CS 683 Emerging Technologies
Fall Semester, 2008
Doc 5 Sockets and Tables
Sept 16 2008

Copyright ©, All rights reserved. 2008 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/openpub/) license defines the copyright on this document.

References

Programming Erlang: Software for a Concurrent World, Armstrong, Chapters 14, 15 16.

Erlang Documentation

Erlang Compiler

Reading

Programming Erlang: Software for a Concurrent World, Armstrong,
Chapter 15, 16

Socket Programming

Socket Programming

```
-module (tcp).
-export ([start_server/0]).

start_server () ->
   {ok , ServerSocket} = gen_tcp:listen(2345, [binary, {packet, 0}, {reuseaddr, true}]),
   listen_for_request(ServerSocket).

listen_for_request(ServerSocket) ->
   {ok , ClientSocket}= gen_tcp:accept(ServerSocket),
   spawn(fun() -> listen_for_request(ServerSocket) end),
   process_request(ClientSocket).

process_request(Socket) ->
   receive
      {tcp, Socket, Binary} ->
         send_response(Socket, Binary);
      {tcp_closed, Socket} ->
         void
   end.
```

Part 2

```
send_response(Socket, Packet) ->
    try send_factorial(Socket, packet_to_integer(Packet))
    catch
        _:_ -> send(Socket, "Internal Sever Error")
    after
        gen_tcp:close(Socket)
    end.

send_factorial (Socket, Request) when Request >0 ->
    Result = local_math:factorial(Request),
    send(Socket, Result);
send_factorial (Socket, _Request) ->
    send(Socket, "Invalid Input").

send(Socket, Message) ->
    gen_tcp:send(Socket, lists:concat([Message , "\r\n"])).

packet_to_integer(Packet) ->
    {String , _Rest} = string:to_integer(binary_to_list(Packet)),
    String.
```

Using the Server

Erlang Shell

```
1> c(tcp).
{ok,tcp}
2> tcp:start_server().
ok
```

Telnet session

```
Al pro 42->telnet 127.0.0.1 2345
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
10
3628800
Connection closed by foreign host.
```

Tables

Tables & Database

ETS

Erlang Term Storage

qlc

SQL like Query language

DETS

Disk Erlang Term Storage

Mnesia

SQL like database

ETS - Erlang Term Storage

Table of Erlang tuples

First element of tuple is the key
- Can change if need be

Resides in memory

Extension to Erlang
- Can modify existing values
- Side-effects

Types of Tables

Set
- All keys must be unique
- Default

Ordered Sets
- All keys must be unique
- Types are sorted

Bags
- Keys can be repeated
- All tuples must be unique

Duplicate Bags
- Keys can be repeated
- Tuples can be repeated

Table Access Levels

Private

- Only the process that created table can access

Protected

- Any process can read table
- Only the process that created table can write
- Default

Public

- Anyone can read/write table

Basic Operations

```
ets:new(Name, Options)
ets:insert(Table, Tuple)
ets:lookup(Table, Key)
ets:match(Table, Pattern)
ets:update_element(Table, Key, {Position, Value})
ets:to_dets(Table, DestinationTable)
ets:from_dets(Table, DestinationTable)
ets:delete(Table)
ets:delete(Table, Key)
```

Example

```
-module (ets_example).
-export ([test_insert_access/0]).

test_insert_access () ->
    TableId = ets:new(test, [set]),
    ets:insert(TableId, {a,1}),
    ets:insert(TableId, {a,2}),
    ets:insert(TableId, {b,1}),
    ets:insert(TableId, {5,"cat"}),
    ets:insert(TableId, {dog,12,"zoo"}),
    List = ets:tab2list(TableId),
    io:format("Initial table ~p~n", [List]),
    ets:delete(TableId, a),
    io:format("After delete ~p~n", [ets:tab2list(TableId)]),
    Lookup = ets:lookup(TableId, b),
    io:format("Lookup ~p~n", [Lookup]),
    ets:delete(TableId).
```

```
1> ets_example:test_insert_access().
Initial table [{b,1},{a,2},{dog,12,"zoo"},{5,"cat"}]
After delete [{b,1},{dog,12,"zoo"},{5,"cat"}]
Lookup [{b,1}]
true
```

