

آموزش LPIC-1 (Exam 101,102)

مروری بر دوره

مدرک LPIC-1 اولین مدرک از مجموعه مدارک حرفه ای سطح بندی شده LPI لینوکس می باشد. این مدرک صلاحیت داوطلب در زمینه اداره کردن امور مرتبط با تعمیر و نگهداری از طریق خط فرمانی (command line)، نصب و پیکربندی یک کامپیوتر برخوردار از سیستم عامل لینوکس و نهایتا تنظیمات اولیه مرتبط با شبکه را تایید می کند.

[دوره آموزش لینوکس](#) در بین متخصصان فناوری اطلاعات (IT) از اهمیت و جایگاه ویژه ای برخوردار است. دلیل استقبال از [دوره های Linux](#) این است که لینوکس همچنان در حوزه رایانه جزو ابر قدرت ها و نام های شناخته این بازار به حساب می آید. طبق گزارشی که W3Techs برای سال ۲۰۱۹ میلادی منتشر کرده است بیش از ۵۰ درصد سهم بازار مربوط به وب سرور، به Linux اختصاص دارد.

در این دوره آموزشی سعی شده است در کنار مالتی مدیاهای آموزشی ناگفته های لینوکس را در قالب کتاب ارائه بدهیم، که به صورت کاملا عملی و سناریو محور می باشد. در واقع این کتاب مکمل مالتی مدیاهای آموزشی LPIC-1 Exam 101,102 است.

این یک کتاب تئوری نیست بلکه در این کتاب صدها مثال کاربردی به همراه تصویر قرار داده شده است تا شما خودتان آنها را اجرا کنید و نحوه کار با دستورات لینوکسی را یاد بگیرید.

فهرست مطالب

فصل اول : آشنایی با معماری سیستم

- تشخیص و پیکربندی سخت افزار.
- راه اندازی سیستم.
- تغییر سطح اجرایی، خاموش کردن و راه اندازی مجدد.

فصل دوم : مفاهیم نصب سیستم عامل و نرم افزارها

- طراحی ساختار دیسک سخت
- نصب نرم افزار راه انداز سیستم ((Boot Manager
- پیکربندی کتابخانه ها در سیستم

- استفاده از مدیریت بسته های دبیان

- استفاده از مدیریت بسته های ردهت

فصل سوم : دستورات کاربردی لینوکس

- کار در ترمینال
- پردازش جریان های متنی با فیلترها
- مدیریت فایلها
- استفاده از جریانها، لوله ها و تغییر مسیرها
- ساختن، مانیتور کردن و حذف کردن پروسه ها

- تغییر اولویت اجرای پروسه ها

- جستجو در فایل‌های متنی با الگوهای متنی

- ویرایش فایل‌ها با ادیتور vi

فصل چهارم : دستگاه ها، ساختار فایل‌ها، استاندارد سلسله مراتبی فایل‌ها

- ایجاد پارتیشن و فایل های سیستمی

- کنترل سلامت فایل سیستم ها

- mount و unmount کردن فایل سیستم ها

- مدیریت ظرفیت فضای دیسک

- مدیریت مجوز فایل‌ها و مالک فایل‌ها

- ایجاد و تغییر لینک های سخت و symbolic لینک ها

- یافتن و قراردادن فایل‌ها در جایگاه صحیح

فصل پنجم : پوسته ها، اسکریپت ها و مدیریت داده ها

- دلخواه سازی و استفاده از محیط shell

- دلخواه سازی یا نوشتن اسکریپت های ساده

- مدیریت داده ها با SQL

فصل ششم : محیط کاربری و دسکتاپ

- نصب و پیکربندی محیط گرافیکی X11

- تنظیمات display manager

- امکانات دسترسی (معلولین)

فصل هفتم : وظایف مدیر سیستم

- مدیریت کاربران و گروه ها و فایل های مرتبط

- خودکار کردن وظایف سیستم

- محلی سازی و بین المللی سازی

فصل هشتم : سرویس های پایه ای سیستم

- مدیریت زمان

- مدیریت سیستم ثبت گزارش (log)

- مبانی انتقال نامه های الکترونیکی (Mail Transfer Agent)

- مدیریت چاپگرها

فصل نهم : مبانی شبکه

- مبانی پروتکل های اینترنت

- مبانی تنظیمات شبکه

- مبانی رفع اشکال شبکه

- استفاده از سیستم دامنه

فصل دهم : امنیت

- تنظیمات مقدماتی امنیت

- تنظیمات امنیتی Host

- مبانی رمز نگاری

فصل اول معماری سیستم

در این فصل به بررسی قسمت های مختلف سیستم عامل لینوکس می پردازیم .

فرض را بر این می گیریم که شما به یک کامپیوتر که بر روی آن لینوکس نصب شده است دسترسی دارید و در Shell یا Terminal آن دستوراتی را که لازم باشد را می توانید وارد کنید و خروجی ها رو مشاهده کنید.

مواردی که مورد بررسی قرار می دهیم :

- تشخیص و پیکربندی تنظیمات سخت افزاری
- بالا آمدن (Boot) سیستم
- تغییر Runlevel ها و خاموش یا Reboot کردن سیستم

معماری سیستم

همان طور که می دانید همه سیستم عامل ها بر روی سخت افزار اجرا می شوند و این سخت افزار بر روند اجرای سیستم عامل تأثیر می گذارد. مثلاً سخت افزار می تواند سریع یا کند، قابل اتکا یا دچار مشکل باشد .

پس تمام کامپیوترها به همراه مجموعه ای از قطعات سخت افزاری روانه ی بازار می شوند که شامل مواردی مانند پردازنده ی مرکزی ، RAM و هارد دیسک هستند . در قلب بسیاری از این قطعات سخت افزاری Firmware قرار دارد که ابزارهای پیکربندی رو فراهم می آورد و روند بالا آمدن سیستم عامل را هم آغاز می کند. با مطالعه ی دفترچه ی راهنمای بسیاری از قطعات سخت افزاری می توانید در مورد Firmware آنها اطلاعات زیادی کسب کنید، اما به یاد داشته باشید که همین که سیستم عامل لینوکس بالا آمد، مدیریت دستگاه های سخت افزاری از طریق ابزارهایی خواهد بود که سیستم عامل لینوکس فراهم آورده است . قسمت های کلیدی که توسط Firmware و بعد از اینکه سیستم عامل بالا آمد، توسط لینوکس مدیریت می شوند عبارتند از: وقفه ها ، آدرس های ورودی و خروجی ، دسترسی مستقیم به حافظه و واسط های ATA هارد دیسک ها که خود بر دو نوع SATA و PATA هستند که اخیراً نوع SATA آن رایج تر شده است.

Firmware

مهمترین Firmware همانی است که بر روی مادربرد سیستم نصب شده است، که نقش راه اندازی قطعات و کنترلرهای سخت افزاری مادربرد کامپیوتر و کنترل روند بالا آمدن سیستم رو بر عهده دارد. تا اواخر سال 2010 اکثر سیستم ها مبتنی بر BIOS بودند که مخفف عبارت Basic Input/Output System است .

از اوایل سال ۲۰۱۱ سیستم های جدید، مبتنی بر Firmware ی هستند که EFI نام دارد که مخفف Extended Firmware Interface است. نسخه ی تازه تر EFI که EFI 0.2 یا همان UEFI نام دارد، به کمک دو شرکت Microsoft و Intel توسعه یافته است که امکانات بسیار بیشتری را در مقایسه با نسخه های قبلی در اختیار مسئول سیستم قرار می دهد. بنابراین استاندارد موجود برای سیستم های کامپیوتری جدید UEFI است. البته به خاطر حفظ سازگاری با نسخه های قبلی سیستم های کامپیوتری بیشتر نسخه های UEFI امکان کار کردن در حالت [BIOS](#) را هم دارند. در صورتی که قطعه ای از قطعات سخت افزاری قدیمی باشد، باز هم امکان کار با آن وجود دارد. در ادامه ابتدا به پیکربندی سخت افزار می پردازیم و بعد هم به دقت روند بالا آمدن سیستم رو بررسی خواهیم کرد.

بررسی قطعات USB

توزیع های جدید لینوکس به همراه راه اندازهای USB ارائه شده اند و نیازی نیست که برای اتصال و راه اندازی قطعات USB معمولی کار خاصی انجام بدهید. بنابراین اگر مثلا یک کیبورد USB رو به کامپیوترتان که بر روی آن لینوکس نصب شده است وصل کنید، لینوکس آن را خواهد شناخت.

