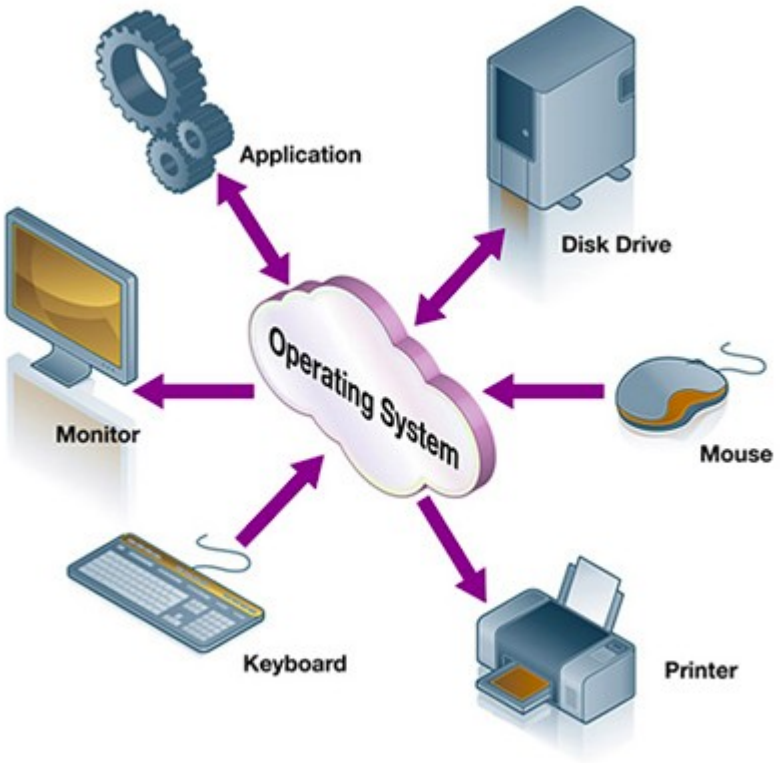
OPERATING SYSTEMS



دانشگاه زنجان

آزمایشگاه سیستم عامل

(Operating System Laboratory)

حسینعلی رحمانی دشتی

دکتر جمال محمدی

دانشگاه زنجان

اسفند ۹۳

توصیفنامه

فایل آموزشی آزمایشگاه سیستم عامل حاصل تلاش یک ترم در این درس زیرنظر **دکتر جمال محمدی در دانشگاه زنجان** می باشد. این فایل در ابتدا به دلیل پراهمیت بودن مطالبی که این استاد عزیز مورد بحث قرار دادند و بعد از آن به دلیل نیاز تعداد کثیری از دانشجویان در سراسر کشور می باشد.

مطالبی که در این فایل E-Book بحث شده‌اند دارای کاربرد فراوانی می باشد. به طور مثال مبحث **کامپایل کرنل لینوکس** در بسیاری از آموزشگاه‌های معتبر نظیر **لایتک دانشگاه شریف** تدریس می‌شود و ویدیو آموزشی آن با هزینه ای زیاد به فروش می‌رسد.

از استاد عزیز دکتر جمال محمدی برای تمامی زحماتی که کشیده‌اند نهایت سپاس را دارم. بی‌شک ایشان هم از نظر علمی و هم اخلاقی الگوی بنده خواهند بود.

از تمامی کسانی که به هر نحوی از این کتاب آموزشی استفاده می‌کنند خواهشمندم نظرات و پیشنهادها و همچنین مشکلات نگارشی و ساختاری و ... را از طریق ایمیل Saeed_Rahmani۷۱@yahoo.com در میان بگذارند.

هزینه استفاده از این کتاب تنها یک صلوات برای امام زمان (عج) و شادی روح شهدا عزیز می باشد.

Edited by : LibreOffice

فهرست

دستورات پرکاربرد در لینوکس

1- Top: این دستور برای نمایش وظیفه ها و حافظه و CPU و swap به کار می رود. (performance monitoring)

-user u : برای نمایش پروسس های مربوط به یک کاربر خاص می باشد که به جای user کاربر مورد نظر قرار می گیرد.

Z : کلید z برای نمایش و پررنگ کردن پردازش های در حال اجرا یا به اصطلاح Running می باشد.

C : کلید c برای نمایش مسیر وظیفه ها به کار می رود.

D : بصورت پیش فرض صفحه نماش در هر ۳ ثانیه به روز می شود با فشردن کلید d و تعیین بازه زمانی این مورد را می تـوانیم تغییر دهیم.

K : برای kill کردن یک پردازش اجرا می شود. که باید شماره پردازش (Process ID) را به آن ارسال نمایید (توسط دستور kill فقط میتوان پردازش های خود کابر را به پایان رساند)

h : برای نمایش help این دستور به کار می رود.

```
top - 00:23:58 up 8 min,  2 users,  load average: 0.36, 0.55, 0.28
Tasks: 118 total,   3 running, 115 sleeping,   0 stopped,   0 zombie
Cpu(s):  6.0%us,  2.0%sy,  0.0%ni, 92.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%
Mem:   1026436k total,   478668k used,   547768k free,    15692k buffers
Swap:  1646620k total,        0k used,  1646620k free,   216820k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3287 ramesh    20   0  143m  66m  19m S  0.0  6.7   0:21.38 firefox
 3752 ramesh    30  10 85676  61m  25m S  0.0  6.1   0:04.67 update-manager
 2463 root      20   0  123m  30m 9888 R  3.0  3.0   0:32.80 Xorg
 3282 ramesh    20   0 40420  23m  13m S  1.7  2.3   0:06.65 gnome-panel
 3283 ramesh    20   0 54984  18m  13m S  0.0  1.8   0:02.32 nautilus
 3413 ramesh    20   0 43748  13m  10m S  0.0  1.4   0:00.58 mixer_applet2
 3293 ramesh    20   0 27748  13m 8044 S  0.0  1.4   0:00.30 python
 3295 ramesh    20   0 27320  13m  10m S  0.0  1.4   0:00.32 update-notifier
 3677 ramesh    20   0 32800  13m 9216 S  1.0  1.3   0:00.90 gnome-terminal
```

2- id : برای نمایش id کاربر و گروه کاربر به کار میرود، یعنی نشان دادن سطح دسترسی و نوع کاربری در سرور

3- sudo : برای اجرای دستورات سطح بالا مربوط به root به کار می رود.

4- lsof : برای نمایش تمام فایل ها و پردازش های باز شده به کار میرود. فایل های باز شامل disk files, network sockets, pipes, devices و processes می باشد.

lsof -p process_id : نمایش لیست فایل های در حال استفاده به وسیله یک برنامه یا پردازش

5- tcpdump : این دستور برای network packet analyzer و packets sniffer به کار می رود. برنامه ای که برای capture و filter TCP/IP packets که دریافت و منتقل می شود.

6- dmidecode –q : این دستور برای نمایش کامل اجزای سخت افزاری به کار می رود. در این دستور کاربرد دستور sudo به چشم میخورد، برای اجرای این دستور باید به صورت زیر عمل شود :

Sudo dmidecode –q که دستور در حالت مدیر اجرا شود.

7- cat : این دستور برای نمایش اطلاعات به کار می رود برای مثال اگر cat /proc/interrupts را اجرا نماییم نمایش خطوط درخواست وقفه (IRQ) سخت افزارهای مختلف می باشد.

8- در گروه دستورات خاموش و راه اندازی مجدد سیستم دستور init 0 برای خاموش کردن و دستور reboot برای راه اندازی مجدد سیستم به کار می رود و همچنین logout برای خروج از سیستم و ورود مجدد (User Switch) به کار میرود.

9- در گروه نمایش **فایل ها و دایرکتوری ها** چندین دستور را ذکر می کنیم :

Ls : نمایش فایل ها و دایرکتوری های موجود در دایرکتوری جاری

سوئیچ l- برای نمایش جزئیات کامل به کار می رود.

سوئیچ a- برای نمایش فایل های مخفی به کار میرود.

Pwd : برای نمایش مسیر جاری به کار میرود.

Mkdir file : برای ایجاد فایلی با نام file به کار میرود.

Rm file : برای حذف file به کار میرود.

Rmdir dir : برای حذف dir که یک دایرکتوری هست به کار می رود.

10- head : دستور های head برای خواندن از ابتدا فایل های متنی بکار می رود. تنها سوئچ این دستور، n- است که تعداد خطوطی که باید از ابتدای فایل ها خوانده شود را تعیین می کند. اگر بدون این پارامتر و بصورت خطوط زیر دستور ها را اجرا کنید بصورت پیش فرض به ترتیب ۱۰ خط اول فایل را نشان می دهد.

head filename.txt

11- tail : دستور tail برای خواندن انتهای فایل های متنی بکار می رود. تنها سوئچ این دستور، n- است که تعداد خطوطی که باید از انتهای فایل ها خوانده شود را تعیین می کند. اگر بدون این پارامتر و بصورت خطوط زیر دستور را اجرا کنید بصورت پیش فرض ۱۰ خط پایانی فایل را نشان می دهد.

tail filename.txt

مثال برای ۹ و ۱۰ :

head -n5 filename.txt : پنج خط اول را نشان می دهد.

tail -n+10 filename.txt : به جز ۱۰ خط انتهایی همه محتوای فایل را چاپ می کند.

