Linux Professional Institute

LPIC-1

جلسه نهم: بوت‌لودرها و سیستم‌های راه‌انداز SysV و systemd

در این جلسه:

ویدئو اول:

- صحبت در مورد فرآیند بوت
- صحبت کلی در مورد محیط‌های گرافیکی
- آشنایی با بوت‌لودر GRUB Legacy
- صحبت اجمالی در مورد GRUB2
- صحبت در مورد چگونگی مشاهده‌ی لاگ‌های فرآیند بوت
- صحبت کلی در مورد سیستم‌های راه‌انداز
- صحبت در مورد runlevelها در SysV

ویدئو دوم:

- صحبت در مورد Startup Scriptها
- آشنایی با دستور `service`
- آشنایی با دستور `chkconfig`
- آشنایی با Password Recovery در GRUB Legacy
- آشنایی با چگونگی قرار دادن رمز روی بوت‌لودر GRUB Legacy
- صحبت در مورد قدرت Passwordها
- صحبت کلی در مورد systemd

فهرست مطالب

مقدمه

جلسه‌ی قبل، در مورد چگونگی جستجو در سیستم‌های لینوکسی صحبت کردیم. سپس به صورت خیلی کلی با مفاهیم شبکه آشنا شدیم و در نهایت چگونگی انجام و مشاهده‌ی تنظیمات شبکه در لینوکس را با هم بررسی کردیم. در این جلسه، در مورد فرآیند بوت سیستم، بوت‌لودرها و همچنین سیستم‌های راه‌اندازی، به خصوص سیستم راه‌اندازی SysV و systemd، صحبت خواهیم کرد.

آشنایی با فرآیند بوت سیستم

آیا تا به حال چگونگی روشن شدن یک سیستم توجه کرده‌اید؟ به احتمال زیاد نه؛ ما معمولا سیستم را روشن می‌کنیم، چند لحظه منتظر میمانیم و به محض مشاهده‌ی صفحه‌ی ورود، یوزرنیم و پسوورد خود را وارد کرده و شروع به استفاده از سیستم می‌کنیم. اما در آن چند لحظه انتظاری که بین روشن کردن سیستم و ورود به سیستم میکشیم، اتفاقات زیادی در پشت‌صحنه می‌افتد که به عنوان یک ادمین، باید به آن اتفاقات واقف باشیم.

به طوری کلی فرآیند بوت سیستم را می‌توانیم به صورت زیر بیان کنیم:

۱- برق به سیستم رسیده و یک سخت‌افزار خاص باعث می‌شود که CPU، کد موجود در یک موقعیتِ از پیش‌تعیین شده را اجرا کند. این کد، برنامه‌ی کوچکی به نام Firmware می‌باشد.

۲- Firmware، کلیه‌ی سخت‌افزارهای سیستم را به طور اجمالی بررسی می‌کند. به این بررسی، POST (مخفف Power-On Self-Test) می‌گویند. پس از بررسی سخت‌افزار، Firmware به دنبال یک برنامه‌ی بوت‌لودر از روی یک دستگاه bootable (هارددیسک، سی‌دی رام و...) می‌گردد.

۳- برنامه‌ی بوت‌لودر، فرماندهی عملیات بوت را از Firmware می‌گیرد و یک کرنل یا یک بوت‌لودر دیگر را اجرا می‌کند.

٤- کرنل اجرا شده توسط بوت‌لودر درون RAM قرار گرفته و شروع به آماده‌سازی سیستم می‌کند (یعنی مثلا پارتیشن `root` را مانت می‌کند و...) و سپس برنامه‌ی راه‌اندازی اولیه (initialization) را اجرا می‌کند. برنامه‌ی راه‌اندازی اولیه، کلیه‌ی برنامه‌های مورد نیاز جهت عملکرد صحیح سیستم را اجرا می‌کند (یعنی مثلا اگر سیستم ما یک وب‌سرور باشد، برنامه‌ی مخصوص وب‌سرور را اجرا می‌کند و...).

این فرآیند در نگاه اول ساده به نظر می‌آید، اما تک‌تک مراحل این فرآیند، پیچیدگی‌های خاص خود را دارند؛ در واقع هر کدام از این مراحل، خود به چندین مرحله‌ی دیگر تقیسم می‌شوند تا بتوانند سیستم را به درستی راه‌اندازی کنند.

مشاهده‌ی فرآیند بوت لینوکس

برای مشاهده‌ی فرآیند بوت لینوکس، کافی است هنگام روشن کردن سیستم، به مانیتور خود نگاه کنیم! البته مشاهده‌ی اطلاعات از این صفحه هنگام روشن شدن سیستم کار بسیار دشواری می‌باشد، چرا که این اطلاعات به سرعت از صفحه عبور می‌کنند. یکی از روش‌های مشاهده‌ی اطلاعات بوت سیستم، استفاده از دستور `dmesg` می‌باشد:

```
[root@localhost ~]# dmesg | less
...
[   30.360041] ip_set: protocol 7
[   30.473797] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
```

```
[    30.514685] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
None
[    30.555936] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
[    30.555988] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[    31.374068] nf_conntrack version 0.5.0 (7779 buckets, 31116 max)
```

همانطور که می‌بینید، خروجی این دستور بسیار طولانی می‌باشد و اگر بخواهیم آن را به صورت کامل از ابتدا مشاهده کنیم، باید آن را درون دستور less، پایپ کنیم. جالب است بدانید که dmesg، اطلاعات مربوط به عملکرد کرنل را درون خود نگاه‌داری می‌کند. اگر بخواهیم فنی‌تر در مورد آن صحبت کنیم، باید بگوییم که dmesg اطلاعات موجود در Kernel Ring Buffer را در خروجی به ما نشان می‌دهد. Kernel Ring Buffer همیشه یک سایز ثابت دارد و پیام‌های قدیمی‌تر را به محض آمدن پیام‌های جدیدتر، پاک می‌کند (در صورتی که پیام‌ها از سایز ثابت آن فراتر رفته باشند).

یکی دیگر از روش‌های کسب اطلاعات در مورد سیستم و چگونگی بوت شدن آن، استفاده از دستور journalctl می‌باشد:

```
[root@localhost ~]# journalctl
-- Logs begin at Tue 2020-09-01 22:38:35 +0430, end at Tue 2020-09-01 23:01:01 +0430. --
Sep 01 22:38:35 localhost.localdomain systemd-journal[107]: Runtime journal is using 6.0M
(max allowed
Sep 01 22:38:35 localhost.localdomain kernel: Initializing cgroup subsys cpuset
Sep 01 22:38:35 localhost.localdomain kernel: Initializing cgroup subsys cpu
...
Sep 01 23:01:01 localhost.localdomain run-parts(/etc/cron.hourly)[1573]: finished 0anacron
Sep 01 23:01:01 localhost.localdomain anacron[1571]: Will run job `cron.daily' in 29 min.
Sep 01 23:01:01 localhost.localdomain anacron[1571]: Will run job `cron.weekly' in 49 min.
Sep 01 23:01:01 localhost.localdomain anacron[1571]: Jobs will be executed sequentially
```

همانطور که می‌بینید، خروجی این دستور نیز بسیار طولانی می‌باشد، اما این دستور به صورت اتوماتیک توسط less به ما نشان داده می‌شود، پس نیازی به پایپ کردن آن نداریم. اگر نگاهی اجمالی به خروجی این دستور بیاندازیم، می‌بینیم که این دستور در مورد تک‌تک سخت‌افزارها و بخش‌های مختلف سیستم، چگونگی شروع آنها و... به ما اطلاعاتی می‌دهد. خروجی این دستور از رنگ نیز استفاده می‌کند که می‌تواند ما را در پیدا کردن قسمت‌های مشکل‌دار سیستم، یاری دهد.

علاوه بر این دو دستور، می‌توانیم با نگاه کردن به فایل /var/log/boot.log نیز فرآیند بوت را بررسی کنیم. این فایل مرحله به مرحله‌ی بوت شدن سیستم را از نظر روشن شدن سرویس‌های متفاوت، به ما نشان می‌دهد. توجه کنید که بعضا ممکن است خود فایل /var/log/boot.log خالی باشد. در چنین حالتی، باید در همان دایرکتوری، به دنبال فایل boot.log به همراه تاریخ مورد نظر خود بگردیم. مثلا برای ما، فایل /var/log/boot.log-20200901 اطلاعات مربوط به بوت در روز اول سپتامبر ۲۰۲۰ را درون خود دارد. بیایید نگاهی به این فایل بیاندازیم:

```
[root@localhost ~]# less -R /var/log/boot.log-20200901
[  OK  ] Mounted Configuration File System.
[  OK  ] Reached target System Initialization.
[  OK  ] Started Show Plymouth Boot Screen.
...
[  OK  ] Started Login Service.
[  OK  ] Started Permit User Sessions.
         Starting Wait for Plymouth Boot Screen to Quit...
[  OK  ] Started Command Scheduler.
         Starting Terminate Plymouth Boot Screen...
```

همانطور که می‌بینید، خروجی این دستور به ما می‌گوید که هنگام روشن شدن سیستم، چه سرویس‌هایی استارت می‌شوند، چه سرویس‌های استاپ می‌شوند و نتیجه‌ی این استارت و استاپ شدن چه می‌باشد (OK، FAILED و ...).

از آنجایی که این فایل بسیار طولانی می‌باشد، ما مجبوریم برای خواندن آن، از دستور less استفاده کنیم. اگر دقت کنید، می‌بینید که ما less را با آپشن -R اجرا کرده‌ایم. دلیل استفاده از -R، این است که less بتواند خط‌های دارای رنگ یا فرمتینگ خاص را به ما نشان دهد.

نکته: ما معمولا به مطالعه‌ی کامل خروجی دستور dmesg، journalctl یا فایل boot.log نمی‌پردازیم؛ چون خروجی این دستورها بسیار طولانی می‌باشد و در نتیجه، پیدا کردن مشکلات از میان آنها کار بسیار دشواری می‌باشد. به جای آن، معمولا خروجی این دستورها (یا فایل) را درون برنامه‌ی grep پایپ کرده و به دنبال اطلاعات مربوط به سخت‌افزارها (مثلا /dev/sda1)، برخی کلمات کلیدی (مثلا disabled) یا برخی از آیتم‌های خاص (نظیر BOOT_IMAGE) می‌گردیم. بدین شکل، ما می‌توانیم خیلی سریع‌تر مشکل سیستم را پیدا کنیم. نگاه‌کردن به رنگ خطوط (در فایل‌هایی که خطوط رنگی دارند) نیز می‌تواند ما را در پیداکردن مشکلات یاری دهد؛ معمولا خط قرمز یعنی در جایی از سیستم مشکلی وجود دارد.

Firmwareها

این که بدانیم پیام‌های مربوط به فرآیند راه‌اندازی سیستم در چه جاهایی ذخیره می‌شوند، بسیار کاربردی می‌باشد، اما این که بدانیم دقیقا چه چیزی این پیام‌ها را ایجاد می‌کند نیز بسیار به کمک ما می‌آید. در این بخش، ما می‌خواهیم در مورد هر کدام از مراحل فرآیند بوت به صورت جزئی صحبت کنیم.

