May 20th, 2018

CSE344 – System Programming – Final Project

Context and objective

There is no denying the existence of cheating students. And unfortunately, the proliferation of technology has widened their opportunities. More specifically, your greedy boss has come up with an idea of an application focusing on getting together *people that solve homeworks for money* (henceforth *the providers*), and *cheating students that want to pay to get their homework done* (henceforth *the clients*).

The main idea is to have a server, with a pool of provider threads. Each provider thread is associated with three numerical parameters: the provider's price (a flat rate between 50 to 1000TL) per homework, performance(quality) (between 1-worst and 5-best) and login duration (i.e. the number of seconds the provider will be active). A provider thread solves all homeworks in its queue, and waits otherwise, until eventually its login time is over.

A client on the other hand will connect to a server, send his/her homework through the socket and a priority parameter (either cost C, quality Q or time T) to the server. Depending on the chosen priority parameter, the server will forward the homework to **either the cheapest, best performing or least busy provider**. Once the homework is done the client will collect the result of computation and terminate.

Risks

- Some providers will logout earlier than others, so the provider ranks (in terms of cost and quality) are not constant.
- Every client will submit her/his homework **and** wait for collecting its result. Careful with synchronization.

How

- For the sake of simplicity all homeworks consist of an integer value (in degrees) whose cosine must be calculated using the Taylor series.
- Your server is threaded, every new connection **MUST be handled as a new thread**. This new thread will post the job to provider(if available) and wait for it to finish.
- All server/provider messages must be printed on stderr.
- **The task queue of every provider has size 2. If for instance the client prioritizes performance and the best performing provider's queue is full, then the server must forward the task to the next best performing provider.**

a) server

Admits as command line argument:
- the connection port for listening to clients
- file of providers
- and the log file

You are provided with a file `data.dat` containing the list of providers and their parameters.
e.g. `Murat 2 200 360` means

Murat is his name, 2 is his performance (i.e. quality of homeworks solved), 200TL is his price and he'll remain logged in for 360 seconds in total.

Your server will read this file at startup, and create one thread per provider, and then wait for client connections.

Once a client connects, your server will forward the homework to the corresponding provider. If no provider is available an error message "NO PROVIDER IS AVAILABLE" will be sent to the client. Your server will continue to work until a termination signal is received, in which case it will first close up all client sockets by sending a polite message such as "SERVER SHUTDOWN" and terminate all provider threads.

b) client
Every client will consist of **a separate process** with the following commandline arguments:
- client name as a string
- her/his priority 'C' for low cost or 'Q' for high quality/performance or 'T' for high speed
- the homework as an integer denoting degree
- server address
- server port address

and will connect to the server send a request in the form of a simple string:

e.g. `"Hileci C 45"` means the client's name is Hileci, his priority is **low cost** and the homework is cos(45)

`"Hileci2 Q 55"` means the client's name is Hileci2, his priority is **high performance** and the homework is cos(55)

`"Hileci3 T 30"` means the client's name is Hileci3, his priority is **high speed** and the homework is cos(30).

Sample output for <u>a client</u> console:
```
Client Hileci is requesting Q 45 from server 127.0.0.1:5555
Hileci's task completed by Ayse in 7.53 seconds, cos(45)=0.707, cost is 900TL,
total time spent 7.89 seconds.
```

c) provider
Every provider will wait for tasks in her/his queue, complete the task, <u>sleep for a random duration between 5-15 seconds</u> (to simulate hard work...), and return its result to the client. Once the login time is up, the provider will logout and her/his thread will be terminated.

Sample output for <u>a provider</u> processing a task:
```
Provider Ayse is waiting for a task
Provider Ayse is processing task number 1: 45
Provider Ayse completed task number 1: cos(45)=0.707 in 7.53 seconds.
Provider Ayse is processing task number 2: 55
Provider Ayse completed task number 2: cos(55)=0.573 in 9.71 seconds.
Provider Ayse is waiting for a task
```

If everything goes well your application should print on screen something like the following:

- server (together with providers)
```
$homeworkServer 5555 data.dat log.dat
Logs kept at log.dat
7 provider threads created
Name        Performance Price Duration
Ayse        5           900   60
Fatma       3           500   180
Murat       2           200   360
```

```
Hakan       4           750   240
Cengiz      1           75    600
Nuray       3           450   90
Ertan       4           600   120
Provider Ayse waiting for tasks
Provider Fatma waiting for tasks
Provider Cengiz waiting for tasks
Provider Murat waiting for tasks
Provider Nuray waiting for tasks
Provider Ertan waiting for tasks
Provider Hakan waiting for tasks
Server is waiting for client connections at port 5555
Client Hileci2 (Q 45) connected, forwarded to provider Ayse
Provider Ayse is processing task number 1: 45
Client Hileci10 (Q 99) connected, forwarded to provider Ayse
Client Hileci10 (C 11) connected, forwarded to provider Cengiz
Provider Ayse completed task number 1: cos(45)=0.707 in 7.53 seconds.
...
...
Termination signal received
Terminating all clients
Terminating all providers
Statistics
Name        Number of clients served
Ayse        2
Fatma       1
Murat       0
Hakan       1
Cengiz      2
Nuray       1
Ertan       1
Goodbye.
```

Evaluation
In order to simulate intensive client connections, use a shell script with a list of consecutive client connections;

e.g.

```
clientApp Hileci1 C 45 127.0.0.1 5555
clientApp Hileci2 Q 99 127.0.0.1 5555
clientApp Hileci3 T 11 127.0.0.1 5555
clientApp Hileci4 C 19 127.0.0.1 5555
...
```