سامانه‌ی پرونده

شیوه‌ی نگهداری و دسترسی به پرونده‌ها در حافظه‌ی بلندمدت

آزمایشگاه سیستم عامل دانشگاه شهید چمران اهواز

وحید محمدی صفارزاده

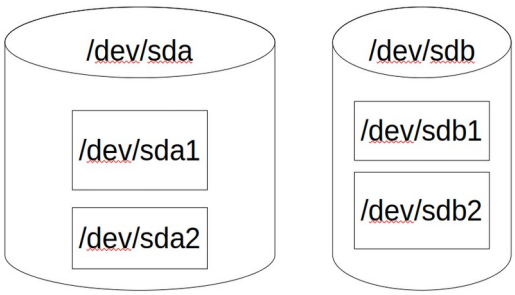
پس از پایان این آزمایش:

- **آشنایی با زیرساخت سامانه‌ی پرونده‌ها**
- **نقش هسته در ساخت و دستکاری سامانه‌ی پرونده‌ها**

پیش‌نیازها:

- **زبان سی**

در لینوکس دستگاه‌های حافظه‌ای با نام‌هایی مانند /dev/sda یا dev/sdb (و مانند آن‌ها) شناخته می‌شوند. برای نمونه، /dev/sda نماینده‌ی دیسک سخت (داده‌گاه پایا) است. اگر برای نمونه، یک حافظه‌ی فلش داشته باشیم با /dev/sdb نمایان می‌شود. این شیوه‌ی نام‌گذاری برای دستگاه‌ها همینگونه دنبال می‌شود؛ مانند /dev/sdc.



نگاره ۱: ساختار داده‌گاه پایا

۱.۱ دستگاه بسته‌ای

دستگاه‌های نگهداری داده از بخش‌هایی تک‌اندازه[1] ای به نام «بسته[2]» ساخته شده‌اند. به این‌ها «دستگاه بسته‌ای[3]» می‌گویند.

۱.۲ بُرش[4]

هر دستگاه نگهداری داده به بخش‌هایی جدا می‌شود که هر کدام را یک «بُرش» از حافظه می‌نامیم. هر برش را با یک شماره نشان می‌دهیم. برای نمونه، /dev/sda1 برش نخست روی دستگاه /dev/sda است. یا /dev/sdb2 برش دوم روی دستگاه /dev/sdb است. در نگاه هسته، هر برش یک «دستگاه بسته‌ای» است. راهنمای این ساختار برش‌داده‌شده در بخش کوچکی از حافظه به نام «جدول برش[5]» در دسترس است.

جدول برش می‌تواند از دو ساختار داشته باشد:

- ساختاری که در «آغاز دیسک سخت[6]» (ام.بی.ار) جای می‌گیرد. رایانه از همین جای حافظه آغاز به خواندن می‌کند و راه‌اندازی می‌شود. تکه[7]ی شماره‌ی «۰» از هر دیسک سخت همان ام.بی.ار است که برای راه‌اندازی رایانه به کار می‌رود. جدول برش‌ها در پایان این تکه جای می‌گیرد. جدول برش نشانی آغاز و پایان برش‌ها (دستور parted -l) را می‌دهد. یکی از برش‌ها برش «راه‌انداز[8]» است؛ همان برشی که دارای «راه‌انداز» است. سامانه‌ی ورودی/خروجی پایه[9]، در هنگام راه‌اندازی، ام.بی.ار را می‌خواند و اجرا می‌کند.

---

1 fixed-size
2 block
3 Block device
4 Partition
5 Partition Table
6 Master Boot Record (MBR)
7 Sector
8 active
9 BIOS: Basic Input-Output System

برنامه‌ی ام.بی.ار نخست برش راه‌انداز را پیدا می‌کند، «بسته»ی نخست آن را می‌خواند (بسته‌ی راه‌انداز[10]) و اجرایش می‌کند. برنامه‌ای که در بسته‌ی راه‌اندازی است سیستم‌عاملِ درون برش را بارگذاری می‌کند.

- ساختار دیگر، «جدول برش تک‌شناسه[11]» است که ساختاری نوین است و برخی کاستی‌های ساختار ام.بی.ار را ندارد.

هر برش می‌تواند برای خودش یک «سامانه‌ی پرونده» جداگانه داشته باشد، این سامانه یک پایگاه داده از پرونده‌ها و پوشه‌هاست که با برنامه‌های تراز کاربر با آن‌ها کار می‌کنیم. برای دسترسی به یک پرونده:

۱. باید برشی که فایل در آن است را از جدول برش پیدا شود.

۲. سپس، درون پایگاه داده‌ی سامانه‌ی پرونده‌ی آن برش به دنبال پرونده‌ی درخواستی جستجو شود.

چندین نرم‌افزار برای بررسی جدول برش‌ها در لینوکس داریم. در اینجا از ابزار parted[12] بهره می‌بریم. انجام نمونه‌ی این دستور در نگاره ۲ دیده می‌شود. می‌بینیم که دو دیسک سخت به نام‌های /dev/sda و /dev/sdb داریم. که دومی همان حافظه‌ی فلش است. همچنین، ساختار جدول برش با msdos نشان داده شده که همان ساختار ام.بی.ار است. برش‌های دیسک /dev/sda هشت تاست و دیگری تنها یک برش دارد. نشانی‌های آغازین و پایانی حافظه و اندازه‌ی آن‌ها نیز روشن است. این جدول برش دارای برش‌هایی از گونه‌های بنیادین یا پیوسته (Primary)، گسترشی یا ناپیوسته (Extended) و هست‌نما (Logical) است:

```
Model: ATA ST1000LM024 HN-M (scsi)
Disk /dev/sda: 1000GB
Sector size (logical/physical): 512B/4096B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system     Flags
 1      1049kB  577MB   576MB   primary   ntfs
 2      577MB   110GB   109GB   primary   ntfs
 3      110GB   215GB   105GB   extended
 6      110GB   111GB   999MB   logical   ext4
 7      111GB   111GB   538MB   logical   fat32           boot, esp
 8      111GB   199GB   87.3GB  logical   ext4
 5      199GB   215GB   16.0GB  logical   linux-swap(v1)
 4      215GB   1000GB  785GB   primary   ntfs


Model: VendorCo ProductCode (scsi)
Disk /dev/sdb: 15.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      32.8kB  15.7GB  15.7GB  primary  fat32        boot, lba
```

نگاره ۲: دستاورد دستور parted

- **برش بنیادین یا پیوسته:** به سادگی، یک زیربخش از دیسک سخت است.
- **برش گسترده یا ناپیوسته:** بخشی از دیسک سخت که خودش می‌تواند چند زیربخش داشته باشد.

10 Boot Block

11 Globally Unique Identifier Partition Table (GPT)

۱۲ نام ویرایش گرافیکی این ابزار gparted است. ابزار کاربردی دیگر fdisk هم هست.

- **برش هسته‌نما:** هر کدام از زیربخش‌های برش «گسترده» را یک برش هسته‌نما می‌نامیم.