Bag & Match

```
test_bag () ->
    TableId = ets:new(test, [bag, private, named_table]),
    ets:insert(test, {a,1}),
    ets:insert(test, {a,2}),
    ets:insert(TableId, {b,1}),
    ets:insert(test, {5,"cat"}),
    ets:insert(TableId, {dog,12,"zoo"}),
    List = ets:tab2list(test),
    io:format("Initial table ~p~n", [List]),
    Match = ets:match(test, {'_', '$1'}),
    io:format("Match ~p~n", [Match]),
    Match2 = ets:match(test, {'$1', 1}),
    io:format("Match2 ~p~n", [Match2]),
    ets:delete(TableId).
```

```
31> c(ets_example).
{ok,ets_example}
32> ets_example:test_bag().
Initial table [{b,1},{a,1},{a,2},{dog,12,"zoo"},{5,"cat"}]
Match [[1],[1],[2],["cat"]]
Match2 [[b],[a]]
true
```

Match

```
ets:match(test, {'_', '$1'})

ets:match(test, {'$1', 1})
```

'_'
matches any value in that position

'$N'
When tuple matches return value in N'th location

atom, bound variable
Match the give value

Update

```
test_update () ->
    TableId = ets:new(test, [set, private, named_table]),
    ets:insert(test, {a,1}),
    ets:insert(test, {a,2}),
    ets:insert(test, {b,1}),
    ets:insert(test, {5,"cat"}),
    ets:insert(TableId, {dog,12,"zoo"}),
    List = ets:tab2list(test),
    io:format("Initial table ~p~n", [List]),
    ets:update_element(test, dog, {3,"mouse"}),
    Lookup = ets:lookup(TableId, dog),
    io:format("Update ~p~n", [Lookup]),
    ets:update_counter(test, b,5),
    Lookup2 = ets:lookup(test, b),
    io:format("Update Counter ~p~n", [Lookup2]),
    ets:delete(test).
```

```
41> ets_example:test_update().
Initial table [{b,1},{a,2},{dog,12,"zoo"},{5,"cat"}]
Update [{dog,12,"mouse"}]
Update Counter [{b,6}]
true
```


update_element only works on sets

Equal verse Identical

Equal
(Value comparing)

1> 1 == 1.0.
true
2> 1 == 1.
true

Identical
(matching)

1> 1 =:= 1.0.
false
2> 1 =:= 1.
true

Ordered Sets

Set that are ordered by value

1 and 1.0 are considered same value

QLC

Works with ETS, DETS, Mnesia

Examples

```
-include_lib("stdlib/include/qlc.hrl").
test_qlc() ->
    create_table(),
    Query = qlc:q([X || X <- ets:table(test)]),
    qlc:eval(Query).

test_qlc2() ->
    create_table(),
    Query = qlc:q([{X,Y} || {X,Y} <- ets:table(test), Y /= "zoo", Y < 2]),
    qlc:eval(Query).

create_table() ->
    ets:new(test, [set, public, named_table]),
    ets:insert(test, {a,1}),
    ets:insert(test, {a,2}),
    ets:insert(test, {b,1}),
    ets:insert(test, {5,"cat"}),
    ets:insert(test, {dog,12,"zoo"}).
```

Queries that Don't Run

```
qlc:q([{X,Y} || {X,Y} <- ets:table(test),  Y > 2])
qlc:q([{X,Y} || {X,Y} <- ets:table(test),  Y < 2, X =:= a])
```

DETS

File based tables

Table name mapped to atom - globally accessible

Some Options

access
- read_write, read
- read_write default

auto_save
- infinity, int()
- 3 minutes default

file

max_no_slots

min_no_slots

keypos

ram_file

type
- bag, duplicate_bag, set

Basic Operations

dets:open_file(name, [Arguments])
dets:insert(Table, Tuple)
dets:lookup(Table, Key)
dets:match(Table, Pattern)
from_ets(DetsTable,EtsTable)
dets:delete(Table, Key)
dets:close(Table)
dets:all()

