

بنام خداوند جان و خرد

برنامه نویسی Erlang

مدیریت فرایندهای توزیع شده 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.
موارد گنجانده شده در این کتاب: Error! Bookmark not defined.
موضوعاتی که در این کتاب آورده نشده است: Error! Bookmark not defined.
در ادامه مواردی را که می تواند، موضوع کتاب های دیگری شود، بیان شده است: Error! Bookmark not defined.

فصل 1 : معرفی ارلنگ	10
1- آشنایی با ارلنگ	10
تاریخچه ارلنگ :	10
1-1 فرایندها در ارلنگ :	10
1-2 مزایای استفاده از ارلنگ :	10
1-2-1 فرایندهای مستقل از سیستم عامل:	11
1-2-2 پردازش موازی (parallel processing):	11
1-2-3 همزمانی سبک وزن (concurrency):	11
1-2-4 داده تغییر ناپذیر (immutable data):	11
1-2-5 مزایای متغیر تغییر ناپذیر:	11
1-3 تطبیق الگو (pattern matching):	11
1-4 مقاومت در برابر خطا (fault-tolerant):	11
1-5 مقیاس پذیری (Scalability) :	12
1-6 توزیع شده (distributed):	12
1-7 تعویض داغ (Hot Swapping) :	12
1-8 بدون توقف :	12
1-9 اجتناب از حافظه اشتراکی :	12
1-10 ارلنگ برای چه کارهای مناسب است :	12
1-11 نصب پورته ارلنگی در ویندوز	12
14 1-11-2 نصب ارلنگ در سیستم عامل های Fedora و Ubuntu/Debian – mac os – Free BSD	
1-11-3 نصب پلاگین ارلنگ در eclipse در ویندوز	14
1-12 تعدادی از فرمان های پورته:	20
فصل 2 : مفاهیم اساسی	22

22	2-2 اعداد صحیح:
23	2-3 اعداد شناور و عملگر های ریاض
24	2-4 تاپل ها tuples
26	2-5 اتم ها :
26	2-6 رشته ها:
27	2-7 عملگر های مقایسه ای
28	2-8 متغیر
29	2-9 ساختار های داده پیچیده
30	2-10 تطبیق الگو
30	2-11 متغیر بی اهمیت (یا اهمیت نده) don't care
31	2-12 توابع
31	2-13 بولین
33	فصل 3 : ماژول ها و توابع
33	3-1 ماژول ها
34	3-2 نام های رزرو شده
34	3-3 fun ها
36	3-4 توابع
36	3-5 ویژگی های ماژولی (module attribute)
36	3-6 اعلان import() - چیست ؟
38	3-7 اعلان export
38	3-8 اعلان module
38	3-9 اعلان compile
41	3-10 ویژگی های تعریف شده توسط کاربر:
42	فصل 4 : ماکرو ها ، map ها ، رکورد ها و فایل های include
42	4-1 macro
43	4-1-1 ماکرو های از پیش تعریف شده ارلنگ :
44	4-1-2 جریان کنترلی در ماکرو ها:
46	4-1-3 ماکرو ها و گارد ها:
46	4-2 فایل های include:
46	فرق بین include و include_lib -

47	map 4-3
48	map های BIF 4-3-1
50	Record ها: 4-4
50	4-4-1 چرا باید از رکورد ها استفاده کنیم؟
51	4-4-2 رکورد ها در عمل:
56	فصل 5 : لیست و مفاهیم لیستی:
57	5-1 در ادامه توابع کاربردی تر ماژول list را معرفی می کنیم:
57	5-1-1 تابع lists:all
58	5-1-2 تابع lists:any
58	5-1-3 تابع lists:delete
58	5-1-4 تابع lists:droplast
58	5-1-5 تابع lists:duplicate
58	5-1-6 تابع lists:last
59	5-1-7 تابع lists:max
59	5-1-8 تابع lists:member
59	5-1-9 تابع lists:min
59	5-1-10 تابع lists:merge
59	5-1-11 تابع lists:sort
60	5-1-12 تابع lists:reverse
60	5-1-13 تابع lists:sum
60	5-1-14 تابع lists:split
60	5-1-15 تابع lists:map
60	5-1-16 lists:filter(Pred, List)
61	5-1-17 lists:foreach(F, List)
61	5-2 ترفند:
62	5-3 عملگر های ++ و --
62	5-4 مفاهیم لیستی: (List Comprehensions)
68	فصل 6: باینری ها:
68	6-1 باینری ها
68	6-2 BIF های ماژول erlang برای باینری ها:

69.....	binary_to_atom(Binary)
69.....	binary_to_integer(Binary).
69.....	binary_to_list(Binary).
70.....	bitstring_to_list(Bitstring).
70.....	bit_size(Bitstring).
70.....	byte_size(Bitstring).
70.....	split_binary(Bin, Pos)
71.....	term_to_binary(Term).
71.....	6-3 توابع مازول binary:
71.....	binary:at/2
71.....	binary:compile_pattern/1
71.....	binary:split/2
71.....	:matches/2
71.....	:match/2
72.....	binary:copy/2
72.....	binary:first/1
72.....	binary:last/1
72.....	binary:part/3
72.....	6-4 گاردها برای باینری،
73.....	is_binary/1
73.....	is_bitstring/1
73.....	6-5 نحو بیتی:
	74 Size
74.....	TypeSpecifierList 6-6
	74 Type
75.....	Signedness
75.....	Endianness
	75 unit
76.....	6-7 مفاهیم باینری و عملگرهای بیتی:
76.....	6-8 عملگرهای بیتی:

77	6-9 استخراج بیتی:
78	فصل 7: ساختارهای if ، case ، guard و for
78	7-1 ساختار Case:
81	7-2 ساختار for
82	7-3 ساختار guard ها
84	7-4 If
86	فصل 8: کار با فایل ها
86	8-1 مازول file
90	8-2 مازول io
92	8-3 تا اینجا یاد گرفتیم:
92	فصل 9: ارتباط ارلنگ با زبان c
92	9-1 مکانیزم های که ارلنگ برای ارتباط با دیگر زبان ها دارد:
92	9-1-1 ارلنگ توزیع شده
92	9-1-2 پورت ها
93	9-2 کتابخانه C
93	9-3 C Node
93	9-4 Linked-in Drivers
93	9-5 NIF (Native Implemented Functions)
93	9-6 ارتباط یک برنامه خارجی به زبان C با یک برنامه به زبان Erlang با استفاده از پورت ..
93	9-7 وظایف کد cmain:
96	9-8 کد کامل cmain.c:
98	9-9 وظایف کد cfunc1 :
98	9-10 کار با کامپایلر برنامه های C
98	9-11 مازول ارلنگی erlang_prog:
98	open_port
99	9-12 لیست کردن پورت ها
99	9-13 بستن پورت
99	9-14 ارسال پیام به پورت
99	9-15 تغییر مالک پورت
100	9-16 توضیح قسمت ارلنگی پورت:

فصل 10: مدیریت خطا در برنامه های ترتیبی.....	103
10-1 معرفی انواع خطا :	103
10-2 انواع کلاس های خطا	103
10-3 مدیریت خطا (Handling Errors)	105
گرفتن خطا با try...catch	105
10-5 Catch روش دیگر به دام انداختن استثنا ها	107
10-3 انواع استثنائها:	107
10-6 چه زمانی از (exit/1 , error/1 , throw/1) استفاده می کنیم؟	114
فصل 11: برنامه های همزمان	115
11-1 مفاهیم ضروری همزمانی:	115
11-2 فرایند و تولید مثل فرایندی :	116
11-3 ارسال و دریافت پیام بین فرایند ها:	117
11-4 مهلت ها :	120
فصل 12: فرایند های توزیع شده	122
12-1 ساخت گره:	122
12-2 تولید فرایند راه دور:	127
12-3 BIF های کوکی:	131
12-4 ثبت فرایند ها در گره های دور:	131
فصل 13: مدیریت خطا در فرایند های همزمان	135
13-1 لینک کردن فرایند ها:	135
13-2 مانیتور کردن یک فرایند:	136
13-3 ثبت فرایند ها:	140
13-4 فرایند های سیستمی و به تله انداختن یک استثنا :	141
13-5 دریافت انتخابی با مرجع ها:	142
فصل 14: جدول ذخیره ترم های ارلنگی: ETS و DETS	146
14-1 انواع جدول ها:	146
14-1-1 نکته ها:	146
14-1-2 استثنا ها :	146
14-1-3 توابع ماژول ETS:	146
14-14 DETS:	152

152	14-2-1 معرفى ماژول DETS :
152	14-2-2 توابع ماژول DETS :
156	فصل 15: پایگاه داده mnesia :
156	15-1 نقاط ضعف و قوت پایگاه داده mnesia :
156	15-1-1 نقاط ضعف :
156	15-1-2 نقاط قوت :
156	15-2 توابع معمول Mnesia :
160	15-3 QLC (Query List Comprehension) :

فصل 1 : معرفی ارننگ

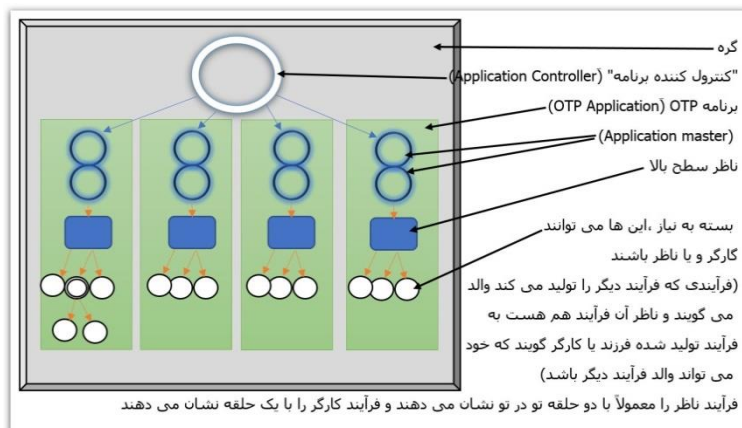
1- آشنایی با ارننگ

نماد	نام انگلیسی	نام فارسی
	vertical bar	میله عمودی
	double vertical bar	دو میله
~	tilde	علامت مَدک 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 ارننگ :

1-1 فرایند ها در ارننگ :



تصویر 1.1: فرایند ها در یک گره

1-2 مزایای استفاده از ارننگ :

