LINUX OS

Masood Mosalman Tabar

Masood.Mmt@ipm.ir

gcg.ipm.ac.ir

فهرست مطالب

- فرایند ها - Processes
- نصب و کامپایل برنامه از سورس کد
- پوسته Bash

- Chroot
- Performance Tuning
- KERNEL
- The Cron System

فرایندها

- منظور از فرایند یک برنامه یا کد در حال اجرا در سیستم می باشد.
- قسمتی از کدهای یک فرایند در مود کاربر و قسمتی در مود کرنل (هسته) اجرا می شود.

- رابطه فرایند ها به صورت سلسله مراتبی است. به این معنی که یک فرایند می تواند یک یا چند فرایند ایجاد کرده و هر کدام از فرایند های جدید نیز به همین منوال می توانند فرایند جدید ایجاد کنند.
- یک فرایند روی فرایندهایی که ایجاد کرده است کنترل کامل دارد و حتی می تواند اجرای آنها را متوقف کند.
- توقف یک فرایند می تواند باعث توقف فرایندهای تولید شده توسط آن فرایند شود.
- در لینوکس به هر کدام از فرایندها یک PID نسبت داده می شود.

فرایندها

- Foreground
  - موقعی که یک دستور را از اعلان اجرامی کنیم و منتظر می‌مانیم تا پس از پایان اجرای آن دوباره شکل اعلان ظاهر شود می گوییم فرایند به صورت پیش زمینه اجرا شده است.
- Background
  - موقعی که بعد از دستور علامت & را قرار می دهیم و دستور را اجرا می کنیم فرایند به صورت پس زمینه اجرا می شود به این معنی که بلافاصله شکل اعلان ظاهر می شود در حالیکه اجرای برنامه هنوز ادامه دارد.

فرایندها

- دستور jobs لیست برنامه‌هایی که در پس‌زمینه در حال اجراست را نشان می دهد.
- برای اینکه یک فرایند را به صورت پس‌زمینه اجرا کنیم، در حین اجرای برنامه ctrl+z را زده و سپس دستور bg را اجرا می‌کنیم.
- دستور fg، برنامه ای که در پس زمینه است را به صورت پیش‌زمینه اجرا می کند.

فرایندها

[taghi@linuxserver behnami]$ cat > a

۱

اجرای دستور به صورت پیش زمینه

۲

کلید ctr+z را بزنید

[۱]+ Stopped cat >a

[taghi@linuxserver behnami]$ bg

۳

این دستور فرایند را پس زمینه می کند

[taghi@linuxserver behnami]$ fg

cat >a

۴

این دستور مجددا فرایند را به صورت پیش زمینه در می آورد.

فرایندها

Daemons

- فرایندهای پس زمینه سیستمی را Daemon می گوییم
- این فرایندها معمولاً در حین بوت شدن دستگاه اجرا می شوند.
- معمولاً این نوع فرایندها با ترمینال کاری ندارند.