نکته : به خاطر داشته باشید که برای استفاده از USB 0.3 باید نسخه ی کرنل لینوکسی که از آن استفاده می کنید 2.6,31 یا بالاتر باشد. برای دیدن نسخه ی [کرنل](#) لینوکسی که دارید دستور [uname -r](#) را بزنید.

```
rashvand@faraznetwork: ~  
File Edit View Search Terminal Help  
rashvand@faraznetwork:~$ uname -r  
4.15.0-29-generic  
rashvand@faraznetwork:~$
```

در خط فرمان لینوکس دستور زیر رو وارد کنید و سپس کلید Enter رو فشار بدهید :

\$ lsusb

خروجی اون به مشابه خروجی زیر خواهد بود :

```
rashvand@faraznetwork: ~  
File Edit View Search Terminal Help  
rashvand@faraznetwork:~$ lsusb  
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub  
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub  
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse  
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub  
rashvand@faraznetwork:~$
```

FarazNetwork.ir

با توجه به خروجی این دستور متوجه میشویم که 4 پورت USB تشخیص داده شده است (بزرگترین شماره در ستون دوم 003 است) در مثال ما هیچ دستگاهی به پورت های USB وصل نیست. اگر دستگاهی به پورت های USB سیستمی که شما بر روی آن lsusb را اجرا می کنید متصل باشد، در خروجی دستور [lsusb](#) اطلاعات آنها را مشاهده خواهید کرد.

معماری سیستم

توصیه می کنیم که حتماً به صفحه ی راهنمای این دستور مراجعه کنید. برای این کار دستور [man lsusb](#) رو بزنید. این دستور آرگومان های زیادی را به شما نمایش می دهد که مهمترین آنها را در جدول زیر مشاهده می کنید:

آرگومان	توضیح
-v	در خروجی دستور توضیحات بیشتری رو به ما نمایش میدهد.
-s [[bus]:]devnum	در خروجی تنها اطلاعات مربوط به bus و devnum داده شده رو چاپ میکند.
-d [vendor]:[product]	به کمک این آرگومان می توانید اطلاعات مربوط به محصول خاصی از تولید کننده خاصی رو ببینید.
-t	اطلاعات را در خروجی به صورت درختی نمایش می دهد، به این ترتیب راحت تر می توان تشخیص داد که چه دستگاهی به چه کنترلری متصل شده است.

دستور lspci

علاوه بر دستگاه های USB، قطعاتی داریم که بر روی بایس های PCI کامپیوتر قرار دارند. جهت مشاهده ی پیکربندی دستگاه های PCI دستور زیر را بزنید:

\$ lspci

```
rashvand@faraznetwork: ~  
File Edit View Search Terminal Help  
rashvand@faraznetwork:~$ lspci  
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge (rev 01)  
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP bridge (rev 01)  
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 08)  
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)  
00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)  
00:07.7 System peripheral: VMware Virtual Machine Communication Interface (rev 10)  
00:0f.0 VGA compatible controller: VMware SVGA II Adapter  
00:10.0 SCSI storage controller: LSI Logic / Symbios Logic 53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI (rev 01)  
00:11.0 PCI bridge: VMware PCI bridge (rev 02)  
00:15.0 PCI bridge: VMware PCI Express Root Port (rev 01)  
00:15.1 PCI bridge: VMware PCI Express Root Port (rev 01)  
00:15.2 PCI bridge: VMware PCI Express Root Port (rev 01)  
00:15.3 PCI bridge: VMware PCI Express Root Port (rev 01)  
00:15.4 PCI bridge: VMware PCI Express Root Port (rev 01)  
00:15.5 PCI bridge: VMware PCI Express Root Port (rev 01)  
00:15.6 PCI bridge: VMware PCI Express Root Port (rev 01)  
00:15.7 PCI bridge: VMware PCI Express Root Port (rev 01)  
00:16.0 PCI bridge: VMware PCI Express Root Port (rev 01)  
00:16.1 PCI bridge: VMware PCI Express Root Port (rev 01)
```

مثلا فرض کنید که می خواهیم پیکربندی PCI کارت شبکه را مشاهده کنیم، دستور زیر را وارد می کنیم :

\$ lspci | grep Eth

که بر روی سیستم ما خروجی زیر را تولید می کند:

```
rashvand@faraznetwork: ~  
File Edit View Search Terminal Help  
rashvand@faraznetwork:~$ lspci | grep Eth  
02:01.0 Ethernet controller: Advanced Micro Devices, Inc. [AMD] 79c970 [PCnet32  
LANCE] (rev 10)  
rashvand@faraznetwork:~$
```

FarazNetwork.ir

پس ما یک پورت شبکه بر روی سیستم داریم. آدرس 02:01.0 برای این کارت شبکه به این معناست که این کارت شبکه بر روی BUS با شماره ی 02، اولین قطعه (01) سخت افزاری است (معمولا اولین قطعه سرعت بالقوه ی بالاتری دارد) و 0. هم شماره ی [Function](#) آن را نمایش می دهد. پس 02:01.0 به صورت Bus: Device Function اطلاعات پیکربندی سخت افزاری را نمایش می دهد.

یکی از نکات مهمی که باید به خاطر داشته باشید این است که تا زمانی که محل اتصال قطعات سخت افزاری را در داخل کامپیوتر جا به جا نکرده باشید، این اطلاعات ثابت خواهند بود. به خصوص این مسئله برای کارت های شبکه اهمیت خاصی دارد چون ممکن است با ایجاد تغییرات نرم افزاری نام آنها عوض شود، مثلاً `eth1` تبدیل شود به `eth4` درحالی که اطلاعات پیکربندی سخت افزاری آن تا زمانی که جابه جایی سخت افزاری صورت نداده باشید، ثابت خواهد بود (معمولاً تغییرات نرم افزاری بسیار بیشتر از تغییرات سخت افزاری است)

توصیه می کنیم که حتماً با اجرای دستور [man lspci](#) به صفحه ی راهنمای این دستور مراجعه نمایید

```
rashvand@faraznetwork: ~
File Edit View Search Terminal Help
lspci(8)                                The PCI Utilities                                lspci(8)

NAME
    lspci - list all PCI devices

SYNOPSIS
    lspci [options]

DESCRIPTION
    lspci is a utility for displaying information about PCI buses in the
    system and devices connected to them.

    By default, it shows a brief list of devices. Use the options described
    below to request either a more verbose output or output intended for
    parsing by other programs.

    If you are going to report bugs in PCI device drivers or in lspci
    itself, please include output of "lspci -vvx" or even better "lspci
    -vvxxx" (however, see below for possible caveats).

    Some parts of the output, especially in the highly verbose modes, are
    probably intelligible only to experienced PCI hackers. For exact defi-
    nitions of the fields, please consult either the lspci manual or
    the pciutils manual.

Manual page lspci(8) line 1 (press h for help or q to quit)
```

آشنایی با Kernel Modules

قطعات سخت افزاری در لینوکس توسط راه اندازهای کرنل ([هسته ی سیستم عامل لینوکس](#)) کنترل می شوند. بسیاری از راه اندازهای کرنل به فرم ماژول هستند که فایل های مستقلی هستند که معمولاً در مسیر `/lib/modules` قرار دارند که می توانند جهت فراهم آوردن دسترسی به قطعات سخت افزاری مرتبط با خودشان بارگذاری شوند، یا جهت غیرفعال کردن چنان دسترسی، `Unload` شوند. لینوکس در هنگام بالا آمدن ماژول هایی را که نیاز دارد، را بارگذاری می کند. فقط در صورتیکه نیاز به بارگذاری ماژول های اضافی که خودتان نیاز دارید، ممکن است که لازم باشد که توانایی `Load` یا `Unload` کردن ماژول ها را داشته باشید. دستور زیر را وارد کنید :

\$ `lsmod`

خروجی آن به این صورت خواهد بود :

```
rashvand@faraznetwork: ~  
File Edit View Search Terminal Help  
rashvand@faraznetwork:~$ lsmod  
Module                Size  Used by  
nls_utf8               16384  1  
isofs                 45056  1  
snd_ens1371           28672  4  
snd_ac97_codec        131072  1 snd_ens1371  
gameport              16384  1 snd_ens1371  
ac97_bus              16384  1 snd_ac97_codec  
crct10dif_pclmul      16384  0  
crc32_pclmul          16384  0  
snd_pcm              98304  2 snd_ac97_codec,snd_ens1371  
ghash_clmulni_intel   16384  0  
cryptd               24576  1 ghash_clmulni_intel  
intel_rapl_perf       16384  0  
snd_seq_midi          16384  0  
snd_seq_midi_event    16384  1 snd_seq_midi  
snd_rawmidi           32768  2 snd_seq_midi,snd_ens1371  
vmw_balloon           20480  0  
snd_seq              65536  2 snd_seq_midi_event,snd_seq_midi  
snd_seq_device        16384  3 snd_seq,snd_rawmidi,snd_seq_midi  
snd_timer            32768  2 snd_seq,snd_pcm  
input_leds            16384  0  
joydev               24576  0  
snd                  81920  15 snd_seq,snd_ac97_codec,snd_timer,snd_rawmidi,sn
```

ستون اول نام تمام Module هایی را که بارگذاری شده اند نمایش می دهد. برای کسب اطلاعات بیشتر در مورد Module خاصی می توانید دستور `modinfo` و بعد نام آن ماژول را وارد کنید. ستون دوم نمایش دهنده ی بخشهایی است که از آن ماژول استفاده می کنند. به خاطر داشته باشید، که خروجی دستور `lspci` ممکن است بر روی برخی سیستم ها بسیار طولانی شود، به همین دلیل توصیه می کنیم که آن را به صورت `lsmod | less` اجرا کنید.

