# EE 4374 Operating Systems Design
# Programming Assignment #3: Bank Client/Server Applications
# Due Date: March 21, 2022 (before Midnight)

**Objectives**:

1) To learn how to use socket programming to create a client/server application pair.
2) To learn how to use mutual exclusion semaphores in pthreads to guarantee mutual exclusion to critical sections.

**Tasks**:

1) Unpack the Assignment 3 template provided by the instructor into your home directory: 'tar zxvf student_prog3.tgz'. This will create a directory called 'student_prog3', please rename this directory to firstinitiallastname_prog3 using the 'mv' command. For example, the instructor would rename the directory by executing 'mv student_prog3 mmcgarry_prog3'.
2) Develop a bank client application (bankClient) that takes the following command line arguments and then communicates with the server to process the transaction:

bankClient servIPAddr servPortNum transaction acctnum value

servIPAddr is the IP address of the bank server
servPortNum is the port number of the bank server
transaction is {B for balance inquiry, D for deposit, or W for withdrawal}
acctnum is the account number
value is the value of the transaction in pennies (ignored for a balance inquiry)

The protocol between the client and server is simple: they exchange a single message type in either direction.

```
/* Bank Transaction Types */
#define BANK_TRANS_DEPOSIT    0
#define BANK_TRANS_WITHDRAW   1
#define BANK_TRANS_INQUIRY    2

typedef struct
{
   unsigned int trans;          /* transaction type */
   unsigned int acctnum;        /* account number */
   unsigned int value;          /* value */
} sBANK_PROTOCOL;
```

The server responds in the following way to transaction requests from the client:

trans and acctnum are repeated from client message, value depends on the transaction
For a deposit, the deposited value is set in the value field.

For a withdrawal, if there is insufficient funds then a zero is set in the value field, otherwise the withdrawn amount is set in the value field.
For a balance inquiry, the balance of the account is set in the value field.

You can test your client with the instructor's server application that will be available at IP address 129.108.32.2 port 26207.

3) Develop a bank server application (bankServer) that processes transactions from clients. The server should handle multiple concurrent clients, and protect against race conditions using pthreads mutex semaphores. Bank account information is represented with the following structure:

```
#define NUM_ACCTS   100

typedef struct
{
   unsigned int       balance;
   pthread_mutex_t mutex;
} sBANK_ACCT_DATA;

sBANK_ACCT_DATA acctData[NUM_ACCTS];
```

4) Place the definition of the bank protocol structure in a header file named firstinitiallastname_banking.h.

5) Use a Makefile to build your program. You will rename your Makefile when you submit it.

6) Submit the deliverables, indicated below, as a single zip file named firstinitiallastname_prog3.zip through Blackboard.

**Deliverables**:

1) Submit all of the source files in your Assignment 3 directory as a single tarball file. You can create this by changing to the directory above your Assignment 3 directory and execute 'tar zcvf firstinitiallastname_prog3.tgz firstinitiallastname_prog3'. As an example, the instructor would execute 'tar zcvf mmcgarry_prog3.tgz mmcgarry_prog3'. This file will be submitted through Blackboard.

**Scoring**:

Lab grades will be based on four criteria that will determine your overall grade. The first criterion determines if your program compiles and executes correctly. The correct result must adhere to the **Tasks** section. The second criterion determines if your program uses the specified libraries and/or function prototypes as specified in the **Task** section. The last criterion determines if your source code is well-documented and adheres to submission guidelines. Your source code must include (at the top) your name, class

section, due date, assigned date, and a small description of your program. Also, every function/method must be commented by specifying the purpose of the input values (if any) and their respective output values (e.g. void foo(int x) /* input: x > 0, output: x = x * 2*/).

| **Operation/Successful Demonstration** | **70%** |
|---|---|
| Does the program compile? 30% | |
| Can the client work correctly with our server? 15% | |
| Can the server work correctly with our client? 15% | |
| Is the bank server application multi-threaded? 5% | |
| Did you identify the critical section with comments in the bank server application? 2% | |
| Did you guarantee mutual exclusion to the critical section using the **pthreads mutex semaphore** in the bank account data structure? 3% | |
| **Adherence to Interface Specification** | **20%** |
| Does your client program use the command line arguments specified in the lab assignment? 10% | |
| Does your program use the structure definitions specified in the lab assignment? 10% | |
| **Comments/Adherence to Submission Specification** | **10%** |
| Does your submission adhere to the filename guidelines? 5% | |
| Is the source code well-documented? 5% | |
| **Lateness** | **10%** per day (including weekends and holidays) |