Funkcionális és Párhuzamos Programozási technológiák

Tóth Melinda

2011/2012 tavaszi félév

Funkcionális programok

- Típus-, osztály- és függvénydeklaráció
- Kezdeti kifejezés
- Végrehajtás
- Matematikai modell – Lambda kalkulus
- Turing teljes

Jellemzők

- Hivatkozási helyfüggetlenség
- (Szigorú/statikus típusosság)
- Magasabbrendű függvények
- (Curry módszer – részleges függvényalkalmazás)
- Függvény alkalmazás önmagukra
- (Lusta)/mohó kiértékelés
- Zermelo-Frankel halmazkifejezések
- Argumentumok mintaillesztés
- (Margószabály)
- IO modell

Történet

- 1982 - 1987 – útkeresés
- 1988 - 1990 – tapasztalat, prototípus telekommunikációs szoftverek
- 1993 – elosztott programozás / Első könyv (The BOOK)
- 1996 – OTP R1
- 1998 – nyilvános
- 2005 – R11 multicore

Erlang – Tulajdonságok

- Deklaratív – Funkcionális, magas absztrakció, olvasható
- Dinamikusan típusos
- Beépített párhuzamosság – átlátszó, explicit párh., LWP
- Valós-idejű alkalmazások
- Robosztus – supervison trees
- Elosztott – átlátszó, explicit, hálózati
- Integrálhatóság, nyíltság – “port”-ok
- Portolhatóság – Unix, Win., ... , heterogén hálózat
- SMP támogatás – multicore
- “Hot code loading”

Erlang – Ericsson Language

- Erlang, Agner Krarup (1878-1929)
- dán matematikus
- Erlang formula
- erlang – forgalom intenzitása

Mikor?

- Komplex, leállási idő nélküli, skálázható, karbantartható, elosztott
- Gyors és hatékony fejlesztés
- Nagy hibatűrő (szoftver, hardver) rendszerek
- Hot-code loading

Kik?

- Ericsson – telekommunikáció (pl. AXD301 ATM switch), szimuláció, tesztelés, 3G, GPRS
- Amazon – Simple DB adatbázis
- Yahoo – könyvjező
- Facebook – chat server
- T-Mobile – SMS gateway
- Motorola – hívás feldolgozás
- MochiWeb – http szerver
- CouchDb – dokumentum adatbázis szerver (multicore, multiserver clusters)
- YAWS – Yet Another Web Server
- Wings3D – 3D-s modellező

Erlang shell

- erl (erl -noshell)
- 1 + 2. "alma".
- q(). – init:stop().
- BREAK menu: Ctrl-C / Ctrl-Break
- User Switch Command: Ctrl-G

Hasznos shell parancsok

- help() / h()
- i()
- memory()
- c(ModName)
- ls() / ls(Dir)
- b()
- f() vagy f(X)
- e(Number) vagy e(-1)
- v()
- modulenév:függvény(Params)
- m(ModName) vagy modulnév:module_info()
- pwd() / cd(Path)

Típusok – Termek

- Numbers (Integer, Float)
- Binaries/Bitstrings
- Atoms
- Tuples
- Lists (Strings)
- Unique identifiers: pids, ports, references
- Funs

Számok

- Integer – 10, 2#10101, 36#PQ3, $w
- Floats – 0.01, 17.2, 11.12E-10
- Aritmetika: +, -, *, /, div, rem
- math -> sqrt
- N bsl K, N bsr K
- band, bor, bxor, bnot

Binaries & Bitstrings

- Binary: byte-ok sorozata
- Bitsrting: bit-ek sorozata
- `<<>>, << 0,1,2,3>>`
- `<< "hello", 0, "almafa">>`

Atom

- String constant
- Atom – lower + letter + digit + @ + . + _
- 'An atom'
- %% Boolean – true, false
- ok & undefined
- Max 255 hosszú, Max 1048576 db.