12- watch : شاید لازم داشته باشید در لینوکس یک دستور یا اسکریپتی که خودتان نوشتید را هر چند ثانیه یکبار اجرا کنید. مثلا بررسی کنید هر ۱۰ ثانیه چه کسانی به سیستم وارد شده اند.بطور معمول باید دستور w را هر ۱۰ ثانیه یک بار بصورت دستی اجرا کنید که بسیار زمان گیر است. در لینوکس دستور watch وجود دارد که یک دستور را بعنوان ورودی گرفته و آنرا هر n ثانیه یکبار اجرا می کند که کافیست بجای n زمان دلخواه تان را جایگزین کنید.

watch –n <sec> -d CMD

سوئیچ n برای تعیین زمان برای تکرار دستور است و مقدار زمان بر حسب ثانیه بجای SEC جایگزین می شود. سوئیچ d برای اینکه هر بار تغییرات را مشخص یا highlight کنیم استفاده می شود.

13- ps : برای نمایش همه پردازش های ایجاد شده توسط کاربر جاری به کار می رورد.

توسط دستور kill (PID) میتوانیم پردازش مورد نظر را به پایان برسانیم، مثل : kill 3576

و یا با استفاده از کلید های ترکیبی alt+ctrl+esc که بعد از فشردن این کلیدها می توان در لیست بر روی پردازش مورد نظر کلیک کرد و آن را kill نمود.

e- : نمایش تمامی فرایندها. این سوئیچ معادل سویچ a- نیز است.

f- : خروجی های بیشتری نشان میدهد. دستور بالا را بدون این سوئیچ استفاده کنید تا تفاوت را بینید.

U- : برای نمایش به تفکیک نام کاربری. شکل کلی آن بصورت زیر است. اگر چیزی بجای username نوشته نشود بصورت پیش فرض root در نظر گرفته میشود.

ps -u username

14- free –m : برای نمایش وضعیت حافظه اصلی به مگابایت به کار می رود.

15- lsmod : نمایش ماژول های بارگذاری شده توسط هسته

16- uname –a : نمایش مشخصات کرنل سیستم عامل

17- chmod : برای دادن مجوز دسترسی به فایل

chmod 777 index.php : **دادن مجوز کامل به فایل** index.php

۷ یعنی نمایش بیتی ۱۱۱ که نماد read و write و execute و سه بار بکار برده شده که اولی برای owner (مالک فایل) و دومی Groups (گروه ها) و Others (دیگران) می باشد.

18- zip و unzip : برای فشرده سازی و خارج کردن از فشرده سازی بکار می رود.

19- pushd و popd : اولی شاخه جاری را در استک قرار می دهد و دومی زمانی که آدرس لازم شد از استک بر می دارد.

20- gpg : برای encrypt کردن فایل و decrypt کردن فایل به کار می رود.

بصورت مقابل : gpg -c file برای encrypt و gpg file.gpg برای decrypt

21- find : برای جستجو به کار می رود. برای مثال دستور زیر برای نمایش فایل و دایرکتوری های متعلق به user1 بکار می رود.

Find / -user user1

22- who : برای نمایش کسانی که هم اکنون در سیستم login کرده اند.

23- nice : برای اجرای یک دستور با اولویت پایین تر. برای مثال nice info دستور اولویت پایین info را اجرا می کند.

24- less : هنگامی که نیاز به خواندن لاگ فایل شد می توان از این دستور استفاده کرد. (خواندن فایل های طولانی)

less -N file : **نمایش محتویات فایل ها به همراه شماره گذاری سطرهای فایل**

و همچنین دستور cat محتویات فایل متنی را نمایش داده و سپس خط فرمان را نشان می دهد.

cat -n file : نمایش محتویات فایل ها به همراه شماره گذاری سطرهای فایل

```
kghahremani-Dell-System-Vostro-3450 Desktop # cat -n iclub.txt
     1  hello
     2  hello
     3  hello
     4  iclub
     5  iclub
     6  iclub
kghahremani-Dell-System-Vostro-3450 Desktop #
```

25- du : دستور du برای دیدن فضاهای اشغال شده توسط فایل ها و دیگر اجزای داخل دایرکتوری مورد نظر است.

سوئیچ h- برای نمایش بهتر و قابل فهم تر اعداد بکار می رود.

26- history : برای نمایش تاریخچه دستورات استفاده شده می باشد. برای جستوجو در history کلید ترکیبی Control+R بزنید بعد کلمه کلیدی مورد نظر رو وارد کنین این جوری دستور قبلی که توش اون کلمه کلیدی بوده رو میاره و شما بازدن enter میتونین اونو اجرا کنین.

27- دستورات مربوط به user :

برای ساختن user چندین تا دستور وجود داره که این جا برای ساختن user دستور useradd داریم و برای پاک کردن user دستور deluser را داریم. حالا اگه بخایم که برای یک کاربر محدودیت زمانی و بعضی محدودیتای دیگه بزاریم از دستور usermod استفاده میکنیم.

groupadd : Add a Group
groupmod : Modify a Group
chgrp : Change Group
groupdel : Delete Group

28- fsck : برای چک کردن فایل سیستم به کار رفته در سیستم به کار می رود و اگر فایل سیستم آسیب دیده باشد نسبت به اشکال زدایی آن اقدام می کند.

29- مدیریت فایروال سیستم :

روشن کردن فایروال: ufw enable
خاموش کردن فایروال: ufw disable
مشاهده حالت،وضعیت،وقوانین پیاده شده در فایروال: ufw status
بستن پرت مربوطه: ufw deny port

30- crontab : برای اجرای هر دستور در زمان معینی از روز، ماه یا سال استفاده می‌کنند. Crontab مدیران سیستم را بی نیاز از نگه‌داری و اجرای دستورات تکراری روزانه می‌کند.

```
crontab -u ali -l
```

دستور فوق موجب لیست‌کردن دستوراتی را که کاربر ali برای اجرا توسط crontab تعیین کرده است می‌شود.

31- دستورات پروسس :

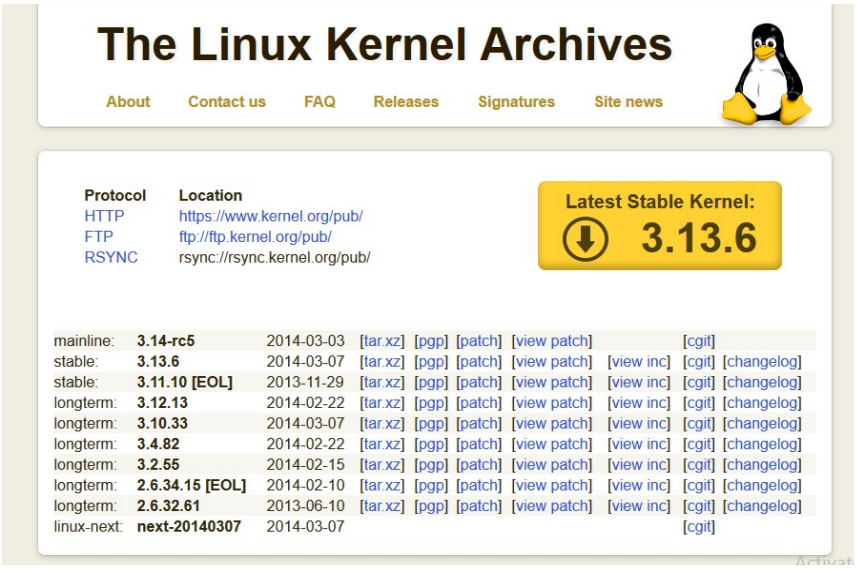
استفاده از کاراکتر & در انتهای دستور باعث فرستادن دستور در حال اجرا به پس زمینه می شود.

نکته: در صورتی که دستوری که به background فرستاده شده نیاز به ورودی داشته باشد، اجرای آن متوقف شده و تا زمانی که ورودی خود را دریافت کند منتظر میماند.

کامپایل کرنل لینوکس

در این گزارش نحوه کامپایل کرنل لینوکس را بررسی خواهیم کرد. برای شروع کار یک نرم افزار مجازی سازی نظیر **VMWare Workstation** و یا **VM Virtual Box** را نصب کرده و سپس یک **Distro** از لینوکس مثل **Mint** و یا **Ubuntu** را نصب کنید.

حالا کافیست به سایت کرنل لینوکس (http://www.kernel.org) رفته و جدیدترین نسخه را دانلود نمایید. سپس نسخه دانلود شده (در زمان تایپ این پست نسخه **linux-3.13.3** می باشد) را در ماشین مجازی خود کپی نمایید. برای مثال در صفحه دسکتاپ کپی نمایید (اگر ماشین مجازی درایوهای شما را نشان نمی دهد از طریق فلش اقدام کنید)



در این مرحله ترمینال را باز کرده و توسط دستور cd به مسیر جاری که فایل را در آنجا کپی کردید بروید.

