**بنام خداوند جان و خرد**

**برنامه نویسی Erlang**

**مدیریت فرایند‌های توزیع‌شده و مقاوم در برابر خطا**

**مؤلف:**

**آرمان شناور**

**آدرس ایمیل مؤلف : arman.shenavar.erlang@gmail.com**

فهرست مطالب

[معرفی این کتاب: **Error! Bookmark not defined.**](#_Toc174263667)

[موارد گنجانده شده در این کتاب: **Error! Bookmark not defined.**](#_Toc174263668)

[موضوعاتی که در این کتاب آورده نشده است: **Error! Bookmark not defined.**](#_Toc174263669)

[در ادامه مواردی را که می تواند، موضوع کتاب های دیگری شود، بیان شده است: **Error! Bookmark not defined.**](#_Toc174263670)

[فصل 1 : معرفی ارلنگ 10](#_Toc174263671)

[1-آشنایی با ارلنگ 10](#_Toc174263672)

[تاریخچه ارلنگ : 10](#_Toc174263673)

[1-1 فرایند ها در ارلنگ : 10](#_Toc174263674)

[1-2 مزایای استفاده از ارلنگ : 10](#_Toc174263675)

[1-2-1 فرایند های مستقل از سیستم عامل: 11](#_Toc174263676)

[1-2-2پردازش موازی (parallel processing): 11](#_Toc174263677)

[1-2-3 همزمانی سبک وزن(concurrency): 11](#_Toc174263678)

[1-2-4 داده تغییر ناپذیر (immutable data): 11](#_Toc174263679)

[1-2-5مزایای متغیر تغییر ناپذیر: 11](#_Toc174263680)

[1-3 تطبیق الگو(pattern matching): 11](#_Toc174263681)

[1-4 مقاومت در برابر خطا(fault-tolerant): 11](#_Toc174263682)

[1-5 مقیاس پذیری (Scalability) : 12](#_Toc174263683)

[1-6 توزیع شده (distributed): 12](#_Toc174263684)

[1-7 تعویض داغ (Hot Swapping) : 12](#_Toc174263685)

[1-8 بدون توقف : 12](#_Toc174263686)

[1-9 اجتناب از حافظه اشتراکی : 12](#_Toc174263687)

[1-10 ارلنگ برای چه کارهای مناسب است : 12](#_Toc174263688)

[1-11 نصب پوسته ارلنگی در ویندوز 12](#_Toc174263689)

[1-11-2 نصب ارلنگ در سیستم عامل های– mac os – Free BSD Ubuntu/Debian و Fedora 14](#_Toc174263690)

[1-11-3 نصب پلاگین ارلنگ در eclipse در ویندوز 14](#_Toc174263691)

[1-12 تعدادی از فرمان های پوسته: 20](#_Toc174263692)

[فصل 2 : مفاهیم اساسی 22](#_Toc174263693)

[2-1 BIF 22](#_Toc174263694)

[2-2 اعداد صحیح: 22](#_Toc174263695)

[2-3 اعداد شناور و عملگر های ریاض 23](#_Toc174263696)

[2-4 تاپل ها tuples 24](#_Toc174263697)

[2-5 اتم ها : 26](#_Toc174263698)

[2-6 رشته ها: 26](#_Toc174263699)

[2-7 عملگر های مقایسه ای 27](#_Toc174263700)

[2-8 متغیر 28](#_Toc174263701)

[2-9 ساختار های داده پیچیده 29](#_Toc174263702)

[2-10 تطبیق الگو 30](#_Toc174263703)

[2-11 متغیر بی اهمیت (یا اهمیت نده) don’t care 30](#_Toc174263704)

[2-12 توابع 31](#_Toc174263705)

[2-13 بولین 31](#_Toc174263706)

[فصل 3 : ماژول ها و توابع 33](#_Toc174263707)

[3-1 ماژول ها 33](#_Toc174263708)

[3-2 نام های رزرو شده 34](#_Toc174263709)

[3-3 fun ها 34](#_Toc174263710)

[3-4 توابع 36](#_Toc174263711)

[3-5 ویژگی های ماژولی (module attribute) 36](#_Toc174263712)

[3-6 اعلان -import() چیست ؟ 36](#_Toc174263713)

[3-7 اعلان export 38](#_Toc174263714)

[3-8 اعلان module 38](#_Toc174263715)

[3-9 اعلان compile 38](#_Toc174263716)

[3-10 ویژگی های تعریف شده توسط کاربر: 41](#_Toc174263717)

[فصل 4 : ماکرو ها ، map ها ، رکورد ها و فایل های include 42](#_Toc174263718)

[4-1 macro 42](#_Toc174263719)

[4-1-1 ماکرو های از پیش تعریف شده ارلنگ : 43](#_Toc174263720)

[4-1-2 جریان کنترلی در ماکرو ها: 44](#_Toc174263721)

[4-1-3 ماکرو ها و گارد ها: 46](#_Toc174263722)

[4-2 فایل های include: 46](#_Toc174263723)

[فرق بین -include\_libو -include 46](#_Toc174263724)

[4-3 map: 47](#_Toc174263725)

[4-3-1 BIF های map: 48](#_Toc174263726)

[4-4 Record ها: 50](#_Toc174263727)

[4-4-1 چرا باید از رکورد ها استفاده کنیم؟ 50](#_Toc174263728)

[4-4-2 رکورد ها در عمل: 51](#_Toc174263729)

[فصل 5 : لیست و مفاهیم لیستی: 56](#_Toc174263730)

[5-1 در ادامه توابع کاربردی تر ماژول list را معرفی می کنیم: 57](#_Toc174263731)

[5-1-1 تابع lists:all 57](#_Toc174263732)

[5-1-2 تابع lists:any 58](#_Toc174263733)

[5-1-3 تابع lists:delete 58](#_Toc174263734)

[5-1-4 تابع lists:droplast 58](#_Toc174263735)

[5-1-5 تابع lists:duplicate 58](#_Toc174263736)

[5-1-6 تابع lists:last 58](#_Toc174263737)

[5-1-7 تابع lists:max 59](#_Toc174263738)

[5-1-8 تابع lists:member 59](#_Toc174263739)

[5-1-9 تابع lists:min 59](#_Toc174263740)

[5-1-10 تابع lists:merge 59](#_Toc174263741)

[5-1-11 تابع lists:sort 59](#_Toc174263742)

[5-1-12 تابع lists:reverse 60](#_Toc174263743)

[5-1-13 تابع lists:sum 60](#_Toc174263744)

[5-1-14 تابع lists:split 60](#_Toc174263745)

[5-1-15 تابع lists:map 60](#_Toc174263746)

[lists:filter(Pred, List)5-1-16 60](#_Toc174263747)

[lists:foreach(F, List) 5-1-17 61](#_Toc174263748)

[5-2 ترفند: 61](#_Toc174263749)

[5-3 عملگر های ++ و - - 62](#_Toc174263750)

[5-4 مفاهیم لیستی: (List Comprehensions) 62](#_Toc174263751)

[فصل6: باینری ها: 68](#_Toc174263752)

[6-1 باینری ها 68](#_Toc174263753)

[6-2 BIF های ماژول erlang برای باینری ها: 68](#_Toc174263754)

[binary\_to\_atom(Binary). 69](#_Toc174263755)

[binary\_to\_integer(Binary). 69](#_Toc174263756)

[binary\_to\_list(Binary). 69](#_Toc174263757)

[bitstring\_to\_list(Bitstring). 70](#_Toc174263758)

[bit\_size(Bitstring). 70](#_Toc174263759)

[byte\_size(Bitstring). 70](#_Toc174263760)

[split\_binary(Bin, Pos) 70](#_Toc174263761)

[term\_to\_binary(Term). 71](#_Toc174263762)

[6-3 توابع ماژول binary: 71](#_Toc174263763)

[binary:at/2 71](#_Toc174263764)

[binary:compile\_pattern/1 71](#_Toc174263765)

[binary:split/2 71](#_Toc174263766)

[matches/2: 71](#_Toc174263767)

[match/2: 71](#_Toc174263768)

[binary:copy/2 72](#_Toc174263769)

[binary:first/1 72](#_Toc174263770)

[binary:last/1 72](#_Toc174263771)

[binary:part/3 72](#_Toc174263772)

[6-4 گاردها برای باینری 72](#_Toc174263773)

[is\_binary/1 73](#_Toc174263774)

[is\_bitstring/1 73](#_Toc174263775)

[6-5 نحو بیتی: 73](#_Toc174263776)

[Size 74](#_Toc174263777)

[6-6 TypeSpecifierList 74](#_Toc174263778)

[Type 74](#_Toc174263779)

[Signedness 75](#_Toc174263780)

[Endianness 75](#_Toc174263781)

[unit 75](#_Toc174263782)

[6-7 مفاهیم باینری و عملگرهای بیتی: 76](#_Toc174263783)

[6-8 عملگرهای بیتی: 76](#_Toc174263784)

[6-9 استخراج بیتی: 77](#_Toc174263785)

[فصل 7: ساختارهای guard ، case ، if و for 78](#_Toc174263786)

[7-1 ساختار Case: 78](#_Toc174263787)

[7-2 ساختار for 81](#_Toc174263788)

[7-3 ساختار guardها 82](#_Toc174263789)

[7-4 If 84](#_Toc174263790)

[فصل 8: کار با فایل ها 86](#_Toc174263791)

[8-1 ماژول file 86](#_Toc174263792)

[8-2 ماژول io 90](#_Toc174263793)

[8-3 تا اینجا یاد گرفتیم: 92](#_Toc174263794)

[فصل9: ارتباط ارلنگ با زبان c 92](#_Toc174263795)

[9-1 مکانیزم های که ارلنگ برای ارتباط با دیگر زبان ها دارد: 92](#_Toc174263796)

[9-1-1 ارلنگ توزیع شده 92](#_Toc174263797)

[9-1-2 پورت ها 92](#_Toc174263798)

[9-2 کتابخانه C 93](#_Toc174263799)

[9-3 C Node 93](#_Toc174263800)

[9-4 Linked-in Drivers 93](#_Toc174263801)

[9-5 NIF (Native Implemented Functions) 93](#_Toc174263802)

[9-6 ارتباط یک برنامه خارجی به زبان C با یک برنامه به زبان Erlang با استفاده از پورت 93](#_Toc174263803)

[9-7 وظایف کد cmain: 93](#_Toc174263804)

[9-8 کد کامل cmain.c : 96](#_Toc174263805)

[9-9 وظایف کد cfunc1 : 98](#_Toc174263806)

[9-10 کار با کامپایلر برنامه های C 98](#_Toc174263807)

[9-11 ماژول ارلنگی erlang\_prog: 98](#_Toc174263808)

[open\_port 98](#_Toc174263809)

[9-12 لیست کردن پورت ها 99](#_Toc174263810)

[9-13 بستن پورت 99](#_Toc174263811)

[9-14 ارسال پیام به پورت 99](#_Toc174263812)

[9-15 تغییر مالک پورت 99](#_Toc174263813)

[9-16 توضیح قسمت ارلنگی پورت: 100](#_Toc174263814)

[فصل 10: مدیریت خطا در برنامه های ترتیبی 103](#_Toc174263815)

[10-1 معرفی انواع خطا : 103](#_Toc174263816)

[10-2 انواع کلاس های خطا 103](#_Toc174263817)

[10-3 مدیریت خطا (Handling Errors) 105](#_Toc174263818)

[گرفتن خطا با try...catch 105](#_Toc174263819)

[10-5 Catch روش دیگر به دام انداختن استثنا ها 107](#_Toc174263820)

[10-3 انواع استثنائها: 107](#_Toc174263821)

[10-6 چه زمانی از (exit/1 , error/1 , throw/1) استفاده می کنیم؟ 114](#_Toc174263822)

[فصل 11: برنامه های همزمان 115](#_Toc174263823)

[11-1 مفاهیم ضروری همزمانی: 115](#_Toc174263824)

[11-2 فرایند و تولید مثل فرایندی : 116](#_Toc174263825)

[11-3 ارسال و دریافت پیام بین فرایند ها: 117](#_Toc174263826)

[11-4 مهلت ها : 120](#_Toc174263827)

[فصل 12: فرایند های توزیع شده 122](#_Toc174263828)

[12-1 ساخت گره: 122](#_Toc174263829)

[12-2 تولید فرایند راه دور: 127](#_Toc174263830)

[12-3 BIF های کوکی: 131](#_Toc174263831)

[12-4 ثبت فرایند ها در گره های دور: 131](#_Toc174263832)

[فصل 13: مدیریت خطا در فرایند های همزمان 135](#_Toc174263833)

[13-1 لینک کردن فرایند ها: 135](#_Toc174263834)

[13-2 مانیتور کردن یک فرایند: 136](#_Toc174263835)

[13-3 ثبت فرایند ها: 140](#_Toc174263836)

[13-4 فرایند های سیستمی و به تله انداختن یک استثنا : 141](#_Toc174263837)

[13-5 دریافت انتخابی با مرجع ها: 142](#_Toc174263838)

[فصل 14: جدول ذخیره ترم های ارلنگی: ETS و DETS 146](#_Toc174263839)

[14-1 انواع جدول ها: 146](#_Toc174263840)

[14-1-1 نکته ها: 146](#_Toc174263841)

[14-1-2 استثنا ها : 146](#_Toc174263842)

[14-1-3 توابع ماژول ETS: 146](#_Toc174263843)

[2-14 DETS: 152](#_Toc174263844)

[14-2-1معرفی ماژول DETS: 152](#_Toc174263845)

[14-2-2 توابع ماژول DETS: 152](#_Toc174263846)

[فصل 15: پایگاه داده mnesia : 156](#_Toc174263847)

[15-1 نقاط ضعف و قوت پایگاه داده mnesia: 156](#_Toc174263848)

[15-1-1 نقاط ضعف: 156](#_Toc174263849)

[15-1-2 نقاط قوت: 156](#_Toc174263850)

[15-2 توابع معمول Mnesia: 156](#_Toc174263851)

[15-3 QLC (Query List Comprehension): 160](#_Toc174263852)

# فصل 1 : معرفی ارلنگ

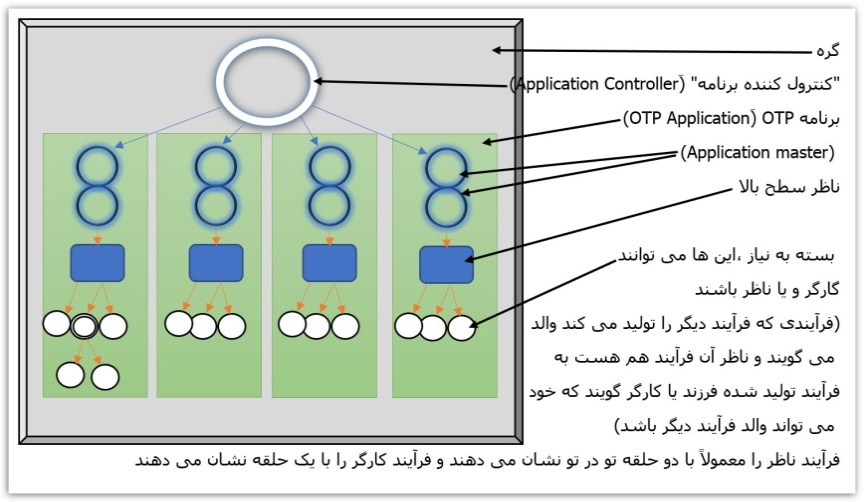
# آشنایی با ارلنگ

|  |  |  |
| --- | --- | --- |
| نماد | نام انگیسی | نام فارسی |
| | | vertical bar | میله عمودی |
| || | double vertical bar | دو میله |
| ~ | tilde | علامت مَدَّک یا پیچ |
| [ ] | square brackets | براکت های مربع |
| # | hash | هش |
| : | colon | کلون |
| << >> | double angled brackets | براکت زاویه دار دوتایی یا نماد نحو باینری |
| \_ | underscore | خط زیر |
| $ | Character notation | نشانه گذاری کاراکتر |

جدول 1.1 : نام علائم

# تاریخچه ارلنگ :

# 1-1 فرایند ها در ارلنگ :



تصویر 1.1: فرایند ها در یک گره

# 1-2 مزایای استفاده از ارلنگ :

# 1-2-1 فرایند های مستقل از سیستم عامل:

# 1-2-2پردازش موازی (parallel processing):

# 1-2-3 همزمانی سبک وزن(concurrency):

# 1-2-4 داده تغییر ناپذیر (immutable data):

مثال ارلنگی:

1. V1= # {s1=>1 , s2=>2 , s3=>3}.
2. #{s1 => 1,s2 => 2,s3 => 3}
3. V1#{s3:=4}.
4. #{s1 => 1,s2 => 2,s3 => 4}
5. V1.
6. #{s1 => 1,s2 => 2,s3 => 3}
7. V2 = V1#{s3:=4}.
8. #{s1 => 1,s2 => 2,s3 => 4}
9. V2.
10. #{s1 => 1,s2 => 2,s3 => 4}

# 1-2-5مزایای متغیر تغییر ناپذیر:

# 1-3 تطبیق الگو(pattern matching):

1. sum([]) -> 0;
2. sum([N]) -> N;
3. sum([N | Ns]) -> N + sum(Ns).

# 1-4 مقاومت در برابر خطا(fault-tolerant):

جو آرمسترانگ در سال ۲۰۱۳ در مصاحبه با Rackspace اشاره می‌کند: «اگر در جاوا یک بار نوشته می شود و همه جا اجرا می شود، در ارلنگ یک بار نوشته می‌شود و همیشه اجرا می‌شود.»

fa.wikipedia.org/wiki/ارلنگ

# 1-5 مقیاس پذیری (Scalability) :

# 1-6 توزیع شده (distributed):

# 1-7 تعویض داغ (Hot Swapping) :

# 1-8 بدون توقف :

# 1-9 اجتناب از حافظه اشتراکی :

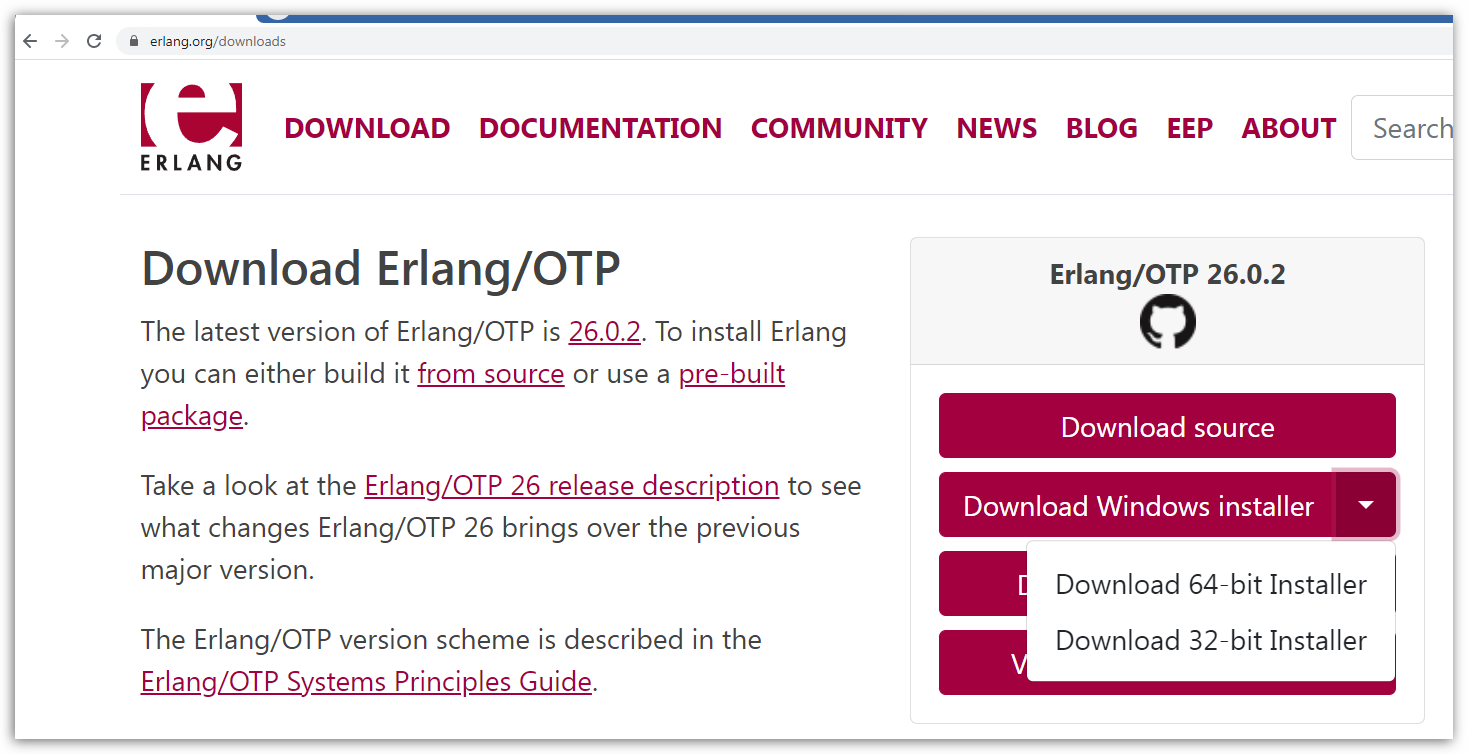
# 1-10 ارلنگ برای چه کارهای مناسب است :

OTP:

# 1-11 نصب پوسته ارلنگی در ویندوز

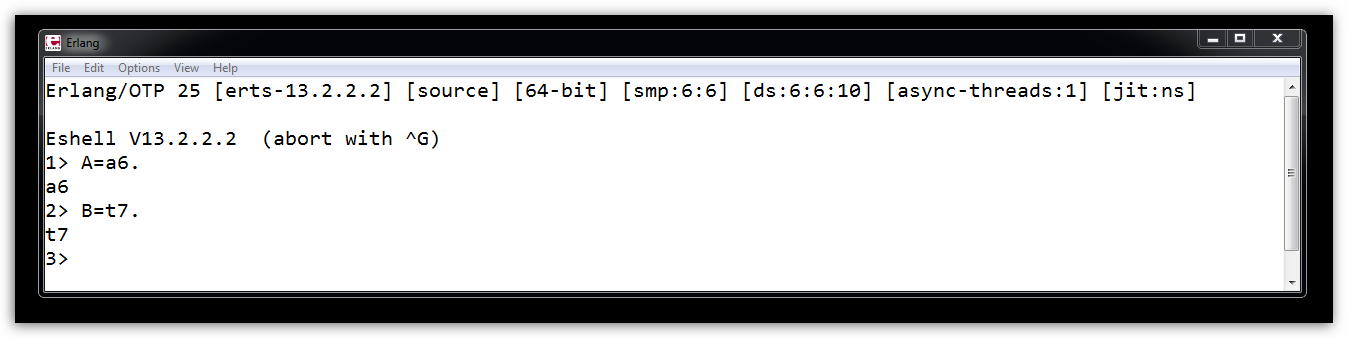
من از پوسته ارلنگ (Eshell) استفاده می کنم. برای نصب آن در ویندوز مراحل زیر را دنبال می کنید.

1: وارد آدرس اینترنتی <https://www.erlang.org/downloads> می شودید (مانند شکل زیر). سپس کلید Download Windows installer را کلید می کنید تا آخرین نسخه از آن دانلود شود و سپس مراحل نصب معمول را طی می کنید.



تصویر 1.2 : سایت erlang.org

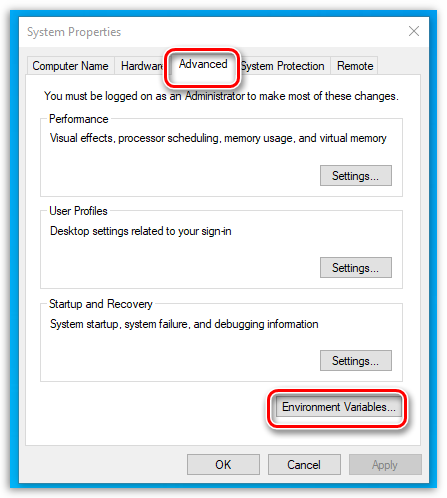
2: سپس آیکون ارلنگ در دسکتاپ نمایان می شود رو ی آن کلیک کنید پنجره ای مانند زیر نشان داده می شود



تصویر 1.3 : پوسته ارلنگ در ویندوز

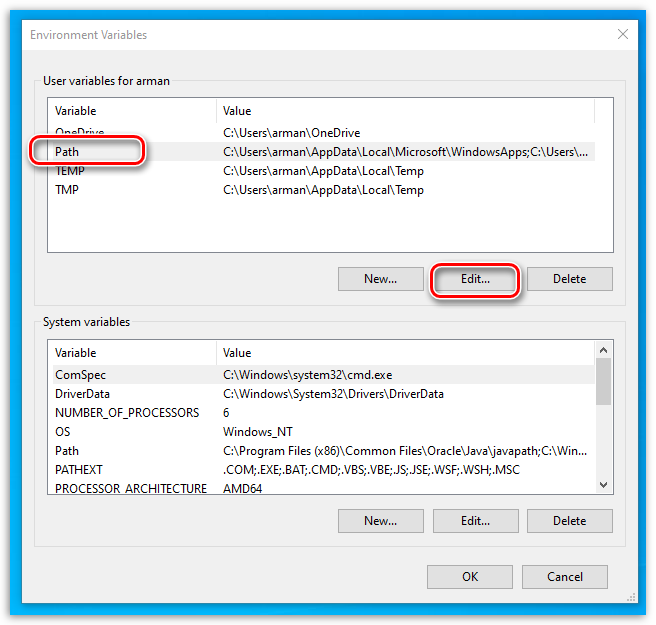
در اینجا 1> و مانند آن خط فرمان ارلنگی است که به ما می گوید آماده دریافت دستورات از ما است. بعد از اتمام دستورات باید حتماً نقطه گذاشته شود در غیر اینصورت ارلنگ فکر می کند که هنوز دستورات شما پایان نیافته است و پردازش را آغاز نمی کند.

