CEng 536 Advanced Unix
Fall 2018
HW1
Due: 17/11/2018

1 Description

In this homework you will write an IPC socket based tool assignment system for a gym. Assume there are k tools (i.e. a running mill) of a kind in a gym and a typical person in the gym waits for a tool, use it for a period of time, than rest for a while, than use another turn in the same tool (for simplicity, we assume there is only one kind of tool in the gym). After repeating these in couple of turns, person leaves the gym.

The owner of the gym is sensitive in fairness, so asks for an application making sure tools are uniformly shared by the customers when they are fully utilized. The fairness can be established by such set of rules:

- System keeps track of each customers time spent on the tools accumulatively (called *share*)
- Each customer is allowed to use a tool for q time units without intervention once s/he gets the tool.
- After q, the list of waiting customers is inspected if there is a customer with a smaller *share*. If so, customer leaves the tool and enlisted in waiting customer list with his/her updated *share*. The other customer gets the tool.
- When a new customer arrives, s/he is assigned to the average *share* of all (on tool, waiting and resting) customers.
- When a customer rests for a long period, his/her *share* may be too small so that s/he can occupy a tool for long periods of time. In order to avoid it, there is a Q value which is the upper limit a customer can use a tool without interrupt. After this limit, customer will leave the tool and assigned to waiting list and smallest *share* customer in the waiting list will have a chance to use the tool.
- After q, requests from any of the resting customers with smaller *share* will cause customer to leave the tool.

2 Application

This application will be implemented on a stream socket scenario. The application will be a multiprocess server. There will be a master process accepting connections, k tool processes simulating tools in the gym, and for

each connection an agent process serving the customer session, controlling the per customer communication. When an external client program connects the master, an agent process is created. The agent is responsible for handling the following text commands from the client:

- REQUEST to get a tool. The agent will put the customer in the list of waiting customers and inform tool processes. When a tool is available, customer will be informed with id of the tool and customer may start using the tool.
- REST to leave a tool and take a rest. During the rest, customer will be in the gym but not on any waiting list. Even if there is an available tool, customer is not going to be assigned to that.
- REPORT to get a full report of the gym including number of customers in the waiting list, in the rest, average share, current share, share of customers on the tools etc.
- QUIT to leave the gym. Connection will be closed, customer will be deleted in all lists s/he exists. Agent will terminate. Connection close will have the same affect with QUIT.

Newly arrived customers will be on the resting state. Agents will inform clients in all events like when tool is taken away due to q and Q limits.

Since this is a multi-process implementation, customer information has to be kept on a shared memory data structure. In the minimum case, the number of customers, total share of customers and list of waiting customer with their shares has to be kept in a shared memory data structure. For sake of simplicity, there is an upper bound of number of waiting customers which is 1000000. Since the access to the waiting customer list will be based on minimum *share*, you had better use a priority queue for $\mathcal{O}(log(n))$ complexity.

Also processes have to synchronize with eachother in the following cases:

1. Tool process informs the relevant agent process when tool is taken away from the customer
2. Tool processes go in sleep when no customer waits. In this case if a new request arrives, they need to be waken up.
3. A tool process is informed by the assigned customers agent when there is a REST or QUIT request.
4. Tool processes are informed when customer with a smaller share is arrived in waiting list (if assigned customer has spent more than q in the current run). Only tool with the maximum current share is informed. In case of a tie any tool process can be selected.

You can use multithreading within each process but cannot combine master, tool and agent processes, they need to be seperate processes.

3 Execution

Your code will compile into `hw1`. The following command line arguments are going to be supported:
`./hw1 conn q Q k`

conn is the address to be listened by the master process. If it starts with a `@`, it is assumed to be a unix path, thus a unix domain socket. Otherwise it should be in `ip:port` format, IP and port of the IPv4 socket address.

q and *Q* are integers for minimum and maximum tool usage limit in a single run for a customer. They are taken as milliseconds ($1/1000th$ of a second). *k* is the number of the tool processes started at the beginning.

4 Input/Output

Clients typically send `REQUEST` and `REST` requests repeatedly. The agents will send events related to the customer. One of the following output lines will be generated (shown in printf syntax):

```
Customer with share %d is assigned to tool %d.
Customer with share %d is removed from tool %d due to a customer with %d.
Customer with share %d leaves tool %d
```

The output of `REPORT` command is as follows:

```
k: %d, customers: %d waiting, %d resting, %d in total
average share: %.2f
waiting list:
customer      share        waits for
-------------------------------------
%-12d %20d %12d
```

Customers are reported by id of their agent processes. The time customer inserted in waiting list is preserved in milliseconds and report produces a "waits for" column by subtracting insertion time from current time. The report is ordered by "share" in ascending order.

5 Submission and Questions

Use metuclass for submission. Provide only `hw1.c` or `hw1.cpp` file in your submission. You can ask any homework related questions on course newsgroup (CoW or NNTP).