برای مثال : cd Desktop

حال با دستور زیر فایل را از حالت فشرده خارج نمایید:

tar -xf linux-3.13.3.tar.xz

سپس مسیر جاری را به درون پوشه هسته تغییر می دهید:

cd linux-3.13.3

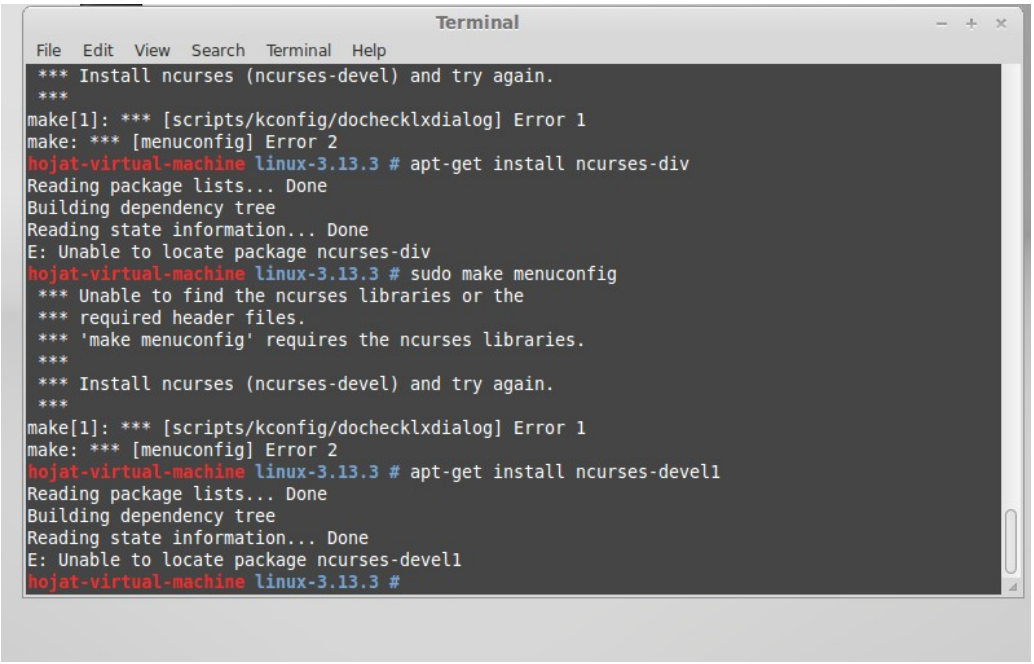
حالا دستور make menuconfig را اجرا کنید.

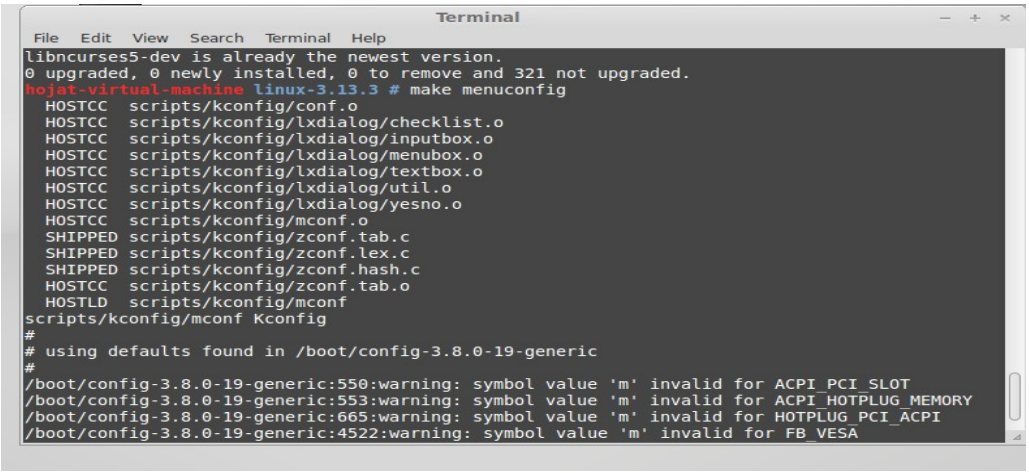
اگر در هنگام اجرای این دستور پیام خطایی مبتنی بر متن زیر نمایش داده شود مراحل زیر را انجام دهید:

```
*** Unable to find the ncurses libraries or the
*** required header files.
*** 'make menuconfig' requires the ncurses libraries.
***
*** Install ncurses (ncurses-devel) and try again.
***
make[1]: *** [scripts/kconfig/dochecklxdialog] Error 1
make: *** [menuconfig] Error 2
```

ابتدا توسط sudo –s به عنوان root وارد شوید (در هنگام ورود پسورد را باید وارد نمایید) و سپس دستور زیر را اجرا نمایید تا بسته مورد نیاز از طریق اینترنت (از صحت برقراری اتصال به اینترنت اطمینان حاصل کنید) دریافت و نصب نمایید.

apt-get install ncurses-dev

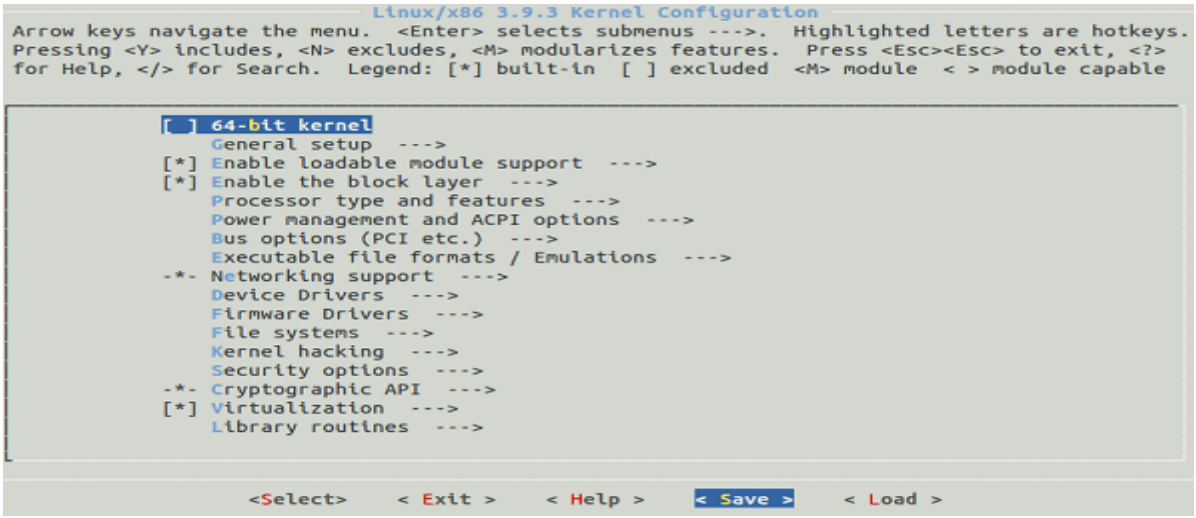




اگر عملیات با موفقیت پیش رفته باشد مراحل زیر صورت خواهد گرفت :

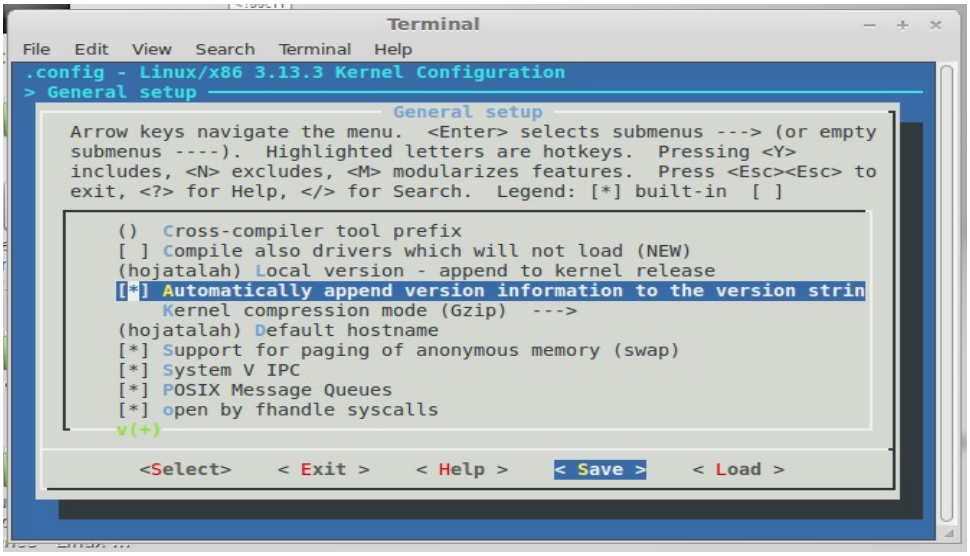
```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
libtinfo-dev
Suggested packages:
ncurses-doc
The following NEW packages will be installed:
libncurses5-dev libtinfo-dev
0 upgraded, 2 newly installed, 0 to remove and 194 not upgraded.
Need to get 328 kB of archives.
After this operation, 1,452 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://us.archive.ubuntu.com/ubuntu/ raring/main libtinfo-dev amd64 5.9-10ubuntu4 [105 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu/ raring/main libncurses5-dev amd64 5.9-10ubuntu4 [224 kB]
Fetched 328 kB in 4s (73.8 kB/s)
Selecting previously unselected package libtinfo-dev:amd64.
(Reading database ... 162746 files and directories currently installed.)
Unpacking libtinfo-dev:amd64 (from .../libtinfo-dev_5.9-10ubuntu4_amd64.deb) ...
Selecting previously unselected package libncurses5-dev.
Unpacking libncurses5-dev (from .../libncurses5-dev_5.9-10ubuntu4_amd64.deb) ...
Setting up libtinfo-dev:amd64 (5.9-10ubuntu4) ...
Setting up libncurses5-dev (5.9-10ubuntu4) ...
```

