מבנה מערכות הפעלה – רטוב 2

מגישים:

מנור צבי – 204030720

אביב אזרן – XXXXXXXXX

bank.cpp

* vector<Account> accounts
* pthread\_t [<#ATMs>] atm threads + 2 (commission & printer)

<#ATMs> + 2 threads

Main methods:

* Atm thread handle
* Commission thread handle
* Printer thread handle

Synchronization mechanism:

* Readers – writers on accounts vector
* Readers – writers on ‘dones’ counter
* Readers – writers on bank account
* Mutex on logger utility

Account#

Account#

Account#

Account#

Account#

Atm thread #

Atm thread #

Atm thread #

Atm thread #

Atm thread #

Commissioner thread

Printer thread

Account.h/cpp

* Int balance
* bool isVIP
* int id
* int passward

synchronization:

* Readers – writers on balance
* Readers – writers on isVIP

Methods:

* Get\_\* - as service
* Log\_\* - main functionality against the bank
* Give\_commision – zero sleep (when engaging with the bank)
* Get\_commision – only as a bank – also an account

סכמת בלוקים עבור התכנית:

As accounts vector

Each one is

As atm threads array

As instancing of threads

1. Main file in bank.cpp.
2. There we have array of threads, one for each atm, and two more threads: one to collect commission and one to print to monitor the status.
3. We manage vector of object of class **Account**, each one of them act as a **monitor** with respect to it’s isVIP & balance fields. It has locks on those fields, to make sure that multiple agents can read it on the same time, but only one can write (and when he does – no one can read).
4. We manage also readers – writers mechanism on the accounts vector itself: multiple agents can search for accounts and interacts with them (including writing to given account), but only one can add account to the vector. The parallel writing is possible due to different locks on each account instance, and the vector itself.