1-2-1 فرایند های مستقل از سیستم عامل:**1-2-2** پردازش 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}

```

1-2-5 مزایای 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 می‌شود و همه جا اجرا می‌شود، در ارلنگ یک بار نوشته می‌شود و همیشه اجرا می‌شود.»

ارلنگ 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 :

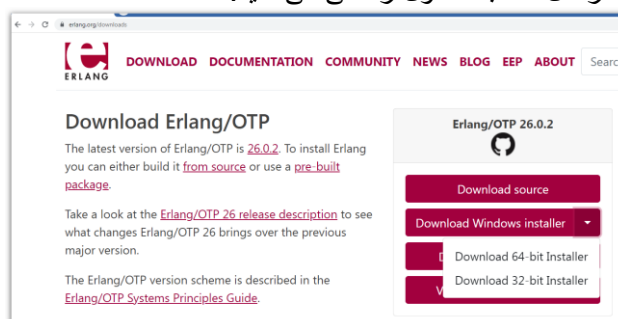
1-10 ارلنگ برای چه کارهای مناسب است :

:OTP

1-11 نصب پوسته ارلنگی در ویندوز

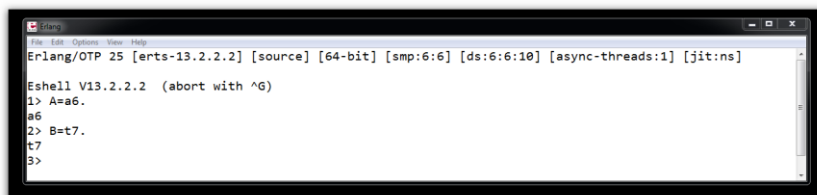
من از پوسته ارلنگ (Eshell) استفاده می کنم. برای نصب آن در ویندوز مراحل زیر را دنبال می کنید.

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



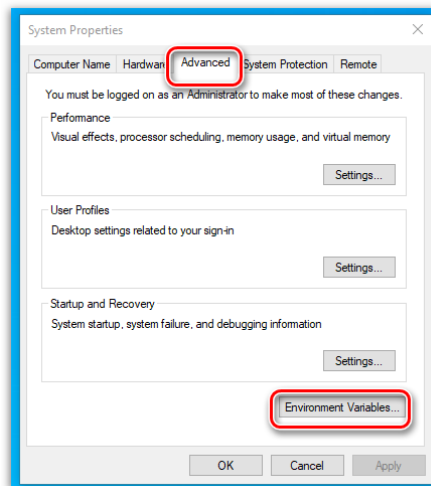
تصویر 1.2 : سایت erlang.org

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

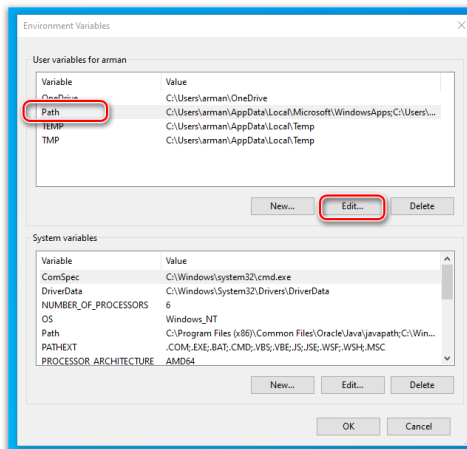


تصویر 1.3 : پوسته ارلنگ در ویندوز

در اینجا >1 و مانند آن خط فرمان ارلنگ 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 کلیک کنید environment variables... را کلیک کنید:

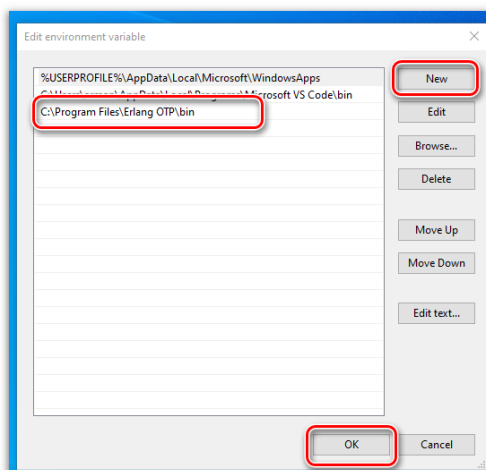


تصویر 1.4: پنجره تنظیمات سیستمی ویندوز
سپس پنجره environment variables باز می شود و باید گزینه path را انتخاب کنید و سپس کلیک edit... را کلیک کنید :



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 Free BSD – mac os – Fedora و Ubuntu/Debian

Mac OS X:

```
sudo port install erlang
```

Linux:

Ubuntu/Debian:

```
sudo apt-get update
sudo apt-get install erlang
```

Fedora:

```
sudo yum install erlang
```

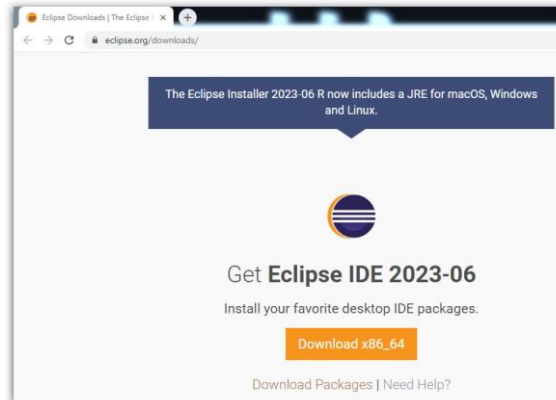
FreeBSD:

```
sudo pkg update
sudo pkg install erlang
```

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

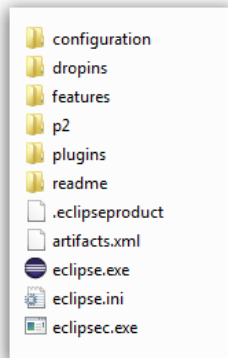
برای نصب پلاگین ارلنگ در eclipse در ویندوز مراحل زیر را دنبال کنید:

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



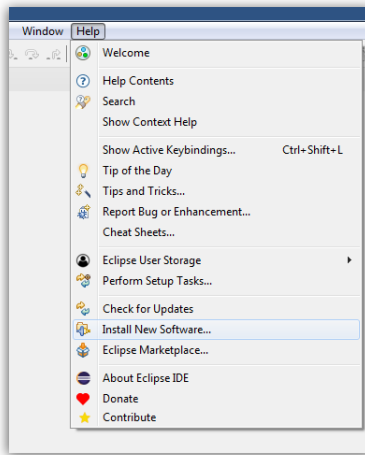
تصویر 1.7 : سایت www.eclipse.org

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:



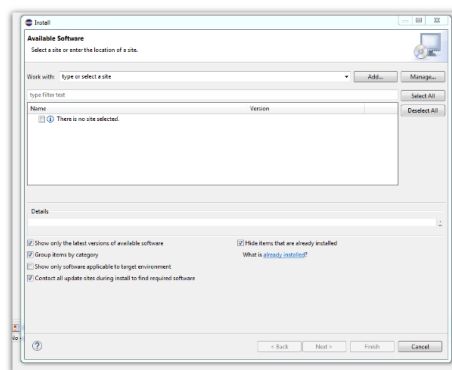
تصویر 1.8 : دایرکتوری eclipse

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



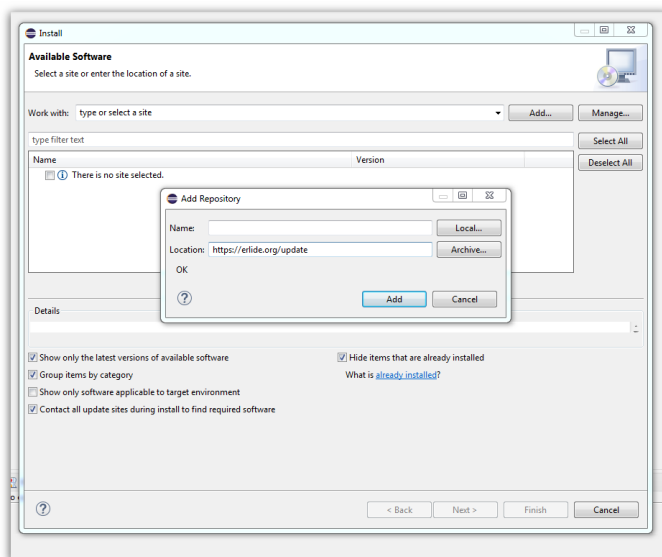
تصویر 1.9 : منوی help

4: پنجره زیر باز می شود



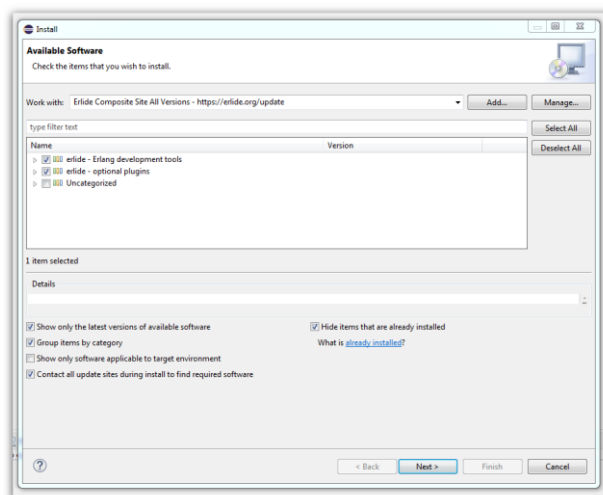
تصویر 1.10 : پنجره Install New Software در منوی help

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.



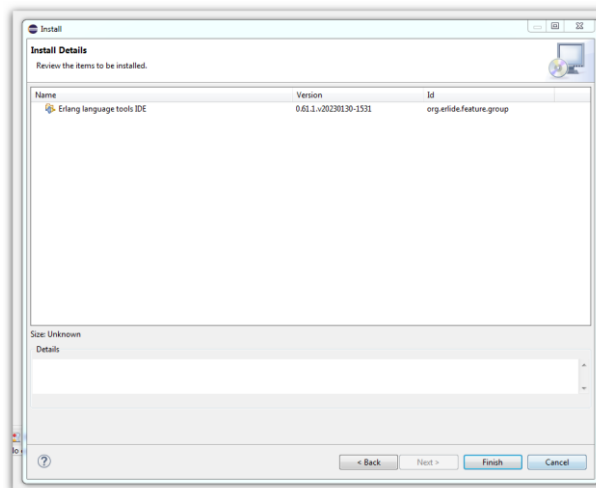
تصویر 1.11 : پنجره add repository

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



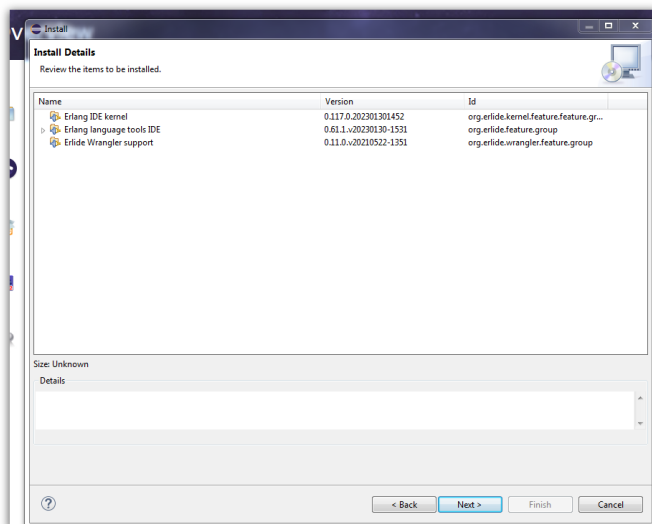
تصویر 1.12 : پنجره Install کلیک next

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



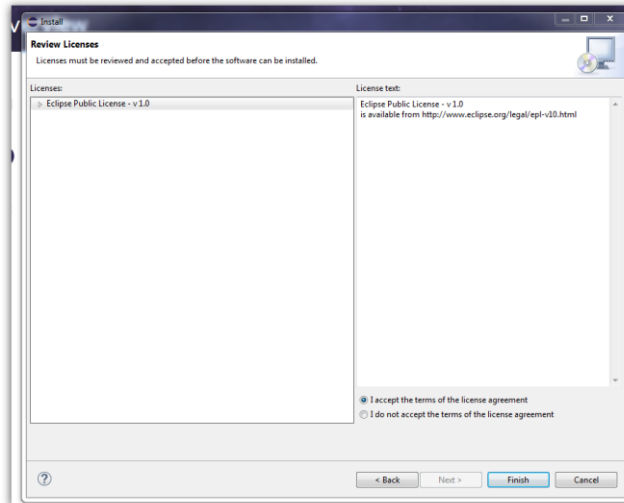
تصویر 1.13 : پنجره پنجره Install کلیک finish

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



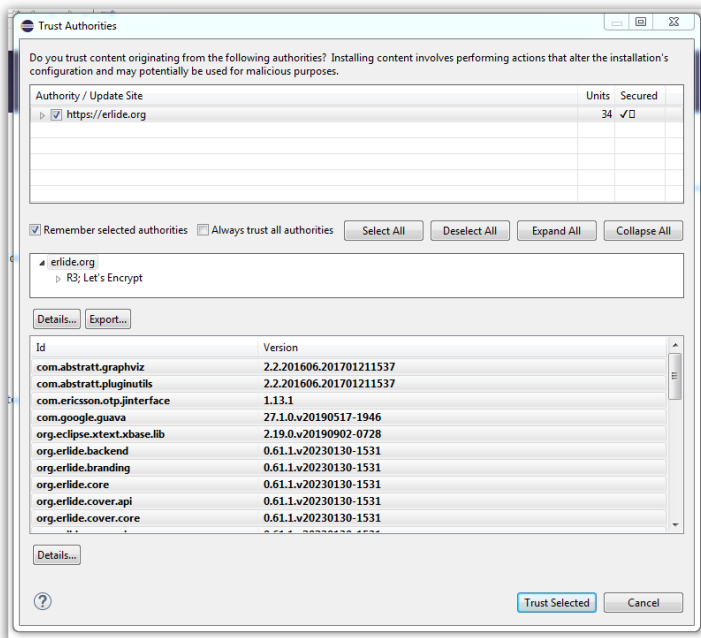
تصویر 1.14 : پنجره Install Details

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



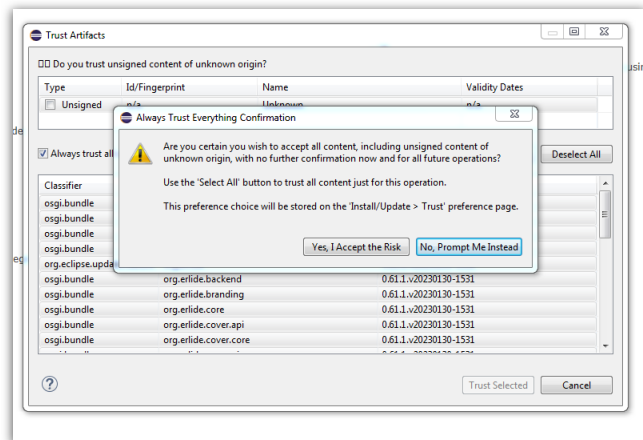
تصویر 1.15 : پنجره Review Licenses

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

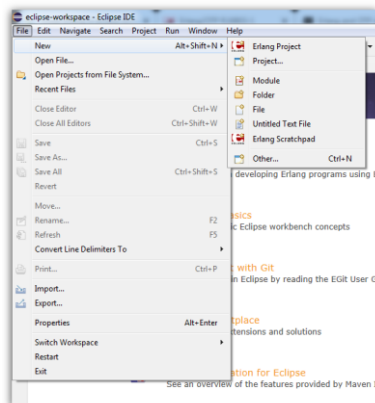


تصویر 1.16 : پنجره Trust authorities

11: در پنجره زیر ... , Yes را بزنید.



تصویر 1.17 : پنجره Always Trust Everything Confirmation
 12: یک بار برنامه را ببندی و دوباره باز کنی . 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
1.	q().	یک خروج کنترل شده را به همراه دارد.

2.	erlang:halt()	توقف بلادرنگ سیستم .
3.	ctrl+A	به ابتدای خط می پرد
4.	Ctrl+E	به انتهای خط می پرد
5.	Ctrl+D	کاراکتر جاری را حذف می کند
6.	Ctrl+F	به کاراکتر جلوی می رود (با کلید های جهتی هم می شود)
7.	Ctrl+B	به کاراکتر قبلی می رود (با کلید های جهتی هم می شود)

جدول 1.2 : فهرست فرمان های پوسته ای

q(). این فرم 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 مازول دیدید تایپ کنید. زیرا تمام کد وارد شده در مازول عبارات محاسباتی نیستند بنابراین در پوسته ارلنگی درک نمی شوند.
مثال:

```
3> -export([sum/2]).
** exception error: an error occurred when evaluating an arithmetic expression
   in operator  '/'/2
   called as sum / 2
```

تصویر 1.19 : خطا در پوسته ارلنگی

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

فصل 2 : مفاهيم اساسی

BIF 2-1

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 اضافه و سپس منفی می کند)	bnot X
2	عدد صحیح	تقسیم صحیح	$X \text{ div } Y$
2	عدد صحیح	باقیمانده تقسیم	$X \text{ rem } Y$
2	عدد صحیح	عملگر بیتی and (اگر هر دو 1 بود جواب 1)	$X \text{ band } Y$
3	عدد	$X + Y$	$X + Y$
3	عدد	$X - Y$	$X - Y$
3	عدد صحیح	عملگر بیتی or (اگر ورودی 0 یا 1 باشد ؛ اگر حداقل یکی 1 باشد جواب 1 می شود در غیر اینصورت جواب 0 می شود)	$X \text{ bor } Y$
3	عدد صحیح	عملگر بیتی xor اگر ورودی ها متفاوت باشد جواب 1 می شود در غیر اینصورت 0 می شود	$X \text{ bxor } Y$
3	عدد صحیح	جایگاه بیتی X را N بیت به چپ می برد مثال در ادامه آمده است	$X \text{ bsl } N$
3	عدد صحیح	برعکس $X \text{ bsl } N$ است	$X \text{ bsr } N$

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
```

```
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
```

```
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
```

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 متغير

مثال:

A يا Abc يا 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 تطبيق 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

```
12.-module(calculator).
13.-export([sum/1, mult/1, sub/1, div1/1]).
14.
15.
16.sum      ({sum , A,B})      -> A+B.
17.mult     ({mult , A,B})     -> A*B.
18.sub      ({sub , A,B})      -> A-B.
19.
20.div1     ({div_d, A,B})     -> A/B;
21.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

```

```

12. -module(calculator2).
13. -export([sum/3, mult/1, sub/1, div1/1]).
14.
15.
16. sum    (sum2 , A,B)      -> A+B.
17. mult   ({mult , A,B})    -> A*B.
18. sub    ({sub , A,B})     -> A-B.

```

```

1> calculator2:sum(sum2,5,2).
7

```

```

20. div1 ({div_d, A,B})      -> A/B;
21. 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

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

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.

```

5. test1() -> map(fun(X) -> X + 1 end, [1,2,3,4,5]).
6.
7. test2() -> map(fun sum/1, [1,2,3,4,5]).
8.
9. 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.
6. start() ->
7. io:fwrite("data~p~n",[date()]),
8. fwrite("data~p~n",[date()]).
9.
10. calculator({sum,A,B}) -> sum({sum,A,B});
11.
12. calculator({mult,A,B})-> multx:mult({mult,A,B});
13. calculator({sub,A,B}) -> subx:sub({sub,A,B});
14.
15. calculator({div_d,A,B})-> divx:div1({div_d,A,B});

```

```
16. 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 بعداً 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 نویسه های دیگری غیر از حروف عددی، زیرخط (_) یا @ باشد، اتم باید در گیومه های تکی (') قرار گیرد.

مثال:

```
-attribute_prog({div1,sum1}).
```

فرخوانی 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

macro 4-1

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` و `-include_lib`

```
-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"
```

:map 4-3

```
#{Key1=>value1, key2=>value2, ....}.
```

الگوهای آن مانند مورد زیر است:

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

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

```
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
```

BIF 4-3-1 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 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]
```

Record 4-4 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, {'I=II'}).
-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.
4. -record(first_record, {a=aa1,b=11,c}).
5.
6. %------
7. new_record_1() ->

```

```

8. #first_record{a="aa2", b=22,c=cc2}.
9.
10. edit_record_1(#first_record{a=A,b=B} = New_record_2) when
    A=="aa2" ->
11. New_record_2#first_record{b=B+6,c=cc3}.
12.
13. edit_record_2(#first_record{a=A,b=B}) when A=="aa2" ->
14. [B+10,A].
15.
16. %-----
17. print_r(#first_record{a=A,b=B,c=C}) ->
18. io:format("a:~p b:~p c:~p ~n", [A,B,C]).
19.
20. print_e([B,A]) ->
21. io:format("b:~p a:~p ~n", [B,A]).
22.
23. %-----
24. test_a() ->
25. print_r(#first_record{a="aa22", b=22,c=cc2}).
26. test_b() ->
27. print_r(edit_record_1(new_record_1())).
28. test_c() ->
29. print_e(edit_record_2(new_record_1())).
30.