Tuple

- Rögzített hosszú rendezett n-es
- Tuple – {...}
- {alma, korte, {1,2}, "almafa"}
- {}
- Tagged tuple: {int, 42}, {pos, 23, 43}
- element(n Tuple)

List

- Változó hosszú adatszerkezet
- List – [...], String – "..."
- `[], "", [1,23, \{alma, 2\}, [egy, [ketto]]]`
- `[1 | []], [1 | [2]], [1,2,3 | [4,5]]`
- `[1,2] ++ [3,4,5]`
- `[97,98,99,100] == "hello" == [$a, $b, $c, $d]`
- ??? `[0 | v(Num)]` ???
- Proper & improper lists

Unique identifiers and Funs

- Pid – `< 0.4.2>`
- Port – `#Port<0.472>`
- Reference – `#Ref<0.0.0.42>`
- Fun – `#Fun<...>`

Termek

- Integer – 10 (számrendszer, karakter)
- Floats – 17.2, 11.12E-10
- Binaries and Bitstrings
- Atom – lower + letter + digit + @ + . + _
- %% Boolean – true, false
- Tuple – {...}
- List – [...] , String – "..."
- Egyedi azonosítók: pid, port, reference
- Fun

Összehasonlítás

number < atom < fun < port < pid < tuple < list < binary

- '<' '>'
- '=<' '>='
- '/=' '=='
- '=:=' '=/='

Emacs – példa

```
(add-to-list 'exec-path "/path_to_erlang/bin")
(add-to-list 'load-path "/path_to_erlang/lib/tools-2.6.5/emacs")
(require 'erlang-start)
```

Modulok

- .erl kiterjesztés
- `-module(név).`
- formokból állnak
- '.'

Függvények – ModName:FunName/Arity

```
name(Arg11, ..., Arg1N) [when Guard1] ->
    ExprList1;
name(Arg21, ..., Arg2N) [when Guard2] ->
    ExprList2;
...
name(ArgM1, ..., ArgMN) [when GuardM]->
    ExprListM.

Exprlist ::=  Expr1, Expr2, ... ExprK.

modname:funname(Par1, ..., Parn)
        funname(Par1, ..., Parn)
```

Egyszerű függvények

```
fact(0) -> 1;
fact(N) when N>0 ->
    N * fact(N-1).

fib(1) -> 1;
fib(2) -> 1;
fib(N) when N>2 ->
    fib(N-1)+fib(N-2).
```

További elemek

- Attribútumok – module, export, import, include, compile, behaviour, user_defined
- Makrók
- Rekordok
- Header fájlok
- Comment - %

Fordítás és betöltés

- c(ModName)
- l(ModName)
- code:get_path()
- erlc

BIF-s

- erlang modulban vannak
- amit nem vagy csak nehezen lehetne megvalósítani
- a rendszer viselkedését változtatja
- stb...
- hd/1, tl/1, length/1, size/1, element/2, setelement/2, list_to_tuple/1, ...

Dokumentáció és könyvtári modulok

- array, lists, dict, string, queue
- erlang
- io
- file, filename
- random, math
- Lásd: http://www.erlang.org/doc/man/

Változók

- Nagy betűvel vagy aláhúzással kezdődik
- Nem változik
- Nem kell definiálni
- Érték szerint paraméterátadás
- Függvény - lokális változó
- ??? Globális ???

Pattern Matching

- Esetszétválasztás
- Változó érték kötés
- Adatszerkezetek felbontása
- $X = 2$
- $\{A, A, X\} = \{1, 1, 3\}$
- $[Head|Tail] = [1, 2, 3]$

Listák

- http://www.erlang.org/doc/man/
- '- -' és '++'
- length/1, tl/1, hd/1
- lists: {max/1, sum/1, nth/2, last/1, reverse/1, member/2, delete/2, sort/1, usort/1, zip/2, split/2}
- proplists
- litsagenerátorok: [hogyan? || honnan? , milyen?]

```
['elemek' || generátor1, ..., filter1, ...]
generátor:  Var <- List
filter:  logikai értékű
```

