Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SS 2007 Prof. T. Roscoe

Vernetzte Systeme
Übung 9

Ausgabe: **31. Mai 2007**
Abgabe: **15. Juni 2007**

Bitte schreiben Sie immer Ihre(n) Namen auf die Lösungsblätter.

1 Writing a single-user eggtimer server

The exercise is to write an eggtimer server. It will work as follows: the user opens a `telnet` connection to the server, which then outputs a set of "Tick" messages which count down progressively each second from an initial value. When the count reaches zero, the server closes the connection. A typical session might look as follows:

```
$ telnet localhost 4000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Tick: 60
Tick: 59
Tick: 58
...
Tick: 3
Tick: 2
Tick: 1
Connection closed by foreign host.
$
```

What makes it more interesting is that the user can type a new value into the telnet session while the timer is running (without interrupting or delaying it), and the timer server will reset the clock accordingly (for example, if the egg is taking longer than expected). This might look like this (user input is in *italics*):

```
$ telnet localhost 4000
gilles: ..coe/teaching/u12_select> telnet localhost 4000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Tick: 60
Tick: 59
Tick: 58
07
Tick: 7
Tick: 6
Tick: 5
60
Tick: 60
Tick: 59
Tick: 58
```

```
2
Tick: 2
Tick: 1
Connection closed by foreign host.
$
```

Note that this means the server has to juggle two tasks: reading input from the user (across the network), and in the meantime sending out the ticks once per second.

For the moment, the server is single-user: it need only support one connected user at a time. However, it must support repeated sequential connections without exiting (except in the case of serious errors).

Additional requirements for the server are as follows:

- It should not be possible to crash the server from the client.
- The server should not leak memory.
- The server should not use much CPU. It must not run at 100% CPU utilization - it's only an eggtimer!
- The ticks should be regular – one per second – except when the user specifies a new time value. When this happens, the system should start a new tick immediately (much like resetting the seconds to zero on a digital clock when you set the time).

Furthermore, the server must be built under the following constraints:

- It must be written in C using only the standard C library and regular system calls.
- The server cannot use threads or fork processes: it must be completely single-threaded. This in practice means using an event-driven style with `select()`.

To make this easier, you can assume that `send()` is not going to block, i.e. the server can always immediately send data without waiting for the network to become "free". Consequently, you don't need to wait for writeable sockets.

For extra credit, structure your code so that it can handle the case where `send()` might block.

2 Multi-user server

For the second part, refactor your server so that it can handle an arbitrary number of clients simultaneously. Clients can disconnect and reconnect at any time.

Hints:

- You now have to juggle $2n + 1$ tasks on a single thread: reading from and writing to each client, and one handling additional new connection. Recall the discussion in the lectures about using an event loop...
- You may find it helpful to write a couple of convenience functions to manipulate and compare the timeout values used for `select()`.
- Define a structure to represent a client, and keep a list of active clients.