در ساختار ام.بی.ار تنها تا چهار برش بنیادین می‌توان داشت و می‌توان برش‌های بیشتری از گونه‌ی هسته‌نما **درون** برش گسترده ساخت. ولی، در ساختار جدول برش تک‌شناسه چنین چالش و چارچوبی نداریم و بیش از چهار برش بنیادین می‌توانیم بسازیم و نیازی به برش هسته‌نما (و گسترده) نیست.

۱.۳ گونه‌های پرونده

سه گونه پرونده داریم: پرونده‌های ساده، پوشه‌ها و پیوندهای نمادین کار کرد. پرونده‌های ساده و پوشه‌ها روشن است. درباره‌ی پیوندها نیز در دنباله‌ی نوشته گفته خواهد شد.

۱.۴ سبک[۱۳]ها و چارچوب دسترسی[۱۴]

هر پرونده در لینوکس دارای دسته‌ای از چارچوب‌های دسترسی است که نشان می‌دهد که یک کاربر می‌تواند پرونده را بخواند، روی آن بخواند، یا آن را اجرا کند. دستور ls -l این چارچوب‌ها را نشان می‌دهد. خروجی مانند زیر به دست می‌دهد:

```
lrwxrwxrwx 1 oslab oslab 25 Nov 17_11:03 sym_link -> /home/oslab/original_dir/
```

نگاره ۳: آنچه دستور ls -l به دست می‌دهد.

چارچوب دسترسی به پرونده در بخش نخست این خروجی نمایش داده شده است.

گونه‌ی پرونده	دسترسی کاربر	دسترسی گروه	دیگر دسترسی‌ها
l	rwx	rwx	rwx

گونه‌ی پرونده می‌تواند پیوند نمادین (l)، یا پوشه (d) یا یک پرونده‌ی ساده (ـ)، لوله‌ی نامدار (p) باشد.

دسترسی‌ها نشان می‌دهد که هر تراز کاربری چگونه به پرونده دسترسی داشته باشد. هر چارچوب دسترسی دارای چهار نماد است:

- r نشان می‌دهد پرونده خواندنی است.
- w نشان می‌دهد که پرونده نوشتنی است.
- x نشان می‌دهد که پرونده اجرایی است.
- \- نشان‌دهنده‌ی چیزی نیست.

بودن هر کدام از این‌ها در رشته نماد «۱» و نبود آن‌ها نماد «۰» است. از این رو چارچوب‌های دسترسی سه عدد هستند که با سه بیت در مبنای ۲ نمایش داده می‌شوند. کاربر در اینجا همان کاربری است که دارنده‌ی پرونده است.

13 File Mode
14 Permission

در نمونه‌ی بالا کاربر oslab است. دسته‌ی دوم، گروه، گروه پرونده است. نشان می‌دهد که کاربرانی که در این گروه هستند چنین چارچوب دسترسی دارند. با اجرای دستور groups می‌توانید ببینید که کاربر شما در چه گروهی جای دارد. دیگران در یک سامانه با دسته‌ی سوم دسترسی‌ها به پرونده دسترسی دارند.

۱.۴.۱دستکاری دسترسی‌ها

برای دستکاری چارچوب‌های دسترسی دستور chmod به کار می‌آید. برای نمونه، برای افزودن توان خواندن یک پرونده به گروه دستور زیر را به کار می‌بریم.

```
$ chmod   g+r  <نام پرونده>
```

به جای g می‌توان از u برای کاربر و o برای دیگران بهره برد. همچنین، می‌توان دو تا از این‌ها را در کنار هم به کار برد و همزمان دسترسی‌ها را دگرگون کرد.

```
$ chmod   go+r  <نام پرونده>
```

رشته‌ی دودویی ساختار ۹ تایی چارچوب دسترسی را می‌توان با عدد در دستور chmod شناساند. برای نمونه، دستور زیر می‌تواند به کار رود:

```
$ chmod   644  <نام پرونده>
```

در اینجا ۶ همان رشته‌ی ۱۱۰ است که نشان‌دهنده‌ی rw_، و ۴ همان رشته‌ی ۱۰۰ که نشان‌دهنده‌ی r__ است.

پرونده‌های اجرایی که چارچوب دسترسی آن‌ها در گروه سرپرستی (ریشه_ادمین) است را باید با پیش‌دستور sudo اجرا نمود. این دستور کوتاه‌شده‌ی superuser do است و پس از آن از شما رمز کاربر سرپرست خواسته می‌شود.

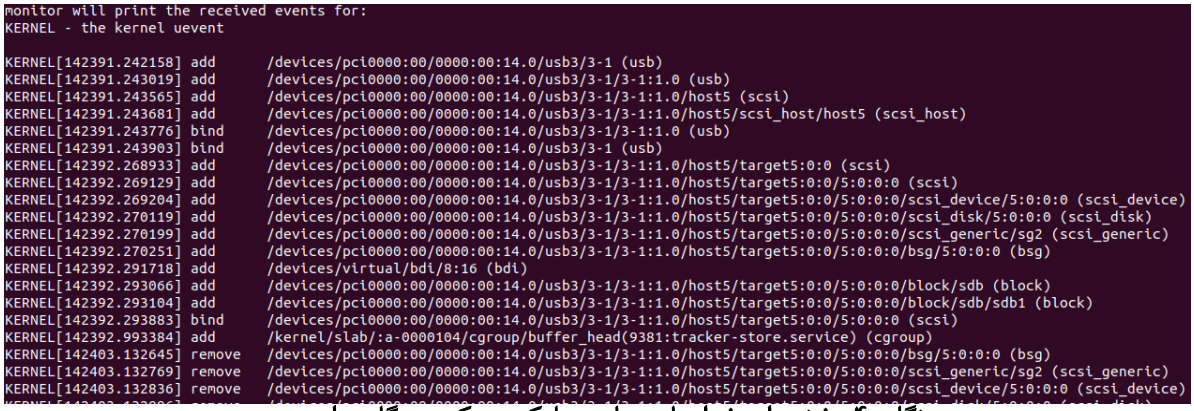
```
$ sudo <نام پرونده>
```

۱.۵ سامانه‌ی پرونده‌ها چیست؟

پیوند میان هسته و فرایندهای تراز کاربر با «سامانه‌ی پرونده‌ها[15]» انجام می‌شود. این همان بخشی است که ما با دستورهایی مانند ls و cd تاکنون با آن کار کرده‌ایم. همانگونه که پیشتر گفتیم، این سامانه یک «پایگاه داده» است و یک «دستگاه بسته‌ای» را مانند یک درختواره‌ی رده‌بندی‌شده از پوشه‌ها، زیرپوشه‌ها و پرونده‌ها برای کاربر فراهم می‌سازد.

سامانه‌ی پرونده به روش و ساختارهای داده‌ای گفته می‌شود که هر سیستم‌عامل به کمک آن‌ها می‌تواند جای فایل‌ها و داده‌ها را در حافظه‌ی بلندمدت (سخت‌دیسک) ردگیری کند. به زبانی دیگر، سامانه‌ی پرونده چشم سیستم‌عامل برای نگاه به داده‌های ۰ و ۱ انبارشده در حافظه‌ی بلندمدت است. هر سیستم‌عامل دارای سامانه‌ی پرونده‌ی خود

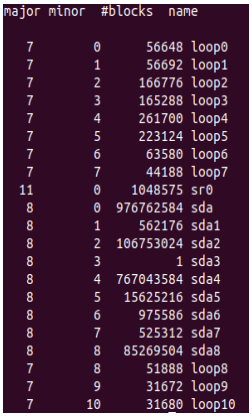
15 Filesystem



نگاره ۴: بخشی از رخدادهای زمان وصل‌کردن یک دستگاه یو.اس.بی

است؛ هر سیستم‌عامل هارد کامپیوتر را به شیوه‌ی ویژه‌ی خود می‌بیند. از همین رو، در ویندوز و لینوکس دارای سامانه‌ی پرونده‌های ناهمسانی هستیم.

