Assigments week 8

Joe Armstrong

February 17, 2014

Contents

1 Introduction

These problems are for lectures F10 and F11, held in week 8 of 2014.

Additional material can be found in Chapters 19 to 22 of the course book *Programming Erlang, 2'nd edition*.

2 Problems

The problems are devided into two categories:

- **normal** solve these to get *Godkänd*.
- **advanced** solve these to get *Väl Godkänd*.

The sections and subsections in this paper are marked appropriately.

3 normal: ETS

For these exercises you will need to download the file `country_codes.txt` (from http://www.`http://github.com/joearms/paradis`). Write a module called `country_codes.erl` with the following API:

`country:start() -> EtsTable.`
: Returns an ETS table containing country code data read from the file `country_codes.txt`.

`country:lookup(EtsTable, Code) -> {ok, FullName} | error.`
: Lookup the country with code `Code` in the country code ets table, Return the full name of the country if it is in the table, otherwise return `error`

For example:

```
$ erl
1> I = country_code:start()
...
2> country_code:lookup(I, "SE").
{ok,"Sweden"}
3> country_code:lookup(I, "JA").
error
```

4 advanced:ETS and DETS

Start by writing a function that returns a random name and a random telephone number. The name should be from 5 to 10 characters long, The telephone number should be from 6 to 12 digits long.

Hint: `crypto:rand_uniform(Lo, Hi) -> N` Generates a random number `N`, where `Lo =< N < Hi`.

Generate 1 million random names and telephone numbers and store them in a dets table. Measure how long time this takes.

5 normal: Mnesia

Create a mnesia database on your machine. Create a record to represent country codes. Write routines to initialize the database with the data in the file `country_codes.txt`. Write access routines to read and update the country code table.

6 Gen server exercises

These exercises involve using the `gen_server` module.

6.1 normal:A simple file tracker

Make a `gen_server` in the module tracker. The tracker keeps track of a set of client IPs who are interested in sharing a particular file.

The tracker API is as follows:

`tracker:start()`
: Starts the tracker.

`tracker:i_want(File, IP) -> [IP]`
: This means that the host with address IP wants or has the file called File, it returns a list of all IP addresses who are interested in this file.

`tracker:i_am_leaving(IP)`
: IP is no longer interested in any files.

`tracker:who_wants(File) -> [IP]`
Return a list of IPs who are currently interested in the file called File.

Here's a sample session with the tracker:

```
$ erl
1> tracker:start().
ok.
2> tracker:i_want(file1, "123.45.1.45").
["123.45.1.45"]

2> tracker:i_want(file1, "223.45.12.145").
["123.45.1.45", "223.45.12.145"].

3> tracker:who_wants(file2).
[]

4> tracker:who_wants(file1).
["123.45.1.45", "223.45.12.145"].

5> tracket:i_want(file3, "123.45.1.45").
...

6> tracker:i_am_leaving("123.45.1.45").
ok

7> tracker:who_wants(file1).
["223.45.12.145"].
```

Note: To solve this you will need to store lists of IP addresses that are associated with a particular file. I suggest you use the `dict` module to store the lists of IP addresses. `dict` has the following interface:

`dict:new() -> Dict`
Return a new dictionary:

`dict:store(Key, Value, Dict) -> NewDict`
Stores a `Key`, `Val` association in the dictionary.

`dict:find(Key, Dict) -> {ok, Value} | error`
: Looks up `Key` in the dictionary. This returns `error` if there is no item in the dictionary or `{ok, Value}`.

`dict:delete(Key, Dict) -> NewDict`
: Deletes a the item with `key` form the dictionary.

.

6.2 advanced: Adding timeouts

Add an additional function to the tracker API.

`tracker:ping(IP) -> ok`
: Calling `ping()` informs the tracker that the host `IP` is still interested in the file. Hosts who are interested in files should ping the tracker every ten seconds to tell the tracker that they are still alive. If a ping is not received every ten seconds the tracker should assume that the host has lost interest and the host IP address should be removed from all lists of active IPs.