اکثر کامپیوترها، از یک Firmware داخلی برای کنترل و مدیریت چگونگی اجرای سیستم عامل استفاده می‌کنند. نام این Firmware در سیستم‌های قدیمی‌تر، BIOS یا Basic Input/Output System می‌باشد و در سیستم‌های جدیدتر، این Firmware با نام UEFI یا Unified Extensible Firmware Interface شناخته می‌شود. ما قبلا در مورد BIOS و UEFI صحبت کرده بودیم، اما بار دیگر به بررسی این دو می‌پردازیم.

فرآیند بوت سیستم با BIOS

BIOS که هنوز هم در برخی از کامپیوترهای قدیمی موجود می‌باشد، از نظر قابلیت‌ها و عملکرد، بسیار محدود می‌باشد. بایوس یک اینترفیس گرافیکی بسیار ساده دارد که به ما اجازه می‌دهد تا برخی از تنظیمات مربوط به سخت‌افزارهای سیستم و همچنین چگونگی اجرای سیستم‌عامل را از طریق آن تنظیم کنیم. همانطور که قبلا هم گفتیم، یکی از محدودیت‌های بایوس این است که فقط می‌تواند اطلاعات موجود در اولین سکتور هارد دیسک را بخواند. مسلما فضای یک سکتور برای قرار دادن یک سیستم‌عامل کافی نیست؛ پس به همین دلیل، بسیاری از سیستم‌عامل‌ها پروسه‌ی بوت را در دو مرحله اجرا می‌کنند.

یعنی بایوس ابتدا بوت‌لودر (bootloader) را اجرا می‌کند و سپس بوت‌لودر به سراغ راه‌اندازی سیستم عامل می‌رود. بوت‌لودر یک برنامه‌ی کوچک می‌باشد که سخت‌افزارهای مورد نیاز برای اجرا کردن سیستم عامل را پیدا و آنها را روشن می‌کند. سیستم عامل معمولا بر روی همان هارددیسکی که بوت‌لودر روی آن وجود دارد قرار می‌گیرد، اما بعضا ممکن است سیستم‌عامل بر روی هارددیسک یا هر دستگاه جانبی دیگری قرار

داشته باشد. بوت‌لودر معمولا یک فایل تنظیمات دارد که ما از طریق آن، می‌توانیم به بوت‌لودر بگوییم در کجا به دنبال سیستم‌عامل بگردد.

برای این که کلیه‌ی کارهایی که گفتیم انجام شود، بایوس باید بداند که بوت‌لودر را از روی کدام دستگاه ذخیره‌سازی می‌تواند پیدا کند. اکثر سیستم‌های بایوس به ما اجازه می‌دهند که بوت‌لودر را روی دستگاه‌های ذخیره‌سازی زیر قرار دهیم:

- هارددیسک اینترنال
- هارددیسک اکسترنال
- CD یا DVD
- فلش‌مموری‌ها
- یک سرور تحت شبکه

اگر بخواهیم بایوس، بوت‌لودر را از روی یک هارددیسک پیدا کند، باید به بایوس بگوییم که کدامین هارددیسک، بوت‌لودر را روی خود ذخیره کرده است تا بایوس MBR یا Master Boot Record موجود بر روی آن هارددیسک را بخواند.

MBR، اولین سکتور موجود بر روی یک هارددیسک می‌باشد. همانطور که گفتیم، بایوس فقط می‌تواند اطلاعات موجود بر روی اولین سکتور موجود بر روی هارددیسک را بخواند؛ پس بایوس هنگام برخورد با یک هارددیسک، به دنبال MBR موجود بر روی هارددیسک رفته و برنامه‌ای که در MBR ذخیره شده (همان بوت‌لودر) را درون RAM قرار می‌دهد. از آنجایی که بوت‌لودر باید درون یک سکتور جا بگیرد، نمی‌تواند برنامه‌ی حجیمی باشد و در نتیجه، نمی‌تواند قابلیت‌های زیادی داشته باشد. این بوت‌لودر، می‌تواند مستقیما به موقعیت فایل کرنل سیستم‌عامل که در یک پارتیشن دیگر از سیستم موجود می‌باشد اشاره کند. لازم به ذکر است که کرنل سیستم‌عامل می‌تواند هر سایزی داشته باشد و محدودیت حجمی خاصی ندارد.

نکته: بوت‌لودر مجبور نیست به موقعیت کرنل سیستم‌عامل اشاره کند؛ بوت‌لودر می‌تواند به هر نوع برنامه‌ی دیگر، اعم از یک بوت‌لودر دیگر نیز دیگر اشاره کند. ما می‌توانیم کاری کنیم که بوت‌لودر، به موقعیت یک بوت‌لودر موجود در یک قسمت دیگر از هارددیسک اشاره کند و بدین صورت، محدودیت حجمی بایوس را دور بزنیم و همچنین امکان انتخاب از بین چند سیستم‌عامل را داشته باشیم. به چنین کاری chainloading می‌گویند، چرا که دو یا چند بوت‌لودر را در یک زنجیره یا chain قرار می‌دهیم.

فرآیند بوت سیستم با UEFI

با این که بایوس محدودیت‌های زیادی داشت، سال‌های سال سازندگان کامپیوتر با محدودیت‌های آن کنار آمدند و از آن استفاده کردند. شرکت اینتل در سال ۱۹۹۸، سیستم EFI یا Extensible Firmware Interface را جهت رفع محدودیت‌های بایوس ارائه کرد. پذیرش EFI فرآیند آهسته‌ای داشت، اما تا سال ۲۰۰۵، اکثر سازندگان کامپیوتر نیاز به EFI را حس کردند و Universal EFI یا UEFI تبدیل به استانداردی شد که همه‌ی سازندگان می‌توانستند از آن استفاده کنند. امروزه تقریبا همه‌ی کامپیوترها و سرورها از سیستم UEFI برای بوت سیستم استفاده می‌کنند.

UEFI، به جای این که به دنبال بوت‌لودر بر روی یک سکتور بگردد، یک پارتیشن مخصوص که به آن ESP یا EFI System Partition می‌گویند را ایجاد کرده و بوت‌لودر را درون آن پارتیشن قرار می‌دهد. این باعث می‌شود که برنامه‌ی بوت‌لودر، بتواند هر حجمی داشته باشد و ما به راحتی بتوانیم چندین بوت‌لودر برای سیستم‌عامل‌های متفاوت را روی سیستم داشته باشیم.
پارتیشن ESP، از فایل‌سیستم FAT برای ذخیره‌ی برنامه‌ی بوت‌لودر استفاده می‌کند. در سیستم‌های لینوکسی، پارتیشن ESP معمولا در موقعیت `/boot/efi` مانت و فایل‌های مربوط به بوت‌لودر معمولا با پسوند `.efi` در آن ذخیره می‌شوند.

نکته: برخی از توزیع‌های لینوکس، هنوز از UEFI پشتیبانی نمی‌کنند. پس اگر از سیستمِ دارایِ UEFI استفاده می‌کنید، مطمئن شوید که توزیع مورد نظر شما، از UEFI پشتیبانی می‌کند. اگر بخواهیم ببینیم که سیستم ما از UEFI استفاده می‌کند یا نه، کافی است دستور `ls /sys/firmware/efi` را وارد کنیم. اگر پیامی با مضمون `no such file or directory` را در خروجی دریافت کردیم، یعنی سیستم ما از BIOS استفاده می‌کند؛ اما اگر یک سری فایل در خروجی دریافت کردیم، یعنی سیستم ما از UEFI استفاده می‌کند.

به محض این که Firmware (بایوس یا UEFI) بوت‌لودر را پیدا و آن را اجرا کند، کارش تمام شده است و سایر کارها به عهده‌ی بوت‌لودر می‌باشد.

بوت‌لودرها

همانطور که گفتیم، بوت‌لودر به عنوان پُلی بین Firmware سیستم و کرنل سیستم‌عامل عمل می‌کند. سیستم‌عامل لینوکس، چندین بوت‌لودر متفاوت دارد. یکی از این بوت‌لودرها، GRUB (مخفف GRand Unified Bootloader) می‌باشد که در سال ۱۹۹۹ ایجاد و تبدیل به یکی از بوت‌لودرهای پراستفاده شده است. امروزه به این بوت‌لودر، GRUB Legacy می‌گویند، چرا که در سال ۲۰۰۵، بوت‌لودر دیگری به نام GRUB 2 برای سیستم‌های لینوکسی ایجاد شد و بسیاری از توزیع‌ها، به سراغ استفاده از این بوت‌لودر رفتند. GRUB 2، نسخه‌ی از اول نوشته شده‌ی GRUB Legacy می‌باشد و قابلیت‌های پیشرفته‌تری را همراه خود دارد. ما در این بخش، به توضیح این دو نوع بوت‌لودر می‌پردازیم.

GRUB Legacy

بوت‌لودر GRUB Legacy برای ساده‌سازی فرآیند ایجاد منوهای بوت و همچنین اعمال آپشن‌های متفاوت به کرنل‌ها ایجاد شده است. GRUB Legacy به ما این امکان را می‌دهد که بتوانیم با استفاده از یک منو، از بین چندین کرنل یا سیستم‌عامل متفاوت، یکی را برای اجرا انتخاب کنیم. علاوه بر این، GRUB Legacy یک شِل نیز در اختیار ما قرار می‌دهد که از طریق آن می‌توانیم از بین چندین کرنل یا سیستم‌عامل، یکی را انتخاب کنیم و آن را با آپشن‌های مورد نظر اجرا کنیم.

نکته: با توجه به این که CentOS 7 از GRUB 2 استفاده می‌کند، جهت تنظیم GRUB Legacy و صحبت در مورد آن، مجبور هستیم از CentOS 6 استفاده کنیم. می‌توانید این سیستم‌عامل را از اینجا دانلود کنید. ما قبلا در مورد چگونگی نصب سیستم‌عامل به عنوان یک ماشین مجازی صحبت کرده‌ایم، پس دیگر به توضیح آن نمی‌پردازیم. پیشنهاد می‌کنیم که نسخه‌ی Minimal سیستم‌عامل CentOS 6 را نصب کنید.

هم منو و هم شِل GRUB مجموعه‌ای از دستورات متفاوت دارند که ویژگی‌های متفاوت بوت‌لودر را فعال یا غیر فعال می‌کند. ما تا به ایجا زیاد در مورد منو صحبت کرده‌ایم، اما این منو چیست؟

ممکن است هنگام روشن کردن سیستم با چنین صفحه‌ای مواجه شده باشید:

تصویر ۱- با زدن هر دکمه‌ای روی کیبورد، به منوی GRUB Legacy برده می‌شویم.

اگر هگام مشاهده‌ی این صفحه، یکی از دکمه‌های کیبورد را فشار دهید، به صفحه‌ی نظیر تصویر ۲ برده می‌شوید. این، منوی بوت‌لودر می‌باشد:

تصویر ۲- منوی بوت‌لودر GRUB Legacy

ما می‌توانیم به GRUB بگوییم که در این منو، چه مواردی را به ما نشان دهد. در ادامه، درباره‌ی چگونگی تنظیم GRUB Legacy صحبت می‌کنیم.