هر گونه‌ی درختواره برای کاربرد ویژه‌ای است. برای نمونه، برخی از آنها برای کارهای پشتیبان‌گیری داده‌ها، برخی



نگاره ۵: داده‌های برش‌ها که در پرونده‌ی /proc/partitions نگه داشته می‌شود.

برای سرویس‌دهی به صورت آنلاین و وظایف دیگر بهینه شده‌اند.

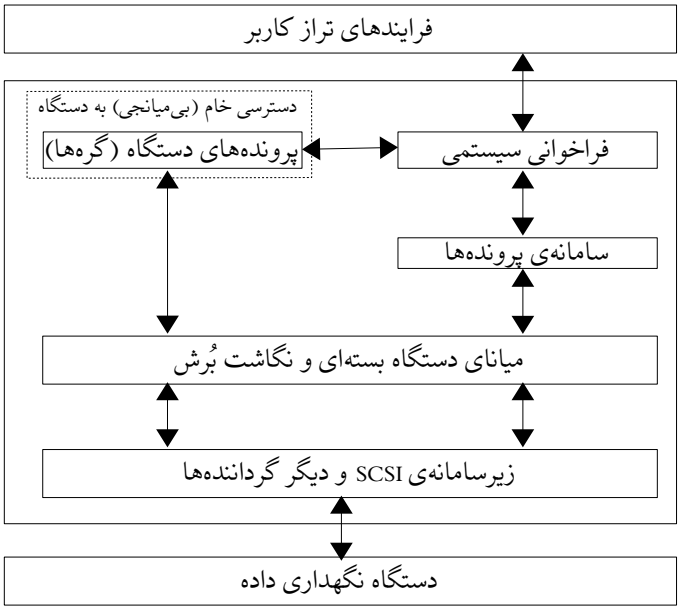
کار با میاناهای سیستمی در /sys و /proc از چیزهایی هستند که سامانه‌ی پرونده انجام می‌دهد. این سامانه در هسته پیاده‌سای می‌شود (اجرا در تراز هسته).

برای اینکه همه‌ی فرایندهای تراز کاربر دارای شیوه‌ای یکپارچه برای دسترسی به سامانه‌های پرونده داشته باشند، یک لایه‌ی دیگر به نام سامانه‌ی پرونده‌ی یکپارچه (VFS) نهاده شده است. با بودن این ابزار، فرایندهای تراز کاربر با همه‌ی پرونده‌ها به یک شیوه‌ی یکسان رفتار می‌کنند. برای نمونه، یک برنامه‌ی کارخواه[16] برای نوشتن در یک پرونده چه در رایانه‌ی خود و چه روی شبکه روش یکسانی را به‌کار می‌گیرد ولی، در لایه‌های زیرین شیوه‌ی دسترسی به این پرونده‌ها در سامانه‌ی پرونده یکسان نیست.

16 client

۱.۶ شیوه‌ی کارکرد هسته و دیسک سخت

روش دسترسی برنامه‌های تراز کاربر به داده‌ها در دیسک سخت در نگاره‌ی زیر نشان داده شده است.



۱.۷ دستکاری جدول برش

دستکاری جدول برش ساده است ولی، باید با هشیاری انجام شود تا دچار چالش نشویم. پس، اگر داده‌های باارزشی دارید نخست از آن‌ها پشتیبان بگیرید. همچنین، برای کار روی یک برش باید آن را از رایانه پیاده کنید[۱۷]. می‌توانید برای آزمایش، این کار را روی یک دستگاه یو.اس.بی انجام دهید.

می‌توان با ابزاری مانند parted و fdisk جدول برش را دستکاری کرد. این دو ابزار دارای ویژگی‌هایی همسان و ناسان با یکدیگر هستند. دگرگونی‌هایی که هر برنامه روی دیسک می‌تواند انجام دهد این‌ها هستند: ساخت، دستکاری و پاک‌کردن سامانه‌ی پرونده.

ویژگی‌های	parted	fdisk
ناسان	هر دگرگونی همان زمانی که در خواست می‌شود، روی دیسک انجام می‌شود.	تنها زمانی همه‌ی دگرگونی‌های درخواستی را انجام می‌دهد که از برنامه بیرون بیاییم.
	به ازای هر دگرگونی یک فراخوانی سیستمی به هسته می‌فرستد.	پس از بستن برنامه، تنها یک فراخوانی سیستمی می‌فرستد.
	-	هسته یک پیام گره‌زدایی در dmesg می‌نویسد.
همسان	هر دو جدول برش را در تراز کاربر دستکاری می‌کنند و نیازی به رفتن به تراز هسته نیست.	

با کمک دستور man می‌توانید این دو ابزار را بیشتر بررسی کنید.

برای دیدن رخدادهایی که در سامانه‌ی پرونده رخ می‌دهد می‌توان از دستور زیر بهره برد:

```
$ udevadm monitor -kernel
```

17 unmount

برای نمونه، نگاره ۴ بخشی از رخدادهایی را، که هنگام وصل‌کردن یک دستگاه یو.اس.بی رخ می‌دهد، نشان می‌دهد. همچنین، برای دیدن داده‌های برش‌های یک سامانه می‌توان پرونده‌ی proc/partitions/ را با دستوری مانند زیر بررسی کرد (نگاره ۵).

```
$ cat /proc/partitions
```

داده‌های دیگری را نیز می‌توان از sys/block/ به‌دست آورد.

۱.۸ گونه‌های سامانه‌ی پرونده

هرگونه سامانه‌ی پرونده دارای کارکرد ویژه‌ای است. گونه‌هایی مانند:

- چهارمین سامانه‌ی پرونده‌ی گسترده[۱۸] (ext4): دارای ویژگی «ردگیری[۱۹]» با توانایی پشتیبانی از پرونده‌ها و پوشه‌هایی با اندازه‌های بالا؛
- ایزو۹۶۶۰[۲۰] یک استانده‌ی سی.دی.رام است.
- و سامانه‌های دیگری مانند FAT و NTFS که بیشتر با ویندوز شناخته می‌شوند.