حال مجدد دستور make menuconfig را اجرا نمایید تا پس از اجرای تعدادی دستور منوی زیر نمایش داده شود :

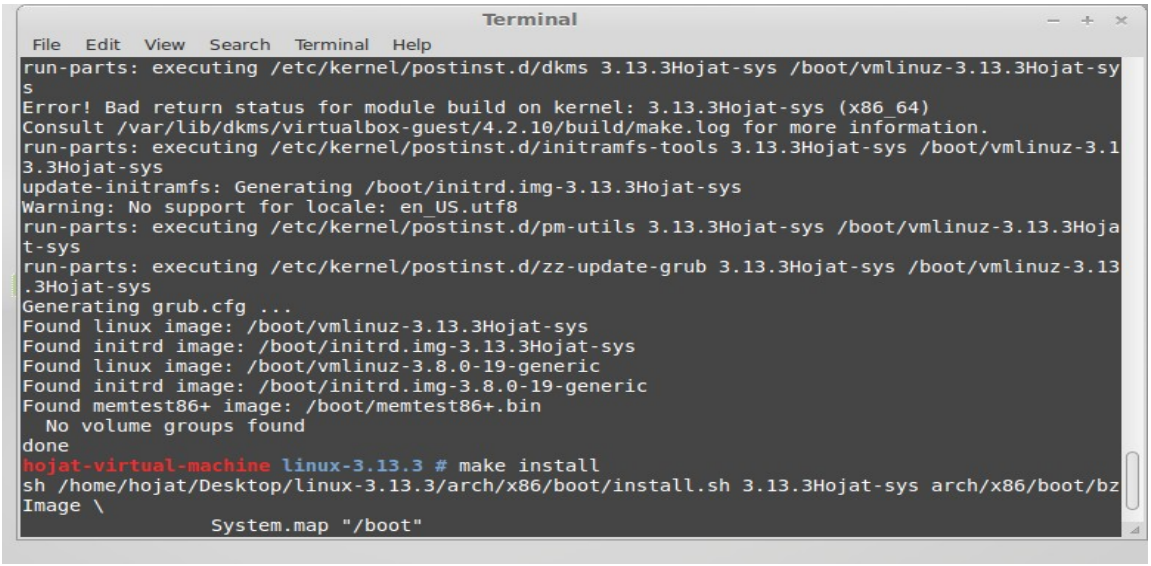


در منو ظاهر شده می توانید بعضی از گزینه های نصب را غیر فعال کنید و یا فعال کنید.

برای اینکه کرنل به نام خودتان کامپایل شود در قسمت General Setup رفته و سپس در قسمت Local Version نام خودتان را درج نمایید و گزینه پایینی آن را فعال نمایید (همچنین از بخش Host Name هم می توانید استفاده نمایید)
سپس توسط کلید Tab و جابه جایی در بخش پایینی منو میتوانید موارد تغییر یافته را ذخیره نمایید. (توسط گزینه Save)



در مرحله بعد توسط دستور make bzImage مراحل نصب را ادامه دهید. بعد مدتی که عملیات نصب ادامه پیدا کرد توسط دستور make modules، ماژول ها را فراخوانی نمایید و سپس توسط دستور make modules_install ماژول ها را نصب نمایید.
پس از انجام مراحل بالا در یک بازه زمانی حدود ۳۰ دقیقه ای توسط دستور make install کرنل را نصب نمایید و پس از آن مراحل انجام کار به پایان می رسد.



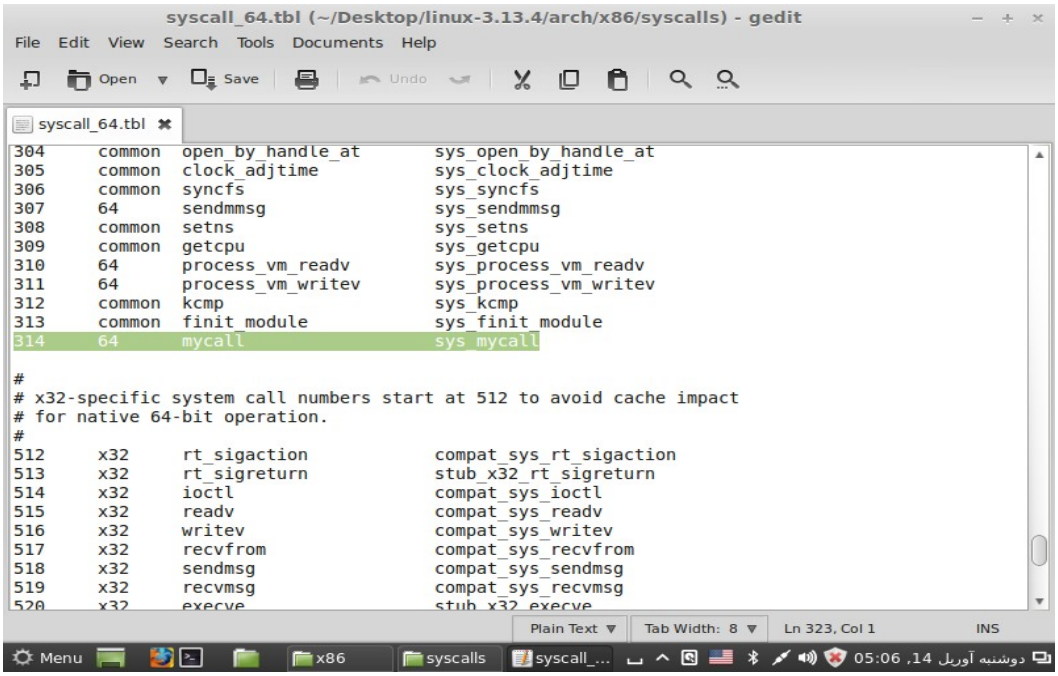
نحوه اضافه کردن فراخوان سیستمی در لینوکس

در این آموزش نحوه قرار دادن یکی فراخوانی سیتمی را بررسی خواهیم کرد. برای شروع کار یک نرم افزار مجازی سازی نظیر **VMWare Workstation** و یا **VM Virtual Box** را نصب کرده و سپس یک **Distro** از لینوکس مثل **Mint** و یا **Ubuntu** را نصب کنید.

حالا کافیست به سایت کرنل لینوکس (http://www.kernel.org) رفته و جدیدترین نسخه را دانلود نمایید. سپس نسخه دانلود شده (در زمان تایپ این پست نسخه **linux-3.13.3** می باشد) را ماشین مجازی خود کپی نمایید. برای مثال در صفحه دسکتاپ کپی نمایید (اگر ماشین مجازی درایوهای شما را نشان نمی دهد از طریق فلش اقدام کنید)

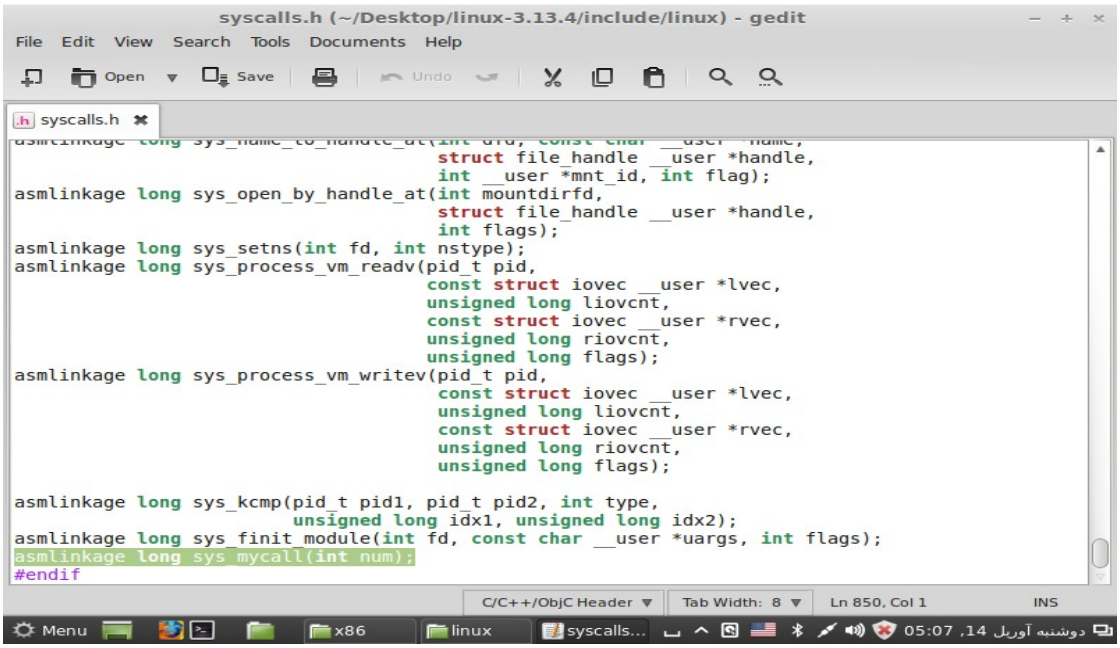
در شاخه https://www.kernel.org/pub/linux/kernel/ میتوانید نسخه های مختلف را دانلود کنید.
به طور مثال نسخه linux-3.13.4 که در این آموزش استفاده شده است در شاخه V3.0 قرار دارد. بعد از اینکه هسته را دانلود کرده و در مسیری استخراج (Extract) کرده و مراحل زیر را انجام می دهیم. ابتدا به مسیر linux-<some version>/arch/x86/syscalls/ را باز کرده و فایل syscall_64.tbl را باز کرده و در انتهای فراخوانی سیستم های معرفی شده، فراخوان سیستمی خود را معرفی می نماییم.

