Linux Professional Institute

LPIC-1

جلسه یازدهم: ژورنالینگ، مدیریت ساعت و زمان‌بندی عملیات

در این جلسه:

ویدئو اول:

- یادآوری ساختار لاگ و Syslog
- صحبت در مورد دستور `logger`
- صحبت در مورد Rotate کردن فایل‌های لاگ (ما در جزوه‌ی جلسه‌ی قبل به آن پرداختیم.)
- تلاش (و عدم موفقیت) برای نصب و استفاده از Graylog (ما به توضیح این بخش نمی‌پردازیم.)

ویدئو دوم:

- صحبت در مورد تنظیم زمان سیستم
- آشنایی با دستور `date`
- آشنایی با پروتکل NTP
- صحبت در مورد سرورهای NTP
- آشنایی با دستور `timedatectl`
- آشنایی با زمان‌بندی عملیات با `cron`
- آشنایی با `anacron`
- آشنایی با `at`

فهرست مطالب

مقدمه ۱

ژورنالینگ با *systemd-journald* ۱

چگونگی عملکرد *systemd-journald* ۱

تنظیم *systemd-journald* ۲

پیدا کردن فایل‌های ژورنال ٤

مشاهده‌ی فایل‌های ژورنال با *journalctl* ۵

مدیریت و نگهداری فایل ژورنال ۸

لایه‌بندی لاگ‌ها ۹

نوشتن موارد جدید درون ژورنال ۱۰

مدیریت زمان سیستم ۱۱

مفاهیم اولیه زمان در لینوکس ۱۱

زمان محلی (localtime) و زمان هماهنگ جهانی (UTC) ۱۱

منطقه‌ی زمانی (timezone) ۱۲

زمان سخت‌افزاری و نرم‌افزاری ۱۲

مشاهده و تغییر زمان ۱۳

استفاده از دستور *hwclock* برای مدیریت و مشاهده‌ی ساعت سخت‌افزاری ۱۳

استفاده از دستور *date* ۱۳

استفاده از *timedatectl* ۱٤

استفاده از Network Time Protocol (NTP) ۱۵

مفاهیم اولیه‌ی NTP ۱۵

انتخاب یک منبع زمانی مناسب ۱۷

استفاده و تنظیم برنامه‌ی *ntp* ۱۸

تنظیم *ntp* ۱۹

مدیریت سرویس *ntp* ۲۰

استفاده از برنامه‌ی *chrony* ۲۱

تنظیم *chrony* ۲۱

مدیریت سرویس *chrony* ۲۲

زمان‌بندی کارها ۲۳

درک عملکرد *cron* ۲۳

ایجاد کرون‌جاب‌های سیستمی ۲۴
ایجاد کرون‌جاب‌های کاربری ۲۶
استفاده از *anacron* ۲۷
استفاده از *at* ۲۹

مقدمه

جلسه قبل، دانش خود در مورد ادیتور vim را بهبود بخشیدیم و سپس به صورت کامل در مورد مدیریت حساب‌های کاربری صحبت کردیم. پس از آن با مفهوم لاگ و سیستم‌های لاگ در لینوکس آشنا شدیم و پروتکل Syslog را به طور کامل توضیح دادیم. در این جلسه، با لاگینگ از طریق systemd-journald آشنا می‌شویم و سپس در مورد مدیریت و تنظیم زمان در لینوکس و همچنین زمان‌بندی عملیات متفاوت صحبت می‌کنیم.

ژورنالینگ با systemd-journald

ژورنالینگ بسیار شبیه به لاگینگ می‌باشد، با این تفاوت که ژورنالینگ اطلاعات بیشتری (اطلاعات آماری و...) نسبت به لاگ‌ها ذخیره می‌کند و به عبارت دیگر، ژورنالینگ کامل‌تر از لاگینگ می‌باشد.

همانطور که قبلا گفتیم، امروزه اکثر سیستم‌های لینوکسی از سیستم راه‌انداز systemd جهت راه‌اندازی سیستم استفاده می‌کنند. اما systemd خیلی فراتر از یک سیستم راه‌انداز ساده می‌باشد؛ یکی از سرویس‌های پیشرفته‌ای که توسط systemd ارائه می‌شود، سرویس ژورنالینگ systemd-journald می‌باشد.

این سرویس عملکردی نظیر rsyslogd (که در جلسه‌ی قبل با آن آشنا شدیم) دارد؛ یعنی دائما در سیستم اجرا و به دنبال جمع‌آوری لاگ‌ها می‌باشد. با این حال، systemd-journald علاوه بر داشتن قابلیت‌های پیشرفته‌تر، از نظر چگونگی مدیریت و پاسخ‌دهی به پیام‌های لاگ، کاملا با rsyslogd تفاوت دارد.

به طور کلی، سرویس ژورنالینگ systemd از ۳ بخش اصلی تشکیل شده است:

- برنامه (daemon) اصلی به نام systemd-journald که دائما در پشت صحنه در حال اجرا و بررسی وضعیت سیستم می‌باشد.
- فایل تنظیمات /etc/systemd/journald.conf که رفتار و عملکرد این سرویس را دیکته می‌کند.
- ابزار journalctl، که ما را در جستجو و مشاهده‌ی فایل‌های لاگ یاری می‌دهد.

ما در این بخش، می‌خواهیم به صورت کامل با سرویس ژورنالینگ systemd و چگونه‌ی مدیریت و کار با آن، آشنا شویم.

چگونگی عملکرد systemd-journald

همانطور که گفتیم، systemd-journald یک سرویس سیستمی می‌باشد که پیام‌های لاگ را جمع‌آوری و ذخیره می‌کند. پیام‌هایی که توسط این سرویس جمع‌آوری می‌شوند منابع متفاوتی داشته و از قسمت‌های متفاوتی از سیستم گردآوری می‌شوند. از انواع لاگ‌های جمع‌آوری شده توسط این سرویس، می‌توان به لاگ‌های کرنل، لاگ‌های مربوط به بوت سیستم، لاگ‌های مربوط به سرویس‌های نصب شده و... اشاره کرد.

نکته‌ی قابل توجه این است که این سرویس، قابلیت جمع‌آوری پیام‌های Syslog را نیز دارد.

شاید بتوان گفت مهم‌ترین چیزی که systemd-journald را از برنامه‌ای rsyslogd متمایز می‌کند، این است که فایل‌های لاگ‌ها ایجاد شده توسط systemd-journald، ایندکس شده هستند و دارای یک ساختار مشخص می‌باشند. این ساختارمندی و ایندکس بودن، یکی از ویژگی‌های فایل‌های ژورنال می‌باشد. جستجو در فایل‌های ژورنال بسیار سریع بوده و این فایل‌ها از نظر امنیتی (چه کسی می‌تواند آنها را بخواند یا روی آنها چیزی بنویسید) بهتر از فایل‌های Syslog معمولی می‌باشند.

یکی از مزیت‌های استفاده از systemd-journald این است که بر خلاف سیستم‌های معمول Syslog، این برنامه ژورنال‌ها را به صورت اتوماتیک Rotate می‌کند. این امر دردسرهای مربوط به نگه‌داری فایل‌های لاگ را کمتر می‌کند و از پیچیدگی‌های سیستم می‌کاهد.

به صورت پیش‌فرض، systemd-journald فایل‌های ژورنال را در دایرکتوری /var/log/journal قرار می‌دهد، البته به شرطی که بدین شکل تنظیم شده باشد. در بخش بعد، منظور از این حرف را خواهیم فهمید.

تنظیم systemd-journald

سرویس systemd-journald مانند سایر سرویس‌های لینوکسی، تنظیمات خود را از روی یک فایل متنی می‌خواند. این فایل، در موقعیت /etc/systemd/journald.conf قرار دارد و شامل دستورالعمل‌هایی می‌باشد که رفتار و عملکرد این سرویس را دیکته می‌کنند. بیایید نگاهی به این فایل بیاندازیم:

```
[root@localhost ~]# cat /etc/systemd/journald.conf
....
[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
.....
#MaxLevelConsole=info
#MaxLevelWall=emerg
#LineMax=48K
```

همانطور که می‌بینید، در این فایل تعداد زیادی دستورالعمل وجود دارد. با اولین نگاه، ممکن است فکر کنید که هیچکدام از دستورالعمل‌های موجود در این فایل، در تنظیمات systemd-journald اعمال نشده‌اند، چرا که در ابتدای هر دستورالمل یک علامت # وجود دارد. اما واقعیت امر این است که مقادیری که در حال حاضر در این فایل مشاهده می‌کنیم، مقادیر پیش‌فرضی هستند که systemd-journald با آن اجرا شده است. برای تغییر هر کدام از این مقادیر، کافی است علامت # را از ابتدای هر خط برداشته و پس از علامت =، مقدار مورد نظر را قرار دهیم.

برخی از مهم‌ترین دستورالعمل‌های موجود در این فایل به شرح زیر می‌باشند:

جدول ۱ - برخی از مهم‌ترین دستورالعمل‌های موجود در فایل journald.conf

دستورالعمل	توصیف
Storage	مشخص می‌کند که systemd-journald به چه نحو پیام‌های رویدادهای متفاوت را ذخیره می‌کند. این دستورالعمل می‌تواند یکی از مقادیر volatile، persistent، auto و none را داشته باشد (مفهوم هر کدام از این مقادیر را به زودی توضیح می‌دهیم). (مقدار پیش‌فرض آن auto می‌باشد.)
Compress	فشرده‌سازی یا عدم فشرده‌سازی فایل‌های ژورنال را مشخص می‌کند. این دستورالعمل می‌تواند یکی از مقادیر yes یا no را داشته باشد. عملکرد هر کدام از این مقادیر واضح است، پس به توضیح آنها نمی‌پردازیم. (مقدار پیش‌فرض آن auto می‌باشد.)

ForwardToSyslog	مشخص می‌کند که پیام‌های دریافتی باید به یک برنامه‌ی جانبی مدیریت Syslog (مانند rsyslogd) ارسال شوند یا نه. این دستورالعمل می‌تواند یکی از مقادیر yes یا no را داشته باشد. (مقدار پیش‌فرض آن yes می‌باشد.)
ForwardToWall	مشخص می‌کند که آیا پیام‌های دریافتی باید به صورت پیام‌های Wall به همه‌ی کاربران موجود در سیستم نشان داده شوند یا نه. این دستورالعمل می‌تواند یکی از مقادیر yes یا no را داشته باشد. پیام‌های Wall، پیام‌هایی هستند که روی ترمینال، فارغ از عملکرد کنونی سیستم، نشان داده می‌شوند. (مقدار پیش‌فرض آن yes می‌باشد.)
MaxFileSec	مشخص می‌کند که پس از چه مدتی، فایل ژورنال باید Rotate شود. برای مشخص کردن این زمان، باید یک عدد، به علاوه‌ی واحد زمانی مورد نظر (نظیر month، week یا day) به این دستورالعمل بدهیم. (مقدار پیش‌فرض آن 1month می‌باشد.)
SystemMaxFileSize	مشخص می‌کند که یک فایل ژورنال در حالت persistent، می‌تواند چه مقدار از فضای هارددیسک را اشغال کند. برای مشخص کردن این مقدار، باید یک عدد، به علاوه‌ی واحد حجمی مورد نظر (نظیر K، M یا G) را به این دستورالعمل بدهیم.
SystemMaxUse	مشخص می‌کند که به طور کلی، سرویس systemd-journald چه مقدار از هارددیسک را در حالت persistent می‌تواند مصرف کند. برای مشخص کردن این مقدار، باید یک عدد، به علاوه‌ی واحد حجمی مورد نظر (نظیر K، M یا G) را به این دستورالعمل بدهیم. (مقدار پیش‌فرض، ۱۰ درصد از حجم کنونی هارددیسک می‌باشد.)

از میان دستورالعمل‌های ذکر شده در این جدول، مقادیر دستورالعمل Storage نیاز به توضیح بیشتری دارند. به طور کلی:

- مقدار auto:

باعث می‌شود systemd-journald به دنبال دایرکتوری /var/log/journal بگردد و لاگ‌ها را در آن دایرکتوری قرار دهد. اگر این دایرکتوری وجود نداشته باشد، پیام‌ها را در دایرکتوری موقت /run/log/journal قرار می‌دهد، البته محتویات این دایرکتوری پس از خاموش شدن سیستم، پاک می‌شوند.

- مقدار persistent:

باعث می‌شود systemd-journald دایرکتوری /var/log/journald را (در صورت عدم وجود) به صورت اتوماتیک ایجاد کند و لاگ‌ها را در آنجا ذخیره کند.

- مقدار volatile:

باعث می‌شود که سرویس systemd-journald، پیام‌ها را فقط و فقط درون دایرکتوری /run/log/journal قرار دهد.

- مقدار none:

باعث می‌شود کلیه‌ی پیام‌ها دور ریخته شوند.

همانطور که در بالا دیدیم، در فایل تنظیمات journald.conf سیستم‌عامل CentOS، دستورالعمل Storage، به صورت پیش‌فرض، مقدار auto را دارد. برای این که بتوانیم در مورد journald بیشتر صحبت کنیم، بهتر است مقدار Storage را به persistent تبدیل کنیم. البته می‌توانیم از مقدار auto استفاده کرده و خودمان دایرکتوری /var/log/journald را ایجاد کنیم، اما مدیریت مجوزها و مالکیت این دایرکتوری کار دردسرسازی خواهد بود، پس بهتر است اجازه دهیم خود journald این دایرکتوری را ایجاد کند. پس بیایید فایل journald.conf را به صورت زیر تغییر دهیم:

```
[root@localhost ~]# cat /etc/systemd/journald.conf
....
[Journal]
Storage=persistent
.....
```

پس از تغییر این دستورالعمل و ذخیره‌ی فایل، باید سرویس systemd-journald را restart کنیم:

```
[root@localhost ~]# systemctl restart systemd-journald
```

پس از این کار، دایرکتوری /var/log/journal ایجاد می‌شود و فایل‌های ژورنال درون آن قرار می‌گیرند.