نکته مهم: ممکن است شما دو کامپیوتر داشته باشید که سخت افزارهای کاملاً یکسانی دارند ولی خروجی دستور `lsmod` بر روی آنها متفاوت است. یکی از دلایلی که میتواند باعث این مسئله شود این است که امکان [Compile](#) کردن و چسباندن Module به خود هسته ی لینوکس وجود دارد که در این صورت دیگر نام آن در خروجی `lsmod` نخواهد آمد، زیرا که آن Module دیگر بخشی از هسته ی سیستم عامل محسوب می شود و از همان ابتدای کار که هسته بارگذاری می شود در حافظه ی اصلی حضور خواهد داشت. برای بارگذاری Moduleی که نیاز دارید ولی قبلاً بارگذاری نشده است، می توانید از دو دستور `insmod` و `modprobe` استفاده کنید. به عنوان مثال:

```
$ modprobe floppy
```

همان کاری را انجام می دهد که فرمان

```
$ insmod /lib/modules/2.6.26/kernel/drivers/block/floppy.k
```

انجام خواهد داد. معمولاً روش اول توصیه می شود.

برای Unload کردن یک Module که قبلاً بارگذاری شده است می توانید از دستور [modprobe -r](#) یا `rmmod` استفاده کنید.

نکته : جهت اطلاعات بیشتر صفحات `man` دستورات گفته شده در بالا را مطالعه کنید. مثلاً دستور `man modprobe` را وارد کنید و بعد آن را مطالعه کنید.

بررسی دایرکتوری /dev

این دایرکتوری شامل فایل های ویژه جهت نمایش دستگاه های متصل شده به سیستم است .

دستورات زیر را اجرا کنید:

```
$ cd /dev
```

```
$ ls -l | less
```

```
rashvand@faraznetwork: /dev
File Edit View Search Terminal Help
total 0
crw----- 1 root    root      10, 175 نئوژ 16 12:47 agpgart
crw-r--r-- 1 root    root      10, 235 نئوژ 16 12:47 autofs
drwxr-xr-x 2 root    root      480 نئوژ 16 12:47 block
drwxr-xr-x 2 root    root       80 نئوژ 16 12:46 bsg
crw----- 1 root    root     10, 234 نئوژ 16 12:46 btrfs-control
drwxr-xr-x 3 root    root       60 نئوژ 16 12:46 bus
lrwxrwxrwx 1 root    root        3 نئوژ 16 12:47 cdrom -> sr0
lrwxrwxrwx 1 root    root        3 نئوژ 16 12:47 cdrw -> sr0
drwxr-xr-x 2 root    root    3540 نئوژ 16 12:47 char
crw----- 1 root    root        5,  1 نئوژ 16 12:48 console
lrwxrwxrwx 1 root    root        11 نئوژ 16 12:46 core -> /proc/kcore
crw----- 1 root    root     10,  59 نئوژ 16 12:47 cpu_dma_latency
crw----- 1 root    root     10, 203 نئوژ 16 12:46 cuse
drwxr-xr-x 7 root    root      140 نئوژ 16 12:47 disk
crw-rw----+ 1 root    audio    14,  9 نئوژ 16 12:47 dmide
drwxr-xr-x 3 root    root     100 نئوژ 16 12:46 dri
lrwxrwxrwx 1 root    root        3 نئوژ 16 12:47 dvd -> sr0
crw----- 1 root    root     10,  61 نئوژ 16 12:47 ecryptfs
crw-rw---- 1 root    video    29,  0 نئوژ 16 12:46 fb0
lrwxrwxrwx 1 root    root       13 نئوژ 16 12:46 fd -> /proc/self/fd
crw-rw-rw- 1 root    root        1,  7 نئوژ 16 12:47 full
crw-rw-rw- 1 root    root    10, 229 نئوژ 16 12:47 fuse
:█
```

برای رفع ابهام، در این قسمت تنها یکبار به صورت بسیار جزئی و ساده نحوه ی نوشتن دستور بالا را توضیح می دهیم:

توضیح دستور بالا: بنویسید ls سپس یک فاصله قرار دهید، بعد علامت منها را بزنید و بعد از آن، بدون فاصله، ال کوچک انگلیسی را تایپ کنید. حالا یک فاصله قرار دهید و کلید شیفٹ را نگه دارید و کلید \ را بر روی کیبورد خود فشار دهید تا کاراکتر | که به آن Pipe می گویند نوشته شود. حالا یک فاصله قرار دهید و بنویسید less. بعد از اجرای دستور بالا، با فشار دادن فلش های بالا و پایین می توانید در متن خروجی، در صورتی که بیش از یک صفحه باشد، حرکت کنید. برای خروج کلید q را فشار دهید.

با دقت در خروجی دستورات بالا متوجه می شویم که در زیر دایرکتوری /dev دستگاه هایی ([Devices](#)) که به سیستم متصل هستند، مثل ماوس، کیبورد، هارد دیسک و ... قرار دارند.

در خروجی دستور بالا (ls -l | less) سمت راست ترین فیلد اسم فایل، در کنارش تاریخ و ساعت و مالک فایل و سپس مجوزهای دسترسی و استفاده از فایل است . در لینوکس تقریباً همه چیز یک نمایش به صورت فایل دارد، بنابراین مثلاً فلش درایو شما می تواند در آدرس /dev/sdb نمود فایلی داشته باشد.

تشخیص نوع هارد دیسک و دستگاه های کامپیوتر از روی نام فایل در زیر /dev

در زیر دایرکتوری /dev فایل هایی وجود دارند که نماینده ی دستگاه های سخت افزاری هستند. فایل های مربوط به هارددیسک های IDE معمولاً به صورت /dev/hdX است که در آن X یک حرف است ، مثلاً hda و در صورتی که این دیسک پارتیشن بندی شده باشد، مواردی همچون /dev/hda1 و /dev/hda2 نیز خواهیم داشت (که مثلاً دو عدد پارتیشن بر روی این هارددیسک پیکربندی شده است). در صورتی که هارددیسک از نوع SATA باشد به صورت نشان داده می شود. درایوهای فلش نیز به صورت /dev/sdX نمایش داده می شوند. CDROM و DVDROM نیز به صورت /dev/scdX و /dev/srX نمود پیدا می کنند.

دسته بندی دستگاه ها (Devices) از یک دیدگاه می توان دستگاه های کامپیوتری را به دو دسته تقسیم کرد :

۱. دستگاه های بلوکی (Block Devices) : تبادل اطلاعات میان چنین دستگاهی و سایر بخشهای کامپیوتر به صورت بلوک به بلوک است هارد دیسک یک نمونه از این نوع دستگاه ها است. یا مثلاً اگر بخواهید از CD ROM یک بایت بخوانید مجبور هستید که یک [Block](#) بخوانید و سپس از آن Block خوانده شده، بایت مورد نظر شما برای شما برگردانده می شود؛ دلیل این است که اطلاعات به صورت بلوک در CDROM ذخیره شده است . خواندن به صورت بلوکی از این دستگاه ها به دلیل نیاز به کارایی بالاست ، زیرا که خواندن به صورت بیت به بیت سرعت انتقال داده ها را پایین می آورد.
۲. دستگاه های کاراکتری (Character Devices) : این نوع از دستگاه ها اطلاعات را به صورت حرف به حرف برمی گردانند. به عنوان مثال کیبورد، مودم و پرینتر در این دسته قرار می گیرند.