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         ۵     ۱  ۰  ۱۹۹۹ ?        ۰۰:۰۰:۱۴ [kswapd]
bin        ۲۵۴     ۱  ۰  ۱۹۹۹ ?        ۰۰:۰۰:۰۰ [portmap]
root       ۳۰۷     ۱  ۰  ۱۹۹۹ ?        ۰۰:۰۰:۲۳ syslogd -m ۰
root       ۳۵۰     ۱  ۰  ۱۹۹۹ ?        ۰۰:۰۰:۳۴ httpd
```

در قسمت ترمینال ؟ آمده است. به این معنی که این فرایند ترمینال ندارد.

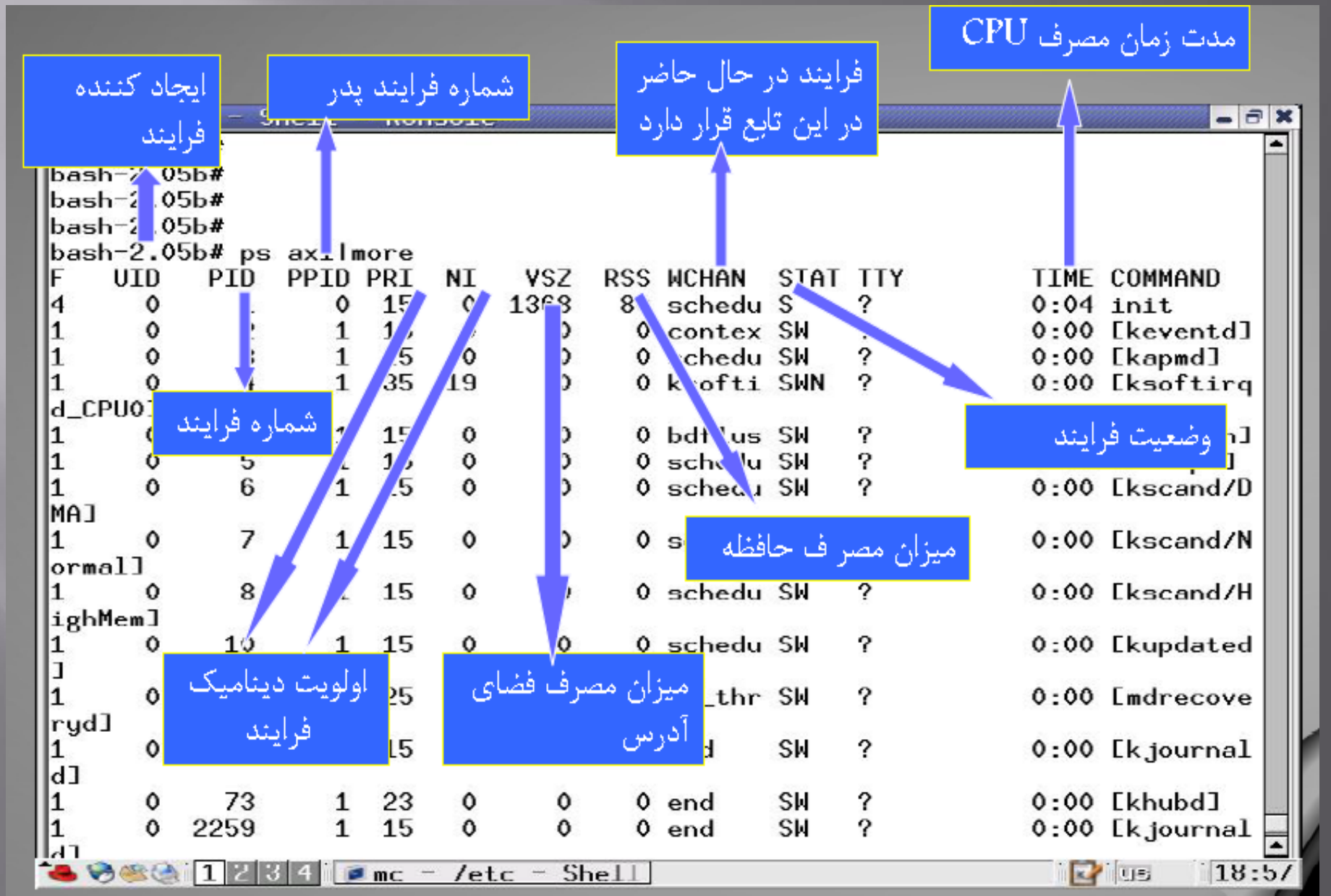
فرایندها


این علامت باعث می شود فرایند به صورت پس زمینه اجرا شود.
شماره فرایند یا PID
[root@penguinvm log]# sleep ۱۰h &
[۱] ۶۷۱۸
[root@penguinvm log]# ps -ef
UID PID PPID C STIME TTY TIME CMD
root ۶۷۱۸ ۶۶۹۲ ۰ ۱۴:۴۹ ttyp۰ ۰۰:۰۰:۰۰ sleep ۱۰h
شمار کار یا job
شمار فرایند پدر یا PPID
ترمینالی که فرایند روی آن در حال اجرا است
کسی که فرایند را ایجاد کرده است.

فرایندها

- PID
  - این فیلد یک شناسه منحصر به فرد است که در حین اجرای یک فرایند به آن نسبت داده می شود.
  - با اجرای هر فرایند جدید در سیستم این فیلد افزایش می یابد
  - با دانستن شماره یک فرایند یا فیلد PID می توانیم اطلاعات مختلفی راجع به فرایند استخراج کنیم از طریق دستورهایی مثل ps، pstree، jobs
- PPID
  - فرایندی که یک فرایند جدید ایجاد می کند پدر و فرایند جدید فرزند نامیده می شود. فرایند پدر از طریق فیلد ppid شناخته می شود.

دستور ps


مدت زمان مصرف CPU
ایجاد کننده فرایند
شماره فرایند پدر
فرایند در حال حاضر در این تابع قرار دارد
شماره فرایند
وضعیت فرایند
میزان مصرف حافظه
اولویت دینامیک فرایند
میزان مصرف فضای آدرس
bash-2.05b#
bash-2.05b#
bash-2.05b#
bash-2.05b# ps axl|more
F UID PID PPID PRI NI VSZ RSS WCHAN STAT TTY TIME COMMAND
4 0 0 15 0 1368 8 schedu S ? 0:04 init
1 0 1 15 0 0 0 contex SW ? 0:00 [keventd]
1 0 1 15 0 0 0 schedu SW ? 0:00 [kapmd]
1 0 1 35 19 0 0 ksofti SWN ? 0:00 [ksoftirq
d_CPU0]
1 0 1 15 0 0 0 bdflus SW ? 0:00 [bdflush]
1 0 5 1 15 0 0 0 schedu SW ? 0:00 [kupdated]
