Pazhoheshi.ir | پژوهشی دات آی آر

echo Security | tr ity n | tr "Secur" e2ma3 | head -c 6 | xargs -I 23 echo 23

آموزش فارسی لینوکس : قسمت پنجم

چهارشنبه, ۱۸ تیر ۱۳۹۳، ۰۱:۵۵ ب.ظ

103.2: Process Text Streams Using Filters - Part 2

Description : Candidates should should be able to apply filters to text streams.

Key Knowledge Areas:

Send text files and output streams through text utility filters to modify the output using standard UNIX commands found in the GNU textutils package.

The following is a partial list of the used files, terms and utilities:

fmt | od | join | paste | pr | split | uniq

در قسمت قبل با بعضی از دستورات آشنا شدیم و مابقی رو برای این قسمت گذاشتیم که لیست دستورات رو در بالا می تونید ببینید .

دستور uniq :

از این دستور برای پاک کردن خط های تکراری استفاده می شود به این صورت که خط هایی که دقیقا پشت سر هم به صورت تکراری هستند را پاک می کند .

```
$cat /etc/group | cut -d: -f4
$cat /etc/group | cut -d: -f4 | uniq
$cat /etc/group | cut -d: -f4 | sort | uniq
$cat /etc/group | cut -d: -f4 | sort | uniq -d
```

نکته : از سویچ d- برای نمایش خطوط تکراری استفاده می شود .

نکته : از سویچ c- برای نمایش تعداد خطور تکراری استفاده می شود .

```
$cat /etc/group | cut -d: -f4 | sort | uniq -u
$cat /etc/group | cut -d: -f4 | sort | uniq | tail -n 2
```

نکته : از سویچ u- فقط برای نمایش خطور منحصر به فرد استفاده می شود .

دستور split :

از این دستور برای تیکه تیکه کردن هر نوع فایلی استفاده می شود . به عنوان مثال :

```
$cat /etc/passwd | split -l10 pswd_
$ls pswd_*
```

نکته : با استفاده از سویچ l- تعداد خطوطی هر تیکه را مشخص کردیم .

نکته : ابتدای اسم فایلی که قرار است تیکه تیکه شود را pswd_ انتخاب کردیم .

نکته : از سویچ b- برای تقسیم به فایل هایی در حسب KB استفاده می کنیم .

نکته : به جای استفاده از حروف الفبایی در اسم فایل های تیکه تیکه شده . می توانیم از عدد استفاده کنیم . برای این منظور از سویچ b- استفاده می کنیم .

نکته : به صورت پیش فرض تعداد حروفی که در فایل های تیکه تیکه شده استفاده شده است 3 می باشد . برای تغییر این عدد می توانیم از سویچ a- استفاده کنیم .

طبقه بندی موضوعی

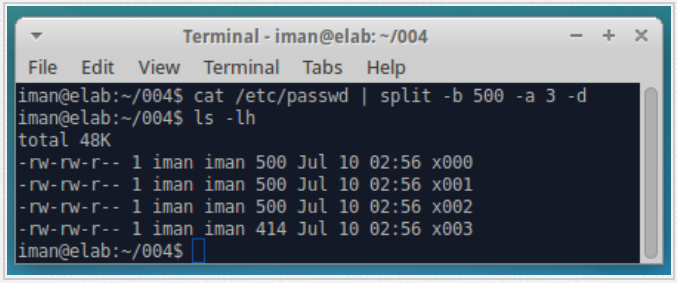
- مقاله آموزشی(۵)
- کتاب آموزشی(۱)
- فیلم آموزشی(۶)
- برنامه ی اختصاصی(۶)
- ارسال بلاگ(۵)
- آموزش فارسی لینوکس(۲)

خلاصه آمار

مجموع نمایش‌ها	۹۶۱۶
مجموع بازدیدکننده‌ها	۳۶۱۹
بازدیدکننده‌های امروز	۳
نمایش‌های دیروز	۱۷۳
مجموع مطالب	۲۳
مجموع نظرات	۳۷
عمر سایت	۹۳ روز
حاضرین در سایت	۱

بایگانی

- تیر ۱۳۹۳(۵)
- خرداد ۱۳۹۳(۲)
- اردیبهشت ۱۳۹۳(۵)
- فروردین ۱۳۹۳(۱۱)



دستور paste :