Example

```
test_dets () ->
    case dets:open_file(test, [{file,"Database"}]) of
        {ok,_Table_name} ->
            ok = dets:insert(test, {a,10}),
            ok = dets:insert(test, {b, 12});
        {error, _Reason} ->
            io:format("Cannot open dets table")
    end,
    EtsTable = dets:to_ets(test,ets:new(dontCare,[private]) ),
    io:format("Initial table ~p~n", [ets:tab2list(EtsTable)]),
    dets:close(test).
```

```
47> ets_example:test_dets().
Initial table [{b,12},{a,10}]
ok
```

Tables are Globally Accessable

```
test_dets_open() ->
    dets:open_file(test, [{file,"Database"}]),
    dets:open_file(grades, [{file,"cs683"}]).
```

```
49> ets_example:test_dets_open().
{ok,grades}
50> dets:all().
[grades,test]
51> dets:lookup(test,b).
[{b,12}]
52> dets:close(test).
ok
53> dets:close(grades).
ok
```

Mnesia

Distributed database management system

SQL like queries

Stores Erlang data structures

Example Table

grades

name	test	final
roger	50	25
avi	78	68

Database and Table setup

mnesia_example.erl

```
-module (mnesia_example).
-compile(export_all).

-include_lib("stdlib/include/qlc.hrl").

-record(grades, {name,test=0, final=0}).

do_this_once() ->
    mnesia:create_schema([node()]),
    mnesia:start(),
    mnesia:create_table(grades, [{attributes, record_info(fields, grades)}]),
    mnesia:stop().
```

Some Explanation

mnesia_example.erl

```
-module (mnesia_example).

-compile(export_all).        %development short cut to export everything

-include_lib("stdlib/include/qlc.hrl").
     % needed for the query language

-record(grades, {name,test=0, final=0}).
     % Erlang record - like tuple with name-value pairs
     %      grades is the name of the record
     %      has three name-value pairs
     % Used to define table
     % test & final have default values
```

Some Explanation

mnesia_example.erl

```
do_this_once() ->
   mnesia:create_schema([node()]),
         %Creates database for just the current Erlang node

   mnesia:start(),
   mnesia:create_table(grades, [{attributes, record_info(fields, grades)}]),
         % record_info fake function compiled in to modules with records
         %       provides information about the record

mnesia:stop().
```

Running Startup

```
57> c(mnesia_example).
{ok,mnesia_example}
58> mnesia_example:do_this_once().
stopped

=INFO REPORT==== 15-Sep-2008::22:13:04 ===
    application: mnesia
    exited: stopped
    type: temporary
59>
```

```
Al pro 15-> ls
mnesia_example.beam
mnesia_example.erl
Mnesia.nonode@nohost/

Al pro 16->ls Mnesia.nonode@nohost/
LATEST.LOG      schema.DAT
```

Start Stop

```
start() ->
    mnesia:start(),
    mnesia:wait_for_tables([grades], 20000).
      %Wait until checks with all nodes

stop() ->
    mnesia:stop().
```

Inserting

```
add_grade(Name, Test, Final) ->
    Row = #grades{name=Name, final = Final, test = Test},
    Add = fun () ->
            mnesia:write(Row)
        end,
    mnesia:transaction(Add).
```

Simple Select

```
select_all() ->
    do(qlc:q([X || X <- mnesia:table(grades)])).

do(Q) ->
    F = fun() -> qlc:e(Q) end,
    {atomic, Val} = mnesia:transaction(F),
    Val.
```

Sample Run

```
62> mnesia_example:start().
ok
63> mnesia_example:add_grade("roger",50,25).
{atomic,ok}
64> mnesia_example:select_all().
[{grades,"roger",50,25}]
66> rr("grades.hrl").
[grades]
67> Result = mnesia_example:select_all().
[#grades{name = "roger",test = 50,final = 25}]
68>
```

Records and Shell

grades.hrl

-record(grades, {name,test=0, final=0}).