نکته دیگر که باید به آن توجه شود آن است که اگر می خواهید erlang در محیط های مانند vscode کار کند ، بعد از نصب ارلنگ باید به edit the system environment variables بروید ، سپس پنجره ای مانند تصویر زیر نمایان می شود. سپس از سربرگ Advanced کلید environment variables… را کلیک کنید:



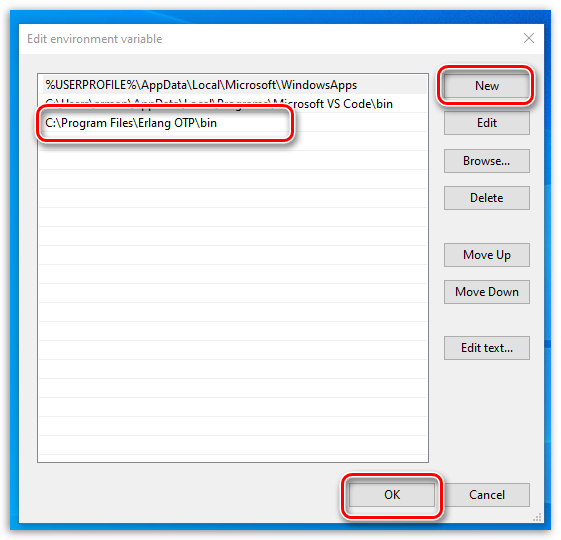
تصویر 1.4: پنجره تنظیمات سیستمی ویندوز

سپس پنجره environment variables باز می شود و باید گزینه path را انتخاب کنید و سپس کلید edit… را کلیک کنید :



تصویر 1.5 : پنجره environment variables

سپس پنجره edit environment variables باز می شود . در این پنجره باید نخست کلید New را کلیک کنید سپس آدرس پوشه bin در محل نصب erlang را وارد کنید مانند تصویر زیر ، سپس کلید ok را کلیک کنید.



تصویر 1.6 : پنجره edit environment variables

# 1-11-2 نصب ارلنگ در سیستم عامل های– mac os – Free BSD Ubuntu/Debian و Fedora

**Mac OS X:**

sudo port install erlang

**Linux:**

Ubuntu/Debian:

sudo apt-get update  
sudo apt-get install erlang

**Fedora:**

sudo yum install erlang

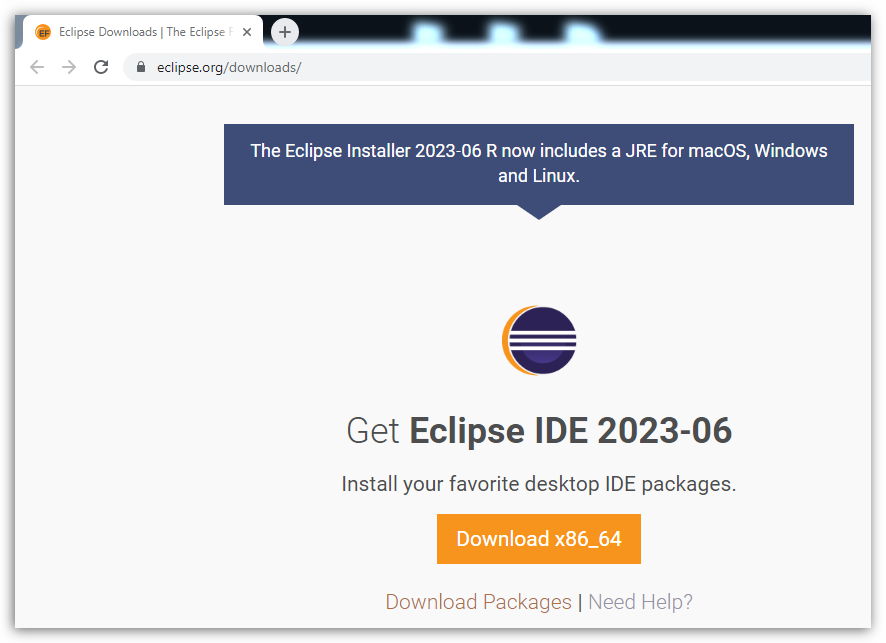
**FreeBSD:**

sudo pkg update  
sudo pkg install erlang

# 1-11-3 نصب پلاگین ارلنگ در eclipse در ویندوز

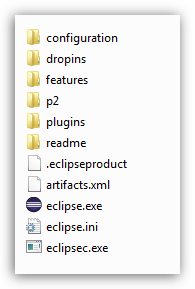
برای نصب پلاگین ارلنگ در eclipse در ویندوز مراحل زیر را دنبال کنید:

1: eclipse را از سایت <https://www.eclipse.org/downloads> دانلود کنید .



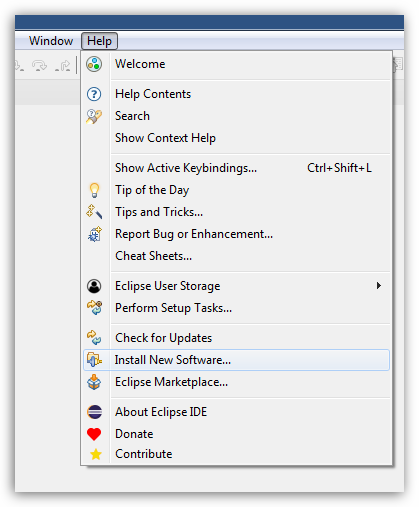
تصویر 1.7 : سایت www.eclipse.org

2: فایل فشرده را در جایی که می خواهید برنامه را اجرا کنید استخراج کنید چون (تا زمان نوشتن این کتاب) فایل نصبی وجود ندارد و فقط یک آرشیو است. محتوای آن چیزی شبیه به تصویر زیر است:



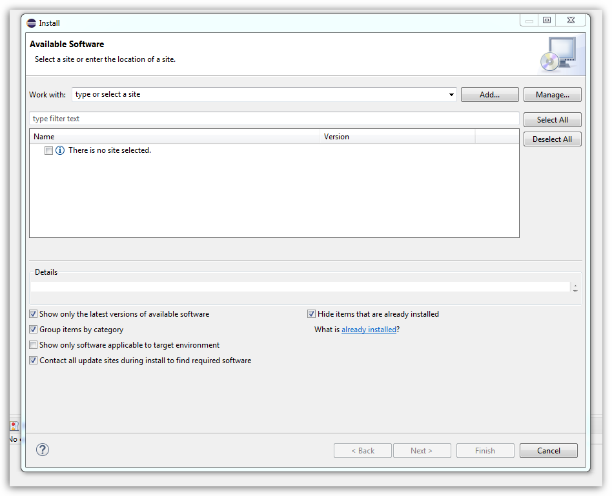
تصویر 1.8 : دایرکتوری eclipse

3: بعد از اجرای برنامه، در منوی help گزینه Install New Software را انتخاب کنید:



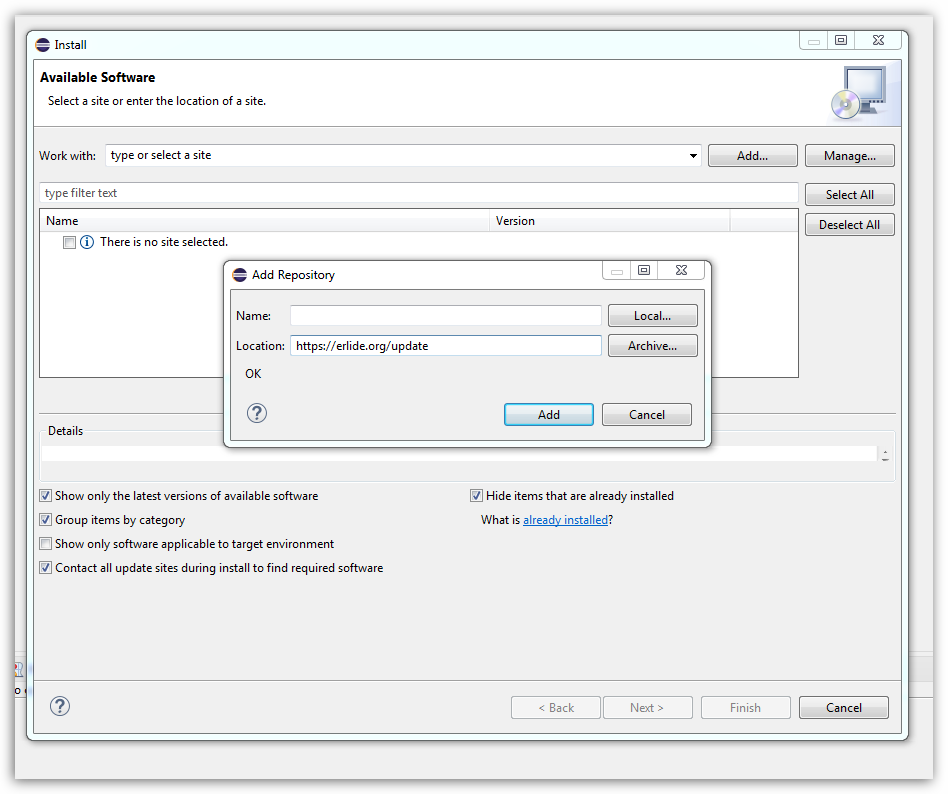
تصویر 1.9 : منویhelp

4: پنجره زیر باز می شود



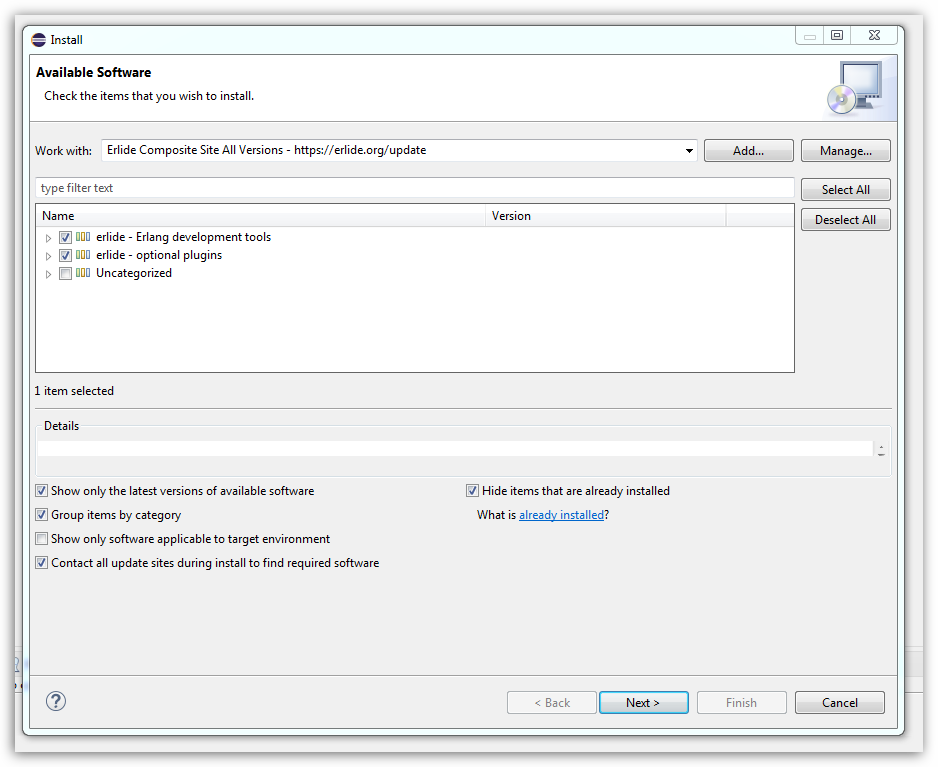
تصویر 1.10 : پنجره Install New Software در منوی help

5: add را کلیک می کنید و پنجره زیر باز می شود و در قسمت location عبارت <https://erlide.org/update> را وارد کنید و سپس add را کلیک کنید.



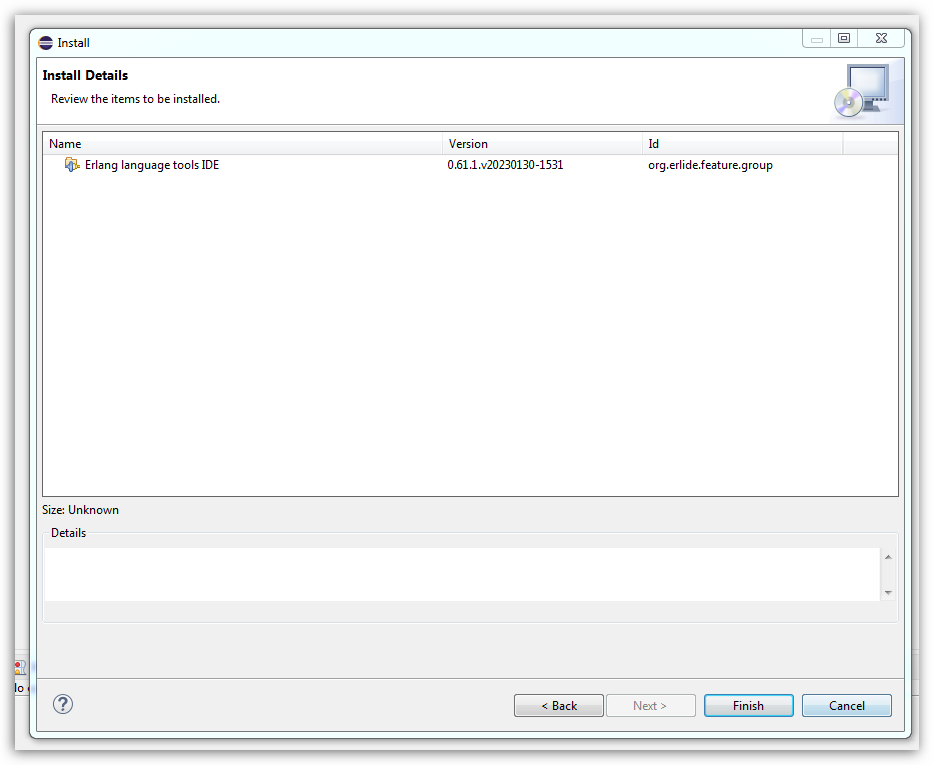
تصویر 1.11 : پنجره add repository

6:سپس پنجره زیر باز می شود که موارد تیک خورده را باید شما هم تیک بزنید سپس Next را کلیک کنید.



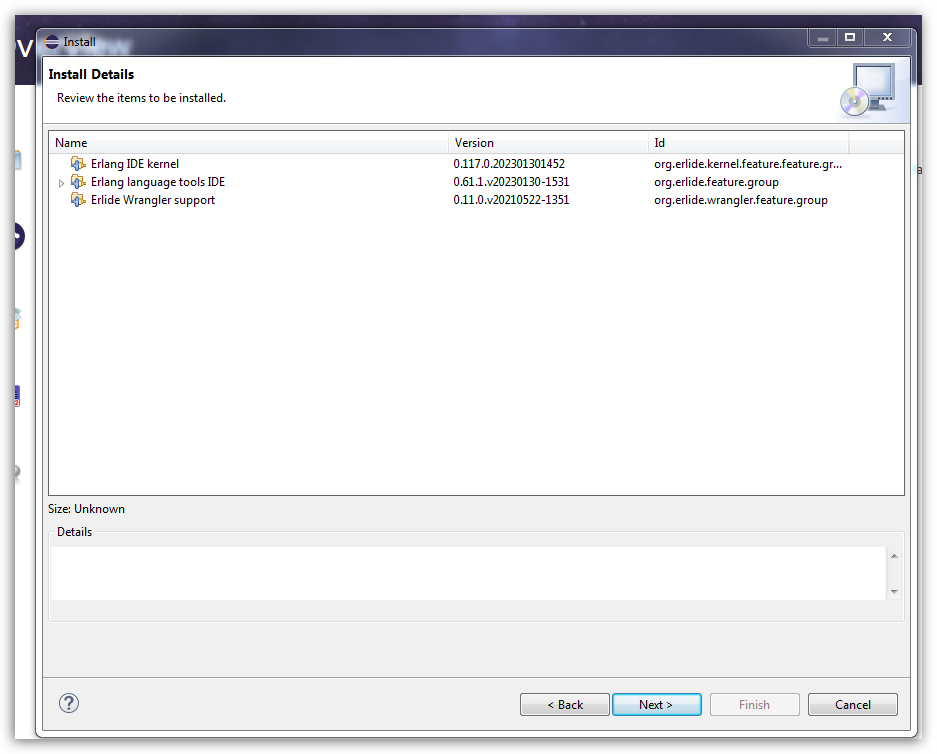
تصویر 1.12 : پنجره Install کلید next

7: حال پنجره زیر نمایان می شود که باید finish را کلیک کنید.



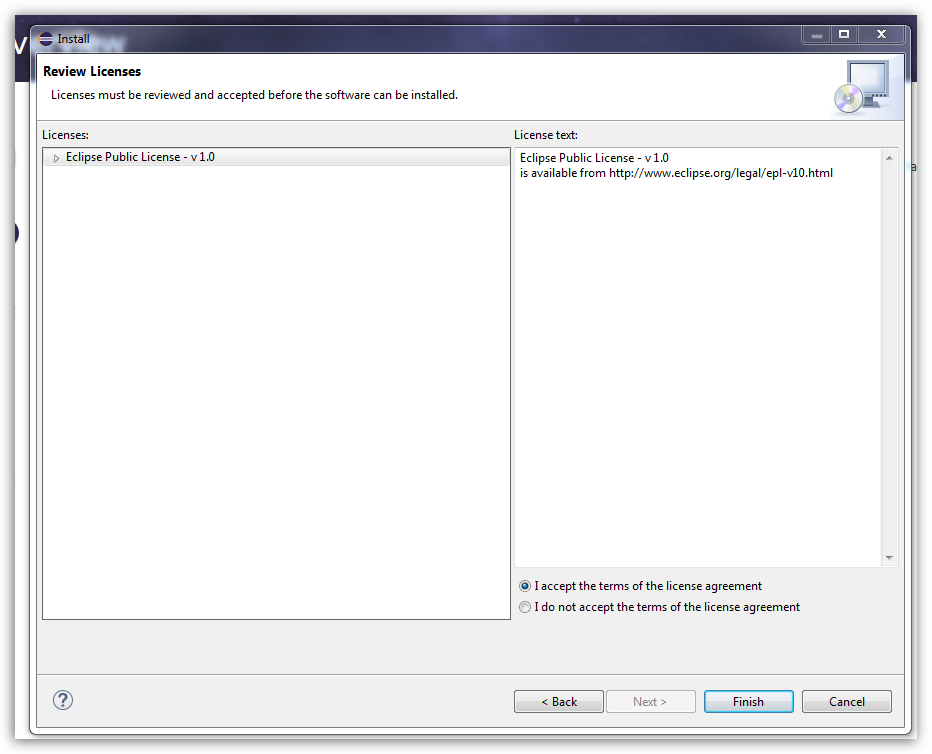
تصویر 1.13 : پنجره پنجره Install کلید finish

8: پنجرا زیر ظاهر می شود که روی Next کلیک می کنید



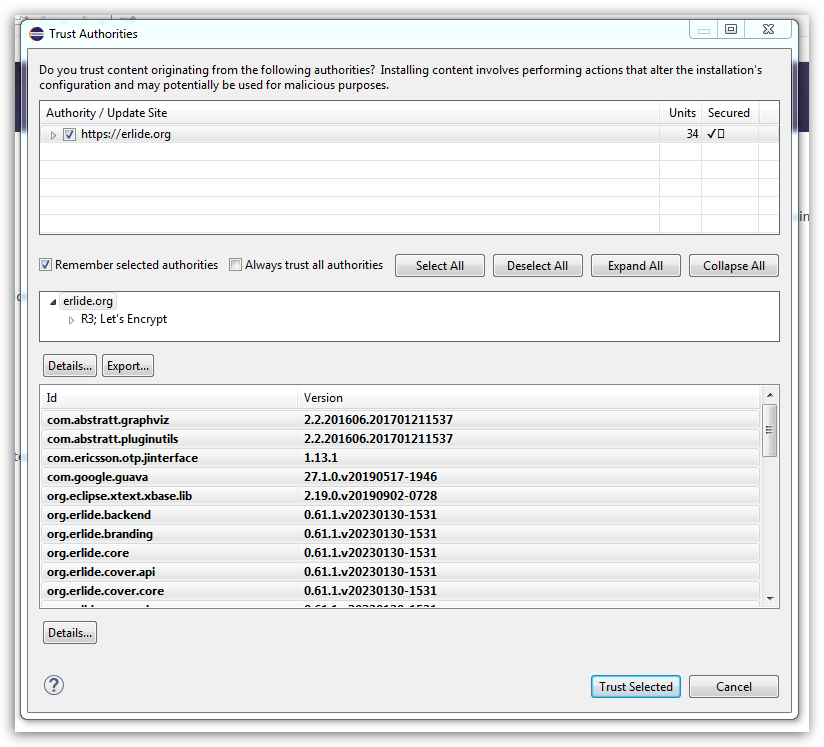
تصویر 1.14 : پنجره Install Details

9: در پنجره زیر کلید رادیویی I accept… را انتخاب می کنید



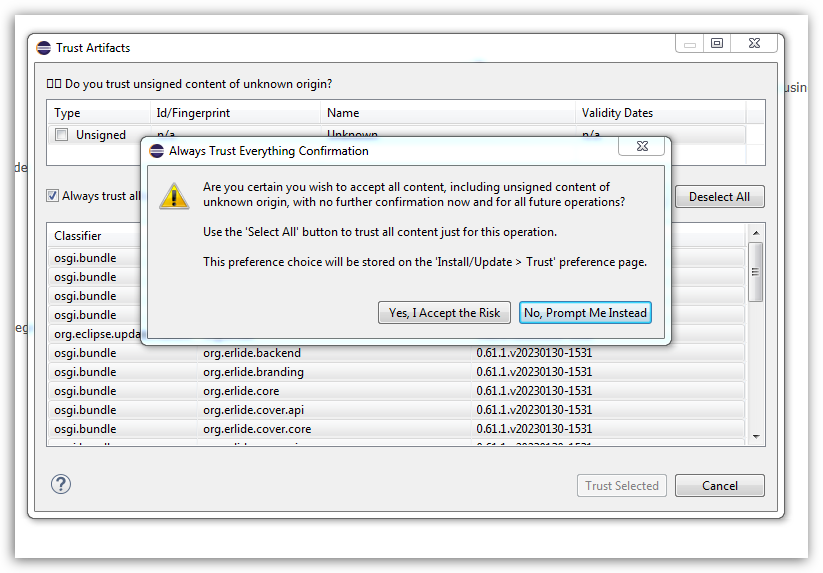
تصویر 1.15 : پنجره Review Licenses

10: در پنجره زیر کلید Trust selected را کلیک کنید.



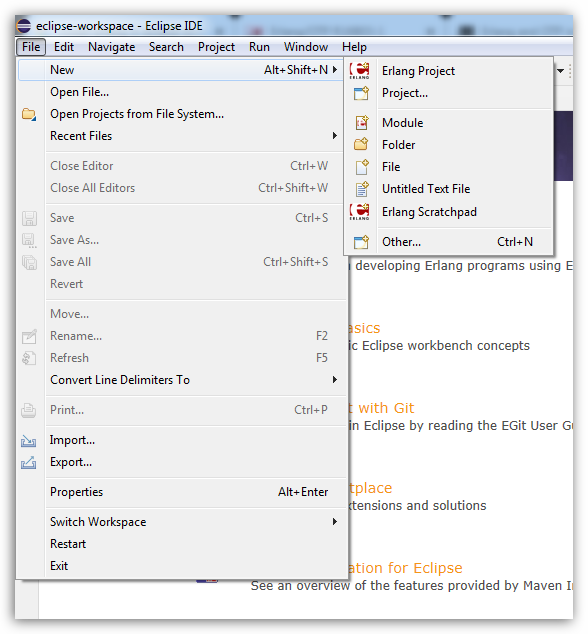
تصویر 1.16 : پنجره Trust authorities

11: در پنجره زیر Yes , … را بزنید.



تصویر 1.17 : پنجره Always Trust Everything Confirmation

12: یک بار برنامه را ببندی و دوباره باز کنید . شما موفق شدید!



تصویر 1.18: پایان نصب

# 1-12 تعدادی از فرمان های پوسته:

نصب ارلنگ و شروع آن را پیشتر گفتیم. حال کار با پوسته ؛ در ادامه فهرست فرمان های پوسته ای که در پوسته می توانید وارد کنید را آورده ایم.

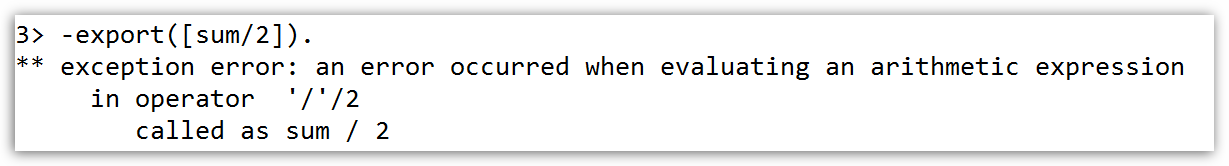
|  |  |  |
| --- | --- | --- |
| # | فرمان | توضیح |
|  | q(). | یک خروج کنترل شده را به همراه دارد. |
|  | erlang:halt() | توقف بلادرنگ سیستم . |
|  | ctrl+A | به ابتدای خط می پرد |
|  | Ctrl+E | به انتهای خط می پرد |
|  | Ctrl+D | کاراکتر جاری را حذف می کند |
|  | Ctrl+F | به کاراکتر جلوی می رود (با کلید های جهتی هم می شود) |
|  | Ctrl+B | به کاراکتر قبلی می رود(با کلید های جهتی هم می شود) |

جدول 1.2 : فهرست فرمان های پوسته ای

q(). این فرمن پوسته ای از تمام برنامه ها خارج می شود و تمام فایل های باز را می بندد و پایگاه داده را متوقف می کند. q() جانشین پوسته ای برای init:stop(). است. اگر init:stop(). را هم در پوسته تایپ کنید نتیجه مشابه دارد.

اگر چیزی در پوسته نتیجه خطا داد معمولاً بهترین کار بستن پوسته و شروع مجدد آن است. در پوسته نمی توانید هر چیزی را که در کد ماژول دیدید تایپ کنید . زیرا تمام کد وارد شده در ماژول عبارات محاسباتی نیستند بنابراین در پوسته ارلنگی درک نمی شوند.

مثال:



تصویر 1.19 : خطا در پوسته ارلنگی

در تصویر بالا برای نشان دادن اینکه نمی توان هر کدی در ماژول را در پوسته نوشت ، اعلان صادر کردن یک تابع را در پوسته نوشتم که پیام خطایی را نشان داد.

# فصل 2 : مفاهیم اساسی

# 2-1 BIF

# 2-2 اعداد صحیح:

1> 2#1000.

8

2> 4#1000.

64

3> 8#1000.

512

4> 10#1000.

1000

5> 16#1000.

4096

6> 32#1000.

32768

7> 64#1000.

\* 2:1: illegal base '64'

7> A =16#553.

1363

8> B =-16#553.

-1363

9> $1.

49

10> $A.

65

11> $a.

97

12> $2-1.

49

1>1=2.

\*\* exception error: no match of right hand side value 2

# 2-3 اعداد شناور و عملگر های ریاض

1> io\_lib:format("**~.3f**", [123456.78e2]).