```

```

7. new_record_1() ->
8. #first_record{a="aa2", b=22,c=cc2}.

```

```

10. edit_record_1(#first_record{a=A,b=B} = New_record_2) when
    A=="aa2" ->
11. New_record_2#first_record{b=B+6,c=cc3}.

```

```

26. test_b() ->
27. print_r(edit_record_1(new_record_1())).

```

```
13.edit_record_2(#first_record{a=A,b=B}) when A=="aa2" ->
14.[B+10,A].
```

```
20.print_e([B,A]) ->
21.io:format("b:~p a:~p ~n", [B,A]).
```

```
28.test_c() ->
29.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 ماثول **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
```

lists:any تابع 5-1-2

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
```

lists:delete تابع 5-1-3

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]
```

lists:droplast تابع 5-1-4

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]
```

lists:duplicate تابع 5-1-5

مثال:

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

lists:last تابع 5-1-6

مثال:

```
1> List_1=[1,2,3,4,5,6,7,8,9,10],
1> L2=lists:last(List_1).
```

10

lists:max تابع 5-1-7

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
```

lists:member تابع 5-1-8

```
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
```

lists:min تابع 5-1-9

```
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
```

lists:merge تابع 5-1-10

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"]
```

lists:sort تابع 5-1-11

```
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]
```

lists:reverse تابع 5-1-12

```
1> L=[1,2,3,4,5].  
[1,2,3,4,5]  
2> lists:reverse(L).  
[5,4,3,2,1]
```

lists:sum تابع 5-1-13

```
1> L=[1,2,3,4,5].  
[1,2,3,4,5]  
2> lists:sum(L).  
15
```

lists:split تابع 5-1-14

```
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]}
```

lists:map تابع 5-1-15

```
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 عملگرهای ++ 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: '.'
```

فصل 6: 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>>
```

BIF 6-2 های مازول 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 E_i 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
```

TypeSpecifierList 6-6

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>>.
<<64,74,128,0,0,0,0,64,69,128,0,0,0,0,64,74,128,0,0,0,
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,-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 `for` و `if` ، `case` ، `guard`

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,map
1/2]).
3.
4. dinner (Day) ->
5. case Day of
6.   saturday      -> {'sandwich'};
7.   sunday        ->{'Pizza'};
8.   monday        ->{'Gheymeh stew'};
9.   tuesday       ->{'Ghormeh Sabzi stew'};
10.   wednesday    ->{'Soltani Kebab'};
11.   thursday     ->{'Bakhtiari kebab'};
12.   friday       ->{'kubideh kebab'};
13.   Otherwise->
   io:fwrite("no match:~w~n",[Otherwise])
14.   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.
2. number3 (N) ->
3. case N of
4. {A,B}when A==1,B==1->
   io:fwrite('a=>~w~n',[{A,B+1}]);
5. {A,B}when A==1,B==2-> io:fwrite('b=>~w~n',[N]);
6. {A,B}when A==2,B==1-> io:fwrite('c=>~w~n',[N]);
7. {A,B}when A==2,B==2-> io:fwrite('d=>~w~n',[N]);
8. {A,B}when A==3,B==1-> io:fwrite('e=>~w~n',[N]);
9. {A,B}when A==3,B==2-> io:fwrite('f=>~w~n',[N]);
10. {A,B}when A==3,B>2-
   > io:fwrite('f=>~w~n',[N]);
11. Otherwise->
   io:fwrite("no match:~w~n",[Otherwise])
12. 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}
```



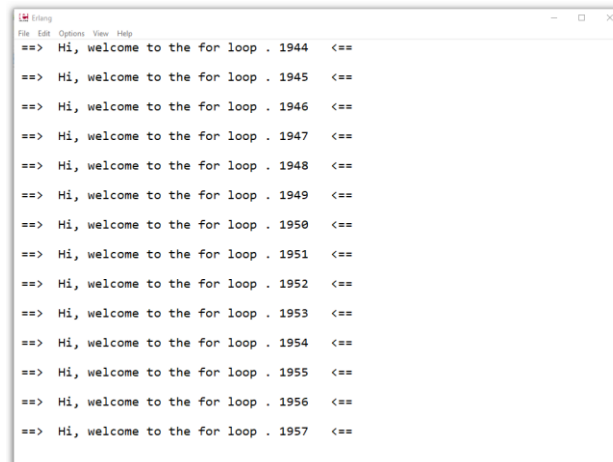
```
ok
6> case_1:number3({2,1}).
c=>{2,1}
ok
```

7-2 ساختار for

```
1. -module(for1).
2. -export([for/2]).
3.
4. for(I, J) when is_integer(I),erlang:is_integer(J),
   I > J ->
5. io:fwrite("~n <== End of the for loop. ==> ~n");
6.
7. for(I, J) when is_integer(I),erlang:is_integer(J)
   , I <= J ->
8. io:fwrite("~n ==> Hi, welcome to the for loop .
   ~w <== ~n",[I]),
9.
10.      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).
```



```
File Edit Options View Help
==> Hi, welcome to the for loop . 1944 <==
==> Hi, welcome to the for loop . 1945 <==
==> Hi, welcome to the for loop . 1946 <==
==> Hi, welcome to the for loop . 1947 <==
==> Hi, welcome to the for loop . 1948 <==
==> Hi, welcome to the for loop . 1949 <==
==> Hi, welcome to the for loop . 1950 <==
==> Hi, welcome to the for loop . 1951 <==
==> Hi, welcome to the for loop . 1952 <==
==> Hi, welcome to the for loop . 1953 <==
==> Hi, welcome to the for loop . 1954 <==
==> Hi, welcome to the for loop . 1955 <==
==> Hi, welcome to the for loop . 1956 <==
==> Hi, welcome to the for loop . 1957 <==
```

تصویر 1.7 : اجرای ناخواسته و نامحدود یک تابع

7-3 ساختار guard ها

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 قدیم	نام جدید
atom(X)	is_atom(X)
binary(X)	is_binary(X)
constant(X)	is_constant(X)
float(X)	is_float(X)
integer(X)	is_integer(X)
tuple(X)	is_tuple(X)

نام قدیم	نام جدید
list(X)	is_list(X)
number(X)	is_number(X)
pid(X)	is_pid(X)
port(X)	is_port(X)
reference(X)	is_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)
```

If 7-4

```

if
Guard1 ->
Body1;
Guard2->
Body2;
true-> TrueBody
end.

```

```

1. -module(if_a).
2. -export([if_1/1]).
3.
4. if_1(X)->
5. if
6. X==5          -> io:fwrite('~w~n',[X*2]);
7. is_float(X)   -> io:fwrite('~w is float~n',[X]);
8. X > 5 andalso X < 7 -> six;
9. %sum(X,2)==4   -> {'four'}
10.      true     -> {'no match'}
11.      end .
12.
13.      %sum(A,B)->
14.      %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 مآزول 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

```

io ماژول 8-2

```
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]]

2> io:fwrite("~w~n", [A]).

[[[{a123},{b123},{c123}],[{aa123},{bb123},{cc123}]],{aaa},
 [bbb]]
ok

3> io:fwrite("~p~n", [A]).

[[[{a123},{b123},{c123}],[{aa123},{bb123},{cc123}]],{aaa},
 [bbb]]
ok

4> io:fwrite("~50p~n", [A]).

[[[{a123},{b123},{c123}],
 [{aa123},{bb123},{cc123}]],
 {aaa},
 [bbb]]
ok
```

```
io:read (Prompt, Format)
io:read(IoDevice, Prompt, Format)
```

مثال:

```
1>io:read("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>}

5> io:fread(T2, " ", "~s").

{ok,["number_1={2024}"]}
6> file:close(T2).
ok

```

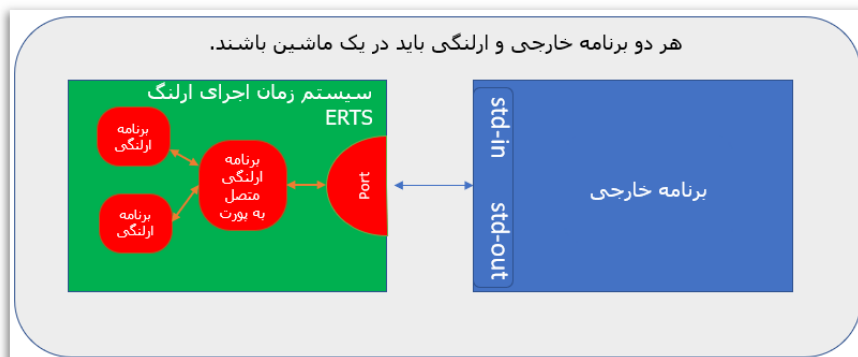
8-3 تا اینجا یاد گرفتیم:

فصل 9: 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 پورتهی

9-2 کتابخانه C

```
binary_to_term(Binary)
term_to_binary(Erlang_Term)
```

C Node 9-3

Linked-in Drivers 9-4

(Native Implemented Functions) NIF 9-5

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

```
25.     /*read(read_cmd)-----*/
26.     int read_cmd(byte *buf) /*byte
    buf[100](line:7)///called in line:8*/
27.     {
28.         int len;
29.         if (read_exact(buf, 2) != 2)
30.             return (-1);
31.         len = (buf[0] << 8) | buf[1];
32.         return read_exact(buf, len);
33.     }
34.     /*read(read_exact)-----*/
35.     int read_exact(byte *buf, int len)
36.     {
37.         int I, got = 0;
38.         do
39.         {
40.             if ((I = read(0, buf + got, len - got)) <= 0)
41.             {
42.                 return (i);
43.             }
44.             got += I;
45.         } while (got < len);
46.         return (len);
47.     }
```

```
if ((i = read(0, buf, len)) <= 0)
{
    return (i);
}
```

cmain.c

```

49.  /*write (write_cmd)----- */
50.
51.  int write_cmd(byte *buf, int len)
52.  {
53.  byte li;
54.  li = (len >> 8) & 0xff;
55.  write_exact(&li, 1);
56.  li = len & 0xff;
57.  write_exact(&li, 1);
58.  return write_exact(buf, len);
59.  }
60.  /*write (write_exact)----- */
61.  int write_exact(byte *buf, int len)
62.  {
63.  int i, wrote = 0;
64.  do
65.  {
66.  if ((i = write(1, buf + wrote, len - wrote)) <= 0)
67.  return (i);
68.  wrote += i;
69.  } while (wrote < len);
70.  return (len);
71.  }

```

تابع write:

```

if ((i = write(1, buf, len)) <= 0)
    return (i);

```

```

li = len & 0xff;

```

: cmain.c كد كامل 9-8

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 مازول 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

```
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).
```

erlang_prog.erl

```
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.
```

erlang_prog.erl

```
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. %-----

```

:

erlang_prog.erl

```

32. %-----
33. encode({foo, X}) -> [1, X];
34. encode({bar, Y}) -> [2, Y].
35. decode([Int]) -> Int.
36. %-----

```

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},
12. [{packet, 2}]),
13. loop(Port).
14. %-----
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) , exit(Pid, Reason)


```
1> catch exit(reason).
{'EXIT',reason}
```

مثال برای کلاس :error

error1.erl

```
1. -module(error1).
2. -export([do/1]).
3.
4. do(I) when is_atom(I)->
5. 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-  
5- try_catch() ->  
6- [a2(I) || I <- [1,4,3]].  
7-  
8- a2(N) ->io:format("~n=> a2/1 the function ~n"),  
9- try f2(N) of  
10-   a -> io:format("{a} try_section ~n");  
11-   b -> io:format("{b} try_section ~n");  
12-   c -> io:format("{c} try_section ~n")  
13-   catch  
14-   Class:Exception ->  
    {catch_section,Class,Exception,N}  
15-   after  
16-   io:format("the after_section ~n")  
17-   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]
```

Catch 10-5 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.
5. try_clause1() ->
6. [a(I) || I <- [1,2,3,4]].
7.
8. a(N) ->
9. try f1(N) of
10.     a -> a1;
11.     b -> b2;
12.     c -> c3
13. catch
14.     Class:Exception -> {Class,Exception,N}
15. 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.
3. A={ali,pas_ali},
4. A={Name,Pas},
5.
6. 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:

دلیل استثنا	معنای استثنا
timeout_value	مقدار زمان در یک عبارت "receive..after" به چیزی غیر از یک عدد صحیح یا infinity ارزیابی شود.
Noproc	تلاش برای پیوند یا نظارت به یک فرایند یا پورت غیر موجود.
noconnection	یک پیوند یا مانیتور به یک فرایند راه دور، شکسته شد زیرا ارتباط بین گره ها برقرار نشد یا قطع شد.
{nocatch,Term_x}	تلاش برای ارزیابی یک throw بدون ساختار catch Term_x عبارت پرتاب شده است.
system_limit	به محدودیت سیستم رسیده است.

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("~nPlease 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 فرایندها:

```
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]]).

```

```

24. start()->
25. Apid= spawn (m4,a,[]),
26. spawn (m4,b,[[Apid,5]]).

```

```

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.

```

```

20. _Other -> Apid1 ! {error},
21. 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
```

می توانیم فقط نام گره را وارد کنیم مانند دستور زیر:

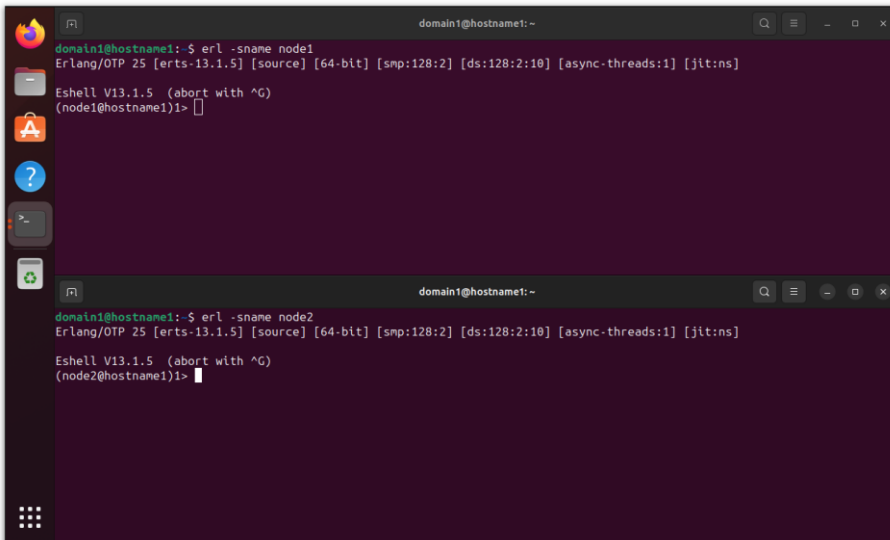
```
erl -sname node_name
```

-sname مخفف **short name** است. این پرچم باعث می شود سیستم زمان اجرای ارلنگ به یک گره توزیع شده تبدیل شود. توجه کنید که نام گره کاملاً واجد شرایط نیست یعنی آنکه قسمت نقطه و بعد از آن را ندارد (مانند **.com**). این پرچم تمام سرور های مورد نیاز برای توزیع یک گره را فراخوانی می کند. بعلاوه یک کوکی جادویی هم تولید می کند بعداً یاد خواهیم گرفت که چگونه آن را تغییر دهیم.

برای ساخت دو گره با نام کوتاه در اوبونتو دو ترمینال را شروع می کنیم و سپس دستور زیر را وارد می کنیم تا شل ارلنگ همراه با گره شروع شود:

```
erl -sname node_name
```

توجه کنید که چطور قبل از اعلان ارلنگ (**>1**) نام گره (**node1@hostname1**) آمده است و قسمت بعد از **@** را اضافه کرده اما نه نقطه و بعد از آن:



The image shows two terminal windows side-by-side. Both windows have a title bar that says 'domain1@hostname1:~'. The top window shows the command 'erl -sname node1' being executed, followed by system information and the Erlang shell prompt '(node1@hostname1)1>'. The bottom window shows the command 'erl -sname node2' being executed, followed by similar system information and the Erlang shell prompt '(node2@hostname1)1>'. On the left side of the terminal windows, there is a vertical dock with several application icons including a web browser, a file manager, and a terminal.

تصویر 12.1 : شروع دو گره در دو ترمینال لینوکسی

در اینجا این سوال پیش می آید که ارلنگ نام `hostname1` را از کجا آورده ، زمانی که ما آن را مشخص نمی کنیم مقدار آن برابر با نام کامپیوتر خواهد بود . پس نام کامپیوتر من " `hostname1` " است اما می توانستم آن را در زمان ساخت گره تغییر دهم. `-name` مانند `-sname` است اما نام آن می تواند کاملاً واجد شرایط باشد. پس گره ای که با دستور `-sname` ساخته شده باشد نمی تواند نقطه و بعد از نقطه را داشته باشد.

تصویر پایین صرفاً یک مثال برای نام بلند گره است:

```
(node2@hostname1)4> domain1@hostname1:~$ erl -name mynode@myhost.com
Erlang/OTP 25 [erts-13.1.5] [source] [64-bit] [smp:128:2] [ds:128:2:10] [async-threads:1] [jit:ns]

Eshell V13.1.5 (abort with ^G)
(mynode@myhost.com)1>
```

تصویر 12.2 : نام بلند گره

ما قصد داریم یک برنامه ساده را در دو گره داشته باشیم و از گره دیگر آن را فراخوانی کنیم. ماژول ما `d.erl` نام دارد:

```
1. -module(d).
2. -compile([export_all]).
3.
4. f(From)->
5.   From ! node().
6.
7. p()->
8.   io:format("my pid = ~w ~n my node = ~w ~n"
9.     ,[self(),node()]).
10.
11. c(A,B)->
12.   C2 = A+B,
13.   io:format("A+B = ~w ~n",[C2]).
```

تابع `f/1` یک `Pid` از فراخوانی کننده را می گیرد و نام گره خود را می فرستد. تابع `p/0` مقدار `pid` خود و آدرس گره خود را بر می گرداند. تابع `c/2` هم دو عدد می گیرد و حاصل جمع آنها را بر می گرداند. هدف از تابع `f/1` این است که تا زمانی که از گره دیگری آن را فراخوانی می کنیم تفاوت آدرس گره فراخوانی کننده و گره ای که ماژول در آن است را ببینیم و ببینیم چطور اطلاعات را به گره فراخوانی کننده می فرستد. هدف از `p/0` این است که ببینیم چطور بدون آنکه آدرس گره فراخوانی کننده را به آن ماژول بفرستیم اطلاعات را به شکلی به مان نشان می دهد که انگار ما در همان گره ماژول درخواست را فرستاده ایم. هدف از تابع `c/2` آن است که نشان دهیم عملیات محاسباتی را می توان به گره دیگر سپرد و به این شکل بار محاسباتی را بین گره ها تقسیم کرد.

این ماژول در یک گره قرار می گیرد و در گره دیگر توابع آن فراخوانی می شود. نخست اجازه دهید آن را در محیط شل محلی (غیر توزیع شده) اجرا کنیم:

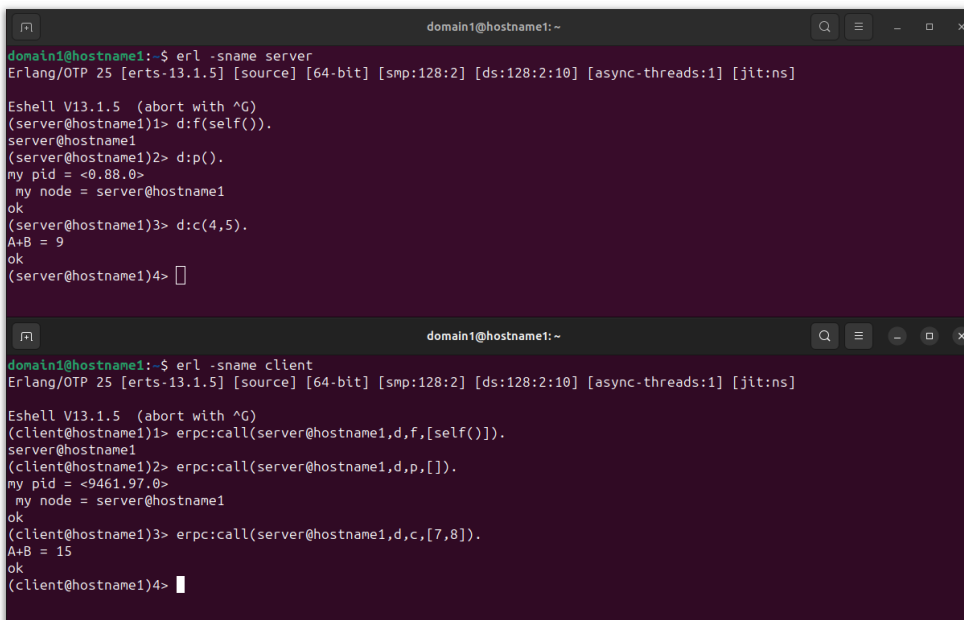
```
1> d:f(self()).
node1@hostname1

2> d:p().
my pid = <0.81.0>
my node = node1@hostname1
ok

3> d:c(3,4).
A+B = 7
ok
```

حال اجازه آن را در دو گره اجرا کنیم. نام دو گره را client و server می گذاریم. برای فراخوانی یک تابع از راه دور دو ماژول استاندارد در اختیار دارید rpc و erpc این دو ماژول توابعی برای ارسال درخواست به گره راه دور و جمع آوری پاسخ دارند. ماژول erpc جدیدتر است و در مستندات آن آمده است که زیرمجموعه ای از توابع پیشرفته در rpc است و در مقدار برگشتی و استثنا ها تمایز بسیاری دارد همچنین در عملکرد و مقیاس پذیری بهتر عمل می کند. بنابراین ما هم از ماژول erpc استفاده می کنیم. برای اطلاعات بیشتر kernel-8.5.4.1 را مطالعه کنید.

نکته: هر دو گره باید از erpc پشتیبانی کنند. به طور معمول از OTP 23 به بعد از این ماژول پشتیبانی می شود.