1 0 6 1 15 0 0 0 schedu SW ? 0:00 [kscand/D
MA]
1 0 7 1 15 0 0 0 schedu SW ? 0:00 [kscand/N
ormal]
1 0 8 1 15 0 0 0 schedu SW ? 0:00 [kscand/H
ighMem]
1 0 10 1 15 0 0 0 schedu SW ? 0:00 [kupdated
]
1 0 25 0 0 0 mdrecov SW ? 0:00 [mdrecove
ryd]
1 0 15 0 0 0 kjourn SW ? 0:00 [kjournal
d]
1 0 73 1 23 0 0 0 end SW ? 0:00 [khubd]
1 0 2259 1 15 0 0 0 end SW ? 0:00 [kjournal
d]
1 2 3 4
mc - /etc - Shell
18:57

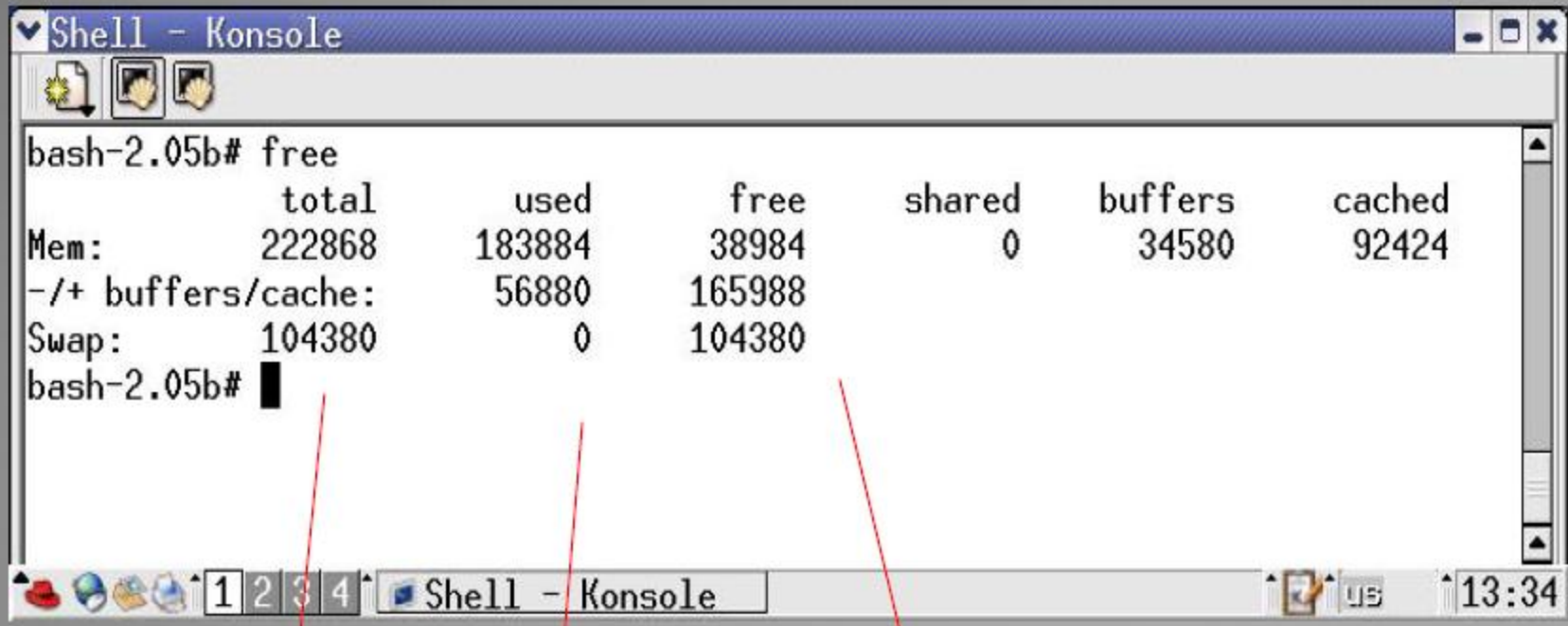
دستور kill, killall

- kill: از طریق این دستور و دادن شماره یک process به آن فرایند، یک سیگنال ارسال می کنیم.
- برای دیدن لیست سیگنالهایی که می توان به یک فرایند ارسال کرد دستور Kill -l را اجرا کنید.
- killall: با این دستور می توانیم به یک process از طریق نام آن سیگنال ارسال کنیم.

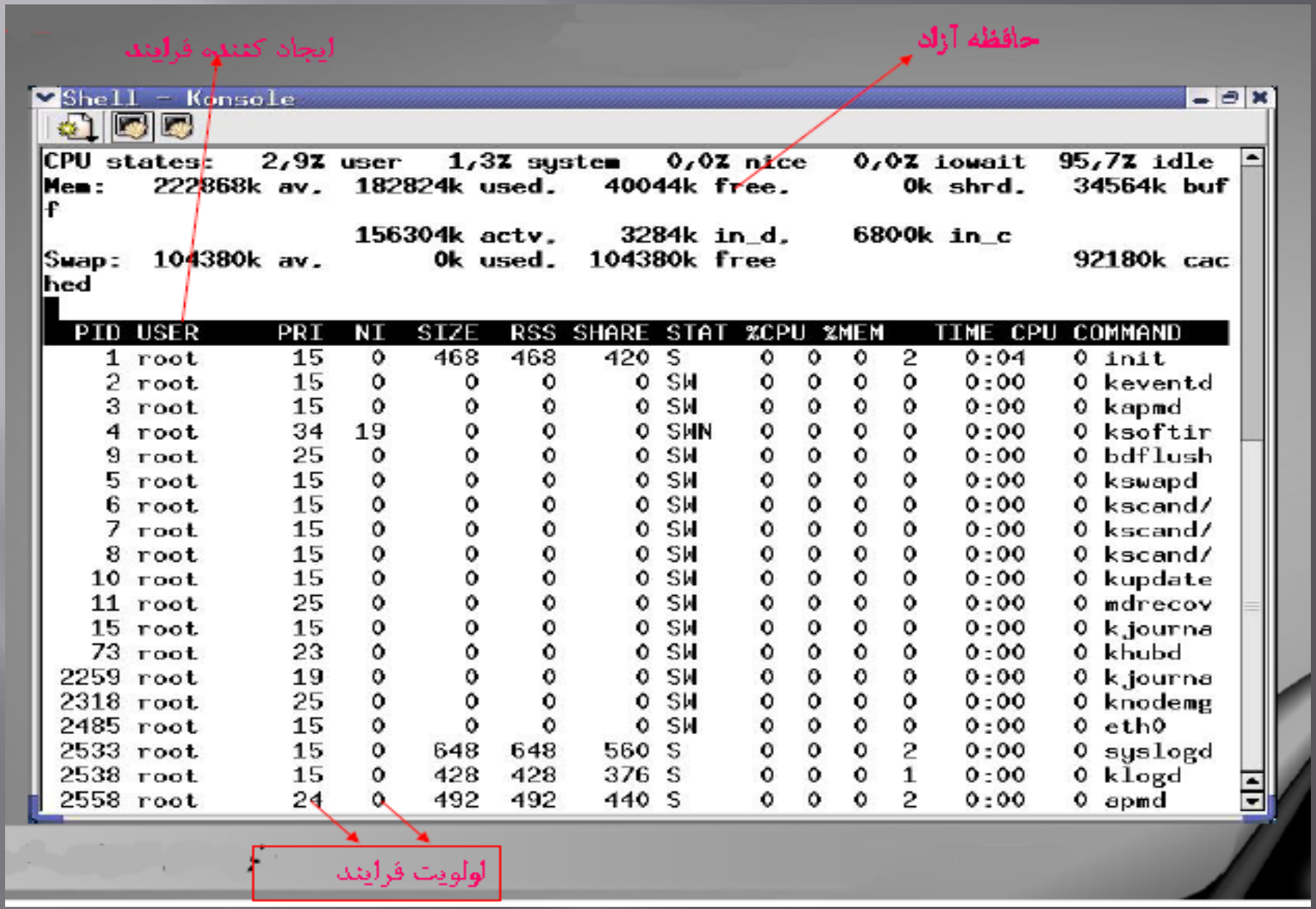
دستور top, free

- free: این دستور اطلاعاتی درباره حافظه استفاده شده و دست نخورده دستگاه شما می دهد.
- top: این دستور لیست فرایندها را نشان می دهد. این دستور فرایندها را بر اساس میزان مصرف منابعی چون حافظه و پردازنده مرتب می کند.

دستور free


Shell - Konsole
bash-2.05b# free
total used free shared buffers cached
Mem: 222868 183884 38984 0 34580 92424
-/+ buffers/cache: 56880 165988
Swap: 104380 0 104380
bash-2.05b#
1 2 3 4
Shell - Konsole
us
13:34
کل حافظه موجود
حافظه مصرف شده
حافظه آزاد

دستور top


ایجاد کننده فرایند
حافظه آزاد
Shell - Konsole
CPU states: 2,9% user 1,3% system 0,0% nice 0,0% iowait 95,7% idle
Mem: 222868k av. 182824k used. 40044k free. 0k shrd. 34564k buf
f
156304k actv. 3284k in_d. 6800k in_c
Swap: 104380k av. 0k used. 104380k free 92180k cac
hed
PID USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME CPU COMMAND
1 root 15 0 468 468 420 S 0 0 0 2 0:04 0 init
2 root 15 0 0 0 0 SW 0 0 0 0 0:00 0 keventd