"12345678.000"

23> io\_lib:format("~.2f",[123456.2345678910e-5]).

"1.23"

24> io\_lib:format("~.2f",[123456.2345678910e5]).

"12345623456.79"

|  |  |  |  |
| --- | --- | --- | --- |
| اولویت | نوع ورودی | شرح عملگر | عملگر |
| 1 | عدد | + X | + X |
| 1 | عدد | - X | - X |
| 2 | عدد | X \* Y | X \* Y |
| 2 | عدد | X / Y | X / Y |
| 2 | عدد صحیح | عملگر بیتی not (ورودی را یکی اضاف و سپس منفی می کند) | bnot X |
| 2 | عدد صحیح | تقسیم صحیح | X div Y |
| 2 | عدد صحیح | باقیمانده تقسیم | X rem Y |
| 2 | عدد صحیح | عملگر بیتی and (اگر هر دو 1 بود جواب 1) | X band Y |
| 3 | عدد | X + Y | X + Y |
| 3 | عدد | X - Y | X - Y |
| 3 | عدد صحیح | عملگر بیتی or (اگر ورودی 0یا1 باشد ؛ اگر حداقل یکی1 باشد جواب 1 می شود در غیر اینصورت جواب 0 می شود) | X bor Y |
| 3 | عدد صحیح | عملگر بیتی xor اگر ورودی ها متفاوت باشد جواب 1 می شود در غیر اینصورت 0 می شود | X bxor Y |
| 3 | عدد صحیح | جایگاه بیتی X را N بیت به چپ می برد مثال در ادامه آمده است | X bsl N |
| 3 | عدد صحیح | برعکس X bsl N است | X bsr N |

جدول 2.1 : عملگر ها

مثال:

1> 1 **bsl** 1.

2

2> 2 **bsl** 1.

4

3> 2 **bsl** 2.

8

4> 2 **bsl** 3.

16

1> 2+3\*4\*5-6.

56

4> 2+3\*4\*(5-6).

-10

5> 12\*(-1).

-12

|  |  |
| --- | --- |
| 8> 8-9.  -1  9> 8+9.  17  10> 55-5.  50  11> 1 **bor** 0.  1  12> 1 **bxor** 0.  1  13> 1 **bxor** 1.  0  14> 0 **bxor** 0.  0 | 1> -20.  -20  2> 2\*3.  6  3> 5/2.  2.5  4> **bnot** 5.  -6  5> 5 **div** 2.  2  6> 5 **rem** 2.  1  7> 1 **band** 0.  0 |

# 2-4 تاپل ها tuples

نکته : برای استفاده از مواردی مانند(\* یا { یا ...) باید آنها را در علامت کوتیشن تکی یا دوتایی قرار دهید در غیر اینصورت پیام خطا دریافت می کنید.

2> M1={apple,1,orange,5}.

{apple,1,orange,5}

3> M2={buy,apple,1,orange,5}.

{buy,apple,1,orange,5}

4> M4={car,{name,porsche},{model,911},{body\_style,coupe}}.

{car,{name,porsche},{model,911},{body\_style,coupe}}

نکته : بین اتم ها در تاپل نباید فاصله باشد؛ می توانید بجای فاصله از مورد دیگری مانند (\_) استفاده کنید.

7> AtomX='1234-abcd'.

'1234-abcd'

8> TAtomX={AtomX}.

{'1234-abcd'}

9> StringX="1234=abcd".

"1234=abcd"

10> TStringX={StringX}.

{"1234=abcd"}

12> T1={arman,{from,iran}}.

{arman,{from,iran}}

13> {arman,{F,I}}=T1.

{arman,{from,iran}}

14> F.

from

15> I.

iran

مثال:

13> {arman,{\_,I}}=T1.

{arman,{from,iran}}

14> \_.

\* 1:1: variable '\_' is unbound

# 2-5 اتم ها :

# 2-6 رشته ها:

1> A="abcde".

"abcde"

2> B='abcde'.

abcde

3> is\_atom(A).

false

4> is\_atom(B).

true

5> is\_list(A).

true

6> is\_list(B).

false

7> [Q|W]=A.

"abcde"

8> Q.

97

9> W.

"bcde"

10> [E|R]=B.

\*\* exception error: no match **of** right hand side value abcde

1> AT='abcde'.

abcde

2> is\_atom(AT).

true

3> is\_list(AT).

false

4> atom\_to\_list(AT).

"abcde"

5> is\_list(AT).

false

6> AL=atom\_to\_list(AT).

"abcde"

7> is\_list(AL).

true

8> L="abcde".

"abcde"

9> A=list\_to\_atom(L).

abcde

10> $P.

80

11> $A.

65

12> $R.

82

13> $S.

83

14> [80,65,82,83].

"PARS"

# 2-7 عملگر های مقایسه ای

|  |  |  |
| --- | --- | --- |
| نتیجه | مثال | عملگر |
| false | 5==10. | == |
| true | 5/=10. | =/ |
| true | 5<10. | > |
| false | 5>10. | < |
| false | 5>=10. | =< |
| true | اگر ورودی ها هم در نوع و هم در مقدار برابر باشند | =:= |
| true | اگر ورودی ها یا در نوع و یا در مقدار نامساوی باشند | =/= |

جدول 2.2 : عملگر های مقایسه ای

1> 1/=1.

false

2> 1/=1.0.

false

3> 2/=1.0.

true

1> 1=/=1.

false

2> 1=/=1.0.

true

3> 1=/=2.

true

4> 1=/=2.0.

true

مثال:

1> 1=:=2.

false

2> 1=:=1.

true

3> 1=:=1.0.

false

4> a=:="A".

false

5> a=:=a.

true

6> a=:='A'.

false

7> "A"=:='A'.

false

# 2-8 متغیر

مثال:

A یا Abc یا Apple\_2

مثال:

1> A=5.

5

2> A=5+5.

\*\* exception error: no match **of** right hand side value 10

3> A\_2=A+5.

10

**مثال:**

1> Z=1.

1

2> X=1.

1

3> Z=2.

\*\* exception error: no match **of** right hand side value 2

4> f().

ok

5> Z=2.

2

6> X=2.

2

7> f(X).

ok

8> Z=3.

\*\* exception error: no match **of** right hand side value 3

9> X=3.

3

# 2-9 ساختار های داده پیچیده

1. A\_expression=

[shoping\_list,{apple,5},{orange,3},{pear,6},{t\_shirt,[{red,1},{blue,1}]}].

[shoping\_list,

{apple,5},

{orange,3},

{pear,6},

{t\_shirt,[{red,1},{blue,1}]}]

# 2-10 تطبیق الگو

مثال:

1> A=1.

1

2> B=1.0.

1.0

3> A=B.

\*\* exception error: no match **of** right hand side value 1.0

4> [C|D]=[1,2,3,4].

[1,2,3,4]

5> C.

1

6> D.

[2,3,4]

9> {E\_1,E\_1,X}={2,2,4}.

{2,2,4}

10> {E\_2,E\_2,X}={2,3,4}.

\*\* exception error: no match **of** right hand side value {2,3,4}

21> [A,B,C|D]=[1,2,3].

[1,2,3]

22> [A2,B2,C2,D2]=[1,2,3].

\*\* exception error: no match **of** right hand side value [1,2,3]

23> D.

[]

# 2-11 متغیر بی اهمیت (یا اهمیت نده) don’t care

مثال:

7> Cars=[body\_class,{'S500',sedan},{'BMW125i',hatchback}].

[body\_class,{'S500',sedan},{'BMW125i',hatchback}]

8> [\_,{X1,X2},{\_B,\_b}]=Cars.

[body\_class,{'S500',sedan},{'BMW125i',hatchback}]

9> [\_,{X1,X2},{\_,\_}]=Cars.

[body\_class,{'S500',sedan},{'BMW125i',hatchback}]

10> [\_,{X1,X2},{\_B,\_B}]=Cars.

\*\* exception error: no match **of** right hand side value

[body\_class,{'S500',sedan},{'BMW125i',hatchback}]

11> \_.

\* 1:1: variable '\_' is unbound

12> \_b.

hatchback

13> \_B.

'BMW125i'

# 2-12 توابع

name\_of\_function(آرگومان ها) ->

بدنه تابع

.

مثال:

sum1(Number1, Number2) -> Number1+ Number2.

name\_of\_function(\_anythings) -> {error, invalid}.

# 2-13 بولین

1> false **and** true.

false

2> false **or** true.

True

3> false **xor** true.

true

4> **not** true.

false

3> 1 **band** 0.

0

4> 1 **and** 0.

\*\* exception error: bad argument

in operator **and**/2

called as 1 **and** 0

5> true **and** false.

false

# فصل 3 : ماژول ها و توابع

مواردی را که در این فصل درباره آنها صحبت می کنیم در ادامه آمده است:

* ماژول ها
* fun ها
* توابع معمولی
* ویژگی های ماژولی (module attribute)
* اعلان -import()
* اعلان export
* اعلان module
* اعلان compile
* ویژگی های تعریف شده توسط کاربر

# 3-1 ماژول ها

1. -module(calculator).
2. -export([sum/1, mult/1, sub/1, div1/1]).
3. sum ({sum , A,B}) **->** A+B.
4. mult ({mult , A,B}) **->** A\*B.
5. sub ({sub , A,B}) **->** A-B.
6. div1 ({div\_d, A,B}) **->** A/B;
7. div1 ({div\_a , A,B}) **->** A **div** B.

اجازه دهید این ماژول را در پوسته آزمایش کنیم:

1> cd("e:").

e:/

ok

2> c(calculator).

{ok,calculator}

3> calculator:sum({sum,5,2}).

7

4> calculator:mult({mult,5,2}).

10

5> calculator:sub({sub,5,2}).

3

6> calculator:div1({div\_d,5,2}).

2.5

7> calculator:div1({div\_a,5,2}).

2

1. -module(calculator2).
2. -export([sum/3, mult/1, sub/1, div1/1]).
3. sum (sum2 , A,B) **->** A+B.
4. mult ({mult , A,B}) **->** A\*B.
5. sub ({sub , A,B}) **->** A-B.

1> calculator2:sum(sum2,5,2).

7

1. div1 ({div\_d, A,B}) **->** A/B;
2. div1 ({div\_a , A,B}) **->** A **div** B.

6> calculator:div1({div\_d,5,2}).

2.5

7> calculator:div1({div\_a,5,2}).

2

# 3-2 نام های رزرو شده

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bsr | bsl | bor | bnot | begin | band | andalso | and | after |
| let | if | fun | end | div | cond | catch | case | bxor |
| xor | when | try | rem | receive | orelse | or | of | not |

جدول 3.1 : جدول کلمات کلیدی در ارلنگ

# 3-3 fun ها

3> Myfun=**fun**(I)**->**(I+5) **end**.

#Fun<erl\_eval.42.3316493>

4> Myfun(4).

9

توجه: کلمه end آخر خط 3 ممکن است فراموش شود که باعث خطا است .

1> F1=**fun**(A,B)**->**(A\*2)+(B\*2) **end**.

#Fun<erl\_eval.41.3316493>

2> F1(2,3).

10

مثال زیر تقریباً مانند مثال بالا است اما چرا و کجا باید از کد مثال پایین استفاده کنیم؟

3> F2= **fun**(A)**->fun**(B)**->**(A\*2)+(B\*2) **end** **end** .

#Fun<erl\_eval.42.3316493>

4> F3=F2(2).

#Fun<erl\_eval.42.3316493>

5> F3(3).

10

حال مثال پیچیده تر از fun ها که بسیار جالب است:

1>Shopping=**fun**({apple\_p,Price,Number})**->**{apple,Price\*Number};

1> ({orange\_p,Price,Number})**->**{orange,Price\*Number} **end**.

#Fun<erl\_eval.42.3316493>

2> F3=Shopping({orange\_p,5,6}).

{orange,30}

3> F4=Shopping({orange\_p,5,10}).

{orange,50}

1. -module(fun\_test).
2. -export([test1/0, test2/0,sum/1]).
3. -import(lists, [map/2]).
4. test1() **->** map(**fun**(X) **->** X +1 **end**, [1,2,3,4,5]).
5. test2() **->** map(**fun** sum/1, [1,2,3,4,5]).
6. sum(X) **->** X +1.

1> c(fun\_test).

{ok,fun\_test}

2> fun\_test:test1().

[2,3,4,5,6]

3> fun\_test:test2().

[2,3,4,5,6]

# 3-4 توابع

# 3-5 ویژگی های ماژولی (module attribute)

-Tag(Value).

# 3-6 اعلان -import() چیست ؟

1. -module(calculator4).
2. -export([calculator/1, start/0]).
3. -import(io,[fwrite/2]).
4. -import(sumx,[sum/1]).
5. start() **->**
6. io:fwrite("data**~p~n**",[date()]),
7. fwrite("data**~p~n**",[date()]).
8. calculator({sum,A,B}) **->** sum({sum,A,B});
9. calculator({mult,A,B})**->** multx:mult({mult,A,B});
10. calculator({sub,A,B}) **->** subx:sub({sub,A,B});
11. calculator({div\_d,A,B})**->** divx:div1({div\_d,A,B});
12. calculator({div\_a,A,B})**->** divx:div1({div\_a,A,B}).

اجازه دهید این ماژول را در پوسته اجرا کنیم (با فرض آنکه آن را قبلاً کامپایل کردید- کامپایل را صفحه 56 و 57 دیدیم و به شکل فراخوانی c(calculator4) در پوسته انجام می شود بعدا کامپایل را با جزئیات بیشتر در همین فصل خواهیم دید ) در پوسته ارلنگ زیر ، در خط 1 ما وارد درایوی که ماژول کامپایل شده در آن است شدیم. در خط 2 ،تابع start را فراخوانی کردیم که نشان می دهد اگر نام ماژول تابع BIF (date) را که ماژول (erlang) باشد در ماژولی که نوشته ایم، وارد کنیم یا وارد نکنیم خروجی یکسان است. این در خط 5 و 6 در این پوسته هم نشان داده شده است. خط 3 و 4 هم فراخوانی توابع را نشان می دهد که قبلاً توضیح داده ایم.

1> cd("e:").

e:/

ok

2> calculator4:start().

data{2024,3,31}

data{2024,3,31}

ok

3> calculator4:calculator({sum,1,2}).

3

4> calculator4:calculator({div\_d,5,2}).

2.5

5> date().

{2024,3,31}

6> erlang:date().

{2024,3,31}

1> c(divx).

{ok,divx}

2> c(multx).

{ok,multx}

3> c(subx).

{ok,subx}

4> c(sumx).

{ok,sumx}

5> c(calculator4).

{ok,calculator4}

%------------------------------------------------------ divx.erl

-module(divx).

-export([div1/1]).

div1({div\_d, A,B})   -> A/B;

div1({div\_a  , A,B}) -> A div B.

%------------------------------------------------------ subx.erl

-module(subx).

-export([sub/1]).

sub ({sub ,  A,B})  -> A-B.

%------------------------------------------------------ multx.erl

-module(multx).

-export([mult/1]).

mult({mult , A,B})  -> A\*B.

%------------------------------------------------------ sumx.erl

-module(sumx).

-export([sum/1]).

sum ({sum ,  A,B})  -> A+B.

%------------------------------------------------------

# 3-7 اعلان export

-export([

Func\_name\_1/ArityX,

Func\_name\_2/ArityX,

Func\_name\_N/ArityX]).

# 3-8 اعلان module

-module(mod\_name)

# 3-9 اعلان compile

-compile(option1).

-compile([option1,…,optionN]).

مثال:

-compile([export\_all,brief]).

c(module\_name).

c(module\_name, [option1,…,optionN]).

compile:file(module\_name, [option1,…,optionN]).

نکته: ماژول compile تابعی غیر از file هم دارد اما ما فقط بعضی گزینه های کاربردی تابع file را توضیح می دهیم باقی موارد را در صفحه راهنمای compiler می توانید پیدا کنید . می توانید به آدرس نصب ارلنگ رفته و راهنمای این ماژول را در قالب html یا pdf پیدا کنید. مثال:

…\Erlang OTP\lib\compiler-8.2.6.3\doc\pdf\compiler-8.2.6.3.pdf

**Brief**

-compile([export\_all,brief]).

error\_1().

1> c(c\_1).

c\_1.erl:6:10: syntax error before: '.'

% 6| error\_1().

% | ^

c\_1.erl:3:2: Warning: record first\_record1 is unused

% 3| -record(first\_record1, {a=aa1,b=11,c}).

% | ^

c\_1.erl:5:2: Warning: export\_all flag enabled - all functions will be exported

% 5| -compile([export\_all,brief]).

% | ^

error

2> c(c\_1).

c\_1.erl:3:2: Warning: record first\_record1 is unused

c\_1.erl:5:2: Warning: export\_all flag enabled - all functions will be exported

{ok,c\_1}

توجه : هشدار ها مانع کامپایل نمی شوند اما خطا ها مانع می شوند.

**compressed**

**debug\_info**

**deterministic**

مثال:

1> c(c\_1).

...

{ok,c\_1}

2> c\_1:module\_info(compile).

[{version,"8.2.6.3"},

{options,[debug\_info]},

{source,"d:/c\_1.erl"}]

3> c(c\_1,**deterministic**).

...

{ok,c\_1}

4> c\_1:module\_info(compile).

[{version,"8.2.6.3"}]

**‘p’**

**{outdir,Dir}**

مثال:

1. c(c\_1,{outdir,'d:/dir\_x'}).

**{i,Dir}**

**-vsn(****TermVsn)**

مثال:

-module(c\_1).

-vsn([[vsn\_div,2.1]]).

-vsn([[vsn\_sum,3.4]]).

1> beam\_lib:version(c\_1).

{ok,{c\_1,[[vsn\_div,2.1],[vsn\_sum,3.4]]}}

2> {F,{A,[[B,C],[D,E]]}}=beam\_lib:version(c\_1).

{ok,{c\_1,[[vsn\_div,2.1],[vsn\_sum,3.4]]}}

3> {ok,{A\_modname,[[vsn\_div,C],[vsn\_sum,E]]}}=

beam\_lib:version(c\_1).

{ok,{c\_1,[[vsn\_div,2.1],[vsn\_sum,3.4]]}}

4> E.

3.4

# 3-10 ویژگی های تعریف شده توسط کاربر:

-Tagx (Value).

تعریف ارلنگ از اتم و ترم:

یک قطعه داده از هر نوع داده ای "ترم" نامیده می شود.

اتم یک لفظ (مجموعه نویسه)، یک ثابت با نام است. اگر با حروف کوچک شروع نشود یا دارای نویسه‌های دیگری غیر از حروف عددی، زیرخط (\_) یا @ باشد، اتم باید در گیومه‌های تکی (') قرار گیرد.

مثال:

-attribute\_prog({div1,sum1}).

فراخوانی در پوسته :

1> c\_1:module\_info().

[...

{attributes,[{vsn,[[vsn\_div,2.1]]},

{vsn,[[vsn\_sum,3.4]]},

**{attribute\_prog,[{div1,sum1}]}]},**

...]

# فصل 4 : ماکرو ها ، map ها ، رکورد ها و فایل های include

# 4-1 macro

1. -define(Const, Replacement).

2. -define(Func(Var1,...,VarN), Replacement).

مثال :

macros\_1.erl

%---------------------------------test1

1. -define(cast, 100).
2. test1()**->**io:format("**~p~n**",[?cast]).

1> c(macros\_1).

{ok,macros\_1}

2> macros\_1:test1().

100

ok

3> c(macros\_1,['P']).

\*\* Warning: No object file created - nothing loaded \*\*

ok

macros\_1.P

1. test1() **->**
2. io:format("**~p~n**", [100]).

Macros\_1.erl

%---------------------------------test2

1. -define(sum(A,B), A+B).
2. test2()**->**
3. A=?sum(10,2),
4. io:format("**~p~n**",[A]).

4> macros\_1:test2().

12

ok

macros\_1.P

test2() **->**

A = 10 + 2,

io:format("**~p~n**", [A]).

مثال:

Macros\_1.erl

1. -define(v(AA),io:format("**~s** = **~s~n**",[ ??AA,AA])).
2. test3() **->** ?v(is\_atom(hi)).

5> macros\_1:test3().

is\_atom ( hi ) = true

ok

Macros\_1.P

test3() **->**

io:format("**~s** = **~s~n**", ["is\_atom ( hi )", is\_atom(hi)]).

# 4-1-1 ماکرو های از پیش تعریف شده ارلنگ :

Macros\_1.erl

1. test4()**->**
2. A=?MODULE\_STRING,
3. B='?MODULE\_STRING',
4. do(A,B),
5. A1=?MODULE,
6. B1='?MODULE',
7. do(A1,B1),
8. A2=list\_to\_atom(?FILE),
9. B2='?FILE',
10. do(A2,B2),
11. A3=?LINE,
12. B3='?LINE',
13. do(A3,B3),
14. A4=?MACHINE,
15. B4='?MACHINE',
16. do(A4,B4),
17. A5=?FUNCTION\_NAME,
18. B5='?FUNCTION\_NAME',
19. do(A5,B5),
20. A6=?FUNCTION\_ARITY,
21. B6='?FUNCTION\_ARITY',
22. do(A6,B6),
23. A7=?OTP\_RELEASE,
24. B7='?OTP\_RELEASE',
25. do(A7,B7).
26. do(A,B)**->**
27. io:fwrite("**~n** **~w**=**~w** **~n**",[B,A]).

6> macros\_1:test4().

'?MODULE\_STRING'=[109,97,99,114,111,115,95,49]

'?MODULE'=macros\_1

'?FILE'='macros\_1.erl'

'?LINE'=36

'?MACHINE'='BEAM'

'?FUNCTION\_NAME'=test4

'?FUNCTION\_ARITY'=0

'?OTP\_RELEASE'=25

ok

# 4-1-2 جریان کنترلی در ماکرو ها:

1. -ifdef(M1).
2. -define(M2 …).
3. -else.
4. -define(M3 …).
5. -endif.

Macros\_1.erl

1. %-----------------------------------test5= ifdef & else & define & endif
2. -ifdef(sum).
3. -define(if\_sum(N), io:format(" defined :**~p** **~n**",[N])).
4. -else.
5. -define(if\_sum(N), not\_defined).
6. -endif.
7. test5() **->**
8. ?if\_sum(1).

7> macros\_1:test5().

defined :1

ok

Macros\_1.erl

1. %------------------------------------------test6= ifndef
2. -ifndef(sum1).
3. -define(if\_sum1(N), io:format("defined :**~p** **~n**",[N])).
4. -else.
5. -define(if\_sum1(N), not\_defined).
6. -endif.
7. test6() **->**
8. ?if\_sum1(2).

8> macros\_1:test6().

defined :2

ok

Macros\_1.erl

1. -define(if\_sum2(N), io:format("if\_sum2 :**~p** **~n**",[N])).
2. -undef(if\_sum2).
3. -ifndef(if\_sum2).
4. -define(if\_sum3(N), io:format("Macro if\_sum2 is not defined :**~p** **~n**",[N])).
5. -else.
6. -define(if\_sum3(N), 'macro\_(if\_sum2)\_is\_defined').
7. -endif.
8. test7() **->**
9. ?if\_sum3(3).

9> macros\_1:test7().

Macro if\_sum2 is not defined :3

ok

# 4-1-3 ماکرو ها و گارد ها:

macros\_1.erl

%------------------------------------------test8= guards-macros

1. -define(atom\_x(X),is\_atom(X)).
2. test8(X) **when** ?atom\_x(X) **->** true\_x;
3. test8(\_X) **->** false\_x.

# 4-2 فایل های include:

-include("File.hrl").

-include\_lib("File.hrl").

-include("directory\_x/r3.hrl").

# فرق بین -include\_libو -include

-include\_lib("File.hrl").

یک مثال:

-include\_lib("kernel/include/ logger.hrl").

1> code:lib\_dir(kernel).

"c:/Program Files/Erlang OTP/lib/kernel-8.5.4.1"

# 4-3 map:

#{Key1=>value1, key2=>value2, ….}.

الگوهای آن مانند مورد زیر است:

(NewValue=:OldKey) یا (>NewValue=NewOrOldKey)

1> A=#{ali\_1=>88}.

#{ali\_1 => 88}

2> B=A#{ali1:=99}.

\*\* exception error: bad key: ali1

in function maps:update/3

called as maps:update(ali1,99,#{ali\_1 => 88})

\*\*\* argument 3: not a map

in call from erl\_eval:'-expr/6-fun-0-'/2 (erl\_eval.erl, line 309)

in call from lists:foldl/3 (lists.erl, line 1350)

3> B=A#{ali1=>99}.

#{ali1 => 99,ali\_1 => 88}

به مثال های زیر توجه کنید:

1> #{a=>1,b=>2,c=>3}.

#{a => 1,b => 2,c => 3}

2> A=#{a=>1,b=>2,c=>3}.

#{a => 1,b => 2,c => 3}

3> B=A#{a=>1,b=>2,c=>4,d=>4}.

#{a => 1,b => 2,c => 4,d => 4}

4> C=A#{a=>1,b=>2,c=>4,d:=4}.

\*\* exception error: bad key: d

in function maps:update/3

called as maps:update(d,4,#{a => 1,b => 2,c => 4})

\*\*\* argument 3: not a map

in call from erl\_eval:'-expr/6-fun-0-'/2 (erl\_eval.erl, line 309)

in call from lists:foldl\_1/3 (lists.erl, line 1355)

5> D=A#{a=>1,b=>2,c:=5}.

#{a => 1,b => 2,c => 5}

مثال:

1> A=#{ {a,b}=>1 }.

#{{a,b} => 1}

2> B=#{ [c,d]=>2 }.

#{[c,d] => 2}

3> C=#{5=>6}.

#{5 => 6}

4> D=#{a5=>6}.

#{a5 => 6}

نکته: کلید های یک map به طور خودکار مرتب می شوند. مثال:

1> E=#{c=>1,d=>3,a=>5,b=>9}.

#{a => 5,b => 9,c => 1,d => 3}

#{نام فیلد مورد نظر := نام متغیر جدید} نام مپی که کلید و مقدار مورد نظر در آن است =

1> M1=#{k1=>a,k2=>b,k3=>c}.

#{k1 => a,k2 => b,k3 => c}

2> #{k2:=R}=M1.

#{k1 => a,k2 => b,k3 => c}

3> R.

b

# 4-3-1 BIF های map:

1> Map2=#{key\_1=>val\_1, key\_2=>val\_2, key\_3=>val\_3, key\_4=>val\_4 }.

#{key\_1 => val\_1,key\_2 => val\_2,key\_3 => val\_3,key\_4 => val\_4}

2> L1= maps:to\_list(Map2).