برای چسباندن فایل های تیکه تیکه شده در مرحله ی قبل از این دستور استفاده می کنیم . مثال :

```
$cat /etc/passwd | cut -d: -f3 > uid
$cat /etc/passwd | cut -d: -f1 > name
$paste uid name > users
```

نکته : به صورت پیش فرض بین کلمه های وصل شده tab قرار دارد . برای تبدیل tab به هر چیز دیگری می توانیم از سوییچ d- استفاده کنیم . مثال :

```
$paste -d: name uid > users
```

نکته : از سوییچ s- برای چاپ به صورت موازی استفاده می شود .

paste one file at a time instead of in parallel

دستور od :

برای Dump کردن فایل (ها) به فرمت Octal و یا فرمت های دیگر از این دستور استفاده می کنیم . مثال :

```
$cat /etc/passwd | od -ta
```

نکته : t مشخصه ی نوع فرمت و a مشخصه ی حالت ASCII است .

نکته : برای نمایش به صورت فرمت کارکتری از c استفاده می کنیم . مثال :

```
$cat /etc/passwd | od -ta
```

نکته : فرق بین حالت ASCII و کارکتری در جاهایی مثال new line مشخص می شود . به این صورت که در حالت ASCII , خط جدید به صورت nl و در حالت کارکتری خط جدید به صورت \n نمایش داده می شود .

نکته : از o برای نمایش به صورت مقدار Octal و از x برای نمایش به صورت hexadecimal استفاده می شود . مثال :

```
$cat /etc/passwd | od -to
$cat /etc/passwd | od -tx
```

دستور join :

دستور pr :

از این دستور برای چاپ فایل ها به صورتی که فایل قابل پرینت باشند استفاده می کنند . مثال :

```
$cat /etc/passwd | pr
$cat /etc/passwd | pr -h Linux Document
```

نکته : از سوییچ h- برای نمایش یک عبارت در بالا صفحه استفاده می شود . مثال :

```
$cat /etc/passwd | pr -h Pazhoheshi -l 10 -o 6
```

نکته : برای مشخص کردن تعداد خطوط در هر صفحه از سویچ l- استفاده می کنیم .

نکته : برای گذاشتن Space در اول هر خط می توانیم از سویچ o- استفاده کنیم .

103.4 Use streams, pipes and redirects

Description : Candidates should be able to redirect streams and connect them in order to efficiently process textual data. Tasks include redirecting standard input, standard output and standard error, piping the output of one command to the input of another command, using the output of one command as arguments to another command and sending output to both stdout and a file.

Key Knowledge Areas:

- Redirecting standard input, standard output and standard error.
- Pipe the output of one command to the input of another command.
- Use the output of one command as arguments to another command.
- Send output to both stdout and a file.

The following is a partial list of the used files, terms and utilities:

- tee
- xargs

Using "tee" Command ...

در قسمت های قبل دیدیم که می توانیم خروجی درست و یا خروجی غلط یک برنامه را به صورت ورودی به برنامه ای دیگر بدهیم . برای مثال خروجی دستور cat را به عنوان ورودی وارد دستور cut می کردیم . اما زمانی پیش می آید که در وسط این لوله کشی (انتقال خروجی یک برنامه به ورودی برنامه ای دیگر) می خواهیم خروجی برنامه قبل وارد مسیری دیگر شود . اصطلاحا وارد یک 3 راهی بشود .

به این صورت که اطلاعات هم به عنوان ورودی وارد برنامه ای جدید شده و هم در مسیری دیگر (مثلا به عنوان ورودی در برنامه ای دیگر) قرار بگیرد .

برای این منظور از دستوری به نام tee استفاده می کنیم . مثال :

```
$cat /etc/passwd | cut -d: -f1,3 | tee /tmp/users | sort -t: -k2 -n | \
>tail -10 > 10LastUids
```

در دستور بالا ابتدا فایل passwd را خوانیم سپس آن را وارد cut کردیم تا فقط قسمت username و uid آن جدا شود سپس با استفاده از دستور tee خروجی قسمت قبل را در فایل users واقع در دایرکتوری tmp ذخیره کردیم سپس همان خروجی (خروجی قسمت cut) را به عنوان ورودی وارد برنامه ی sort کردیم تا اطلاعات بر حسب uid مرتب شود . در مرحله ی بعد 10 خط آخر را خواندیم و در نهایت در فایل 10LastUids ذخیره کردیم .