در ستون اول شماره فراخوان سیستمی (در اینجا ۳۱۴) و در ستون بعد چون هسته ما بصورت ۶۴ بیتی می خواهد نصب شود عدد ۶۴ را درج می نماییم و در ستون سوم نام فراخوان سیستمی (به طور مثال mycall) و در ستون چهارم به ابتدای نام فراخوان سیستمی sys را اضافه کرده (sys_mycall) و بعد از آن فایل مورد نظر را ذخیره می نماییم.



در مرحله بعد مسیر linux-<some version>/include/linux/ رفته و سپس فایل syscall.h را باز کرده و در پایان فایل قبل از #endif هدر تابع فراخوان سیستمی را معرفی می نماییم، بطور مثال ما هدر زیر را درج می نماییم :

asmlinkage long sys_mycall(int num);

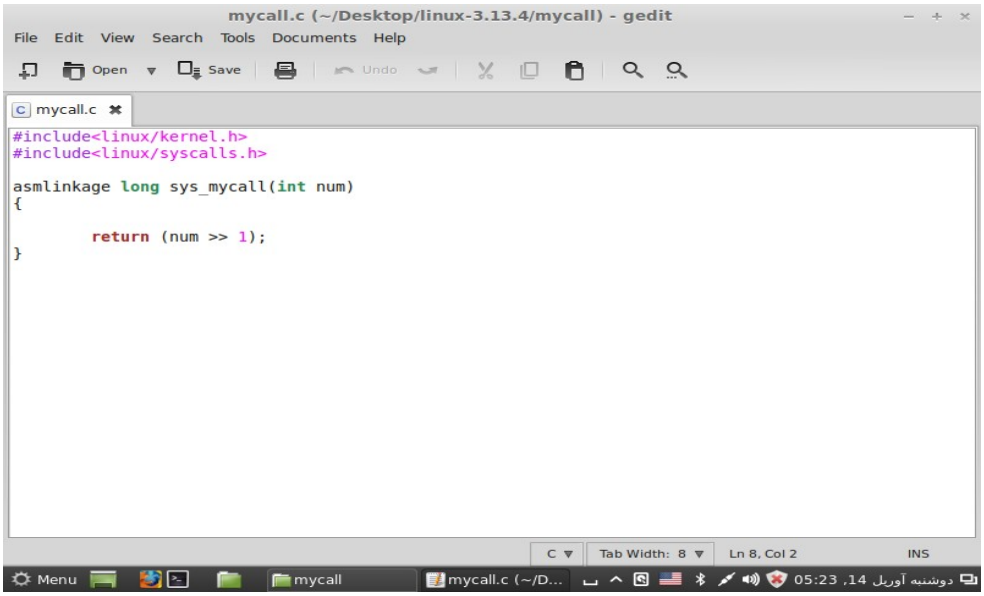


و فایل مورد نظر را ذخیره کرده و در مرحله بعد در مسیر ریشه فایل هسته یک پوشه برای فراخوان سیستمی مورد نظر ایجاد می نماییم.

Linux-<some version>/mycall

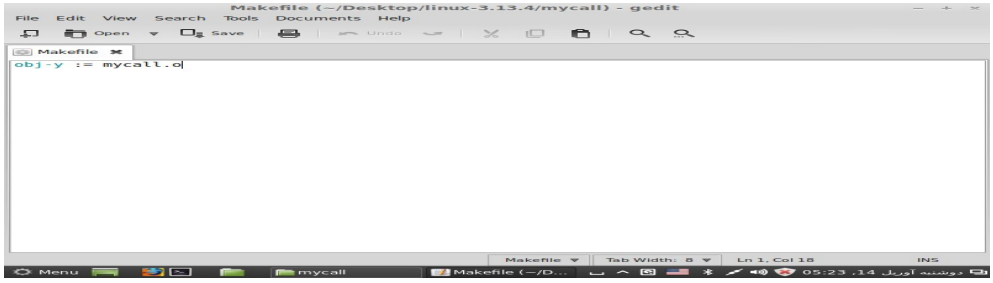
سپس در این پوشه یک فایل متنی برای نوشتن کدهای برنامه سی ایجاد می نماییم و دستورات زیـر را در آن درج مـی نمـاییم و سـپس فایـل را ذخیره می نماییم.

```
#include<linux/kernel.h>
#include<linux/syscalls.h>
asmlinkage long sys_mycall(int num)
{
        return ( num >> 1);
}
```



و در همان مسیر و در کنار فایل برنامه مورد نظر فایل جدیدی ایجاد کرده و نام آن را Malefile قرار مـی دهیـم و سـپس در آن دسـتور زیـر را تایپ کرده که objectای برای فراخوان سیستمی ما می باشد.

obj-y := mycall.o



سپس به ریشه پوشه کرنل خود رفته و فایل Makefile را باز کرده و به دنبال متنی با عنوان core-y می گردیم و پس از یـافتن آن در عبـارت رو به روی آن آدرس (نام پوشه ایجاد شده برای فراخوان سیستمی خودمان) را اضافه می نماییم.

core-y := usr/ mycall/

حال فایل را ذخیره کرده و سپس شروع به کامپایل هسته می نماییم.

بعد از کامپایل و شروع به کار مجدد سیستم به عنوان یک کاربر عادی وارد شده و یک برنامه نوشته در سطح کاربر کـه از فراخـوان سیسـتمی مـا استفاده نماید.
در زیر یک نمونه برنامه به زبان سی نوشته شده است که از فراخوان سیستمی ما استفاده می نماید.
برای اینکار یک فایل متنی در دسکتاپ ایجاد کرده و دستورات زیر را در آن وارد می نماییم و نام فایل را SCI.c قرار می دهیم :

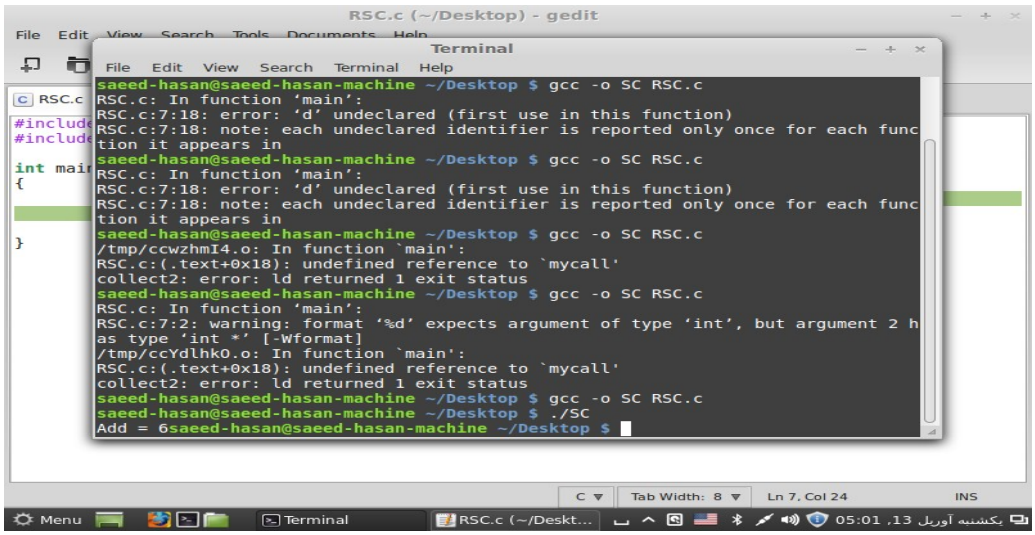
```
#include <unistd.h>
#include <stdio.h>
int main()
{
        int x = syscall(314, 20);
        printf("your number division by two :%d", x);
        return 0;
}
```

سپس توسط دستور زیر فایل را کامپایل کرده و به یک فایل اجرایی تبدیل می نماییم:

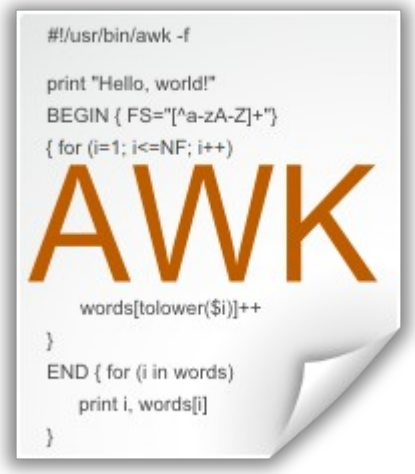
```
gcc -o SC SCI.c
```

سپس توسط دستور زیر آن را اجرا می نماییم :

```
./SC
```



AWK


#!/usr/bin/awk -f
print "Hello, world!"
BEGIN { FS="[^a-zA-Z]+"}
{ for (i=1; i<=NF; i++)
AWK
words[tolower($i)]++
}
END { for (i in words)
print i, words[i]
}

AWK چیست ؟

اسکریپت AWK یک زبان برای پردازش فایل های متنی است، هر خط فایل متنی یک رکورد محسوب می شود و هر خط شـامل چنـدین فیلـد می باشد. در حقیقت نام این زبان برنامه نویسی بر اساس حرف اول نام نویسندگان آن **Alfred Aho, Peter Weinberger and Brian Kernighan** نام گذاری شده است.