[{key\_1,val\_1},{key\_2,val\_2},{key\_3,val\_3},{key\_4,val\_4}]

3> erlang:is\_map(Map2).

true

4> Map3=maps:from\_list(L1).

#{key\_1 => val\_1,key\_2 => val\_2,key\_3 => val\_3,key\_4 => val\_4}

برای دانستن اندازه یک map از BIF زیر استفاده می کنیم:

5> erlang:map\_size(Map2).

4

برای دانستن آنکه یک کلید خاص در map هست یا خیر از BIF زیر استفاده می کنیم:

6> maps:is\_key(key\_1, Map2).

true

7> maps:is\_key(key\_x, Map2).

false

برای پیدا کردن مقدار یک کلید در یک map خاص از BIF زیر استفاده می کنیم:

8> maps:get(key\_1, Map2).

val\_1

نکته : اگر کلید اشتباه باشد خطای استثنا(exception error) دریافت می کنید نه اتم false:

9> maps:get(key\_x, Map2).

\*\* exception error: bad key: key\_x

in function maps:get/2

called as maps:get(key\_x,

#{key\_1 => val\_1,key\_2 => val\_2,key\_3 => val\_3,

key\_4 => val\_4})

\*\*\* argument 1: not present in map

BIF بعدی هم مقدار یک کلید را پیدا می کند اما اگر کلید را اشتباه وارد کنیم خطای استثنا نمی دهد بلکه فقط اتم error را بر می گرداند:

10> maps:find(key\_1, Map2).

{ok,val\_1}

11> maps:find(key\_x, Map2).

error

اگر بخواهیم کلید های یک map را پیدا کنیم از BIF زیر استفاده می کنیم:

12> maps:keys(Map2).

[key\_1,key\_2,key\_3,key\_4]

اگر بخواهیم فقط یک کلید را از یک map حذف کنیم از BIF ، maps:remove/2 استفاده می کنیم:

13> Map2.

#{key\_1 => val\_1,key\_2 => val\_2,key\_3 => val\_3,key\_4 => val\_4}

14> Map4=maps:remove(key\_1, Map2).

#{key\_2 => val\_2,key\_3 => val\_3,key\_4 => val\_4}

15> Map2.

#{key\_1 => val\_1,key\_2 => val\_2,key\_3 => val\_3,key\_4 => val\_4}

16> Map4.

#{key\_2 => val\_2,key\_3 => val\_3,key\_4 => val\_4}

اگر بخواهیم چند کلید و مقدار آنها از map حذف شود آنگاه از BIF زیر استفاده می کنیم:

17> maps:without([key\_1,key\_2],Map2).

#{key\_3 => val\_3,key\_4 => val\_4}

اگر بخواهیم دو map را ادغام کنیم، از BIF ، maps:merge/2 استفاده می کنیم:

18> Map1.

#{a => 2}

19> Map2.

#{key\_1 => val\_1,key\_2 => val\_2,key\_3 => val\_3,key\_4 => val\_4}

20> Map5=maps:merge(Map1,Map2).

#{a => 2,key\_1 => val\_1,key\_2 => val\_2,key\_3 => val\_3,

key\_4 => val\_4}

21> Map5.

#{a => 2,key\_1 => val\_1,key\_2 => val\_2,key\_3 => val\_3,

key\_4 => val\_4}

اگر فقط مقادیر کلید های یک map را بخواهیم از BIF زیر استفاده می کنیم:

22> maps:values(Map2).

[val\_1,val\_2,val\_3,val\_4]

# 4-4 Record ها:

-record(Name\_of\_record, {

key\_1=val\_1,

key\_2=val\_2,

key\_3=val\_3,

key\_4

}).

# 4-4-1 چرا باید از رکورد ها استفاده کنیم؟

# 4-4-2 رکورد ها در عمل:

r1.hrl

-record(r1, {a=aa, b=bb, c=cc, d=dd}).

-record(r2, {e=ee, f=ff, g=gg, h=hh}).

-record(r3, {'[l=ll]'}).

-record(r4, {k}).

در ادامه قصد داریم با فایل هدر بالا که شامل 4 رکورد است در پوسته کار کنیم:

1> cd("d:").

d:/

ok

2> rr("r1.hrl").

[r1,r2,r3,r4]

3> #r1{}.

#r1{a = aa,b = bb,c = cc,d = dd}

4> N1=#r1{a=bb}.

#r1{a = bb,b = bb,c = cc,d = dd}

5> N2=N1#r1{c=bb}.

#r1{a = bb,b = bb,c = bb,d = dd}

6> N3=N1#r1{d=bb}.

#r1{a = bb,b = bb,c = cc,d = bb}

7> N3.

#r1{a = bb,b = bb,c = cc,d = bb}

8> N5=#r1{a=11,b=22,c=33,d=44}.

#r1{a = 11,b = 22,c = 33,d = 44}

**9>** **#r1{d=Ad1,c=Bc1}=N5.**

#r1{a = 11,b = 22,c = 33,d = 44}

10> Ad1.

44

11> Bc1.

33

**12> N5 # r1.b.**

22

دستور rr()

دستور rd()

1> rd (name\_record,{a=1,b=2}).

name\_record

دستور rl()

2> rl().

-record('1r',{a = aa2, b = bb, c = cc, d = dd}).

-record(name\_record,{a = 1, b = 2}).

…

-record(r44,{a = aa2, b = bb, c = cc, d = dd}).

ok

دستور rf()

3> rf().

[]

4> rl().

Ok

5> rd (name\_record,{a=1,b=2}).

name\_record

6> rd (name\_record2,{a2=1,b2=2}).

name\_record2

7> rd (name\_record3,{a3=1,b3=2}).

name\_record3

8> rl().

-record(name\_record,{a = 1, b = 2}).

-record(name\_record2,{a2 = 1, b2 = 2}).

-record(name\_record3,{a3 = 1, b3 = 2}).

ok

9> rf(name\_record2).

ok

10> rl().

-record(name\_record,{a = 1, b = 2}).

-record(name\_record3,{a3 = 1, b3 = 2}).

ok

1> rr("r1.hrl").

[r1,r2,r3,r4]

2> record\_info(size,r3).

2

3> record\_info(fields,r1).

[a,b,c,d]

1. -module(record\_1).
2. -export([test\_a/0]).
3. -record(first\_record, {a=aa1,b=11,c}).
4. new\_record\_1() **->**
5. #first\_record{a="aa2", b=22,c=cc2}.
6. print\_r(#first\_record{a=A,b=B,c=C}) **->**
7. io:format("a:**~p** b:**~p** c:**~p** **~n**", [A,B,C]).
8. test\_a() **->**
9. print\_r(new\_record\_1()).

1. -module(record\_1\_include).
2. -export([test\_a/0]).
3. -include("r3.hrl").
4. %-record(first\_record, {a=aa1,b=11,c}).
5. new\_record\_1() **->**
6. #first\_record{a="aa2", b=22,c=cc2}.
7. print\_r(#first\_record{a=A,b=B,c=C}) **->**
8. io:format("a:**~p** b:**~p** c:**~p** **~n**", [A,B,C]).
9. test\_a() **->**
10. print\_r(new\_record\_1()).

r3.hrl

-record(first\_record, {a=aa1,b=11,c}).

1> c(record\_1).

{ok,record\_1}

2> record\_1:test\_a().

a:"aa2" b:22 c:cc2

ok

3> c(record\_1\_include).

{ok,record\_1\_include}

4> record\_1\_include:test\_a().

a:"aa2" b:22 c:cc2

1. -module(record\_2).
2. -export([test\_a/0, ,test\_c/0]).
3. -record(first\_record, {a=aa1,b=11,c}).
4. %------------------------------------------
5. new\_record\_1() **->**
6. #first\_record{a="aa2", b=22,c=cc2}.
7. edit\_record\_1(#first\_record{a=A,b=B} = New\_record\_2) **when** A=="aa2" **->**
8. New\_record\_2#first\_record{b=B+6,c=cc3}.
9. edit\_record\_2(#first\_record{a=A,b=B}) **when** A=="aa2" **->**
10. [B+10,A].
11. %------------------------------------------
12. print\_r(#first\_record{a=A,b=B,c=C}) **->**
13. io:format("a:**~p** b:**~p** c:**~p** **~n**", [A,B,C]).
14. print\_e([B,A]) **->**
15. io:format("b:**~p** a:**~p** **~n**", [B,A]).
16. %------------------------------------------
17. test\_a() **->**
18. print\_r(#first\_record{a="aa22", b=22,c=cc2}).
19. test\_b() **->**
20. print\_r(edit\_record\_1(new\_record\_1())).
21. test\_c() **->**
22. print\_e(edit\_record\_2(new\_record\_1())).

1. new\_record\_1() **->**
2. #first\_record{a="aa2", b=22,c=cc2}.

10. edit\_record\_1(#first\_record{a=A,b=B} = New\_record\_2) **when** A=="aa2" **->**

1. New\_record\_2#first\_record{b=B+6,c=cc3}.
2. test\_b() **->**
3. print\_r(edit\_record\_1(new\_record\_1())).

1. edit\_record\_2(#first\_record{a=A,b=B}) **when** A=="aa2" **->**
2. [B+10,A].
3. print\_e([B,A]) **->**
4. io:format("b:**~p** a:**~p** **~n**", [B,A]).
5. test\_c() **->**
6. print\_e(edit\_record\_2(new\_record\_1())).

# فصل 5 : لیست و مفاهیم لیستی:

مثال:

1> L1=[1234,abcd,[point,p1],{apple,5}].

[1234,abcd,[point,p1],{apple,5}]

مثال:

2> L2=[65,66,67].

"ABC"

3> L3=[651,661,671].

[651,661,671]

# 

4> L2=[65,66,67].

"ABC"

5> io:format("~w~n",[L2]).

[65,66,67]

ok

1> List=[a,b,c,d].

[a,b,c,d]

2> [Head\_1|Tail\_1]=List.

[a,b,c,d]

3> Head\_1.

a

4> Tail\_1.

[b,c,d]

1> List\_5=[a,b,c,d].

[a,b,c,d]

2> [Head\_1,Head\_2|Tail\_1]=List\_5.

[a,b,c,d]

3> Head\_1.

a

4> Head\_2.

b

5> Tail\_1.

[c,d]

5> List=[a,b,c,d,e|f].

\*\* exception error: no match of right hand side value [a,b,c,d,e|f]

مثال هایی برای مشاهده خطاها و موفقیت ها در افزودن عنصر جدید به یک لیست:

4> List=[a,b,c,d].

[a,b,c,d]

%-------------------

6> List\_2=[List|e].

[[a,b,c,d]|e]

7> List\_3=[e|List|e].

\* 1:15: syntax error before: '|'

7> List\_3=[e|List].

[e,a,b,c,d]

8> List\_4=[List,e].

[[a,b,c,d],e]

# 5-1 در ادامه توابع کاربردی تر ماژول list را معرفی می کنیم:

# 5-1-1 تابع lists:all

مثال:

1>List\_1=[2,4,6,8,10],

1> Only\_even\_numbers= fun (Number\_x) -> Number\_x rem 2==0 end ,

1>Result= lists:all(Only\_even\_numbers,List\_1).

true

# 5-1-2 تابع lists:any

مثال:

2> List\_2 = [1,2,3,4,5,6,7,8,9,10],

2> Any\_even\_numbers=

fun(Number\_x) -> Number\_x rem 2==0 end,

2>Result= lists:any(Any\_even\_numbers,List\_2).

true

# 5-1-3 تابع lists:delete

مثال:

3> List\_2=[1,2,3,4,5,6,7,8,9,10],

3> List\_3=lists:delete(9,List\_2).

[1,2,3,4,5,6,7,8,10]

# 5-1-4 تابع lists:droplast

مثال:

4> List\_1=[1,2,3,4,5,6,7,8,9,10],

4> List\_2=lists:droplast(List\_1).

[1,2,3,4,5,6,7,8,9]

# 5-1-5 تابع lists:duplicate

مثال:

1> L1=lists:duplicate(3,5).

[5,5,5]

# 5-1-6 تابع lists:last

مثال:

1> List\_1=[1,2,3,4,5,6,7,8,9,10],

1> L2=lists:last(List\_1).

10

# 5-1-7 تابع lists:max

مثال:

1> List\_1=[100,2,3,4,5,6,7,8,9,10].

[100,2,3,4,5,6,7,8,9,10]

2> List\_2=lists:max(List\_1).

100

# 5-1-8 تابع lists:member

1> List\_1=[100,2,3,4,5,6,7,8,9,10].

[100,2,3,4,5,6,7,8,9,10]

2> List\_2=lists:member(100,List\_1).

true

3> List\_3=lists:member(200,List\_1).

false

# 5-1-9 تابع lists:min

1> List\_1=[100,2,3,4,5,6,7,8,9,10].

[100,2,3,4,5,6,7,8,9,10]

2> L2=lists:min(List\_1).

2

# 5-1-10 تابع lists:merge

مثال:

1> R=[1,2,5].

[1,2,5]

2> lists:merge([[1],[r],[R],["R"],[3],[2]]).

[1,2,3,r,[1,2,5],"R"]

# 5-1-11 تابع lists:sort

1> L=[1,3,5,7,9,2,4,6,8].

[1,3,5,7,9,2,4,6,8]

2> lists:sort(L).

[1,2,3,4,5,6,7,8,9]

# 5-1-12 تابع lists:reverse

1> L=[1,2,3,4,5].

[1,2,3,4,5]

2> lists:reverse(L).

[5,4,3,2,1]

# 5-1-13 تابع lists:sum

1> L=[1,2,3,4,5].

[1,2,3,4,5]

2> lists:sum(L).

15

# 5-1-14 تابع lists:split

1> lists:split(4,[a,b,c,d,e]).

{[a,b,c,d],[e]}

2> lists:split(2,[a,b,c,d,e]).

{[a,b],[c,d,e]}

# 5-1-15 تابع lists:map

1> lists:map(fun(X)-> 2\*X end, [1,2,3,4,5]).

[2,4,6,8,10]

%--------------------------------------

2> L=[1,2,3,4,5].

[1,2,3,4,5]

3> lists:map(fun(X)->2\*X end,L).

[2,4,6,8,10]

4> F=fun(X)->2\*X end.

#Fun<erl\_eval.42.3316493>

5> lists:map(F,L).

[2,4,6,8,10]

# lists:filter(Pred, List)5-1-16

1> F=fun(X) -> X > 10 end.

#Fun<erl\_eval.42.3316493>

2> List=[11,21,31,41,5,6,7,8,9].

[11,21,31,41,5,6,7,8,9]

3> lists:filter(F, List).

[11,21,31,41]

# lists:foreach(F, List) 5-1-17

مثال:

1> List=[11,21,31,41,5,6,7,8,9].

[11,21,31,41,5,6,7,8,9]

2> F3=fun(X) ->Z=X\*2, io:format("~p~n", [Z]) end.

#Fun<erl\_eval.42.3316493>

3> lists:foreach(F3, List).

22

42

62

82

10

12

14

16

18

ok

حال همین مورد را بدون تابع io:format اجرا می کنیم :

1> List=[11,21,31,41,5,6,7,8,9].

[11,21,31,41,5,6,7,8,9]

2> F4=fun(X) ->Z=X\*2 end.

#Fun<erl\_eval.42.3316493>

3> lists: (F4, List).

ok

# 5-2 ترفند:

مثال:

lists:foldl(Fun, Acc0, List).

در مستندات می گوید این تابع :

مثالی هم در مستندات است:

1> lists:foldl(fun(X, Sum) -> X + Sum end, 0, [1,2,3,4,5]).  
15  
2> lists:foldl(fun(X, Prod) -> X \* Prod end, 1, [1,2,3,4,5]).  
120

foldl(F, Accu, List) when is\_function(F, 2) ->

    case List of

        [Hd | Tail] -> foldl\_1(F, F(Hd, Accu), Tail);

        [] -> Accu

    end.

foldl\_1(F, Accu, [Hd | Tail]) ->

    foldl\_1(F, F(Hd, Accu), Tail);

foldl\_1(\_F, Accu, []) ->

Accu.

# 5-3 عملگر های ++ و - -

مثال:

2> [i]++[r,a]++[n].

[i,r,a,n]

3> [1,2,3,4]--[4,5].

[1,2,3]

4> ([1,2,3]--[1,6])--[2].

[3]

5> [1,2,3]--[1,6]--[2].

[2,3]

# 5-4 مفاهیم لیستی: (List Comprehensions)

17> List\_1=[a,b,c,d,e].

[a,b,c,d,e]

18> [{[X2]++[X2]}||X2<-List\_1].

[{[a,a]},{[b,b]},{[c,c]},{[d,d]},{[e,e]}]

19> lists:map(fun(L1)->{[L1]++[L1]}end,List\_1).

[{[a,a]},{[b,b]},{[c,c]},{[d,d]},{[e,e]}]

1> List\_1=[a,b,c,d,e].

[a,b,c,d,e]

2> F=fun(L1)->{[L1]++[L1]}end.

#Fun<erl\_eval.42.3316493>

3> [F(L1)||L1<-List\_1].

[{[a,a]},{[b,b]},{[c,c]},{[d,d]},{[e,e]}]

7> L2=[1,2,3,4].

[1,2,3,4]

8> L3=[2,3,4,5].

[2,3,4,5]

9> [B\*A||A<-L2,B<-L3].

[2,3,4,5,4,6,8,10,6,9,12,15,8,12,16,20]

31> L6=[1,2].

[1,2]

32> L7=[a,b].

[a,b]

33> [{B,A\*2}||A<-L6,B<-L7].

[{a,2},{b,2},{a,4},{b,4}]

-module(age1).

-export([age/1]).

age(ali)   -> 1360;

age(reza)  -> 1365;

age(armin) -> 1370;

age(hasan) -> 1375;

age(ahmad) -> 1380.

1> c(age1).

{ok,age1}

2> Now=[1402].

[1402]

3> People=[ali,hasan].

[ali,hasan]

4> [{P,N-age1:age(P)}||N<-Now,P<-People].

[{ali,42},{hasan,27}]

5> [{'name==>',P,'age==>',N-age1:age(P)}||

N<-Now,P<-People].

[{'name==>',ali,'age==>',42},

{'name==>',hasan,'age==>',27}]

-module(age2).

-export([sum\_age/2]).

sum\_age(People,Now)->

[{'name==>',P,'age==>',N-age1:age(P)}||N<-Now,P<-People].

1> c(age2).

{ok,age2}

2> age2:sum\_age([ali],[1402]).

[{'name==>',ali,'age==>',42}]

3> Now1=[1402].

[1402]

4> People1=[ali,hasan].

[ali,hasan]

5> age2:sum\_age(People1,Now1).

[{'name==>',ali,'age==>',42},

{'name==>',hasan,'age==>',27}]

6> age2:sum\_age([People1],[Now1]).

\*\* exception error: no function clause matching age1:age([ali,hasan]) (age1.erl, line 4)

     in function  age2:'-sum\_age/2-lc$^1/1-1-'/4 (age2.erl, line 5)

[Expr || Qualifier1,...,QualifierN]

1> L1=[1,2,3,4].

[1,2,3,4]

2> [A||A<-L1].

[1,2,3,4]

3> [{A}||A<-L1].

[{1},{2},{3},{4}]

4> [[A]||A<-L1].

[[1],[2],[3],[4]]

5> [[A+1]||A<-L1].

[[2],[3],[4],[5]]

Pattern <- ListExpr

6> L5=[{1,2},{3,4},{5,6},{7,8},{9,0}].

[{1,2},{3,4},{5,6},{7,8},{9,0}]

7> [[A,B+2]||{A,B}<-L5].

[[1,4],[3,6],[5,8],[7,10],[9,2]]

1>D=[{ali,ahvaz,1328},{arman,ahvaz,1363},

{reza,tehran,1363},{armin,tehran,1328},

{hasan,yazd,1340},{ahmad,yazd,1350},

{naser,shiraz,1340},{hamed,shiraz,1350}].

[{ali,ahvaz,1328},

 {arman,ahvaz,1363},

 {reza,tehran,1363},

 {armin,tehran,1328},

 {hasan,yazd,1340},

 {ahmad,yazd,1350},

 {naser,shiraz,1340},

 {hamed,shiraz,1350}]

2> [{"name:",Name}||{Name,City,Age}<-D].

[{"name:",ali},

 {"name:",arman},

 {"name:",reza},

 {"name:",armin},

 {"name:",hasan},

 {"name:",ahmad},

 {"name:",naser},

 {"name:",hamed}]

3>[{[mr]++[Person],"age\_shamsi=",Age,"age\_miladi=", Age+621}||{Person,City,Age}<-D].

[{[mr,ali],"age\_shamsi=",1328,"age\_miladi=",1949},

 {[mr,arman],"age\_shamsi=",1363,"age\_miladi=",1984},

 {[mr,reza],"age\_shamsi=",1363,"age\_miladi=",1984},

 {[mr,armin],"age\_shamsi=",1328,"age\_miladi=",1949},

 {[mr,hasan],"age\_shamsi=",1340,"age\_miladi=",1961},

 {[mr,ahmad],"age\_shamsi=",1350,"age\_miladi=",1971},

 {[mr,naser],"age\_shamsi=",1340,"age\_miladi=",1961},

 {[mr,hamed],"age\_shamsi=",1350,"age\_miladi=",1971}]

4>[{"name:",Name,"age:",Age,"city:",City}||

{Name,City,Age}<-D,City/=ahvaz].

[{"name:",reza,"age:",1363,"city:",tehran},

 {"name:",armin,"age:",1328,"city:",tehran},

 {"name:",hasan,"age:",1340,"city:",yazd},

 {"name:",ahmad,"age:",1350,"city:",yazd},

 {"name:",naser,"age:",1340,"city:",shiraz},

 {"name:",hamed,"age:",1350,"city:",shiraz}]

5>[{"name:",Name,"age:",Age,"city:",City}||

{Name,City,Age}<-D,Age==1363].

[{"name:",arman,"age:",1363,"city:",ahvaz},

 {"name:",reza,"age:",1363,"city:",tehran}]

6>[{[name],[Name],[age],[Age],[city] ,[City]}||{Name,City,Age}<-D,Age==1363].

[{[name],[arman],[age],[1363],[city],[ahvaz]},

 {[name],[reza],[age],[1363],[city],[tehran]}]

7>[{[name]++[Name],[age]++[Age],[city]++[City]} ||{Name,City,Age}<-D,Age==1363].

[{[name,arman],[age,1363],[city,ahvaz]},

 {[name,reza],[age,1363],[city,tehran]}]

8> [{[name:]++[Name],[age]++[Age],[city ]++[City]} || {Name,City,Age}<-D,Age==1363].

\* 4:9: syntax error before: ']'

1> [";",":","."].

[";",":","."]

2> [;].

\* 2:2: syntax error before: ';'

2> [:].

\* 1:2: syntax error before: ':'

2> [.].

\* 1:2: syntax error before: '.'

# فصل6: باینری ها:

1> A= <<1,2,3>>.

<<1,2,3>>

1> <<0>>.

<<0>>

2> <<256>>. %% 256-256=0

<<0>>

3> <<512>>. %% 512 - (256\*2) =0

<<0>>

4> <<1>>.

<<1>>

5> <<257>>. %% 257-256=1

<<1>>

6> <<97,98,99>>.

<<"abc">>

7> <<"a,b,c">>.

<<"a,b,c">>

# 6-1 باینری ها

مثال:

1> L1= [1,2,3,4,5].

[1,2,3,4,5]

2> erlang:list\_to\_binary(L1).

<<1,2,3,4,5>>

3> binary:list\_to\_bin(L1).

<<1,2,3,4,5>>

# 6-2 BIF های ماژول erlang برای باینری ها:

# binary\_to\_atom(Binary).

این BIF یک باینری می گیرد و معادل اتم آن را برمی گرداند.

1> B2= <<"a,b,c">>.

<<"a,b,c">>

2> A1= binary\_to\_atom(B2).

'a,b,c'

3> B3= <<97,98,99>>.

<<"abc">>

4> binary\_to\_atom(B3).

abc

5> B5= <<"97","98">>.

<<"9798">>

6> binary\_to\_atom(B5).

'9798'

توجه : هنگام تعریف باینری آنها را در کوتیشن تکی قرار نمی دهیم .

7> atom\_to\_binary(A1).

<<"a,b,c">>

# binary\_to\_integer(Binary).

integer\_to\_binary(Integer).  
binary\_to\_integer(Binary, Base).

مثال:

1> Int = binary\_to\_integer(<<"22">>).

22

2> Bin = integer\_to\_binary(Int).

<<"22">>

3> Int2 = binary\_to\_integer(<<"f">>,16). %% base (2~36)

15

# binary\_to\_list(Binary).

binary\_to\_list(Binary).

binary\_to\_list(Binary, Start, Stop).

list\_to\_binary(List).

مثال:

1> Bin1 = <<1,2,3,4,5>>.

<<1,2,3,4,5>>

2> L1= binary\_to\_list(Bin1).

[1,2,3,4,5]

3> B1= list\_to\_binary(L1).

<<1,2,3,4,5>>

4> L2= binary\_to\_list(Bin1,2,4).

[2,3,4]

# bitstring\_to\_list(Bitstring).

1> Bit= list\_to\_bitstring("welcome").

<<"welcome">>

2> List1= bitstring\_to\_list(<<"welcome">>).

"welcome"

# bit\_size(Bitstring).

# byte\_size(Bitstring).

1> bit\_size(<<"welcome">>).

56

2> byte\_size(<<"welcome">>).

7

# split\_binary(Bin, Pos)

1> {B1,B2}= split\_binary(<<1,"1",2,2,3,4,5,6,7,8,9,10>>, 5).

{<<1,49,2,2,3>>,<<4,5,6,7,8,9,10>>}

2> B1.

<<1,49,2,2,3>>

3> B2.

<<4,5,6,7,8,9,10>>

4> byte\_size(B1).

5

# term\_to\_binary(Term).

1> Term= "a,a,a".

"a,a,a"

2> Binary= term\_to\_binary(Term).

<<131,107,0,5,97,44,97,44,97>>

3> Term2= binary\_to\_term(Binary).

"a,a,a"

# 6-3 توابع ماژول binary:

# binary:at/2

1> A= <<7,6,1,2,3,4,5>>.

<<7,6,1,2,3,4,5>>

2> binary:at(A, 1).

6

# binary:compile\_pattern/1

# binary:split/2

1> Ref1= binary:compile\_pattern(<<8,8,8,8>>).

{bm,#Ref<0.1878032878.3512598534.102254>}

2> Bin1 = <<1,2,3,4,8,8,8,8,5,6,7,8,9>>.

<<1,2,3,4,8,8,8,8,5,6,7,8,9>>

3> [A,B]= binary:split(Bin1,Ref1).