پیدا کردن فایل‌های ژورنال

بسته به چگونگی تنظیم systemd-journald، ممکن است یک یا چند فایل ژورنال فعال در سیستم وجود داشته باشد. برای مثال، اگر دستورالعمل Storage برابر با persistent باشد، ما می‌توانیم با استفاده از دستورالعمل SplitMode، فایل‌های ژورنال را به چند فایل مجزا تقسیم کنیم؛ به طوری که برای هر کاربر (یا هر UID) یک فایل ژورنال و برای کل سیستم نیز، یک فایل ژورنال مجزا داشته باشیم.

همانطور که در بخش قبل نیز به آن اشاره کردیم، موقعیت قرارگیری فایل‌های ژورنال، کاملا بستگی به تنظیمات دستورالعمل Storage دارد. فایل ژورنال فعال مربوط به رویدادهای سیستمی، system.journal نام دارد و فایل‌های ژورنال مربوط به هر کاربر، به صورت user-*UID*.jornal نام گذاری می‌شوند، به طوری که *UID* نشان دهنده‌ی User ID هر کاربر می‌باشد.

فایل‌های ژورنال پس از گذر یک زمان خاص یا پس از رسیدن به یک میزان حجم، به صورت اتوماتیک Rotate می‌شوند. نام فایل‌های Rotate شده با system یا user-*UID* شروع شده، پس از آن یک علامت @ می‌آید و پس از آن، تعدادی حروف و رقم قرار می‌گیرند و پسوند journal. خواهند داشت.

برای آشنایی بیشتر با طریقه‌ی نام گذاری ژورنال‌های Rotate شده، بیایید به چگونگی نام‌گذاری فایل‌های ژورنال درون یک سرور تحت بار (توزیع Ubuntu) نگاه کنیم:

```
behnam@ravan-music:~$ ls -1 /var/log/journal/293017dc3353bc188e4680a37a30e5bb/
system@185527dea4e44f1987a36ba75c5982aa-0000000000000001-0005b07266c49fbb.journal
system@793d35954cf54734aba5d52bf86c5aae-0000000000000001-0005b0723f3aae5a.journal
...
system.journal
```

```
..
user-1000@23a004c5db92457a9b32c7c3fd6981d8-000000000000060c-0005b0736fed7941.journal
user-1000.journal
...
user-1001@8b373f460a9d42c3ae478dc93bb7ec73-000000000001c9ef-0005b2fdd19b5c01.journal
user-1001.journal
```

همانطور که می‌بینید، این سرور هم از فایل‌های ژورنال کاربر و هم از فایل‌های ژورنال سیستم استفاده می‌کند. طریقه‌ی نام‌گذاری فایل‌های ژورنال کاربر و سیستم و همچنین چگونگی تغییر نام آنها پس از Rotate شدن نیز کاملا واضح می‌باشد.

مشاهده‌ی فایل‌های ژورنال با *journalctl*

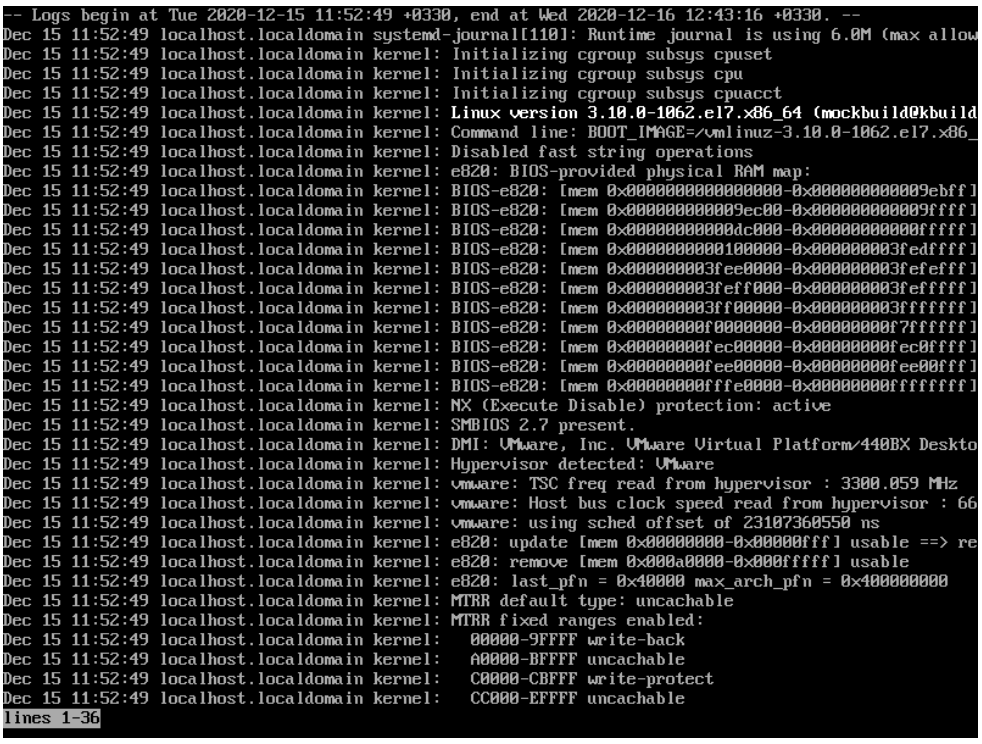
سرویس systemd-journald، ژورنال‌های خود را در فایل‌های متنی ذخیره نمی‌کند، بلکه آنها را به عنوان یک سری فایل باینری مخصوص که به دیتابیس شبیه می‌باشند، ذخیره می‌کند. این امر خواندن فایل‌ها را کمی دشوارتر می‌کند، اما در عوض، جستجو درون ژورنال‌ها را بسیار ساده و سریع‌تر می‌کند.

ما برای مشاهده‌ی فایل‌های ژورنال، از ابزار journalctl استفاده می‌کنیم. این دستور که یک ابزار بسیار قدرتمند می‌باشد، به ما اجازه می‌دهد که اطلاعات موجود در هر ژورنال را مشاهده و فیلتر کنیم.

برای مشاهده‌ی کلیه‌ی اطلاعات ژورنال سیستم (از قدیمی‌ترین به جدیدترین)، کافی است فقط دستور journalctl را بدون هیچ آپشنی، اجرا کنیم:

```
[root@localhost ~]# journalctl
```

به محض اجرای این دستور، با نمایی نظیر تصویر ۱ مواجه می‌شویم:



تصویر ۱- نمای مشاهده شده پس از اجرای journalctl

همانطور که در تصویر بالا می‌بینید، اطلاعات بسیار زیادی توسط این دستور به ما نشان داده می‌شود. خط اول خروجی این دستور، به ما می‌گوید که ژورنال کنونی در چه ساعتی آغاز شده و آخرین پیام آن، در چه زمانی درون آن نوشته شده است.

اگر توجه کنید، می‌بینید که پس از خط اول، همه‌ی موارد موجود در خروجی، از چندین فیلد تشکل شده‌اند. برای مثال، چندین فیلد ابتدایی در هر خط، نشان دهنده‌ی تاریخ و زمان دقیق وقوع یک رویداد می‌باشند. پس از این فیلدها، فیلد Hostname سیستم (`localhost.localdomain`) قرار گرفته و در فیلد بعد، موقعیت وقوع هر پیام ژورنال قرار گرفته است. همانطور که می‌بینید، کرنل تعداد زیادی پیام در ژورنال نوشته است که دلیل این امر، شروع به کار کردن `systemd.journald` هنگام بوت سیستم می‌باشد.

دستور `journalctl`، قابلیت‌های بسیار زیادی دارد و برای صحبت در مورد آنها، باید ابتدا در مورد Syntax کلی این دستور صحبت کنیم. به طور کلی، به صورت زیر از این دستور استفاده می‌کنیم:

```
journalctl [OPTIONS...] [MATCHES...]
```

به طوری که OPTIONS، مشخص کننده‌ی آپشن‌های `journalctl` می‌باشد که رفتار `journalctl` را مشخص می‌کنند و MATCHES، مشخص کننده‌ی الگوهایی می‌باشند که به ما کمک می‌کنند موارد موجود در ژورنال را فیلتر کرده و فقط موارد مورد نظر را مشاهده کنیم.

بیایید ابتدا با برخی از معروف‌ترین آپشن‌های `journalctl` آشنا شویم:

جدول ۲- پرکاربردترین آپشن‌های `journalctl`

کوتاه	بلند	عملکرد
-a	**--all**	کلیه‌ی فیلدها را در خروجی نشان می‌دهد.
-e	**--pager-end**	موارد موجود در انتهای ژورنال را نشان می‌دهد.
-k	**--dmesg**	فقط موارد مربوط به کرنل را نشان می‌دهد.
-n *num*	**--lines=***num*	تعداد *num* از اخیرترین موارد موجود در ژورنال را نشان می‌دهد.
-r	**--reverse**	جدیدترین موارد موجود در ژورنال را در بالای صفحه نشان می‌دهد.
-S *date*	**--since=***date*	موارد ژورنالی که از تاریخ *date* به بعد ایجاد شده‌اند را نشان می‌دهد. *date*، به صورت YYYY-MM-DD:HH:MM:SS نوشته می‌شود. اگر HH:MM:SS مشخص نشود، زمان 00:00:00 در نظر گرفته می‌شود. توجه کنید که به جای تاریخ، می‌توانیم از کلماتی نظیر yesterday، today، و now نیز استفاده کنیم.
-U *date*	**--until=***date*	موارد موجود در ژورنال را تا تاریخ *date* نشان می‌دهد. *date* دقیقا همانطور که در بالا توضیح دادیم مشخص می‌شود.
-u *unit* or *pattern*	**--unit=***unit* or *pattern*	موارد موجود در ژورنال که فقط مربوط به یونیت *unit* در systemd هستند یا یونیت‌هایی که نامشان با *pattern* مطابقت دارد را نشان می‌دهد.

بیایید از یکی از این آپشن‌ها استفاده کنیم. برای مثال، بیایید فقط ۱۰ تا از آخرین موارد وارد شده در فایل ژورنال توسط سرویس sshd را مشاهده کنیم، برای این کار:

```
[root@localhost ~]# journalctl -u sshd -n 10
-- Logs begin at Fri 2020-12-18 10:00:06 +0330, end at Tue 2020-12-22 10:03:46 +0330. --
Dec 21 11:11:06 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
Dec 21 12:15:42 localhost.localdomain sshd[1809]: Accepted password for root from 192.168.1.102 port 15327 ssh2
Dec 21 14:07:35 localhost.localdomain sshd[1260]: Received signal 15; terminating.
Dec 21 14:07:35 localhost.localdomain systemd[1]: Stopping OpenSSH server daemon...
Dec 21 14:07:36 localhost.localdomain systemd[1]: Stopped OpenSSH server daemon.
-- Reboot --
Dec 22 09:44:00 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
Dec 22 09:44:01 localhost.localdomain sshd[1262]: Server listening on 0.0.0.0 port 22.
Dec 22 09:44:01 localhost.localdomain sshd[1262]: Server listening on :: port 22.
Dec 22 09:44:01 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
Dec 22 10:03:46 localhost.localdomain sshd[1806]: Accepted password for root from 192.168.1.102 port 5084 ssh2
```

همانطور که می‌بینید، با استفاده از آپشن u- و مشخص‌کردن یونیت sshd به journalctl گفتیم که فقط به دنبال لاگ‌های یونیت sshd هستیم و با مشخص کردن آپشن n- و وارد کردن عدد ۱۰، به journalctl گفتیم که فقط ۱۰ مورد آخر موجود در فایل ژورنال را به ما نشان دهد.

حال که با آپشن‌های این دستور بیشتر آشنا شدیم، بیایید با برخی از پرکاربردترین الگوها برای فیلتر کردن خروجی دستور journalctl آشنا شویم:

جدول ۳- پرکاربردترین الگوها برای فیلتر خروجی journalctl

الگو	عملکرد
field	فقط فیلد *field* را در خروجی به ما نشان می‌دهد. برای نشان دادن چندین فیلد در خروجی، کافی است *field* های مورد نظر را با فاصله در کنار هم قرار دهیم.
OBJECT_PID=*pid*	فقط مواردی که توسط برنامه‌ی دارای پراسس‌آی‌دی *pid* ایجاد شده‌اند را به ما نشان می‌دهد.
PRIORITY=*value*	فقط مواردی که دارای میزان شدت *value* می‌باشند را به ما نشان می‌دهد. *value* می‌توانند یکی از اعداد مربوط به کلیدواژه‌های میزان شدت باشد: emerg(0), alert(1), crit(2), err(3), warning(4), notice(5), info(6), debug(7)
_HOSTNAME=*host*	فقط مواردی که توسط سیستمِ دارای هاست‌نیم *host* ایجاد شده را به ما نشان می‌دهد.
_SYSTEMD_UNIT=*unit.type*	فقط مواردی که توسط یونیت *unit.type* ایجاد شده را به ما نشان می‌دهد.
_TRANSPORT=*transport*	فقط مورادی که با روش *transport* توسط ژورنال دریافت شده را به ما نشان می‌دهد.
_UDEV_SYSNAME=*dev*	فقط مواردی که از دستگاه *dev* دریافت شده را به ما نشان می‌دهد.

_UID=*userid*	فقط مواردی که توسط کاربر دارای UID برابر با *userid* ایجاد شده را به ما نشان می‌دهد.