۱.۹ نشاندن یک سامانه‌ی پرونده

به فرایند گنجاندن یک پرونده‌ی سامانه در ساختار سامانه‌ی پرونده‌ی کنونی سیستم‌عامل «نشاندن[۲۱]» یا «سوارکردن» گوییم. هنگام بالاآمدن سیستم، هسته برخی داده‌های پیکربندی می‌خواند و سامانه‌ی پرونده‌ی «ریشه» (/) را بر پایه‌ی داده‌های پیکربندی را بارگذاری می‌کند. برای نشاندن یک پرونده‌ی سامانه باید این‌ها را دانست:

- دستگاهِ سامانه‌ی پرونده (مانند یک برش روی دیسک سخت، جایی که داده‌های سامانه‌ی پرونده‌ی اصلی جای دارند)؛
- گونه‌ی سامانه‌ی پرونده؛
- نشیمن[۲۲]: جایگاهی در ساختار درختی و رده‌بندی است که سامانه‌ی پرونده را روی آن سوار می‌کنیم (می‌نشانیم). برای نمونه، cdrom/ یک نشیمن برای دستگاه‌های CD-ROM است. هرجایی از ساختار درخت فایلی سیستم می‌تواند گزینه‌ای برای نشیمن باشد. همچنین می‌گوییم:

«یک دستگاه را در یک نشیمن می‌نشانیم.»

18 Fourth Extended filesystem (ext4)
19 Journaling
20 iso9660
21 mount
22 Mount point

دستور زیر نشان می‌دهد که چه دستگاه‌هایی در سامانه هم‌اکنون نشسته‌اند. هر خط برابر یک پرونده‌ی سامانه‌ی نشانده‌شده است.

```
$ mount
```

برای نشاندن یک پرونده‌ی سامانه، از دستور mount مانند زیر بهره می‌بریم:

```
$ mount  -t  <گونه‌ی سامانه‌ی پرونده>  <دستگاه>  <نشیمن>
$ mount  -t  ext4  /dev/sdf2  /home/extra
```

به فرایند جداکردن یک سامانه‌ی پرونده از درختواره‌ی سامانه‌ی پرونده «پیاده‌کردن[۲۳]» گوییم. برای پیاده‌کردن یک سامانه‌ی پرونده باید دستور زیر را به کار برد:

```
$ umount  <نشیمن>   یا   $ umount  <دستگاه>
```

۱.۱۰ شناسه‌ی یکتای پرونده‌ی سامانه[۲۴]

نام‌های دستگاه (device) شاید یکسان دربیایند. از این رو، دستوری مانند mke2fs در زمان ساخت یک پرونده‌ی سامانه، یک شماره به آن می‌دهد. فهرست دستگاه‌ها و شناسه‌های سامانه‌های پرونده را با دستور زیر می‌توان دید (نگاره ۶):

```
$ blkid
```

```
/dev/sda5: UUID="f7682836-ab0a-426f-9b5c-d0e13543690a" TYPE="swap" PARTUUID="52cf0800-05"
/dev/sda8: UUID="6056e41f-acaa-44d5-9e08-c8bdc36c35ba" TYPE="ext4" PARTUUID="52cf0800-08"
```

نگاره ۶: دستاورد دستور blkid

می‌توان از این شناسه برای نشاندن یک سامانه‌ی پرونده بهره برد:

```
$ mount UUID=”2c73b02a-570f-4c39-b282-446c7df8d4cf”  /home/extra
```

برای دستکاری این شناسه از دستور tune2fs می‌توان بهره برد.

۱.۱۱ درون یک پرونده‌ی سامانه

دو بخش پایه‌ای سامانه‌ی پرونده در یونیکس این‌ها هستند:

- استخری از بسته‌های داده، جاهایی که داده‌ها نگهداری می‌شوند.
- یک پایگاه داده که این استخر داده را می‌گرداند. پایگاه داده پیرامون یک ساختار به نام inode (گره نمایه[۲۵]) می‌چرخد.

23 unmount
24 Universally Unique Identifier (UUID)
25 index node (inode)

هر نمایه دسته‌ای از داده‌هاست که ویژگی‌های یک فایل مانند گونه‌ی آن، پروانه‌ها، و به‌ویژه جای داده‌های فایل در استخر داده را در بر دارد. نمایه‌ها شماره‌هایی دارند که در یک «جدول نمایه» نگهداری می‌شوند. نمایه‌ی یک پوشه دارای فهرستی از نام پرونده‌ها و پیوند[۲۶]های آن‌ها به دیگر نمایه‌هاست. نمایه برای ردگیری این است که کدام «بسته»ی داده از آنِ کدام «پرونده» است.

برای کار روی سامانه‌ی پرونده‌ها دو راه داریم:

۱. انجام کارها روی یک دیسک فلش

۲. بخش‌بندی دیسک سخت ماشین مجازی به دو بخش.

کارهای زیر را انجام می‌دهیم:

۱. یک سامانه‌ی پرونده تازه روی حافظه‌ی فلش در نشانی /dev/sdb می‌سازیم.

۲. آن را در یک نشیمن سوار می‌کند (می‌نشانیم با mount).

۱.۱۱.۱شناسه‌ی پرونده

مانند هر فرایند که در سیستم‌عامل دارای یک شناسه است، به هر پرونده‌ای که «باز می‌شود» نیز یک شماره به نام «شناسه‌ی پرونده[۲۷]» داده می‌شود. «داده‌های درون یک پرونده» و «داده‌های درباره‌ی آن پرونده» یکسان نیستند. همه‌ی پرونده‌ها دارای دنباله‌ای از «۸تایی[۲۸]»ها هستند، به جز پرونده‌های دستگاهی[۲۹] و پرونده‌های ویژه‌ی سامانه‌ی پرونده. هر پرونده خودش دارای داده‌های پایشی، مانند اندازه‌اش و جداکننده‌ی پایان پرونده[۳۰] (EOF) است.

همه‌ی داده‌هایی که سامانه‌ی پرونده نیاز دارد تا بداند که با پرونده چگونه رفتار کند در یک ساختمان داده به نام «گره نمایه» جای گرفته است. هر پرونده دارای گره نمایه‌ی خود است، که سامانه‌ی پرونده برای شناسایی پرونده از آن بهره می‌برد. در استانده‌ی پوزیکس داده‌های زیر برای یک پرونده در سامانه‌ی پرونده فراهم شده است:

- گونه‌ی پرونده (ساده، پوشه، پیوند نمادین، لوله و لوله‌ی بی‌نام، پایانه‌ی شبکه[۳۱])
- شمار پیوندهای سختِ[۳۲] هر پرونده
- اندازه‌ی پرونده به بایت (۸تایی)
- شناسه‌ی دستگاه (شناسه‌ای برای دستگاهی که پرونده در آن جای دارد.)

26 link
27 File Descriptor
28 Byte
29 Device file
30 End-of-File (EOF)
31 socket
32 Hard link

- شماره‌ی گره نمایه که پرونده را در سامانه‌ی پرونده می‌شناساند
- شناسه‌ی کاربر (UID) دارنده‌ی پرونده
- شناسه‌ی گروه کاربری[۳۳] پرونده
- چندین برچسب زمانی[۳۴]
- پروانه‌های دسترسی و سبک پرونده

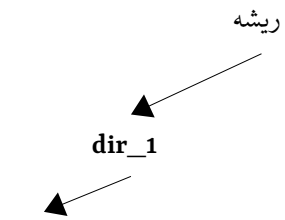
۱.۱۱.۲ساخت سامانه‌ی پرونده

هرگاه یک دستگاه حافظه‌ی خام سوار می‌کنیم نیاز است که سامانه‌ی پرونده‌ای را روی آن بسازیم. ساخت یک سامانه‌ی پرونده روی دستگاه کنونی همه‌ی داده‌ها را پاک می‌کند. در تراز کاربر می‌توانیم یک گونه از پرونده‌ی سامانه را بسازیم. این را با ابزار mkfs می‌توان انجام داد. برای نمونه، برای ساخت یک برش از گونه‌ی ext4 روی /dev/sdf2 دستور زیر را به کار می‌بریم:

```
$ mkfs -t ext4 /dev/sdf2
```

این دستور خودش پیش‌فرض‌هایی را می‌گذارد، ولی، شما نیز می‌توانید آن‌ها را دستکاری کنید.