در سمت چپ خروجی دستور ([ls -l | less](#)) در زیر دایرکتوری /dev مجوزهای دسترسی نوشته شده است . اولین حرف این مجوزها، معنای خاصی دارد.

اولین حرف	نمایش دهنده
d	دایرکتوری
خط تیره (-)	File
b	Block Device
p	Character Device
	Pipe
s	Socket

پس b برای دستگاه های بلوکی و c برای دستگاه های کاراکتری است .

نمایش محتوای فایل

با فرمان `cat` میتوان محتوای فایل را نشان داد. به عنوان مثال برای اینکه محتویات هارد دیسک را نشان دهیم، فرمان زیر را اجرا می کنیم :

```
$ cat /dev/sda
```

از آنجا که خروجی ممکن است طولانی باشد و نمایش آن زمان بر شود، بعد از اجرای دستور جهت خاتمه ی کار می توانید از ترکیب کلیدهای `Control+C` استفاده کنید.

روش دادن اسم دوّم به یک Device

یکی از روشهای انجام این کار استفاده از فرمان [ln](#) است :

```
$ ln /dev/fd0 a:
```

```
$ ls a: fd0
```

```
4349          a:
```

```
4349          fd
```

نکته ی کاربردی: یکی از ویژگی های خوب دایرکتوری `dev` این است که می توانید بفهمید آخرین باری که یک Device کار کرده چه زمانی بوده است . به این منظور

کافی است به تاریخ ایجاد فایل ها در خروجی دستور `ls -alh` نگاه کنید.

حالا می خواهیم ببینیم چند تا Printer داریم و پورت های آنها را ببینیم :

```
$ ls -l /dev/lp*
```

```
crw-rw----1 root lp 6,0 aug 18 2006 jan fd0
```

در اینجا ما یک پرینتر داریم. ممکن است در سیستم شما پرینتر وصل باشد یا نباشد.

تعریف یک دستگاه جدید با دستور `mknod`

```
$ cd /dev
```

```
$ mknod lp1 c 6 1
```

```
name: lp1
```

```
type: c
```

```
major: 6
```

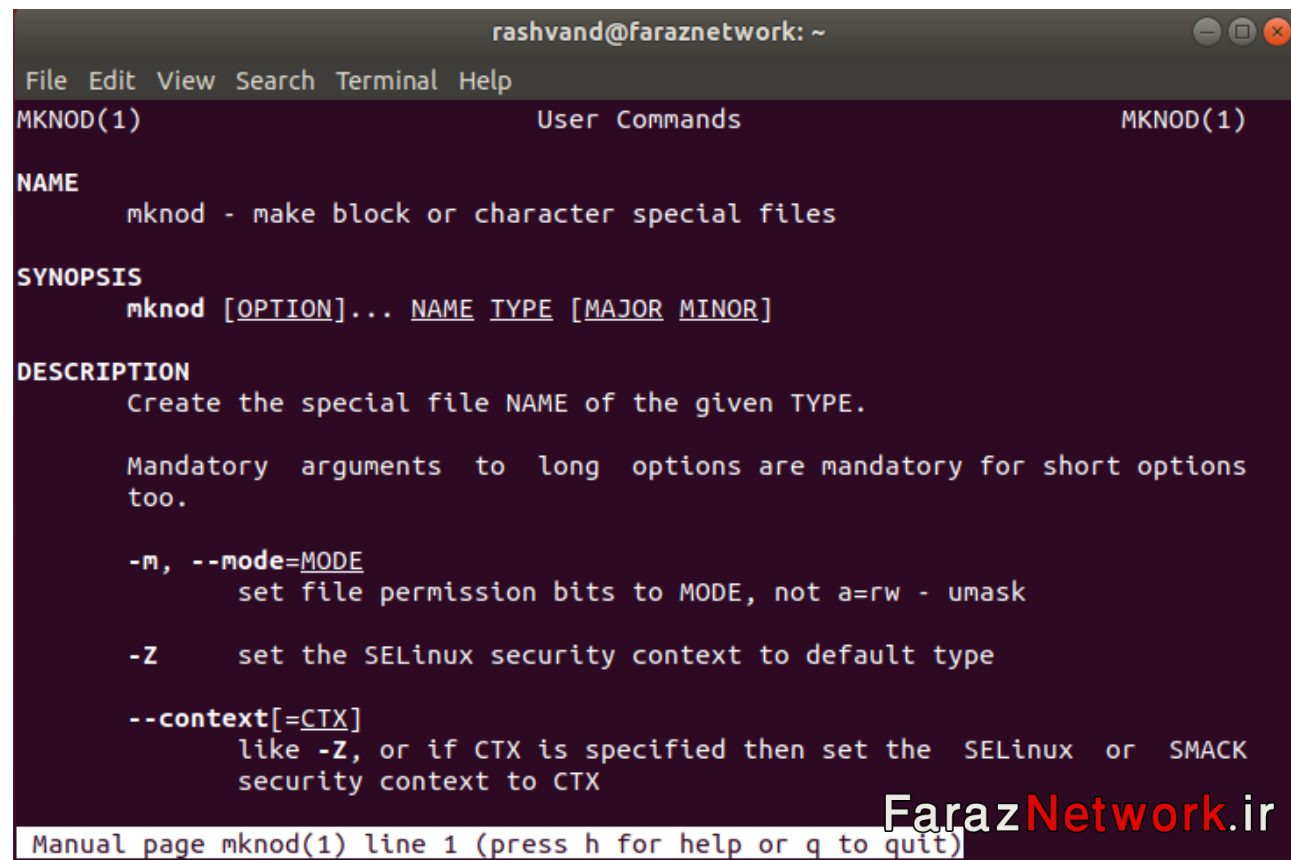
```
minor: 1
```

با دستور بالا یک پرینتر دیگر هم تعریف کردیم. البته توجه کنید که متناظر با پرینتر با نام `lp1` دستگاه پرینتر وجود ندارد، بنابراین با ارسال فایل به این `Device` چیزی چاپ نخواهد شد. عدد ۶ عدد `major` به معنی اصلی است که هسته ی لینوکس ([Kernel](#)) با استفاده از آن تعیین می کند که چه `Device Driver` ی را انتخاب کند و با آن کار کند. عدد دوم را `minor` می گوئیم و با استفاده از آن معلوم می شود که یک `Device` چه مدلی دارد و همچنین کدام یک از پرینترهای متصل به کامپیوتر است. `type: c` نشان دهنده ی نوع `Device` است که در اینجا با نوع دستگاه کاراکتری سر و کار داریم پس برخی فایل ها در زیر دایرکتوری `/dev` نشان دهنده ی دستگاه های `Character` ی یا `Block` ی هستند و برخلاف فایل های معمولی قد و حجم ندارند، بلکه `major` و `minor` دارند که `major` شماره `Device` است و `minor` تعداد دستگاه با آن نوع را مشخص می کند. مثلاً وقتی چند تا هارد دیسک به یک کامپیوتر متصل هستند که از یک نوع `Driver` استفاده می کنند برای تشخیص آنها از یکدیگر کافی است که به عدد `minor` مراجعه شود. به این ترتیب با این دو عدد `Device` ها از هم تفکیک و شناخته می شوند.

یادتان باشد که فقط Root اجازه ساخت فایل های Device را دارد، و در واقع به طور معمول این فایل ها را ما نمی سازیم و معمولاً این خود سیستم عامل است که بعد از متصل شدن دستگاه به کامپیوتر آن را تشخیص می دهد و نمود فایلی آن را در زیر دایرکتوری /dev می آورد. بنابراین فایل های زیر دایرکتوری dev پویا هستند و خود سیستم Device هایش را شناسایی می کند.