3 root 15 0 0 0 0 SW 0 0 0 0 0:00 0 kapmd
4 root 34 19 0 0 0 SWN 0 0 0 0 0:00 0 ksoftir
9 root 25 0 0 0 0 SW 0 0 0 0 0:00 0 bdflush
5 root 15 0 0 0 0 SW 0 0 0 0 0:00 0 kswapd
6 root 15 0 0 0 0 SW 0 0 0 0 0:00 0 kscand/
7 root 15 0 0 0 0 SW 0 0 0 0 0:00 0 kscand/
8 root 15 0 0 0 0 SW 0 0 0 0 0:00 0 kscand/
10 root 15 0 0 0 0 SW 0 0 0 0 0:00 0 kupdate
11 root 25 0 0 0 0 SW 0 0 0 0 0:00 0 mdrecov
15 root 15 0 0 0 0 SW 0 0 0 0 0:00 0 kjourna
73 root 23 0 0 0 0 SW 0 0 0 0 0:00 0 khubd
2259 root 19 0 0 0 0 SW 0 0 0 0 0:00 0 kjourna
2318 root 25 0 0 0 0 SW 0 0 0 0 0:00 0 knodemg
2485 root 15 0 0 0 0 SW 0 0 0 0 0:00 0 eth0
2533 root 15 0 648 648 560 S 0 0 0 2 0:00 0 syslogd
2538 root 15 0 428 428 376 S 0 0 0 1 0:00 0 klogd
2558 root 24 0 492 492 440 S 0 0 0 2 0:00 0 apmd
اولویت فرایند

دستورات id, w, who, uname, last

- id: برای دریافت UIDها و GID کاربر جاری در سیستم از این دستور استفاده می شود.
- w: این دستور نشان می دهد که چه کسانی داخل سیستم شده اند و چه کاری انجام می دهند.
- who: نشان می دهد که چه کسانی وارد سیستم شده اند.
- uname: اطلاعاتی درباره سیستم به کاربر می دهد مثلا" از طریق پارامتر s– کاربر میتواند از نسخه kernel آگاه شود.
- last: این دستور لیستی از اسامی کاربرانی که وارد سیستم شده اند در اختیار شما قرار می دهد.

- $id
- uid=۵۲۲(karamoz) gid=۵۲۲(karamoz) groups=۵۲۲(karamoz)

Last

- این دستور آخرین کسانی را که روی سیستم وارد شده اند را نشان می دهد.
- علاوه بر نام افراد ساعت ورود و خروج آنها از سیستم نیز مشخص می باشد.

دستور diff, find, grep

Diff

دستوری برای تشخیص و یافتن تفاوت بین فایل های متنی حتی در کوچکترین تغییر

Find

دستوری برای یافتن فایل ها بر روی کامپیوتر، این دستور پرچم های زیاد و پیچیده ای برای جستجو حتی در داخل فایل ها دارد

Grep

جستجوی فایل ها از روی یک یا چند الگوی آرگومانی، String یا هرگونه توصیف قابل جستجو