۱.۱۱.۳ساخت یک نمونه



نگاره ۷: یک ساختار نمونه‌ی درخت پرونده

درخت بالا در تراز کاربر را با دستورهای زیر می‌سازیم.

```
$mkdir dir_1
$echo abc > dir_1/file_1
```

نگاره ۸: دستورهای ساخت درخت بالا

اکنون می‌خواهیم ببینیم ساختار گره‌های نمایه این ساختار درختی چگونه است. دستور زیر برای دیدن داده‌های نمایه‌ی پوشه‌ی کنونی است:

```
$ ls -i
```

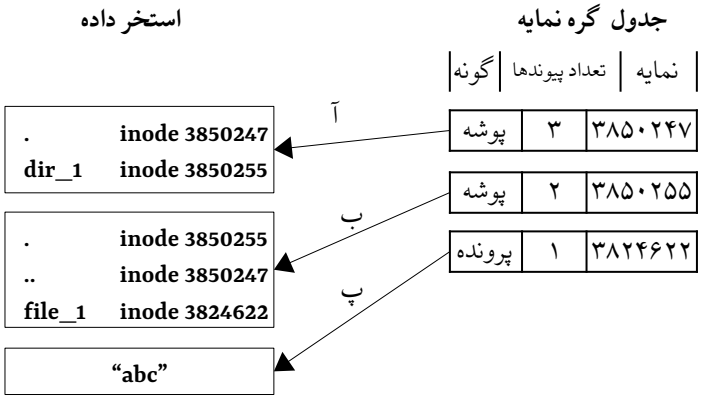
33 User group ID
34 Timestamps

برای درخت بالا دستورهای زیر را انجام می‌دهیم:

```
$ls -i
3850255 dir_1
$ls -i dir_1
3824622 file_1
$ls -i ..
```

نگاره ۹: داده‌های نمایه‌های درخت پرونده‌ها

بر پایه‌ی دستورهای بالا و نیز داده‌های درون پرونده‌ها درمیابیم که پرونده‌ها و پوشه در ساختارهایی مانند زیر جای دارند.



نگاره ۱۰: جدول گره‌های نمایه و استخر داده

هسته چگونه به پرونده‌ی dir_1/file_2 دست می‌یابد:

۱. نخست، بخش‌های نشانی را وارسی می‌کند؛ نام پوشه: dir_1، و نام پرونده: file_1

۲. با پیگیری گره نمایه (inode) ریشه به داده‌های پوشه‌ی درون آن دست می‌یابد (بخش آ در نگاره ۱۰).

۳. نام dir_1 را در داده‌های پوشه می‌یابد، نمایه‌ی این پوشه ۳۸۵۰۲۵۵ است (بخش آ در نگاره ۱۰).

۴. به دنبال نمایه‌ی ۳۸۵۰۲۵۵ در جدول نمایه‌ها می‌گردد و در می‌یابد که نمایه از گونه‌ی «پوشه» است.

۵. گره با نمایه‌ی ۳۸۵۰۲۵۵ را نیز پی می‌گیرد تا به داده‌های این پوشه در استخر داده‌ها برسد (بخش ب در نگاره ۱۰).

۶. بخش دوم نشانی (file_1) را پیدا می‌کند که گره با نمایه‌ی ۳۸۲۴۶۲۲ را نشان می‌دهد.

۷. در جدول نمایه‌ها به دنبال نمایه‌ها ۳۸۲۴۶۲۲ می‌گردد و در می‌یابد که یک پرونده است. در اینجا هسته با داشتن اینکه یک فایل (پرونده) است می‌تواند با پی گرفتن پیوند داده‌های این نمایه آن داده را بخواند که abc است (بخش پ در نگاره ۱۰).

۱.۱۲ پیوندهای نمادین

یک «پیوند نمادین[۳۵]» یک «پرونده» است که نماینده‌ی یک «پرونده» یا «پوشه»ی دیگر است. با ساخت یک پیوند نمادین یک میانبر[۳۶] به یک پرونده‌ی دیگر درست کرده‌ایم. کاربرد این پیوندها دسترسی به پرونده‌ها یا پوشه‌هایی است که نشانی‌های ساده‌ای ندارند. این پیوندهای نمادین به‌سادگی نام‌ها ونشانی‌هایی هستند که نماینده‌ی دیگر نام‌ها و نشانی‌ها هستند. برای نمونه، با دستور ls -l برای یک پرونده به نام sym_link چیزی مانند زیر به دست می‌آید. این نشان می‌دهد که دسترسی به پرونده‌ی sym_link همان دسترسی به پوشه‌ی /home/oslab/original_dir خواهد بود.

```
lrwxrwxrwx 1 oslab oslab 25 Nov 17_11:03 sym_link -> /home/oslab/original_dir/
```

نگاره ۱۱: نمونه‌ای از پیوند نمادین

۱.۱۲.۱ ساخت پیوند نمادین

از دستور ln در لینوکس مانند زیر برای ساخت پیوند نمادین است.

```
$ ln  -s      <نشانگاه>      <نام پیوند>
```

برای نمونه پیوند نگاره ۱۱ اینگونه به‌دست آمده است:

```
$ ln  -s      /home/oslab/original_dir    sym_link
```

نبودن s- در دستور یک «پیوند سخت[۳۷]» را به جای پیوند نمادین می‌سازد. در پیوند سخت تنها «**یک نام دیگر**» به «یک پرونده» داده می‌شود.

۱.۱۲.۲ تعداد پیوندها

همانگونه که در نگاره ۱۰ می‌بینیم، هر گره نمایه دارای یک گزینه به نام «تعداد پیوندها[۳۸]» است. برای یک پرونده، این گزینه نشان‌دهنده‌ی این است که چند پیوند سخت به آن پرونده ساخته شده است به‌همراه خود پرونده. پس، در نگاره ۱۰ این تعداد برای گره نمایه‌ی file_1 یک است، چون تنها خودش است و پیوندی به آن ساخته نشده است. ولی، این تعداد برای یک پوشه در آغاز ساخت پوشه «۲» و با افزودن هر زیرپوشه یکی زیاد می‌شود. از این‌رو برای پوشه‌ی ریشه (که درخت پرونده‌ای در آن ساخته شده) این تعداد برابر ۳ (چون یک زیر پوشه دارد) و برای dir_1 برابر ۲ است چون زیرپوشه ندارد. برای پیدا کردن تعداد پیوندهای پوشه‌ها و پرونده‌های درون یک پوشه می‌توان از دستور زیر بهره برد.

```
$ ls   -dl    <نام پوشه یا پرونده>
```

برای نمونه دنباله‌ی دستورهای زیر و برایند آن‌ها را در نگاره ۱۲ بررسی کنید.

