CSE 344 Midterm Project

A simple Bank Simulator

Due on May 28th 2025@ 9:00

In this project you are expected to implement a banking simulator via client-server based implementation using processes. The main server (the bank itself) program will be responsible for opening/closing and checking the relative information about existing bank accounts. For each client the main server is expected to create a "Teller" process. The Teller process will be responsible for serving client requests and checking the validity of these requests. The client requests can be *(i)* deposit credits, *(ii)* withdraw credits, to the clients bank account. The Teller will receive the data from the clients and send it to the main server (through a pipe connection), so that the main server can update the banks database accordingly. At the end of each transaction the Teller will inform the client about the latest information about their account. Only the main server process should have direct access to the bank database and will update the corresponding log file time to time to ensure that the log is always up to date.

When a new client deposits some amount of credit to the bank the main server will create a new account and assign a bank account Id specific to the client. When a client withdraws all the credits in their account their client Id should be removed from the banks database. Aside from the new clients all clients requesting to connect to the server program should provide their bank account Id to the main server program.

The connection between the clients and the main server is expected via a Server FIFO generated in the current working directory of the bank server. The teller process to client connection is expected to be via a client generated named pipe.

A sample Server program output under normal operation might in the form :

> **BankServer AdaBank #ServerFIFO_Name**

```
Adabank is active….
No previous logs.. Creating the bank database
Waiting for clients @ServerFIFO_Name…
 - Received 3 clients from PIDClientX..
 -- Teller PID01 is active serving Client01…
 -- Teller PID02 is active serving Client02…
 -- Teller PID03 is active serving Client03…
. . .
Client01 deposited 300 credits… updating log
Client03 deposited 2000 credits updating log
Client02 withdraws 30 credit.. operation not permitted.
. . .
Waiting for clients @ServerFIFO_Name…
 - Received 2 clients from PIDClientY
 -- Teller PID04 is active serving Client01…Welcome back Client01
 -- Teller PID05 is active serving Client04…
. . .
Client01 withdraws 300 credits… updating log… Bye Client01
Client04 deposited 20 credits updating log
. . .
Waiting for clients @ServerFIFO_Name…
Signal received closing active Tellers
Removing ServerFIFO… Updating log file…
Adabank says "Bye"...
```

The corresponding Client side output might in the form :

(the contents of a possible client files…)

> cat Client01.file

```
   N deposit 300
   BankID_None withdraw 30
   N deposit 2000
>
```

> cat Client02.file

```
   BankID_01 withdraw 300
   N deposit 20
>
```

> cat Client03.file

```
   BankID_02 withdraw 30
   N deposit 2000
   BankID_02 deposit 200
   BankID_02 withdraw 300
   N withdraw 20
>
```

> **BankClient <Client01.file> #ServerFIFO_Name**

```
   Reading Client.file..
   3 clients to connect.. creating clients..
   Connected to Adabank..
   Client01 connected..depositing 300 credits
   Client02 connected..withdrawing 30 credits
   Client03 connected..depositing 2000 credits
   ..
   Client01 served.. BankID_01
   Client02 something went WRONG
   Client03 served.. BankID_02
   exiting..
```

Then connect again

> **BankClient <Client02.file> #ServerFIFO_Name**

```
   Reading Client02.file
   2 clients to connect.. creating clients..
   Connected to Adabank..
   Client01 connected..withdrawing 300 credits
   Client04 connected..depositing 20 credits
   ..
   Client01 served.. account closed
   Client04 served.. BankID_03
   exiting..
```

connect again (but no bankserver running)

> **BankClient <Client03.file> #ServerFIFO_Name**

```
   Reading Client03.file 5 clients to connect.. creating clients..
   Cannot connect ServerFIFO_Name...
   exiting..
```

And example Server log at the end of the run

> **cat AdaBank.bankLog**

```
   # Adabank Log file updated @10:37 April 18 2025

   # BankID_01 D 300 W 300  0
   BankID_02 D 2000  2000
   BankID_03 D 20  20

   ## end of log.
```

Both the server and client programs should handle signals properly. If the client action is interrupted by a signal, the teller process on the server side should disconnect from the client connection without crashing. The bank database should only be updated when a client request is completed.

Implementing up to this point should be sufficient to get 50/100.
Those of you aiming for a DD grade can stop here.!!!!

In order to obtain full credit, instead of using fork, implement the Teller process creation via

```
pid_t Teller (void* func, void* arg_func);
```

and wait for the child process via

```
int waitTeller(pid_t pid, int* status);
```

Here, the Teller call should create a new process that will only execute the function defined by `func` with the arguments given in `arg_func.` The return of the function `Teller` is the PID of the child created. The process functions are limited to `deposit()` and `withdraw()`. Use shared memory and semaphores for bank server to teller communications (not the status).

The project requires inter-process communication and multiple synchronization primitives. Test you code with multiple clients (up to 20), check for memory leaks. Write a report showing your test results and explaining how you have implemented the project . Provide a makefile to compile the overall project, sample client files for multiple client connections.

Deliverables :

- A document describing your system architecture, design decisions, and implementation details.
- The source code (for the Server, Client and extras if necessary) and a makefile to compile the project as a whole.
- A test plan and results demonstrating that your system meets the requirements

Best of Luck