پس در این قسمت یاد گرفتیم که فرمانی که امکان ساخت Device File را میدهد [mknod](#) است و البته فقط کاربر Root امکان استفاده از آن را دارد. برای مشاهده ی صفحه ی راهنمای دستور mknod، دستور زیر را اجرا کنید :

\$ man mknod



```
rashvand@faraznetwork: ~
File Edit View Search Terminal Help
MKNOD(1) User Commands MKNOD(1)

NAME
    mknod - make block or character special files

SYNOPSIS
    mknod [OPTION]... NAME TYPE [MAJOR MINOR]

DESCRIPTION
    Create the special file NAME of the given TYPE.

    Mandatory arguments to long options are mandatory for short options
    too.

    -m, --mode=MODE
        set file permission bits to MODE, not a=rw - umask

    -Z
        set the SELinux security context to default type

    --context[=CTX]
        like -Z, or if CTX is specified then set the SELinux or SMACK
        security context to CTX

Manual page mknod(1) line 1 (press h for help or q to quit)
```

حال به عنوان Root دستور زیر را اجرا کنید :

```
$ cd /dev
```

```
$ su
```

```
# mknod faraz b 250 44
```

b یعنی Block Device و شماره ی اول major و شماره ی بعدی minor است . البته ما اجازه ساخت نداریم چون در کامپیوتر ما برای هسته ی لینوکس دستگاهی با شماره ی major ۲۵۰ وجود ندارد، پس پیغام No Such Device را مشاهده می کنیم. پس به جای اینکه به طور تصادفی یک عدد major بنویسیم به خروجی دستور زیر مراجعه می کنیم و متناسب با دستگاه مورد نظرمان شماره ی major را انتخاب می کنیم :

```
$ cat /proc/devices
```

به جز c، d، b و - ما در سمت چپ دایرکتوری dev حرف های دیگری هم داریم مثل p به فرمان های زیر توجه کنید:

```
$ mknod faraz p
```

```
$ ls -l
```

```
total 8
```

```
faraz prw-r--r--1 n. faraz lpi1 0 تاریخ
```

یک Pipe File ساختیم. Pipe File ها که معمولاً ابتدا و انتها ندارند و اطلاعات پشت سر هم نمی آید. به طور معمول [Pipe File](#) ها خیلی کاربرد دارند و البته اطلاعات شان هم تمامی ندارد. از جمله دستگاه هایی که به صورت Pipe File نمایش داده می شوند رطوب سنج و رادار هستند.

علاوه بر فرمان `mknod` در لینوکس یک فرمان دیگر به نام [MAKEDEV](#) (با حروف بزرگ) وجود دارد و به منظور ایجاد فایل دستگاه ها در زیر دایرکتوری `dev` به کار می رود (جهت مشخص شدن آخرین وضعیت دستگاه ها و لیست کردن مجدد آنها در زیر `/dev`) با این فرمان، دیگر نیازی نداریم `mknod` را به کار ببریم.

بررسی دایرکتوری `/proc`

این دایرکتوری، یک دایرکتوری خاص ساکن در حافظه و حاوی اطلاعات مختلف درباره کارهای در حال اجرا در سیستم لینوکس است. در زیر دایرکتوری `proc` مجموعه ای از دایرکتوری های عددی و تعدادی فایل مرتبط با تجهیزات وجود دارد.

```
$ cd /proc
```

```
$ ls -l | less
```

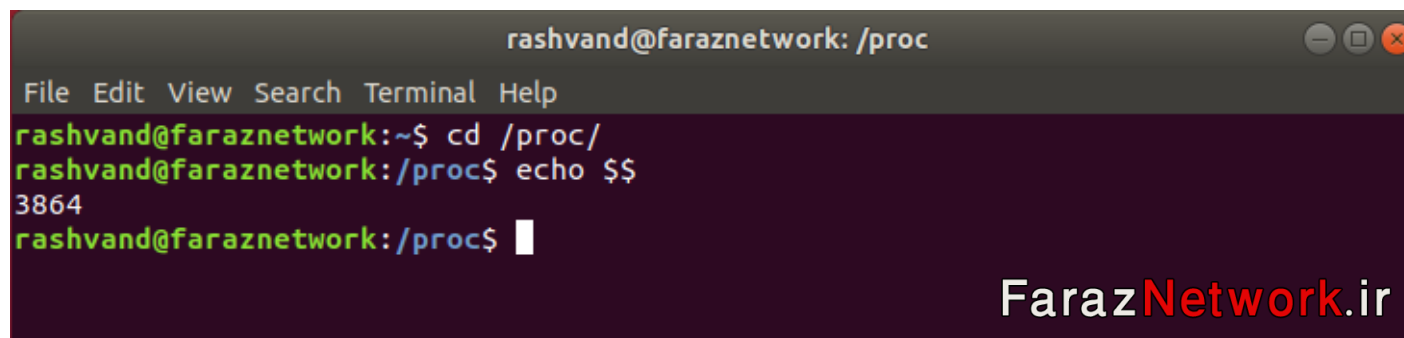
```
rashvand@faraznetwork: /proc
File Edit View Search Terminal Help
total 0
dr-xr-xr-x  9 root      root      0  16 12:46 1
dr-xr-xr-x  9 root      root      0  16 12:46 1
0
dr-xr-xr-x  9 rtkit     rtkit     0  16 12:47 1
000
dr-xr-xr-x  9 gdm      gdm       0  16 12:48 1
015
dr-xr-xr-x  9 gdm      gdm       0  16 12:48 1
018
dr-xr-xr-x  9 gdm      gdm       0  16 12:48 1
021
dr-xr-xr-x  9 gdm      gdm       0  16 12:48 1
025
dr-xr-xr-x  9 root      root      0  16 12:48 1
032
dr-xr-xr-x  9 root      root      0  16 12:48 1
035
dr-xr-xr-x  9 gdm      gdm       0  16 12:48 1
037
dr-xr-xr-x  9 gdm      gdm       0  16 12:48 1
045
dr-xr-xr-x  9 gdm      gdm       0  16 12:48 1
:
```

دایرکتوری `proc` فقط به صورت Logical است و موجودیت مستقلی نیست و آینه ی تمام نما از اطلاعات Kernel است . با استفاده از داده های زیردایرکتوری `/proc` می توانید ببینید که کرنل چه فایل هایی را Open کرده و چند نفر login هستند. به محض اینکه سیستم را Down کنید دایرکتوری `proc` گم می شود و به محض اینکه Up کنید ساخته می شود. در زیر این دایرکتوری به ازای هر [Process](#) ی یک دایرکتوری داریم. وقتی فرمان زیر را در زیر دایرکتوری `proc` می زنیم به ما یک عدد می دهد، که شماره پروسس (شل) مرتبط به شما می باشد.

```
$ cd /proc
```

```
$ echo $$
```

```
3864
```



```
rashvand@faraznetwork: /proc
File Edit View Search Terminal Help
rashvand@faraznetwork:~$ cd /proc/
rashvand@faraznetwork:/proc$ echo $$
3864
rashvand@faraznetwork:/proc$
```

FarazNetwork.ir

هر کاری که انجام دهید و هر فرمانی که بزنید Admin از طریق فایل های زیر دایرکتوری `proc` می تواند ببیند که کجا هستید. تعداد دایرکتوری هایی که زیر دایرکتوری `proc` وجود دارند ثابت نیست و پیوسته در حال تغییر است .

نکته : فرمان های `who` و `w` برای اینکه لیست کاربران Login کرده را تهیه کنند، از داده های زیر دایرکتوری `proc` استفاده می کنند.

البته در زیر این دایرکتوری تعدادی فایل مرتبط با تجهیزات نیز داریم. فرمان زیر را اجرا کنید :

\$ cat cpuinfo

```
rashvand@faraznetwork: /proc
File Edit View Search Terminal Help
rashvand@faraznetwork:/proc$ cat cpuinfo
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 42
model name    : Intel(R) Pentium(R) CPU G630 @ 2.70GHz
stepping      : 7
microcode     : 0x14
cpu MHz       : 2693.909
cache size    : 3072 KB
physical id   : 0
siblings      : 1
core id       : 0
cpu cores     : 1
apicid        : 0
initial apicid : 0
fpu           : yes
fpu_exception : yes
cpuid level   : 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clfl
ush mmx fxsr sse sse2 ss syscall nx rdtscp lm constant_tsc arch_perfmon nopl xtopology tsc_rel
iable nonstop_tsc cpuid pni pclmulqdq ssse3 cx16 pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline
_timer xsave hypervisor lahf_lm pti tsc_adjust arat
bugs          : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass
bogomips      : 5387.81
clflush size  : 64
cache_alignm  : 64
address sizes : 43 bits physical, 48 bits virtual
power managem :
rashvand@faraznetwork:/proc$
```

cpuinfo اطلاعات CPU را می دهد. ممکن است کسی از شما بپرسد که CPU این کامپیوتر از چه مجموعه دستوراتی (Instruction Set) پشتیبانی می کند؛ برای پاسخ به این سؤال کافی است به فایل [/proc/cpuinfo](#) مراجعه نمایید.

نکته: برای اجرای برخی از برنامه های کاربردی به خصوص در سطح سیستمی نیاز است که مجموعه دستورات خاصی توسط CPU پشتیبانی شود، که می توانید از خروجی فرمان `cat /proc/cpuinfo` در این مورد استفاده کنید.