35 Symbolic Link
36 Alias or Shortcut
37 Hard link
38 link count

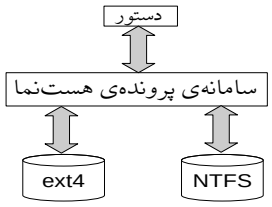
```
$ ls -dl oslab_fs/
drwxrwxr-x 3 oslab oslab 4096 Nov 20 19:27 oslab_fs/
$ ls -dl oslab_fs/dir_1/
drwxrwxr-x 2 oslab oslab 4096 Nov 20 19:28 oslab_fs/dir_1/
$ ls -dl oslab_fs/dir_1/file_1
-rw-rw-r-- 1 oslab oslab 4 Nov 20 19:28 oslab_fs/dir_1/file_1
$ ln oslab_fs/dir_1/file_1 oslab_fs/link_to_file_1
$ ls -dl oslab_fs/dir_1/file_1
-rw-rw-r-- 2 oslab oslab 4 Nov 20 19:28 oslab_fs/dir_1/file_1
$
$ ln -s oslab_fs/dir_1/file_1 oslab_fs/link_to_file_1_2
$ ls -dl oslab_fs/dir_1/file_1
-rw-rw-r-- 2 oslab oslab 4 Nov 20 19:28 oslab_fs/dir_1/file_1
```

نگاره ۱۲: دنباله‌ی دستورهایی که نشان‌دهنده‌ی «تعداد پیوندها» برای پرونده‌ها و پوشه‌هاست.

خروجی دستور ls -dl دارای چندین بخش است. بخش نخست که نشان‌دهنده‌ی پروانه‌های دسترسی به پوشه یا پرونده است (در آینده بررسی خواهد شد.) شماره‌ای که در بخش دوم آمده همان تعداد پیوندهاست. برای نمونه، تعداد پیوندهای پوشه‌ی oslab_fs شده است ۳؛ چراکه در آغاز ۲ است و یک زیرپوشه نیز دارد که می‌شود ۳. همچنین، می‌بینید که پرونده‌ی file_1 پیش از ساخت پیوند سخت link_to_file_1 برابر ۱ و پس از ساخت این پیوند برابر ۲ می‌شود.

۱.۱۳ سامانه‌ی پرونده‌ی هسته‌نما (یکپارچه)[۳۹]

سامانه‌ی پرونده‌ی هسته‌نما، سامانه‌ی پرونده‌ای است که دارای ساختاری نیست که خودش با پرونده‌ها کار کند. این سامانه یک برنامه‌ی **تراز هسته** است که همه‌ی فراخوانی‌های سیستمی که با سامانه‌های پرونده‌ای لینوکس کار دارند را پاسخ می‌دهد. ویژگی پایه‌ی این برنامه فراهم‌شدن یک میانآی یکپارچه برای دسترسی به همه‌ی گونه‌های سامانه‌های پرونده است. این میانآ به هسته این توانایی را می‌دهد که بتواند روش‌های دسترسی به گونه‌های سامانه‌ی پرونده داشته باشد.



نگاره ۱۳

۱.۱۳.۱فراخوانی‌های سیستمی سامانه‌ی پرونده‌ی هسته‌نما

برنامه‌ی تراز هسته‌ی سامانه‌ی پرونده‌ی هسته‌نما به چندین فراخوانی سیستمی که با سامانه‌ی پرونده‌ها کار می‌کنند رسیدگی می‌کند. فراخوانی‌هایی mount و umount پیشتر گفته شد. برخی فراخوانی‌های دیگر که با آن‌ها با کمک

39 Virtual File System (VFS)

دستورها یا تابع‌ها کار کرده‌اید نیز در جدول آمده است. برخی از فراخوانی‌هایی که در اینجا آمده است دارای ویرایش‌های دیگری نیز هستند.

نام فراخوانی	کارکرد	نام فراخوانی	کارکرد
chdir(), fchdir()	تغییر پوشه‌ی کنونی	chmod(), fchmod(), utime()	دستکاری ویژگی‌های پرونده‌ها
mkdir(), rmdir()	ساخت و پاک‌کردن پوشه	stat(), access(), fstat()	خواندن ویژگی‌های پرونده‌ها
rename(), unlink()	دستکاری درون‌شو[40]های پوشه‌ها	open(), close(), creat(), umask()	بازکردن، بستن وساخت پرونده
readlink(), symlink()	دستکاری پیوندها	dup(), fcntl()	دستکاری توصیف‌گر پرونده
chown()	دستکاری دارنده‌ی پرونده	select(), poll()	ماندن برای دسته‌ای از رخدادها بر توصیف‌گر پرونده
truncate()	دگرگونی اندازه‌ی پرونده	lseek()	دگرگونی نشان‌گر پرونده
read(), write(), sendfile()	انجام کارکردهای ورودی/خروجی	mmap(), madvise(), mincore()	گرداندن نگاشت حافظه
sync(), flock()	همزمان‌سازی داده‌های پرونده	setxattr(), removexattr()	دستکاری ویژگی‌های گسترده‌ی پرونده‌ها
ioperm(), ioctl(), mknod()	کار روی دستگاه‌ها	pipe()	کار با لوله‌ها
socket(), connect(), bind()	کار با سوکت‌ها و برای پیاده‌سازی کارهای شبکه		

فرایندها تنها می‌توانند به پرونده‌های «باز» دسترسی داشته باشند. دستور زیر در زبان سی برای باز کردن پرونده است:

fd = open(نشانی, پرچم, سبک)

که سه آرگومان دارد: **نشانی**: نشانی پرونده‌ای که می‌خواهیم باز شود. **پرچم:** نشان می‌دهد که پرونده چگونه باید باز شود (برای خواندن، نوشتن، خواندن/نوشتن، افزودن). همچنین، می‌توان با آن برگزید که پرونده‌ای که نیست ساخته شود. **سبک:** شیوه‌ی دسترسی به پرونده را برمی‌گزیند.

این فراخوانی پایین‌رده یک «ساختمان (شی) پرونده» را «می‌سازد» و یک «شناسه‌ی پرونده» برای آن برمی‌گرداند. هر ساختمان پرونده دارای برخی داده‌های پرونده و نیز نشانگرهایی به تابع‌های هسته است که فرایند می‌تواند فرابخواند.

هر شناسه‌ی پرونده نماینده‌ی پیوستگی میان یک فرایند و یک پرونده‌ی باز است. ساختمان پرونده نیز دارای داده‌هایی در زمینه‌ی این پیوستگی است. گاهی چندین شناسه‌ی پرونده در یک فرایند نماینده‌ی تنها یک پرونده‌ی باز هستند. همچنین، گاهی چند فرایند همروند[41] یک پرونده را باز می‌کنند. در چنین زمان‌هایی، سامانه‌ی پرونده به هر کدام از آن پرونده‌های باز یک شناسه‌ی پرونده‌ی جداگانه می‌دهد و با هر کدام نیز یک ساختمان (شی) پرونده‌ی جداگانه همراه می‌سازد. از این رو، سامانه‌ی پرونده‌ی لینوکس هیچگونه هماهنگی[42] برای کارهای ورودی/خروجی،

40 entry
41 Concurrent
42 synchronization

انجام‌شده به‌دست فرایندها روی آن پرونده‌ی یکسان، انجام نمی‌دهد. ولی، چند فراخوانی سیستمی مانند flock() هستند تا با کمک آن‌ها فرایندها بتوانند خودشان را روی بخشی یا همه‌ی پرونده با هم هماهنگ کنند. برای ساخت پرونده، گاهی فراخوانی سیستمی create() به‌کار رود که هسته با آن درست مانند open() برخورد می‌کند.