مثالی دیگر :

```
$cat /etc/passwd | cut -d: -f1 | sort | tee users
```

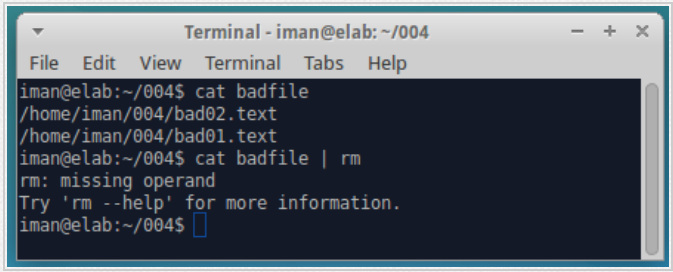
در دستور بالا ابتدا فایل passwd خوانده شده سپس وارد دستور cut شده و در نهایت هم با برنامه ی sort مرتب شده است . حال خروجی این مراحل وارد یک 3 راهی ای به نام tee می شود . راه اول که خروجی دستورات قبلی است که به عنوان ورودی وارد 3 راهی می شود . راه دوم نمایش خروجی در Standard Output است که در این جا صفحه ی نمایش و Terminal است و راه سوم از 3 راهی ذخیره کردن خروجی دستورات قبل در فایل users است .

در واقع از دستور بالا برای نمایش و ذخیره به صورت هم زمان استفاده شده است .

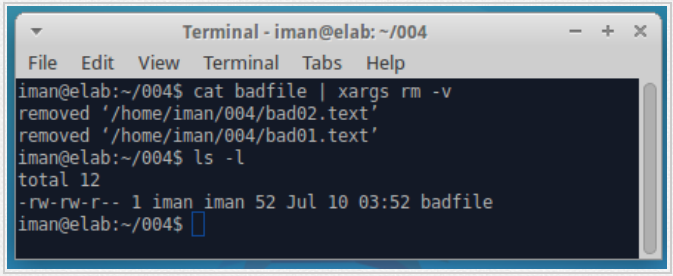
Using "xargs" Command ...

در قسمت قبل و قسمت های قبلی دیدیم که می توان خروجی یک برنامه و یا دستور را به عنوان ورودی وارد برنامه ای دیگر کرد . برای مثال خروجی دستور cat به عنوان ورودی وارد دستور head می تواند شود . اما دسته ای از برنامه ها هستند که این قابلیت را ندارند !

یعنی نمی توانند از خروجی دستور قبل ورودی دریافت کنند مثال برنامه ی rm . به عنوان مثل :



همان طور که از تصویر پیداست برنامه ی rm اروری مبنی بر **missing operand** دریافت کرده است . چاره ی کار استفاده از برنامه ای به عنوان xargs است . در واقع این برنامه خروجی دستور قبل که در این جا cat است را به عنوان آرگمان در مقابل دستور rm قرار می دهد . پس xargs را می توان به عنوان یک پچ برای هر برنامه ای که ورودی پذیر نیست در نظر گرفت .



نکته : سویچ p- برای حالت interactive است یعنی سوالی مبنی بر اجرای دستور از کاربر می پرسد .

نکته : در این مثال برنامه ی rm قابلیت پاک کردن دو فایل به صورت هم زمان را دارد یعنی دستور زیر (به عنوان مثال) :

```
$rm 01.text 02.text
```

اما زمانی را فرض کنید که یک برنامه تنها یک آرگمان می تواند بگیرد . در این حالت از سویچ n- برای این منظور استفاده می کنیم . به عنوان مثال :

```
$cat badfile | xargs -p -n1 rm
```

در مثال بالا در واقع دستور rm را 2 بار اجرا کرده ایم . به صورت : rm bad01.text و rm bad02.text

مثالی دیگر :

```
$cat badfile | xargs chmod o-rwx
```

در مثال بالا Standard Output دستور cat را به عنوان آرگمان به دستور chmod داده ایم .

103.8 Perform basic file editing operations using vi

Description : Candidates should be able to edit text files using vi. This objective includes vi navigation, basic vi modes, inserting, editing, deleting, copying and finding text.

Key Knowledge Areas:

- Navigate a document using vi.
- Use basic vi modes.
- Insert, edit, delete, copy and find text.