```
find /etc/mail -exec grep -l "Your Text" {} \;
```

نصب و کامپایل برنامه و سورس کد در لینوکس

تقریبا تمامی سورس کد ها به صورت tarballهستند زیرا استفاده از tarballها می تواند حجم سورس را کمپرس کرده و تمامی فایل های لازم یک سورس را در د اخل یک فایل برای ما فراهم آورد .

برای کامپایل کردن یک سورس کد ابتدا ما باید فایل tarآنرا Extractکرده و فایل های داخل آنرا بیرون بیاوریم .

tar zxvf packagename.tar.gz -C ~/source
tar jxvf packagename.bz -C ~/source

حالا باید وارد فلدر Extractشده ی سورس شوید و دستورات زیر را به ترتیب اجرا کنید
./configure

این دستور تمامی Dependencyهای لازم (وابستگی ها و Libraryها) برای نصب یک برنامه را چک کرده و محیط را برای نصب برنامه آماده می کند .

نصب و کامپایل برنامه و سورس کد در لینوکس

make

این دستور ، سورس کد را کامپایل کرده و آنرا به زبانی که برای کامپیوتر ما قابل فهم باشد تبدیل می کند .

sudo make install

به کمک این دستور ما دسترسی های مدیریتی لازم برای نوشتن در فایل ها و فلدر های سیستمی را اخذ کرده و برنامه ی سورس را که هم اکنون کامپایل شده ، نصب می کنیم .

make clean

و در آخر به کمک این دستور عمل پاکسازی فایل های تمپ و اضافه که بعد از عملیات کامپایل و نصب دیگر به آنها نیازی نداریم را انجام میدهیم .

ذکر این نکته لازم است که در اوبونتو شما برای انجام عملیات کامپایل و نصب برنامه های سورس به برنامه های :

build-essential, checkinstall, libgtk1.2-dev

نیاز دارید

SHELL

شل یا پوسته همان واسط بین یک کاربر و سیستم عامل می باشد

مفسر متنی است که از طریق آن می توان همه خط فرمان ها و ابزار ها را فراخوانی کرد.

انواع bsh; ksh; csh, bash, tcsh را شامل می شود که در اینجا به پرکاربرد ترین آنها اشاره می کنیم

BASH

bash

- هر برنامه ای که در سیستم یونیکس اجرا می شود با سه فایل استاندارد کار می کند.
- فایل ورودی استاندارد که برنامه، ورودی خود را از این فایل می خواند.
- فایل خروجی استاندارد، که برنامه خروجی خود را به این فایل ارسال می کند.
- فایل استاندارد خطاها، که برنامه پیامهای خطا را با این فایل می فرستد
- این سه فایل به ترتیب با اعداد ۰ و ۱ و ۲ در سیستم شناخته می شوند
- به صورت پیش فرض فایل ورودی، کنسول یا همان صفحه کلید است.
- فایل خروجی استاندارد، ترمینال (صفحه تصویر) می باشد
- فایل استاندارد خطاها نیز ترمینال است.
- ما می توانیم به جای این فایلهای استاندارد از فایلهای دیگری برای ورودی یا خروجی دستورات استفاده کنیم که برای انجام این کار از علائم زیر استفاده می کنیم.

bash

علائم redirection	توضیح علامت
cmd > file	خروجی برنامه را به file می فرستد
cmd ۰<file	ورودی استاندار برنامه را از file می خواند
cmd ۱>file	خروجی برنامه را به file می فرستد
cmd ۲>file	خطاهای برنامه را به file می فرستد
cmd >>file	خروجی برنامه را به انتهای file اضافه می کند.
cmd <<TAG	برنامه ورودی استاندارد را تا رسیدن به TAG می خواند
cmd۱ | cmd۲	خروجی استاندارد cmd۱ به عنوان ورودی استاندارد cmd۲ استفاده می شود

Bash - wildcards

علامت	مفهوم
*	این علامت با رشته ای از کاراکترها تطبیق می کند. طول این رشته می تواند صفر باشد.
?	این علامت با یک کاراکتر دلخواه تطبیق می کند.
[۰-۹]	این علامت یک مجموعه را تعریف می کند. برای مثال [a-z] یعنی مجموعه حروف a تا z
[^a-z]	این علامت با کاراکترهایی تطبیق می کند که در این مجموعه نباشند.
{a,b,c}	مقادیری که در {} آمده است به ترتیب جایگزین می شوند.

Bash - wildcards

- *#ls x -al*
- لیست تمام فایلهای دایرکتوری جاری را نشان می دهد.
- *#ls [abc,۰-۹]x*
- تمام فایلهایی که با یکی از حروف b ، a یا c و یا با یک عدد شروع شده باشد
- *#ls ???d*
- نام تمام فایلهای ۴ حرفی که به d ختم می شوند را نشان می دهد.
- *#rm f{۱,۲,۴}.c*
- فایلهای f۱.c،f۲.c،f۴.c پاک می شوند.
- نکته مهم:
- تفسیر علائم فوق توسط shell انجام می شود بدین معنی که در مثال اول قبل از آنکه دستور ls اجرا شود shell علامت × را تفسیر کرده است و به جای این علامت عملاً لیست کلیه فایلها به دستور ls داده می شود.

Bash - wildcards

علامت	مفهوم
echo “the $var text”	در خروجی این دستور $var با مقدار آن جایگزین میشود.
echo ‘the $var text’	علائم داخل ‘ ’ پردازش نمی شود. خروجی این دستور رشته the $var text می باشد.
I=`ls *.c` echo $I	دستور ls اجرا شده و خروجی آن در I قرار می گیرد.

Bash - wildcards

علامت	مفهوم
cmd۱; cmd۲; cmd۳	ابتدا cmd۱ اجرا می شود. پس از پایان اجرای این دستور، cmd۲ اجرا می شود.
cmd۱ && cmd۲	cmd۱ اجرا می شود. اگر اجرای آن با موفقیت همراه بود cmd۲ نیز اجرا می شود
cmd۱ || cmd۲	cmd۱ اجرا می شود. اگر اجرای آن با موفقیت همراه نبود cmd۲ نیز اجرا می شود.
cmd۱ & cmd۲	cmd۱ به صورت پس زمینه اجرا شده و همراه آن cmd۲ نیز اجرا می شود.

Bash - predefined

- وقتی در محیط shell کار می کنید باید به این نکته توجه داشته باشید که متغیرهای محیطی زیادی وجود دارند که از قبل تعریف شده اند و مقدار آنها قابل استفاده است.
- مثال:
  - HOME: مسیر دایرکتوری کاربر
  - LOGNAME: نام کاربری که الان در سیستم کار می کند
  - PATH: مسیرهایی که سیستم برای پیداکردن برنامه های اجرایی جستجو می کند
  - TERM: ترمینال مورد استفاده

Bash – related files

- etc/profile : این فایل اطلاعات محیط کاربری هر کاربر را ذخیره میکند. این فایل هنگامی اجرا میشود که شما به سیستم وارد شده و شل آغاز به کار میکند. این فایل مقادیر پیش گزیده مسیر، شکل اعلان فرمان، حداکثر تعداد فایلی که شما میتوانید ایجاد کنید و مجوز های پیش گزیده برای فایلهایی که ایجاد میکنید را تعیین میکند. همچنین این فایل متغییر های محیطی مانند محل صندوق پستی و اندازه فایلهای تاریخچه را تنظیم میکند.

Bash – related files

- ~/.bashrc : این فایل حاوی اطلاعات مربوط به bash هر کاربر میباشد. این فایل هنگامی خوانده میشود که به سیستم وارد میشود و هر گاه که یک شل جدید باز میکنید. اینجا بهترین مکان برای ذخیره متغیرهای محیطی و فرمانهای مستعار خاص خودتان است.

- ~/.bash_logout : این فایل هر گاه که شما از سیستم خارج میشوید اجرا میشود. این فایل فقط صفحه نمایش را پاک میکند.

CHROOT

فرمان chroot را می توان یک زندان مخوف برای userid ها و کاربرانی که بعضی ازقوانین امنیتی را رعایت ننمایند به حساب آورد.

با استفاده از امکانات chroot می توان root فایل سیستم را که "/" می باشد به دایرکتوری دیگری تغییر داده و نرم افزاری که تحت root جدید می باشد به هیچ وجه نمی تواند به "/" مراجعه نموده و از انجا به فایل سیستم و یا دایرکتوری های دیگر برود و اصطلاحا می گویند که با chroot می توان پروسس را زندانی نمود.

CHROOT

فورمت فرمان chroot به صورت زیر می باشد.

chroot NEWROOT [COMMAND...]

به عنوان مثال اگر دایرکتوری به نام myoldroot داشته باشیم که تحت این دایرکتوری ساختاری مانند "/" داشته باشد می توان تحت دایرکتورب myoldroot رفته و اگر بخواهیم فایل /etc/passwd را بررسی نمائیم، سیستم اجازه دسترسی به فایل واقعی /etc/passwd را نداده و فایل /etc/passwd که در اختیار ما قرارخواهد گرفت، فایل /myoldroot/etc/passwd خواهد بود.

chroot /myoldroot vi /etc/passwd

CHROOT

chroot ناجی راهبر!

نرم افزار راه انداز Linux bootable بعد از mount نمودن دیسک سخت به شما اطلاع می دهد که می توانید با اجرای chroot سراغ دیسک سخت رفته و بدانید که mount point دیسک سخت معادل "/" بوده و تمام فرمانها و فایل های دیسک سخت شما دردسترس و برای خواندن و یا تغییر آماده می باشد.

chroot برنامه کاربردی را زندانی و راهبر را نجات می دهد

CHROOT

chroot ناجی راهبر!

اگر سیستم به دلایل زیر، بالا نیاید:

- بهم ریختگی kernel
- خراب شدن و یا بهم ریختگی /boot
- یا هردلیل دیگری که سیستم درست بالا نیاید.

شما بایستی از CD مربوط به نصب Linux استفاده نمائید.

بعد از قراردادن CD اول، Linux یکی از انتخاب ها Rescue Mode می باشد و وقتی که این حالت را انتخاب می نمائید

CHROOT

نجات سیستم!

قطعا اسامی پارتیشن را قبلا یادداشت نموده اید.

- boot with Boot-CD
- mkdir /mnt /new
- mount /dev/hda1 /mnt/new
- mount /dev/hda2 /mnt/new/boot
- mount /dev/hda3 /mnt/new/var

CHROOT

نجات سیستم!

برای link نمودن /proc واقعی به ساختار جدید، دستور زیر را به کار می بریم.

- mount -o bind /proc /mnt/new/proc/

همین فعالیت برای link نمودن /dev واقعی به ساختار جدید انجام می دهیم.

- mount –o bind /dev /mnt/new/dev/

CHROOT

نجات سیستم!

اگر فکر می کنید که فایل پیکربندی شبکه خراب شده است دستورزیر را اجرا نمائید.

- cp /etc/resolv.conf /mnt/new/etc/

با اجرای دستور chroot می توانید وارد دیسک سخت شده و هر فعالیتی را انجام دهید.

- chroot /mnt/new /bin/bash

CHROOT

فرمان زیر، شما را به root کامپیوتر معیوب برده و از هم اکنون می توانید هر فعالیتی از جمله Compile نمودن kernel اجرای lilo و حتی می توانید بازی! کنید.