تنظیم GRUB Legacy

برای تنظیم موارد نمایش داده شده در منوی GRUB Legacy، باید فایل تنظیمات GRUB را تغییر دهیم. تنظیمات GRUB در اکثر توزیع‌های لینوکسی، در فایل `/boot/grub/menu.lst` قرار گرفته است. در توزیع‌های Red Hat-based، تنظیمات GRUB در فایل `/boot/grub/grub.conf` قرار گرفته است. البته در این سیستم‌ها، فایلی به نام `menu.lst` نیز وجود دارد، اما این فایل در واقع به فایل `grub.conf` سافت‌لینک شده است.

بیایید نگاهی به محتویات این فایل بیاندازیم:

```
[root@localhost ~]# cat /boot/grub/grub.conf
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/mapper/VolGroup-lv_root
#          initrd /initrd-[generic-]version.img
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title CentOS 6 (2.6.32-754.el6.x86_64)
       root (hd0,0)
       kernel /vmlinuz-2.6.32-754.el6.x86_64 ro root=/dev/mapper/VolGroup-lv_root
rd_NO_LUKS       LANG=en_US.UTF-8       rd_NO_MD       rd_LVM_LV=VolGroup/lv_swap
SYSFONT=latarcyrheb-sun16       crashkernel=auto       rd_LVM_LV=VolGroup/lv_root
KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
       initrd /initramfs-2.6.32-754.el6.x86_64.img
```

فایل تنظیمات GRUB Legacy، از دو بخش تشکیل شده است:

- متغیرهای گلوبال

متغیرهایی هستند که رفتار و عملکرد کلی منوی GRUB را مشخص می‌کنند. این تعاریف باید در ابتدای فایل تنظیمات قرار گیرند.

- متغیرهای مربوط به بوت یک سیستم‌عامل

متغیرهایی هستند که اطلاعات مربوط به هر سیستم‌عامل موجود بر روی سیستم را مشخص می‌کنند.

بیایید ابتدا در مورد متغیرهای گلوبال صحبت کنیم. درفایل تنظیمات GRUB Legacy، بهتر است کلیه‌ی متغیرهای گلوبال را در ابتدی فایل قرار دهیم. به طور کلی، تشخیص متغیرهای گلوبال کار دشواری نمی‌باشد، چرا که در GRUB Legacy، تعداد محدودی متغیر گلوبال وجود دارد که آنها را در جدول ۱ می‌بینیم:

جدول ۱- متغیرهای گلوبال GRUB Legacy و عملکرد آنها

متغیر گلوبال	عملکرد
color	رنگ پس‌زمینه صفحه‌ی منو و رنگ نوشته‌های موجود در منو را تنظیم می‌کند.
default	گزینه‌ای که در منو به صورت پیش‌فرض انتخاب می‌شود را مشخص می‌کند.
fallback	به عنوان نقشه‌ی B عمل می‌کند؛ اگر گزینه‌ی default به درستی اجرا نشود (بوت‌لودر نتواند کرنل آن را پیدا کند و...)، گزینه‌ی مشخص شده در fallback اجرا می‌شود.
hiddenmenu	در صورت وجود این متغیر، به جای نشان دادن منوی GRUB، صفحه‌ای نظیر تصویر ۱ به کاربر نشان داده می‌شود. اگر این گزینه وجود نداشته باشد، منوی GRUB به صورت مستقیم (تصویر ۲) به کاربر نشان داده می‌شود.

spashimage	به عکسی اشاره می‌کند که می‌تواند در پس‌زمینه‌ی صفحه‌ی منو نشان داده شود.
timeout	مدت زمانی که سیستم منتظر انتخاب می‌ماند تا گزینه‌ی پیش‌فرض موجود در منو را انتخاب کند را مشخص می‌کند.

نکته: در فایل grub.conf، کلیه‌ی خط‌هایی که با علامت # شروع می‌شوند، به عنوان کامنت در نظر گرفته می‌شوند؛ یعنی توسط GRUB خوانده نمی‌شوند.

برای اعمال مقدار به یکی از متغیر گلوبال، کافی است نام متغیر را درون فایل grub.conf نوشته و مقدار مورد نظر را در مقابل آن بنویسیم. برای مثال:

```
timeout 25
```

بدین شکل، هنگام روشن شدن سیستم، ۲۵ ثانیه زمان داریم تا با زدن یک دکمه (تصویر ۱) منوی GRUB Legacy را مشاهده کنیم (تصویر ۲).

از آنجایی که نشان دادن خروجی و عملکرد تک تک متغیرهای گلوبال بسیار وقت‌گیر می‌باشد، ما تست عملکرد سایر متغیرهای گلوبال نظیر color و... را به خودتان می‌سپاریم.

پس از مشخص کردن متغیرهای گلوبال، باید متغیرهای مربوط به هر سیستم‌عامل را نیز درون فایل grub.conf وارد کنیم. ما باید برای هر سیستم‌عامل، یک سری متغیر جداگانه تعریف کنیم. تنظیمات خیلی زیادی برای مشخص کردن چگونگی پیدا کردن کرنل سیستم‌عامل‌ها توسط بوت‌لودر وجود دارد، اما برای تعریف یک سیستم‌عامل، کافی است متغیرهای زیر را به خاطر بسپاریم:

- **title**: اولین خطی که هنگام تعریف یک سیستم عامل می‌نویسیم می‌باشد. این متغیر، نام گزینه‌ی مربوط به این سیستم‌عامل در منوی بوت (تصویر ۲) را مشخص می‌کند.
- **root**: هارددیسک و پارتیشن boot (پارتیشنی که دایرکتوری /boot/ درون آن قرار دارد) را مشخص می‌کند. GRUB Legacy، هارددیسک‌ها و پارتیشن‌ها را کاملا متفاوت با لینوکس نام‌گذاری می‌کند. برای مثال:

```
root (hd0,0)
```

این یعنی پارتیشن بوت، بر روی پارتیشن اول، روی هارددیسک اول متصل به سیستم موجود می‌باشد. ما در محیط لینوکس، به چنین هارددیسک و پارتیشنی، sda1 (یا hda1، با توجه به نوع اتصال هارددیسک به سیستم) می‌گفتیم.

به طور کلی:

  - در GRUB Legacy، همه‌ی دیسک‌ها، چه PATA باشند، چه SATA یا SCSI، با استفاده از حروف hd مشخص می‌شوند.
  - در GRUB Legacy، شماره‌گذاری همه‌ی هارددیسک‌ها و پارتیشن‌ها از شماره‌ی صفر شروع می‌شود؛ در حالی که در لینوکس، هارددیسک‌ها با حروف و پارتیشن‌های آنها از شماره‌ی ۱ نام‌گذاری می‌شدند (هارددیسک اول، پارتیشن اول -> sda1، هارددیسک اول پارتیشن دوم -> sda2، هارددیسک دوم پارتیشن اول -> sdb1 و...).

- در GRUB Legacy، پارتیشن‌های Extended از شماره‌ی ٤ نام‌گذاری می‌شوند. این یعنی اگر فقط ۱ پارتیشن Primary داشته باشیم و بقیه‌ی پارتیشن‌ها Extended باشند، باز هم باید پارتیشن‌های Extended را از شماره‌ی ٤ نام‌گذاری کنیم.
- در GRUB Legacy، شماره‌ی اول، نشان دهنده‌ی شماره‌ی هارددیسک و شماره‌ی دوم، نشان‌دهنده‌ی شماره‌ی پارتیشن روی آن هارددیسک می‌باشد.

برای درک بهتر طریقه‌ی نام‌گذاری پارتیشن‌ها در GRUB Legacy، به جدول زیر توجه کنید:

جدول ۲-چگونگی نام‌گذاری هارددیسک‌ها و پارتیشن‌ها در GRUB Legacy و لینوکس

نام در GRUB	معنی	نام در لینوکس
(hd0)	اولین هارددیسک در سیستم	/dev/sda یا /dev/hda
(hd1)	دومین هارددیسک در سیستم	/dev/sdb یا /dev/hdb
(hd0,0)	اولین هارددیسک، اولین پارتیشن	/dev/sda1 یا /dev/hda1
(hd1,0)	دومین هارددیسک، اولین پارتیشن	/dev/sdb1 یا dev/hdb1/
(hd1,1)	دومین هارددیسک، دومین پارتیشن	/dev/sdb2 یا /dev/hdb2

- **kernel**: موقعیت کرنل سیستم‌عامل که در فولدر boot/ ذخیره شده را مشخص می‌کند (به صورت relative).
- **initrd**: موقعیت RAM disk (Initial RAM Disk) مربوط به سیستم‌عامل، که شامل درایورهای مورد نیاز کرنل جهت راه‌اندازی سیستم می‌باشد را مشخص می‌کند. توضیح initrd می‌تواند کمی دشوار باشد. اگر به خاطر داشته باشید، گفتیم که بوت‌لودر، کرنل سیستم‌عامل را پیدا می‌کند و کرنل وظیفه‌ی انجام سایر مراحل راه‌اندازی سیستم را بر عهده دارد. کرنل ممکن است برای انجام سایر مراحل راه‌اندازی سیستم، احتیاج به یک سری درایور، مثل درایورهای شبکه و... داشته باشد. با توجه به موقعیت قرارگیری این درایورها، کرنل ممکن است قابلیت لود کردن آنها را نداشته باشد؛ به همین دلیل، ما یک فایل‌سیستم اولیه به نام initrd ایجاد می‌کنیم که ماژول‌های مورد نیاز کرنل جهت دسترسی به سایر سخت‌افزارها را درون خود دارد.

پس بوت‌لودر پس از پیداکردن موقعیت کرنل، فایل‌سیستم initrd را به کرنل می‌دهد، کرنل initrd را مانت می‌کند و ماژول‌های مورد نیاز خود برای راه‌اندازی سیستم را از روی آن بر میدارد. سپس کرنل initrd را بیرون انداخته و پارتیشن root اصلی را جایگزین آن می‌کند.

نکته: تا قبل از نسخه‌ی ۲,۶ کرنل، فایل initrd ماژول‌های مورد نیاز کرنل جهت راه‌اندازی سیستم را درون خود داشت. از نسخه‌ی ۲,۶ به بعد، فایل initramfs (Initial RAM Filesystem) ماژول‌های مورد نیاز کرنل جهت راه‌اندازی سیستم را درون خود نگهداری می‌کند. به عبارت دیگر، از نسخه‌ی ۲,۶ به بعد، فایل initramfs جایگزین initrd شده است. با این حال، چه از initrd استفاده کنیم و چه از initramfs، هنگام معرفی این فایل‌ها به GRUB Legacy، باید از متغیر initrd استفاده کنیم.

- **rootnoverify**: پارتیشن‌های بوت مربوط به سیستم‌عامل‌های غیر لینوکسی (مثلا ویندوز و...) را مشخص می‌کند.

