W':

withdrewFromAccount(fileInput, itemAtm);

//sleep(1);

break;

case 'B':

checkBalance(fileInput, itemAtm);

//sleep(1);

break;

case 'Q':

closeAccount(fileInput, itemAtm);

//sleep(1);

break;

case 'T':

transferAccounts(fileInput, itemAtm);

//sleep(1);

break;

}

while (checkBuffer[0] != '\n' && w < itemAtm->lengthBuffer - 1) {

w++;

checkBuffer++;

}

checkBuffer++;

}

}

//Create new account

void createNewAccount(char\* fileInput, ATM itemAtm) {

int checkPrev = 0;

int ret\_in, ret\_out;

//ssize\_t ret\_in, ret\_out;

int amount, accountID, accountPassword, amountCharBalance, flagCheckNext = 0, accountTransfer;

int i, j, k = 1, digitsIDAccount = 0, digitsBalance = 0, flagO = 2, flagD = 0, flagW = 0, flagB = 0, flagQ = 0

, digitsAmount = 0, password, flagTarget = 0, flagCheck = 0;

char charConditionATM, \*charAmount, \*tempStartAccountsBank, amountChar[100], accountNotExist[100], accountTargetNotExist[100];

char errorMessage[1000], Message[1000];

Account newAccount, temp, start,returnStartBankAccounts;

char outlog[100];

temp = bankAllAccounts->bankAccounts;

start = bankAllAccounts->bankAccounts;

newAccount = (Account)malloc(sizeof(struct account) \* 1);

newAccount->numAccount = strtol(fileInput, &fileInput, 10);

sprintf(newAccount->numAccountChar, "%d", newAccount->numAccount);

newAccount->password = strtol(fileInput, &fileInput, 10);

sprintf(newAccount->passwordChar, "%d", newAccount->password);

newAccount->balance = strtol(fileInput, &fileInput, 10);

sprintf(newAccount->balanceChar, "%d", newAccount->balance);

newAccount->mutex = (sem\_t\*)malloc(sizeof(sem\_t));

if (sem\_init(newAccount->mutex, 0, 1) != 0) {

fprintf(stderr, "Error by creating semaphore\n");

return 3;

}

newAccount->next = NULL;

newAccount->prev = NULL;

while (temp != NULL) {

if (newAccount->numAccount == temp->numAccount) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - account with the same id exists\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

flagO = 0;

}

temp = temp->next;

}

if (flagO == 2) {

sem\_wait(&newAccount->mutex[0]);

if (bankAllAccounts->bankAccounts == NULL) {

bankAllAccounts->bankAccounts = newAccount;

}

while (start != NULL && flagCheckNext == 0) {

if (newAccount->numAccount < start->numAccount) {

if (start->prev == NULL)

checkPrev = 1;

newAccount->next = start;

start->prev = newAccount;

start = newAccount;

if (checkPrev == 1)

bankAllAccounts->bankAccounts = newAccount;

flagCheckNext = 1;

}

if (start->next == NULL && flagCheckNext == 0) {

start->next = newAccount;

newAccount->prev = start;

flagCheckNext = 1;

}

if(flagCheckNext == 0)

start = start->next;

}

strcpy(Message, itemAtm->numATMChar);

strcat(Message, ": New account id is ");

strcat(Message, newAccount->numAccountChar);

strcat(Message, " with password ");

strcat(Message, newAccount->passwordChar);

strcat(Message, " and initial balance ");

strcat(Message, newAccount->balanceChar);

strcat(Message, "\n");

if ((ret\_out = write(logText, Message, strlen(Message))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

bankAllAccounts->numOfAccounts++;

sem\_post(&newAccount->mutex[0]);

}

}

//Deposit to account

void depositToAccount(char\* fileInput, ATM itemAtm) {

int ret\_in, ret\_out;

//ssize\_t ret\_in, ret\_out;

int amount, accountID, accountPassword, amountCharBalance, flagCheckNext = 0, accountTransfer;

int i, j, k = 1, digitsIDAccount = 0, digitsBalance = 0, flagO = 2, flagD = 0, flagW = 0, flagB = 0, flagQ = 0