نکته : جهت دستیابی به اطلاعات دقیق تر در مورد CPU برنامه x86info را نصب کنید (`apt-get install x86info` یا `yum install x86info`) و سپس فرمان [x86info -c](#) را اجرا نمایید. خروجی این دستور اطلاعات بسیار دقیقی در مورد معماری پردازنده سیستم و معماری سیستم Cache آن را در اختیار شما می گذارد؛ این فرمان علاوه بر کاربرد در توسعه ی نرم افزارهای سیستمی، جهت خرید سیستم های حرفه ای و بررسی صحت اطلاعات موجود در کاتالوگ ها می تواند استفاده شود.

نکته : Bogomips (پردازش جعلی) : در پردازنده و زبان اسمبلی دستوری به نام `nop(no operation)` وجود دارد که فقط یک Cycle از CPU می برد، بدون اینکه کاری بکند و فقط زمانی را از پردازنده صرف خود می کند. مثلاً Device Driver دیسک می داند که اگر یک فرمان به دیسک بدهد باید نیم ثانیه بعد جواب بیاید به اندازه نیم ثانیه `nop` را انجام می دهد تا جواب بیاید. و اگر جواب نیامد Time out می دهد.

interrupts فایل

فایل بعدی که در زیر /proc بررسی می کنیم interrupts است . در این فایل شماره هر [interrupt](#) و تعداد تکرار آن را می بینیم.

\$ cat interrupts

```
rashvand@faraznetwork: /proc
File Edit View Search Terminal Help
rashvand@faraznetwork:/proc$ cat interrupts
CPU0
 0:      10   IO-APIC   2-edge    timer
 1:     1411   IO-APIC   1-edge    i8042
 8:        1   IO-APIC   8-edge    rtc0
 9:         0   IO-APIC   9-fasteoi acpi
12:     26418   IO-APIC  12-edge    i8042
14:         0   IO-APIC  14-edge    ata_piix
15:         0   IO-APIC  15-edge    ata_piix
16:     8578   IO-APIC  16-fasteoi vmwgfx, snd_ens1371
17:    29613   IO-APIC  17-fasteoi ehci_hcd:usb1, ioc0
18:        62   IO-APIC  18-fasteoi uhci_hcd:usb2
19:    50191   IO-APIC  19-fasteoi ens33
24:         0   PCI-MSI 344064-edge    PCIE PME, pciehp
25:         0   PCI-MSI 346112-edge    PCIE PME, pciehp
26:         0   PCI-MSI 348160-edge    PCIE PME, pciehp
27:         0   PCI-MSI 350208-edge    PCIE PME, pciehp
28:         0   PCI-MSI 352256-edge    PCIE PME, pciehp
29:         0   PCI-MSI 354304-edge    PCIE PME, pciehp
30:         0   PCI-MSI 356352-edge    PCIE PME, pciehp
31:         0   PCI-MSI 358400-edge    PCIE PME, pciehp
32:         0   PCI-MSI 360448-edge    PCIE PME, pciehp
33:         0   PCI-MSI 362496-edge    PCIE PME, pciehp
34:         0   PCI-MSI 364544-edge    PCIE PME, pciehp
```

هر مادربرد تشکیل یافته از تعدادی قطعه است که به یک مورد اصلی (یا بهتر است بگوییم یک مورد اصلی و چند مورد دیگر) وصل است. وقتی هر کدام از قطعه ها بخواهند با مورد مادر صحبت کنند Interrupt می دهند. همچنین دستگاه هایی هم که می خواهند با یکدیگر صحبت کنند Interrupt می دهد.

```
rashvand@faraznetwork: /proc
File Edit View Search Terminal Help
35: 0 PCI-MSI 366592-edge PCIe PME, pciehp
36: 0 PCI-MSI 368640-edge PCIe PME, pciehp
37: 0 PCI-MSI 370688-edge PCIe PME, pciehp
38: 0 PCI-MSI 372736-edge PCIe PME, pciehp
39: 0 PCI-MSI 374784-edge PCIe PME, pciehp
40: 0 PCI-MSI 376832-edge PCIe PME, pciehp
41: 0 PCI-MSI 378880-edge PCIe PME, pciehp
42: 0 PCI-MSI 380928-edge PCIe PME, pciehp
43: 0 PCI-MSI 382976-edge PCIe PME, pciehp
44: 0 PCI-MSI 385024-edge PCIe PME, pciehp
45: 0 PCI-MSI 387072-edge PCIe PME, pciehp
46: 0 PCI-MSI 389120-edge PCIe PME, pciehp
47: 0 PCI-MSI 391168-edge PCIe PME, pciehp
48: 0 PCI-MSI 393216-edge PCIe PME, pciehp
49: 0 PCI-MSI 395264-edge PCIe PME, pciehp
50: 0 PCI-MSI 397312-edge PCIe PME, pciehp
51: 0 PCI-MSI 399360-edge PCIe PME, pciehp
52: 0 PCI-MSI 401408-edge PCIe PME, pciehp
53: 0 PCI-MSI 403456-edge PCIe PME, pciehp
54: 0 PCI-MSI 405504-edge PCIe PME, pciehp
55: 0 PCI-MSI 407552-edge PCIe PME, pciehp
56: 10169 PCI-MSI 1130496-edge ahci[0000:02:05.0]
57: 0 PCI-MSI 129024-edge vmw_vmci
58: 0 PCI-MSI 129025-edge vmw_vmci

rashvand@faraznetwork: /proc
File Edit View Search Terminal Help
56: 10169 PCI-MSI 1130496-edge ahci[0000:02:05.0]
57: 0 PCI-MSI 129024-edge vmw_vmci
58: 0 PCI-MSI 129025-edge vmw_vmci
NMI: 0 Non-maskable interrupts
LOC: 369159 Local timer interrupts
SPU: 0 Spurious interrupts
PMI: 0 Performance monitoring interrupts
IWI: 24 IRQ work interrupts
RTR: 0 APIC ICR read retries
RES: 0 Rescheduling interrupts
CAL: 0 Function call interrupts
TLB: 0 TLB shutdowns
TRM: 0 Thermal event interrupts
THR: 0 Threshold APIC interrupts
DFR: 0 Deferred Error APIC interrupts
MCE: 0 Machine check exceptions
MCP: 65 Machine check polls
HYP: 0 Hypervisor callback interrupts
ERR: 0
MIS: 0
PIN: 0 Posted-interrupt notification event
NPI: 0 Nested posted-interrupt event
PIW: 0 Posted-interrupt wakeup event
rashvand@faraznetwork: /proc$
```

Timer Interrupt

اگر فرمان [vmstat -s](#) را بزنیم در انتهای خروجی آن به ما می گوید که Context Switching چند بار انجام شده است . Context Switching به معنای واگذاری کنترل از یک Process به Process دیگر است.

\$ vmstat -s

```
rashvand@faraznetwork: ~  
File Edit View Search Terminal Help  
rashvand@faraznetwork:~$ vmstat -s  
1951340 K total memory  
1191184 K used memory  
1000248 K active memory  
285052 K inactive memory  
162544 K free memory  
55016 K buffer memory  
542596 K swap cache  
969960 K total swap  
0 K used swap  
969960 K free swap  
7479 non-nice user cpu ticks  
210 nice user cpu ticks  
11851 system cpu ticks  
1969077 idle cpu ticks  
207 IO-wait cpu ticks  
0 IRQ cpu ticks  
405 softirq cpu ticks  
0 stolen cpu ticks  
487862 pages paged in  
154292 pages paged out  
0 pages swapped in  
0 pages swapped out  
485154 interrupts  
1017369 CPU context switches  
1560685651 boot time  
3929 forks  
rashvand@faraznetwork:~$
```

وقتی در دایرکتوری [proc](#) هستیم با فرمان زیر می توانیم مشاهده کنیم که هارد دیسک ما چند قسمت شده است :

\$ cat partitions

```
rashvand@faraznetwork: ~  
File Edit View Search Terminal Help  
major minor #blocks name  
7 0 13300 loop0  
7 1 4120 loop1  
7 2 91524 loop2  
7 3 36160 loop3  
7 4 14840 loop4  
7 5 144040 loop5  
7 6 3732 loop6  
7 7 15096 loop7  
11 0 1907568 sr0  
8 0 20971520 sda  
8 1 20969472 sda1  
7 8 144260 loop8  
7 9 2376 loop9  
7 10 3796 loop10  
7 11 88964 loop11  
7 12 1008 loop12  
7 13 154636 loop13  
7 14 55008 loop14  
7 15 90560 loop15  
7 16 54996 loop16  
7 17 35472 loop17  
rashvand@faraznetwork:~$
```