[<<1,2,3,4>>,<<5,6,7,8,9>>]

4> A.

<<1,2,3,4>>

5> B.

<<5,6,7,8,9>>

# matches/2:

1> binary:matches(<<1,2,3,4,4,5,6,7,7,8,9>>, [<<4>>,<<7>>]).

[{3,1},{4,1},{7,1},{8,1}]

# match/2:

1> binary:match(<<1,2,3,4,4,5,6,7,7,8,9>>, [<<4>>,<<7>>]).

{3,1}

2> binary:match(<<1,2,3,3,3,5,6,7,7,8,9>>, [<<4>>,<<7>>]).

{7,1}

3> binary:match(<<1,2,3,3,3,5,6,6,7,8,9>>, [<<4>>,<<7>>]).

{8,1}

# binary:copy/2

1> binary:copy(<<"hi\_">>,3).

<<"hi\_hi\_hi\_">>

# binary:first/1

این تابع اولین بایت در مجموعه باینری را برمی گرداند. مثال:

1> binary:first(<<10,2,3>>).

10

# binary:last/1

این تابع آخرین بایت در باینری را برمی گرداند. مثال:

1> binary:last(<<10,2,3>>).

3

# binary:part/3

1> Bin = <<1,2,3,4,5,6,7,8,9,10>>.

<<1,2,3,4,5,6,7,8,9,10>>

2> binary:part(Bin, 4, 2).

<<5,6>>

3> binary:part(Bin, byte\_size(Bin), -5).

<<6,7,8,9,10>>

در خط 3 مشخص می کند که می خواهیم 5 بایت آخر باینری برگردانده شود.

# 6-4 گاردها برای باینری

# is\_binary/1

1> B1= <<"a,1">>.

<<"a,1">>

2> B2= "a,1".

"a,1"

3> is\_binary(B1).

true

4> is\_binary(B2).

False

# is\_bitstring/1

1> Bitst= <<X:4>>.

<<3:4>>

2> is\_bitstring(Bitst).

true

3> is\_binary(Bitst).

false

# 6-5 نحو بیتی:

1> Bit\_string1= <<1,2,65>>.

<<1,2,65>>

2> Bit\_string2= <<65>>.

<<"A">>

3> Bit\_string3=<<65>>.

\* 1:14: syntax error before: '<'

<<E1,...,En>>

هر Ei یک سگمنت از رشته بیت را مشخص می کند. و شکل نحوی زیر را دارد.

Value:Size/TypeSpecifierList

Value

1. << Value >>
2. << Value:Size >>
3. << Value/TypeSpecifierList >>

# Size

مثال:

1> bit\_size (<<"a">>).

8

2> bit\_size (<<"a":2>>).

2

3> bit\_size (<<"a":2,"a":8>>).

10

# 6-6 TypeSpecifierList

**Type-** **Signedness-** **Endianness-** **Unit**

# ****Type****

مثال:

1> A= <<<<1:32>>/binary>>.

<<0,0,0,1>>

2> B= <<<<"a,b":16>>/binary>>.

<<0,97,0,44,0,98>>

3> bit\_size(<<"a"/utf8>>).

8

4> bit\_size(<<"a"/utf32>>).

32

5> bit\_size(<<5.5/float>>).

64

5> A3= <<"5+5.5"/float>>.

<<64,74,128,0,0,0,0,0,64,69,128,0,0,0,0,0,64,74,128,0,0,0,

0,0,64,71,0,0,0,...>>

6> A2= <<(5+5.5)/float>>.

<<64,37,0,0,0,0,0,0>>

# Signedness

مانند خط در مثال زیر 4 که <<-123>> معادل با 133 در نظر گرفته شده است:

|  |
| --- |
| 1> <<X1/signed>> = <<123>>.  <<"{">>  2><<X2/signed>> = <<-123>>.  <<133>>  3> <<X3/unsigned>> = <<123>>.  <<"{">>  4> <<X4/unsigned>> = <<-123>>.  <<133>> |
| 5> X1.  123  6> X2.  -123  7> X3.  123  8> X4.  133 |

# Endianness

مثال:

1> <<"a":32/little>>.

<<97,0,0,0>>

2> <<"a":32/big>>.

<<0,0,0,97>>

3> <<"a":32/native>>.

<<97,0,0,0>>

4> <<"a":32/ big, "b":32/little >>.

<<0,0,0,97,98,0,0,0>>

5> <<"a":64/little>>.

<<97,0,0,0,0,0,0,0>>

# unit

مثال:

1> <<"a":32/ big, "b":8/little >>.

<<0,0,0,97,98>>

2> <<"a":32/ big, "b":4/little >>.

<<0,0,0,97,2:4>>

3> <<"a":32/ big, "b":4/little-unit:2 >>.

<<0,0,0,97,98>>

# 6-7 مفاهیم باینری و عملگرهای بیتی:

مثال:

1> << <<X>> || <<X>> <= <<"a","b","c",1,2,3>>>>.

<<97,98,99,1,2,3>>

نکته: در مفاهیم لیستی زمانی که از باینری ها استفاده می کنید ،وارد کردن سایز الزامی است اما تعیین نوع الزامی نیست.

1> [A || <<A:8/bitstring>> <= <<1,2,3,4,5,6,7,8,9,97,"a">> ].

[<<1>>,

<<2>>,

<<3>>,

<<4>>,

<<5>>,

<<6>>,

<<7>>,

<<"\b">>,<<"\t">>,<<"a">>,<<"a">>]

2> [A || <<A:8>> <= <<1,2,3,4,5,6,7,8,9,97,"a">> ].

[1,2,3,4,5,6,7,8,9,97,97]

3> [A || <<A/bitstring>> <= <<1,2,3,4,5,6,7,8,9,97,"a">> ].

\* 1:9: binary fields without size are not allowed in patterns of bit string generators

# 6-8 عملگرهای بیتی:

مثال برای bnot :

1> [A || <<A:1>> <= <<1,2>>].

[0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0]

2> [bnot(A) || <<A:1>> <= <<1,2>>].

[-1,-1,-1,-1,-1,-1,-1,-2,-1,-1,-1,-1,-1,-1,-2,-1]

3> [bnot(A-1) || <<A:1>> <= <<1,2>>].

[0,0,0,0,0,0,0,-1,0,0,0,0,0,0,-1,0]

مثال:

1> [A band B || <<B:1, A:1>> <= <<1:1,1:1,0:1,1:1>>].

[1,0]

2> [A bor B || <<B:1, A:1>> <= <<1:1,1:1,0:1,1:1>>].

[1,1]

3> [A bxor B || <<B:1, A:1>> <= <<1:1,1:1,0:1,1:1>>].

[0,1]

مثال برای bsl و bsr :

1> Y= 2#10001.

17

2> X= 2#10001 bsl 1.

34

3> Z= X bsr 1.

17

# 6-9 استخراج بیتی:

1> B1= <<5,6,7>>.

<<5,6,7>>

2> <<A,B,C>> =B1.

<<5,6,7>>

3> A.

5

4> <<AA:8,BB:8,CC:8>> =B1.

<<5,6,7>>

5> <<D:4,E:4,F:4>> =B1.

\*\* exception error: no match of right hand side value <<5,6,7>>

6> [Bit1,Bit2|Bits]= binary\_to\_list(<<1,2,3,4>>).

[1,2,3,4]

7> Bit1.

1

8> Bits.

[3,4]

# فصل 7: ساختارهای guard ، case ، if و for

# 7-1 ساختار Case:

case Expression of

        Pattern\_1 [when Guard\_X] -> Expression1,…,ExpressionN;

        ...

        Pattern\_N [when Guard\_X] -> Expression1,…,ExpressionN;

Otherwise-> {error\_X,[Otherwise]}

        end

1. -module(case\_1).
2. -export([dinner/1,number1/1,number2/1,number3/1,map1/2]).
3. dinner (Day) ->
4. case Day of
5. saturday    -> {'sandwich'};
6. sunday      ->{'Pizza'};
7. monday      ->{'Gheymeh stew'};
8. tuesday     ->{'Ghormeh Sabzi stew'};
9. wednesday   ->{'Soltani Kebab'};
10. thursday    ->{'Bakhtiari kebab'};
11. friday      ->{'kubideh kebab'};
12. Otherwise->

io:fwrite("no match:~w~n",[Otherwise])

1. end.

1> case\_1:dinner(sunday).

{'Pizza'}

2> case\_1:dinner(s).

no match:s

number1 (N) when N>0 ->

case N of

1-> io:fwrite('The sum of the number you entered (1) and ~w is ~w~n',[1,N+1]);

...

6-> io:fwrite('The sum of the number you entered (6) and ~w is ~w~n',[6,N+6]);

Otherwise-> io:fwrite("no match:~w~n",[Otherwise])

end.

1>case\_1:number1(1).

The sum of the number you entered (1) and 1 is 2

ok

2> case\_1:number1(7).

no match:7

ok

3> case\_1:number1(0).

\*\* exception error: no function clause matching case\_1:number1(0) (case\_1.erl, line 16)

number1 (N) when N>0 ->

number2 (N) when N>0, N<6 ->

case N of

...

end.

مثال:

1. map1(Func, [H|T]) ->
2. case Func(H) of
3. false -> map1(Func, T);
4. true -> [H|map1(Func, T)]
5. end;
6. map1(\_Func, []) ->
7. [].

1> case\_1:map1(fun(I)->(I rem 2)=:=0 end,[1,2,3,4,5,6,7,8,9]).

[2,4,6,8]

1. number3 (N) ->
2. case N of
3. {A,B}when A==1,B==1-> io:fwrite('a=>~w~n',[{A,B+1}]);
4. {A,B}when A==1,B==2-> io:fwrite('b=>~w~n',[N]);
5. {A,B}when A==2,B==1-> io:fwrite('c=>~w~n',[N]);
6. {A,B}when A==2,B==2-> io:fwrite('d=>~w~n',[N]);
7. {A,B}when A==3,B==1-> io:fwrite('e=>~w~n',[N]);
8. {A,B}when A==3,B==2-> io:fwrite('f=>~w~n',[N]);
9. {A,B}when A==3,B>2->  io:fwrite('f=>~w~n',[N]);
10. Otherwise->

io:fwrite("no match:~w~n",[Otherwise])

1. end.

1>case\_1: ({1,2}).

b=>{1,2}

ok

2> case\_1:number3({1,3}).

no match:{1,3}

ok

3> case\_1:number3({1,1}).

a=>{1,2}

ok

6> case\_1:number3({2,1}).

c=>{2,1}

ok

# 7-2 ساختار for

1. -module(for1).
2. -export([for/2]).
3. for(I, J) when is\_integer(I),erlang:is\_integer(J), I > J ->
4. io:fwrite("~n  <== End of the for loop. ==> ~n");
5. for(I, J) when is\_integer(I),erlang:is\_integer(J) , I =< J ->
6. io:fwrite("~n ==>  Hi, welcome to the for loop . ~w  <== ~n",[I]),
7. for(I + 1, J).

1> for1:for(2,5).

==> Hi, welcome to the for loop . 2 <==

==> Hi, welcome to the for loop . 3 <==

==> Hi, welcome to the for loop . 4 <==

==> Hi, welcome to the for loop . 5 <==

<== End of the for loop. ==>

ok

2> for1:for(20,5).

<== End of the for loop. ==>

ok

3> for1:for(5,5).

==> Hi, welcome to the for loop . 5 <==

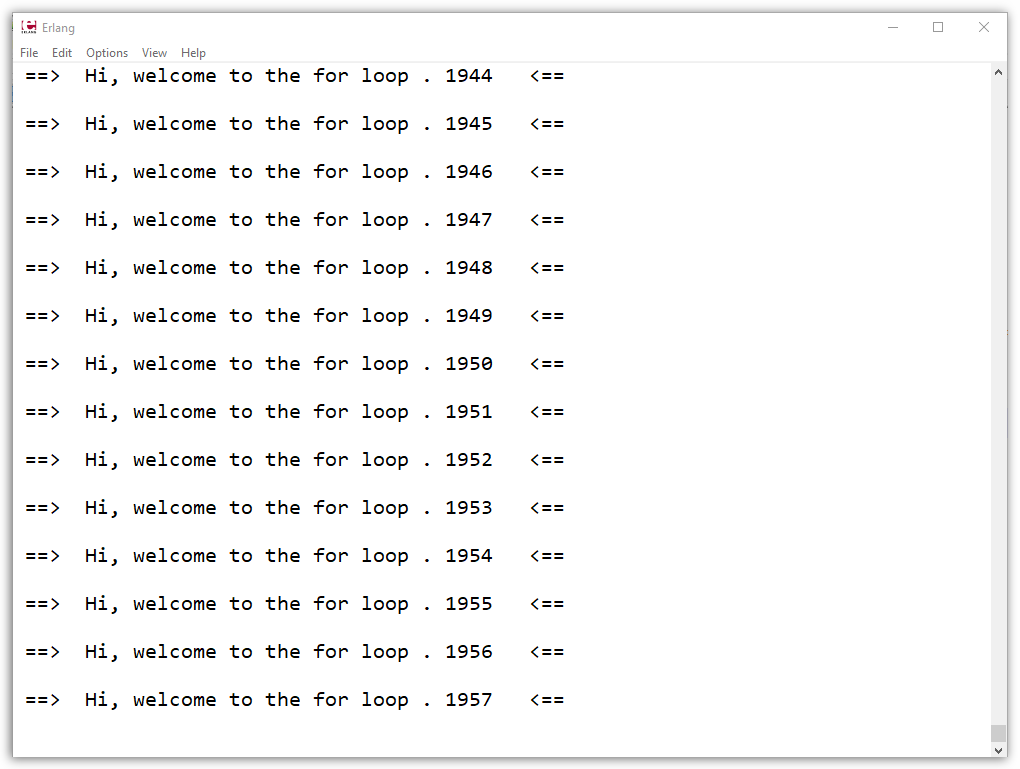
<== End of the for loop. ==>

ok

4>for1:for(5,a).

\*\* exception error: no function clause matching for1:for(5,a) (for1.erl, line 4)

1>for1:for(5,a).



تصویر 1.7 : اجرای ناخواسته و نامحدود یک تابع

# 7-3 ساختار guardها

مثال:

if1()-> if2(5).

if2(Age)->

    if

        Age >= 18 -> ok;

        true -> no

      end.

**عملگر های بولی** (مانند : and , not, or و ....)

* **عملگر های محاسباتی** (مانند: +،\*،-، div و ...)
* **توابع داخلی گارد ها** (مانند float(X) که یک عدد را می گیرد و آن را اعشاری می کند)
* **BIF های تست نوع** (مانند : is\_atom یا is\_tuple یا is\_integer یا is\_float) توجه کنید که تابع is\_float برای تعیین نوع استفاده می شود و تابع float برای تبدیل استفاده می شود.
* **عملگر های مقایسه ترم ها** (مانند: ==یا =:= یا >=... )

**عبارات گاردی** معتبر می تواند شامل هر کدام از عبارات ارلنگی شود که در بالا ذکر شد.

مثال ها:

A=5.

B=10.

C=

fa(A,B) when A<B     ->B;

fa(A,B)             ->A.

fb(A,B) when A<B , B>9   ->B;

fb(A,B)             ->A.

fc(A)when is\_integer(A) ->io:write('is integer');

fc(A)                   ->io:write('no integer').

%------------------------------------------------------------

1> C='at'.

at

2> is\_atom(C).

true

3> D="at".

"at"

4> is\_atom(D).

false

5> (is\_atom(C)) and (is\_atom(D)).

false

6> (is\_atom(C)) or (is\_atom(D)).

true

|  |  |
| --- | --- |
| نام جدید | نام قدیم |
| is\_atom(X) | atom(X) |
| is\_binary(X) | binary(X) |
| is\_constant(X) | constant(X) |
| is\_float(X) | float(X) |
| is\_integer(X) | integer(X) |
| is\_tuple(X) | tuple(X) |

|  |  |
| --- | --- |
| نام جدید | نام قدیم |
| is\_list(X) | list(X) |
| is\_number(X) | number(X) |
| is\_pid(X) | pid(X) |
| is\_port(X) | port(X) |
| is\_reference(X) | reference(X) |

جدول 7.1 : نام های قدیمی و معادل جدید آنها

1> X=5.

5

2> is\_integer(X).

true

3> integer(X).

\*\* exception error: undefined shell command integer/1

**کلام آخر:**

1. -module(guard1).
2. -compile([export\_all]).
3. case1(X) ->
4. case {is\_integer(X), X/=5} of
5. {true,true} -> X;
6. \_           -> "error"
7. end.
8. case2(X) when is\_integer(X)->
9. case {X/=5} of
10. {true} -> X;
11. \_      -> "error"
12. end.

حال اجازه دهید آن را در پوسته اجرا کنیم.

1> guard1:case1(10).

10

2> guard1:case1(-5).

-5

3> guard1:case1(5).

"error"

4> guard1:case1(a).

"error"

5> guard1:case2(10).

10

28> guard1:case2(-5).

-5

6> guard1:case2(5).

"error"

7> guard1:case2(a).

\*\* exception error: no function clause matching guard1:case2(a) (guard1.erl, line 9)

# 7-4 If

if

Guard1 ->

Body1;

Guard2->

Body2;

true-> TrueBody

end.

1. -module(if\_a).
2. -export([if\_1/1]).
3. if\_1(X)->
4. if
5. X=:=5         -> io:fwrite('~w~n',[X\*2]);
6. is\_float(X)   -> io:fwrite('~w is float~n',[X]);
7. X > 5 andalso X < 7 -> six;
8. *%sum(X,2)=:=4       -> {'four'}*
9. true           -> {'no match'}
10. end .
11. *%sum(A,B)->*
12. *%A+B.*

1> cd("e:").

e:/

ok

2> c(if\_a).

{ok,if\_a}

3> if\_a:if\_1(5).

10

ok

4> if\_a:if\_1(6).

six

5> if\_a:if\_1(6.5).

6.5 is float

ok

6> if\_a:if\_1(7).

{'no match'}

1. -module(if\_a).
2. -export([if\_1/1]).
3. if\_1(X)->
4. if
5. X=:=5           -> io:fwrite('~w~n',[X\*2]);
6. is\_float(X)     -> io:fwrite('~w is float~n' ,[X]);
7. X > 5 andalso X < 7 -> six
8. *%sum(X,2)=:=4       -> {'four'}*
9. *%true               -> {'no match'}*
10. end .

9> c(if\_a).

{ok,if\_a}

10> if\_a:if\_1(7).

\*\* exception error: no true branch found when evaluating an if expression

     in function  if\_a:if\_1/1 (if\_a.erl, line 5)

# فصل 8: کار با فایل ها

# 8-1 ماژول file

file:close(IoDevice) **->** ok | {error, Reason}.

file:consult(Filename) **->** {ok, Terms} | {error, Reason}.

**term.txt:**

{[a,b2],[c1,{d,e},f3],g6}.

{arman, ahvaz, 1363}.

**no\_term.txt:**

{a},

{b}.

1> file:consult("term.txt").

{ok,[{[a,b2],[c1,{d,e},f3],g6},{arman,ahvaz,1363}]}

2> file:consult("no\_term.txt").

{error,{2,erl\_parse,"bad term"}}

1> file:copy("term.txt","new\_term.txt").

{ok,49}

2> file:delete("new\_term1.txt").

ok

3> file:del\_dir("dir1").

ok

⠀4> file:del\_dir\_r("dir2").

ok

term2.txt:

A=10.

B=20.

C=A+B.

io:format("**~w**+**~w**=**~w~n**",[A,B,C]).

5> file:eval("term2.txt").

10+20=30

ok

6> file:get\_cwd().

{ok,"c:/Program Files/Erlang OTP/usr"}

7> file:list\_dir("dir1").

{ok,["file1.erl",...]}

دستور زیر یک دایرکتوری جدید ایجاد می کند:

8> file:make\_dir("e:/dir3/dir6").

ok

{ok, Io\_device} =file:open(File, Modes).

file:pread(Io\_device ,Start, Number).

مثال:

Term3.txt:

{a1,a2,a3,a4,a5}.

{b1,b2,b3,b4,b5}.

{c1,c2,c3,c4,c5}.

{d1,d2,d3,d4,d5}

9> {ok, Io\_device} = file:open("term3.txt", [raw,read]).

{ok,{file\_descriptor,...

10> file:pread(Io\_device, 1, 30).

{ok,"a1,a2,a3,a4,a5}.**\r\n**{b1,b2,b3,b4"}

11> {ok, Io\_device2} = file:open("term4.txt", [raw,write]).

{ok,{file\_descriptor,...

12> file:pwrite(Io\_device2, 2, "hi").

Ok

13> file:close(Io\_device2).

ok

14> {ok, Io} = file:open("term3.txt", [raw,write,read]).

{ok,{file\_descriptor...

15> T2 = file:read(Io,11).

{ok,"{a1,a2,a3,a"}

16> file:close(Io).

ok

17> T6 = file:read\_file("term3.txt").

{ok,<<"{a1,a2,a3,a4,a5}.**\r\n**{b1,b2,b3,b4,b5}.**\r\n**{c1,c2,c3,c4,c5}.**\r\n**{d1,d2,d3,d4,d5}.**\r\n**">>}

18> {ok, Io} = file:open("term3.txt", [raw,write,read]).

{ok,{file\_descriptor...

19> file:read\_line(Io).

{ok,"{a1,a2,a3,a4,a5}.**\n**"}

20> file:read\_line(Io).

{ok,"{b1,b2,b3,b4,b5}.**\n**"}

21> file:read\_line(Io).

{ok,"{c1,c2,c3,c4,c5}.**\n**"}

22> file:read\_line(Io).

{ok,"{d1,d2,d3,d4,d5}.**\n**"}

23> file:read\_line(Io).

eof

تابع rename/1 نام یک فایل را تغییر می دهد. مثال:

1> file:rename("term3.txt", "ll").

ok

1> file:script("a.txt").

10+20=30

{ok,ok}

1> file:set\_cwd("d:").

ok

2> cd("d:").

d:/

ok

3> file:get\_cwd().

{ok,"d:/"}

4> cd("").

d:/

ok

5> file:write\_file("term4.txt", "Data").

ok

# 8-2 ماژول io

io:format(Format)

io:format(Format,)

io:format(IoDevice, Format, Data)

io:fwrite(Format)

io:fwrite(Format, Data)

io:fwrite(IoDevice, Format, Data)

1> io:fwrite("|**~s**|**~n**", [a234567890b]).

|a234567890b|

ok

2> io:fwrite("|**~10s**|**~n**", [a234567890b]).

|a234567890|

ok

3> io:fwrite("|**~10.2s**|**~n**", [a234567890b]).

| a2|

ok

1> io:fwrite("**~s~n**", [[65,66,255]]).

ABÿ

ok

2> io:fwrite("**~s~n**", [[65,66,256]]).

\*\* exception error: bad argument

in function io:fwrite/2

called as io:fwrite("**~s~n**",[[65,66,256]])

\*\*\* argument 1: failed to format string

3> io:fwrite("**~ts~n**", [[65,66,256]]).

ABĀ

ok

1. A=[[{[{a123},{b123},{c123}]},{[{aa123},{bb123},{cc123}]}], {aaa},[bbb]].

[[{[{a123},{b123},{c123}]},{[{aa123},{bb123},{cc123}]}],

{aaa},

[bbb]]

1. io:fwrite("**~w~n**", [A]).

[[{[{a123},{b123},{c123}]},{[{aa123},{bb123},{cc123}]}],{aaa}, [bbb]]

ok

1. io:fwrite("**~p~n**", [A]).

[[{[{a123},{b123},{c123}]},{[{aa123},{bb123},{cc123}]}],{aaa}, [bbb]]

ok

1. io:fwrite("**~50p~n**", [A]).

[[{[{a123},{b123},{c123}]},

{[{aa123},{bb123},{cc123}]}],

{aaa},

[bbb]]

ok

io:fread (Prompt, Format)

io:fread(IoDevice, Prompt, Format)

مثال:

1>io:fread("enter your name> ","**~s**").

enter your name> arman

{ok,["arman"]}

1> {ok, T1} = file:open("term6.txt", [read,write]).

{ok,<0.84.0>}

2> io:fwrite(T1,"number\_1=**~w~n**", [{2024}]).

ok

3> file:close(T1).

ok

4> {ok, T2} = file:open("term6.txt", [read,write]).

{ok,<0.96.0>}

1. io:fread(T2," ","**~s**").