حال که به درک بهتری از متغیرهای مخصوص سیستم‌عامل رسیدیم، بیایید به محتویات بخش سیستم‌عامل در فایل grub.conf سیستم CentOS 6 خود نگاهی بیاندازیم:

```
[root@localhost ~]# cat /boot/grub/grub.conf
...
   title CentOS 6 (2.6.32-754.el6.x86_64)
   root (hd0,0)
   kernel /vmlinuz-2.6.32-754.el6.x86_64 ro
   root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS LANG=en_US.UTF-8
   rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16
   crashkernel=auto rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc
   KEYTABLE=us rd_NO_DM rhgb quiet
   initrd /initramfs-2.6.32-754.el6.x86_64.img
```

همانطور که در فایل grub.conf می‌بینید، نام نمایش داده شده برای سیستم‌عامل ما در منوی بوت‌لودر، CentOS 6 (2.6.32-754.el6.x86_64) خواهد بود:

```
title CentOS 6 (2.6.32-754.el6.x86_64)
```

سپس، می‌بینیم که پارتیشن بوت این سیستم‌عامل، برروی اولین پارتیشن موجود بر روی اولین هارددیسک قرار دارد:

```
root (hd0,0)
```

پس از این خط، موقعیت کرنل لینوکس مشخص شده است. همانطور که می‌بینید، کرنل لینوکس در فایلی به نام vmlinuz-2.6.32-754.el6.x86_64 در دایرکتوری /boot قرار دارد:

```
kernel /vmlinuz-2.6.32-754.el6.x86_64
```

همانطور که گفتیم، موقعیت کرنل به صورت relative نسبت به /boot مشخص می‌شود. این یعنی در اینجا، کرنل در موقعیت /boot/vmlinuz-2.6.32-754.el6.x86_64 قرار دارد. پس از موقعیت کرنل، تنظیمات دیگری نیز وجود دارند که ما با آنها کاری نداریم.

در نهایت، موقعیت initrd سیستم مشخص شده است. همانطور که می‌بینید، initrd در فایلی به نام initramfs-2.6.32-754.el6.x86_64.img در دایرکتوری /boot قرار دارد. موقعیت این فایل نیز دقیقا مانند موقعیت کرنل، به صورت relative نسبت به /boot مشخص می‌شود:

```
initrd /initramfs-2.6.32-754.el6.x86_64.img
```

نکته‌ی قابل توجه این است که این سیستم، از initramfs استفاده می‌کند، اما باز هم آن را با متغیر initrd برای GRUB Legacy تعریف کرده‌ایم.

اگر یک سیستم‌عامل ویندوزی داشته باشیم، کافی است پس از مشخص کردن نام آن در منوی بوت‌لودر، موقعیت هارددیسک و پارتیشن قرارگیری ویندوز را با استفاده از متغیر rootnoverify مشخص کنیم.

یعنی مثلا:

```
title Windows
rootnoverify (hd1,0)
chainloader (hd1,0) +1
```

همانطور که می‌بینید، در اینجا ما نام سیستم‌عامل در منوی بوت‌لودر را Windows قرار دادیم و سپس به GRUB Legacy گفتیم که ویندوز بر روی اولین پارتیشن موجود در دومین هارددیسک قرار گرفته است.

از آنجایی که GRUB نمی‌تواند سیستم عامل ویندوز را به صورت مستقیم راه‌اندازی کند، از متغیر chainloader نیز استفاده کرده‌ایم. متغیر chainloader، به GRUB می‌گوید که هنگام انتخاب سیستم ویندوز از منو، فرآیند راه‌اندازی سیستم را به بوت‌لودر ویندوز بسپارد؛ چرا که سیستم‌عامل ویندوز از بوت‌لودر مخصوص به خود استفاده می‌کند. بدین ترتیب ما از طریق بوت‌لودر GRUB Legacy، یک بوت‌لودر دیگر را اجرا می‌کنیم.

نصب‌کردن GRUB Legacy

همانطور که گفتیم، بوت‌لودر باید روی سکتور اول هارددیسک قرار گیرد (MBR)، اما ما چگونه می‌توانیم بوت‌لودر را روی سکتور اول یک هارددیسک قرار دهیم؟ ما می‌توانیم با استفاده از دستور grub-install، بوت‌لودر GRUB Legacy را روی یک سیستم نصب کنیم (البته پس از نصب برنامه‌ی GRUB با استفاده از yum یا هر روش دیگر). برای استفاده از دستور grub-install، کافی است دیوایس‌فایل هارددیسکی که می‌خواهیم GRUB Legacy روی MBR آن قرار گیرد را مشخص کنیم. برای مثال:

```
[root@localhost ~]# grub-install /dev/sda
```

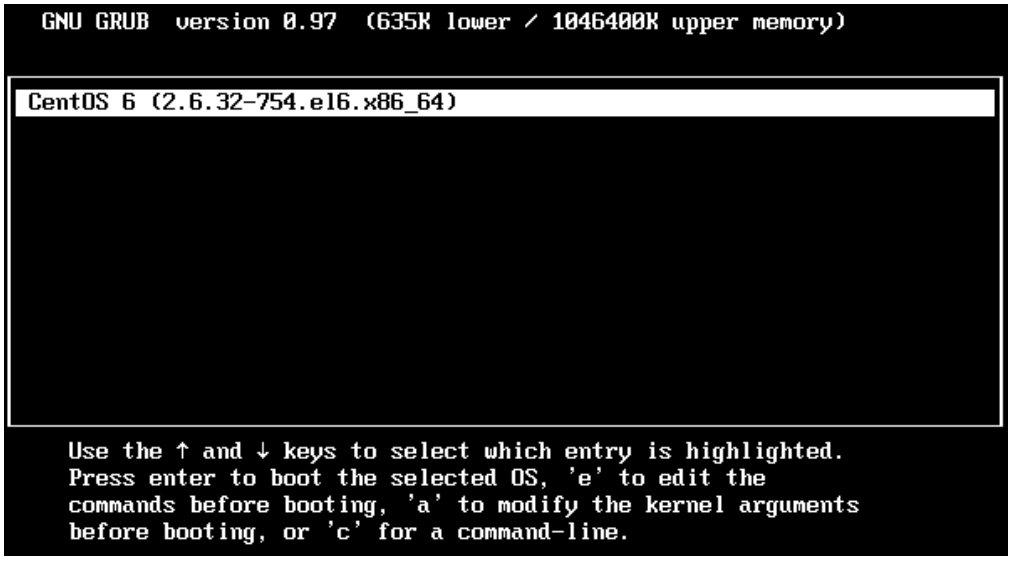
توجه کنید که در اینجا، ما به جای ارائه‌ی دیوایس‌فایل یک پارتیشن (مثلا sda1)، دیوایس‌فایل هاردیسک مورد نظر (sda) را به grub-install دادیم.

البته ما می‌توانیم به جای نصب GRUB Legacy بر روی MBR یک هارددیسک، آن را را بر روی اولین سکتور یک پارتیشن دیگر نیز نصب کنیم. در این حالت، ما از بوت‌لودر نصب شده در آن سکتور، به صورت chainloading استفاده می‌کنیم. برای این کار:

```
[root@localhost ~]# grub-install /dev/sda2
```

تعامل با منوی GRUB Legacy

بیایید بار دیگر به منوی بوت‌لودر GRUB Legacy نگاهی بیاندازیم:



تصویر ۳- منوی GRUB Legacy

همانطور که در تصویر ۳ می‌بینید:

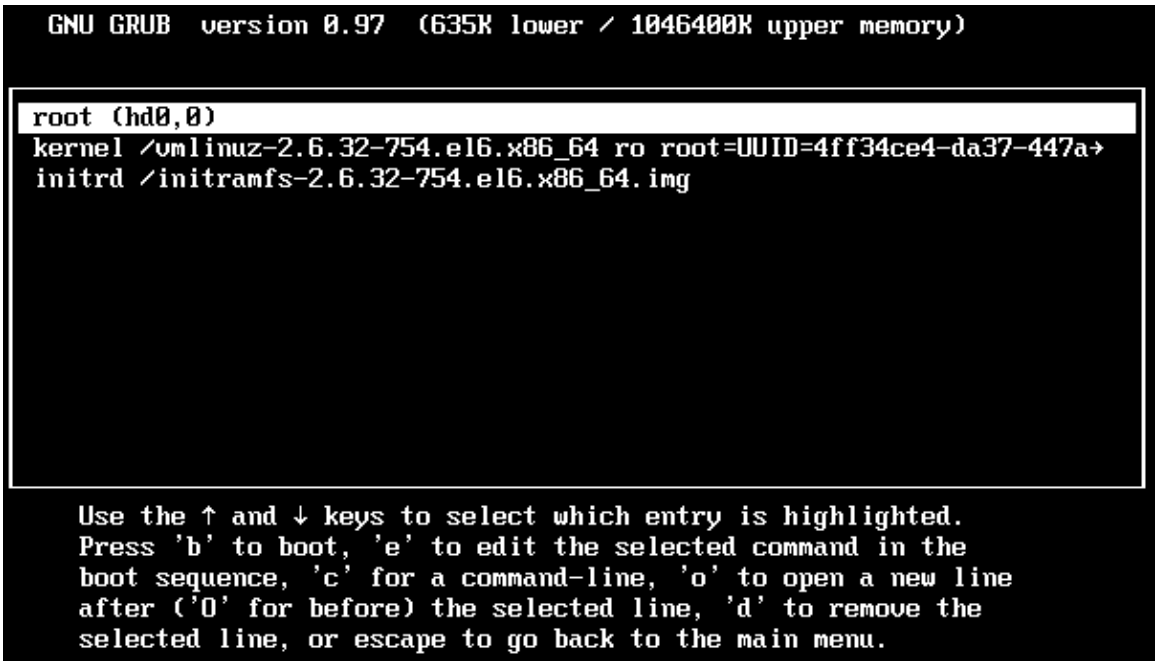
- با زدن دکمه‌ی Enter، می‌توانیم گزینه‌ی (سیستم‌عامل) مشخص شده راه‌اندازی کنیم.

- با زدن دکمه‌ی E، می‌توانیم برخی از متغیرهای مربوط به گزینه‌ی انتخاب شده را تغییر دهیم. این متغیرها شامل `root`، `kernel` و `initrd` می‌باشند. ما در این منو می‌توانیم سیستم‌عامل را با یک کرنل دیگر (مثلا نسخه‌ی جدیدتر یا قدیمی‌تر) اجرا کنیم.
- با زدن دکمه‌ی A، می‌توانیم آپشن‌های کرنل را تغییر داده یا آپشن جدید به آن اضافه کنیم. لیستی از آپشن‌های موجود برای کرنل را می‌توانید در اینجا مشاهده کنید.
- با زدن دکمه‌ی C یک کامندلاین محدود، که به آن GRUB Command-Line می‌گویند به ما نشان داده می‌شود.
- اگر برای بوت‌لودر رمزی قرار داده باشیم، با زدن دکمه‌ی P، می‌توانیم رمز را وارد کنیم (بعدا در مورد قرار دادن رمز روی منوی GRUB صحبت می‌کنیم).

نکته: اعمال هرگونه تغییر در تنظیمات کرنل (و بوت) از طریق منوی GRUB، فقط روی بوت بعدی تاثیر می‌گذارد. یعنی اگر ما تنظیمات کرنل یک سیستم عامل را از طریق منوی بوت‌لودر تغییر دهیم و سپس سیستم را بوت کنیم، سیستم با تنظیمات جدید بوت می‌شود. اما به محض خاموش کردن سیستم و روشن کردن دوباره‌ی آن، سیستم با تنظیمات همیشگی (تنظیمات موجود در فایل تنظیمات GRUB) خود بوت می‌شود.