بیایید از یکی از این الگوها استفاده کنیم. برای مثال، فرض کنید می‌خواهیم هر موردی که از دیروز تا الان درون ژورنال نوشته شده و دارای میزان شدت warning می‌باشد را مشاهده کنیم. پس:

```
[root@localhost ~]# journalctl --since=yesterday PRIORITY=3
-- Logs begin at Fri 2020-12-18 10:00:06 +0330, end at Tue 2020-12-22 12:01:01 +0330. --
Dec 21 11:10:44 localhost.localdomain kernel: sd 0:0:0:0: [sda] Assuming [...]
Dec 21 11:10:50 localhost.localdomain kernel: piix4_smbus [...]
Dec 21 11:11:05 localhost.localdomain systemd[1]: Failed to [...]
-- Reboot --
Dec 22 09:43:37 localhost.localdomain kernel: sd 2:0:0:0: [sda] Assuming [...]
Dec 22 09:43:43 localhost.localdomain kernel: piix4_smbus 0000:00:07.3: SMBus [...]
Dec 22 09:44:00 localhost.localdomain systemd[1]: Failed to [...]
Dec 22 09:44:01 localhost.localdomain rsyslogd[1263]: omfwd: error 11 [...]
```

همانطور که می‌بینید، ما با استفاده از آپشن --since، و ارائه‌ی مقدار yesterday به آن، از journalctl خواستیم که فقط مواردی که از دیروز داخل فایل ژورنال نوشته شده‌اند را به ما نشان دهد و سپس با استفاده از الگوی PRIORITY و ارائه‌ی مقدار ۳ به آن، به journalctl گفتیم که فقط پیام‌هایی که دارای میزان شدت warning می‌باشند را به ما نشان دهد.

مدیریت و نگهداری فایل ژورنال

علاوه بر تنظیمات موجود در فایل journal.conf، چندین روش دستی برای مدیریت و نگهداری فایل‌های ژورنال نیز موجود می‌باشد. برای مثال، ما می‌توانیم میزان حجم مصرفی فایل‌های ژورنال را با استفاده از آپشن --disk-usage دستور journalctl بررسی کنیم. برای مثال، اجرای این دستور بر روی یک سرور تحت بار کم، نتیجه‌ی زیر را به ما می‌دهد:

```
behnam@ravan-music:~$ sudo journalctl --disk-usage
Archived and active journals take up 400.4M in the file system.
```

همانطور که می‌بینید، این دستور به ما می‌گوید که کلیه‌ی فایل‌های ژورنال، اعم از فایل‌های فعال و فایل‌های آرشیو شده، چقدر حجم اشغال کرده‌اند.

با این که systemd-journald می‌تواند به صورت اتوماتیک و با توجه به تنظیمات موجود در journal.conf به پاک‌سازی فایل‌های ژورنال بپردازد، ما می‌توانیم این کار را به صورت دستی نیز انجام دهیم. ما می‌توانیم با به کارگیری از آپشن‌های --vacuum-size یا --vacuum-time، فایل‌ها را با توجه به حجم یا زمان، پاکسازی کنیم.

همانطور که نام آن پیداست، --vaccum-size فایل‌های ژورنال را تا جایی حذف می‌کند که حجم اشغال شده توسط آنها، به مقدار مشخص شده برای این آپشن برسد. ما پس از مشخص کردن این آپشن، یک عدد و پس از آن، یک واحد حجم (K، M، G یا T) قرار می‌دهیم. توجه کنید که این آپشن فقط فایل‌های ژورنال آرشیو شده را پاک می‌کند و به فایل‌های ژورنال فعال کاری ندارد.

به عبارت دیگر، اگر دستور journalctl --vaccum-size 200M را وارد کنیم، journalctl به سراغ فایل‌های ژورنال آرشیو شده رفته و آنها را تا جایی پاک می‌کند که فقط 200M حجم از هارد دیسک مصرف کنند.

هنگام استفاده از آپشن `--vaccum-time`-، ما مشخص می‌کنیم که قدیمی‌ترین مورد موجود در ژورنال باید مربوط به چه زمانی باشد. پس از این کار، کلیه موارد موجود در ژورنال که قبل از زمان مشخص شده ایجاد شده بوده‌اند، پاک می‌شوند. برای مشخص کردن یک مقدار برای این آپشن، یک عدد و پس از آن یک واحد زمانی (`s`، `min`، `h`، `days`، `months`، `weeks` یا `years`) وارد می‌کنیم. توجه کنید که این آپشن نیز فقط روی فایل‌های ژورنال آرشیو شده تاثیر دارد.

بیایید این امر را امتحان کنیم:

```
[root@localhost ~]# journalctl --disk-usage
Archived and active journals take up 410.0M on disk.
[root@localhost ~]# journalctl --vacuum-size=300M
Vacuuming done, freed 90.0M of archived journals on disk
[root@localhost ~]# journalctl --disk-usage
Archived and active journals take up 320.0M on disk.
```

نکته‌ی قابل توجه در مورد خروجی دستورهای بالا، این است که پس از اجرای دستور پاکسازی ژورنال (`journalctl --vaccum-size=300M`)، خروجی دستور `journalctl --disk-usage` به ما مقدار `320.0M` را نشان می‌دهد؛ این در حالی است که ما به آپشن `--vacuum-size` مقدار `300M` را دادیم. دلیل دریافت مقدار `320.0M` در خروجی، این است که آپشن `--disk-usage` هم سایز فایل‌های آرشیو شده و هم سایز فایل‌های فعال را نشان می‌دهد، در حالی که آپشن `--vacuum-size` فقط روی فایل‌های آرشیو شده کار می‌کند.

لایه‌بندی لاگ‌ها

در صورت نیاز (یا لزوم)، می‌توانیم کاری کنیم که برنامه‌ی `systemd-journald` و یک برنامه‌ی Syslog نظیر `rsyslog` به صورت همزمان با هم اجرا شده و با یکدیگر همکاری کنند. دو روش اصلی برای انجام این کار وجود دارد:

- روش ژورنال-کلاینت

این روش به یک برنامه‌ی Syslog اجازه می‌دهد که به عنوان یک کلاینتِ ژورنال، عمل کند و موارد موجود در فایل‌های ژورنال را بخواند. این روش، معمولا روش بسیار مناسبی است، چرا که باعث می‌شود پیام‌های مهم بوت که قبل از اجرا شدن برنامه‌ی Syslog به وجود می‌آیند از دست نروند. در سیستم‌های جدیدتر، معمولا `rsyslog` به صورت پیش‌فرض در این حالت اجرا می‌شود.

اگر `rsyslog` در سیستم ما به این شکل تنظیم نشده باشد یا بخواهیم از تنظیم شدن آن به این شکل اطمینان حاصل کنیم، کافی است به تنظیمات `rsyslog` در فایل `/etc/rsyslog.conf` نگاه کنیم. در این فایل، باید ماژول‌های `imuxsock` و `imjournal` توسط `ModLoad` (بدون وجود علامت `#` در ابتدای خط) لود شده باشند. یعنی:

```
[root@localhost ~]# grep -E "ModLoad (imjournal|imuxsock)" /etc/rsyslog.conf
$ModLoad imuxsock # provides support for local system logging (e.g. via logger command)
$ModLoad imjournal # provides access to the systemd journal
```

همانطور که می‌بینید، این دو ماژول داخل فایل ما لود شده‌اند، پس `rsyslog` در حالت ژورنال-کلاینت کار می‌کند.

- روش ارسال به Syslog

این روش، از فایل `/run/systemd/journal/syslog` استفاده می‌کند. در این حالت، پیام‌ها به این فایل ارسال می‌شوند و برنامه‌ی Syslog، پیام‌ها را از روی این فایل می‌خواند. برای استفاده از این روش، باید دستورالعل `ForwardToSyslog` در فایل `journald.conf` را برابر با `yes` قرار دهیم. توجه کنید که پس از تغییر مقدار یک دستورالعمل، باید یک بار `systemd-journald` را restart کنیم.

نوشتن موارد جدید درون ژورنال

ما می‌توانیم با استفاده از دستور `systemd-cat`، موارد مورد نظر خود را درون ژورنال بنویسیم. برای این کار، کافی است STDOUT یک دستور را درون `systemd-cat`، پایپ کنیم. برای مثال، فرض کنید می‌خواهیم عبارت `Hello, Journal` را درون ژورنال بنویسیم:

```
[root@localhost ~]# echo "Hello, Journal" | systemd-cat
```

همانطور که می‌بینید، ما از دستور `echo` استفاده کردیم و خروجی آن (STDOUT) را داخل برنامه‌ی `systemd-cat` پایپ کردیم. حال بیایید از صحت وارد شدن این عبارت در ژورنال مطمئن شویم:

```
[root@localhost ~]# journalctl | grep "Hello, Journal"
Dec 24 10:02:25 localhost.localdomain unknown[1823]: Hello, Journal
```

همانطور که می‌بینید، عبارت مورد نظر ما درون ژورنال نوشته شده است.

اگر در سیستم ما یک برنامه‌ی Syslog که در حالت ژورنال کلاینت قرار دارد نیز وجود داشته باشد، ما می‌توانیم با استفاده از دستور `logger` مواردی را درون ژورنال بنویسیم. در این حالت، استفاده از `logger` دقیقا مانند قبل می‌باشد، یعنی:

```
[root@localhost ~]# logger "testing logger"
```

همانطور که می‌بینید، ما فقط دستور `logger` را وارد کرده و در مقابل آن، پیامی که میخواستم در ژورنال نوشته شود را وارد کردیم. حال بیایید از صحت نوشته شدن پیام مورد نظر درون ژورنال، اطمینان حاصل کنیم:

```
[root@localhost ~]# journalctl -r
-- Logs begin at Fri 2020-12-18 10:00:06 +0330, end at Thu 2020-12-24 10:41:47 +0330.
--
Dec 24 10:41:47 localhost.localdomain root[1887]: testing logger
[...]
Dec 24 10:10:15 localhost.localdomain unknown[1834]: Hello, Journal
```

همانطور که می‌بینید، پیامی که با `logger` ایجاد کردیم درون ژورنال قرار گرفته است. اگر دقت کنید، می‌بینید که پیامی که با `logger` درون ژورنال نوشتیم یوزرنیم ما را نیز درون ژورنال قرار می‌دهد.

شاید لازم باشد کمی در مورد آپشن `r-` نیز صحبت کنیم. اگر به خاطر داشته باشید، آپشن `r-` ژورنال را از پایین به بالا نشان میداد، یعنی جدیدترین پیام ژورنال را در بالای صفحه نشان میداد و هر چه به انتهای صفحه نزدیک میشدیم، پیام‌های ژورنال قدیمی‌تر می‌شدند.

مدیریت زمان سیستم

لینوکس و همه‌ی سرویس‌های موجود در آن، به یک ساعت دقیق نیاز دارند. در بحش قبل دیدیم که نگهداری، کنترل و مدیریت لاگ‌ها و ژورنال‌ها در یک سیستم امر بسیار مهمی می‌باشد و ما را در رفع مشکلات سیستم یاری می‌دهد، اما اگر سیستم یک ساعت دقیق نداشته باشد و ما ندانیم که اتفاقات ذکر شده در لاگ‌ها دقیقا در چه زمانی رخ داده‌اند، لاگ‌های ما عملا به‌دردنخور می‌شوند. علاوه بر این، کارهایی نظیر زمان‌بندی برخی عملیات برای اجرا در آینده نیز نیازمند یک ساعت دقیق و قابل اعتماد می‌باشد.

بنابراین، پرواضح است که داشتن زمان صحیح و دقیق روی یک سرور، امری بسیار مهم می‌باشد. ما در این بخش می‌خواهیم در مورد مفاهیم اولیه‌ی زمان و دستورهای مربوط به کار با زمان صحبت کنیم.

مفاهیم اولیه زمان در لینوکس

زمان محلی (localtime) و زمان هماهنگ جهانی (UTC)

بیایید ابتدا در مورد زمانی که در صحبت‌های روزمره از آن استفاده می‌کنیم صحبت کنیم. به این زمان، زمان محلی (local time) یا زمان دیواری (wall time) می‌گویند. دلیل این نام‌گذاری این است که مردم همیشه با نگاه کردن به ساعت روی دیوار، زمان کنونی را تشخیص داده و در مورد آن صحبت می‌کنند. نام استاندارد این زمان، localtime می‌باشد.

برخی از سیستم‌عامل‌ها، ساعت خود را برابر با ساعت محلی قرار می‌دهند. این رویکرد برای مردم عادی که به زمان محلی عادت دارند بسیار مناسب می‌باشد، اما این ساعت برای کارهایی نظیر مدیریت شبکه، اصلا مناسب نمی‌باشد. برای مثال ساعت ۹ صبح در تهران، برابر با ساعت ۵ و نیم صبح در لندن می‌باشد؛ بنابراین پروتکل‌های شبکه که برای عملکرد صحیح به زمان دقیق نیاز دارند (تقریبا همه‌ی پروتکل‌ها) گیج شده و علاوه بر این، فایل‌های لاگ به‌دردنخوری ایجاد می‌کنند.

بنابراین استاندارد زمان محلی، حداقل برای کارهای شبکه، به‌دردنخور می‌باشد و باعث دردسر می‌شود. به همین دلیل، بسیاری از سیستم‌های لینوکسی، از استاندارد زمانی دیگری به نام «ساعت هماهنگ جهانی (Coordinated Universal Time)» یا UTC استفاده می‌کنند. UTC زمانی می‌باشد که با توجه به محل زندگی فرد، دچار تغییر نمی‌شود؛ یعنی زمان UTC در تهران با زمان UTC در سیدنی کاملا یکسان می‌باشد. به عبارت دیگر، ساعت ۲۱ در استاندارد UTC، اشاره به یک زمان یکسان در تهران و لندن یا هر جای دیگر، دارد.

سیستم‌هایی که از استاندارد UTC استفاده می‌کنند، باید Time Zone یا منطقه‌ی زمانی که در آن قرار دارند را بدانند. برای مثال، اگر به خاطر داشته باشید، دستور `ls -l` لیستی از محتویات یک دایرکتوری به همراه زمان ایجاد آنها را به ما نشان می‌داد. زمانی که این دستور را در سیستم وارد می‌کنیم، لینوکس زمان ایجاد این فایل‌ها و دایرکتوری را در استاندارد UTC می‌خواند، سپس از زمان نشان داده شده مقداری را کم می‌کند یا اضافه می‌کند تا زمان نشان داده شده تبدیل به زمان محلی ما شود. مقدار اضافه شده یا کم شده، مقداری است که توسط منطقه‌ی زمانی مشخص می‌شود.