این زبان ساختاری شبیه به زبان برنامه نویسی سی دارد و برای برنامه نویسانی که با این زبان کار کرده اند بسیار آسان می باشد.

ساختار کلی این زبان به صورت زیر می باشد:

```
BEGIN { }
   pattern { actions }
END   { }
```

در این ساختار کلی دستورات موجود در بخش pattern به ازای تک تک رکوردهای فایل مورد نظر ما اجرا خواهند شد.
مانند یک حلقه foreach عمل خواهد کرد. دستورات Begin در شروع کار اجرا خواهد شد و دستورات END در پایان اجرای برنامه به اجـرا در خواهد آمد.

برخی معرفی ها درباره گرامر این زبان اسکریپتی در زیر آورده شده است :

- $i : $ در حقیقت به ستون های یک فایل اشاره می کند، به طور پیش فرض جداساز ستون ها یک کاراکتر خـالی (blank) در نظـر گرفته می شود و i به شماره آن ستون اشاره می نماید.
- به تعریف متغیر نیازی نیست و مانند زبان پایتون کافیست نام متغیر را بنویسید و به طور پیش فرض مقادیر متغیرها ۰ می باشد.
- علامت ~ مانند like در دستورات Sql عمل می کند و نگاه می کند رشته مورد نظر در بخشی از رشته های فایل آورده شده است یـا خیر، البته عبارت مورد نظر باید در بین دو علامت / / قرار بگیرد.
- آرایه به صورت X[] تعریف می شود که جای اندیس می توان مقادیر ستونی را در اندیس آن قرار داد.
- علامت شارپ (#) برای درج توضیحات به کار میرود.

برای کار با این زبان و تایپ دستورات کافیست یک فایل متنی باز کزده و پسوند آن را به awk تغییر دهید.

حال می توانید دستورات را با الگوهای ذکر شده بنویسید.

فرض می کنیم فایلی از log یک squid را در اختیار داریم می خواهیم اطلاعاتی را از آن استخراج کنیم.

برخی سوالات و پاسخ های داده شده به آن توسط زبان AWK که برای پردازش یک فایل و استخراج اطلاعات صورت گرفته است:

این دستورت در بخش **Pattern** خواهد آمد.

1. نام کاربری shahrdari مجموعا چندبار تکرار شده است ؟

if ($4 == "shahrdari") #1

Count++

این دستور در ستون چهار هر رکورد نگاه میکند و اگر مقدار Shahrdari موجود بود به متغییر Count یک واحد اضافه می کند.

2. نام کاربری Tehran چه مقدار ترافیک ارسال کرده است؟

if ($4=="tehran") #2

ttsum += $10

3. چه نام کاربری به وب سایت www.afkarnews.ir مراجعه کرده است ؟

if ($12 ~ /www.afkarnews.ir/) #3

arrAfkarUsers[$4] = 1

در این دستور از عملگر مشابه بودن رشته استفاده شده است و همینطور از آرایه که هر نام کاربری را یم اندیس در آرایه قرار می دهد و توسط مساوی یک از درج نام کاربری های تکراری جلوگیری می کند.

4. چه میزان ترافیک از وب سایت www.afkarnews.ir دریافت شده است؟

if ($12 ~ /www.afkarnews.ir/) #4

afkGetSum += $7

ستون ۱۲ به آدرس سایت ها و ستون ۷ به ترافیک های دریافتی اشاره می کند.

5. در تاریخ September 3 چه کسانی از سیستم استفاده کرده اند؟

```
if ($1 == "September" && $2 == "3") #5
        sepUsers[$4] = 1
```

6. بیشترین میزان ترافیک دریافتی / ارسالی متعلق به کدام نام کاربری بوده است؟

```
{#6
    if ($10 > longestTraffic) {
        longestTraffic = $10
        longestTrafficUser = $4
    }
    if ($7 > longestTraffic) {
        longestTraffic = $7
        longestTrafficUser = $4
    }
    }
```

7. نام کاربری ahmadi از چه آی پی هایی استفاده کرده است؟
8. نام کاربری ahmadi به چه وب سایت هایی مراجعه کرده است؟

```
if ($4 == "Ahmadi") {
        ahmadiIPs[$8] = 1 #7
        ahmadiSites[$12] = 1 #8
    }
```

9. بیشترین وب سایت مراجه شده چیست؟

```
sitesVisits[$12]++ #9
```

این دستور سایت های هر رکورد را در آرایه ذخیره کرده و اگر تکراری در آدرس سایت ها دیده شود به اندیس آن یک واحد اضافه می نماید.

10. بیشترین ترافیک ارسالی به چه وب سایتی بوده است؟

```
if ($10 > maxPostTraffic) { #10
        maxPostTraffic = $10
        maxPostTrafficSite = $12
    }
```

در قسمت END دستورات زیر را برای نمایش خروجی های هر دستور تایپ کنید، توجه کنید که دستورات این بخش تنها یک بار اجرا می شود.

```
END {
    print "#1", Count
    print "#2", ttsum
    printf "#3 "
    for (key in arrAfkarUsers) {
        printf "%s    ", key
    }
    printf "\n"
    print "#4", afkGetSum
    printf "#5 "
    for (key in sepUsers) {
        printf "%s    ", key
    }
    printf "\n"
    print "#6", longestTrafficUser
    printf "#7 "
    for (key in ahmadiIPs) {
        printf "%s    ", key
    }
    printf "\n"
    print "#8 Ahmadi sites:"
    for (key in ahmadiSites) {
        print "    -", key
    }
    for (key in sitesVisits) {
        if (sitesVisits[key] > maxSiteVisits) {
            maxSiteVisits = sitesVisits[key]
            maxSiteVisitsName = key
        }
    }
    print "#9", maxSiteVisitsName
```

```
        print "#10", maxPostTrafficSite
}
```

اجرای برنامه :

برای اجرا کافیست فایل کد و فایل مورد پردازش را در یک مسیر قرار داده و به آن مسیر رفته و ستور زیر را تایپ نمایید:

```
awk –f source.awk access.log
```

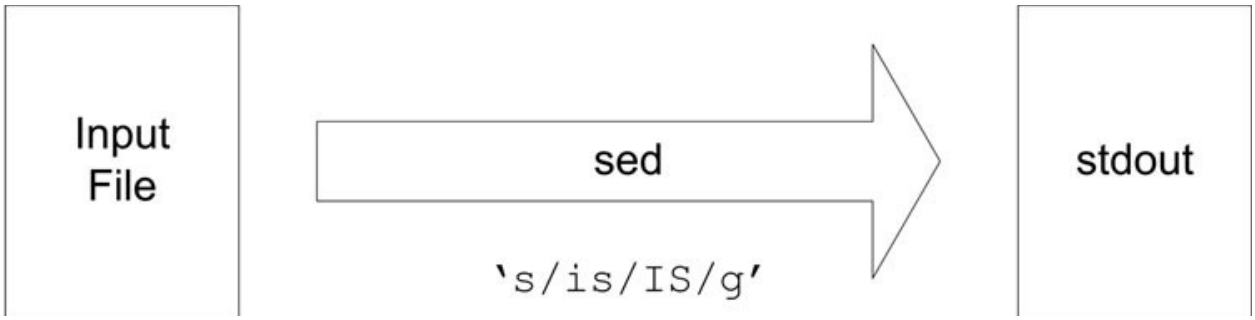
بخش access.log به فایل مورد پردازش اشاره می نمایید و بخش source.awk به سورس برنامه نوشته شده اشاره می کند.