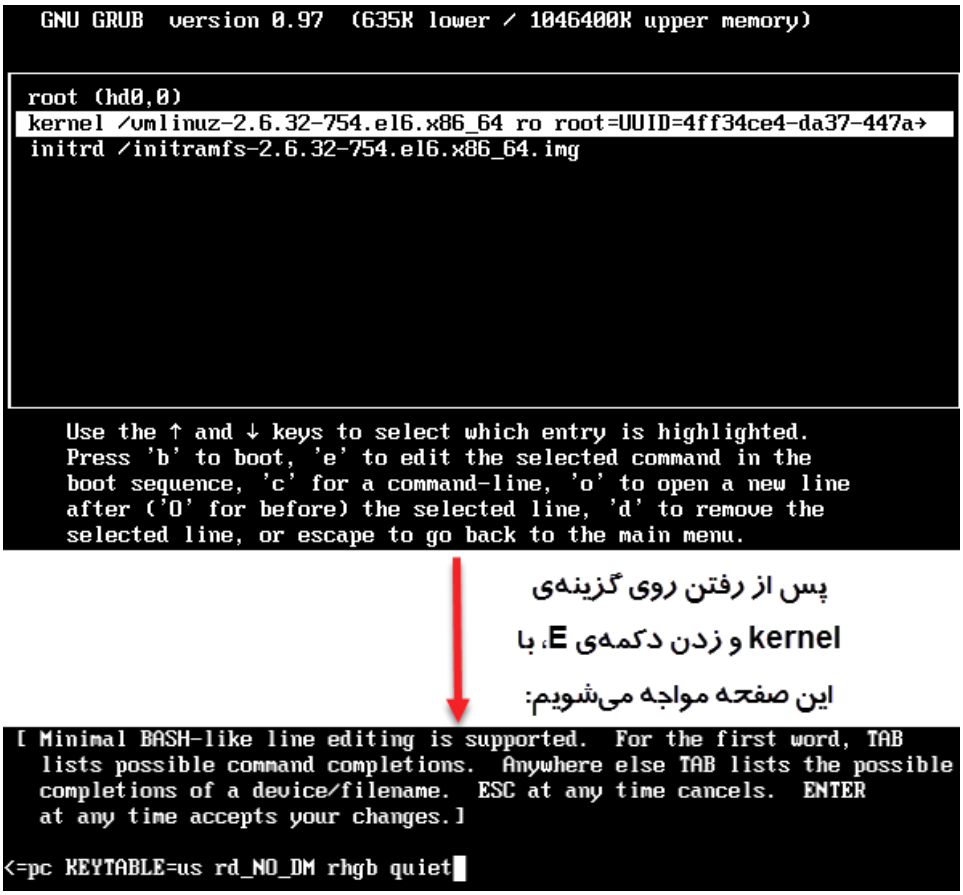
Password Recovery با GRUB Legacy

در حال حاضر، می‌خواهیم در مورد مفهومی به نام Password Recovery صحبت کنیم. گاهی اوقات ممکن است که ما رمز `root` سیستم را فراموش کنیم. ما میتوانیم با استفاده از قابلیت Password Recovery (یا به صورت فنی‌تر، بوت کردن سیستم در Single User Mode) رمز `root` را تغییر دهیم. برای این کار، باید هنگام مشاهده‌ی منوی بوت‌لودر، روی سیستم‌عامل مورد نظر رفته و دکمه‌ی E را فشار دهیم. پس از انجام این کار، با صفحه‌ی نظیر زیر تصویر ٤ مواجه می‌شویم:



تصویر ٤- صفحه‌ی تغییر متغیرهای بوت سیستم‌عامل

در این صفحه، ما با زدن دکمه‌ی ↓ کیبورد، روی گزینه‌ی kernel می‌رویم و سپس دکمه‌ی E را می‌زنیم. پس از انجام این کار، با صفحه‌ای نظیر تصویر ۵ مواجه می‌شویم. ما در این صفحه، می‌توانیم به کرنل سیستم‌عامل، یک سری آپشن اضافه کنیم.



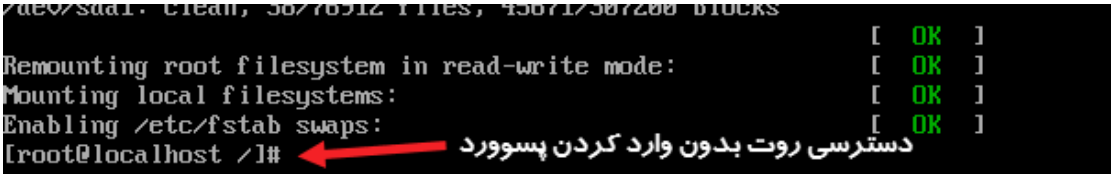
تصویر ۵- رفتن به صفحه‌ی تغییر آپشن‌های کرنل

در صفحه‌ی تغییر آپشن‌های کرنل، کافی است عبارت single را به انتهای این خط، اضافه کنیم (البته ممکن است GRUB به صورت اتوماتیک انتهای این خط را به شما نشان دهد). مطابق زیر:

تصویر ۶- اضافه کردن عبارت single به انتهای خط آپشن‌های کرنل

توجه کنید که به جای عبارت single، می‌توانیم از عدد 1 نیز استفاده کنیم. قرار دادن این عبارت در انتهای خط آپشن‌های کرنل، باعث می‌شود که سیستم در حالت Single User Mode بوت شود. حال که این عبارت را به آپشن‌های کرنل اضافه کردیم، کافی است دکمه‌ی Enter را فشار دهیم تا تغییراتمان ذخیره شوند. پس از ذخیره کردن تغییرات، دوباره به صفحه‌ی موجود در تصویر ٤ باز می‌گردیم. در اینجا کافی است دکمه‌ی B را فشار دهیم تا سیستم با آپشن‌های جدید بوت شود.

پس از بوت شدن، با صفحه‌ای نظیر زیر مواجه می‌شویم:



تصویر ۷- دسترسی روت بدون وارد کردن رمز

حال ما در حالت Single User Mode هستیم و دسترسی root به سیستم داریم. ما می‌توانیم با وارد کردن دستور passwd، فشردن دکمه‌ی Enter و سپس وارد کردن رمز جدیدی که می‌خواهیم برای کاربر root مشخص کنیم، رمز روت را تغییر دهیم. ما در جلسات بعد در مورد این دستور صحبت می‌کنیم.

نکته: روش عنوان شده برای Password Recovery، فقط در سیستم‌عامل‌هایی که از سیستم راه‌انداز SysV استفاده می‌کنند، ممکن می‌باشد و انجام Password Recovery در سیستم‌عامل‌هایی که از systemd استفاده می‌کنند، کاملا متفاوت و کمی دشوار تر می‌باشد. ما جلوتر در مورد سیستم‌های راه‌اندازی صحبت خواهیم کرد.

قرار دادن رمز روی منوی تنظیمات بوت‌لودر GRUB Legacy

اگر به حرف‌هایی که در بخش قبل زدیم دقت کرده باشید، خواهید دید که هر کس، فارغ از داشتن رمز root سیستم، می‌تواند هنگام راه‌اندازی تنظیمات بوت‌لودر را تغییر دهد و کرنل را با هر آپشنی راه‌اندازی کند. علاوه بر این، دسترسی به تنظیمات بوت‌لودر به کاربر این امکان را می‌دهد که بتواند رمز کاربر root را تغییر دهد (password recovery). طبیعتا این یک ضعف امنیتی به حساب می‌آید و ما باید جلوی آن را بگیریم. راه جلوگیری از این مشکل، قرار دادن رمز بر روی بوت‌لودر می‌باشد.

قرار دادن رمز روی بوت‌لودر GRUB Legacy بسیار ساده می‌باشد. از آنجایی که فایل تنظیمات GRUB Legacy به صورت Plain Text می‌باشد، ما باید رمز مورد نظر را خود را Hash کرده و سپس آن را درون فایل تنظیمات GRUB Legacy وارد کنیم.

فرض کنید ما می‌خواهیم روی بوت‌لودر، رمز IHateMyLife را قرار دهیم. همانطور که گفتیم، قرار دادن این رمز در فایل تنظیمات GRUB به صورت Plain Text کار اشتباهی است و در نتیجه باید رمز مورد نظر را Hash کنیم. برای Hash کردن این رمز، از دستور grub-md5-crypt استفاده می‌کنیم:

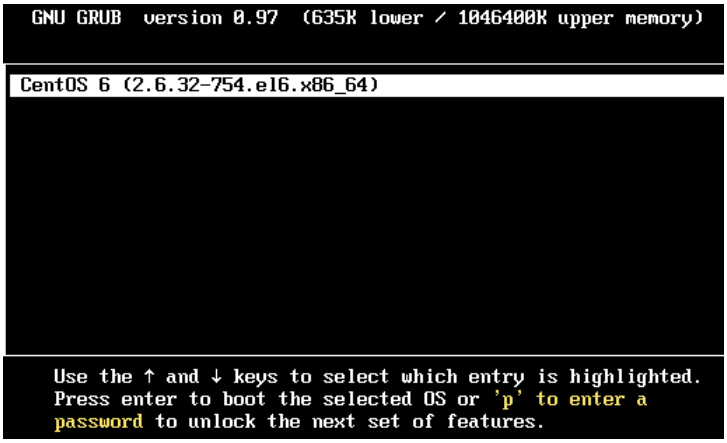
```
[root@localhost ~]# grub-md5-crypt
Password:
Retype password:
$1$N8Xf51$.qi18Q6JqZwUutuXcgWSX.
```

همانطور که می‌بینید، ما ابتدا دستور grub-md5-crypt را وارد و دکمه‌ی Enter را زدیم. سپس این دستور از ما خواست که رمزی که می‌خواهیم آن را Hash کنیم را وارد و سپس آن را بار دیگر جهت تایید وارد کنیم. پس از وارد کردن رمز دلخواه، رمز Hash شده در خط آخر به ما داده شد.