منطقه‌ی زمانی (timezone)

در بخش قبل به مفهوم منطقه‌ی زمانی اشاره کردیم. یکی از مهم‌ترین جنبه‌های زمان و مدیریت آن در سیستم، تنظیم منطقه‌ی زمانی می‌باشد. هر کشور با توجه به وسعت خود، یک یا چند منطقه‌ی زمانی دارد. منطقه‌ی زمانی مشخص می‌کند که ساعت محلی یک کشور، چه میزان با ساعت UTC تفاوت دارد.

در توزیع‌های Red Hat-based، منطقه‌ی زمانی یک سیستم توسط فایل /etc/localtime تعریف می‌شود. این فایل، یک فایل متنی نیست و ما نمی‌توانیم آن را خوانده یا موارد موجود در آن را تغییر دهیم. فایل موجود در /etc/localtime، در واقع یک سافت‌لینک به یک فایل باینری موجود در /usr/share/zoneinfo می‌باشد.

برای مشاهده‌ی منطقه‌ی زمانی کنونی سیستم، کافی است به صورت زیر عمل کنیم:

```
[root@localhost ~]# ls -l /etc/localtime
lrwxrwxrwx. 1 root root 33 Mar 20  2020 /etc/localtime -> ../usr/share/zoneinfo/Asia/Tehran
```

همانطور که می‌بینید، فایل /etc/localtime یک سافت‌لینک به فایل /usr/share/zoneinfo/Asia/Tehran می‌باشد. این یعنی که منطقه‌ی زمانی سیستم ما تهران می‌باشد.

برای تغییر منطقه‌ی زمانی سیستم، کافی است کاری کنیم که فایل /etc/localtime به یکی از فایل‌های منطقه‌ی زمانی موجود در /usr/share/zoneinfo سافت‌لینک بزند. برای این کار، ابتدا باید فایل localtime کنونی را پاک کرده یا نام آن را تغییر دهیم، و سپس با استفاده از دستور ln -s، سافت لینک را ایجاد کنیم. یعنی:

```
[root@localhost ~]# mv /etc/localtime /etc/localtime.old
[root@localhost ~]# ln -s /usr/share/zoneinfo/Asia/Pyongyang /etc/localtime
```

حال بیایید از صحت تغییر منطقه‌ی زمانی اطمینان حاصل کنیم:

```
[root@localhost ~]# ls -l /etc/localtime
lrwxrwxrwx. 1 root root 34 Feb  9 18:42 /etc/localtime -> /usr/share/zoneinfo/Asia/Pyongyang
```

همانطور که می‌بینید، منطقه‌ی زمانی ما به پیونگ‌یانگ تغییر پیدا کرد.

زمان سخت‌افزاری و نرم‌افزاری

مفهوم دیگری که پس از آشنایی با زمان محلی و UTC باید با آن آشنا شویم، مفهوم زمان سخت‌افزاری و نرم‌افزاری می‌باشد. زمان سخت‌افزاری، که به آن ساعت Real-Time نیز می‌گویند، سعی می‌کند زمان صحیح را حتی پس از خاموش شدن سیستم نگهداری کند. این ساعت، این امر را با استفاده از یک باتری انجام می‌دهد. وقتی سیستم بوت شود، لینوکس زمان کنونی را از ساعت سخت‌افزاری گرفته و ساعت نرم‌افزاری خود را آپدیت می‌کند. ساعت نرم‌افزاری، فقط زمانی که لینوکس در حال اجرا می‌باشد فعال است و بسیاری از نرم‌افزارهای لینوکسی از این ساعت استفاده می‌کنند، به همین دلیل بعضا به این ساعت، زمان سیستم (System Time) نیز می‌گویند.

زمان سخت‌افزاری و نرم‌افزاری، به طرز فجیعی روی سخت‌افزارهای معمول (x86 و x86-64) غیر قابل اطمینان هستند. ساعت سخت‌افزاری وابسته به باتری می‌باشد و این سبب می‌شود که زمان این ساعت دچار رانش شود. ساعت نرم‌افزاری نیز پس از دریافت زمان از ساعت سخت‌افزاری، کاملا توسط خود کرنل مدیریت می‌شود و این سبب می‌شود که ساعت پس از چند ماه روشن ماندن سیستم، تا چندین دقیقه رانش داشته باشد.

برای حل این مشکل، سیستم‌های لینوکسی از برخی روش‌های تحت شبکه برای مدیریت و نگهداری زمان استفاده می‌کنند. یکی از این روش‌ها، استفاده از پروتکل Network Time Protocol یا NTP می‌باشد که در بخش‌های بعدی با آن بیشتر آشنا می‌شویم.

مشاهده و تغییر زمان

ما می‌توانیم ساعت سخت‌افزاری و نرم‌افزاری را با استفاده از ابزارهای متفاوت موجود در لینوکس مشاهده کرده یا تغییر دهیم. هر کدام از این ابزارها، مزیت‌ها و کمبودهای خود را دارند. در این بخش، می‌خواهیم با این ابزارها بیشتر آشنا شویم.

استفاده از دستور *hwclock* برای مدیریت و مشاهده‌ی ساعت سخت‌افزاری

دستوری که معمولا از آن برای مدیریت و مشاهده‌ی ساعت سخت‌افزاری استفاده می‌شود، hwclock نام دارد. بیایید ابتدا این دستور را بدون هیچ آپشنی اجرا کنیم:

```
[root@localhost ~]# hwclock
Thu 24 Dec 2020 01:25:33 PM +0330  -0.478939 seconds
```

همانطور که می‌بینید، خروجی این دستور تاریخ و ساعت کنونی را به ما گزارش داد. در خروجی، به ترتیب نام روز، شماره‌ی روز، نام ماه، شماره‌ی سال و ساعت در فرمت AM/PM به ما نشان داده می‌شود. پس از آن، مقدار 0330+ نشان داده شده است. این امر به ما می‌گوید که ساعت نمایش داده شده، برای منطقه‌ی زمانی ما تنظیم شده است و در واقع ساعت نمایش داده شده، ۳ ساعت و نیم از ساعت UTC جلوتر می‌باشد. در واقع این دستور، ساعت localtime را به ما نشان داده است.

دستور hwclock چندین آپشن دارد که برخی از کاربردی‌ترین آنها به شرح زیر می‌باشند:

جدول ٤- کاربردی‌تری آپشن‌های hwclock

کوتاه	بلند	عملکرد
N/A	--localtime	کاری می‌کند که ساعت سخت‌افزاری، از استاندارد localtime استفاده کند.
-r	--show	ساعت کنونی سخت‌افزاری را نشان می‌دهد. اگر هیچ آپشنی به hwclock ندهیم، به صورت پیش‌فرض با این آپشن اجرا می‌شود.
-s	--hctosys	ساعت کنونی سخت‌افزاری را خوانده و ساعت نرم‌افزاری را برابر با آن قرار می‌دهد.
-u	--utc	باعث می‌شود ساعت سخت‌افزاری از استاندارد UTC استفاده کند.
-w	--systohc	ساعت نرم‌افزاری کنونی را خوانده و ساعت سخت‌افزاری را برابر با آن قرار می‌دهد.

استفاده از دستور *date*

معمولا از دستور date برای مشاهده یا تنظیم ساعت نرم‌افزاری استفاده می‌کنیم. در ساده‌ترین حالت، کافی است این دستور را بدون هیچ آپشنی وارد کنیم:

```
[root@localhost ~]# date
Sat Dec 26 10:24:54 +0330 2020
```

همانطور که می‌بینید، این دستور در خروجی به ما زمان محلی یا localtime را نشان داد.

ما می‌توانیم با استفاده از دستور date، تاریخ و زمان سیستم را به صورت دستی تغییر دهیم. اگر زمانی که مشخص می‌کنیم localtime باشد، می‌توانیم تاریخ و زمان مورد نظر را بدون هیچ‌گونه آپشنی به date بدهیم، اما اگر بخواهیم زمان UTC سیستم را مشخص کنیم، باید از آپشن u- یا utc-- استفاده کنیم. به طور کلی، فرمت مورد استفاده برای مشخص کردن زمان و تاریخ به صورت زیر می‌باشد:

MMDDhhmmCCYY

به طوری که MM مشخص کننده‌ی ماه (۰۱ تا ۱۲)، DD مشخص کننده‌ی روز (۰۱ الی آخر)، hh مشخص کننده‌ی ساعت (۰۰ الی ۲۳)، mm مشخص کننده‌ی دقیقه (۰۰ الی ۵۹)، CC مشخص کننده‌ی قرن و YY مشخص کننده‌ی سال می‌باشد.

برای مثال، بیایید تاریخ سیستم را به ۲۶ دسامبر ۲۰۲۰، ساعت ۱۳ تغییر دهیم:

```
[root@localhost ~]# date 122613002020
Sat Dec 26 13:00:00 +0330 2020
```

همانطور که می‌بینید، پس از وارد کردن دستور date، دقیقا طبق فرمت ذکر شده، ابتدا شماره‌ی ماه دسامبر (۱۲) را وارد کردیم، سپس شماره‌ی روز (۲۶) را وارد کردیم، پس از آن ساعت و دقیقه (۱۳۰۰) را وارد کردیم و در نهایت، سال مورد نظر (۲۰۲۰) را وارد کردیم.

نکته: اگر سیستم ما از پروتکل NTP یا SNTP استفاده کند، نمی‌توانیم با دستور date، زمان و تاریخ سیستم را به صورت دستی تغییر دهیم.

استفاده از *timedatectl*

می‌توان گفت ابزار timedatectl، بهترین ابزار برای مدیریت دستی ساعت نرم‌افزاری لینوکس می‌باشد. وارد کردن این دستور بدون هیچ آپشن جانبی، می‌تواند اطلاعات بسیار زیادی را به ما ارائه دهد. بیایید به جای حرف زدن، این دستور را وارد کرده و نگاهی به خروجی آن بیاندازیم. توجه کنید که خروجی این دستور بسته به توزیعی که از آن استفاده می‌کنیم، متفاوت خواهد بود:

```
[root@localhost ~]# timedatectl
      Local time: Sat 2020-12-26 13:14:44 +0330
  Universal time: Sat 2020-12-26 09:44:44 UTC
        RTC time: Sat 2020-12-26 09:41:44
       Time zone: Asia/Tehran (+0330, +0330)
     NTP enabled: n/a
NTP synchronized: no
 RTC in local TZ: no
      DST active: no
 Last DST change: DST ended at
                  Sun 2020-09-20 23:59:59 +0430
                  Sun 2020-09-20 23:00:00 +0330
 Next DST change: DST begins (the clock jumps one hour forward) at
                  Sun 2021-03-21 23:59:59 +0330
                  Mon 2021-03-22 01:00:00 +0430
```

اگر خروجی این دستور را با سایر دستورها مقایسه کنیم، می‌بینیم که اطلاعات بسیار بیشتری را به ما نشان می‌دهد. برای مثال، این دستور ابتدا زمان محلی، سپس زمان UTC و سپس زمان سخت‌افزاری (که به آن RTC هم می‌گویند) را به ما نشان می‌دهد. پس از آن، منطقه‌ی زمانی ما مشخص شده است. در دو خط بعدی، وجود NTP و سینک بودن آن بررسی شده و در خط بعد، مشخص شده که آیا ساعت سخت‌افزاری

به وقت منطقه‌ی زمانی ما می‌باشد یا نه. در دو خط نهایی، زمان وقوع آخرین تغییر ساعت تابستانی و زمان وقوع تغییر ساعت تابستانی بعدی مشخص شده است.

برای تغییر زمان سیستم با استفاده از دستور timedatectl، باید از فرمان set-time این دستور استفاده کنیم و تاریخ و ساعت مورد نظر را با فرمت زیر وارد کنیم:

```
“YYYY-MM-DD HH:MM:SS”
```

برای مثال، بیایید تاریخ و زمان سیستم را به ۳۰ دسامبر ۲۰۲۰ ساعت ٤ و ۲۰ دقیقه صبح تغییر دهیم:

```
[root@localhost ~]# timedatectl set-time "2020-12-30 04:20"
```

همانطور که که می‌بینید، پس از وارد کردن دستور timedatectl، فرمان set-time را به آن دادیم و سپس داخل دو علامت ""، ابتدا سال، ماه و روز را مشخص کردیم (با قرار دادن یک خط تیره بین هر مقدار) و سپس ساعت مورد نظر را در فرمت ٢٤ ساعته مشخص کردیم.

بیایید از صحت تغییر تاریخ و زمان مطمئن شویم:

```
[root@localhost ~]# date
Wed Dec 30 04:20:18 +0330 2020
```

همانطور که می‌بینید، ساعت و تاریخ دقیقا همانطور که خواسته بودیم، دچار تغییر شده است.

نکته: اگر سیستم ما از پروتکل NTP یا SNTP استفاده کند، نمی‌توانیم با دستور timedatectl، زمان و تاریخ سیستم را به صورت دستی تغییر دهیم.

دستور timedatectl آپشنها و فرمان‌های بیشتری نیز دارد که ما به آنها نمی‌پردازیم. برای کسب اطلاعات بیشتر، می‌توانید manpage این دستور را مطالعه کنید.

استفاده از Network Time Protocol (NTP)

گفتیم که وجود ساعت دقیق روی سیستم، امر بسیار مهمی می‌باشد و برخی از سیستم‌ها بدون یک ساعت دقیق، نمی‌توانند به درستی کار کنند. سیستم‌های مربوط به بورس، آب‌وهوا، نجوم، سرورهای شبکه و... از انواع سیستم‌هایی هستند که نیاز به ساعت بسیار دقیق دارند. ساعت سخت‌افزاری و نرم‌افزاری دقت مورد نیاز این سیستم‌ها را برآورده نمی‌کند و این سیستم‌ها نیازمند یک ساعت قابل اعتمادتر هستند.