{ok,["number\_1={2024}"]}

6> file:close(T2).

ok

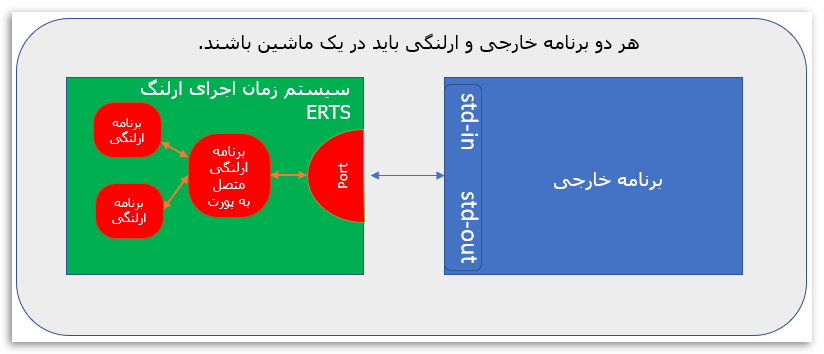
# 8-3 تا اینجا یاد گرفتیم:

# فصل9: ارتباط ارلنگ با زبان c

# 9-1 مکانیزم های که ارلنگ برای ارتباط با دیگر زبان ها دارد:

# 9-1-1 ارلنگ توزیع شده

# 9-1-2 پورت ها



تصویر 9.1 : شکل ارتباط پورتی

# 9-2 کتابخانه C

binary\_to\_term(Binary)

term\_to\_binary(Erlang\_Term)

# 9-3 C Node

# 9-4 Linked-in Drivers

# 9-5 NIF (Native Implemented Functions)

# 9-6 ارتباط یک برنامه خارجی به زبان C با یک برنامه به زبان Erlang با استفاده از پورت

# 9-7 وظایف کد cmain:

cmain.c

1. #include <stdio.h>
2. #include <unistd.h>
3. typedef unsigned char byte;
4. int main()
5. {
6. int fn, arg, res;
7. byte buf[100];
8. while (read\_cmd(buf) > 0)
9. {
10. fn = buf[0];
11. arg = buf[1];
12. if (fn == 1)
13. {
14. res = foo(arg);
15. }
16. else if (fn == 2)
17. {
18. res = bar(arg);
19. }
20. buf[0] = res;
21. write\_cmd(buf, 1);
22. }
23. }

cmain.c

1. /\*read(read\_cmd)----- --------------------\*/
2. int read\_cmd(byte \*buf) /\*byte buf[100](line:7)///called in line:8\*/
3. {
4. int len;
5. if (read\_exact(buf, 2) != 2)
6. return (-1);
7. len = (buf[0] << 8) | buf[1];
8. return read\_exact(buf, len);
9. }
10. /\*read(read\_exact)-------------------\*/
11. int read\_exact(byte \*buf, int len)
12. {
13. int I, got = 0;
14. do
15. {
16. if ((I = read(0, buf + got, len – got)) <= 0)
17. {
18. return (i);
19. }
20. got += I;
21. } while (got < len);
22. return (len);
23. }

if ((i = read(0, buf, len)) <= 0)

        {

            return (i);

        }

cmain.c

1. /\*write (write\_cmd)------ ---------------------\*/
2. int write\_cmd(byte \*buf, int len)
3. {
4. byte li;
5. li = (len >> 8) & 0xff;
6. write\_exact(&li, 1);
7. li = len & 0xff;
8. write\_exact(&li, 1);
9. return write\_exact(buf, len);
10. }
11. /\*write (write\_exact)------- ---------------\*/
12. int write\_exact(byte \*buf, int len)
13. {
14. int i, wrote = 0;
15. do
16. {
17. if ((i = write(1, buf + wrote, len - wrote)) <= 0)
18. return (i);
19. wrote += i;
20. } while (wrote < len);
21. return (len);
22. }

تابع write:

if ((i = write(1, buf, len)) <= 0)

            return (i);

li = len & 0xff;

# 9-8 کد کامل cmain.c :

cmain.c

#include <stdio.h>

#include <unistd.h>

typedef unsigned char byte;

int main()

{

    int fn, arg, res;

    byte buf[100];

    while (read\_cmd(buf) > 0)

    {

        fn = buf[0];

        arg = buf[1];

        if (fn == 1)

        {

            res = foo(arg);

        }

        else if (fn == 2)

        {

            res = bar(arg);

        }

        buf[0] = res;

        write\_cmd(buf, 1);

    }

}

/\*read(read\_cmd)-----------------------------------------------\*/

int read\_cmd(byte \*buf) /\*byte buf[100](line:7)///called in line:8\*/

{

    int len;

    if (read\_exact(buf, 2) != 2)

        return (-1);

    len = (buf[0] << 8) | buf[1];

    return read\_exact(buf, len);

}

/\*read(read\_exact)-----------------------------------------------\*/

int read\_exact(byte \*buf, int len)

{

    int i, got = 0;

    do

    {

        if ((i = read(0, buf + got, len - got)) <= 0)

        {

            return (i);

        }

        got += i;

    } while (got < len);

    return (len);

}

/\*write (write\_cmd)------------- ---------------------\*/

int write\_cmd(byte \*buf, int len)

{

    byte li;

    li = (len >> 8) & 0xff;

    write\_exact(&li, 1);

    li = len & 0xff;

    write\_exact(&li, 1);

    return write\_exact(buf, len);

}

/\*write (write\_exact)---------- ----------------------\*/

int write\_exact(byte \*buf, int len)

{

    int i, wrote = 0;

    do

    {

        if ((i = write(1, buf + wrote, len - wrote)) <= 0)

            return (i);

        wrote += i;

    } while (wrote < len);

    return (len);

}

# 9-9 وظایف کد cfunc1 :

cfunc1.c

/\* cfunc1.c -----------------------------\*/

int foo(int x) {

return x+1;

}

int bar(int y) {

return y\*2;

}

# 9-10 کار با کامپایلر برنامه های C

gcc -o cprog cmain.c cfunc1.c

# 9-11 ماژول ارلنگی erlang\_prog:

# open\_port

open\_port(PortName, PortSettings) -> port().

{spawn, Command}

{spawn\_driver, Command}

{spawn\_executable, Command}

{fd, In, Out}

{packet, N}

stream

{line, MaxBit}

{cd, Dir}

in

با وجود این مورد ،پورت فقط برای ورودی(input) استفاده می شود.

out

با وارد کردن این مورد ، پورت فقط برای خروجی استفاده می شود.

binary

# 9-12 لیست کردن پورت ها

برای لیست کردن پورت ها از تابع ports/0 استفاده می کنیم:

1> erlang:ports().

[#Port<0.1>,#Port<0.3>,#Port<0.4>,#Port<0.6>,#Port<0.7>]

# 9-13 بستن پورت

مثال:

1> [A,B]=erlang:ports().

[#Port<0.1>,#Port<0.2>]

2> A.

#Port<0.1>

3> port\_close(A).

true

4> erlang:ports().

[#Port<0.2>]

# 9-14 ارسال پیام به پورت

1. port\_command(Port, Data)
2. Port ! {PortOwner, {command, Data}}

# 9-15 تغییر مالک پورت

port\_connect(Port, Pid)

Port ! {Owner, {connect, Pid}}

# 9-16 توضیح قسمت ارلنگی پورت:

erlang\_prog.erl

1. start(ExtPrg) **->**
2. spawn(?MODULE, init, [ExtPrg]).
3. %-------------------------------
4. init(ExtPrg) **->**
5. register(complex, self()),
6. process\_flag(trap\_exit, true),
7. Port = open\_port({spawn, ExtPrg}, [{packet, 2}]),
8. loop(Port).

erlang\_prog.erl

1. %-------------------------------
2. stop() **->**
3. complex ! stop.
4. %-------------------------------
5. foo(X) **->**
6. call\_port({foo, X}).
7. %-------------------------------
8. bar(Y) **->**
9. call\_port({bar, Y}).
10. %-------------------------------
11. call\_port(Msg) **->**
12. complex ! {call, self(), Msg},
13. **receive**
14. {complex, Result} **->**
15. Result
16. **end**.

erlang\_prog.erl

1. %-------------------------------
2. loop(Port) **->**
3. **receive**
4. {call, Caller, Msg} **->**
5. Port ! {self(), {command, encode(Msg)}},
6. **receive**
7. {Port, {data, Data}} **->**
8. Caller ! {complex, decode(Data)}
9. **end**,
10. loop(Port);
11. stop **->**
12. Port ! {self(), close},
13. **receive**
14. {Port, closed} **->**
15. exit(normal)
16. **end**;
17. {'EXIT', Port, Reason} **->**
18. exit(port\_terminated)
19. **end**.
20. %-------------------------------

:

erlang\_prog.erl

1. %-------------------------------
2. encode({foo, X}) **->** [1, X];
3. encode({bar, Y}) **->** [2, Y].
4. decode([Int]) **->** Int.
5. %-------------------------------

در ادامه تمام ماژول erlang\_prog و روش اجرای آن در پوسته را می بینید:

erlang\_prog.erl

1. -module(erlang\_prog).
2. -export([start/1, stop/0, init/1]).
3. -export([foo/1, bar/1]).
4. %-------------------------------
5. start(ExtPrg) **->**
6. spawn(?MODULE, init, [ExtPrg]).
7. %-------------------------------
8. init(ExtPrg) **->**
9. register(complex, self()),
10. process\_flag(trap\_exit, true),
11. Port = open\_port({spawn, ExtPrg}, [{packet, 2}]),
12. loop(Port).
13. %-------------------------------
14. loop(Port) **->**
15. **receive**
16. {call, Caller, Msg} **->**
17. Port ! {self(), {command, encode(Msg)}},
18. **receive**
19. {Port, {data, Data}} **->**
20. Caller ! {complex, decode(Data)}
21. **end**,
22. loop(Port);
23. stop **->**
24. Port ! {self(), close},
25. **receive**
26. {Port, closed} **->**
27. exit(normal)
28. **end**;
29. {'EXIT', Port, Reason} **->**
30. exit(port\_terminated)
31. **end**.
32. %-------------------------------
33. encode({foo, X}) **->** [1, X];
34. encode({bar, Y}) **->** [2, Y].
35. decode([Int]) **->** Int.
36. %-------------------------------
37. stop() **->**
38. complex ! stop.
39. %-------------------------------
40. foo(X) **->**
41. call\_port({foo, X}).
42. %-------------------------------
43. bar(Y) **->**
44. call\_port({bar, Y}).
45. %-------------------------------
46. call\_port(Msg) **->**
47. complex ! {call, self(), Msg},
48. **receive**
49. {complex, Result} **->**
50. Result
51. **end**.

اجرا در پوسته :

1> c(erlang\_prog).

…

{ok,erlang\_prog}

2> erlang\_prog:start("./cprog").

<0.132.0>

3> erlang\_prog:foo(5).

6

4> erlang\_prog:bar(5).

13

# فصل 10: مدیریت خطا در برنامه های ترتیبی

# 10-1 معرفی انواع خطا :

# 10-2 انواع کلاس های خطا

error(Reason)

{'EXIT',{Reason,Stack}}

در صورت گرفتن آن با دستور catch مقداری شبیه زیر دریافت می کنیم:

**catch** error(term).

{'EXIT',{term,[{shell,apply\_fun,3....

throw (Reason)

my\_throw1.erl

1. throw1()**->**
2. io:format("pid throw1/0 **~w~n**",[self()]),
3. throw(my\_throw).
4. .
5. .
6. function2()**->**
7. spawn\_link(my\_throw1,throw1,[]),
8. **receive**
9. Msg**->**
10. io:format("exception:**~w~n**", [Msg])
11. **after** 2000 **->**
12. io:format("Time is up:**~w~n**",[self()])
13. **end**.

1> my\_throw1:function2().

pid throw1/0 <0.122.0>

exception:{'EXIT',<0.122.0>,{{**nocatch**,my\_throw},

....

my\_throw1.erl

1. throw1()**->**
2. io:format("pid throw1/0 **~w~n**",[self()]),
3. throw(my\_throw).
4. .
5. .
6. catch\_throw() **->**
7. **try** throw1() **of**
8. \_A **->** io:format("receive:**~w** **~n**",[\_A])
9. **catch**
10. Class:Exception **->** {catch\_section,Class,Exception}
11. **after**
12. io:format("the after\_section **~n**")
13. **end**.

1> my\_throw1:catch\_throw().

pid throw1/0 <0.126.0>

the after\_section

{catch\_section,throw,my\_throw}

1> throw (my\_reason).

\*\* exception throw: my\_reason

2> catch throw (my\_reason).

my\_reason

exit (Reason) و exit(Pid, Reason)

1> **catch** exit(reason).

{'EXIT',reason}

مثال برای کلاس error:

error1.erl

1. -module(error1).
2. -export([do/1]).
3. do(I) **when** is\_atom(I)**->**
4. io:fwrite("Hi, to this program.**~w** **~n**",[I]).

1> error1:do(a).

Hi, welcom to this program.a

ok

2> error1:do(5).

\*\* exception error: no function clause matching error1:do(5) (error1.erl, line 4)

# 10-3 مدیریت خطا (Handling Errors)

# گرفتن خطا با try...catch

try Exprs of

Pattern1 [when GuardSeq1] -> Body1;

...;

PatternN [when GuardSeqN] -> BodyN

catch

Class1:ExceptionPattern1[:Stacktrace] [when ExceptionGuardSeq1] -> ExceptionBody1;

...;

ClassN:ExceptionPatternN[:Stacktrace] [when ExceptionGuardSeqN] -> ExceptionBodyN

after

AfterBody

end.

مثال تابع try\_catch/0 در ماژول error1 :

1. f2(1) **->** a;
2. f2(2) **->** b;
3. f2(3) **->** c.
4. try\_catch() **->**
5. [a2(I) || I <- [1,4,3]].
6. a2(N) **->**io:format("**~n**=> a2/1 the function **~n**"),
7. **try** f2(N) **of**
8. a **->** io:format("{a} try\_section **~n**");
9. b **->** io:format("{b} try\_section **~n**");
10. c **->** io:format("{c} try\_section **~n**")
11. **catch**
12. Class:Exception **->** {catch\_section,Class,Exception,N}
13. **after**
14. io:format("the after\_section **~n**")
15. **end**.

در ادامه اجرای تابع try\_catch/0 را در پوسته داریم:

1> error1:try\_catch().

1. a2/1 the function
2. {a} try\_section
3. the after\_section
4. a2/1 the function
5. the after\_section
6. a2/1 the function
7. {c} try\_section
8. the after\_section
9. [ok,{catch\_section,error,function\_clause,4},ok]

# 10-5 Catch روش دیگر به دام انداختن استثنا ها

1. catch\_1()**->**
2. [[**catch** I+1]||I<-[1,'a',"a"]].
3. catch\_2()**->**
4. [[I+1]||I<-[1,'a',"a"]].

1> error1:catch\_1().

[[2],

[{'EXIT',{badarith,[{error1,'-catch\_1/0-lc$^0/1-0-',1,

…

[{'EXIT',{badarith,[{error1,'-catch\_1/0-lc$^0/1-0-',1,

…

2> error1:catch\_2().

\*\* exception error: an error occurred when evaluating an arithmetic expression

in function error1:'-catch\_2/0-lc$^0/1-0-'/1 (error1.erl, line 102)

in call from error1:'-catch\_2/0-lc$^0/1-0-'/1 (error1.erl, line 102)

3> catch error1:catch\_2().

{'EXIT',{badarith,[{error1,'-catch\_2/0-lc$^0/1-0-',1,

[{file,"error1.erl"},{line,102}]},

{error1,'-catch\_2/0-lc$^0/1-0-',1,

[{file,"error1.erl"},{line,102}]},

{erl\_eval,do\_apply,7,[{file,"erl\_eval.erl"},{line,748}]},

{erl\_eval,expr,6,[{file,"erl\_eval.erl"},{line,480}]},

{shell,exprs,7,[{file,"shell.erl"},{line,691}]},

{shell,eval\_exprs,7,[{file,"shell.erl"},{line,647}]},

{shell,eval\_loop,3,[{file,"shell.erl"},{line,632}]}]}}

# 10-3 انواع استثنائها:

مثال در پوسته:

1> error1:do(5).

\*\* exception error: no function clause matching error1:do(5) (error1.erl, line 4)

2> catch error1:do(5).

{'EXIT',{function\_clause,[{ error1,do,

[5],

[{file," error1.erl"},{line,4}]},

{erl\_eval,do\_apply,7,

[{file,"erl\_eval.erl"},{line,748}]},

{erl\_eval,expr,6,[{file,"erl\_eval.erl"},{line,480}]},

{shell,exprs,7,[{file,"shell.erl"},{line,691}]},

{shell,eval\_exprs,7,[{file,"shell.erl"},{line,647}]},

{shell,eval\_loop,3,

[{file,"shell.erl"},{line,632}]}]}}

**try\_clause**

مثال:

error1.erl

1. f1(1) **->** a;
2. f1(2) **->** b;
3. f1(3) **->** c.
4. try\_clause1() **->**
5. [a(I) || I <- [1,2,3,4]].
6. a(N) **->**
7. **try** f1(N) **of**
8. a **->** a1;
9. b **->** b2;
10. c **->** c3
11. **catch**
12. Class:Exception **->** {Class,Exception,N}
13. **end**.

1> catch error1:try\_clause1().

[a1,b2,c3,{error,function\_clause,4}]

**try** f1(N) **of**

a **->** a1;

%b -> b2;

c **->** c3

سپس برنامه را دوباره کامپایل و اجرا می کنیم:

2> c(error1).

{ok,error1}

3> catch error1:try\_clause1().

{'EXIT',{{**try\_clause**,b},

...

**Undef**

1> catch error1:do(5,6).

{'EXIT',{**undef**,[{error1,do,[5,6],[]},

…

2> catch error1:doxxx(5).

{'EXIT',{**undef**,[{error1,doxxx,[5],[]},

…

**function\_clause**

مثال:

3> catch error1:do(100).

{'EXIT',{**function\_clause**, [{error1,do,"d",

[{file,"error1.erl"},{line,4}]},

...

4> catch error1:do(a).

Hi, to this program.a

ok

**case\_clause**

مثال:

1. case1(N) **->**
2. **case** N **of**
3. a **->** a;
4. 1 **->** one
5. **end**.

اجازه دهید آن را در پوسته اجرا کنیم:

5> catch error1:case1(1).

one

6> catch error1:case1(0).

{'EXIT',{{**case\_clause**,0},

…

**if\_clause**

مثال :

1. if1(N)**->**
2. **if**
3. N == 1 **->** {'one'};
4. N == 2 **->** {'two'}
5. **end**.

اجازه دهید آن را در پوسته اجرا کنیم:

7> catch error1:if1(1).

{one}

8> catch error1:if1(3).

{'EXIT',{**if\_clause**,[{error1,if1,1,

…

مثال:

1. if\_case1(N) **->**
2. **case** N **of**
3. a **->** a;
4. 1 **->** one;
5. \_**->**
6. **if**
7. N < 0 **->** {'N < 0'};
8. N > 0 **->** {'N > 0'}
9. **end**
10. **end**.
11. %-------------------------------------------
12. if\_case2(N)**->**
13. **if**
14. N == 1 **->** {'one'};
15. N == 2 **->** {'two'};
16. N **->**
17. **case** N **of**
18. a **->** a;
19. 1 **->** one
20. **end**
21. **end**.

اجازه دهید آن را در پوسته اجرا کنیم:

9> catch error1:if\_case1(1).

one

10> catch error1:if\_case1(0).

{'EXIT',{if\_clause,[{error1,if\_case1,1,

…

11> catch error1:if\_case2(1).

{one}

12> catch error1:if\_case2(0).

{'EXIT',{if\_clause,[{error1,if\_case2,1,

…

**shell\_undef**

مثال:

13> catch error1:if\_case2(1).

{one}

14> catch if\_case2(1).

{'EXIT',{{**shell\_undef**,if\_case2,1,[]},

...

**badmatch**

1> A=5.

5

2> catch A=6.

{'EXIT',{{badmatch,6},

...

3> catch {a}={b}.

{'EXIT',{{badmatch,{b}},

...

1. badmatch1({Name,Pas})**->**
2. A={ali,pas\_ali},
3. A={Name,Pas},
4. io:format("welcome ali **~n**").

اجازه دهید آن را در پوسته فراخوانی کنیم:

4> catch error1:badmatch1({ali,pas\_ali}).

welcome ali

ok

5> catch error1:badmatch1({reza,passs\_reza}).

{'EXIT',{{badmatch,{reza,passs\_reza}},

…

**badarg**

مثال:

badarg1(A,B)**->**

A++B.

اجازه دهید آن را در پوسته فراخوانی کنیم:

1> catch error1:badarg1(5,4).

{'EXIT',{badarg,[{erlang,'++',

…

**badarith**

badarith1(A,B)**->**

A+B.

اجازه دهید آن را در پوسته اجرا کنیم:

1> catch error1:badarith1(a,4).

{'EXIT',{badarith,[{erlang,'+',

…

**badfun**

مثال:

sum1(A,B)**->**A+B.

bad\_fun(Sum,[A,B]) **->** Sum(A,B).

اجازه دهید آن را در پوسته اجرا کنیم:

1> error1:bad\_fun( fun error1:sum1/2 ,[1,2]).

3

2> error1:bad\_fun(a , [1,2]).

\*\* exception error: bad function a

in function error1:bad\_fun/2 (error1.erl, line 141)

**bad\_arity**

bad\_arity()**->** **fun**(A,B)**->** A + B **end**.

sum1(A,B)**->**A+B.

اجازه دهید آن را در پوسته اجرا کنیم:

1> R6= error1:bad\_arity().

#Fun<error1.0.97259097>

2> R6(1,2).

3

3> R6(1,2,3).

\*\* exception error: error1:'-**bad\_arity**/0-fun-0-'/2 called with 3 arguments

4> error1:sum1(1,2).

3

5> error1:sum1(1,2,3).

\*\* exception error: **undefined function** error1:sum1/3

باقی دلایل خروج

در ادامه باقی دلایل خروج را به صورت خلاصه در یک جدول می بینیم:

|  |  |
| --- | --- |
| معنای استثنا | دلیل استثنا |
| مقدار زمان در یک عبارت "receive..after" به چیزی غیر از یک عدد صحیح یا infinity ارزیابی شود. | timeout\_value |
| تلاش برای پیوند یا نظارت به یک فرایند یا پورت غیر موجود. | Noproc |
| یک پیوند یا مانیتور به یک فرایند راه دور، شکسته شد زیرا ارتباط بین گره ها برقرار نشد یا قطع شد. | noconnection |
| تلاش برای ارزیابی یک throw بدون ساختار catch  Term\_x عبارت پرتاب شده است. | {nocatch,Term\_x} |
| به محدودیت سیستم رسیده است. | system\_limit |

جدول 10.1 : دلایل خروج

# 10-6 چه زمانی از (exit/1 , error/1 , throw/1) استفاده می کنیم؟

**exit/1:**

**error/1:**

**throw/1:**

**مثال برای exit/1 , error/1 , throw/1 :**

-module(error2).

-compile(export\_all).

*%-------------------------------------------*

error2()->

io:format("start error.~n"),

error(my\_reason),

error3(),

io:format("end of error.~n").

error3()->io:format("start error3333333.~n").

*%-------------------------------------------*

throw2()->

io:format("start throw.~n"),

 throw(my\_reason),

throw3(),

io:format("end of throw.~n").

throw3()->io:format("start throw33333333.~n").

*%-------------------------------------------*

exit2()->

io:format("start exit.~n"),

 exit(my\_reason),

exit3(),

io:format("end of exit.~n").

exit3()->io:format("start exit33333333.~n").

*%-------------------------------------------*

اجازه دهید توابع این ماژول را در پوسته امتحان کنیم:

1> catch error2:error2().

start error.

{'EXIT',{my\_reason,[{error2,error2,0,

[{file,"error2.erl"},{line,20}]},

{erl\_eval,do\_apply,7,[{file,"erl\_eval.erl"},{line,748}]},

{erl\_eval,expr,6,[{file,"erl\_eval.erl"},{line,480}]},

{shell,exprs,7,[{file,"shell.erl"},{line,691}]},

{shell,eval\_exprs,7,[{file,"shell.erl"},{line,647}]},

{shell,eval\_loop,3,[{file,"shell.erl"},{line,632}]}]}}

2> catch error2:exit2().

start exit.

{'EXIT',my\_reason}

3> catch error2:throw2().

start throw.

my\_reason

# فصل 11: برنامه های همزمان

# 11-1 مفاهیم ضروری همزمانی:

* **همزمانی برنامه ها** :
* **فرق بین موازی و همزمانی: (این تعریف من است)**
* **فرایند** :
* **رشته اجرایی** :
* **Pid یا شناسه فرایندی** :
* **ارسال پیام** :
* **صندوق پستی فرایند**:
* **صف انتظار :**
* **مهلت زمانی**:
* **ثبت فرایندی** :
* **تولید مثل** :

# 11-2 فرایند و تولید مثل فرایندی :

برای تولید فرایند از شکل استاندارد زیر استفاده می کنیم:

spawn(Module\_name, Exported\_function, [List\_of\_Arguments]).

Pid1= spawn(**fun**()**->**Do\_somting() **end**).

مثال تولید فرایند به روش fun:

1> Pid3=

spawn(**fun**() **->** io:format("**~n**Please enter two numbers! **~n**"),

**receive** {A,B} **->** io:format("A+B=**~w~n**",[A+B]) **end** **end**).

Please enter two numbers!

<0.85.0>

2>

2> Pid3 ! {7,7}.

A+B=14

{7,7}

مثال:

1. -module(m1).
2. -export([sum1/2,sub1/2,start/0,start2/2]).
3. sum1(A,B) **->**
4. C= A+B,
5. io:fwrite("**~w**+**~w**=**~w** **~n**",[A,B,C]).
6. sub1(A,B) **->**
7. D= A-B,
8. io:fwrite("**~w**-**~w**=**~w** **~n**",[A,B,D]).
9. start()**->**
10. spawn (m1,sum1,[5,10]),
11. spawn (m1,sub1,[5,10]).
12. start2(A,B)**->**
13. spawn (m1,sum1,[A,B]),
14. spawn (m1,sub1,[A,B]).

1> m1:start().

5+10=15

5-10=-5

<0.90.0>

2>

2> m1:start2(4,5).

4+5=9

4-5=-1

<0.93.0>

# 11-3 ارسال و دریافت پیام بین فرایند ها:

receive

Pattern\_1 when GuardX -> Actions1;

..

Pattern\_N when GuardX -> ActionsN

End

نکته: با توجه به نکته قبلی اگر بخواهیم صف پیام ها را خالی کنیم باید الگویی را در قسمت receive قرار دهیم که با هر پیام مطابقت پیدا می کند مانند: (**\_ ->**) یا (X ->).

مثال :

1. -module(m3).
2. -export([multx/0,start/0]).
3. multx()**->**
4. **receive**
5. X **->**
6. Mul= **fun** (Number\_x) **->**
7. Number\_x \* 2 **end**,
8. Result = lists:map (Mul, X),
9. io:format("Result = **~w** **~n**", [Result])
10. **end**.
11. start()**->**
12. spawn (m3,multx,[]).

1> Pid1=m3:start().

<0.136.0>

2> Pid1![1,2,3,4,5,6,8].

Result = [2,4,6,8,10,12,16]

[1,2,3,4,5,6,8]

3> Pid1![1,2,3,4,5,6,8].

[1,2,3,4,5,6,8]

4> Pid2=m3:start().

<0.140.0>

5> Pid2![1,2,3,4,5,6,8].

Result = [2,4,6,8,10,12,16]

[1,2,3,4,5,6,8]

مثال:

1. -module(m4).
2. -export([a/0,b/1,start/0]).
3. %------------------------
4. a()**->**
5. **receive**
6. {X,Bpid1}**->**
7. io:format("a() pid =**~w**, a\_masege\_received\_from\_**~w**{**~w**}**~n**",[self(),Bpid1,X]),
8. Bpid1! {a\_masege,self()}
9. **end**,
10. a().
11. %------------------------
12. b([\_A,0])**->**
13. io:format("finished**~n**");
14. b([Apid1,X])**->**
15. Apid1 ! {X, self()},
16. **receive**
17. {a\_masege,Apid1}**->**
18. io:format("b() pid =**~w**, a\_masege\_received\_from\_**~w~n**",[self(),Apid1]),
19. b([Apid1,X-1]);
20. \_other **->** Apid1 ! {error},
21. b([Apid1,X])
22. **end**.
23. %------------------------
24. start()**->**
25. Apid= spawn (m4,a,[]),
26. spawn (m4,b,[[Apid,5]]).

1. start()**->**
2. Apid= spawn (m4,a,[]),
3. spawn (m4,b,[[Apid,5]]).
4. b([\_A,0])**->**
5. io:format("finished**~n**");
6. b([Apid1,X])**->**
7. Apid1 ! {X, self()},
8. **receive**
9. {a\_masege,Apid1}**->**
10. io:format("b() pid =**~w**, a\_masege\_received\_from\_**~w~n**",[self(),Apid1]),
11. b([Apid1,X-1]);
12. \_Other **->** Apid1 ! {error},
13. b([Apid1,X])
14. **end**.