, digitsAmount = 0, password, flagTarget = 0, flagCheck = 0;

char charConditionATM, \*charAmount, \*tempStartAccountsBank, amountChar[100], accountNotExist[100], accountTargetNotExist[100];

char errorMessage[1000], Message[1000];

Account newAccount, temp, start;

char outlog[100];

accountID = strtol(fileInput, &fileInput, 10);

sprintf(accountNotExist, "%d", accountID);

temp = bankAllAccounts->bankAccounts;

start = bankAllAccounts->bankAccounts;

while (temp != NULL) {

if (accountID == temp->numAccount) {

flagW = 1;

accountPassword = strtol(fileInput, &fileInput, 10);

if (accountPassword != temp->password) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - password for account id ");

strcat(errorMessage, temp->numAccountChar);

strcat(errorMessage, " is incorrect");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

else {

sem\_wait(&temp->mutex[0]);

amount = strtol(fileInput, &fileInput, 10);

temp->balance = temp->balance + amount;

sprintf(temp->balanceChar, "%d", temp->balance);

sprintf(amountChar, "%d", amount);

strcpy(Message, itemAtm->numATMChar);

strcat(Message, ": Account ");

strcat(Message, temp->numAccountChar);

strcat(Message, " new balance is ");

strcat(Message, temp->balanceChar);

strcat(Message, " after ");

strcat(Message, amountChar);

strcat(Message, " $ was deposited\n");

if ((ret\_out = write(logText, Message, strlen(Message))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

sem\_post(&temp->mutex[0]);

}

}

temp = temp->next;

}

if (flagW == 0) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - account id ");

strcat(errorMessage, accountNotExist);

strcat(errorMessage, " does not exist\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

}

//Withdrew from account

void withdrewFromAccount(char\* fileInput, ATM itemAtm) {

int ret\_in, ret\_out;

//ssize\_t ret\_in, ret\_out;

int amount, accountID, accountPassword, amountCharBalance, flagCheckNext = 0, accountTransfer;

int i, j, k = 1, digitsIDAccount = 0, digitsBalance = 0, flagO = 2, flagD = 0, flagW = 0, flagB = 0, flagQ = 0

, digitsAmount = 0, password, flagTarget = 0, flagCheck = 0;

char charConditionATM, \*charAmount, \*tempStartAccountsBank, amountChar[100], accountNotExist[100], accountTargetNotExist[100];

char errorMessage[1000], Message[1000];

Account newAccount, temp, start;

char outlog[100];

accountID = strtol(fileInput, &fileInput, 10);

sprintf(accountNotExist, "%d", accountID);

temp = bankAllAccounts->bankAccounts;

start = bankAllAccounts->bankAccounts;

while (temp != NULL) {

if (accountID == temp->numAccount) {

flagW = 1;

accountPassword = strtol(fileInput, &fileInput, 10);

if (accountPassword != temp->password) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - password for account id ");

strcat(errorMessage, temp->numAccountChar);

strcat(errorMessage, " is incorrect");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

else {

amount = strtol(fileInput, &fileInput, 10);

if (temp->balance - amount < 0) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - account id ");

strcat(errorMessage, temp->numAccountChar);

strcat(errorMessage, " balance is lower than ");

strcat(errorMessage, amount);

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

else {

sem\_wait(&temp->mutex[0]);

temp->balance = temp->balance - amount;

sprintf(temp->balanceChar, "%d", temp->balance);

sprintf(amountChar, "%d", amount);

strcpy(Message, itemAtm->numATMChar);

strcat(Message, ": Account ");

strcat(Message, temp->numAccountChar);

strcat(Message, " new balance is ");

strcat(Message, temp->balanceChar);

strcat(Message, " after ");

strcat(Message, amountChar);

strcat(Message, " $ was withdrew\n");

if ((ret\_out = write(logText, Message, strlen(Message))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

sem\_post(&temp->mutex[0]);

}

}

}

temp = temp->next;

}

if (flagW == 0) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - account id ");

strcat(errorMessage, accountNotExist);

strcat(errorMessage, " does not exist\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

}

//Check balance

void checkBalance(char\* fileInput, ATM itemAtm) {

int ret\_in, ret\_out;