نیاز این سیستم‌ها به یک ساعت بسیار دقیق، باعث به وجود آمدن پروتکل‌های متفاوتی برای هماهنگ‌سازی ساعت سیستم‌ها با ساعت‌های خیلی دقیق شده است. از میان این پروتکل‌ها، معروف‌ترین پروتکل، Network Time Protocol یا NTP می‌باشد. ما در این بخش، ابتدا در مورد مفاهیم اولیه‌ی NTP صحبت می‌کنیم و سپس به سراغ تنظیم NTP برای سیستم خود می‌رویم.

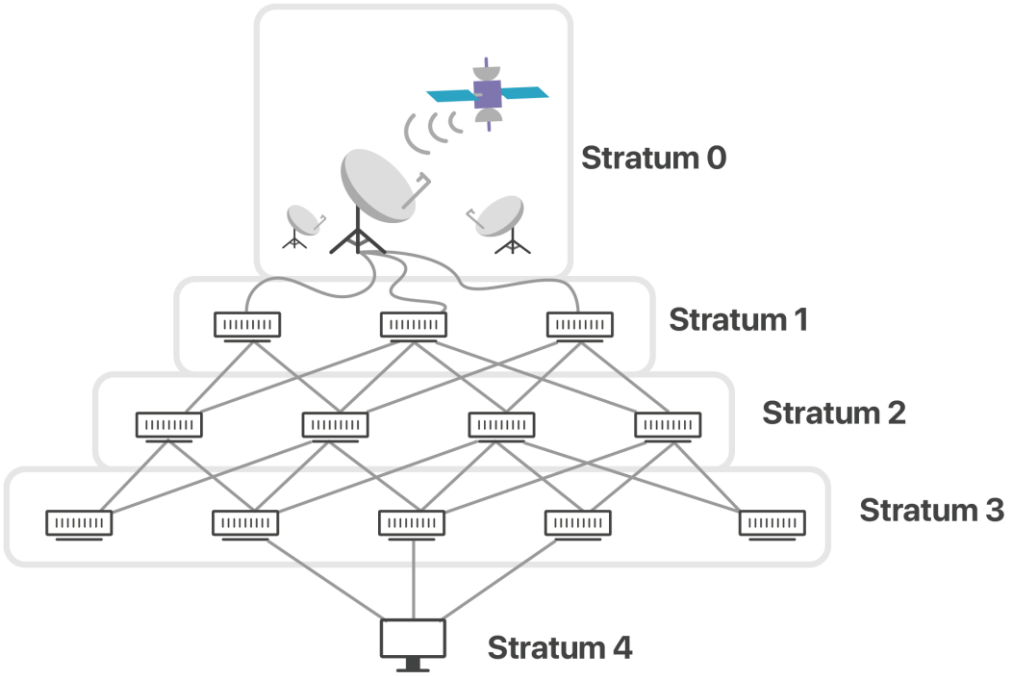
مفاهیم اولیه‌ی NTP

NTP یکی از معروف‌ترین، دقیق‌ترین و انعطاف‌پذیرترین پروتکل‌های زمانی می‌باشد. NTP برای ارائه‌ی یک زمان بسیار دقیق، از معماری چینه‌ی ساعتی (Clock Stratum) استفاده می‌کند. این معماری، چندین منبع زمانی را به صورت سلسله مراتبی در چینه‌های متفاوت قرار می‌دهد.

دستگاه‌های موجود در چینه‌ی ۰ (بالاترین لایه‌ی سلسله مراتب)، منابع زمانی بسیار دقیق، نظیر ساعت‌های اتمی، می‌باشند. منابع زمانی موجود در این چینه، فقط توسط دستگاه‌های موجود در چینه‌ی ۱ قابل دسترسی هستند.

دستگاه‌های موجود در چینه‌ی ۱، سرورهای NTP می‌باشند که زمان دقیق را به سرورهای موجود در چینه‌ی ۲ تحویل می‌دهند و سرورهای چینه‌ی ۲ نیز، زمان را به سرورهای چینه‌ی ۳ تحویل می‌دهند و به همین ترتیب تا به پایان سلسله مراتب (معمولا چینه‌ی ۱۵) برسیم.

بنابراین، هر چینه می‌تواند هزاران کلاینت NTP داشته باشد که زمان دقیق را از سرورهای NTP موجود در چینه‌ی بالاتر (یا حتی دستگاه‌های موجود در چینه‌ی خود) دریافت می‌کنند. تصویر ۲ نمای مناسبی از این مفهوم را نشان می‌دهد.



تصویر ۲- نمایی از معماری پروتکل NTP

نکته‌ی کلیدی در مورد NTP این است که هر سرور، می‌تواند به تعداد زیادی از کلاینت‌ها زمان دقیق را ارائه دهد. برای مثال، یک سرور در چینه‌ی ۱، می‌تواند ۱۰۰۰ کلاینت داشته باشد، به طوری که هر کدام از آن هزار کلاینت، خود ۱۰۰۰ هزار کلاینت دیگر دارند. علاوه بر این، اگر در یک شبکه باشیم، می‌توانیم یکی از کامپیوترهای شبکه را تبدیل به سرور NTP کنیم و از سایر سیستم‌ها بخواهیم که زمان خود را از آن سرور محلی دریافت کنند.

نکته‌ی قابل توجه در مورد چینه‌ها این است که هرشماره‌ی چینه‌ها بالاتر رود، میزان دقت زمان دریافتی از سرورهای هر چینه پایین‌تر می‌آید. برای مثال یک کلاینت در چینه‌ی ٤، به میزان کمتر از یک ثانیه با ساعت واقعی اختلاف دارد. این میزان کاهش دقت، بسیار جزئی است و برای بسیاری از کاربردها، قابل چشم‌پوشی می‌باشد، اما باید از وجود آن مطلع باشیم.

اگر بخواهیم کمی دقیق‌تر در مورد چگونگی مشخص کردن زمان سیستم توسط NTP صحبت کنیم، باید بگوییم که NTP با اندازه‌گیری زمان تاخیر چرخشی یا Round-Trip Time موجود بین بسته‌های (Packetهای) ارسالی بین سرور NTP و کلاینت NTP زمان خود را تنظیم می‌کند. ما می‌دانیم که سرور و کلاینت روی بسته‌های ارسالی به یکدیگر یک Timestamp قرار می‌دهند. در NTP، کلاینت زمان خود را پس

از محاسبه‌ی اختلاف ساعت نوشته شده در Timestamp بسته‌ی دریافتی از سرور و زمان تاخیر چرخشی موجود، تنظیم می‌کند. به عبارت دیگر، کلاینت زمان خود را با در نظر گرفتن زمان دریافتی از سرور و همچنین میزان تاخیر بین سرور و کلاینت، تنظیم می‌کند. به همین دلیل، ما باید سروری را انتخاب کنیم که کمترین میزان تاخیر با ما را داشته باشد.

برنامه‌ی NTP در لینوکس، هم می‌تواند به عنوان سرور و هم به عنوان کلاینت عمل کند. در حالت سرور، این برنامه ابتدا به یک سرور NTP متصل شده و سپس به سایر کلاینت‌ها اجازه می‌دهد که به خود او متصل شده و زمان را از او بگیرند. در حالت کلاینت، برنامه فقط به یک سرور NTP متصل شده و زمان را دریافت می‌کند. قبل از این که در مورد این برنامه صحبت کنیم، بیایید در مورد چگونگی انتخاب یک سرور زمانی مناسب صحبت کنیم.

انتخاب یک منبع زمانی مناسب

اغلب کاربران فکر می‌کنند که انتخاب یک منبع زمانی دارای شماره‌ی چینه‌ی پایین (مثلا چینه‌ی ۱) ایده‌آل‌ترین انتخاب می‌باشد، اما این امر صحت ندارد؛ چرا که اکثر کامپیوترهای معمولی اجازه‌ی دسترسی به سرورهای زمانی دارای شماره‌ی چینه‌ی پایین را ندارند. دلیل عدم وجود اجازه‌ی دسترسی، سبک نگه داشتن بار موجود روی سرورهای دارای شماره‌ی چینه‌ی پایین می‌باشد. این باعث می‌شود عملکرد کلی شبکه‌ی NTP بهبود یابد.

برای انتخاب یک سرور NTP مناسب، می‌توانیم به یکی از منابع زیر رجوع کنیم:

- سرویس‌دهنده‌ی اینترنت یا ISP شما

برخی از ISPها، یک سرورNTP اجرا کرده و به مشتریان خود قابلیت استفاده از از آن سرورها را می‌دهند. خوبی این سرورها در این است که از نظر شبکه‌ای به شما نزدیک‌تر هستند و در نتیجه تاخیر کمتری بین شما و سرور وجود خواهد داشت.

- سرور NTP توزیع مورد استفاده

برخی از توزیع‌های لینوکسی یک سرور NTP اجرا کرده و آن را در اختیار کاربران خود قرار می‌دهند. اگر از نظر شبکه‌ای به این سرورها نزدیک باشیم، این سرورها انتخاب مناسبی خواهند بود.

- لیست‌های عمومی سرور NTP

لیستی از سرورهای عمومی NTP در این سایت قابل دسترسی می‌باشند. این سرورها انتخاب مناسبی برای استفاده به عنوان سرور NTP می‌باشند، اما باید خودمان به صورت دستی سروری که از نظر شبکه‌ای به ما نزدیک‌تر می‌باشد را انتخاب کنیم.

- سرور پول‌های عمومی NTP

دامین pool.ntp.org شامل سرورهایی می‌باشد که به صورت داوطلبانه به عنوان سرورهای عمومی NTP عمل می‌کنند. دسترسی به این سرورها به صورت گردشی و بر حسب Hostname می‌باشد، پس هربار که NTP را اجرا می‌کنیم، آپدیت زمان را از یک سرور متفاوت دریافت خواهیم کرد. این امر باعث می‌شود که استفاده از سرور پول عمومی NTP با کمی ریسک همراه باشد، چرا

که ممکن است برخی از اوقات به سرورهای دارای تاخیر بالا متصل شویم، اما به طور کلی و برای اکثر کارهای معمولی، استفاده از سرور پول‌ها دقت قابل قبولی خواهد داشت.

یکی از قابلیت‌های جالب سرور پول NTP در این است که ما می‌توانیم داخل این پول، به دنبال کلاستری از سرورها که نیازمندی خاصی را برآورده می‌کنند بگردیم. برای مثال، بسیاری از توزیع‌های لینوکسی، تعدادی سرور مختص به خود درون سرور پول دارند. مثلا سرورهای NTP توزیع CentOS در آدرس centos.ntp.pool.org قابل دسترسی می‌باشد. خوبی این سرورهای مخصوص به یک توزیع در این است که به ما اجازه می‌دهند با استفاده از آنها، سرویس NTP خود را به درستی تنظیم کرده و پس از تنظیم صحیح، به سراغ انتخاب سرورهای بهتر برویم.

سایر کلاسترهای موجود در سرور پول به ما اجازه می‌دهند که سرورهایی که از نظر فیزیکی و جغرافیایی به ما نزدیک‌تر هستند را انتخاب کنیم. برای مثال اگر در قاره‌ی آسیا باشیم، می‌توانیم از سرورهای موجود در asia.pool.ntp.org استفاده کنیم. لازم به ذکر است که در صورت استفاده از pool.ntp.org، خود نرم‌افزار NTP سعی در پیدا کردن بهترین سرور می‌کند.

نکته: نزدیک‌ترین سرور به شما از نظر شبکه‌ای، لزوما از نظر جغرافیایی به شما نزدیک نمی‌باشد، چرا که ممکن است ISP شما برای رسیدن به سروری که از نظر جغرافیایی به شما نزدیک است، مسیر شبکه‌ای طولانی‌تری را دنبال کند و در نتیجه، تاخیر بیشتری داشته باشد.

- سرورهای زمان گوگل

یکی از مسائلی که در نگهداری و مدیریت زمان وجود دارد، مسئله‌ی ثانیه‌ی کبیسه (Leap Second) می‌باشد. از آنجا که در طول زمان سرعت چرخش زمین آهسته‌تر شده است، طول یک روز کامل حدود ۰,۰۰۱ ثانیه کمتر از ۲٤ ساعت می‌باشد. برای حل این مشکل در کامپیوترها، از مفهوم ثانیه‌ی کبیسه استفاده می‌شود. حدودا هر ۱۹ ماه یک بار، NTP سعی به حل ثانیه‌ی کبیسه می‌کند. در اکثر مواقع، این مسئله بدون هیچ مشکلی حل شده و ساعت‌ها به اندازه‌ی یک ثانیه به عقب کشیده می‌شوند. اما برخی از سیستم‌ها و برنامه‌ها با این مسئله مشکل دارند و نمی‌توانند ثانیه‌ی کبیسه را به درستی برطرف کنند.

برای رفع این مشکل، شرکت گوگل اقدام به ایجاد سرورهای NTP عمومی کرده است که سعی می‌کنند مقدار ثانیه‌ی کبیسه را در یک بازه‌ی زمانی معمولی، پخش کنند و بدین ترتیب، نیازی برای عقب کشیدن ساعت به مدت یک ثانیه نخواهد بود. به این کار، Leap-Smearing می‌گویند. این سرورهای گوگل در آدرس time*n*.google.com قابل دسترسی می‌باشند، به طوری که *n* عددی بین ۱ تا ٤ می‌باشد.

استفاده و تنظیم برنامه‌ی `ntp`

برنامه‌ی `ntp` سالهای سال است که در لینوکس به عنوان برنامه‌ی اصلی NTP به کار برده می‌شود، اما این برنامه محدودیت‌هایی نیز دارد؛ برای مثال این برنامه در شبکه‌های پرترافیک، به سختی می‌تواند زمان دقیق را حفظ کند و به همین دلیل، برخی نرم‌افزارهای جایگزین نظیر `chrony` به وجود آمده‌اند.