حال می‌خواهیم این رمز را روی بوت‌لودر، قرار دهیم. برای این کار، رمز Hash شده را کپی کرده، فایل /boot/grub/grub.conf را باز کرده و پس از تعاریف گلوبال، با استفاده از متغیر password --md5، رمز بوت‌لودر را مشخص می‌کنیم:

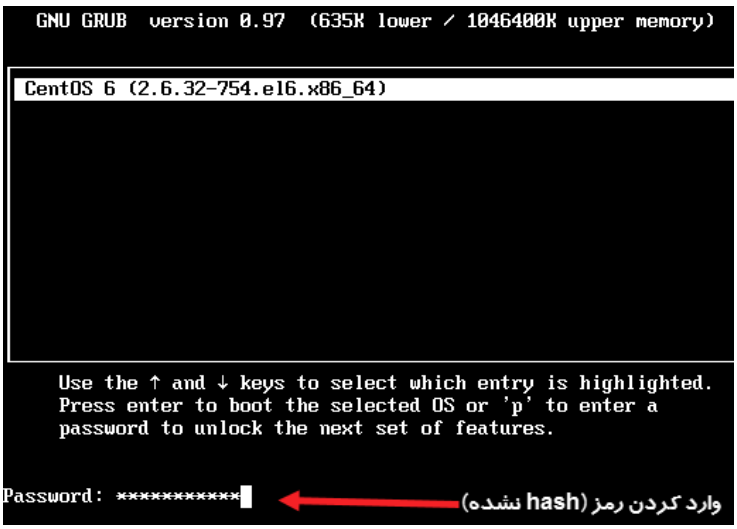
```
[root@localhost ~]# vi /boot/grub/grub.conf
...
hiddenmenu
password --md5 $1$N8Xf51$.qi18Q6JqZwUutuXcgWSX.
...
```

حال کافی است فایل را ذخیره کنیم تا تنظیمات جدید اعمال شوند. بیایید سیستم را Restart کرده و سعی به تغییر تنظیمات بوت‌لودر CentOS6 از منوی GRUB کنیم:



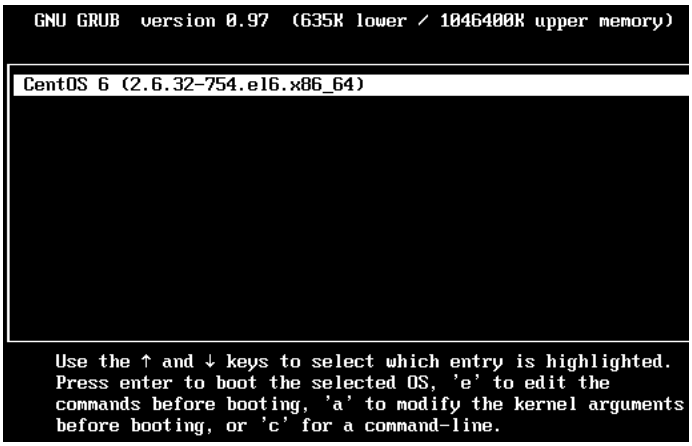
تصویر ۸- منوی GRUB Legacy پس از فعال کردن رمز

اگر تصویر ۱۰ را با تصویر ۳ مقایسه کنید، می‌بینید که در اینجا، نمیتوانیم با زدن دکمه‌ی E، وارد تنظیمات بوت‌لودر سیستم عامل شویم؛ بلکه ابتدا باید دکمه‌ی P را فشار داده و سپس رمزبوت‌لودر را وارد کنیم:



تصویر ۹- فشردن دکمه‌ی P و وارد کردن رمز بوت‌لودر

پس از وارد کردن رمز، منوی بوت‌لودر بار دیگر لود شده و این بار می‌توانیم با زدن دکمه‌ی E، تنظیمات را تغییر دهیم:



تصویر ۱۰- منوی GRUB Legacy پس از وارد کردن رمز بوت‌لودر

GRUB2

همانطور که گفتیم، GRUB2 نسخه‌ی بهبود یافته‌ی GRUB Legacy می‌باشد و به همین دلیل، بسیاری از مفاهیم اولیه آنها شبیه به هم می‌باشد؛ اما این دو بوت‌لودر تفاوت‌هایی نیز با هم دارند. برای مثال، در GRUB2، ما می‌توانیم برای هر سیستم‌عامل، یک سری ماژول خاص درون کرنل لود کنیم. علاوه بر این، GRUB2 از عبارت‌های شرطی نیز پشتیبانی می‌کند که به ما امکان می‌دهد تا در صورت وجود یک شرط خاص، یک سری ماژول را درون کرنل لود کنیم یا حتی یک سری آیتم را درون منوی GRUB2 نشان دهیم.

تنظیم GRUB2

نام فایل تنظیمات GRUB 2، به grub.cfg تغییر یافته است، اما موقعیت ذخیره‌سازی آن بستگی به Firmware سیستم ما دارد. یعنی:

- اگر سیستم ما از BIOS استفاده کند، فایل grub.cfg یا در دایرکتوری /boot/grub یا /boot/grub2 قرار گرفته است.
- اگر سیستم ما از UEFI استفاده کند، موقعیت فایل grub.cfg در دایرکتوری به نام /boot/efi/EFI/distro-name می‌باشد؛ به طوری که distro-name، نشان‌دهنده‌ی نام توزیعی می‌باشد که از آن استفاده می‌کنیم.

مثلا در CentOS 7 با سیستم BIOS، فایل‌های تنظیمات GRUB2 در دایرکتوری زیر قرار می‌گیرد:

```
[root@localhost ~]# ls -l /boot/grub2
total 32
-rw-r--r--. 1 root root   64 Mar 20 10:47 device.map
drwxr-xr-x. 2 root root   25 Mar 20 10:47 fonts
-rw-r--r--. 1 root root 4291 Mar 20 10:48 grub.cfg
-rw-r--r--. 1 root root 1024 Mar 20 10:48 grubenv
drwxr-xr-x. 2 root root 8192 Mar 20 10:47 i386-pc
drwxr-xr-x. 2 root root 4096 Mar 20 10:47 locale
```

بیایید نگاهی به محتویات فایل grub.cfg بیاندازیم:

```
[root@localhost ~]# less /boot/grub2/grub.cfg
```

این فایل پر از عبارت‌های شرطی می‌باشد و شاید درک آن در اولین نگاه کمی دشوار باشد. اما اگر به سراغ خط‌هایی که با عبارت menuentry شروع می‌شوند برویم، با الگوهایی شبیه به الگوهای موجود در فایل تنظیمات GRUB Legacy مواجه می‌شویم:

```
menuentry 'CentOS Linux (3.10.0-1062.el7.x86_64) 7 (Core)' --class centos --
class gnu-linux --class gnu --class os --unrestricted $menuentry_id_option
'gnulinux-3.10.0-1062.el7.x86_64-advanced-be85247b-b1f9-400f-9b1e-2772695dd003'
{
        load_video
        set gfxpayload=keep
        insmod gzio
        insmod part_msdos
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-
efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1'  81e90005-8ae2-
4b31-80f8-4938a33873b9
        else
          search --no-floppy --fs-uuid --set=root 81e90005-8ae2-4b31-80f8-
4938a33873b9
        fi
```

```
        linux16 /vmlinuz-3.10.0-1062.el7.x86_64 root=/dev/mapper/centos-root ro
crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet
LANG=en_US.UTF-8
        initrd16 /initramfs-3.10.0-1062.el7.x86_64.img
}
```

برای درک عبارات به کار برده شده برای تعریف سیستم‌عامل‌ها در GRUB2، کافی است موارد زیر را به خاطر بسپاریم:

- **menuentery**: اولین خط هنگام تعریف یک سیستم‌عامل جدید می‌باشد. این متغیر عملکردی شبیه به `title` در GRUB Legacy دارد. نامی که در مقابل این متغیر نوشته می‌شود، دقیقا در منوی بوت‌لودر قرار گرفته و به ما نشان داده می‌شود.
- **set root**: این متغیر هارددیسک و پارتیشنی که دایرکتوری /boot درون آن قرار دارد را مشخص می‌کند.

طریقه‌ی شماره‌گذاری پارتیشن‌ها در GRUB2 با GRUB Legacy متفاوت است. GRUB2 همچنان از عدد ۰ برای مشخص کردن اولین هارددیسک موجود در سیستم استفاده می‌کند، اما برای مشخص کردن اولین پارتیشن، از عدد ۱ استفاده می‌کند. یعنی اگر بخواهیم به GRUB2 بگوییم که دایرکتوری boot روی اولین پارتیشنِ اولین هارددیسک قرار دارد، باید به صورت زیر عمل کنیم:

```
set root=(hd0,1)
```

اما این تنها روش مشخص کردن هارددیسک و پارتیشن نیست. برخی از اوقات ممکن است به جای مشخص کردن شماره‌ی پارتیشن، از روش پارتیشن‌بندی به همراه شماره‌ی پارتیشن استفاده کنیم. برای مثال:

```
set root='hd0,msdos1'
```

همانطور که می‌بینید، ما در اینجا از `msdos` برای مشخص کردن پارتیشن‌بندی به روش DOS (همان روش MBR) استفاده کرده‌ایم. ما در مورد این روش‌ها و سیستم‌های متفاوت پارتیشن‌بندی در جلسه‌ی پنجم صحبت کردیم، پس در اینجا به توضیح آنها نمی‌پردازیم. اگر درک این بخش برایتان دشوار است، می‌توانید به فایل راهنمای اصلی GRUB2 در اینجا مراجعه کنید.

- **linux**، **linux16**: در سیستم‌هایی که از BIOS استفاده می‌کنند، این متغیر موقعیت کرنل را مشخص می‌کند. موقعیت کرنل، به صورت relative نسبت به `/boot` مشخص می‌شود.
- **linuxefi**: در سیستم‌هایی که از UEFI استفاده می‌کنند، این متغیر موقعیت کرنل را مشخص می‌کند. موقعیت کرنل، به صورت relative نسبت به `/boot` مشخص می‌شود.
- **initrd**: در سیستم‌هایی که از BIOS استفاده می‌کنند، این متغیر موقعیت Initial Ram Disk را مشخص می‌کند. `initrd` درایورهایی که کرنل برای ارتباط با سخت‌افزارهای سیستم نیاز دارد را درون خود دارد.
- **initrdefi**: در سیستم‌هایی که از UEFI استفاده می‌کنند، این متغیر موقعیت Initial Ram Disk را مشخص می‌کند. `initrd` درایورهایی که کرنل برای ارتباط با سخت‌افزارهای سیستم نیاز دارد را درون خود دارد.

نکته‌ی قابل توجه در GRUB2 این است که پس از مشخص کردن متغیر menuentery و آپشن‌های آن، سایر متغیرها را باید بین علامت {} قراردهیم. علاوه بر این، برای مشخص کردن سیستم‌عامل‌های غیرلینوکسی در GRUB2، دیگر از متغیر rootnoverify استفاده نمی‌کنیم. کلیه‌ی سیستم‌های غیرلینوکسی، دقیقا مانند سیستم‌های لینوکسی، با استفاده از دستور root تعریف می‌شوند.

با این که کلی در مورد فایل /boot/grub2/grub.cfg صحبت کردیم، ما هیچ‌وقت نباید تغییراتی را در این فایل به وجود آوریم، چرا که GRUB2 از یک سری اسکریپت و ابزار برای مدیریت اتوماتیک فایل grub.cfg استفاده می‌کند. برای اعمال تغییرات در تنظیمات GRUB2، باید فایل‌های موجود در دایرکتوری /etc/grub.d و همچنین فایل /etc/default/grub را تغییر دهیم. پس از اعمال تغییرات در این فایل‌ها، باید بار دیگر از سیستم بخواهیم که فایل grub.cfg را بازسازی کند. این دقیقا برعکس عملکرد GRUB Legacy می‌باشد که پس از اعمال تغییرات در فایل تنظیماتش، نیازی به بازسازی هیچکدام از فایل‌های آن وجود نداشت.