SED

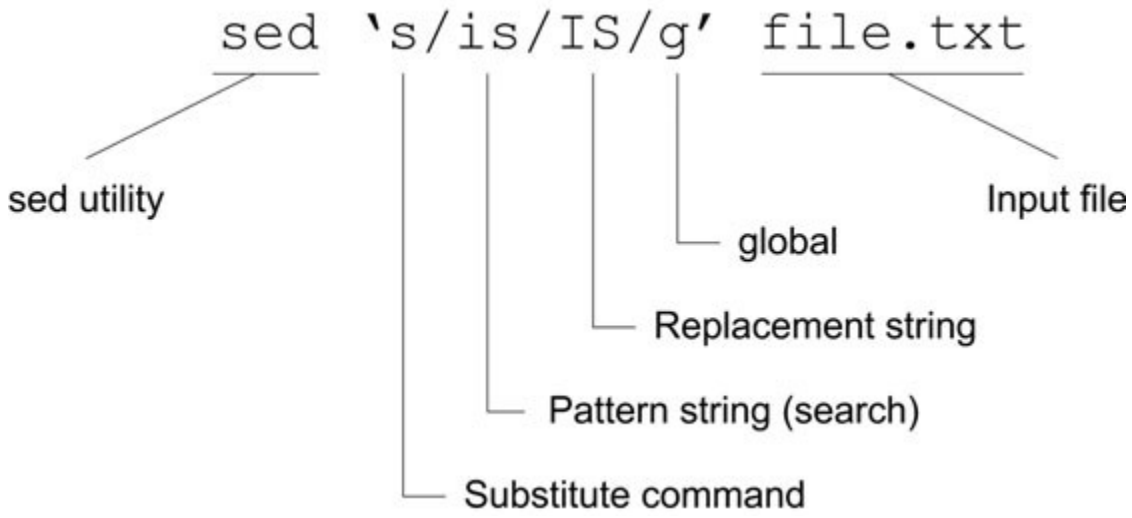
SED **چیست ؟**

SED یکی از دستورهای پایه‌ای و اصلی سیستم عامل های شبیه یونیکس (لینوکس) است. نام این دستور از ترکیب Stream Editor آمده‌است که به معنای ویرایشگر استریم یا همان جریانی است. این دستور متن ورودی را تجزیه و تحلیل (پردازش) می‌کند و با پیاده‌سازی یک زبان برنامه‌نویسی، متن ورودی را به متنی با ساختار دلخواه کاربر تبدیل می‌کند. این دستور متن ورودی را خط به خط (به صورت ترتیبی) می‌خواند و تغییرهای لازم را که از طریق خط فرمان (یا توسط اسکریپت SED) به دستور داده شده است را بر متن اعمال می‌کند و سپس خط (خطوط) را خروجی می‌دهد. این دستور را لی ای. مک ماهون در آزمایشگاه‌های بل در سال‌های ۱۹۷۳ تا ۱۹۷۴ برای سیستم‌عامل یونیکس نوشت که در حال حاضر در بیشتر سیستم عامل ها موجود است و کاربردهای فراوانی دارد.

ساختار کلی دستور SED :



ابزار SED بعد از اعمال تغییرات بر روی فایل ورودی فایل خروجی را در ترمینال همراه با تغییرات نمایش میدهد.



معرفی اجزای الگوی استفاده شده در حالت کلی:

SED: کلمه کلیدی برای ابزار مربوطه.
S: نشان دهنده و مشخص کننده عمل جایگزینی در فایل است.
is: الگوی مورد نظر برای جستجو در فایل.
IS: رشته مورد نظر برای جایگزینی.
g: بیان گر این است که عمل جستجو و جایگزینی در هم فایل انجام گیرد (از ابتدا تا انتها).
file.txt: مشخص کننده فایل ورودی است.

SED به طورپیش فرض خروجی را در ترمینال نمایش میدهد که با استفاده از دستور زیر و سوئیچ –i میتوان تغییرات را در فایل اصلی اعمال کرد.

```
sed –i ‘s/is/IS/g’ input file.txt
```

و یا با استفاده از دستور زیر خروجی را در فایل جداگانه ای ذخیره کرد.

```
sed –e ‘s/is/IS/g’ input file.txt > file.txt
```

مثال هایی در زمینه SED:

همه مثال ها از روی فایل log است که این فایل، log ابزار squid می باشد. این فایل شامل ستون هایی برای ماه، روز، نام کاربری، ساعت اتصال، آدرس IP، ترافیک دریافتی و ترافیک ارسالی، آدرس وب سایت مراجعه شده و ... می باشد.

- تمامی zanjan ها را به Tabriz تغییر دهید.

```
sed -e 's/tehran/zanjan/' access.log
```

از دستور s برای جایگذاری استفاده می کنیم، در اینجا فقط کلمه هایی با کلمه zanjan جایگزین میشوند که Tehran باشد، و با حرف کوچک شروع شده باشد.

- تمامی انواع - zanjan, zanjAn ، ZanjAn را به ZANJAN تغییر دهید.

sed حساس به حروف کوچک و بزرگ می باشد برای غیر فعال کردن آن باید از دستور /I استفاده کنیم تا تمام zanjan ها به هر صورتی که نوشته شده را با ZANJAN جایگزین کند.

```
sed 's/zanjan/ZANJAN/I' access.log
```

- خط ۲۰۰ فایل را نمایش دهید.

```
sed –n  '200  p' access.log
```

ابزار sed به طور پیش فرض تمام خطوط را چاپ می کند با استفده از پارامتر n- از این کار جلوگیری کرده و خطوطی که کاربر تعیین می کند را چاپ می کند.

- خط ۷۶۲ را حذف کنید.

```
sed '762 d' access.log
```

از دستور d برای حذف استفاده می شود.

- تمامی خطوط شامل shahrdari را حذف کنید.

```
sed  '/shahrdari/d'  access.log
```

- شماره خط را در اول هر سطر اضافه کنید.

```
sed '=' access.log
```

برای به دست آوردن شماره سطر از = استفاده می کنیم.

- تعداد خطوط را چاپ کنید.

```
sed -n '$=' access.log
```

در اینجا از عملگر مساوی که شمار خط جاری را ذخیره کرده است استفاده شده و با علامت $ نشان داده ایم که شماره آخرین خط را نمایش دهد.

- تمامی whitespace ها (space, enter, tab) را حذف کنید.

```
sed -r 's/\s+//g' access.log
```

این دستور فاصله های خالی تک کاراکتری و تب ها را از فایل حذف میکند و بدون سوئیچ **r**- کار نمیکند.

- ۵ فاصله در ابتدای هر خط اضافه کنید.

```
sed 's/^/     /' access.log
```

- تمام shahrdari ها را با ALI جایگزین کنید. فقط در خطوطی که شامل POST هستند.

```
sed -e '/POST/{s/shahrdari/ALI/g}' access.log
```

- تمام shahrdari ها را با ALI جایگزین کنید. به جز خطوطی که شامل POST هستند.

```
sed -e '/POST/!{s/shahrdari/ALI/g}' access.log
```

برای معکوس کردن از ! استفاده می کنیم.

- کاما را به دشته های عددی اضافه کنید: مثلا "۱۲۳۴۵۶۷" به "۱,۲۳۴,۵۶۷" تغییر کند.

```
sed -e :a -e 's/\(.*[0-9]\)\([0-9]\{3\}\)/\1,\2/;ta' access.log
```

- **یک خط خالی بعد از هر ۵ خط اضافه کنید.**

```
sed -e '5a\ ' access.log
```

- **خطوط شامل POST را چاپ کنید.**

```
sed -n -e '/POST/p' access.log
```

- **خطوط تکراری را حذف کنید.**

```
sed -e '$!N; /^\(.*\)\n\1$/!P; D' access.log
```

- **خطوط حاوی کمتر از ۱۶۰ کاراکتر را نمایش دهید.**

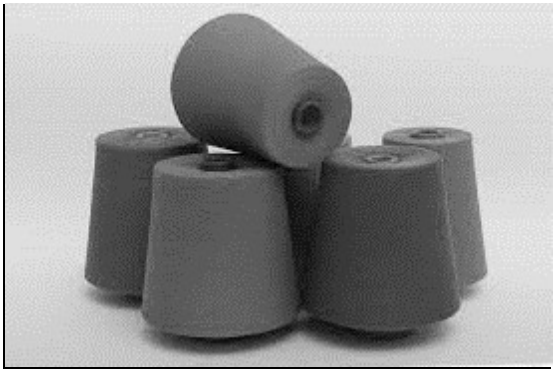
```
sed -n '/^.\{160\}/!p' access.log
```

- **یک قابلیت جالب توجه از کاربرد** SED **:**

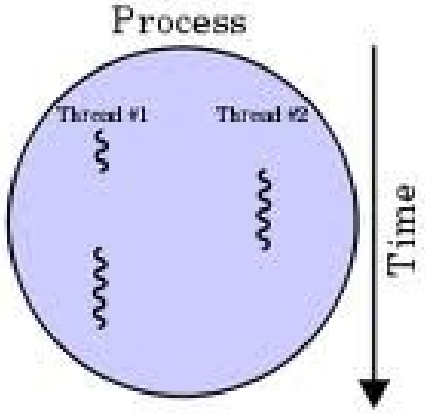
با توجه به قدرت پردازش این ابزار می توان از آن در جداسازی متون از گرامر زبان ها برنامه نویسی مظیر HTML استفاده کرد. به عنوان مثال با جستجوی عبارت use of sed for replace syntax in java در موتورهای جستجو به اطلاعاتی در مورد جایگذاری بعضی دستورات در بین دستورات زبان برنامه نویسی جاوا توسط ابزار SED دست پیدا خواهید کرد.