The following is a partial list of the used files, terms and utilities:

- vi
- /, ?
- h,j,k,l
- i, o, a
- c, d, p, y, dd, yy
- ZZ, :w!, :q!, :e!

در این قسمت در مورد کار با فایل و ویرایش آن ها آشنا خواهیم شد . تعداد زیادی editor هم به صورت گرافیکی و هم به صورت متنی وجود دارد اما در این جا فقط ویرایشگر VI و VIM را بررسی می کنیم .

در ابتدا باید گفت که VI ویرایشگری متن باز است که در سال 1976 با زبان C برای نسخه های اولیه ی BSD نوشته شده است . نسخه ی توسعه یافته ی این ویرایشگر VIM نام دارد که اولین نسخه ی آن در 1991 منتشر شده است

تعریف ها :

حالت Insert : در این حالت (Mode) فایل را ویرایش می کنیم .
حالت Command : حالت صدور فرمان
حالت Ex یا Command line : همان حالت معروف " : " است که دستورات را در این حالت وارد می کنیم .
حالت Visual : فقط در Vim موجود است و برای high light و انتخاب کردن متن استفاده می شود .

سویچ ها :

h : برای رفتن به سمت چپ به اندازه یک کارکتر

j : برای آمدن به پایین به اندازه یک خط

k : برای رفتن به بالا به اندازه یک خط

k : برای رفتن به سمت چپ به اندازه یک کارکتر

H : برای رفتن به ابتدای صفحه در همان صفحه که هستید مثلا رفتن به ابتدای صفحه در صفحه ی سوم .

L : برای رفتن به پایین صفحه در همان صفحه که هستید مثلا رفتن به پایین صفحه در صفحه ی دوم .

gg : برای رفتن به ابتدای صفحه ی نخست .

G : برای رفتن به انتهای صفحه ی آخر .

w : برای رفتن به ابتدای لغت بعدی در همان خط .

b : برای برگشتن به ابتدای لغت قبلی در همان خط .

0 : برای برکشت به اول خط در همان خط که هستید .

$: برای رفتن به آخر خط در همان خط که هستید .

Ctrl-B : برای رفتن به یک صفحه عقب .

Ctrl-F : برای رفتن به یک صفحه جلو .

i : برای رفتن به حالت Insert در همان موقعیتی که هستید .

I : برای رفتن به اول خط همراه با رفتن به حالت Insert .

a : برای رفتن به یک کارکتر جلو همراه با رفتن به حالت Insert .

A : برای رفتن به آخر خط همراه با رفتن به حالت Insert .

o : ساخت یک خط جدید بعد از همان خطی که در آن هستید همراه با رفتن به حالت Insert .

O : ساخت یک خط جدید قبل از همان خطی که در آن هستید همراه با رفتن به حالت Insert .

r : برای Replace کردن کارکتری که در آن جا هستید .

R : شروع Replace کردن در همان جا که هستید . برای مثال یک کلمه را به این صورت ویرایش کردید حال اگر از Back Space استفاده کنید , کلمه ی قبل را مشاهده خواهید کرد .

x : برای پاک کردن در همان جایی که هستید . معادل این سویچ Del می باشد .

X : برای به عقب پاک کردن در همان جایی که هستید . معادل این سویچ Back Space در حالت Insert می باشد .

s : برای پاک کردن کارکتری که در آن جا هستید همراه با رفتن به حالت Insert .

S : برای پاک کردن خطی که در آن جا هستید همراه با رفتن به حالت Insert .

dd : برای پاک کردن کل خط و رفتن در اول خط بر .

D : برای پاک کردن کل خط , از انتهای خط تا آن جا که مکان نما قرار دارد .

C : برای پاک کردن کل خط , از انتهای خط تا آن جا که مکان نما قرار دارد همراه با رفتن به حالت Insert .

yy : برای کپی کردن کل خطی که در آن جا هستید .

P : پیست کردن قبل از خطی که در آن جا هستید .

p : پیست کردن بعد از خطی که در آن جا هستید .

/ : برای سرچ کردن (بعد از / کلمه ای که می خواهید سرچ کنید را وارد کنید)

n : رفتن به کلمه ی پیدا شده ی بعدی .

N : رفتن به کلمه ی پیدا شده ی قبلی .