```
domain1@hostname1: ~
domain1@hostname1:~$ erl -sname server
Erlang/OTP 25 [erts-13.1.5] [source] [64-bit] [smp:128:2] [ds:128:2:10] [async-threads:1] [jit:ns]

Eshell V13.1.5 (abort with ^G)
(server@hostname1)1> d:f(self()).
server@hostname1
(server@hostname1)2> d:p().
my pid = <0.88.0>
my node = server@hostname1
ok
(server@hostname1)3> d:c(4,5).
A+B = 9
ok
(server@hostname1)4> █

domain1@hostname1: ~
domain1@hostname1:~$ erl -sname client
Erlang/OTP 25 [erts-13.1.5] [source] [64-bit] [smp:128:2] [ds:128:2:10] [async-threads:1] [jit:ns]

Eshell V13.1.5 (abort with ^G)
(client@hostname1)1> erpc:call(server@hostname1,d,f,[self()]).
server@hostname1
(client@hostname1)2> erpc:call(server@hostname1,d,p,[]).
my pid = <9461.97.0>
my node = server@hostname1
ok
(client@hostname1)3> erpc:call(server@hostname1,d,c,[7,8]).
A+B = 15
ok
(client@hostname1)4> █
```

تصویر 12.3 : ارتباط بین گره ها

همانطور که در تصویر بالا می بینید در گره server توابع ماژول d را به صورت محلی فراخوانی کردیم و در خط 1 نام گره ای را که ماژول d در آن در حال اجرا است را برمی گرداند. در خط 2 شناسه فرایندی و نام گره را برمی گرداند. در خط 3 هم جمع دو عدد را برمی گرداند.

حال آن را از گره دیگری فراخوانی می کنیم. در گره client با تابع call/4 از ماژول erpc، 3 در خواست را به گره server فرستادیم و 3 تابع ماژول d را در گره server اجرا کردیم ولی نتایج را در گره client دریافت کردیم توجه کنید که در خط 1 در پوسته گره client ما تابع d:f/1 را فراخوانی کردیم که نام گره ای را برمی گرداند که در آن اجرا می شود، حال نام آن گره چیست server@hostname1. در خط 2 در گره client از گره server تابع p/0 را از ماژول d فراخوانی کردیم که آدرس گره server را می دهد این مورد نشان می دهد که ارزیابی آن تابع از گره دریافت کننده درخواست (در اینجا گره server) انجام شده. این مورد به ما می گوید که می توانیم از یک گره درخواستی را به گره دیگر بفرستیم و انتظار داشته باشیم عمل ارزیابی در آن گره انجام شود (البته در صورت انجام تنظیمات و وجود ماژول کامپایل شده مربوطه). در خط 3 از گره client تابع c/2 را با دو آرگومان در گره server اجرا کردیم و این یعنی آنکه ما محاسبات را به گره دیگر هدایت کردیم (این فقط یک مثال است، بار محاسبات بسیار سنگین را در نظر بگیرید که باید در چند سرور تقسیم شود). حال بیاید کمی بیشتر با تابع erpc:call/4 آشنا شویم:

```
erpc:call(Node, Module, Function, [Args]).
```

مشخص است که منظور از Node، چيست. Module، Function، Args هم که نام گره است که تابع در آن اجرا خواهد شد یادآوری می کنم که باید حتماً ماژول در Node به صورت کامپایل شده وجود داشته باشد. اگر استثنای تولید نشود و عملیات موفق باشد نتیجه مورد انتظار را برمی گرداند. توجه کنید که آرگومان ها حتماً باید در لیست باشند. تا اینجا محاسبه توزیع شده را بین دو گره انجام دادیم. حال اگر بخواهیم یک درخواست را به طور همزمان برای چند گره ارسال کنیم باید از تابع multical/4 استفاده کنیم:

```
erpc: multical ([Node1, Node2,...], Module, Function, [Args]).
```

Node1 و Node2 نام گره های هستند که درخواست برای آنها ارسال می شود.

مثال:

```
(client@hostname1)4> erpc:multicall([server@hostname1,server2@hostname1],d,p,[]).
my pid = <9734.94.0>
my node = server@hostname1
my pid = <9735.93.0>
my node = server2@hostname1
[{ok,ok},{ok,ok}]
```

تصویر 12.4 : فراخوانی یک تابع در چند گره

توجه کنید که ما در خواست اجرای تابع `d:p/0` را به دو گره `server@hostname1` و `server2@hostname1` ارسال کردیم . نتیجه این فراخوانی برگرداندن نام هر دو گره بود که نشان می دهد جواب ها از دو گره آمده است.

یادآوری می کنم که در گره های که روی ماشین های فیزیکی مجزا هستند باید ماژول در تمامی آنها کپی و کامپایل شود.

اگر می خواهید که در خواستی را به یک گره بفرستید و منتظر جواب آن نیستید باید از `cast/4` استفاده کنید. این تابع مانند `call/4` است با این تفاوت که منتظر جواب نمی ماند . (مثلاً فرض کنید که می خواهید یک محاسبه سنگین انجام شود و نتیجه آن باید ذخیره شود و فعلاً نیازی به دانستن جواب ندارید) شکل کلی این تابع کتابخانه ای در ادامه آمده است.

erpc:cast(Node, Module, Function, Args).

اگر چند گره داشته باشیم و بخواهیم مراقبت کنیم که یک کره اشتباهی به یک گره دیگر درخواستی نفرستد می توانیم از کوکی ها استفاده کنیم. ارلنگ کوکی پیش فرض دارد که هر زمان یک گره را بدون کوکی راه اندازی کنید آن را در فایلی به نام `erlang.cookie` قرار می دهد و در زمان های بعدی که بخواهید یک گره بدون کوکی راه اندازی کنید از کوکی پیش فرض استفاده می کند. اگر بخواهیم آن کوکی را تغییر دهیم از پرچم `-setcookie X` استفاده می کنیم (X می تواند هر مقداری از حروف و اعداد باشد). اجازه دهید آن را در عمل ببینیم. ما سه گره قبلی را (`server@hostname1` و `server2@hostname1` و `client@hostname1`) دوباره راه اندازی کردیم اینبار با تنظیم یک کوکی و همان درخواست قبلی را فرستادیم ، نتیجه را در تصاویر زیر ببینید.

در تصویر زیر می بینید که یک گره با نام `server` و کوکی با مقدار `c1` راه اندازی کردیم:

```
domain1@hostname1:~$ erl -sname server -setcookie c1
Erlang/OTP 25 [erts-13.1.5] [source] [64-bit] [smp:128:2] [ds:128:2:10] [async-threads:1] [jit:ns]

Eshell V13.1.5 (abort with ^G)
(server@hostname1)1>
```

تصویر 12.5 : تنظیم کوکی برای گره server

در تصویر زیر می بینید که یک گره با نام `server2` و کوکی با مقدار `c2` راه اندازی کردیم:

```
(server2@hostname1)3> domain1@hostname1:~$ erl -sname server2 -setcookie c2
Erlang/OTP 25 [erts-13.1.5] [source] [64-bit] [smp:128:2] [ds:128:2:10] [async-threads:1] [jit:ns]
```

تصویر 12.6 : تنظیم کوکی برای گره server2

در تصویر زیر می بینید که یک گره با نام `client` و کوکی با مقدار `c1` راه اندازی کردیم:

```
(client@hostname1)6> domain1@hostname1:~$ erl -sname client -setcookie c1
Erlang/OTP 25 [erts-13.1.5] [source] [64-bit] [smp:128:2] [ds:128:2:10] [async-threads:1] [jit:ns]
```

تصویر 12.7 : تنظیم کوکی برای گره client

در ادامه در گره `client` ، مثل قبل یک درخواست را به هر دو گره فرستادیم. اما اینبار پاسخ فقط از طرف گره ای (`server`) تولید شد که کوکی آن مشابه کوکی گره `client` بود. اما از طرف گره `server2` یک پیام خطا به شکل `{error,{erpc,noconnection}}` دریافت کردیم که نشان می دهد اتصال برقرار نیست (به دلیل کوکی نامشابه) .