فایل‌ها و اسکریپت‌های موجود در /etc/grub.d، کاوشگرهای سیستم‌عامل‌ها می‌باشند؛ یعنی این اسکریپت‌ها به دنبال سیستم‌عامل‌ها و کرنل‌های موجود در سیستم می‌روند و به ازای هر سیستم‌عامل و کرنلی که پیدا می‌کنند، به صورت اتوماتیک تنظیمات مربوطه را درون فایل grub.cfg می‌ریزند. بیایید نگاهی به محتویات این دایرکتوری بیاندازیم:

```
[root@localhost ~]# ls -l /etc/grub.d/
total 72
-rwxr-xr-x. 1 root root  8702 Aug  8  2019 00_header
-rwxr-xr-x. 1 root root  1043 Mar 22  2019 00_tuned
-rwxr-xr-x. 1 root root   232 Aug  8  2019 01_users
-rwxr-xr-x. 1 root root 10781 Aug  8  2019 10_linux
-rwxr-xr-x. 1 root root 10275 Aug  8  2019 20_linux_xen
-rwxr-xr-x. 1 root root  2559 Aug  8  2019 20_ppc_terminfo
-rwxr-xr-x. 1 root root 11169 Aug  8  2019 30_os-prober
-rwxr-xr-x. 1 root root   214 Aug  8  2019 40_custom
-rwxr-xr-x. 1 root root   216 Aug  8  2019 41_custom
-rw-r--r--. 1 root root   483 Aug  8  2019 README
```

اگر یک سیستم‌عامل یا کرنل غیرمعمول داشته باشیم که به صورت اتوماتیک توسط GRUB2 تشخیص داده نشود، می‌توانیم اطلاعات مربوط به آن را در فایل 40_custom قرار دهیم. به عبارت دیگر، ما می‌توانیم از فایل 40_custom برای قرار دادن کرنل‌ها و سیستم‌عامل‌های دلخواه خود استفاده کنیم. نحوه‌ی وارد کردن اطلاعات مربوط به سیستم‌عامل یا کرنل جدید در این فایل، دقیقا همانطوری که در بالا توضیح دادیم می‌باشد؛ یعنی تعریف هر سیستم‌عامل با menuentry و آپشن‌های مربوط به آن شروع می‌شود، سپس سایر متغیرهای مربوط به تعریف سیستم‌عامل، بین دو علامت {} قرار می‌گیرند.

فایل /etc/default/grub، متغیرهای گلوبال مربوط به GRUB2 را درون خود دارد. بیایید نگاهی به محتویات این فایل بیاندازیم:

```
[root@localhost ~]# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb
quiet"
GRUB_DISABLE_RECOVERY="true"
```

همانطور که می‌بینید، متغیرهای گلوبال در GRUB2 با متغیرهای گلوبال GRUB تفاوت دارند. برای مثال، در GRUB2، متغیر GRUB_TIMEOUT مدت زمان نمایش منوی GRUB را مشخص می‌کند و پس از سپری شدن آن زمان، گزینه‌ی سیستم‌عامل پیش‌فرض را راه‌اندازی می‌کند و GRUB_DEFAULT سیستم‌عامل پیش‌فرضی که توسط بوت‌لودر راه‌اندازی می‌شود را مشخص می‌کند. مقدار saved برای این متغیر، یعنی بوت‌لودر سیستم‌عاملی که آخرین بار توسط کاربر انتخاب شده را در دفعه‌ی بعدی به عنوان سیستم‌عامل پیش‌فرض انتخاب می‌کند.

GRUB2 متغیرهای گلوبال بسیار زیادی دارد که توضیح همه‌ی آنها از حوصله‌ی ما خارج است. پیشنهاد می‌کنیم در صورت تمایل، به مطالعه‌ی راهنمای GRUB2 در این سایت بپردازید.

نصب GRUB2

گاهی اوقات، ممکن است مجبور شویم بوت‌لودر را به دلیل یک مشکل، از اول نصب کنیم، یا حتی بخواهیم بوت‌لودر را روی یک هارددیسک دیگر نصب کنیم. ما می‌توانیم با استفاده از دستور grub2-install، بوت‌لودر GRUB2 را بر روی MBR یک هارد دیسک نصب کنیم؛ اما قبل از آن، باید فایل تنظیمات GRUB2 را ایجاد کنیم (البته قبل از آن باید GRUB2 را با yum یا هر روش دیگری روی سیستم نصب کرده باشیم). ما این کار را با دستور grub2-mkconfig (یا در برخی توزیع‌ها، grub-mkconfig) انجام می‌دهیم. به طور کلی، این دستور فایل‌های موجود در دایرکتوری /etc/grub.d را می‌خواند و سپس به سراغ بازسازی فایل /boot/grub2/grub.cfg می‌رود. ما از این دستور، به صورت زیر استفاده می‌کنیم:

```
[root@localhost ~]# grub2-mkconfig > /boot/grub2/grub.cfg
```

توجه کنید که هنگام استفاده از این دستور، حتما باید خروجی آن را درون فایل grub.cfg، ریدایرکت کنیم. در غیر این صورت، grub2-mkconfig فایل بازسازی شده‌ی تنظیمات را فقط روی STDOUT میریزد و فایل اصلی تنظیمات تغییری پیدا نمی‌کند.

سپس برای نصب GRUB2 بر روی MBR هارددیسک، کافی است دیوایس‌فایل دستگاهی که می‌خواهیم GRUB2 روی MBR آن قرار گیرد را به دستور grub2-install بدهیم. یعنی مثلا:

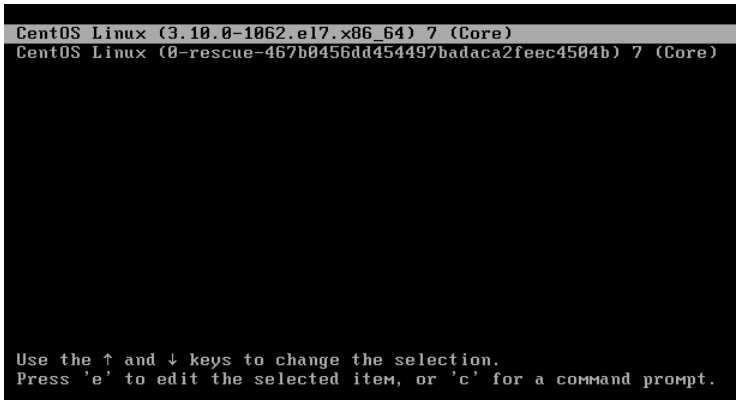
```
[root@localhost ~]# grub2-install /dev/sda
```

توجه کنید که در اینجا، ما به جای ارائه‌ی دیوایس‌فایل یک پارتیشن (مثلا sda1)، دیوایس‌فایل هاردیسک مورد نظر (sda) را به grub-install دادیم.

نکته‌ی قابل توجه دیگر این است که با ایجاد هر تغییر در فایل‌های تنظیمات GRUB2، باید فایل grub.cfg را از اول با دستور grub2-mkconfig بازسازی کنیم. این دقیقا برعکس GRUB Legacy می‌باشد که اعمال تغییرات در آن، نیازی به بازسازی فایل اصلی تنظیمات نداشت.

تعامل با منوی GRUB2

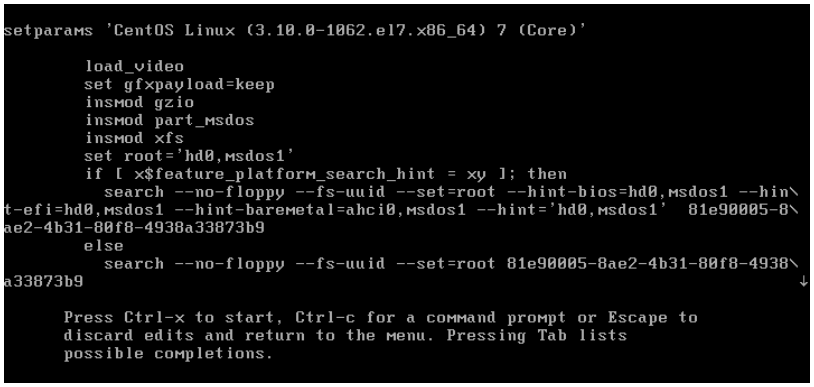
ما در بخش‌های قبلی، نگاهی به منوی بوت‌لودر GRUB Legacy انداختیم. همانطور که در تصویر ۸ می‌بینید، منوی GRUB و GRUB2 از نظر قیافه‌ای بسیار شبیه به هم می‌باشند.



تصویر ۱۱- منوی GRUB2

در این منو:

- با زدن دکمه‌ی Enter، گزینه‌ی مشخص شده راه‌اندازی می‌شود.
- با زدن دکمه‌ی C، یک کامندلاین محدود، که به آن GRUB Command-Line می‌گویند به ما نشان داده می‌شود.
- اگر برای بوت‌لودر رمزی قرار داده باشیم، با زدن دکمه‌ی P، می‌توانیم رمز را وارد کنیم.
- بازدن دکمه‌ی E می‌توانیم تغییراتی را در یکی از موارد مشخص شده در منوی GRUB2 به وجود آوریم. به محض زدن این دکمه، با صفحه‌ای نظیر تصویر ۹ مواجه می‌شویم:



تصویر ۱۲- اعمال تغییرات در یکی از گزینه‌های موجود در منوی GRUB2

با توجه به این که نمای این صفحه کمی متفاوت از محیط‌هایی که با آن آشنایی داریم می‌باشد، کمی در مورد چگونگی تعامل با آن صحبت می‌کنیم. ما می‌توانیم با فشردن دکمه‌های جهتی کیبورد در این صفحه، بین خطوط مختلف بالا و پایین رویم. با زدن دکمه‌ی Enter، می‌توانیم یک خط جدید به وجود آوریم، با زدن دکمه‌ی ESC، به منوی GRUB (تصویر ۸) باز می‌گردیم و کلیه‌ی تغییراتی که در بخش تنظیمات این گزینه به وجود آوردیم دور ریخته می‌شود. با زدن دکمه‌ی Ctrl + C یک کامندلاین به ما نمایش داده می‌شود و با زدن دکمه‌ی Ctrl و X، سیستم با تنظیمات اعمال شده‌ی جدید راه‌اندازی می‌شود.

ما در این صفحه، علاوه بر امکان تغییر تنظیمات بوت‌لودر، می‌توانیم آپشن‌هایی متفاوتی را به خود کرنل لینوکس، اعمال کنیم. این کار می‌تواند ما را در پیدا کردن یا دور زدن برخی مشکلات بوت، یاری دهد. برای اعمال آپشن‌های متفاوت به کرنل، پس از انتخاب سیستم‌عامل مورد نظر و زدن دکمه‌ی E (مواجه شدن با تصویر ۹)، خطی که با کلمه‌ی `linux` شروع می‌شود را پیدا می‌کنیم و

به انتهای آن خط، پس از قرار دادن یک فاصله‌ی خالی، آپشن‌های مورد نظر را اضافه می‌کنیم. برخی از کاربردی‌ترین آپشن‌ها به شرح زیر می‌باشند:

جدول ۳- آپشن‌های کرنل

آپشن	معنی
`initrd=`	موقعیت RAM disk اولیه‌ی سیستم را تغییر می‌دهد.
`mem`	میزان کل RAM اختصاص یافته به سیستم را مشخص می‌کند.
`ro`	پارتیشن root را در حالت read-only مانت می‌کند.
`rw`	پارتیشن root را در حالت read-write مانت می‌کند.
`root=`	موقعیت پارتیشن root را تغییر می‌دهد.
`selinux`	SELinux را هنگام راه‌اندازی سیستم، غیر فعال می‌کند.