```
cd /
```

بعد از پایان فعالیت و تنظیم و تعمیر! سیستم عامل، فرمان exit را وارد نموده و سپس با استفاده از دستور umount با دیسک سخت خداحافظی نموده و سیستم را reboot نمائید به امید اینکه سیستم از طریق دیسک سخت بالا بیاید....

Performance Tuning

راهبر همواره بایستی سیستم را کنترل نموده و با ابزارهائی که در اختیار دارد، اجزای مختلف خادم را تحت نظرداشته و با بدست آوردن اطلاعات و داده های جمع آوری شده به بالابردن کارائی سیستم بپردازد.

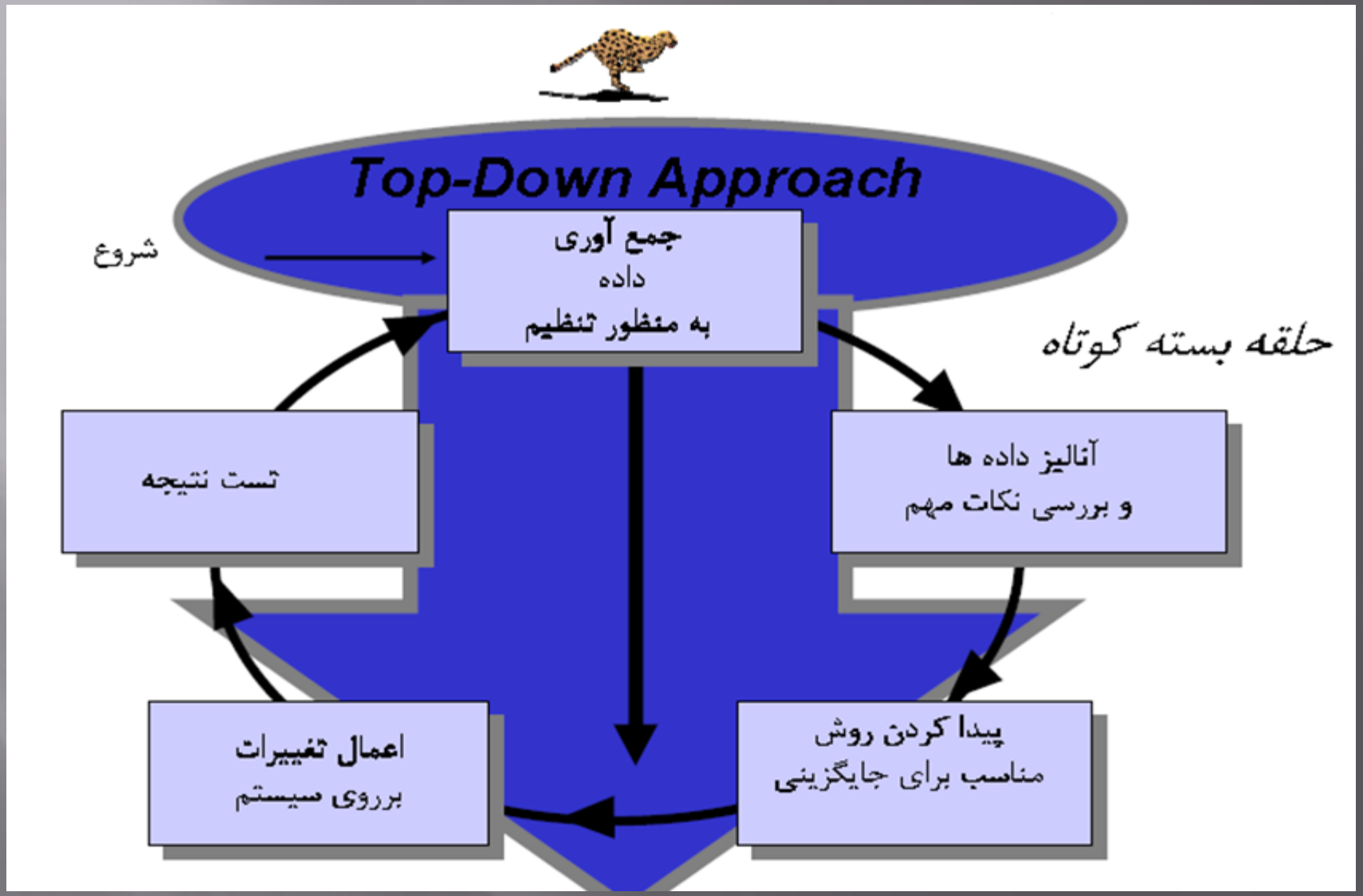
شاید بتوان گفت که تا حدی بالابردن کارائی سیستم یک هنر است که فقط با ممارست، علاقه و کنترل سیستم و گوش دادن به درد دل کاربران جدی کامپیوتر است که می توان سیستم را کارا تر نمود.

Performance Tuning

سوالاتی که بایستی پاسخ آن را بیابید

- آیا تنظیم به صورت سیستماتیک را تجربه کرده اید؟
- آیا ابزار های مناسب برای تنظیم سیستم را دراختیاردارید ؟
- آیا می دانید که همه عوامل مرتبط با تنظیم سیستم نبایستی به یکباره تغییر نماید ؟

Performance Tuning



Performance Tuning

انواع مونیتور نمودن

- **مونیتورنمودن سیستم**
  - پروسسور
  - حافظه
  - شبکه
  - دیسک
  - تجهیزات دیگری که به کارگرفته شده است.

Performance Tuning

تحت سیستم عامل **Linux** ابزارهای زیادی برای بررسی سیستم وجود دارد که تعدادی از آن ها عبارتند از:

- iostat
- sar
- vmstat
- top
- ps
- free
- nfsstat
- netstat
- mpstat
- uptime
- Ifconfig
- w

Performance Tuning

SAR(System Activity Report)

بوسیله فرمان sar می توان از وضعیت پردازنده و حافظه و فضای swap و بسیاری از اطلاعات دیکر مطلع گردید.

ضمنا می توانیر sar را طوری تنظیم نمائید که در مقاطع مختلف اطلاعات مورد نیاز را در فایلی یادداشت نماید.

استفاده نادرست از sar باعث خواهد شد که سیستم به دلیل اجرای این فرمان کند تر شده و بجای اینکه وضع بهتر شود، کارائی سیستم پائین آمده و کاربر ناراضی تر خواهد شد.

Performance Tuning

SAR(System Activity Report)

فرمات کلی فرمان sar نسبتا مفصل بوده و سعی خواهد شد که با ذکر مثال به بررسی این دستور پرداخته شود.

مثال ۱:

sar 2 12

فرمان sar هر ۲ ثانیه وضعیت سیستم را نمایش داده، پس از ۱۲ بار نمایش، با نشان دادن میانگین متوقف خواهد شد.

Performance Tuning

IOSTAT فرمان

avg-cpu:	%user	%nice	%sys	%idle									
	2.2	0.02	0.23	97.6									
Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	svctm	%util
/dev/sda	0.2	1.2	0.36	1.19	4.51	19.51	2.26	9.75	15.5	2.68	864	812	12.6
/dev/sda1	0	0	0	0	0	0	0	0	5.12	0	268	174	0
/dev/sda2	0	0	0	0	0	0	0	0	8	0	66.7	66.7	0
/dev/sda3	0.2	1.2	0.36	1.19	4.51	19.51	2.25	9.75	15.5	0.63	864	810	12.6

آمار مربوط به وضعیت پردازنده و آمار ورودی خروجی هر پارتیشن

KERNEL

سه نوع دسته بندي كلي براي **هسته سيستم‌هاي عامل** وجود دارد:

1. هسته يكپارچه Monolithic، كه انتزاع سخت‌افزاري نيرومندي را فراهم مي‌آورد.
2. ريز هسته Microkernel، كه مجموعه‌اي كوچك از انتزاع ساده سخت‌افزاري را به وجود مي‌آورد و از نرم‌افزارهايي با نام سرويس‌دهنده Serverاستفاده مي‌كنند تا قابليت بيشتري را ارايه دهند.
3. هسته دورگه Hybridيا "ريز هسته اصلاح شده"، كه شباهت زيادي به ريز هسته دارد، با اين تفاوت كه به منظور اجراي سريع‌تر، شامل كدهايي اضافي در فضاي هسته مي‌باشد.

برون‌هسته Exokernel، كه هيچ گونه انتزاعي را فراهم نمي‌كنند، ولي با استفاده از كتابخانه‌اي از توابع librariesبراي افزايش كارايي، دسترسي مستقيم يا نيمه‌مستقيم به سخت‌افزار را فراهم مي‌كنند.

KERNEL

هسته يكپارچه (Monolithi)

هسته يكپارچه، يك رابط مجازي سطح بالا بر روي سخت‌افزار تعريف مي‌كند. همچنين مجموعه‌اي از توابع براي پياده‌سازي سرويس‌دهنده‌هاي سيستم عامل، مانند مديريت پردازش‌ها Process Management، همزماني Concurrencyو مديريت حافظه را فراهم مي‌آورد.

اگر تمام اجزايي كه به اين عمليات سرويس مي‌دهند از كل مجموعه هسته جدا باشند، از لحاظ همبستگي كد در تنگنا سختي خواهيم بود و با توجه به اينكه تمام اجزا در يك فضا اجرا مي‌شوند، بروز خطايي در يكي از آنها مي‌تواند كل سيستم را مختل كند.

KERNEL

طرفداران هسته‌هاي يكپارچه عقيده دارند كه اگر كدي خطا دارد نبايستي در هسته قرار داشته باشد (متعلق به هسته باشد). چرا كه در غير اين صورت، برتري اندكي نسب به ريز هسته‌ها خواهند داشت. سيستم‌هاي عامل Linuxو Unixرا مي‌توان جزو پيشرفته‌ترين هسته‌هاي يكپارچه دانست

از طرفي ديگر، وقتي كه پياده‌سازي تكميل و قابل اطمينان شد، شرايط همبستگي تنگاتنگ بين اجزاي داخلي باعث مي‌شود كه امكانات سطح پايين سيستم به طور موثري در دسترس قرار گيرد و منجر به يك هسته يكپارچه، با كارآيي بسيار بالا شود.

KERNEL

Kernel Moduleلینوکس چیست

ماژول، قطعه ی نرم افزاری در بخشی جدا از Coreسیستم هستش که در زمان فراخوانی، پیوند و فعال میشه و یکسری عملیات تعریف شده ای رو انجام میده.
این ماژول می تونه سرویس باشه، سیستم فایل باشه، پروتکل شبکه باشه، سیستم callباشه و یا درایور یک سخت افزار باشه; که در هر صورت **ماژول** نام داره.

ماژول ها در لینوکس به دو گروه تقسیم میشن:
1- Kernel module
2- Loadable kernel module (LKM)

Kernel moduleکه با قرار دادن سورس ماژول در داخل پوشه های سورس Kernel، همراه با Kernelکامپایل میشه.
Loadable kernel moduleکه با Loadکردن ماژول کامپایل شده در داخل سیستم در حال اجرا ایجاد میشه.

KERNEL

پیکر بندی هسته

۱. حرکت به دایرکتوری که Source هسته در آن وجوددارد
۲. تغییرات در پیکربندی پیش فرض(اختیاری)
۳. تعریف کد Extraversion
۴. مشخص نمودن پیش فرض های پیکربندی هسته
۵. ساخت هسته و ماجول ها
۶. نصب هسته و ماجول ها
۷. تولید ramdisk
۸. تجدید نظر نمودن در پیکربندی grub
۹. تولید Boot Disk جدید

KERNEL

بخشی از اجزاء /usr/src/linux

دایرکتوری مربوط به پیکربندی Linux