```
(client@hostname1)> erpc:multicall([server@hostname1,server2@hostname1],d,p,[]).
my pid = <9459.92.0>
my node = server@hostname1
[{ok,ok},{error,{erpc,noconnection}}]
```

تصویر 12.8 : فراخوانی یک تابع در چند گره بعد از تنظیم کوکی

12-2 تولید فرایند راه دور:

در قسمت قبل یاد گرفتیم که به یک ماژول در چند گره در خواستی ارسال کنیم اما اگر این ماژول از بین برود چه کاری باید انجام دهیم؟ مانند فصل "برنامه های همزمان" اینجا هم می توانیم فرایند های را روی گره ها ایجاد کنیم. برای تولید فرایند BIF های زیر را در اختیار داریم:

spawn(Node, Fun) -> pid()

این BIF یک فان Fun را روی گره Node شروع می کند و یک PID برمی گرداند.

spawn_link(Node, Fun) -> pid()

این BIF هم مانند BIF قبلی است با این تفاوت که به آن لینک هم می کند. تولید فرایند جدید و لینک فرایند فراخوانی کننده و فرایند جدید را به صورت یک واحد انجام می دهد. اگر گره Node وجود نداشته باشد یک سیگنال خروج با دلیل noconnection برگردانده می شود.

spawn(Node, Module, Function, Args) -> pid()

این BIF یک فرایند از تابع Function را با لیست آرگومان های Args از ماژول Module را روی گره Node تولید و شروع می کند و یک PID برمی گرداند.

spawn_link(Node, Module, Function, Args) -> pid()

این BIF هم مانند BIF قبلی است با این تفاوت که به آن لینک هم می کند. تولید فرایند جدید و لینک بین فرایند فراخوانی کننده و فرایند جدید را به صورت یک واحد انجام می دهد. اگر گره Node وجود نداشته باشد یک سیگنال خروج با دلیل noconnection برگردانده می شود. این BIF خیلی شبیه موارد مشابه ای است که در فصل "برنامه های همزمان" معرفی شده است. این کار نوشتن برنامه و درک آن را ساده می کند. در ادامه یک ماژول را می نویسیم به نام distribution که شبیه ماژول d است با این تفاوت که برای تولید فرایند بهینه شده است:

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.
9. {calculator,A,B} ->
10. C2= A+B,
11. io:format("A+B = ~w ~n",[C2]);
12.
13. _Other ->
14. io:format("error ~n"),
15. start1()
16. end.
17. %-----
18. start2(Node) ->
19. spawn(Node, fun() -> start1() end).
20. %-----
21. start3(Node)->
22. spawn(Node,io,format,["my pid = ~w ~n my node = ~w
~n",[self(),node()]]).
%-----

```

تابع `start1` اگر تاپلی به شکل `{print_address}` را در صندوق پستی خود دریافت کند آنگاه `pid` و آدرس گره خود را بر می گرداند. اگر تاپلی به شکل `{calculator,A,B}` بگیرد که بجای `A` و `B` باید عدد باشد، آنگاه آنها را جمع و حاصل آن را بر می گرداند. اگر چیز دیگری دریافت کند آنگاه یک پیام خطا چاپ می کند سپس در خط 15 به اول تابع بر می گردد. تابع `start2` آدرس یک گره را می گیرد و تابع `start1` را مانند یک `fun` در آن گره اجرا می کند. تابع `start3` آدرس یک گره را می گیرد و سعی می کند تابع `io` از ماژول `format` را با آدرس گره و `pid` خود بر گرداند.

در ادامه آنها را در ارلنگ اجرا می کنیم. نخست دو گره در دو ترمینال می سازیم به نام های `node1` و `node2`:

```
~$ erl -sname node1
```

```
~$ erl -sname node2
```

در ترمینال اول، دستور بالا یک گره به نام `node1` می سازد. توجه کنید دو گره را باید قبل از دستورات ادامه بسازید.

```

(node1@computername1)1>
Pid1=distribution:start2(node2@computername1).
<8995.92.0>

```


دستور بالا باعث می شود در گره محلی، تابع `start2` از ماژول `distribution` فراخوانی شود و آدرس گره دوم به عنوان آرگومان به آن داده شود. این تابع (`start2`) باعث می شود تابع `start1` در گره `node2` ساخته شود و PID آن برگرداند شود. حال `Pid1` شامل PID تابع `start1` در گره `node2` است.

```
(node1@computername1)2> Pid1 ! {print_address}.
{print_address}
my pid = <8995.92.0>
my node = node2@computername1
```

در دستور بالا یک پیام به شکل `{print_address}` به `Pid1` ارسال کردیم و یادمان هست که در تابع `start1` با ورود این پیام، نتیجه اجرای BIF های `self()` و `node()` برگردانده می شود. نتیجه این دستور به ما نشان می دهد تابع `start1` در گره `node2` اجرا شده است.

```
(node1@computername1)3>
Pid2=distribution:start3(node2@computername1).
<8995.93.0>
my pid = <0.86.0>
my node = node1@computername1
```

دستور بالا با فراخوانی تابع `start3` از ماژول `distribution` هم BIF های `self()` و `node()` را در گره `node2` چاپ می کند، اما چرا آدرس گره `node1` را برگردانده است؟ چون تابع `start3`، BIF های `self()` و `node()` را در گره `node1` اجرا می کند و نتیجه (در خط 22) را در گره `node2` چاپ می کند.

```
(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
```

دستورات دو خط بالا (خط 4 و 5) فقط برای نشان دادن آن است که چطور می توانیم از پوسته، یک فرایند را به شکل یک fun ، روی یک گره دیگر تولید کنیم .

```
(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
```

دستورات دو خط بالا فقط برای نشان دادن آن است که چطور می توانیم از پوسته، یک فرایند را به شکل MFA روی یک گره دیگر تولید کنیم .

برای اجرای صحیح برنامه لازم است که یک ورژن از ماژول و یک ورژن مشابه از ماشین مجازی ارلنگ را در تمام گره ها داشته باشیم. برای کپی کردن ماژول در تمام گره های متصل می توان از دستور nl(name_module) را اجرا کنیم . برای آنکه گره ها به هم متصل شوند دو تابع کتابخانه ای در اختیار داریم. تابع اول ping:

```
net_adm:ping(Node)
```

اگر اتصال برقرار شود خروجی pong خواهد بود . در غیر اینصورت pang را برمی گرداند. اطلاعات بیشتر را در صفحه راهنما در فایل kernel-8.5.4.1.pdf می توانید پیدا کنید.

تابع دوم connect_node:

```
net_kernel:connect_node(Node).
```

اگر اتصال به گره Node برقرار شود یا Node از قبل متصل باشد یا گره محلی باشد ، true برمی گرداند. اگر اتصال برقرار نشود false برمی گرداند . اگر گره Node زنده نباشد نادیده گرفته می شود. بعد می توانید با BIF ، nodes() ببینید که آیا این گره متصل شده است یا خیر.

توجه کنید که در بعضی منابع از net_kernel:connect (Node) استفاده می کنند. این تابع از ارلنگ حذف شده و net_kernel:connect_node(Node) جایگزین شده است.

BIF ، nodes() یک لیست از گره های متصل را نشان می دهد. اما BIF ، node () نام گره جاری را برمی گرداند. برای قطع یک اتصال بین دو گره از دستور زیر در یکی از آن گره ها استفاده می کنیم:

```
disconnect_node(Node).
```

اگر اتصال قطع شود true برمی گرداند . اگر اتصال قطع نشود false برمی گرداند. اگر گره زنده نباشد ignored برمی گرداند.

12-3 BIF های کوکی:

در ادامه با چهار BIF مرتبط با کوکی ها آشنا می شویم:

```
erlang:set_cookie(Cookie) -> true
```

اگر بعد از ساخته شدن گره بخواهید کوکی آن را تغییر دهید می توانید از این BIF استفاده کنید. اگر گره زنده باشد و تغییر کوکی موفق باشد true را برمی گرداند. این کوکی برای اتصال به تمام گره ها استفاده می شود. بنابراین باید کوکی دیگر گره ها هم همین کوکی باشد.

```
erlang:set_cookie(Node, Cookie) -> true
```

بعضی وقت ها چند گره داریم و آنها کوکی های متفاوتی دارند . در این صورت می توانیم با این BIF به سیستم بگوییم که کوکی گره جاری برای ارتباط با گره Node معادل Cookie است. در این صورت برای ارتباط با دیگر گره ها از کوکی اصلی استفاده می کند.

```
erlang:get_cookie() -> Cookie | nocookie
```

این BIF کوکی گره جاری را نشان می دهد. اگر گره زنده است کوکی آن را برمی گرداند در غیر اینصورت nocookie را برمی گرداند. اگر هیچ کوکی در زمان ساخت گره و بعد از آن تنظیم نکنیم آنگاه کوکی پیش فرض که در فایل erlang.cookie است معیار استفاده خواهد بود.

12-4 ثبت فرایند ها در گره های دور:

ثبت فرایند در گره ها هم مانند ثبت فرایند در یک گره است اما تفاوت در ارسال پیام به این فرایند ها است.
شکل کلی:

```
{Registered_Name , Node_name} ! Message
```

در اینجا Registered_Name نام ثبت شده فرایند است، بدون نام مازول و Node_name نام گره ای است که مازول و نام فرایندی ثبت شده در آن است. حال اگر چند فرایند ثبت شده در چند گره داشته باشیم و بخواهیم به همه آنها همزمان پیامی را ارسال کنیم به شکل زیر عمل می کنیم:

```
{Registered_Name , Node_name1} ! {Registered_Name,  
Node_name2} ! Message
```

توجه داشته باشید که باید علامت ! در انتهای هر تاپل "نام ثبت شده" باشد.

ما ماژول **distribution** را کمی تغییر دادیم تا برای ثبت فرایند مناسب باشد. برای کار با این ماژول نخست باید در هر گره **start2** را فراخوانی کرد. این تابع یک فرایند از تابع **start1** می‌سازد و سپس PID آن فرایند را با نام **reg_pid1** ثبت می‌کند. و برای آنکه این فرایند نام گره و PID خود را برگرداند ما باید یک پیام به شکل **{print_address}** به آن ارسال کنیم. اگر بخواهیم که فرایند تمام شود پیام **{down}** را به آن ارسال می‌کنیم. اگر پیام دیگری ارسال کنیم، برمی‌گردد به اول تابع. در ادامه می‌خواهیم سه گره بسازیم و از یک گره پیامی را برای آن نام ثبت شده در گره‌های دیگر ارسال کنیم.

نکته:

1- تابع **start2** باید در هر سه گره فراخوانی شود.

2- نام ثبت شده با اینکه در هر سه گره مشابه است اما معرف سه فرایند مجزا است.

```

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.
10. {down} -> ok,{down}.
11.
12. _Other ->
13. io:format("error ~n"),
14. start1()
15. end.
16. %-----
17. start2() ->
18. Pid1=spawn(fun() -> start1() end),
19. register (reg_pid1,Pid1).
20. %-----

```

گره اول client :

```
1- ~$ erl -sname client
```

```

2- (client@computername1)1>
    {reg_pid1,server2@computername1} !
    {reg_pid1,server1@computername1} !
    {print_address}.

3- {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

```

در گام نخست ما 3 ترمینال باز کردیم و خط 1 (که در بالا می بینید) را برای هر ترمینال وارد کردیم. سپس در گره های server1 و server2 دستور فراخوانی node_reg:start2() را وارد کردیم (خط 2 در ترمینال گره های server1 و server2) و خروجی آن فقط اتم true (در خط 3) بود نه خط 4 و 5. سپس به گره client رفتیم و تاپل {print_address} (خط 2) را به آن دو گره ارسال کردیم، تقریباً بلافاصله خروجی خط 4 و 5 در دو گره server2 و server1 برگردانده شد. باقی تاپل های مشخص شده در مازول را امتحان کنید.