Listák

Gyorsrendezés (QuickSort)

```
quicksort([X || X <- List, X =< Y])
   ++ [Y]
      ++ quicksort([X || X <- List, X > Y])
```

Rekurzió

- direct recursion: sum
- tail-recursion: sum_acc

Feltételes vezérlés

- case Expr of
  Pattern1 [when Guard1] -> ExprList1;
  ...
  PatternN [when GuardN] -> ExprListN
  end.
- if
  Guard1 -> ExprList1;
  ...
  GuardK -> ExprListK
  end.
- változó láthatóság
- 'true' és '_'

Őrfeltételek

- ',' és ';'
- kötött változó
- literál
- összehasonlítás
- aritmetikai művelet
- logikai operátor
- típus vizsgálat
- guard built in functions (free of side effects)

Függvény kifejezések

```
fun (Patterns1) [when Guard1] -> ExprList1;
...
    (PatternsN) [when GuardN] -> ExprListN
end

fun ModName:FunName/Arity
```

Feladatok

Listagenerátor, map, filter, zip, flatten, függvény kifejezés segítségével:

- egy listában lévő számok osztói (map + fun + listgen)
- szám és osztói párok (zip)
- listában lévő számok közöl a primeket (filter + fun)

Dinamikus konstrukciók

- apply/2 – apply(FunExpr, [Params])
- apply/3 – apply(ModName, FunName, [Params])
- Fun(Params)
- Mod:Fun(Params)

Futásidejű hibák

- function_clause
- case_clause
- if_clause
- badmatch
- badarg
- undef
- badarith

Hibakezelés

```
try ExprList of
  Pattern1 [when Guard1] -> ExprList1;
  ...
  PatternN [when GuardN] -> ExprListN
catch
  Class1:ExcPattern1 [when ExcGuard1] -> ExcExprList1;
  ...
  ClassK:ExcPatternK [when ExcGuardK] -> ExcExprListK
after
  ...
end
```

Classes: error, throw, exit

```
catch Expr
```

Rekordok

- `-record(name,`
  `{field1 [= default1],`
  `...,`
  `fieldn [= defaultn]}).`
- `RecordExpr#name.field`
- `RecordExpr#name{..., fieldi = NewValue, ...}`
- Mintaillesztés rekordokra
- `rr/1, rd/2, rl/1, rf/1`
- BIF: `record_info/2 (size, fields)`, `is_record/2`

Makrók

- `-define(Name, Replacement)`
- `?Name`
- `-define(Name(Var1, ..., VarN), Replacement)`
- `?Name(Var1, ..., VarN)`
- ??Var
- ?MODULE, ?MODULE_STRING, ?FILE, ?LINE
- `-include("valami.hrl")`

Hibajavítás és makrók

- `-undef(Flag).`
- `-ifdef(Flag).`
- `-ifndef(Flag).`
- `-else.`
- `-endif.`
- c(Module, [{d, debug}]), c(Module, [{u, debug}])

“Lusta” lista

```
next(Seq)->
  fun()-> [Seq | next(Seq + 1)] end.

SeqFun0 = next(0).
[Seq1 | SeqFun1] = SeqFun0().
...
```

Bitstring and binary

- `<< Segment1, ...., SegmentN>>`
- Segment: `Data, Data:Sizerrr, Data:TypeSpecifier, Data:Size/TypeSpec`
- TypeSpec: `{integer, float, binary, bytes, bitstrings, bits,} {utf8, utf16, utf32} - {signed, unsigned} - {big, little, native}`
- pattern matching: `<<Var1:4/bits, Remaining>>`
- `<< <<X:3>> || X <- [1, 2, 3, 4, 5, 6, 7]>>`
- `<< <<X:8>> || <<X:3>> <= <<41, 203, 23:5>> >>`
- `[ X || <<X:3>> <= <<41, 203, 23:5>> ]`
- `[ <<X>> || <<X:3>> <= <<41, 203, 23:5>> ]`

