Multi ~~threaded~~ **Processing** Bank Server

Jungsoo Park

The simpler half first – the client of my multiprocessing bank initially waits to connect to the server. If the server is unavailable, it will continue attempting to connect every three seconds until it connects. From there, the client will fork off into two separate processes. The parent process handles receiving all messages from the server, while the child process handles taking input to the user and writing it to the server.

The server first sets up a timer to send SIGALRM and a signal handler to print the bank status every 20 seconds. Then, we open up a file (located in /tmp/bankinfo.bin) for memory-mapping to hold all the information for the bank. From there, different server processes will be able to asynchronously access the bank information. Access of the bank data is mutex protected at both the bank level and individual account level to prevent race conditions. After setting up the memory-mapping to file, the server process perpetually waits to accept client connections. When a client connects, this process (the session-acceptor process) forks child processes for each client connection, which then goes into the client-handling loop. I/O to and from the client is fully synchronous.

shmserver.c is an alternate version of the server using shared-memory to store the bank data instead of memory-mapped files. It works exactly the same as the original version of the server.