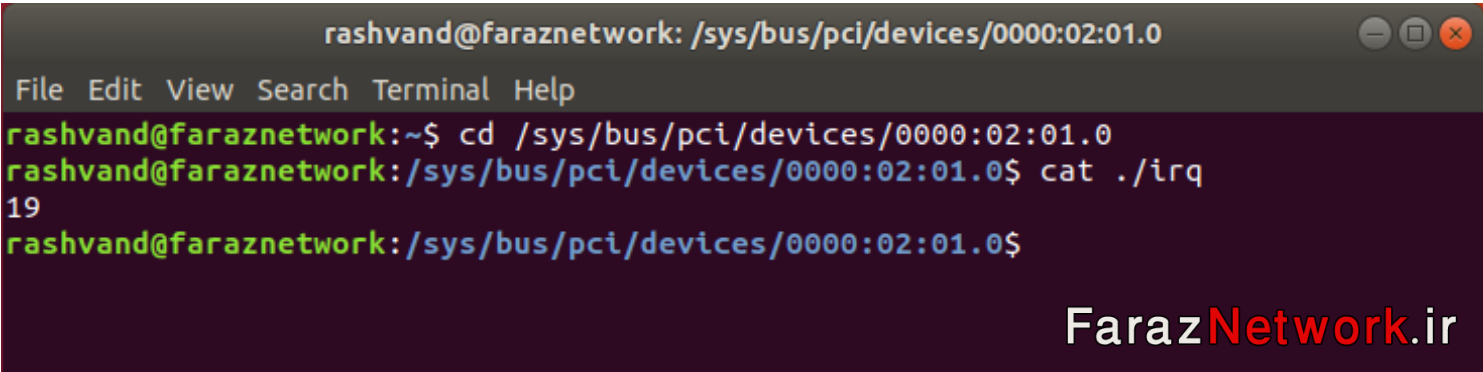
بررسی دایرکتوری /sys

فایل سیستمی مجازی در زیر دایرکتوری /sys قرار دارد که با نام [sysfs](#) شناخته می شود. این فایل سیستم محلی است که ابزارهای در سطح کاربر (و نه هسته) اطلاعاتشان را در مورد دستگاه های سیستم به دست می آورند. همچنین به واسطه ی این پوسته ی ارائه شده می توان تنظیمات مورد نظر را بر دستگاه های متصل به سیستم اعمال کرد (که البته معمولاً نیازمند دانش عمیق و تخصصی درمورد آن قطعه ی سخت افزاری است). مثلاً بر روی سیستم ما، همانطور که قبلاً دیدیم کارت شبکه در آدرس PCI با مقدار 02:01.0 قرار داشت . پس دستور زیر را اجرا می کنیم، که به دایرکتوری که حاوی فایل های مربوط به آن است وارد شویم:

```
$ cd /sys/bus/pci/devices/0000:02:01.0
```

حال فرض کنید که می خواهید بدانیم که شماره irq آن چند است . بعد از دستور بالا، دستور زیر را وارد می کنیم :

```
$ cat ./irq
```



```
rashvand@faraznetwork: /sys/bus/pci/devices/0000:02:01.0
File Edit View Search Terminal Help
rashvand@faraznetwork:~$ cd /sys/bus/pci/devices/0000:02:01.0
rashvand@faraznetwork:/sys/bus/pci/devices/0000:02:01.0$ cat ./irq
19
rashvand@faraznetwork:/sys/bus/pci/devices/0000:02:01.0$
```

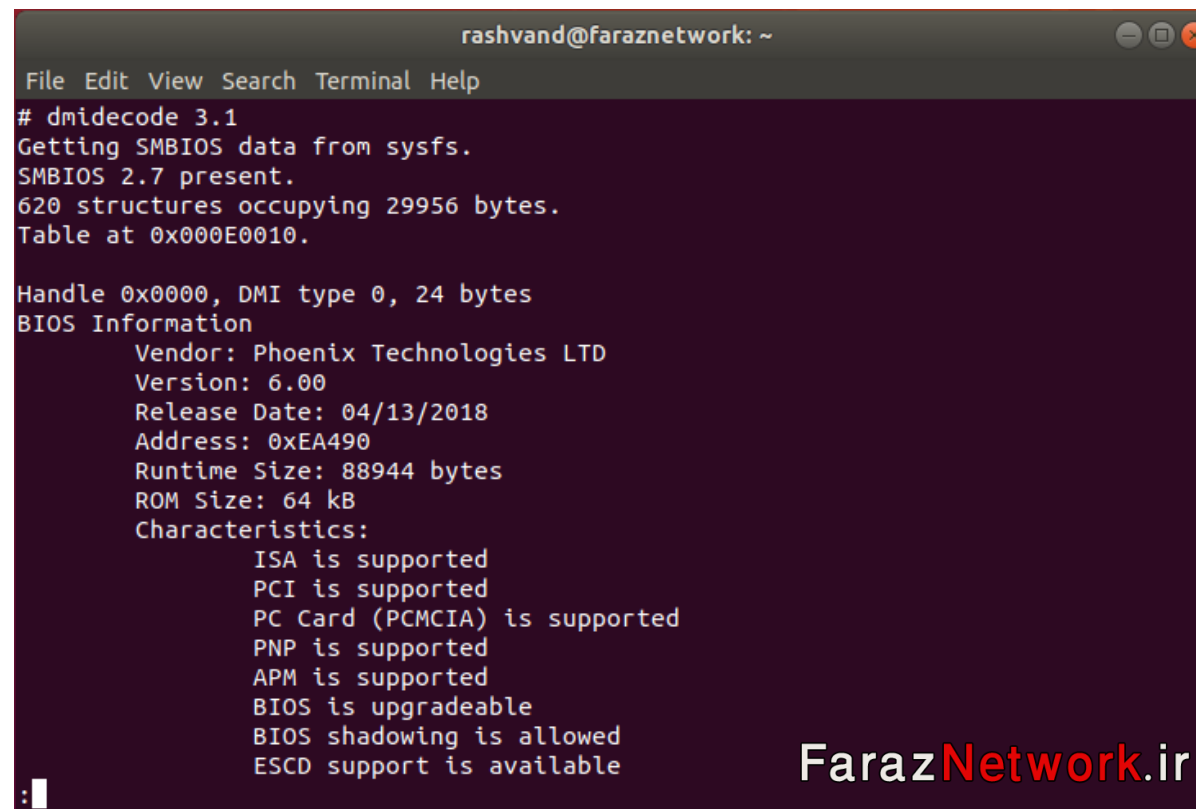
FarazNetwork.ir

که در سیستم ما شماره ی ۱۹ را بر می گرداند.

دستوراتی برای بررسی سخت افزار

زیر proc تعداد زیادی دایرکتوری عددی و فایل سخت افزاری داشتیم. تعدادی فرمان داریم که اطلاعات را از proc می خوانند. یکی از آنها فرمان [dmidecode](#) است که سخت افزار را زیر و رو میکند و اطلاعات را از زیر proc در می آورد.

\$ dmidecode |less

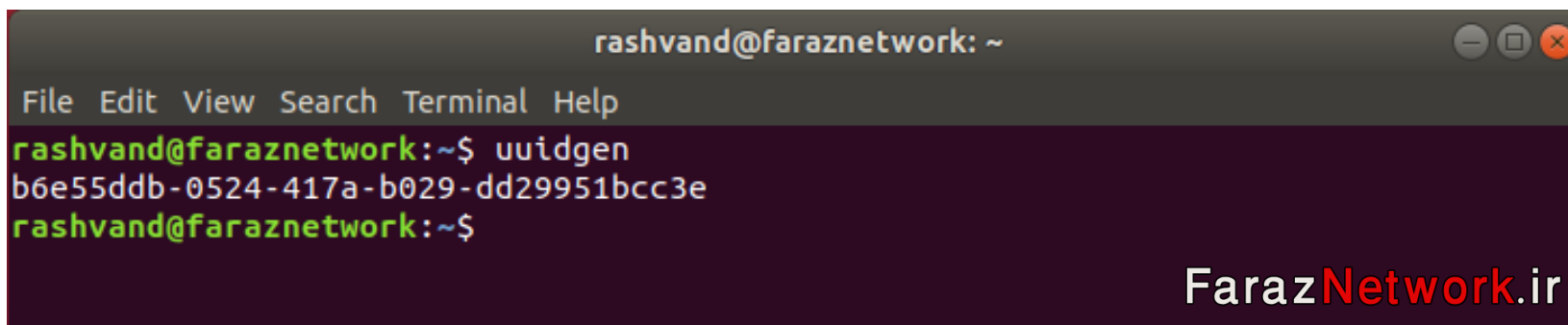
A terminal window titled 'rashvand@faraznetwork: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The output of the 'dmidecode' command is displayed in a dark purple background with white text. The output shows SMBIOS 2.7 present, 620 structures occupying 29956 bytes, and a table at 0x000E0010. It then displays BIOS information for a Phoenix Technologies LTD system, including version 6.00, release date 04/13/2018, and various supported features like ISA, PCI, PC Card, PNP, APM, BIOS upgradeable, BIOS shadowing, and ESCD support. A watermark 'FarazNetwork.ir' is visible in the bottom right corner of the terminal window.

```
rashvand@faraznetwork: ~  
File Edit View Search Terminal Help  
# dmidecode 3.1  
Getting SMBIOS data from sysfs.  
SMBIOS 2.7 present.  
620 structures occupying 29956 bytes.  
Table at 0x000E0010.  
  
Handle 0x0000, DMI type 0, 24 bytes  
BIOS Information  
    Vendor: Phoenix Technologies LTD  
    Version: 6.00  
    Release Date: 04/13/2018  
    Address: 0xEA490  
    Runtime Size: 88944 bytes  
    ROM Size: 64 kB  
    Characteristics:  
        ISA is supported  
        PCI is supported  
        PC Card (PCMCIA) is supported  
        PNP is supported  
        APM is supported  
        BIOS is upgradeable  
        BIOS shadowing is allowed  
        ESCD support is available
```