Input and Output

- io module
- get_line("> ")
- get_chars("> ", 2)
- read("ok, then »") – Erlang term
- write/1
- format(FormatString, [Values])

Formázás

- ' c' – ASCII kód karakterként
- ' f' – float hat decimális pontossággal
- ' e' – float lebegő pontosan hat decimális pontossággal
- ' w' – Erlang term standard szintaxissal
- ' p' – Erlang term 'pretty print'-elve
- ' B' – a számokat 10-es számrendszerben írja ki
- ' W' és ' P' – a beágyazott szerkezetekhez mélységi korlát

File-kezelés

- *file* - open, close, read, write, list dirs
- *filename* - file nevek kezelése (platformfüggetlen)
- *filelib* - kiterjesztés a file modulhoz
- *io* - megnyitott fájlok tartalmának kezelése, formázott szöveg illesztése a fájlba
- {ok, Dev} = file:open(File, [Mode]), file:close(Dev)
- Mode: read, write, append, exclusive, binary, etc.

File-kezelés (folyt.)

Erlang termek beolvasása:

- file:consult("file").
- io:read(Dev, Promt)
- io:read(Dev, Prompt, StartLine)

Sorok beolvasása

- io:get_line(Dev, Prompt)
- io:fread(FileDescr, Prompt, FormatString)
- `~d`, `~u`, `~-`, `~f`, `~#`, `~s`, `~a`, `~c`, `~l`, whitespaces, `~~`

Beolvasása binary-be:

- file:read_file("file")
- io:pread(Dev, Start, LenB), file:close(Dev)

File-kezelés (folyt.)

Erlang termek írása:

- io:format(Dev, FormatString, DataList)
- `~n`, `~s`, `~p`, `~w`, whitespaces `~10.2s`

Byteok írása egy fájlba

- file:write(Dev, Bytes)
- file:write_file(Filename, Bytes) – létre is hozza, ha kell
- file:pwrite(Dev, [{Loc, Bytes}]), file:pwrite(Dev, Loc, Bytes)
- Loc: bof, eof, cur

Alapfogalmak

Process

- aktor aki a saját feladatát végrehajta: saját kód, saját térben
- nem osztanak meg semmit
- saját állapot
- üzenetek másolásával kommunikálhatnak

Alapfogalmak

Kommunikációs modellek

- Shared memory + lock
- Software transactional memory (STM)
- Futures and Promises
- Message passing: szinkron, aszinkron

Alapfogalmak

Ping-pong

```
run() ->
  Pid = spawn(fun ping/0),
  Pid ! self(),
  receive
    pong -> ok
  end.

ping() ->
  receive
    From -> From ! pong
  end.
```

Processzek

- spawn/3, spawn_link/3, etc.
  Pid = spawn(MilyenMod, MilyenFv, [MilyenArg])
- Pid ! Msg
  Pid ! {msg, "Uzenem, hogy itt a veg"}
- processes/1
- self/0, pid/3
- i/1, flush/0

Alapfogalmak

Process link és hibakezelés

- `link/1, spawn_link/3`
- exit signal ha a terminál a process – normal vagy non-normal
- `process flag(trap_exit, true)`
- `{'EXIT', Pid, Reason}` üzenet érkezik
- `unlink/1`
- `exit(Reason), exit(Pid, Reason)` – normal, kill, other
- supervision

Üzenet fogadás

```
receive
  Pattern1 [when Guard1] -> ExprList1;
  ...
  PatternN [when GuardN] -> ExprListN
end
```

Regisztrálás

- register(Alias, Pid)
  register(alma, Pid)
  alma ! {msg, "Uzenem, hogy itt a veg"}
- unregister(Alias)
- registered()
- whereis(Alias)
- regs()

Timeout

```
receive
  Pattern1 [when Guard1] -> ExprList1;
  ...
  PatternN [when GuardN] -> ExprListN
after
  MiliSecond -> ExprListN
end

'infinity'
```