1. \_Other **->** Apid1 ! {error},
2. b([Apid1,X])

1> c(m4).

{ok,m4}

2> m4:start().

a() pid =<0.216.0>, a\_masege\_received\_from\_<0.217.0>{5}

<0.217.0>

b() pid =<0.217.0>, a\_masege\_received\_from\_<0.216.0>

a() pid =<0.216.0>, a\_masege\_received\_from\_<0.217.0>{4}

b() pid =<0.217.0>, a\_masege\_received\_from\_<0.216.0>

a() pid =<0.216.0>, a\_masege\_received\_from\_<0.217.0>{3}

b() pid =<0.217.0>, a\_masege\_received\_from\_<0.216.0>

a() pid =<0.216.0>, a\_masege\_received\_from\_<0.217.0>{2}

b() pid =<0.217.0>, a\_masege\_received\_from\_<0.216.0>

a() pid =<0.216.0>, a\_masege\_received\_from\_<0.217.0>{1}

b() pid =<0.217.0>, a\_masege\_received\_from\_<0.216.0>

finished

# 11-4 مهلت ها :

1. receive
2. Pattern\_1 [when Guard\_1] ->
3. Body\_1;
4. ...;
5. Pattern\_N [when Guard\_N] ->
6. Body\_N
7. after
8. Time ->
9. BodyTimer
10. end

* **در صورت تعیین مقدار صفر:**

* **در صورت تعیین مقدار اتم infinity :**

1. -module(timer1).
2. -compile(export\_all).
3. %-------------------------------------
4. start(Time, F) **->** spawn(**fun**() **->** timer1:a(Time, F) **end**).
5. a(Time, F) **->**
6. **receive**
7. Msg**->**
8. io:format("This message has been received:**~w~n**", [Msg])
9. **after** Time **->**
10. io:format("Time is up.**~n**"),
11. F()
12. **end**.
13. f1()**->**io:format("run f1.**~n**").

1> P1 = timer1:start(1000,fun()->timer1:f1()end).

<0.96.0>

Time is up.

run f1.

1. start2(Time, Fun) **->** spawn(timer1,a2,[Time, Fun]).
2. a2(Time, F) **->**
3. **receive**
4. Msg**->**
5. io:format("This message has been received:**~w~n**", [Msg])
6. **after** Time **->**
7. io:format("Time is up.**~n**"),
8. F()
9. **end**.

مثال :

1. start3(Time, Fun) **->** spawn(timer1,a3,[Time, Fun]).
2. a3(Time, F) **->**
3. **receive**
4. **after** Time **->**
5. io:format("Time is up.**~n**"),
6. F()
7. **end**.

در پوسته :

1> P1 = timer1:start3(11000,fun()->timer1:f1()end).

<0.149.0>

Time is up.

run f1.

1> P2 = timer1:start(infinity,fun()->timer1:f1()end).

<0.159.0>

2> P2 ! hello.

This message has been received:hello

hello

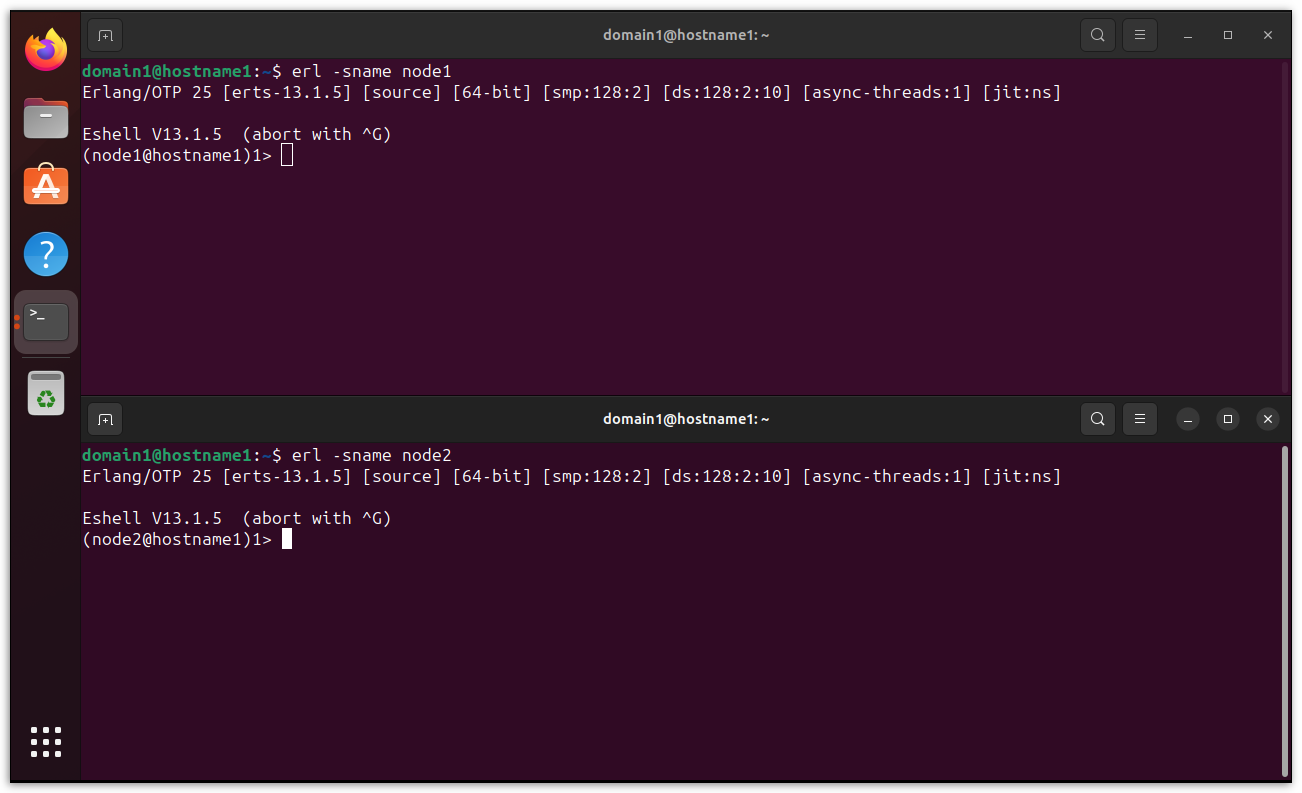
# فصل 12: فرایند های توزیع شده

# 12-1 ساخت گره:

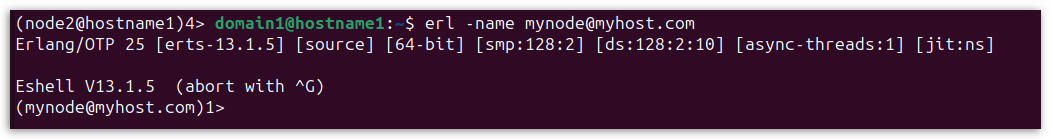
برای ساخت گره با نام کوتاه از دستور زیر استفاده می کنیم:

erl -sname Node\_name@Host\_name

erl -sname node\_name



تصویر 12.1 : شروع دو گره در دو ترمینال لینوکسی

  
تصویر 12.2 : نام بلند گره

1. -module(d).
2. -compile([export\_all]).
3. f(From)**->**
4. From ! node().
5. p()**->**
6. io:format("my pid = **~w** **~n** my node = **~w** **~n**" ,[self(),node()]).
7. c(A,B)**->**
8. C2= A+B,
9. io:format("A+B = **~w** **~n**",[C2]).

1> d:f(self()).

node1@hostname1

2> d:p().

my pid = <0.81.0>

my node = node1@hostname1

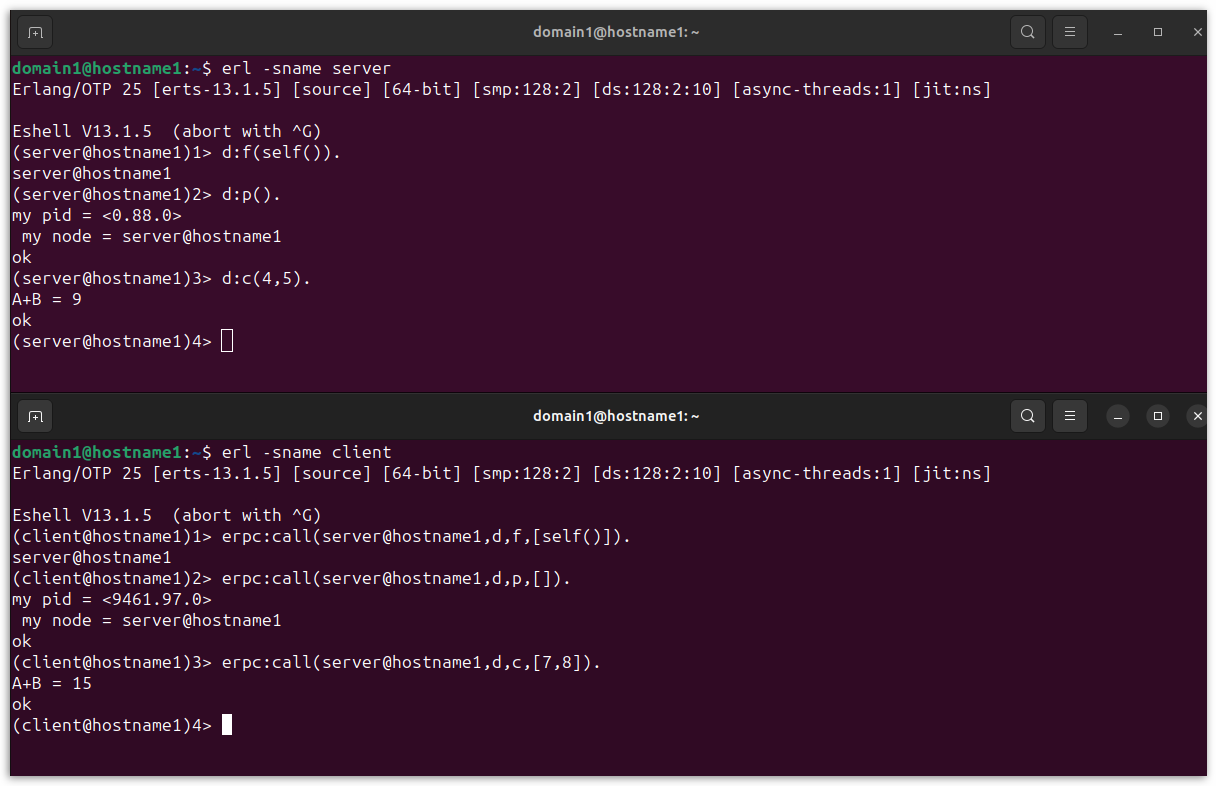
ok

3> d:c(3,4).

A+B = 7

ok

نکته: هر دو گره باید از erpc پشتیبانی کنند . به طور معمول از OTP 23 به بعد از این ماژول پشتیبانی می شود.

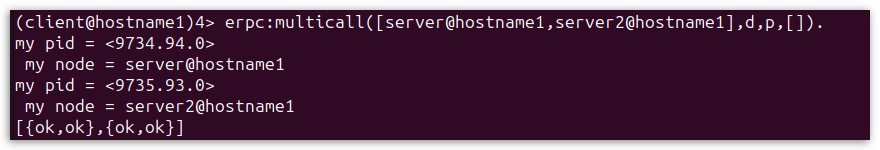


تصویر 12.3 : ارتباط بین گره ها

erpc:call(Node, Module, Function, [Args]).

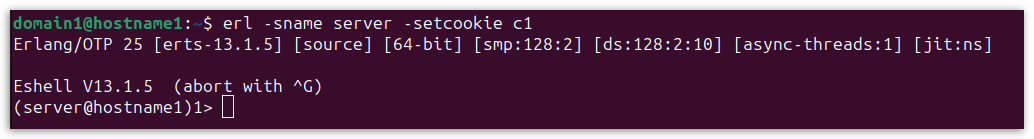
erpc: multicall ([Node1, Node2,…], Module, Function, [Args]).

مثال:

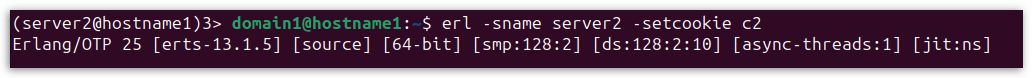


تصویر 12.4 : فراخوانی یک تابع در چند گره

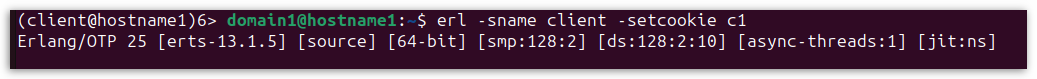
erpc:cast(Node, Module, Function, Args).



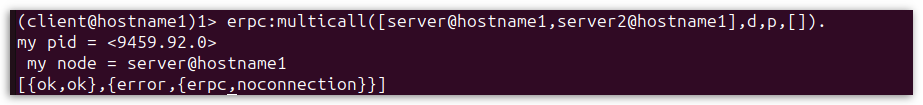
تصویر 12.5 : تنظیم کوکی برای گره server



تصویر 12.6 : تنظیم کوکی برای گره server2



تصویر 12.7 : تنظیم کوکی برای گره client



تصویر 12.8 : فراخوانی یک تابع در چند گره بعد از تنظیم کوکی

# **12-2 تولید فرایند راه دور:**

spawn(Node, Fun) -> pid()

spawn\_link(Node, Fun) -> pid()

spawn(Node, Module, Function, Args) -> pid()

spawn\_link(Node, Module, Function, Args) -> pid()

1. -module(distribution).
2. -compile([export\_all]).
3. %-----------------------------------------------------------------------
4. start1()**->**
5. **receive**
6. {print\_address} **->**
7. io:format("my pid = **~w** **~n** my node = **~w** **~n**" ,[self(),node()]);
8. {calculator,A,B} **->**
9. C2= A+B,
10. io:format("A+B = **~w** **~n**",[C2]);
11. \_Other **->**
12. io:format("error **~n**"),
13. start1()
14. **end**.
15. %-----------------------------------------------------------------------
16. start2(Node) **->**
17. spawn(Node, **fun**() **->** start1() **end**).
18. %-----------------------------------------------------------------------
19. start3(Node)**->**
20. spawn(Node,io,format,["my pid = **~w** **~n** my node = **~w** **~n**" ,[self(),node()]]).

%-----------------------------------------------------------------------

~$ erl -sname node1

~$ erl -sname node2

(node1@computername1)1> Pid1=distribution:start2(node2@computername1).

<8995.92.0>

(node1@computername1)2> Pid1 ! {print\_address}.

{print\_address}

my pid = <8995.92.0>

my node = node2@computername1

(node1@computername1)3> Pid2=distribution:start3(node2@computername1).

<8995.93.0>

my pid = <0.86.0>

my node = node1@computername1

(node1@computername1)4> Pid3 = spawn (node2@computername1,**fun**() **->** distribution:start1() **end**).

<8995.93.0>

(node1@computername1)5> Pid3 ! {print\_address}.

{print\_address}

my pid = <8995.93.0>

my node = node2@computername1

(node1@computername1)6> Pid4 = spawn (node2@computername1,distribution,start1,[]).

<8995.94.0>

(node1@computername1)7> Pid4 ! {print\_address}.

{print\_address}

my pid = <8995.94.0>

my node = node2@computername1

net\_adm:ping(Node)

net\_kernel:connect\_node(Node).

disconnect\_node(Node).

# **12-3 BIF های کوکی:**

erlang:set\_cookie(Cookie) -> true

erlang:set\_cookie(Node, Cookie) -> true

erlang:get\_cookie() -> Cookie | nocookie

# 12-4 ثبت فرایند ها در گره های دور:

{Registered\_Name , Node\_name} **!** Message

{Registered\_Name , Node\_name1} **!** {Registered\_Name, Node\_name2} **!** Message

1. -module(node\_reg).
2. -compile([export\_all]).
3. %-----------------------------------------------------------------------
4. start1()**->**
5. **receive**
6. {print\_address} **->**
7. io:format("my pid = **~w** **~n** my node = **~w** **~n**", [self(),node()]),
8. start1();
9. {down} **->** ok,{down}.
10. \_Other **->**
11. io:format("error **~n**"),
12. start1()
13. **end**.
14. %-----------------------------------------------------------------------
15. start2() **->**
16. Pid1=spawn(**fun**() **->** start1() **end**),
17. register (reg\_pid1,Pid1).
18. %-----------------------------------------------------------------------

گره اول client :

1. ~$ erl -sname client
2. (client@computername1)1> {reg\_pid1,server2@computername1} !

{reg\_pid1,server1@computername1} !

{print\_address}.

1. {print\_address}

گره دوم server1 :

1. **~$** erl -sname server1
2. (server1@computername1)1> node\_reg:start2().
3. true
4. my pid = <0.100.0>
5. my node = server1@computername1

گره سوم server2 :

1. ~$ erl -sname server2
2. (server2@computername1)1> node\_reg:start2().
3. true
4. my pid = <0.101.0>
5. my node = server2@computername1

node(pid).

مثال:

(client@computername1)1> node(<0.101.0>).

server2@computername1

# فصل 13: مدیریت خطا در فرایند های همزمان

# 13-1 لینک کردن فرایند ها:

Pid=spawn(Module\_name,Function\_name,Arg),

link(Pid).

مثال:

1. -module(link1).
2. -compile([export\_all]).
3. %-------------------------------------------------
4. link\_to(Module\_name,Function\_name,Arg)**->**
5. Pid=spawn(Module\_name,Function\_name,Arg),
6. link(Pid).
7. link\_to2()**->**
8. Pid=spawn(**fun**()**->**io:format("link\_to2**~n**")**end**),
9. link(Pid).
10. link\_to3()**->**
11. io:format("link\_to3**~n**"),
12. **receive**
13. Msg**->**
14. io:format("This message has been received:**~w~n**", [Msg])
15. **after** 2000 **->**
16. io:format("Time is up.**~n**")
17. **end**.

در پوسته :

1> link1:link\_to(link1,link\_to2,[]).

link\_to2

true

2> link1:link\_to(link1,link\_to3,[]).

link\_to3

true

Time is up.

3> link1:link\_to2().

link\_to2

true

فرم کلی مانند ادامه است:

spawn\_link (Module\_name,Function\_name,Arg)

مثال:

link\_to4(Module\_name,Function\_name,Arg)**->**

\_Pid=spawn\_link(Module\_name,Function\_name,Arg).

در پوسته:

4> link1:link\_to4(link1,link\_to3,[]).

link\_to3

<0.151.0>

Time is up.

# 13-2 مانیتور کردن یک فرایند:

پیام خروج با فرمت زیر خواهد بود:

{'DOWN', Ref, process, Pid, Reason}

شکل کلی در ادامه آمده است:

{Pid, Ref} = spawn\_monitor(Fun).

{Pid, Ref} = spawn\_monitor(M,F,A).

1. -module(monitor1).
2. -compile([export\_all]).
3. %-------------------------------------------------
4. start1()**->**
5. Pid1=spawn(monitor1,p1,[]),
6. Pid2=spawn(monitor1,p2,[Pid1]),
7. spawn(monitor1,m1,[Pid1,Pid2]),
8. io:format("line'8':**~n**pid1>**~w~n**pid2>**~w~n~n**",[Pid1,Pid2]).
9. %-------------------------------------------------
10. p2(Pid1)**->**
11. Pid1 ! {self(),{2,2}},
12. **receive**
13. {C} **->**
14. io:format("line 14 : **~n**2+2= **~w~n~n**",[C])
15. **after** 10000 **->**
16. io:format("Time is up.(p2)**~n~n**")
17. **end**.
18. %-------------------------------------------------
19. p1()**->**
20. **receive**
21. {M1, {A,B}} **->**
22. M1 ! {{A+B}}
23. **after** 10000 **->**
24. io:format("Time is up.(p1)**~n~n**")
25. **end**.
26. %-------------------------------------------------
27. m1(Pid1,Pid2)**->**
28. Ref1 = erlang:monitor(process, Pid1),
29. Ref2 = erlang:monitor(process, Pid2),
30. io:format("line'30':**~n**ref1>**~w~n**ref2>**~w~n~n**", [Ref1,Ref2]),
31. **receive**
32. {'DOWN', Ref2, process, Pid2, \_Reason2} **->**
33. io:format("line33: ref\_2: **~w~n** Reason2:**~w~n**" ,[Ref2,\_Reason2]),
34. ok
35. **end**,
36. **receive**
37. {'DOWN', Ref1, process, Pid1, \_Reason1} **->**
38. io:format("line38: ref\_1: **~w~n** Reason1:**~w~n~n**" ,[Ref1,\_Reason1]),
39. ok
40. **end**.
41. %-------------------------------------------------
42. Ref1 = erlang:monitor(process, Pid1),
43. Ref2 = erlang:monitor(process, Pid2),
44. io:format("line'30':**~n**ref1>**~w~n**ref2>**~w~n~n**" ,[Ref1,Ref2]),

1. **receive**
2. {'DOWN', Ref1, process, Pid1, \_Reason1} **->**
3. io:format("line38: ref\_1: **~w~n** Reason1:**~w~n~n**" ,[Ref1,\_Reason1]),
4. ok
5. **end**.

1. **receive**
2. {'DOWN', Ref2, process, Pid2, \_Reason2} **->**
3. io:format("line33: ref\_2: **~w~n** Reason2:**~w~n**" ,[Ref2,\_Reason2]),
4. ok
5. **end**,

در پوسته:

1> monitor1:start1().

line'8':

pid1><0.105.0>

pid2><0.106.0>

line'30':

ref1>#Ref<0.1108574724.2027159558.249834>

ref2>#Ref<0.1108574724.2027159558.249835>

line 14 :

2+2= {4}

line33: ref\_2: #Ref<0.1108574724.2027159558.249835>

Reason2:normal

ok

line38: ref\_1: #Ref<0.1108574724.2027159558.249834>

Reason1:normal

%-------------------------------------------------

1. m1(Pid1,Pid2)**->**
2. Ref1 = erlang:monitor(process, Pid1),
3. Ref2 = erlang:monitor(process, Pid2),
4. io:format("line'30':**~n**ref1>**~w~n**ref2>**~w~n~n**" ,[Ref1,Ref2]),
5. m2(Pid1,Pid2,Ref1,Ref2).
6. m2(Pid1,Pid2,Ref1,Ref2)**->**
7. **receive**
8. {'DOWN', Ref2, process, Pid2, \_Reason2} **->**
9. io:format("line36: ref\_2: **~w~n** Reason2:**~w~n**" ,[Ref2,\_Reason2]),
10. ok;
11. {'DOWN', Ref1, process, Pid1, \_Reason1} **->**
12. io:format("line40: ref\_1: **~w~n** Reason1:**~w~n~n**" ,[Ref1,\_Reason1]),
13. ok
14. **end**,
15. m2(Pid1,Pid2,Ref1,Ref2).

%-------------------------------------------------

در پوسته :

1> monitor3:start1().

line'8':

pid1><0.116.0>

pid2><0.117.0>

line'30':

ref1>#Ref<0.2586054147.2706112518.211033>

ref2>#Ref<0.2586054147.2706112518.211034>

line 14 :

2+2= {4}

line40: ref\_1: #Ref<0.2586054147.2706112518.211033>

Reason1:normal

ok

line36: ref\_2: #Ref<0.2586054147.2706112518.211034>

Reason2:normal

13-2-1 حذف یک مانیتور

حذف یک مانیتور با دستور زیر انجام می شود:

demonitor(MonitorRef) -> true

demonitor(MonitorRef, OptionList) -> boolean()

# 13-3 ثبت فرایند ها:

register(Name, PID).

unregister(PID).

می توانید لیستی از فرایند های ثبت شده را با دستورهای زیر بدست آورید:

regs().

registered().

برای بدست آوردن PID یک نام ثبت شده می توان از دستور زیر استفاده کرد:

whereis(Registered\_Name).

مثال:

1. -module(regi).
2. -compile([export\_all]).
3. %-------------------------------------------------
4. a() **->**
5. Pid\_b=spawn(?MODULE, b, []),
6. register(a\_pid, Pid\_b).
7. %-------------------------------------------------
8. b() **->**
9. **receive**
10. {\_Txt}**->**
11. io:format("<b() receive a text>**~n~w**",[\_Txt]),
12. b()
13. **end**.
14. %-------------------------------------------------
15. c(Txt)**->**
16. a\_pid ! Txt.
17. %-------------------------------------------------

در پوسته:

1> regi:a().

true

2> regi:c({mytext}).

<b() receive a text>

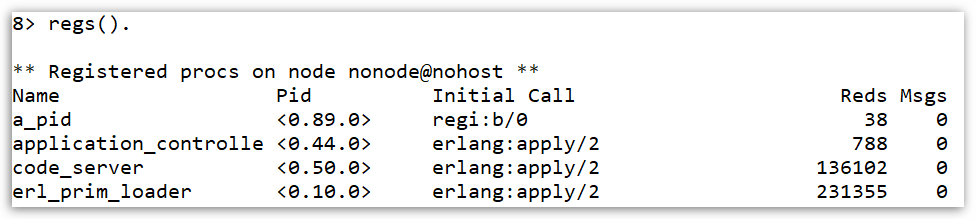
mytext{mytext}

3> whereis(a\_pid).

<0.89.0>

4> is\_pid(a\_pid).

false



تصویر 13.1 : نتیجه دستور regs

مثال:

1> list\_to\_existing\_atom("a\_pid").

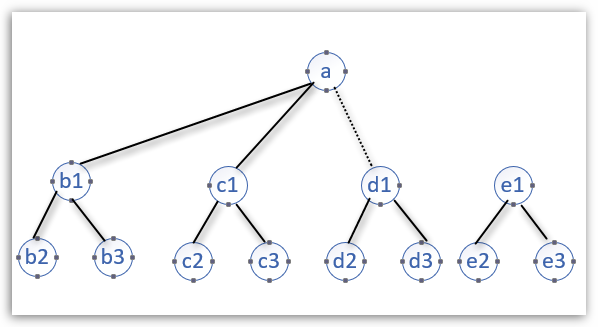
a\_pid

2> list\_to\_existing\_atom("a\_pid555").

\*\* exception error: bad argument...

# 13-4 فرایند های سیستمی و به تله انداختن یک استثنا :

{'EXIT', PID,Reason}



13.2 : انواع اتصال "لینک یا مانیتور" و تاثیر آن بر دریافت پیام یا سیگنال خروج

# 13-5 دریافت انتخابی با مرجع ها:

e1()**->**

**receive**

Any\_Message **->** handle\_msg(Any\_Message)

**end**.

مثال:

%-------------------------------------------------

e2({Tag\_x,Message\_x})**->**

P1 ! {Tag\_x,Message\_x}

**receive**

{Tag\_x, Reply } **->** handle\_msg(Reply)

**end**.

%-------------------------------------------------

مانند زیر :

1> make\_ref().