نرم‌افزار ntp روی اکثر سیستم‌ها به صورت پیش‌فرض نصب می‌باشد. اسم پکیج این برنامه ntp می‌باشد و در سیستم‌های Red Hat-based، می‌توانیم با استفاده از دستور زیر، نصب یا عدم نصب بودن آن را بررسی کنیم:

```
[root@localhost ~]# rpm -q ntp
package ntp is not installed
```

همانطور که می‌بینید این برنامه روی سیستم ما نصب نیست، پس آن را با استفاده از yum نصب می‌کنیم:

```
[root@localhost ~]# yum install ntp
...
Complete!
```

حال که برنامه روی سیستم ما نصب شد، می‌توانیم در مورد تنظیم آن صحبت کنیم.

تنظیم *ntp*

برنامه‌ی ntp یک daemon به نام ntpd دارد و فایل اصلی تنظیمات آن /etc/ntp.conf می‌باشد. این فایل دارای تعدادی دستورالعمل و همچنین آدرس تعدادی سرور NTP می‌باشد. شاید مهم‌ترین دستورالعمل موجود در این فایل، دستورالعمل server باشد که آدرس سرور NTP را مشخص می‌کند. بیایید خط‌هایی که با دستورالعمل server شروع می‌شوند را از فایل ntp.conf بیرون بکشیم:

```
[root@localhost ~]# grep -E "^server" /etc/ntp.conf
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

همانطور که می‌بینید، سرورهای پیش‌فرض انتخاب شده، سرورهای CentOS در پول NTP می‌باشند. نکته‌ی مهم در مشخص کردن سرورها از pool.ntp.org یا یکی از کلاسترهای آن، این است که باید همیشه قبل از آدرس اول، عدد ۰ را قرار دهیم (0.centos.pool.ntp.org)، قبل از آدرس دوم عدد ۱ را قرار دهیم (1.centos.pool.ntp.org) و به همین ترتیب. نکته‌ی قابل توجه این است که پس از آدرس سرور، دستورالعملی به نام iburst قرار گرفته است. این دستورالعمل سرعت همزمان‌سازی اولیه ساعت را بهبود می‌بخشد.

زمانی که سرویس ntpd استارت می‌شود، به تک‌تک سرورهای مشخص شده در ntp.conf، یک بسته ارسال می‌کند. سپس ntpd با آنالیز بسته‌های دریافت شده، سریع‌ترین سرور را به عنوان سرور NTP خود انتخاب می‌کند.

نکته: ntpd از پورت ۱۲۳ برای ارتباط با سرورها استفاده می‌کند، پس اگر مشکلی در ایجاد ارتباط داشتید، تنظیمات فایروال سیستم را چک کنید.

در سیستم‌های قدیمی‌تر لینوکس، اگر ساعت سیستم بیشتر ۱۷ دقیقه ازساعت واقعی تفاوت داشت، سرورهای NTP با سیستم ما صحبت نمی‌کردند و ما مجبور بودیم ابتدا ساعت را به صورت دستی تنظیم کرده و سپس درخواست خود را به سرور NTP ارسال کنیم. به این مشکل، مشکل Insane Time می‌گفتند.
در سیستم‌های جدیدتر، Insane Time دیگر برای ما مشکلی به وجود نمی‌آورد، اما بد نیست در مورد چگونگی آپدیت دستی ساعت با استفاده از دستور ntpdate آشنا شویم. برای استفاده از این دستور، باید آدرس یکی از سرورهای موجود در ntp.conf را داشته باشیم. ما به شکل زیر از این دستور استفاده می‌کنیم:

```
[root@localhost ~]# ntpdate 0.centos.pool.ntp.org
30 Dec 13:10:24 ntpdate[2059]: step time server 93.126.3.22 offset -312622.488069
sec
```

چه ساعت سیستم را ابتدا با ntpdate آپدیت کنیم و چه نکنیم، برای استفاده از سرویس ntpd باید آن را استارت بزنیم:

```
[root@localhost ~]# systemctl start ntpd
```

در حال حاضر اگر سیستم را ریبوت کنیم، ntpd به صورت اتوماتیک استارت نمی‌شود، پس بهتر است آن را enable نیز بکنیم:

```
[root@localhost ~]# systemctl enable ntpd
Created symlink from /etc/systemd/system/multi-user.target.wants/ntpd.service to
/usr/lib/systemd/system/ntpd.service.
```

به محض استارت شدن، ntpd شروع به هماهنگ‌سازی ساعت نرم‌افزاری سیستم می‌کند. البته برای بررسی وضعیت ساعت، بهتر است ۱۰ الی ۱۵ دقیقه صبر کنیم. برای بررسی وضعیت ساعت، از ntpstat استفاده می‌کنیم:

```
[root@localhost ~]# ntpstat
synchronised to NTP server (194.225.150.25) at stratum 3
   time correct to within 96 ms
   polling server every 64 s
```

همانطور که می‌بینید، این دستور به ما می‌گوید که به یک سرور در چینه‌ی سوم متصل شده‌ایم، زمان اصلی سیستم تا ۹۶ میلی‌ثانیه دقت دارد و همچنین هر ۶۴ ثانیه یک درخواست ارسال زمان به سرور NTP ارسال می‌کنیم.

مدیریت سرویس *ntp*

دستور ntpstat که در بخش قبل از آن استفاده کردیم، اطلاعات مناسبی در مورد دقت ساعت کنونی سیستم و آخرین زمان هماهنگ‌سازی به ما می‌دهد، اما ابزارهای دیگری برای مدیریت و مشاهده‌ی وضعیت سرویس ntp نیز موجود می‌باشد.

یکی از این دستورها، دستور ntpq می‌باشد. ما با استفاده از آپشن p- این دستور، می‌توانیم جدولی از سرورهای زمانی مورد استفاده و آخرین زمان هماهنگ سازی ساعت را مشاهده کنیم. برای مثال:

```
[root@localhost ~]# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*ntp5.mobinnet.n 62.12.173.12     2 u   59   64    1   44.924   -1.195   0.812
+77.104.70.70    194.225.150.25   3 u   59   64    1   44.897    0.348   0.455
```

همانطور که می‌بینید، در ستون remote، آدرس سرور NTP نوشته شده است. ستون refid نشان دهنده‌ی سروری می‌باشد که هر کدام از این سرورها به آن متصل هستند. ستون st نشان دهنده‌ی چینه‌ی هر سرور می‌باشد و سایر ستون‌ها نیز اطلاعات فنی‌تر نظیر آخرین زمان هماهنگ‌سازی، میزان تاخیر و... را به ما می‌دهند.

سروری که در کنار اسمش علامت ستاره (*) وجود دارد، سروری است که در حال حاضر آپدیت‌های زمانی را از آن دریافت می‌کنیم. سرورهایی که در کنار اسمشان علامت مثبت (+) قرار داشته باشد، سرورهایی هستند که از نظر زمانی، سرورهای مناسبی هستند اما در حال حاضر به آنها متصل نیستیم. سرورهایی که در

کنار اسمشان علامت‌هایی نظیر - یا x قرار داشته باشد، سرورهایی هستند که به دلایل متفاوت، مناسب هماهنگ‌سازی نیستند.

نکته: در صورت نیاز، پس از هماهنگ‌سازی ساعت نرم‌افزاری سیستم با ساعت دقیق جهانی، می‌توانیم ساعت سخت‌افزاری را برابر با آن ساعت قرار دهیم. کافی است دستور hwclock -systohc را اجرا کرده تا ساعت سخت‌افزاری ما برابر با ساعت نرم‌افزاری شود.

استفاده از برنامه‌ی *chrony*

سرویس chrony قابلیت‌های بسیار بیشتری نسبت به ntp دارد و سعی کرده تا بسیاری از مشکلات سرویس ntp را حل کند. این سرویس، می‌تواند زمان دقیق را در سیستم‌های پرترافیک یا سیستم‌هایی که برای مدت زمانی down می‌شوند، حفظ کند. علاوه بر این، chrony بسیار سریع‌تر از ntp عمل می‌کند و تنظیم آن نیز بسیار راحت‌تر می‌باشد.

chrony در برخی از توزیع‌ها به صورت پیش‌فرض نصب می‌باشد و پیشنهاد می‌شود که از آن به جای ntp استفاده شود. با توجه به این که chrony تبدیل به یک برنامه‌ی استاندارد شده، در ریپازیتوری اصلی اکثر توزیع‌ها موجود می‌باشد و ما می‌توانیم آن را با پکیج منیجر توزیع خود نصب کنیم. پس:

```
[root@localhost ~]# yum install chrony
...
Complete!
```

پس از نصب برنامه، باید این سرویس را استارت زده و همچنین آن را فعال (Enable) کرده تا پس از ریبوت سیستم، به صورت اتوماتیک استارت شود. پس:

```
[root@localhost ~]# systemctl start chronyd
[root@localhost ~]# systemctl enable chronyd
```

تنظیم *chrony*

فایل اصلی تنظیمات chrony، دارای نام chrony.conf می‌باشد و ممکن است در دایرکتوری /etc یا /etc/chrony قرار داشته باشد. معمولا نیازی به تغییر تنظیمات موجود در این فایل نیست، اما بد نیست به برخی از موارد موجود در آن نگاهی بیاندازیم.

فایل تنظیمات chrony نیز دارای تعدادی دستورالعمل می‌باشد. در این میان، دستورالعمل server یا pool، آدرس سرورهای NTP را مشخص می‌کند. بیایید نگاهی به خطوط دارای دستورالعمل server در فایل chrony.conf بیاندازیم:

```
[root@localhost ~]# grep -E "^server" /etc/chrony.conf
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

همانطور که می‌بینید، معرفی سرور NTP در chrony، کاملا شبیه به معرفی سرور در ntp می‌باشد.

نکته: دقت داشته باشید که ممکن است دستور بالا در برخی از سیستم‌ها به ما خروجی ندهد؛ در آن حالت، به جای server، برای عبارت pool جستجو کنید.

یکی دیگر از دستورالعمل‌های قابل توجه در فایل chrony.conf، دستورالعمل rtcsync می‌باشد. این دستورالعمل به chrony می‌گوید که در بازه‌های زمانی مشخص، زمان سخت‌افزاری را آپدیت کند. این دستورالعمل نیاز به آرگمان خاصی ندارد و فقط کافی است در فایل chrony.conf قرار گیرد تا شروع به کار کند:

```
[root@localhost ~]# grep -E "rtcsync" /etc/chrony.conf
rtcsync
```

همانطور که می‌بینید، این دستورالعمل در فایل ما موجود می‌باشد، پس ما دیگر مجبور نیستیم به صورت دستی زمان سخت‌افزاری سیستم را آپدیت کنیم و chrony این کار را برای ما انجام می‌دهد.

نکته: در صورت ایجاد هر گونه تغییر در فایل تنظیمات chrony، باید این سرویس را restart کنیم (systemctl restart chronyd) تا تنظیمات جدید فعال شوند.

مدیریت سرویس *chrony*

دستور chronyc ما را در مدیریت سرویس chrony یاری می‌دهد. برای مثال، برای مشاهده‌ی وضعیت سرورهای NTP، کافی است دستور chronyc sources -v را وارد کنیم:

```
[root@localhost ~]# chronyc sources v
210 Number of sources = 4
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^- ntp5.mobinnet.net             2   7   377    32  +2790us[+2790us] +/-   96ms
^- 77.104.70.70                  3   8   377    31   +592us[ +592us] +/-   90ms
^- ntp.iranet.ir                 2   8   377    97  -1189us[-1265us] +/-   84ms
^* asmanfaraz.22.3.126.93.i>     1   7   377    33   -178us[ -259us] +/-   26ms
```

همانطور که می‌بینید خروجی این دستور بسیار شبیه به خروجی دستور ntpq -p می‌باشد و نیاز به توضیح خاصی ندارد.

دستور chronyc sourcestats نیز عملکردی مشابه به بالا دارد، اما اطلاعات آماری بیشتری را به ما نشان می‌دهد:

```
[root@localhost ~]# chronyc sourcestats
210 Number of sources = 4
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
==============================================================================
ntp5.mobinnet.net          21  11   31m     +0.275      0.668  +2357us   447us
77.104.70.70               23  11   30m     +0.121      0.766    -88us   483us
ntp.iranet.ir              23  11   29m     +0.128      0.450   -396us   253us
asmanfaraz.22.3.126.93.i>  22  10   32m     +0.068      1.311  +4394ns   915us
```

یکی دیگر از دستورهای بسیار کاربردی که وضعیت هماهنگ‌سازی ساعت سیستم را به ما گزارش می‌دهد، دستور chronyc tracking می‌باشد. این دستور، معادل ntpstat می‌باشد، با این تفاوت که اطلاعات بیشتری را به ما نشان می‌دهد:

```
[root@localhost ~]# chronyc tracking
Reference ID    : 5D7E0316 (asmanfaraz.22.3.126.93.in-addr.arpa)
Stratum         : 2
Ref time (UTC)  : Fri Jan 01 07:12:56 2021
System time     : 0.000000146 seconds slow of NTP time
Last offset     : +0.000559724 seconds
RMS offset      : 0.002843956 seconds
Frequency       : 20.682 ppm slow
Residual freq   : +0.068 ppm
```

```
Skew            : 1.504 ppm
Root delay      : 0.048462156 seconds
Root dispersion : 0.002041283 seconds
Update interval : 130.0 seconds
Leap status     : Normal
```

همانطور که می‌بینید، این دستور اطلاعاتی نظیر نام و آدرس سروری که به آن متصل هستیم، شماره‌ی چینه‌ی آن سرور و... را به ما می‌دهد. دستور chronyc قابلیت‌های بسیار بیشتری نیز دارد که ما به توضیح آنها نمی‌پردازیم. پیشنهاد می‌شود manpage این دستور را مطالعه کنید.