//ssize\_t ret\_in, ret\_out;

int amount, accountID, accountPassword, amountCharBalance, flagCheckNext = 0, accountTransfer;

int i, j, k = 1, digitsIDAccount = 0, digitsBalance = 0, flagO = 2, flagD = 0, flagW = 0, flagB = 0, flagQ = 0

, digitsAmount = 0, password, flagTarget = 0, flagCheck = 0;

char charConditionATM, \*charAmount, \*tempStartAccountsBank, amountChar[100], accountNotExist[100], accountTargetNotExist[100];

char errorMessage[1000], Message[1000];

Account newAccount, temp, start;

char outlog[100];

temp = bankAllAccounts->bankAccounts;

start = bankAllAccounts->bankAccounts;

accountID = strtol(fileInput, &fileInput, 10);

sprintf(accountNotExist, "%d", accountID);

while (temp != NULL) {

if (accountID == temp->numAccount) {

flagB = 1;

accountPassword = strtol(fileInput, &fileInput, 10);

if (accountPassword != temp->password) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - password for account id ");

strcat(errorMessage, bankAllAccounts->bankAccounts->numAccountChar);

strcat(errorMessage, " is incorrect\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

else {

strcpy(Message, itemAtm->numATMChar);

strcat(Message, ": Account ");

strcat(Message, temp->numAccountChar);

strcat(Message, " balance is ");

strcat(Message, temp->balanceChar);

strcat(Message, "\n");

if ((ret\_out = write(logText, Message, strlen(Message))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

}

temp = temp->next;

}

if (flagB == 0) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - account id ");

strcat(errorMessage, accountNotExist);

strcat(errorMessage, " does not exist\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

}

//Close account

void closeAccount(char\* fileInput, ATM itemAtm) {

int ret\_in, ret\_out;

sem\_t \*s;

//ssize\_t ret\_in, ret\_out;

int amount, accountID, accountPassword, amountCharBalance, flagCheckNext = 0, accountTransfer,checkPrev = 0;

int i, j, k = 1, digitsIDAccount = 0, digitsBalance = 0, flagO = 2, flagD = 0, flagW = 0, flagB = 0, flagQ = 0

, digitsAmount = 0, password, flagTarget = 0, flagCheck = 0;

char charConditionATM, \*charAmount, \*tempStartAccountsBank, amountChar[100], accountNotExist[100], accountTargetNotExist[100];

char errorMessage[1000], Message[1000];

Account newAccount, temp, start, tempDeleteNext, tempDeletePrev;

char outlog[100];

temp = bankAllAccounts->bankAccounts;

start = bankAllAccounts->bankAccounts;

accountID = strtol(fileInput, &fileInput, 10);

sprintf(accountNotExist, "%d", accountID);

while (temp != NULL) {

if (accountID == temp->numAccount) {

if (temp->prev == NULL)

checkPrev = 1;

flagQ = 1;

accountPassword = strtol(fileInput, &fileInput, 10);

if (accountPassword != temp->password) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - password for account id ");

strcat(errorMessage, temp->numAccountChar);

strcat(errorMessage, " is incorrect\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

else {

sem\_wait(&temp->mutex[0]);

if(checkPrev == 1)

bankAllAccounts->bankAccounts = bankAllAccounts->bankAccounts->next;

strcpy(Message, itemAtm->numATMChar);

strcat(Message, ": Account ");

strcat(Message, temp->numAccountChar);

strcat(Message, " is now closed.Balance was ");

strcat(Message, temp->balanceChar);

strcat(Message, "\n");

tempDeletePrev = temp->prev;

tempDeleteNext = temp->next;

if (tempDeletePrev != NULL)

tempDeletePrev->next = temp->next;

if (tempDeleteNext != NULL)

tempDeleteNext->prev = temp->prev;

free(&temp->mutex[0]);

free(temp);

temp = NULL;

if ((ret\_out = write(logText, Message, strlen(Message))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

}

if (temp != NULL)

temp = temp->next;

}

if (flagQ == 0) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, " Your transaction failed - account id ");

strcat(errorMessage, accountNotExist);

strcat(errorMessage, " does not exist\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