۱.۱۴ برخی ابزارهای کاربردی: ابزار tree

با دستور tree می‌توانید ساختار یک زیرشاخه از پوشه‌ها و پرونده‌ها در سامانه‌ی پرونده را مانند یک درخت در پایانه‌ی دستوری (ترمینال) نمایش دهید. این برنامه را با دستور زیر می‌توانید نصب کنید:

```
$ sudo apt install tree
```

برای نمونه یک ساختار رده‌بندی پرونده‌ای را با دستورهای زیر می‌سازیم.

```
:~$ mkdir oslab_fs
:~$ cd oslab_fs/
:~/oslab_fs$ mkdir dir_1 dir_2
:~/oslab_fs$ echo "Vahid" > dir_1/file_1
:~/oslab_fs$ echo "32" > dir_1/file_2
:~/oslab_fs$ mkdir dir_2/dir_3
:~/oslab_fs$ echo "Ahvaz" > dir_2/dir_3/file_4
:~/oslab_fs$ echo "SCU" > dir_2/file_3
```

اجرای دستور tree در پوشه‌ی oslab_fs چنین خروجی را به دست می‌دهد:

```
.
├── dir_1
│   ├── file_1
│   └── file_2
└── dir_2
    ├── dir_3
    │   └── file_4
    └── file_3

3 directories, 4 files
```

نگاره ۱۴: دستاورد دستور tree

۱.۱۴.۱دستور df

این دستور نمایش‌دهنده‌ی بخش‌های حافظه است. سامانه‌های پرونده نشانده‌شده و نشانده‌نشده، پوشه‌ها، دستگاه‌های نشانده‌شده، اندازه‌ی حافظه‌ی پرشده و اندازه‌ی حافظه‌ی مانده و اینکه حافظه‌ی سامانه‌ی پرونده چگونه به‌کار رفته است. هنگامی که به دستور df نشانی یک پرونده یا پوشه را می‌دهید می‌توانید بدانید که آن پرونده روی کدام دستگاه جای دارد.

۱.۱۴.۲دستور ls

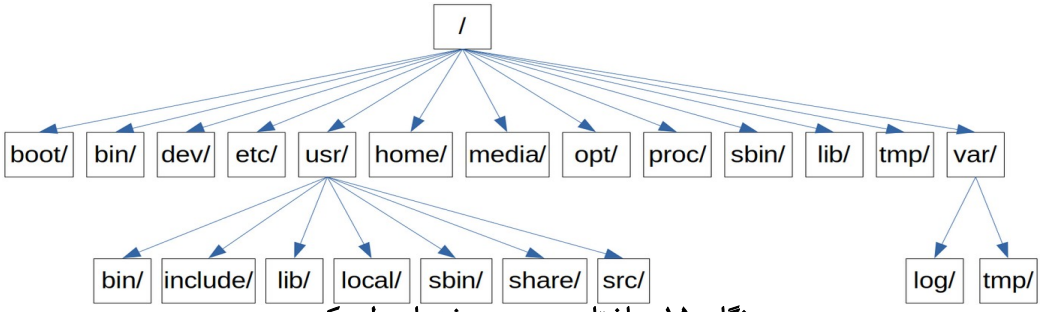
این دستور گزینه‌های زیادی برای نمایش داده‌ها درباره‌ی پرونده‌ها و پوشه‌ها دارد. با انجام دستور زیر می‌بینیم که گزینه‌ی l- داده‌های بیشتری در زمینه‌ی پرونده‌های درون پوشه می‌دهد.

```
$ ls -l
```

۱.۱۵ پروانه‌های دسترسی به پرونده‌ها و پوشه‌ها

۱.۱۶ ساختار رده‌بندی سامانه‌ی پرونده در لینوکس

در نگاره‌ی زیر ساختار درختی پوشه‌ها و پرونده‌ها در لینوکس را می‌بینیم.



نگاره ۱۵: ساختار رده‌بندی پوشه‌ها در لینوکس

در جدول زیر کارکرد این پوشه‌ها گفته شده است.

پوشه	کارکرد
/bin	فایل‌های اجرایی برنامه‌های لینوکس در اینجا هستند (مانند cp و ls). بیشتر آن‌ها کامپایل‌شده‌ی برنامه‌ی نوشته‌شده به زبان سی هستند و برخی دیگر برنامه‌های نوشتاری (اسکریپتی) هستند.
/boot	پرونده‌های راه‌انداز هسته در این پوشه است. این‌ها تنها برای نخستین گام بالاآمدن هسته است و در اینجا داده‌ای در زمینه‌ی چگونگی بالاآمدن سرویس‌ها پیدا نمی‌کنید.
/dev	این‌ها پرونده‌هایی هستند که به گرداننده‌های دستگاه‌ها راه دارند. دستگاه‌ها یا سخت‌افزارهایی مانند دیسک سخت، سی.دی.رام، گذرگاه‌ها (ports).
/etc	دارای رمزهای کاربر و نیز پرونده‌های پیکربندیِ راه‌اندازی (بوت)، دستگاه‌ها، شبکه و ... است.بسیاری از گزینه‌ها در این پوشه ویژه‌ی سخت‌افزارهای رایانه‌ی شما هستند. برای نمونه، پوشه‌ی /etc/X11 دارای پیکربندی‌های کارت گرافیک و سیستم پنجره‌بندی هستند. همچنین، پرونده‌ی /etc/fstab برای پیکربندی بارگذاری «بُرش‌ها» در زمان بالاآمدن هستند.
/home	دارای پوشه‌های جداگانه و ویژه برای هر کاربر است.
/lib	کوتاه‌شده‌ی library. دارای برنامه‌هایی است که برنامه‌های اجرایی از آن‌ها بهره می‌برند. این کتابخانه‌ها دو گونه دارند: ایستا و همووندی. پوشه‌ی /lib بیشتر دارای کتابخانه‌های همووندی است ولی دیگر پوشه‌ها مانند /usr/lib دارای هر دو گونه‌ی کتابخانه و دیگر پرونده‌های کمکی است.
/media	پایه‌ای‌ترین نشیمن برای رسانه‌های جداشوند مانند درایو فلش یا پوشه‌های همووندشده (shared) میان ماشین مهمان و میزبان در VirtualBox
/opt	برخی از نرم‌افزارهای دیگر بیرون از نرم‌افزارهای پایه‌ای لینوکس (Third-Party Software) در این پوشه جای می‌گیرند. بسیاری از سامانه‌ها کاری با این پوشه ندارند.
/proc	آمارهای سامانه را در یک میانآی پوشه و پرونده فراهم می‌کند. داده‌های درباره‌ی فرایندهای کنونی سامانه و برخی گزینه‌های