:q : برای خارج شدن از ویرایشگر . (برای زمانی که تغییری در فایل ایجاد نشده است)

:q! : برای خارج شدن از ویرایشگر بدون ذخیره کردن .

:w : برای ذخیره کردن فایل ویرایش شده .

:w file :برای ذخیره کردن در فایلی دیگر

```
1. :w >> file
2. :w! file
3. :wq
4. :x
5. ZZ
```

1. برای Append شدن در فایلی دیگر .
2. برای ذخیره نشدن در فایل file .
3. برای ذخیره و خارج شدن .
4. راهی میانبر برای ذخیره و خارج شدن سریع .
5. دقیقا همانند شماره ی 4 عمل می کند .

```
1. :n
2. :n!
3. :e file
4. :e! file
5. :r file
6. :! Command
```

1. زمانی پیش می آید که دو یا چند فایل را با یک ویرایشگر باز کرده ایم . به این صورت :

```
$vim file1 file2 file3
```

در این حالت برای جابجایی به سمت فایل بعدی از این دستور استفاده می کنیم .

2. دقیقا مثل دستور قبل عمل می کند با این تفاوت که اگر تغییری در یکی از فایل ها بوجود آمده باشد بدون در نظر گرفتن تغییرات به فایل بعدی می رود .
3. برای فراخوانی فایلی متنی در درون vim
4. برای فراخوانی فایلی متنی بدون در نظر گرفتن تغییرات در فایل قبلی
5. برای خواندن یک فایل دیگر و نوشتن بر روی فایلی که با vim باز کردید (در ادامه ی جایی که مکان نما قرار دارد)
6. برای اجرا یک دستور در VIM بدون بستن vim

نکته : روان شدن در کار با این سویچ ها و در کل کار با این ویرایشگر زمان لازم دارد .

103.5 Create, monitor and kill processes

Description : Candidates should be able to perform basic process management.

Key Knowledge Areas:

- Run jobs in the foreground and background.
- Signal a program to continue running after logout.
- Monitor active processes.
- Select and sort processes for display.
- Send signals to processes.

The following is a partial list of the used files, terms and utilities:

& | bg | fg | jobs | kill | nohup | ps | top | free | uptime | killall

Displaying System Processes ...

دستور ps :

برای مشاهده ی پروسه های در حال اجرا می توان از این دستور استفاده کرد .

نکته : برای مشاهده ی پروسه های در حال اجرای تمامی یوزر ها از سویچ a استفاده می کنیم .

نکته : برای مشاهده ی پروسه هایی که هم در محیط گرافیکی و هم در شل اجرا شده اند از سویچ x استفاده می کنیم .

نکته : برای مشاهده و مرتب کردن پروسس ها بر اساس یوزر از سویچ u استفاده می کنیم . مثال :

```
$ps
$ps a
$ps x
$ps ax
$ps axu
```

نکته : این نوع سویچ هایی که بدون - در مثال بالا استفاده شد را اصطلاحا سویچ های BSD می گویند چرا که در BSD به این صورت استفاده می شود .

نکته : در گنو / لینوکس سویچ ها با دو - یعنی -- مشخص و استفاده می شود .
نکته : در یونیکس سویچ ها با یک - مشخص و استفاده می شود . مثال :

```
$ps ax | wc -l
```

در مثال بالا با استفاده از برنامه ی wc و سویچ l- می توانیم تعداد خطوط را مشاهده کنیم . فقط باید توجه کرد که خط اول جزء پروسس های سیستم نمی باشد و بایسی یک عدد از عددی که برنامه ی wc نمایش می دهد کم کنیم .

نکته : از سویچ U- می توانیم برای مشخص کردن پروسس های یک یوزر خاص استفاده کنیم . مثال :

```
$ps u -U iman
```

دستور pstree :

همان طور که از اسم دستور مشخص است . برای نمایش تمامی پروسس ها به صورت tree از این دستور استفاده می کنیم . مثال :

```
$pstree | less
$pstree -p
```

نکته : از سویچ p- برای نمایش دادن PID استفاده می کنیم .

نکته : وقتی از سویچ p- استفاده می کنیم , این طور به نظر می رسد که تعداد پروسس ها بیش تر شده است . در واقع در دستور pstree (بدون سویچ p-) پروسس های همنام فقط یک بار نشان داده می شود با این که PID های متفاوتی دارند اما چون سویچ p- برای مشخص کردن PID است . در نتیجه تمامی پروسس های هم نام همراه با PIDشان نمایش داده می شود .