برای آنکه بدانیم یک PID به کدام گره مربوط است از BIF زیر استفاده می کنیم:

```
node(pid).
```

مثال:

```

(client@computername1)1> node(<0.101.0>).
server2@computername1

```

در پایان، توجه داشته باشید که هر گره ای که به شبکه اضافه می شود، آن گره به هر گره دیگری وصل می شود و به آن نظارت می کند و پیام ها و سیگنال ها را از این گره ها دریافت و ارسال می کند. بنابراین سیگنال ها و پیام های خروج با پیوستن هر گره به شبکه به شکل

تصادفی افزایش می یابد، اما گره های پنهان به دیگر گره ها اجازه نظارت را نمی دهند بنابراین برای جلوگیری از نظارت دیگر گره ها و در نتیجه کاهش پیام های مربوط به آن از گره پنهان استفاده می کنند. گره های پنهان بحث پیشرفته تری است که اگر نیاز شد باید در مستندات ارلنگ بخوانید.

فصل 13: مدیریت خطا در فرایندهای همزمان

13-1 لینک کردن فرایندها:

```
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':~npid1>~w~npid2>~w~n~n",[Pid1,Pid2]  
   ).  
9. %-----  
10. p2(Pid1)->  
11. Pid1 ! {self(),{2,2}},  
12. receive  
13. {C} ->  
14. io:format("line 14 : ~n2+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':~nref1>~w~nref2>~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.%-----

```

```

28.Ref1 = erlang:monitor(process, Pid1),
29.Ref2 = erlang:monitor(process, Pid2),
30.io:format("line'30':~nref1>~w~nref2>~w~n~n"
    ,[Ref1,Ref2]),

```

```

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.

```

```

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,

```

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':~nref1>~w~nref2>~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 فرایندهای 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

```

```

8> regs().
** Registered procs on node nonode@nohost **
Name                Pid          Initial Call          Reds Msgs
a_pid                <0.89.0>     regi:b/0              38      0
application_controlle <0.44.0>     erlang:apply/2        788      0
code_server          <0.50.0>     erlang:apply/2        136102    0
erl_prim_loader      <0.10.0>     erlang:apply/2        231355    0

```

تصویر 13.1 : نتیجه دستور regs

مثال:

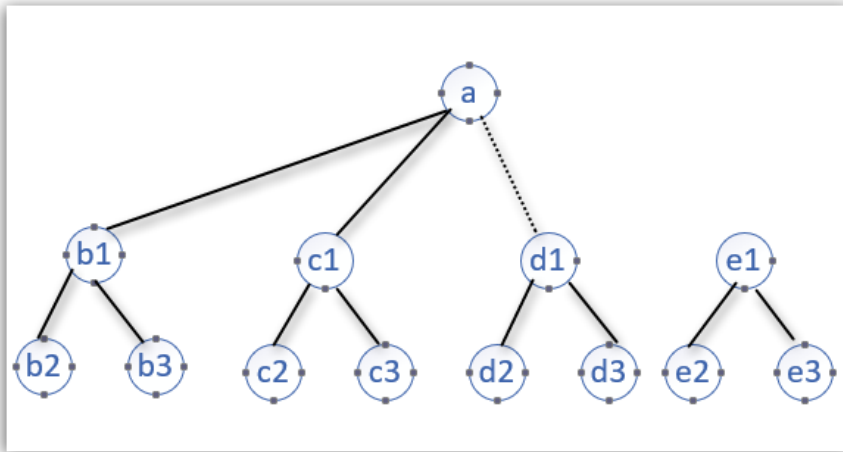
```

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.
14.Ref2 = make_ref(),
15.Pid_calculator1 ! {5,5,self(),Ref2},
16.receiver1(Pid_calculator1,Ref1,Ref2).
17.%-----
18.receiver1(Pid_calculator1,Ref1,Ref2)->
19.receive
20.{C,Pid_calculator1,Ref1}->
21.pid_ali ! {C},
22.receiver1(Pid_calculator1,Ref1,Ref2);

23.{C,Pid_calculator1,Ref2}->
24.pid_reza ! {C},
25.receiver1(Pid_calculator1,Ref1,Ref2)
26.end.
27.%-----
28.calculator1()->
29.receive
30.{A,B,Pid_init1,Ref_x}->
31.Pid_init1 ! {A+B,self(),Ref_x},
32.calculator1()
33.end.
34.%-----
35.ali()->
36.receive

```

```

37. {C}->
38.io:format("~n2+2= ~w~n",[C]),
39.ali()
40.end.
41.%-----
42.reza()->
43.receive
44.{C}->
45.io:format("~n5+5= ~w~n",[C]),
46.reza()
47.end.
48.%-----

```

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},

```

ok

```
{key_1,val_1},  
{key_1,val_1},  
{key_1,val_3}]
```

```
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 Key را 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 ets.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_aset=ets:new(aset, [ordered_set, named_table]).
aset
2> ets:insert(aset,[{a,1},{b,2},{c,3},{d,4},{e,5}]).
true
3> Oaset =ets:prev(aset,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 دهیم 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_aset=ets:new(aset, [ordered_set, named_table]),
6. ets:insert(aset,[{a,1},{b,2},{c,3},{d,4},{e,5}]),

```

7. table_to_file(Table_aset, "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(aset2, [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 ثابت شده `aset` و از نوع `ordered_set` می سازد :

```
Table_aset=ets:new(aset, [ordered_set, named_table]),
```

خط 6 اشیاء را در جدول ، درج می کند:

```
ets:insert(aset, [{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_aset, "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 :

```

12. file:write(File_ID, io_lib:format("~w.", [Data])),
13. %ets:tab2file(Data, File_ID),

```

```
file:close(File_ID).
```

```
{ok, Data} = file:consult("txt2.txt"),
```

```
Table = ets:new(ose2, [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)),
```

```
21.List_of_tuples = ets:tab2list(Table),  
22.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}].
```

:DETS 2-14

:DETS 14-2-1 معرفی ماژول

:DETS 14-2-2 توابع ماژول

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.%-----

```

```

13.match_2_1()->
14.M1=ets:match(my_table, '$1'),
15.io:format("Matched : ~p~n", [M1]).

```

```

1> match1:stat().
ok
2> match1:match_2_1().
Matched : [[{1,a,a1}],[{2,b,b2}],[{3,c,c31}],[{3,c,c31}],[{3,c,c32}]]
ok

```

```

17. match_2_2()->
18. [[M1,M2]]=ets:match(my_table, {'$20',a,'$2'}),
19. io:format("Matched : ~p==~p~n", [M1,M2]).

```

```

3> match1:match_2_2().
Matched : a1==1
ok

```

```
21.match_2_3()->
22.M3=ets:match(my_table, {'$3',c,'$4'}),
23.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
```

```
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("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
```

پیداآوری: 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}
```

```
function1()->  
F = fun() ->  
mnesia:write(...),  
mnesia:write(...)  
.  
.  
.  
end,  
mnesia:transaction(F).
```

:(Query List Comprehension) QLC 15-3

```
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).
```

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

```
4. -record(user_id,      {name, age=[], job}).
5. -record(user_address, {name, email=[], address,
    phone_number}).
```

```
6. user_id_table() ->
7. [
8. {user_id, ali,      50, programmer},
9. {user_id, hasan,    60, baker},
10. {user_id, reza,     70, baker},
11. {user_id, arman,    80, programmer},
12. {user_id, x,       15, student}
13. ].
```

```
14. user_address_table() ->
15. [
16. {user_address,
ali,      "ali@erlang_email.com",    ahvaz,      09000000
1},
17. {user_address,
hasan,    "hasan@erlang_email.com",  address1,   09000000
2},
18. {user_address,
reza,     "reza@erlang_email.com",   address2,   09000000
3},
19. {user_address,
arman,    "arman@erlang_email.com",  ahvaz,      09000000
4},
```

```
20. {user_address,  
x,      "x@erlang_email.com",      address3,      09000000  
5}  
21. ].
```

```
22. match()->  
23. F = fun() ->  
24. mnesia:match_object(user_id,  
{user_id, '_', '_', baker}, read)  
25. %io:format("hi=%d",[M])  
26. end,  
27. mnesia:transaction(F,5).
```

```
28. create_schema_tables() ->  
29. mnesia:create_schema([node()]),  
30. mnesia:start(),  
  
31. mnesia:create_table(user_id,  
[attributes,record_info(fields, user_id)]]),  
  
32. mnesia:create_table(user_address,  
[attributes, record_info(fields, user_address)]]),  
  
33. mnesia:stop().
```

```
34. loading_tables() ->  
35. mnesia:start(),  
36. mnesia:wait_for_tables([user_id,user_address],  
5000).
```

```
37. writing_in_tables() ->  
38. mnesia:clear_table(user_id),  
39. mnesia:clear_table(user_address),  
40. F = fun() ->  
41. lists:foreach(fun mnesia:write/1, user_id_table()),  
42. lists:foreach(fun mnesia:write/1,  
user_address_table()),
```

```

43. T=      #user_address{name=zz1,      email="zz1@a.com",
      address=zzcity, phone_number=99999999},
44. mnesia:write(T)
45. end,
46. mnesia:transaction(F).

```

```

47. view_the_entire_table() ->
48. F=fun()->
49. Q=qlc:q([X || X <- mnesia:table(user_address)]),
50. qlc:eval(Q) end,
51. {atomic, Val} = mnesia:transaction(F),
52. Val.

```

```

53. show_age_and_job() ->
54. F = fun() ->
55. Q=qlc:q([X#user_id.age, X#user_id.job] || X <-
      mnesia:table(user_id)]),
56. qlc:e(Q) end,
57. {atomic, Val} = mnesia:transaction(F),
58. Val.

```

```

59. people_over_60_years_old() ->
60. F=fun()->
61. Q=qlc:q([X || X <-
      mnesia:table(user_id),X#user_id.age>60]),
62. qlc:eval(Q) end,
63. {atomic, Val} = mnesia:transaction(F),
64. Val.

```

```

65. delete_object() ->
66. Oid = {user_id,x},
67. F = fun() ->
68. mnesia:delete(Oid)
69. end,
70. mnesia:transaction(F).

```

```

71.  () ->

```

```
72. F = fun() -> mnesia:read({user_id,ali}) end,  
73. 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}]}
```