در خروجی دستور بالا اطلاعات مختلفی در مورد سخت افزار را مشاهده می کنید.

یک نکته ی مهم در مورد UUID یا "شناساگر یکتای جهانی" این است که وقتی Hard را فرمت می کنید UUID به شما می دهد. به عنوان مثال UUID کامپیوتر کلاس عدد ۳۲رقمی هگزا ۸.۴.۴.۴.۱۲ است. برای اینکه این عدد یکتا باشد و تکراری نباشد، اول Mac Address کارت شبکه کامپیوتر را می گیرد و بعد از روی آن UUID را می سازد. اگر کامپیوتر شما کارت شبکه نداشته باشد cupid را می گیرد. در صورتی که خودتان خواستید UUID بسازید می توانید از فرمان زیر استفاده کنید:

\$ uuidgen

A screenshot of a terminal window titled 'rashvand@faraznetwork: ~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command 'rashvand@faraznetwork:~\$ uuidgen' being executed, followed by the output 'b6e55ddb-0524-417a-b029-dd29951bcc3e'. The prompt returns to 'rashvand@faraznetwork:~\$'. In the bottom right corner of the terminal window, the text 'FarazNetwork.ir' is displayed in red and white.

```
rashvand@faraznetwork: ~  
File Edit View Search Terminal Help  
rashvand@faraznetwork:~$ uuidgen  
b6e55ddb-0524-417a-b029-dd29951bcc3e  
rashvand@faraznetwork:~$  
FarazNetwork.ir
```

پس یادتان باشد اگر خواستید Laptop بخريد چند فرمان را به ياد داشته باشيد و CD Live لينوکس را ببريد و اطلاعات سخت افزاری آن را بررسی کنید. علاوه بر فرمان dmidecode فرمان [lshw](#) نیز اطلاعات مهمی در مورد سخت افزار سیستم می دهد. برای مثال مشخصات کامل مادر برد و باتری و موارد دیگر را در خروجی این دستورات می بینید. هرچند خروجی دستور dmidecode بسیار مفصل است ، در برخی مواقع، هرچند نادر، اطلاعات نادرستی را به نمایش می گذارد.

Hotplug در مقابل Coldplug

هر وقت با یک قطعه ی سخت افزار سر و کار داشتید باید یک نکته را در نظر بگیرید؛ اینکه این قطعه Coldplug است یا Hotplug. تفاوت این دو نوع سخت افزار در این است که قطعات Hotplug را چه در موقع خاموش و چه در موقع روشن بودن سیستم می توانید به سیستم متصل کنید یا از آن جدا کنید، ولی قطعات Coldplug را حتماً باید در موقع خاموش بودن سیستم وصل یا جدا کنید.

تذکر مهم: متصل یا جدا کردن قطعات سخت افزاری Coldplug در هنگامی که کامپیوتر روشن است ممکن است به آن قطعه یا کل کامپیوتر آسیب بزند. بنابراین هرگز چنین کاری نکنید و قبل از متصل یا جدا کردن قطعه ی سخت افزاری خاصی از سیستم از Coldplug یا Hotplug بودن آن اطمینان حاصل نمایید. به طور معمول قطعاتی مانند پردازنده، حافظه ی RAM، کارت های PCI (مثل کارت شبکه)، و هارد دیسک ها Coldplug هستند. هرچند نوع Hotplug آنها نیز برای کاربردهای خاص طراحی شده است. نوع Hotplug این قطعات عموماً برای سرورهایی که امکان خاموش کردن آنها وجود ندارد به کار برده می شوند. مثلاً در کامپیوترهایی که زیرساخت های تلفن همراه را پشتیبانی می کنند از RAM های Hotplug استفاده می شود. به این ترتیب در صورتی که یکی از RAM ها خراب شود، فوراً می توان آن را تعویض کرد و در سیستم کار تلفن ها مشکلی پیش نمی آید.

قطعات سخت افزاری جدید خارجی (که نیازی نیست داخل Case باشند و از خارج کامپیوتر وصل می شوند)، مثل USB، Ethernet و IEEE-1394 یا همان Firewire به صورت Hotplug هستند و در هنگام روشن بودن سیستم امکان اتصال و جدا کردن آنها وجود دارد.

نکته: به برنامه های معمولی که در حال اجرا هستند، خواه توسط یک کاربر معمولی یا Root، برنامه ی User-Space گفته می شود. در مقابل این برنامه ها، برنامه های Kernel-Space قرار دارند که بخشی از هسته ی سیستم عامل لینوکس هستند. برنامه های سمت هسته به طور مستقیم می توانند با سخت افزار در ارتباط باشند، اما برنامه های User-Space با استفاده از امکاناتی که Kernel فراهم می آورد، این کار را انجام می دهند.

HAL Daemon

[HAL](#) یا همان Hardware Abstraction Layer Daemon، یا HALD یک برنامه ی User-Space است که همواره در حال اجراست و به برنامه های User-Space اطلاعاتی در مورد قطعات سخت افزاری موجود می دهد.

D- Bus

BusDesktop همانند HALD یک Daemon است و همواره در حال اجراست . D-Bus این امکان را فراهم می آورد تا پروسس ها با یکدیگر در ارتباط باشند، همچنین در مورد Event ها (مثل متصل شدن یک قطعه ی سخت افزاری جدید از USB) هم از طریق سایر Process ها و هم از طریق سخت افزار کسب اطلاع کنند.

udev

از گذشته تا کنون لینوکس گره های مربوط به سخت افزار را در مسیر /dev ایجاد می کند. وجود قطعات سخت افزاری Hotplug باعث به وجود آمدن udev شد: یک فایل سیستم مجازی که در مسیر /dev ماونت ([Mount](#)) می شود و فایل های پویای دستگاه های سخت افزاری را ایجاد می کند. تنظیمات udev از مسیر /etc/udev انجام می شود. توجه داشته باشید که تنظیمات پیشفرض برای قطعات سخت افزاری رایج، کافی است . اما قطعات سخت افزاری قدیمی تر مثل آنهایی که از پورت RS-232 یا Parallel وصل می شوند (مثلاً ماوس یا کیبورد های قدیمی یا چاپگرهایی که از پورت موازی وصل می شوند)، Coldplug هستند و لازم است که برای اتصال یا جدا کردن آنها کامپیوتر را خاموش کنید. یادتان باشد که اگر در موقع روشن بودن سیستم این کار را بکنید ممکن است به کامپیوتر یا آن دستگاه آسیب وارد شود.

تذکر: ممکن است شما در گذشته مانیتور را در حالتی که کامپیوتر روشن است جدا یا متصل کرده باشید و مشکلی پیش نیامده باشد، اما توصیه می شود که مانیتور را وقتی که کامپیوتر روشن است متصل یا جدا نکنید، زیرا امکان آسیب دیدن کامپیوتر یا مانیتور وجود دارد. توجه داشته باشید که استاندارد VGA چیزی در مورد hotpluggable بودن آن نمی گوید. البته در قطعات سخت افزاری جدید از دیودهای محافظ استفاده می شود، که در این صورت در صورت وجود آسیب، این دیودها خراب می شوند و قطعات داخلی سالم می مانند.

[ادامه دارد ...](#)