زمان‌بندی کارها

برخی از کارهای مربوط به نگهداری سیستم، کارهای تکراری هستند که باید در دوره‌های زمانی مشخص انجام پذیرند. برای مثال، ایجاد بک‌آپ فرآیندی است که حداقل روزی یک بار تکرار می‌شود. ما می‌توانیم این قبیل کارها را با استفاده از ابزارهای زمان‌بندی در لینوکس به صورت اتوماتیک، در یک زمان مشخص و به صورت مداوم، انجام دهیم.

نام ابزاری که این قابلیت را به می‌دهد، cron می‌باشد و به هر کدام از کارهایی که توسط این ابزار در زمان مشخص اجرا می‌شوند، cronjob می‌گویند. ابزار مشابه دیگر، ابزار at می‌باشد که به ما اجازه می‌دهد یک دستور را فقط یک بار در یک زمان خاص در آینده اجرا کنیم. ما در این بخش با این دستورها و طریقه‌ی استفاده از آنها آشنا می‌شویم.

درک عملکرد *cron*

برنامه‌ی cron یک سرویس یا daemon می‌باشد؛ یعنی دائما در پشت صحنه در حال اجرا می‌باشد و منتظر رویدادی است که او را فرا بخواند. این برنامه هر دقیقه یک بار، فایل‌های تنظیمات موجود در /var/spool/cron و /etc/cron.d و همچنین فایل /etc/crontab را بررسی کرده و اگر زمان مشخص شده در آن فایل‌ها با زمان کنونی سیستم همخوانی داشته باشد، دستورات نوشته شده در این فایل‌ها را اجرا می‌کند.

ما به طور کلی دو نوع cron job داریم:

- کرون‌جاب‌های سیستمی:

کرون‌جاب‌های سیستمی، با یوزرنیم کاربر روت اجرا شده و عملیات نگهداری مربوط به کل سیستم را انجام می‌دهند. برای مثال اکثر توزیع‌ها به صورت پیش‌فرض یک کرون‌جاب سیستمی برای پاک‌کردن فایل‌های قدیمی در /tmp دارند.

- کرون‌جاب‌های کاربری:

کرون‌جاب‌های کاربری، توسط کاربران معمولی ایجاد و با یوزرنیم آنها اجرا شده و می‌توانند یک سری برنامه‌ی کاربری را در دوره‌های زمانی مشخص اجرا کنند.

نکته‌ای که باید در مورد کرون‌جاب در نظر داشته باشیم این است که کرون‌جاب‌ها بدون هیچ نظارتی اجرا می‌شوند و به همین دلیل، برنامه‌هایی که نیاز به ورودی از طرف کاربر دارند، نباید داخل یک کرون‌جاب قرار گیرند.

ایجاد کرون‌جاب‌های سیستمی

فایل /etc/crontab وظیفه‌ی مدیریت و کنترل کرون‌جاب‌های سیستمی را بر عهده دارد. بیایید به محتویات این فایل نگاهی بیاندازیم:

```
[root@localhost ~]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

همانطور که می‌بینید، در چندین خط ابتدایی این فایل، متغیرهای محیطی، نظیر $PATH و $MAILTO مشخص شده‌اند و پس از آن، مثالی برای چگونگی ایجاد یک کرون‌جاب آورده شده است. برای توضیح چگونگی ایجاد کرون‌جاب، بهتر است به یک نمونه کرون‌جاب نگاه بیاندازیم:

```
30 00  *  *  *  root  db_backup.sh
```

همانطور که می‌بینید، یک کرونجاب از ۶ بخش کلی تشکیل شده است که مفهوم هر بخش در تصویر ۳ قابل مشاهده می‌باشد.

30	00	*	*	*	root	db_backup.sh
فیلد اول	فیلد دوم	فیلد سوم	فیلد چهارم	فیلد پنجم	فیلد ششم	فیلد هفتم
دقیقه (۰-۵۹)	ساعت (۰-۲۳)	شماره‌ی روزِ ماه (۰-۳۱)	شماره ماه (۰-۱۲)	شماره روزِ هفته (۰-۷)	کاربر اجرا کننده‌ی دستور	دستور یا اسکریپتی که باید اجرا شود.

تصویر ۳ - مفهوم بخش‌های تشکیل دهنده‌ی یک کرون‌جاب

به طور کلی، پنج فیلد اول، مشخص کننده‌ی زمان اجرای یک دستور یا اسکریپت می‌باشند. از این میان، برخی از فیلدها نیاز به توضیح بیشتری دارند:

- در فیلد چهارم، به جای شماره‌ی ماه، می‌توانیم از سه حرف اول ماه نیز استفاده کنیم. یعنی jan، feb و... .
- در فیلد پنجم، شماره‌ی ۰ و همچنین شماره‌ی ۷، هر دو به روز یکشنبه اشاره دارند. به عبارت دیگر، در این فیلد، شماره‌ی ۱، نشان دهنده‌ی اولین روز هفته (دوشنبه) می‌باشد، شماره‌ی ۲ نشان دهنده‌ی دومین روز هفته (سه‌شنبه) می‌باشد و به همین ترتیب تا شماره‌ی ۷ که نشان دهنده‌ی آخرین روز هفته (یکشنبه) می‌باشد. نکته‌ی عجیب در این فیلد این است که عدد ۰ نیز، نشان دهنده‌ی آخرین روز هفته (یکشنبه) می‌باشد.

نکته‌ی قابل توجه این است که ما می‌توانیم به جای شماره‌ی روز، از سه حرف اول موجود در نام روز نیز استفاده کنیم. یعنی mon، tue، wed و... .

ما برای ایجاد یک کرون‌جاب، باید کلیه‌ی این فیلدها را پر کنیم. پر کردن فیلد ششم و هفتم نیاز به توضیح خاصی ندارند، اما برای پر کردن پنج فیلد اول، چندین آپشن داریم. طبیعتا ساده‌ترین کار، مشخص کردن دقیق یک مقدار برای کلیه‌ی این مقادیر می‌باشد. اما اگر بخواهیم کاری به صورت روزانه انجام شود، باید این مقادیر را چطور وارد کنیم؟ اگر بخواهیم کاری هر ساعت انجام شود چه؟ برای این کار، باید به سراغ روش‌های پیشرفته‌ترِ پر کردن فیلدها برویم که به شرح زیر می‌باشند:

- استفاده از علامت * در یک فیلد، به معنای «همه» باشد. یعنی اگر در فیلد اول که مشخص کننده‌ی دقیقه می‌باشد این علامت را قرار دهیم، به معنای همه‌ی دقایق یک ساعت می‌باشد.
- استفاده از آرایه‌ای از مقادیر که با کاما (,) از هم جدا شده‌اند (یعنی مثلا `10,12,14`) باعث می‌شود که cron تک تک مقادیر ذکر شده در آرایه را در نظر بگیرد.
- استفاده از دو مقدار که با یک خط تیره (-) از هم جدا شده‌اند نشان دهنده‌ی محدوده‌ای از مقادیر می‌باشد (محدوده‌ی فراگیر). برای مثال، مقدار `10-15` در فیلد دوم، به معنای کلیه‌ی ساعات موجود در این محدوده، یا ساعت ۱۰، ۱۱، ۱۲، ۱۳، ۱٤ و ۱۵ می‌باشد.
- استفاده از اسلش (/) در کنار سایر روش‌های ذکر شده در بالا، اندازه‌ی گام یا Step Size را مشخص می‌کند. اندازه‌ی گام در یک محدوده، مشخص می‌کند که از کدام مقادیر موجود در محدوده استفاده می‌کنیم. برای مثال، اگر در فیلد اول، که نشان دهنده‌ی دقیقه می‌باشد، بنویسیم `*/10`، نشان دهنده‌ی هر ۱۰ دقیقه می‌باشد. اگر در فیلد ساعت `*/2` بگذاریم، یعنی هر دو ساعت یک بار. اگر در فیلد روزِ ماه، از `10-20/2` استفاده کنیم، فقط در روزهای ۱۰، ۱۲، ۱٤، ۱۶، ۱۸ و ۲۰ ماه کار را انجام می‌دهد.

اگر بخواهیم یک اسکریپت یا یک دستور، روزی یک بار، ماهی یک‌بار و... اجرا شود، مجبور نیستیم برای آن درون فایل `/etc/crontab` یک کرون‌جاب بنویسیم. بیشتر توزیع‌ها، دارای کرون‌جاب‌های سیستمی ساعتی، روزانه، هفتگی و ماهانه می‌باشند. هر کدام از این کرون‌جاب‌ها، دارای یک دایرکتوری با نام `/etc/cron.interval` می‌باشند، به طوری که *interval* می‌تواند یکی از مقادیر `hourly`، `daily`، `weekly` و `monthly` باشد. کافی است ما اسکریپت مورد نظر خود را درون یکی از این دایرکتوری‌ها (با توجه به زمان مورد نظرمان برای اجرا) قرار دهیم تا سیستم آنها را در آن بازه، اجرا کند. ممکن است از خود بپرسید که اگر به جای یک اسکریپت، یک دستور داشته باشیم باید چه کنیم. ما در جلسات بعدی در مورد چگونگی استفاده از دستورها در قالب یک اسکریپت صحبت خواهیم کرد، پس فعلا باید صبر کنید.

نکته: برخی از توزیع‌ها، اسکریپت‌های مربوط به هر کدام از این کرون‌جابها را درون یک دایرکتوری به نام `/etc/cron.d/interval` قرار می‌دهند.

شاید برایتان سوال باشد که زمان دقیق اجرای اسکریپت‌های موجود در این دایرکتوری‌ها چه زمانی است، مثلا اگر یک اسکریپت را درون دایرکتوری `/etc/cron.daily` قرار دهیم، آن اسکریپت دقیقا در چه ساعت از روز اجرا می‌شود؟ متاسفانه این امر کاملا به توزیع مورد استفاده بستگی دارد. معمولا اجرای اسکریپت‌های

موجود در این دایرکتوری‌ها، به anacron سپرده می‌شود (که در بخش بعد در مورد آن صحبت خواهیم کرد) و معمولا در manpage مربوط به anacron، زمان دقیق اجرای این جاب‌ها نوشته می‌شود.

نکته: همیشه قبل از قرار دادن یک دستور درون یک کرون‌جاب، آن را به صورت کامل و دقیق تست کنید. از آنجایی که به احتمال زیاد هنگام اجرای کرون‌جاب، شما پای سیستم نخواهید بود، پیدا کردن مشکلات و باگهای کرون‌جاب شما هنگام اجرا، غیرممکن یا حداقل سخت خواهد بود.

ایجاد کرون‌جاب‌های کاربری

برای ایجاد کرون‌جاب‌های کاربری، ما از دستور crontab استفاده می‌کنیم. توجه کنید که این دستور، با فایل /etc/crontab تفاوت دارد. راحت ترین روش برای توضیح چگونگی عملکرد این دستور، نشان دادن syntax آن و توضیح آپشن‌های آن می‌باشد. به طور کلی، این دستور از syntax زیر پیروی می‌کند:

```
crontab [-u user] [-l | -e | -r]
```

ما می‌توانیم با استفاده از آپشن u-، فایل کرون جاب مخصوص یک کاربر خاص را مشاهده یا تغییر دهیم. استفاده از این آپشن اختیاری است و crontab به صورت پیش‌فرض، فایل کرون‌جاب کاربر کنونی را به ما نشان می‌دهد.

حال بیایید در مورد سه آپشن l-، e- و r- صحبت کنیم:

- آپشن l-، محتویات فایل کرون‌جاب‌های کاربر را نشان می‌دهد.
- آپشن r-، فایل کرون‌جاب‌های کاربر را پاک می‌کند.
- آپشن e-، فایل کرون‌جاب‌های کاربر را داخل یک ادیتور (مثل vim) باز می‌کند تا بتوانیم محتویات درون آن را تغییر دهیم یا کرون‌جاب جدیدی به آن اضافه کنیم.

به طور کلی، نحوه‌ی نوشتن کرون‌جاب‌های کاربری، دقیقا از آداب نوشتاری کرون‌جاب‌های سیستمی پیروی می‌کنند و شامل همان فیلدها می‌باشند، با این تفاوت که در کرون‌جاب‌های کاربری نیازی به مشخص کردن نام کاربر اجرا کننده‌ی دستور، نیست (به عبارت دیگر، فیلد ششم در کرون‌جاب کاربری خالی می‌ماند).

نکته: ما می‌توانیم کرون‌جاب خود را با فرمت معمول کرون‌جاب‌ها، داخل یک فایل بنویسیم و آن فایل را به عنوان یک آرگمان، به دستور crontab بدهیم. یعنی مثلا:

```
[root@localhost ~]# crontab backup_job.cron
```

به طوری که backup_job.cron، یک فایل متنی که درون آن کرون‌جاب ما نوشته شده است می‌باشد.

در ادامه، یک نمونه crontab کاربری را مشاهده می‌کنیم:

```
[root@localhost ~]# crontab -l
SHELL=/bin/bash
MAILTO=puppy
HOME=/home/puppy
0,30  *     *     *     *     ~/fetch_token.sh
0     15    *     fri   *     ~/clean_files.sh
0     0     1-7   *     *     ~/get_monthly_jobs.sh
```

همانطور که می‌بینید، کرون‌جاب اول هر ۳۰ دقیقه یک بار، اسکریپتی به نام `fetch_token.sh` را اجرا می‌کند. کرون‌جاب دوم در ساعت ۱۵:۰۰ روزهای جمعه، اسکریپت `clean_files.sh` را اجرا می‌کند و در نهایت کرون‌جاب سوم، در نیمه شب روزهای اول **تا** هفتم هر ماه، اسکریپت `get_montly_jobs.sh` را اجرا می‌کند.

به طور کلی، فایل `crontab` هر کاربر در یکی از دایرکتوری‌های زیر ذخیره می‌شوند:

```
/var/spool/cron
/var/spool/cron/tabs
/var/spool/cron/crontabs
```

هر کدام از فایل‌های موجود در این دایرکتوری، به نام کاربری که `crontab` به او تعلق دارد می‌باشد. یعنی فایل `crontab` کاربر `puppy` در سیستم‌عامل CentOS 7، در `/var/spool/cron/puppy` قابل دسترسی می‌باشد. دقت کنید که ما نباید این فایل را به صورت دستی تغییر دهیم، بلکه باید از `crontab` برای اعمال تغییرات در آن استفاده کنیم.

دسترسی به `cron` را می‌توان با استفاده از دو فایل زیر کنترل کرد:

- فایل `/etc/cron.allow`

این فایل شامل لیستی از کاربرانی می‌باشد که باید اجازه دسترسی به `cron` را داشته باشند. اگر این فایل در `/etc` وجود داشته باشد، **فقط و فقط** کاربرانی که در این فایل هستند می‌توانند از `cron` استفاده کنند و سایر کاربران اجازه‌ی دسترسی به `cron` را نخواهند داشت. اگر این فایل در سیستم موجود نباشد، همه‌ی کاربران اجازه‌ی دسترسی به `cron` را خواهند داشت.

- فایل `/etc/cron.deny`

این فایل شامل لیستی از کاربرانی می‌باشد که نباید اجازه‌ی دسترسی به `cron` را داشته باشند. اگر این فایل در سیستم موجود باشد، هر کاربری که نامش درون فایل `cron.deny` قرار گیرد، اجازه‌ی دسترسی به `cron` را نخواهد داشت. کاربرانی که نامشان در این فایل قرار ندارد، می‌توانند به راحتی از `cron` استفاده کنند.

استفاده از *anacron*

`cron` ابزار بسیار مناسبی برای انجام عملیات متفاوت در زمان‌های مشخص می‌باشد، اما بیشتر به درد سیستم‌هایی میخورد که همیشه روشن می‌باشند (نظیر سرورها). برای سیستم‌هایی نظیر کامپیوترهای شخصی که به صورت مداوم خاموش می‌شوند، `cron` زیاد آپشن مناسبی نمی‌باشد، چرا که در این حالت ممکن است بسیاری از کرون‌جاب‌ها به دلیل خاموش بودن سیستم، اجرا نشوند. اگر سیستم ما در زمان اجرای یک کرون‌جاب خاموش باشد، آن کرون‌جاب تا زمان اجرای بعدی، اجرا نخواهد شد که می‌تواند امر نامناسبی باشد.

یکی از راه‌های جلوگیری از این مشکل، استفاده از `anacron` می‌باشد. این برنامه به عنوان مکملی برای `cron` عمل می‌کند و اجرا شدن عملیات مورد نظر در فاصله‌های زمانی مناسب را تضمین می‌دهد. `anacron` با نگهداری لیستی از عملیاتی که باید انجام دهد و بازه‌ی زمانی اجرای هر عملیات، کار خود را انجام می‌دهد. بازه‌ی زمانی در `anacron`، در واحد روز می‌باشد. آناکرون در هر بار اجرا بررسی می‌کند که زمان آخرین اجرای تک‌تک عملیاتی که به او سپرده شده چه زمانی بوده است و در صورتی که به زمان اجرای دوباره‌ی یک عملیات رسیده باشد، آن را اجرا می‌کند.

ما می‌توانیم anacron را با استفاده از فایل /etc/anacrontab تنظیم کنیم. بیایید نگاهی به محتویات این فایل بیاندازیم:

```
[root@localhost ~]# cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
1                  5     cron.daily          nice run-parts /etc/cron.daily
7                  25    cron.weekly         nice run-parts /etc/cron.weekly
@monthly           45    cron.monthly        nice run-parts /etc/cron.monthly
```

همانطور که می‌بینید، در این فایل تعدادی کامنت وجود دارد (خط‌هایی که با علامت # شروع می‌شوند)، پس از آن تعدادی خط که متغیرهای محیطی را تنظیم می‌کنند قرار دارند (SHELL، PATH، MAILTO و...) و در نهایت خط‌هایی که عملیاتی که باید توسط anacron انجام شود را تعریف می‌کنند قرار گرفته‌اند.

خط‌های تعریف عملیات، دارای ٤ فیلد می‌باشند. به صورت زیر:

```
period      delay       identifier        command
```

بیایید مفهوم هر کدام از این فیلدها را بررسی کنیم:

- فیلد period:

این فیلد مشخص می‌کند که عملیات ذکر شده، چند روز یک بار باید اجرا شود. مثلا قرار دادن عدد ۱ در این فیلد، به معنای اجرای این دستور به صورت روزانه می‌باشد یا مثلا عدد ۱۸ در این فیلد، یعنی عملیات هر ۱۸ روز یک بار باید اجرا شود. علاوه بر این، ما می‌توانیم در این فیلد از یکی از عبارت‌های @daily، @weekly یا @monthly استفاده کنیم.

- فیلد delay:

این فیلد مشخص کننده‌ی مدت زمانی است که anacron از زمان اجرا شدن تا انجام یک عملیات صبر می‌کند (در واحد دقیقه). دلیل وجود این صبر، عدم overload کردن سیستم می‌باشد، چرا که anacron ممکن است هنگام اجرا، مجبور به انجام چندین عملیات به صورت همزمان شده و بدین ترتیب، سیستم را overload کند.

- فیلد identifier:

این فیلد مشخص کننده‌ی شناسه‌ی عملیات می‌باشد. این شناسه‌ای است که یک عملیات با آن در فایل‌های لاگ قابل شناسایی می‌باشد.

- فیلد command:

در این فیلد، عملیات مورد نظر (اجرای یک دستور خاص یا یک اسکریپت) نوشته می‌شود.

حال بیایید نگاهی به عملیات مشخص شده در فایل anacrontab بیاندازیم:

```
1                   5       cron.daily          nice run-parts /etc/cron.daily
7                   25      cron.weekly         nice run-parts /etc/cron.weekly
@monthly            45      cron.monthly        nice run-parts /etc/cron.monthly
```

همانطور که می‌بینید، در اینجا سه جاب وجود دارد که بسیار شبیه به هم می‌باشند. در جاب اول، آناکرون به صورت روزانه پس از اجرا شدن، ۵ دقیقه صبر کرده و سپس دستور زیر را اجرا می‌کند:

```
nice run-parts /etc/cron.daily
```

اگر از قبل به خاطر داشته باشید، دستور nice به ما کمک می‌کرد تا اولویت انجام یک پراسس (در اینجا یک دستور) درون سیستم بیشتر را یا کمتر کنیم و دستور run-parts، دستوری است که نام یک دایرکتوری را به عنوان ورودی دریافت کرده و کلیه‌ی فایل‌های اجرایی درون آن (مثلا اسکریپت‌ها) را اجرا می‌کند. پس در اینجا آناکرون به صورت روزانه کلیه‌ی اسکریپت‌های موجود در دایرکتوری /etc/cron.daily را با میزان اولویت ۱۰ (مقدار دیفالت nice) اجرا می‌کند. جاب‌های بعدی نیز دقیقا مانند این جاب می‌باشند.

نکته: اگر از بخش cron به خاطر داشته باشید، ما گفتیم که اسکریپت‌های موجود در دایرکتوری‌های cron.daily و... توسط آنا کرون اجرا می‌شوند. ما صحت این امر را در این بخش بررسی کردیم.

البته بر خلاف cron، خود anacron یک سرویس یا daemon نیست و باید به طریقی اجرا شود. در سیستم‌هایی که از سیستم‌راه‌انداز SysV استفاده می‌کنند، anacron از طریق یک Startup Script اجرا می‌شود. در سیستم‌هایی که از سیستم‌راه‌انداز systemd استفاده می‌کنند، anacron توسط خود systemd راه‌اندازی می‌شود. علاوه بر این، ما می‌توانیم anacron را به عنوان یک کرون‌جاب اجرا کنیم.

استفاده از *at*

بعضی اوقات ممکن است بخواهیم یک دستور فقط یک بار در یک زمان خاص اجرا شود. در چنین حالتی، استفاده از cron و anacron زیاد منطقی نیست و به جای آن از ابزاری به نام at استفاده می‌کنیم. این دستور در برخی از توزیع‌ها به صورت پیش‌فرض نصب نمی‌باشد، اما ما می‌توانیم به سادگی با استفاده از پکیج‌منیجرهایی نظیر yum، آن را نصب کنیم.

در ساده‌ترین حالت، دستور at فقط به یک آپشن که مشخص کننده‌ی زمان اجرا می‌باشد نیاز دارد. زمان اعمال شده به at، می‌تواند در چند فرمت متفاوت باشد:

- ساعت در فرمت HH:MM.
- کلیدواژه‌های زمانی مثل noon، midnight یا teatime (teatime اشاره به ساعت ۱۶:۰۰ دارد).
- اگر بخواهیم یک عملیات در زمانی که در محدوده‌ی ۲۴ ساعت بعدی نیست اجرا شود، باید تاریخ مورد نظر برای اجرا را در فرمت MMDDYY، MM/DD/YY یا DD.MM.YY وارد کنیم. علاوه بر این، فرمت month-name day یا month-name day year نیز قابل قبول می‌باشد.
- برای مشخص کردن یک دوره‌ی زمانی مشخص در آینده، از عبارت now به اضافه‌ی علامت + و دوره‌ی زمانی مورد نظر استفاده می‌کنیم. مثلا اگر بخواهیم عملیاتی در دو ساعت دیگر از الان انجام شود، از now + 2 hour استفاده می‌کنیم.

پس از وارد کردن at و مشخص کردن زمان مورد نظر، ابزار at، کامندلاین خودرا به ما نشان می‌دهد. ما می‌توانیم با این کامندلاین، دقیقا مانند bash رفتار کنیم و کلیه‌ی دستوراتی که میخواهیم در بازه‌ی زمانی که مشخص کردیم اجرا شوند را در اینجا وارد کنیم. پس از وارد کردن دستورات مورد نظر، کافی است دکمه‌ی Ctrl + D را فشار دهیم:

```
[root@localhost ~]# at now +5 minute
at> echo hello > text.py
at> <EOT>
job 4 at Thu Jan  7 13:12:00 2021
```

همانطور که می‌بینید، ما دستور at را با now +5 minute اجرا کردیم. این آپشن به at می‌گوید که دستورهای وارد شده را ۵ دقیقه‌ی دیگر از الان، اجرا کند. پس از وارد کردن این دستور، کامندلاین at برای ما باز شد و ما در آن دستور echo hello > text.py را وارد کردیم. سپس با زدن دکمه‌ی Ctrl + D، که به عنوان <EOT> در کامندلاین به ما نشان داده شده است، از این برنامه خارج شدیم و بدین ترتیب دستور وارد شده را برای اجرا در پنج دقیقه‌ی دیگر، زمان‌بندی کردیم. اگر دقت کنید پس از زدن دکمه‌ی Ctrl + D، شماره‌ی کار و همچنین زمان دقیق اجرای آن به ما نمایش داده شد.

به جای استفاده از کامندلاین at، ما می‌توانیم دستوراتی که میخواهیم توسط at اجرا شوند را داخل یک فایل متنی قرار داده و آن را با آپشن f- به at بدهیم. برای مثال:

```
[root@localhost ~]# cat my_commands.sh
echo hello > hello.txt
echo bye > bye.txt
```

```
[root@localhost ~]# at -f my_commands.sh midnight
job 5 at Fri Jan  8 00:00:00 2021
```

همانطور که می‌بینید، ما فایل my_command.sh که شامل یک سری دستور بود را با استفاده از آپشن f-، به دستور at دادیم و پس از آن، زمان اجرای آن را مشخص کردیم.

برای مشاهده‌ی لیستی از کارهایی که at قرار است در آینده انجام دهد، می‌توانیم از دستور atq استفاده کنیم:

```
[root@localhost ~]# atq
5     Fri Jan  8 00:00:00 2021 a root
```

همانطور که می‌بینید در خروجی این دستور، ابتدا شماره‌ی کار (۵) و سپس زمان اجرای آن به علاوه‌ی کاربری که آن را ایجاد کرده است را می‌بینیم.

برای پاک کردن هر کدام از جاب‌های at، از دستور atrm به اضافه‌ی شماره‌ی جاب استفاده می‌کنیم. بیایید جاب شماره‌ی ۵ که در بالا دیدیم را پاک کنیم:

```
[root@localhost ~]# atrm 5
```

حال بیایید از صحت پاک شدن این جاب اطمینان حاصل کنیم:

```
[root@localhost ~]# atq
```

همانطور که می‌بینید، واردکردن دستور atq به ما جوابی در خروجی نداد. این یعنی جاب شماره‌ی ۵، پاک شده است.

محدود کردن دسترسی به at را با استفاده از فایل‌های /etc/at.allow و /etc/at.deny قابل انجام می‌باشد، به طوری که:

- اگر هیچکدم از فایل‌های at.allow و at.deny در سیستم وجود نداشته باشند، فقط روت می‌تواند از at استفاده کند.
- اگر فایل at.allow وجود داشته باشد، **فقط و فقط** کاربرانی که در این فایل لیست شده‌اند می‌توانند از at استفاده کنند.
- اگر فایل at.deny وجود داشته باشد، همه‌ی کاربران، به جز کسانی که نامشان در این فایل قرار دارد می‌توانند از at استفاده کنند.

همانطور که می‌بینید، تنظیمات دسترسی برنامه‌ی at کمی متفاوت با تنظیمات دسترسی cron می‌باشد. در cron در صورت عدم وجود فایل‌های cron.allow و cron.deny، همه‌ی کاربران می‌توانستند از cron استفاده کنند، اما در at، این امر صادق نیست. این امر باعث می‌شود at از نظر امنیتی، ابزار قوی باشد.