Thread

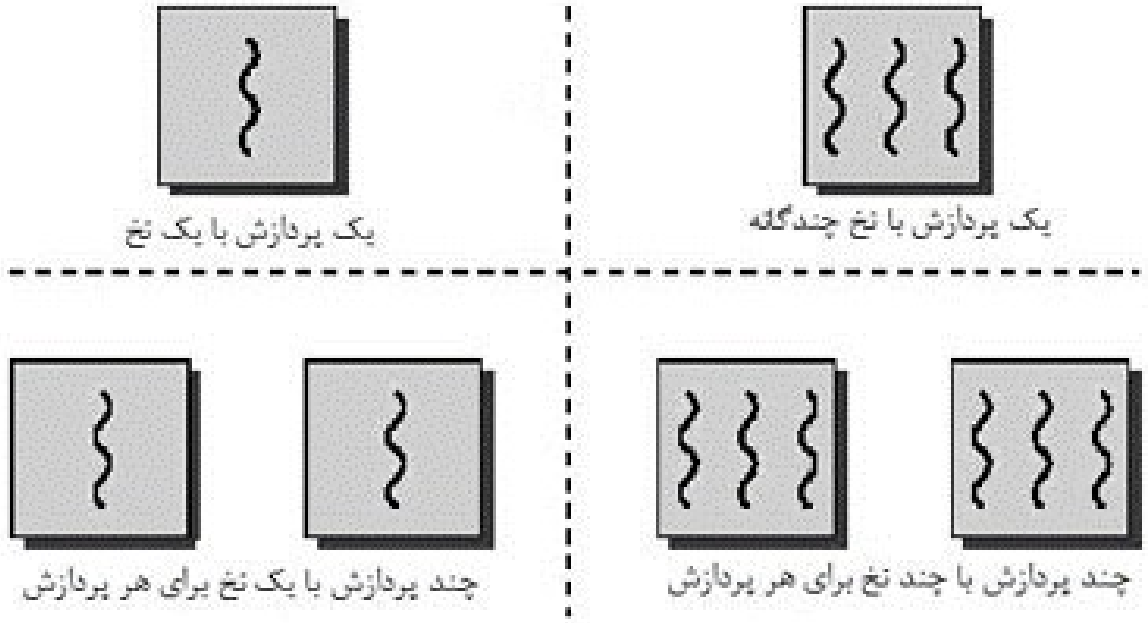
Thread **(نخ) چیست ؟**



Threadیا در اصطلاح "نخ" پراسس های کوچکی هستند که هر کدام یک هدف انجام می دهند و در نهایت پس از پایان یافتن اجرای مجموعه thread ها یک برنامه یا یک پراسس اصلی پایان پیدا می کند. از thread برای انجام کارهای موازی همزمان استفاده میشود. اکثر برنامه هایی که ما می نویسیم فقط یک thread دارند که همون پراسس اصلی ماست و با پایان یافتن آن، برنامه هم به پایان میرسد.



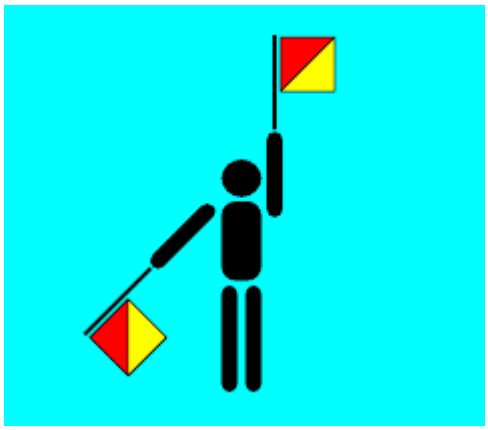
- هر نخ شامل مجموعه ای از دستورات است.
- نخها قسمتی از یک برنامه هستند و یک فرآیند در حال اجرا می تواند یک یا چند نخ داشته باشد.



دلیل استفاده از نخ ها

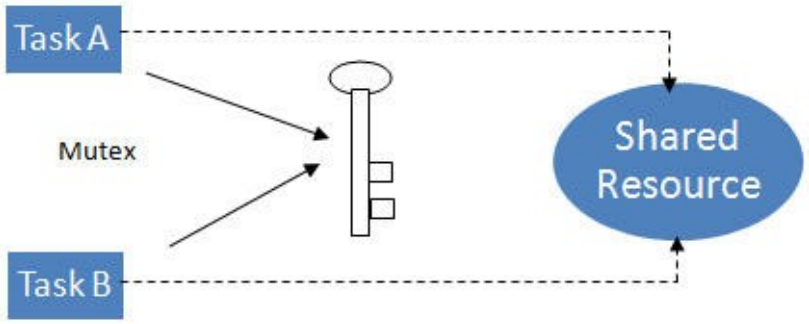
در حالتی که در یک فرآیند چند نخ همزمان اجرا شوند، امکان فراهم آوردن پردازش موازی و همچنین کار با برنامه های بزرگ ساده تر می شود.

Semaphor



سمافور چیست ؟

در علم رایانه نشانبر یا سمافور (به انگلیسی Semaphore) به متغیری گفته می‌شود که در محیط های همروند برای کنترل دسترسی فرایندها به منابع مشترک به کار می‌رود. سمافور می‌تواند به دو صورت دودویی (که تنها دو مقدار صحیح و غلط را دارا است) و یا شمارنده اعداد صحیح باشد. از سمافور برای جلوگیری از ایجاد وضعیت رقابتی میان فرایندها استفاده می‌گردد. به این ترتیب، اطمینان حاصل می‌شود که در هر لحظه تنها یک فرایند به منبع مشترک دسترسی دارد و می‌تواند از آن بخواند یا بنویسد(انحصار متقابل)



سمافورها اولین بار به‌وسیله دانشمند علوم رایانه هلندی، ادسخر دیکسترا معرفی شدند. امروزه به طور گسترده‌ای در سیستم عامل‌ها مورد استفاده قرار می‌گیرند.

اصل اساسی این است که دو یا چند فرایند می‌توانند به وسیله سیگنال‌های ساده با یکدیگر همکاری کنند. هر فرایند را می‌توان در نقطه خاصی از اجرا متوقف نموده، و تا رسیدن سیگنال خاصی از اجرای آن جلوگیری نمود. برای ایجاد این اثر، از متغییرهای خاصی به نام سمافور استفاده می‌گردد.

هر فرایندی که بخواهد به منبع مشترک دسترسی داشته باشد، اعمال زیر را انجام خواهد داد:

۱. مقدار سمافور را بررسی می‌کند.
۲. در صورتی که مقدار سمافور مثبت باشد، فرایند می‌تواند از منبع مشترک استفاده کند. در این صورت، فرایند یک واحد از سمافور می‌کاهد تا نشان دهد که یک واحد از منبع مشترک را استفاده نموده است.
۳. در صورتی که مقدار سمافور صفر یا کوچکتر از صفر باشد، فرایند به خواب می‌رود تا زمانی که سمافور مقداری مثبت به خود بگیرد. در این حالت فرایند از خواب بیدار شده و از مرحله یک شروع می‌کند.

هنگامی که فرایند کار خود را با منبع تمام نمود، یک واحد به سمافور اضافه می‌گردد. هر زمان که مقدار سمافور به صفر و یا بیشتر برسد، یکی از فرایند(هایی) که به خواب رفته به صورت تصادفی یا به روش FIFO توسط سیستم‌عامل بیدار می‌شود. در این حالت بلافاصله فرایند بیدار شده منبع را در دست می‌گیرد و مجدداً پس از اتمام کار یک واحد از سمافور کم می‌شود. اگر مقدار سمافوری صفر باشد و چند فرایند بلوکه شده در آن وجود داشته باشد، با افزایش یک واحدی سمافور، مقدار سمافور همچنان صفر باقی می‌ماند اما یکی از فرایندهای بلوکه شده آزاد می‌شود.

برنامه ای در زیر آورده شده است یک نمونه مثال از اجرای سمافور با نخ ها می باشد، حافظه مترکی در نظر گرفته است که در آن نوشته شده و تابع دیگری آن را می خواند.

```
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>
#include <semaphore.h>

void handler_a(void *ptr);
void handler_b(void *ptr);
sem_t mutex_a;
sem_t mutex_b;
int counter = 0;
int sum1 = 0, sum2 = 0;

int main()
{
        int i[2];

        pthread_t thread_a;
        pthread_t thread_b;

        i[0] = 0;
```

```
        i[1] = 1;

        sem_init(&mutex_a, 0, 1);
        sem_init(&mutex_b, 0, 1);

        pthread_create(&thread_a, NULL, (void *)&handler_a, (void *)&i[0]);
        pthread_create(&thread_b, NULL, (void *)&handler_b, (void *)&i[1]);

        pthread_join(thread_a, NULL);
        pthread_join(thread_b, NULL);

        sem_destroy(&mutex_a);
        sem_destroy(&mutex_b);

        exit(0);
}



void handler_a(void *ptr)
{
        int x, i;
        x = *((int *)ptr);
        printf("Thread %d:enter critical region...\n", x);
        printf("Thread %d:printing counter value...%d \n", x, counter);
        for (i = 0; i<100; i++)
        {
                sem_wait(&mutex_b);


                counter = rand();

                sum1 = sum1 + counter;
                sem_post(&mutex_a);

        }
        printf("Thread %d:printing sum1 value...%d \n", x, sum1);
        printf("Thread %d: Exiting critical region...\n", x);
        pthread_exit(0);
}
void handler_b(void *ptr)
{
        sem_wait(&mutex_a);

        int x, k;
        x = *((int *)ptr);
        printf("Thread %d:enter critical region...\n", x);
        printf("Thread %d:printing counter value...%d \n", x, counter);
        for (k = 0; k<100; k++)
        {
```

```
                sem_wait(&mutex_a);

                sem_post(&mutex_b);
                sum2 = sum2 + counter;


        }
        printf("Thread %d:printing sum2 value...%d \n", x, sum2);
        printf("Thread %d: Exiting critical region...\n", x);
        pthread_exit(0);
}
```

پایان