Elosztott Erlang node-ok

- Indítás: erl -name korte@host | -sname korte -setcookie alma
- node(), nodes()
- Uzenetküldés: {Name, Node} ! {msg, "Uzenem, hogy itt a veg"}
- Kapcsolat: net_adm:ping(Node), monitor_node(Node, true)
- ps ax | grep -i epmd
- "Magic Cookie": get_cookie(), set_cookie(node(), "alma")
- Remote shell: Ctrl-G, r, c

További érdekességek

- pman:start()
- alarm, sleep, flush
- Ring – timer:tc(Mod, Fun, Args)
- skeleton – start -> initialise -> receive-eval-loop -> (stop) -> terminate
- Kliens-szerver: chat program és finomításai

Software Upgrade

- Újrafordítás után a futó alkalmazás kódjának frissítése
- Tárolt régi változatok, amíg valaki használja
- Modulminősítővel ellátott hivatkozások
- `code:load_file(Module)`
- Code Server
  - `code:purge(Module)`
  - `code:soft_purge(Module)`
  - `code:get_path()`
  - `code:add_path*(Path)`

Erlang Term Storage – ETS

- %% Az osztott memória egy formája
- Nagy méretű adatok tárolása és elérése kulcsok alapján
- Konstans elérési idő
- Nem KV Database, nincs tranzakciókezelés
- `TableId = ets:new(TableName, [Options])`
- Options: named_table, set, bag, ordered_set, duplicate_bag, private, protected, public, keypos, Key, (read)write_concurrency
- `ets:delete(TableId)`
- `ets:insert(TableId, Key, Value)`
- `ets:lookup(TableId, Key)`
- `ets:delete(TableId, Key)`

Erlang Term Storage – ETS

- `ets:first(TableId), ets:next(TableId), '$end_of_table'`
- `ets:match(TableId, Pattern) - $1, $0`
- `ets:match_object(TableId, Pattern)`
- `ets:delete_object(TableId, Pattern)`
- `ets:select(TableId, MatchSpec)`

OTP

- Application & Libraries, pl:
  - Alap (stdlib, kernel, runtime systems)
  - Adatbázis (Mnesia, ODBC)
  - Interfész és kommunikáció (Java/C interface, SSH, SSL, XML parsing, Inets)
  - Különböző alkalmazások (Pman, TV, Debugger, Edoc, Syntax tools, Dialyzer)
- System design principles

OTP behaviours

- Tervminták formalizálás (“Design Pattern”)
- Folyamatok hasonló stuktúrával és életciklussal rendelkeznek (start, receive-send, terminate (crash))
- Általános és specifikus részek
- Callback module – server – generic behaviour module

Példa – Kliens-szerver

Általános

- Szerver spawn
- LoopData tárolása
- Kérés küldés a szervernek
- Válasz küldés a kliensnek
- Válasz fogadás a szervertől
- Szerver leállítás

Specifikus

- Szerver inicializálás
- LoopData
- Kliensek kérése
- Kliensek kérdéseinek kezelése
- Szerver válaszainak tartalma
- Végső helyreállítás, megtisztítás

OTP behaviours

- Generic Servers
- Generic Finite State Machine
- Generic Event Handler/Manager
- Supervisors
- Applications

Pro és kontra

Mellette

- Kevesebb megírandó forráskód
- Kevesebb bug
- Tesztelt alap
- Beépített funkcionalitások (log, trace, statistics)
- Könnyű új funkcionalitást felvenni
- Közös progmaozási stílus
- Komponens alapú terminológia

Ellene

- Meg kell ismerni
- Teljesítménybeli hatás

Generic Servers

- Kliens-szerver
- gen_server.erl tartalmazza az általános részt
- call back modult írunk a specifikus rész implementálására
- -behaviour(gen_server).