		هسته. هر فرایند در این پوشه دارای یک زیرپوشه‌ی جداگانه برای خود و با شناسه‌ی فرایند خود است.
/sys		این پوشه همانند proc/ ولی برای داده‌های دستگاه‌ها و میانآی سامانه است.
/sbin		مانند پوشه‌ی bin/ این پوشه نیز دارای پرونده‌های اجرایی برنامه‌هاست، ولی، این برنامه‌ها تنها با داشتن دسترسی کاربر سرپرست (ریشه ــ مدیر) انجام می‌شوند.
/tmp		پوشه‌ای برای نگهداری پرونده‌های بی‌ارزش و کوچکتر. هر کاربر می‌تواند به این پوشه بنویسید و از آن بخواند ولی کاربران نمی‌توانند به پرونده‌های یکدیگر دسترسی داشته باشند. بسیاری از برنامه‌ها از این پوشه برای جای کاری خود بهره می‌برند. در بسیاری از ویراست‌های لینوکس این پوشه با هر بار خاموش و روشن‌شدن پاکسازی می‌شود.
/usr		یک ساختار رده‌بندی پوشه‌ای است که دارای توده‌ای از پوشه‌های سیستمی لینوکس است. بسیاری از پوشه‌ها در usr/ هم‌نام پوشه‌های درون‌ریشه (/) هستند (مانند usr/bin/ و usr/lib/) و همان گونه‌های پرونده را نیز دارا هستند. پس پرونده‌های و پوشه‌های سیستمی در چند پوشه پخش شده‌اند.
	/usr/include	دارای پرونده‌های سربرگ (header) است که کامپایل‌گر زبان سی به کار می‌گیرد.
	/usr/local	جایی است که سرپرستان (admin) سامانه نرم‌افزارهای خودشان را جای‌گذاری می‌کنند و بهتر است ساختاری مانند / یا usr/ داشته باشد.
	/usr/src	برنامه‌ی (پرونده‌های نوشتاری) سیستم عامل در این پوشه هستند.
/var		زیرپوشه‌ای با پرونده‌های دگرگون‌شونده جایی که برنامه‌ها داده‌های زمان اجرای خود را آنجا نگه می‌دارند. رویدادنگاری سامانه‌ای، ردگیری کاربر، حافظه‌های نهان (caches) و دیگر پرونده‌های سیستمی. این پوشه یک زیرپوشه به نام var/tmp/ دارد که این پوشه مانند tmp/ در زمان بالاآمدن سامانه پاک نمی‌شود.

۱.۱۷ دستورکار

۱. نقطه «.» در بالای درخت نمایش‌داده‌شده در نگاره ۱۴ نشان‌دهنده‌ی چیست؟

۲. ساختار درختی در نگاره ۱۴ را در سامانه‌ی پرونده‌ی سیستم خود بسازید. سپس، مانند نگاره ۱۰، جدول گره‌های نمایه و استخر داده‌ی آن را بسازید (در گزارش کار بیاورید و شیوه‌ی بدست‌آوردن آن را در ارایه بگویید.)

۳. برپایه‌ی کاری که در پرسش ۲ انجام دادید، **برنامه‌ای** بنویسید که با دریافت نشانی یک پوشه، همه‌ی گره‌های نمایه و استخر داده‌ی آن را بسازد.

۴. **برنامه‌ای** بنویسید که با دریافت نشانی یک پرونده، هر ۷ گامی را که هسته (زیربخش ۱,۱۱,۳) برای دسترسی به پرونده انجام می‌دهد را انجام دهد تا به پرونده برسد. (برنامه باید در هر گام آنچه را به دست می‌آورد نمایش دهد.)

۵. بگویید هر کدام از پوشه‌های زیر در کدام پوشه یا روی کدام دستگاه «نشانده» شده‌اند:

/sbin, /home, /usr, /usr/share

۶. دستور ls -l را در یک پوشه انجام دهید. عددی که پس از واژه‌ی total آمده است چیست؟

۷. با کمک دستور mkdir یک پوشه‌ی تهی در پوشه‌ی خانه‌ی (home) خود بسازید. دستور ls -l چه به شما داده است؟

- چرا در خروجی این دستور، یک عدد «۲» پس از چارچوب دسترسی‌ها و پیش از نام دارنده آمده است؟
- با دستور cp رونوشت از یک پرونده به درون پوشه‌ی ساخته‌شده بیاورید. و بررسی کنید که چه به سر عدد «۲» می‌آید. آیا عدد «۲» تغییر کرد؟ چرا؟
- یک پوشه در این پوشه‌ی تازه بسازید. (پس، هم‌اکنون شما یک پوشه دارید که دارای یک زیرپوشه و یک پرونده است). همان عدد را دوباره بررسی کنید.چه رخ داده است؟
- با دستور cd به درون پوشه‌ای که تازه ساخته‌اید بروید و دستور ln -s را برای ساخت یک پیوند نمادین برای پرونده‌ای که در این پوشه است به کار ببرید.
- آیا پرونده‌ی اصلی نشانه‌ای از پیوند نمادین ساخته‌شده دارد؟ (می‌توانید با دستور ls بررسی کنید)
- برای زیرپوشه‌ای که در این پوشه ساخته‌اید نیز یک پیوند نمادین بسازید. و باز هم نشانه‌ی پیوند ساخته‌شده را در پوشه‌ی اصلی بررسی کنید.
- تلاش کنید که برای پرونده‌ای که وجود ندارد یک پیوند نمادین یا پیوند سخت بسازید. چه رخ می‌دهد؟

۸. یک پرونده‌ی نوشتاری (متنی) به نام oslab بسازید. دستور زیر را اجرا کنید.

```
$ ln -s oslab oslab
```

- تلاش کنید که فایل oslab ساخته‌شده را با دستور cat بخوانید. چه می‌شود؟ توضیح دهید.
- هم‌اکنون یک پیوند سخت (hard link) بسازید. (ln را این‌بار بدون s- به‌کار ببرید). چه رخ می‌دهد؟
- چارچوب‌های دسترسی پرونده‌ی اصلی را با دستور chmod تغییر دهید. اکنون، نوشته‌های درون پرونده‌ی اصلی را دستکاری کنید. آیا تغییری در خروجی دستور chmod برای این پرونده رخ می‌دهد؟
- زمانی که پرونده‌ی اصلی را پاک می‌کنید چه رخ می‌دهد؟

۹. تلاش کنید که برای یک پوشه یک پیوند سخت بسازید. چه می‌شود؟

۱۰. تلاش کنید برای پرونده‌ای که وجود ندارد یک پیوند سخت بسازید. نتیجه را توضیح دهید؟

۱.۱۸ پرسش‌هایی برای درک ژرف‌تر

۱. نقش «سامانه‌ی پرونده‌ی هست‌نما (یکپارچه_مجازی)» در کار با سامانه‌ی پرونده چیست؟

۲. دستور chown برای چه کاریست؟

۳. فراخوانی سیستمی stat را بررسی کنید. رابطه‌ی inode و stat چیست؟