}

//Transfer account

void transferAccounts(char\* fileInput, ATM itemAtm) {

int ret\_in, ret\_out;

//ssize\_t ret\_in, ret\_out;

int amount, accountID, accountPassword, amountCharBalance, flagCheckNext = 0, accountTransfer;

int i, j, k = 1, digitsIDAccount = 0, digitsBalance = 0, flagO = 2, flagD = 0, flagW = 0, flagB = 0, flagQ = 0

, digitsAmount = 0, password, flagTarget = 0, flagCheck = 0, flagT = 0;

char charConditionATM, \*charAmount, \*tempStartAccountsBank, amountChar[100], accountNotExist[100], accountTargetNotExist[100];

char errorMessage[1000], Message[1000];

Account newAccount, temp, start, tempDeleteNext, tempDeletePrev;

char outlog[100];

temp = bankAllAccounts->bankAccounts;

start = bankAllAccounts->bankAccounts;

accountID = strtol(fileInput, &fileInput, 10);

sprintf(accountNotExist, "%d", accountID);

while (temp != NULL) {

if (accountID == temp->numAccount) {

flagT = 1;

accountPassword = strtol(fileInput, &fileInput, 10);

if (accountPassword != temp->password) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - password for account id ");

strcat(errorMessage, temp->numAccountChar);

strcat(errorMessage, " is incorrect\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

else {

accountTransfer = strtol(fileInput, &fileInput, 10);

sprintf(accountTargetNotExist, "%d", accountTransfer);

amount = strtol(fileInput, &fileInput, 10);

sprintf(amountChar, "%d", amount);

if (temp->balance - amount < 0) {

flagTarget = 1;

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - account id ");

strcat(errorMessage, temp->numAccountChar);

strcat(errorMessage, " balance is lower than ");

strcat(errorMessage, amountChar);

strcat(errorMessage, "\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

else {

while (start != NULL) {

if (accountTransfer == start->numAccount) {

sem\_wait(&temp->mutex[0]);

sem\_wait(&start->mutex[0]);

flagTarget = 1;

start->balance = start->balance + amount;

sprintf(start->balanceChar, "%d", start->balance);

temp->balance = temp->balance - amount;

sprintf(temp->balanceChar, "%d", temp->balance);

strcpy(Message, itemAtm->numATMChar);

strcat(Message, ": Transfer ");

strcat(Message, amountChar);

strcat(Message, " from account ");

strcat(Message, temp->numAccountChar);

strcat(Message, " to account ");

strcat(Message, start->numAccountChar);

strcat(Message, " new account balance is ");

strcat(Message, temp->balanceChar);

strcat(Message, " new target account balance is ");

strcat(Message, start->balanceChar);

strcat(Message, "\n");

if ((ret\_out = write(logText, Message, strlen(Message))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

sem\_post(&temp->mutex[0]);

sem\_post(&start->mutex[0]);

}

start = start->next;

}

if (flagT == 0) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - account id ");

strcat(errorMessage, accountNotExist);

strcat(errorMessage, " does not exist\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

flagCheck = 1;

}

if (flagTarget == 0 && flagCheck == 0) {

strcpy(errorMessage, "Error ");

strcat(errorMessage, itemAtm->numATMChar);

strcat(errorMessage, ": Your transaction failed - account id ");

strcat(errorMessage, accountTargetNotExist);

strcat(errorMessage, " does not exist\n");

if ((ret\_out = write(logText, errorMessage, strlen(errorMessage))) < 0) {

perror("faild to write to the log file\n");

exit(4);

}

}

}

}

}

temp = temp->next;

}

}

//Print all the bank accounts details

void\* print\_bank(void\* bankAccounts) {

Bank bankTemp = (Bank)bankAccounts;

Account temp = bankTemp->bankAccounts;

int balance = 0,i = 0;

printf("Current bank Status\n");

while (temp != NULL && temp->balance != 0) {

balance += temp->balance;

printf("Account %d: balance = %d $ , Account Password = %d\n", temp->numAccount, temp->balance, temp->password);

temp = temp->next;

}

printf("The Bank has %d\n\n", balance);

}

**צילום מסך:**