دستور top :

در واقع شکلی بهتر از دستور ps و pstree را می توان در دستور و یا برنامه ی top مشاهده کرد . تقریبا چیزی مشابه با Task Manager ویندوز . روند کاری این برنامه به این صورت است که پروسس های در حال اجرا را نشان داده و هر 3 ثانیه خود را آپدیت می کند . اگر بر روی سیستم خود برنامه ی top را اجرا کنید خواهید دید که پروسس ها بر حسب بیش ترین استفاده از CPU لیست می کند .

نکته : برای مشخص کردن زمان refresh می توان از سویچ d- استفاده کرد . مثال :

```
$top -d 0.5
```

نکته : برای مشخص کردن تنها پروسس هایی که از CPU استفاده می کنند از سویچ i- استفاده می کنیم . مثال :

```
$top -d 0.5 -i
```

نکته : گفتیم که برنامه هر 3 ثانیه به صورت پیش فرض خود را Refresh می کند . برای مشخص کردن تعداد Refresh ها از سویچ n- استفاده می کنیم . مثال :

```
$top -d 2 -n 3 -i
```

در دستور بالا برنامه را در 3 بازه ی زمانی 2 ثانیه ای اجرا کرده ایم .

نکته : زمانی پیش می آید که می خواهیم خروجی این برنامه را در داخل فایلی ذخیره کنیم . اگر طبق روال و سنت به شکل زیر عمل کنیم , دچار مشکل خواهیم شد .

```
$top -d 2 -n 3 -i > out
```

برای حل این مشکل از سویچ b- استفاده می کنیم . مثال :

```
$top -d 1 -n 10 -i -b > out
```

System Memory and SWAP Space Information ...

دستور free :

برای مشاهده مقدار مصرفی و یا فضای خالی در ram و Swap استفاده می شود . مثال :

```
$free
```

نکته : برای مشاهده ی مقدار فضای استفاده شده و یا خالی بر حسب مگابایت از سویچ m- و برای نمایش جمع کل از سویچ t- استفاده می کنیم . مثال :

```
$free -mt
```

نکته : می توان مشخص کرد که این برنامه برای مثال در 2 ثانیه اجرا شود . در این صورت از سویچ s- استفاده می کنیم . مثال :

```
$free -mt -s 2
```

System Load Average Other Information ...

دستور uptime :

برای مشاهده ی مقدار ساعتی که سیستم روشن بوده و مقدار load average و هم چنین تعداد یوزر های فعال , از این دستور استفاده می کنیم .
مثال :

```
$uptime
```

نکته : در خروجی دستور بالا در قسمت load average ممکن است خروجی ای شبیه به این خروجی دریافت کنید : load average: 0.38, 0.18, 0.15

توجه داشته باشید که قسمت اول از سمت چپ مقدار میانگین مصرف CPU در 1 دقیقه و سپس در 5 و 15 دقیقه است .

ادامه ی 103.5 در جلسه ی بعد

منبع : پژوهشی دات آی آر | Pazhoheshi.ir
نویسنده : e2ma3n

دانلود PDF این مقاله

هر گونه نظر / ایراد / اشکال و ... را از طریق ایمیل e2ma3n@Gmail.com در میون بزارید . با تشکر

E2MA3N ۹۳/۰۴/۱۸ ۰ ۰

آموزش LIC1 101 | آموزش LPIC1 101 103.2 | آموزش LPIC1 101 103.4 | آموزش LPIC1 101 103.5 | آموزش LPIC1 101 103.8

آموزش lpic | آموزش برنامه ی vim | آموزش فارسی لینوکس | پروسس ها در لینوکس | پروسه ها در لینوکس

نظرات (۰)

هیچ نظری هنوز ثبت نشده است

ارسال نظر

ارسال نظر به صورت ناشناس

نام * E2MA3N

پست الکترونیک e2ma3n@gmail.com

سایت یا وبلاگ azhoheshi.ir

پیام *

کد امنیتی *

نظر بصورت خصوصی ارسال شود

پست الکترونیک برای عموم قابل مشاهده باشد

ارسال

بلاگ بیان، رسانه متخصصان و اهل قلم