#Ref<0.1888683861.320602113.180801>

2> monitor(process, Pid1).

#Ref<0.1888683861.320602113.180806>

Pid1 باید PID یک فرایند باشد.

کد ماژول ref1 :

1. -module(ref1).
2. -compile([export\_all]).
3. %-------------------------------------------------
4. start1()**->**
5. Pid\_calculator1=spawn(ref1,calculator1,[]),
6. spawn(ref1,init1,[Pid\_calculator1]),
7. Pid3=spawn(ref1,ali,[]), register(pid\_ali,Pid3),
8. Pid4=spawn(ref1,reza,[]), register(pid\_reza,Pid4).
9. %-------------------------------------------------
10. init1(Pid\_calculator1)**->**
11. Ref1 = make\_ref(),
12. Pid\_calculator1 ! {2,2,self(),Ref1},
13. Ref2 = make\_ref(),
14. Pid\_calculator1 ! {5,5,self(),Ref2},
15. receiver1(Pid\_calculator1,Ref1,Ref2).
16. %-------------------------------------------------
17. receiver1(Pid\_calculator1,Ref1,Ref2)**->**
18. **receive**
19. {C,Pid\_calculator1,Ref1}**->**
20. pid\_ali ! {C},
21. receiver1(Pid\_calculator1,Ref1,Ref2);
22. {C,Pid\_calculator1,Ref2}**->**
23. pid\_reza ! {C},
24. receiver1(Pid\_calculator1,Ref1,Ref2)
25. **end**.
26. %-------------------------------------------------
27. calculator1()**->**
28. **receive**
29. {A,B,Pid\_init1,Ref\_x}**->**
30. Pid\_init1 ! {A+B,self(),Ref\_x},
31. calculator1()
32. **end**.
33. %-------------------------------------------------
34. ali()**->**
35. **receive**
36. {C}**->**
37. io:format("**~n**2+2= **~w~n**",[C]),
38. ali()
39. **end**.
40. %-------------------------------------------------
41. reza()**->**
42. **receive**
43. {C}**->**
44. io:format("**~n**5+5= **~w~n**",[C]),
45. reza()
46. **end**.
47. %-------------------------------------------------

اجرای Ref1 در پوسته :

1> ref1:start1().

2+2= 4

5+5= 10

true

ماژول ref2.erl :

1. -module(ref2).
2. -compile([export\_all]).
3. *%-------------------------------------------------*
4. start1()->
5. Pid\_calculator1=spawn(ref2,calculator1,[]),
6. spawn(ref2,init1,[Pid\_calculator1]),
7. Pid3=spawn(ref2,ali,[]),  register(pid\_ali,Pid3),
8. Pid4=spawn(ref2,reza,[]), register(pid\_reza,Pid4).
9. *%-------------------------------------------------*
10. init1(Pid\_calculator1)->
11. MRef1 = monitor(process, Pid\_calculator1),
12. Pid\_calculator1 ! {2,2,self(),MRef1},
13. Ref2 = make\_ref(),
14. Pid\_calculator1 ! {5,5,self(),Ref2},
15. receiver1(Pid\_calculator1,MRef1,Ref2).
16. *%-------------------------------------------------*
17. receiver1(Pid\_calculator1,MRef1,Ref2)->
18. receive
19. {C,Pid\_calculator1,MRef1}->
20. pid\_ali ! {C},
21. receiver1(Pid\_calculator1,MRef1,Ref2);
22. {C,Pid\_calculator1,Ref2}->
23. pid\_reza ! {C};
24. {'DOWN', MRef1, \_, \_, Reason} ->
25. io:format("Reason DOWN : ~w ~n ",[Reason]),
26. receiver1(Pid\_calculator1,MRef1,Ref2)
27. end.
28. *%-------------------------------------------------*
29. calculator1()->
30. receive
31. {A,B,Pid\_init1,Ref\_x}->
32. Pid\_init1 ! {A+B,self(),Ref\_x}
33. *% ,*
34. *%calculator1()*
35. end.
36. *%-------------------------------------------------*
37. ali()->
38. receive
39. {C}->
40. io:format("~n2+2= ~w ~n ",[C]),
41. ali()
42. end.
43. *%-------------------------------------------------*
44. reza()->
45. receive
46. {C}->
47. io:format("~n5+5=   ~w~n ",[C]),
48. reza()
49. end.
50. *%-------------------------------------------------*

اجازه بدهید این ماژول را در پوسته اجرا کنیم:

1. c(ref2). ...
2. {ok,ref2}
3. ref2:start1().
4. Reason DOWN : normal
5. 2+2= 4
6. true

is\_reference(Ref).

# فصل 14: جدول ذخیره ترم های ارلنگی: ETS و DETS



# 14-1 انواع جدول ها:

# 14-1-1 نکته ها:



# 14-1-2 استثنا ها :



# 14-1-3 توابع ماژول ETS:

ets:new(Name, [Options])



مثال:

1> TableId\_1 = ets:new(table\_name\_1,[named\_table]).

table\_name\_1

2> TableId\_2 = ets:new(table\_name\_2,[]).

#Ref<0.2602051908.4032954372.89808>



insert(Table, ObjectOrObjects)

مثال:

ets:insert(TableId, [{key\_1,val\_1},{key\_2,val\_2}]).

ets:insert(TableId, {key\_1,val\_3}).

مثال:

1. -module(ets1).
2. -export([start/0]).
3. %-------------------------------------------------
4. start() **->**
5. ets\_table(set\_table ,set),
6. ets\_table(ordered\_set\_table ,ordered\_set),
7. ets\_table(bag\_table ,bag),
8. ets\_table(duplicate\_bag\_table ,duplicate\_bag).
9. %-------------------------------------------------
10. ets\_table(Table\_name,Mode) **->**
11. TableId = ets:new(Table\_name, [Mode]),
12. ets:insert(TableId, {key\_1,val\_1}),
13. ets:insert(TableId, {key\_1,val\_1}),
14. ets:insert(TableId, {key\_1,val\_3}),
15. ets:insert(TableId, {key\_2,val\_2}),
16. ets:insert(TableId, {key\_x,key\_x}),
17. List\_of\_tuples = ets:tab2list(TableId),
18. io:format("------------------------------------**~n**"),
19. io:format("**~w** => **~50p~n**", [Mode, List\_of\_tuples]).

1> ets1:start().

------------------------------------

set => [{key\_x,key\_x},

{key\_2,val\_2},

{key\_1,val\_3}]

------------------------------------

ordered\_set => [{key\_1,val\_3},

{key\_2,val\_2},

{key\_x,key\_x}]

------------------------------------

bag => [{key\_x,key\_x},

{key\_2,val\_2},

{key\_1,val\_1},

{key\_1,val\_3}]

------------------------------------

duplicate\_bag => [{key\_x,key\_x},

{key\_2,val\_2},

{key\_1,val\_1},

{key\_1,val\_1},

{key\_1,val\_3}]

ok

1> ets:all().

[logger,ac\_tab,#Ref<0.105866580.2559442950.46660>,

…]

ets:delete(Table\_name).

جدول Table را حذف می کند.

ets:delete(Table\_name, Key).

تمام اشیاء با کلید Keyرا از جدول Table\_name حذف می کند.

ets:delete\_all\_objects(Table\_name)

تمام اشیاء درون جدول Table\_name را حذف می کند.

ets:delete\_object(Table\_name, Object)

ets:file2tab(File\_name)

ets:first(Table\_name)

اولین کلید در جدول را برمی گرداند.

ets:from\_dets(ETS\_Table, DETS\_Table)

ets:give\_away(Table\_name, New\_owner\_Pid, [GiftData]).

ets:i().

اطلاعاتی مربوط به تمام جدول ها در ترمینال جاری را برمی گرداند.

ets:info(Table\_name).

اطلاعاتی درباره جدول برمی گرداند.

مثال:

1> ets:info(code\_names).

[...

{owner,<0.50.0>},

...]

توضیحات آن طولانی است اگر به آن نیاز داشتید به راهنما مراجعه کنید. Stdlib.pdf بخش ets تابع info.

ets:insert\_new(Table\_name, [ObjectOrObjects])

ets:last(Table\_name)

ets:lookup(Table\_name, Key)

ets:member(Table\_name, Key)

ets:next(Table\_name, Key1)

ets:prev(Table\_name, Key1)

پوسته 1:

%----------------------------------------------------------

1> Table\_bag=ets:new(b1, [**bag**, named\_table]).

b1

2> ets:insert(b1,[{a,1},{b,2},{c,3},{d,4},{e,5}]).

true

3> B1=ets:prev(b1,c).

**d**

پوسته 2:

%----------------------------------------------------------

1> Table\_db=ets:new(db, [**duplicate\_bag**, named\_table]).

db

2> ets:insert(db,[{a,1},{b,2},{c,3},{d,4},{e,5}]).

true

3> DB=ets:prev(db,c).

**d**

پوسته 3:

%----------------------------------------------------------

1> Table\_set=ets:new(s1, [**set**, named\_table]).

s1

2> ets:insert(s1,[{a,1},{b,2},{c,3},{d,4},{e,5}]).

true

3> Set1 =ets:prev(s1,c).

**d**

پوسته 4:

%----------------------------------------------------------

1> Table\_oset=ets:new(oset, [**ordered\_set**, named\_table]).

oset

2> ets:insert(oset,[{a,1},{b,2},{c,3},{d,4},{e,5}]).

true

3> Oset =ets:prev(oset,c).

**b**

%----------------------------------------------------------

ets:rename(Table, New\_Name)

tab2file(Table, Filename)

tab2file(Table, Filename, Options)

نکته:بعضی وقت ها این تابع کار نمی کند. بعداً نشان می دهم که در چنین شرایطی باید چکار کنید.

tab2list(Table)

تابع بالا لیستی از تمام اشیاء جدول Table را برمی گرداند.

to\_dets(Table, DetsTab)

whereis(TableName)

1. -module(ets2).
2. -compile([export\_all]).
3. %=================================
4. table\_to\_file()**->**
5. Table\_oset=ets:new(oset, [ordered\_set, named\_table]),
6. ets:insert(oset,[{a,1},{b,2},{c,3},{d,4},{e,5}]),
7. table\_to\_file(Table\_oset, "txt2.txt").
8. %..................................................
9. table\_to\_file(Table, File) **->**
10. Data = ets:tab2list(Table),
11. {ok, File\_ID} = file:open(File, [write]),
12. file:write(File\_ID, io\_lib:format("**~w**.", [Data])),
13. %ets:tab2file(Data, File\_ID),
14. file:close(File\_ID).
15. %===================================
16. file\_to\_table() **->**
17. {ok, Data} = file:consult("txt2.txt"),
18. Table = ets:new(oset2, [named\_table]),
19. lists:foreach(**fun**({Key, Value}) **->**
20. ets:insert(Table, {Key, Value}) **end**, hd(Data)),
21. List\_of\_tuples = ets:tab2list(Table),
22. io:format("**~w~n**", [List\_of\_tuples]).
23. %===================================

خط 5 یک جدول با نام ثبت شده oset و از نوع ordered\_set می سازد :

Table\_oset=ets:new(oset, [ordered\_set, named\_table]),

خط 6 اشیاء را در جدول ، درج می کند:

ets:insert(oset,[{a,1},{b,2},{c,3},{d,4},{e,5}]),

خط 7 تابعی را برای وارد کردن جدول به فایل فراخوانی می کند:

table\_to\_file(Table\_oset, "txt2.txt").

در خط 10 با تابع tab2list/1 جدول را تبدیل به لیست می کنیم:

Data = ets:tab2list(Table),

در خط 11 فایلی File را برای نوشتن باز می کند:

{ok, File\_ID} = file:open(File, [write]),

و اما قسمت مهم کد :

1. file:write(File\_ID, io\_lib:format("**~w**.", [Data])),
2. %ets:tab2file(Data, File\_ID),

file:close(File\_ID).

{ok, Data} = file:consult("txt2.txt"),

Table = ets:new(oset2, [named\_table]),

در خط 19 تا 20 کد ادامه را داریم:

19.lists:foreach(**fun**({Key, Value}) **->**

20.ets:insert(Table, {Key, Value}) **end**, hd(Data)),



1. List\_of\_tuples = ets:tab2list(Table),
2. io:format("**~w~n**", [List\_of\_tuples]).

1> c(ets2).

...

{ok,ets2}

2> ets2:table\_to\_file().

ok

3> ets2:file\_to\_table().

[{e,5},{d,4},{c,3},{b,2},{a,1}]

ok

با وارد کردن دستور خط 2 فایلی به نام txt2.txt ایجاد می شود و محتوای آن شبیه زیر است:

[{a,1},{b,2},{c,3},{d,4},{e,5}].

# 2-14 DETS:

# 14-2-1معرفی ماژول **DETS:**

# 14-2-2 توابع ماژول DETS:

در ادامه تعدادی از توابع کاربردی ماژول dets را خواهیم دید.

dets:all()

تابع بالا، لیستی از تمام جدول های باز شده را برمی گرداند.

dets:close(Tab\_name)

تابع بالا، یک جدول باز را می بندد.

dets:delete(Table\_name, 1) .

تابع بالا، هر شی با کلید 1 در جدول Table\_name را حذف می کند.

delete\_all\_objects(Table\_name)

تابع بالا، تمام اشیا در جدول Table\_name را حذف می کند.

dets:delete\_object(Table\_name, {1,2,3}).

first(Table\_name)

from\_ets(DETS\_Tab, ETS\_Tab)

info(DETS\_Tab)

تابع بالا، اطلاعات یک جدول را برمی گرداند. جزئیات بیشتر را در راهنما پیدا کنید.

insert(DETS\_Tab, Objects)

insert\_new(Name, Objects)

is\_dets\_file(DETS\_Tab)

lookup(DETS\_Tab, Key)

ets:match(Table\_name, Pattern)

match(Table\_name, Pattern, Limit\_number) **->** {[Match],Continuation}

ets:match(Continuation)

1. -module(match1).
2. -compile([export\_all]).
3. %-------------------------------------------
4. start() **->**
5. ets:new(my\_table, [duplicate\_bag, named\_table, public]),
6. ets:insert(my\_table, {1, a,a1}),
7. ets:insert(my\_table, {2, b,b2}),
8. ets:insert(my\_table, {3, c,c31}),
9. ets:insert(my\_table, {3, c,c31}),
10. ets:insert(my\_table, {3, c,c32}),
11. ok.
12. %-------------------------------------------
13. match\_2\_1()**->**
14. M1=ets:match(my\_table, '$1'),
15. io:format("Matched : **~p~n**", [M1]).
16. %-------------------------------------------
17. match\_2\_2()**->**
18. [[M1,M2]]=ets:match(my\_table, {'$20',a,'$2'}),
19. io:format("Matched : **~p**==**~p~n**", [M1,M2]).
20. %-------------------------------------------
21. match\_2\_3()**->**
22. M3=ets:match(my\_table, {'$3',c,'$4'}),
23. io:format("Matched : **~p~n**", [M3]).
24. %-------------------------------------------
25. match\_3\_1()**->**
26. {M4,C}=ets:match(my\_table, {'$5',c,'$6'},2),
27. io:format("Matched : **~p~n**", [M4]),
28. io:format("C : **~p~n**", [C]),
29. M5=ets:match(C),
30. io:format("Matched34 : **~p~n**", [M5]).
31. %-------------------------------------------

1. match\_2\_1()**->**
2. M1=ets:match(my\_table, '$1'),
3. io:format("Matched : **~p~n**", [M1]).

1> match1:start().

ok

2> match1:match\_2\_1().

Matched : [[{1,a,a1}],[{2,b,b2}],[{3,c,c31}],[{3,c,c31}],[{3,c,c32}]]

ok

1. match\_2\_2()**->**
2. [[M1,M2]]=ets:match(my\_table, {'$20',a,'$2'}),
3. io:format("Matched : **~p**==**~p~n**", [M1,M2]).

3> match1:match\_2\_2().

Matched : a1==1

ok

1. match\_2\_3()**->**
2. M3=ets:match(my\_table, {'$3',c,'$4'}),
3. io:format("Matched : **~p~n**", [M3]).

اجازه دهید آن را در پوسته امتحان کنیم :

4> match1:match\_2\_3().

Matched : [[3,c31],[3,c31],[3,c32]]

ok

1. match\_3\_1()**->**
2. {M4,C}=ets:match(my\_table, {'$5',c,'$6'},2),
3. io:format("Matched : **~p~n**", [M4]),
4. io:format("C : **~p~n**", [C]),
5. M5=ets:match(C),
6. io:format("Matched : **~p~n**", [M5]).

5> match1:match\_3\_1().

1. Matched : [[3,c31],[3,c32]]
2. C : {#Ref<0.1930490589.241303553.82359>,23,2,
3. #Ref<0.1930490589.241303553.82393>,
4. [[3,c31]],
5. 1}
6. Matched : {[[3,c31]],'$end\_of\_table'}

match\_delete(Table, Pattern)

match\_object(Table, Pattern)

match\_object2()**->**

M5=ets:match\_object(my\_table, {'$7',c,'$8'}),

io:format("Matched : **~p~n**", [M5]).

6> match1:match\_object2().

Matched : [{3,c,c31},{3,c,c31},{3,c,c32}]

ok

match\_object(Table, Pattern, Limit)

match\_object(Continuation)

# فصل 15: پایگاه داده mnesia :

# 15-1 نقاط ضعف و قوت پایگاه داده mnesia:

# 15-1-1 نقاط ضعف:

# 15-1-2 نقاط قوت:

# 15-2 توابع معمول Mnesia:

**create\_schema/1**

(node1@host1)1> mnesia:create\_schema([node()]).

ok

(node2@host1)1> net\_kernel:connect\_node(node1@host1).

true

(node2@host1)2> mnesia:create\_schema([node()|nodes()]).

ok

تعیین آدرس قرار گیری پایگاه داده

مثال:

(node1@host1)1> application:set\_env(mnesia, dir, "e:/mnesia\_dir2").

ok

(node1@host1)2> mnesia:create\_schema([node()]).

ok

یادآوری: node() گره جاری را برمی گرداند و nodes() نام گره های متصل را برمی گرداند.

2> cd("e:").

e:/

ok

3> mnesia:create\_schema([node()]).

ok

مثال:

C:\Users\arman> cd e:

E:\> erl -sname node1 -mnesia dir "'mnesia\_dir'"

Erlang/OTP 26 [erts-14.2.3] [source] [64-bit] [smp:6:6] [ds:6:6:10] [async-threads:1] [jit:ns]

Eshell V14.2.3 (press Ctrl+G to abort, type help(). for help)

(node1@host1)1> mnesia:create\_schema([node()]).

ok

مثالی که در مستندات mnesia (mnesia-4.13.2.pdf) آورده شده را در ادامه می بینیم:

erl -mnesia dir '"/ldisc/scratch/Mnesia.Company"'

start/0

create\_table/2

mnesia:create\_table(Table\_name,[L]).

{access\_mode, Atom}

{disc\_copies, Nodelist}

{disc\_only\_copies, Nodelist}

{ram\_copies, Nodelist}

{attributes, record\_info(fields, Record\_Name)}

  mnesia:create\_table(user\_id,[{attributes,[name,age,job]}]),

    mnesia:create\_table(user\_address, [{attributes, record\_info(fields, user\_address)}]).

 {type, Type}

table/1

مثال:

 do(qlc:q([X || X <- mnesia:table(table\_1)])).

stop/0

application:stop(mnesia).

wait\_for\_tables/2

mnesia:wait\_for\_tables([table\_1, table\_2,…], TimeOut).

Write/1

mnesia:write(Record)

mnesia:write(Table, Record, LockKind)

1. -record(table1,   {name, email=[], address, phone\_number}).
2. mnesia:create\_table(table1, [{attributes,record\_info(fields, table1)}]).
3. %--------------------------------------------
4. f\_table1() ->
5. [
6. {table1, ali, "ali@erlang\_email.com", ahvaz, 090000001},
7. …
8. ].
9. %----------------- روش اول : ورود تعداد زیادی رکورد به صورت یکجا به جدول
10. F = fun() ->
11. lists:foreach(fun mnesia:write/1, f\_table1()),
12. lists:foreach(fun mnesia:write/1, f\_table2()),
13. end,
14. mnesia:transaction(F).
15. %-----------------------------روش دوم: ورود یک رکورد به جدول
16. F1 = fun() ->
17. T= # table1{name=zz1, email="zz1@a.com", address=zzcity, phone\_number=99},
18. mnesia:write(T)
19. end,
20. mnesia:transaction(F1).

Delete/1

mnesia:delete({Tab, Key})

read/1

read/2

mnesia:read({Table, Key})

mnesia:read(Table, Key)

abort/1

mnesia:abort(xReason).

{aborted, xReason}

clear\_table/1

mnesia:clear\_table(table\_1)

match\_object /3

mnesia:match\_object(Table\_name, Pattern, LockKind).

مثال:

mnesia:match\_object(user\_id, {user\_id, '\_', '\_', baker}, read).

transaction/1

mnesia:transaction(Fun)

mnesia:transaction(Fun, Retries)

{atomic,Result}

{aborted,Reason}

finction1()->

F = fun() ->

mnesia:write(…),

mnesia:write(…)

.

.

.

end,

mnesia:transaction(F).

# 15-3 QLC (**Query List Comprehension**):

qlc:append(QH1, QH2)

A=qlc:cursor(QH).

qlc:delete\_cursor(QueryCursor)

F=fun()->

    Q=qlc:q([X || X <- mnesia:table()]),

    qlc:eval(Q) end,

    {atomic, Val} = mnesia:transaction(F),

    Val.

qlc:e(Q).

-include\_lib("stdlib/include/qlc.hrl").

1. -module(mnesia1).
2. -compile(export\_all).

1. -include\_lib("stdlib/include/qlc.hrl").

1. -record(user\_id,        {name, age=[], job}).
2. -record(user\_address,   {name, email=[], address, phone\_number}).

1. user\_id\_table() ->
2. [
3. {user\_id, ali,     50, programmer},
4. {user\_id, hasan,   60, baker},
5. {user\_id, reza,    70, baker},
6. {user\_id, arman,   80, programmer},
7. {user\_id, x,       15, student}
8. ].

1. user\_address\_table() ->
2. [
3. {user\_address, ali,    "ali@erlang\_email.com",     ahvaz,      090000001},
4. {user\_address, hasan,  "hasan@erlang\_email.com",   address1,   090000002},
5. {user\_address, reza,   "reza@erlang\_email.com",    address2,   090000003},
6. {user\_address, arman,  "arman@erlang\_email.com",   ahvaz,      090000004},
7. {user\_address, x,      "x@erlang\_email.com",       address3,   090000005}
8. ].

1. match()->
2. F = fun() ->
3. mnesia:match\_object(user\_id,

{user\_id, '\_', '\_', baker}, read)

1. *%io:format("hi=%d",[M])*
2. end,
3. mnesia:transaction(F,5).
4. create\_schema\_tables() ->
5. mnesia:create\_schema([node()]),
6. mnesia:start(),
7. mnesia:create\_table(user\_id,

[{attributes,record\_info(fields, user\_id)}]),

1. mnesia:create\_table(user\_address,

[{attributes, record\_info(fields, user\_address)}]),

1. mnesia:stop().
2. loading\_tables() ->
3. mnesia:start(),
4. mnesia:wait\_for\_tables([user\_id,user\_address], 5000).
5. writing\_in\_tables() ->
6. mnesia:clear\_table(user\_id),
7. mnesia:clear\_table(user\_address),
8. F = fun() ->
9. lists:foreach(fun mnesia:write/1, user\_id\_table()),
10. lists:foreach(fun mnesia:write/1, user\_address\_table()),
11. T= #user\_address{name=zz1, email="zz1@a.com", address=zzcity, phone\_number=99999999},
12. mnesia:write(T)
13. end,
14. mnesia:transaction(F).
15. view\_the\_entire\_table() ->
16. F=fun()->
17. Q=qlc:q([X || X <- mnesia:table(user\_address)]),
18. qlc:eval(Q) end,
19. {atomic, Val} = mnesia:transaction(F),
20. Val.
21. show\_age\_and\_job() ->
22. F = fun() ->
23. Q=qlc:q([{X#user\_id.age, X#user\_id.job} || X <- mnesia:table(user\_id)]),
24. qlc:e(Q) end,
25. {atomic, Val} = mnesia:transaction(F),
26. Val.
27. people\_over\_60\_years\_old() ->
28. F=fun()->
29. Q=qlc:q([X || X <- mnesia:table(user\_id),X#user\_id.age>60]),
30. qlc:eval(Q) end,
31. {atomic, Val} = mnesia:transaction(F),
32. Val.
33. delete\_object() ->
34. Oid = {user\_id,x},
35. F = fun() ->
36. mnesia:delete(Oid)
37. end,
38. mnesia:transaction(F).

1. () ->
2. F = fun() -> mnesia:read({user\_id,ali}) end,
3. mnesia:transaction(F).

PS C:\Users\arman> erl -sname node1

(node1@host1)1> mnesia1:create\_schema\_tables().

=INFO REPORT==== 30-Mar-2024::07:46:00.626000 ===

    application: mnesia

    exited: stopped

    type: temporary

stopped

(node1@host1)2> mnesia1:loading\_tables().

ok

(node1@host1)3> mnesia1:writing\_in\_tables().

{atomic,ok}

(node1@host1)4> mnesia1:view\_the\_entire\_table().

[{user\_address,zz1,"zz1@a.com",zzcity,99999999},

 {user\_address,arman,"arman@erlang\_email.com",ahvaz,90000004},

 {user\_address,hasan,"hasan@erlang\_email.com",address1,

               90000002},

 {user\_address,x,"x@erlang\_email.com",address3,90000005},

 {user\_address,ali,"ali@erlang\_email.com",ahvaz,90000001},

 {user\_address,reza,"reza@erlang\_email.com",address2,

               90000003}]

            mnesia1:show\_age\_and\_job().

        (node1@host1)5> mnesia1:show\_age\_and\_job().

[{60,baker},

 {80,programmer},

 {15,student},

 {50,programmer},

 {70,baker}]

 (node1@host1)6> mnesia1:people\_over\_60\_years\_old().

[{user\_id,arman,80,programmer},{user\_id,reza,70,baker}]

(node1@host1)7> mnesia1:delete\_object().

{atomic,ok}

(node1@host1)8> mnesia1:reading\_from\_the\_table().

{atomic,[{user\_id,ali,50,programmer}]}