```
total 1480
drwxr-xr-x  11 root root  4096 May  6 19:38 abi
drwxr-xr-x   3 root root  4096 May  6 19:16 arch
drwxr-xr-x   8 root root  4096 May  6 19:20 b
drwxr-xr-x   2 root root  4096 May  6 19:16 config
-rw-r--r--   1 root root 18691 Feb 24  2003 COPYING
-rw-r--r--   1 root root 79886 Feb 24  2003 CREDITS
drwxr-xr-x   4 root root  4096 May  6 19:38 crypto
drwxr-xr-x  31 root root  4096 May  6 19:16 Documentation
drwxr-xr-x  43 root root  4096 May  6 19:41 drivers
drwxr-xr-x  47 root root  4096 May  6 19:34 fs
drwxr-xr-x  26 root root  4096 Jul 16 11:17 include
drwxr-xr-x   2 root root  4096 May  7 01:38 init
drwxr-xr-x   2 root root  4096 May  6 19:38 ipc
drwxr-xr-x   2 root root  4096 May  6 19:26 kernel
```

CRON Jobs

یک سرویس و یک قابلیت است که به شما این امکان را میدهد که دستورات و یا به طور کلی عملیاتی را در زمان های مورد نظر خود به صورت دوره ای و متناوب اجرا نمایید .

سرویس cronمعمولا به طور پیش فرض بر روی سیستم عامل های لینوکس نصب میشود و تنها کافیست که آن را به کار بگیرید و استفاده نمایید.

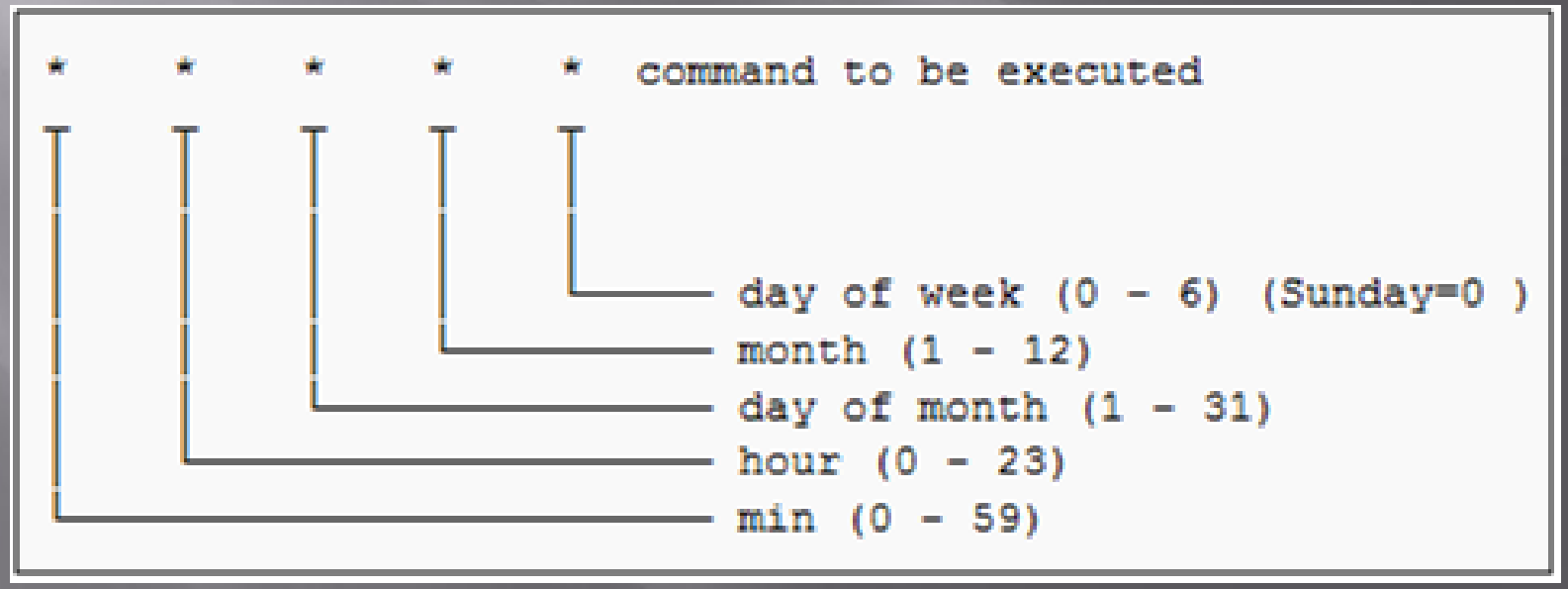
برای ایجاد و یا تغییر فایل CronTabکاربری خودتان از دستور زیر استفاده کنید:

crontab -e

بعد از وارد کردن این دستور فایل CronTabجدید ساخته شده و شما می توانید دستورات را در آن وارد نمایید.

CRON Jobs

هر دستور در فایل CronTabشامل ۶ بخش است که ۵ بخش ابتدایی برای تعریف تاریخ و زمان انجام دستور مورد نظر و بخش آخر برای تعریف دستوری که باید اجرا شود در نظر گرفته شده است.
بخش اول تا پنجم به ترتیب جهت تعریف دقیقه، ساعت، روز، ماه و روز هفته می باشد. و بخش ششم برای تعریف
دستور موردنظر جهت اجرا استفاده می شود که معمولا یک دستور shell است.



CRON Jobs

minute hour day_of_month month day_of_week

مثال:

در دقیقه ۳۰

30 15 * 12 5

/usr/local/bin/backup

ساعت ۳ و نیم بعد از ظهر هر جمعه در ماه دسامبر برنامه backup را اجرا کن.

پایان Linux
Part 2