Starting a Server

```
gen_server:start_link(SrvName, CBModName, Args, Opts)
gen_server:start(SrvName, CBModName, Args, Opts)
gen_server:start_link(CBModName, Args, Opts)
gen_server:start(CBModName, Args, Opts)
```

- {local, Name}, {global, Name}
- CBModName:init(Args) -> ok, LoopData

Passing Messages

```
gen_server:call(Name, Message)
gen_server:call(Name, Message, Timeout)
gen_server:cast(Name, Message)
gen_server:cast(Name, Message, Timeout)
```

- Name, {global, Name}
- szinkron – visszatérési érték Reply
- aszinkron
- nincs handle_call/cast ???

```
handle_call(Message, From, LoopData) ->
  ...
  {reply, Reply, NewLoopData}.
handle_cast(Message, LoopData) ->
  ...
  {noreply, NewLoopData}.
```

Stopping the Server

```
handle_call(Message, From, LoopData) ->
  ...
  {stop, Reason, Reply, NewLoopData}.
handle_cast(Message, LoopData) ->
  ...
  {stop, Reason, NewLoopData}.
```

- handle_call/cast hívással
- Reason: normal (ritkán más egyébb)
- CBModName:terminate(Reason, LoopData)

-behaviour(gen_server)

- handle_info(Msg, LoopData) ...
- code_change/1

Supervisors

```
supervisor:start_link(SrvName, CBModName, Args)
supervisor:start(SrvName, CBModName, Args)
supervisor:start_link(CBModName, Args)
supervisor:start(CBModName, Args)
```

- CBModName:init(Args)
- Returns: {ok, {SupSpec, ChildSpecList}}
- SupSpec: {RestartStrategy, AllowedRestarts, MaxSec}
- RestartStrategy: one_for_one, one_for_all, rest_for_one

Supervisors: ChildSpecList

- {ok, {SupSpec, ChildSpecList}}
- ChildSpec: {Id, {Mod, Fun, Args}, Restart, Shutdown, Type, ModList}
- {Mod, Fun, Args} -> start_link
- Restart: transient, temporary, permanent
- Shutdown: millisec, infinity, brutal_kill
- Type: worker, supervisor
- check_childspecs/1

Supervisors: Dynamic children

- supervisor:start_child/2
- supervisor:terminate_child/2
- supervisor:restart_child/2
- supervisor:delete_child/2

Applications

- app_name.app
- application:start(AppName)
- Application Monitor

NIF – Native Implemented Function

- Függvények C-ben (C++) implementálva
- `so, dll`-re fordítva be kell tölteni az Erlang VM-be
- `gcc -o nif.so -fpic -shared nif.c -I /usr/local/lib/erlang/usr/include/`
- `-on_load(Fun/Arity)`
- ErlNifFunc: {ErlFunName, Arity, CFunName}

Web-servers

- Inets – Lightweight HTTP server built into Erlang
- Yaws (Yet Another Web Server) – HTTP server developed by Claes "Klacke" Wikstrom
- Mochiweb – HTTP server developed by Bob Ippolito/MochiMedia
- Webmachine – HTTP resource server developed by Basho Technologies (runs on Mochiweb under the hood

Database support

- Mnesia
- Tokio Cabinet, Kyoto Cabinet
- CauchDB
- MongoDB
- Hibari
- ODBC-vel MySQL

Port

- Kommunikáció a “külvilággal”
- “Külvilág”: egy OS process-ben futó külső program
- Üzenet fogadás/küldés: standard input/output
- Byte-orientált kommunikáció
- `open_port(PortName, PortSettings) - {spawn, Command}, {packet, N}`
- `{Pid, command, Data}` – adatküldés a portnak
- `{Pid, close}, {Pid, {connect, NewPid}}`
- `{Port, {data, Data}}` – üzenet a Porttól
- `{Port, closed}, {Port, connected}, {'EXIT', Port, Reason}`
- BIF: `port_command/2, port_close/1, port_connect/2, ports/0, port_info/2`

Kapcsolat más nyelvekkel

- ErlInterface
- jinterface (Jungerl)
- OTP.NET