نکته: اعمال هرگونه تغییر در تنظیمات کرنل (و بوت) از طریق منوی GRUB، فقط روی بوت بعدی تاثیر می‌گذارد. یعنی اگر ما تنظیمات کرنل یک سیستم عامل را از طریق منوی بوت‌لودر تغییر دهیم و سپس سیستم را بوت کنیم، سیستم با تنظیمات جدید بوت می‌شود. اما به محض خاموش کردن سیستم و روشن کردن دوباره‌ی آن، سیستم با تنظیمات همیشگی (تنظیمات موجود در فایل تنظیمات GRUB) خود بوت می‌شود.

Password Recovery با GRUB2

انجام عمل Password Recovery در GRUB2 نیز بسیار شبیه به GRUB Legacy می‌باشد. برای این کار کافی است در منوی بوت‌لودر (تصویر ۸)، روی سیستم‌عامل مورد نظر رفته و دکمه‌ی E را بزنیم. در صفحه‌ی ظاهر شده (تصویر ۹)، به انتهای خطی که با کلمه‌ی `linux` آغاز شده، عبارت `single` را اضافه می‌کنیم و در نهایت دکمه‌ی Ctrl و X را جهت بوت سیستم با این تنظیمات فشار می‌دهیم. ما انجام و تست این کار را به خودتان می‌سپاریم.

نکته: روش عنوان شده برای Password Recovery، فقط در سیستم‌عامل‌هایی که از سیستم راه‌انداز SysV استفاده می‌کنند، ممکن می‌باشد و انجام Password Recovery در سیستم‌عامل‌هایی که از systemd استفاده می‌کنند، کاملا متفاوت و کمی دشوار تر می‌باشد. سیستم‌عامل CentOS 7 از systemd استفاده می‌کند و در نتیجه روش ذکر شده برای Password Recovery به درد CentOS 7 نمیخورد. ما جلوتر در مورد سیستم‌های راه‌اندازی صحبت خواهیم کرد.

قرار دادن رمز روی منوی تنظیمات بوت‌لودر GRUB2

اگر به حرف‌هایی که در بخش‌های قبل زدیم دقت کرده باشید، خواهید دید که هرکس و ناکسی می‌تواند با زدن دکمه‌ی E هنگام مشاهده‌ی منوی بوت‌لودر، یک سیستم‌عامل را با تنظیمات و ماژول‌های مورد نظر خود، بوت کند؛ یا حتی رمز روت سیستم را تغییر دهد. روش جلوگیری از پیش‌آمد چنین اتفاقی، قرار دادن رمز بر روی بوت‌لودر می‌باشد. رمزگذاری روی بوت‌لودر GRUB2، کمی دشوارتر از رمزگذاری روی GRUB Legacy می‌باشد. در ضمن در GRUB2، علاوه بر رمز، باید یوزرنیم نیز تعریف کنیم.

از آنجایی که رمز ما باید به صورت hash شده در فایل تنظیمات قرار گیرد، بیایید ابتدا بیایید رمز hash شده‌ی خود را ایجاد کنیم. برای انجام این کار، باید از دستور `grub2-mkpasswd-pbkdf2` استفاده کنیم:

```
[root@localhost ~]# grub2-mkpasswd-pbkdf2
Enter password:
Reenter password:
PBKDF2 hash of your password is
grub.pbkdf2.sha512.10000.0CB82B44CE1E11BA61CE3274E67341526380C7E346CA93F713D6D5D
08CA870122A3715256DD73DD8342BAF241B9DBBC68F3B18DCF41F167DF0CD987CB549F310.4C9FDB
51A06A7D0BA75F21072CF527ACC26CE266496372EF15BCD49DB7BD6954178BDD9FB4DB322EAEF209
060ABE6273A312EFF1B1886C8D81CC6E8E6CC7A6FF
```

همانطور که می‌بینید، ما ابتدا دستور `grub2-mkpasswd-pbkdf2` را وارد و دکمه‌ی Enter را زدیم. سپس رمز دلخواه خود را وارد کردیم و در خروجی، Hash رمز دلخواه را دریافت کردیم.

حال باید تغییراتی در فایل `/etc/grub.d/40_custom` به وجود آوریم. پس این فایل را با یک ادیتور نظیر `vi` باز می‌کنیم:

```
[root@localhost ~]# vi /etc/grub.d/40_custom
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
```

و پس از آخرین خط موجود در این فایل، موارد زیر را وارد می‌کنیم:

```
set superusers="root"
export superusers
```

با استفاده از این دو خط، ما یوزرنیم مورد نیاز جهت وارد شدن به تنظیمات GRUB2 در منوی بوت‌لودر را مشخص می‌کنیم. توجه کنید که در اینجا می‌توانیم هر نامی به `username` بدهیم و مجبور نیستیم از `root` استفاده کنیم. ما حتی می‌توانیم بیش از یک کاربر معرفی کنیم.

پس از نوشتن این دو خط، رمز Hash شده‌ی خود را درون این فایل قرار دهیم. برای این کار:

```
password_pbkdf2 root grub.pbkdf2.sha512.10000.32992CC73229D8A656A25655F8ACC42C24
49D970BDF99CCAEB85B572439CB4A6157B471FC50FFCA1B43FE791B5010C37B73EE084A3D99C0E85
6B1E.22AAE4769596259FC89BE7CECA2858458925A3EFC5D10E48C73934E9028BFB71AB073B14A3B
50FD436DE391014B88D73409981DD92632F250C9210EAB16358A4
```

همانطور که می‌بینید، برای تعریف رمز، ابتدا متغیر `password_pbkdf2` را وارد این فایل می‌کنیم، سپس نام کاربری که برایش این رمز را انتخاب کرده‌ایم را مشخص می‌کنیم. در اینجا ما یک کاربر تعریف کرده‌ایم که `root` می‌باشد، پس می‌نویسیم `root` و پس از آن، رمز Hash شده‌ی خود را وارد می‌کنیم.

پس الان، فایل `/etc/grub.d/40_custom` نمایی نظیر زیر خواهد داشت:

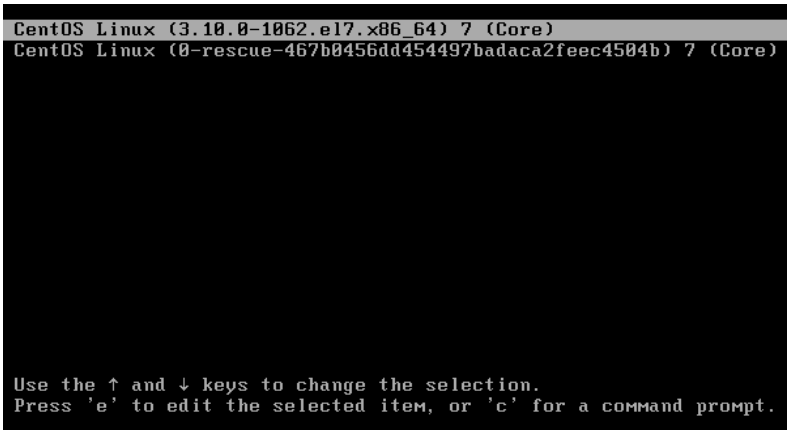
```
[root@localhost ~]# vi /etc/grub.d/40_custom
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
set superusers="root"
export superusers
password_pbkdf2 root grub.pbkdf2.sha512.10000.32992CC73229D8A656A25655F8ACC42C24
49D970BDF99CCAEB85B572439CB4A6157B471FC50FFCA1B43FE791B5010C37B73EE084A3D99C0E85
6B1E.22AAE4769596259FC89BE7CECA2858458925A3EFC5D10E48C73934E9028BFB71AB073B14A3B
50FD436DE391014B88D73409981DD92632F250C9210EAB16358A4
```

حال برای این که این تنظیمات جدید بر روی GRUB2 اعمال شود، باید دستور `grub2-mkconfig` را اجرا کنیم (به استفاده از redirector برای نوشتن تغییرات روی فایل `grub.cfg` توجه کنید):

```
[root@localhost ~]# grub2-mkconfig > /boot/grub2/grub.cfg
Generating grub configuration file ...
```

```
Found linux image: /boot/vmlinuz-3.10.0-1062.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-1062.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-467b0456dd454497badaca2feec4504b
Found initrd image: /boot/initramfs-0-rescue-467b0456dd454497badaca2feec4504.img
done
```

حال بیایید سیستم را ریبوت کرده و سعی در تغییر تنظیمات GRUB2 کنیم. اگر به تصویر ۱۳ نگاه کنید، می‌بینید که در اولین نگاه، منوی بوت‌لودر هیچ تغییری با قبل نکرده است.

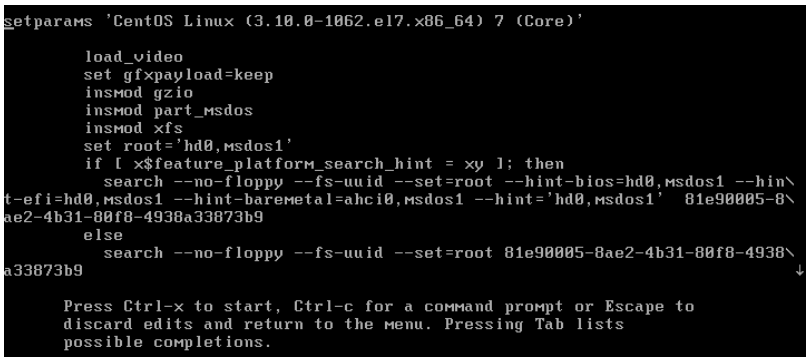


تصویر ۱۳- منوی بوت‌لودر پس از اعمال رمز

اما اگر دکمه‌ی E را جهت اعمال تغییرات بوت‌لودر فشار دهیم، با نمایی نظیر تصویر ۱٤ مواجه می‌شویم:

تصویر ۱٤- منوی نمایش داده شده پس از سعی در تغییر تنظیمات و وارد کردن یوزرنیم و پسوورد

پس از وارد کردن یوزرنیم و پسوورد، به صفحه‌ی تغییر تنظیمات برده می‌شویم:



تصویر ۱۵- صفحه‌ی تغییر تنظیمات پس از وارد کردن رمز

فرآیند راه‌اندازی یا Initialization

پس از این که بوت‌لودر کرنل مربوط به سیستم‌عامل مورد نظر ما را اجرا کرد، ما وارد مرحله‌ی نهایی بوت می‌شویم. در این مرحله، سیستم باید سرویس‌های متفاوت مورد نیاز ما را راه‌اندازی کند. سرویس یا daemon، برنامه‌ای می‌باشد که یک کار خاص را انجام می‌دهد.

سیستم راه‌انداز، تصمیم می‌گیرد که چه سرویس‌هایی باید با چه ترتیبی هنگام روشن شدن سیستم، اجرا شوند و همچنین به ما اجازه می‌دهد که سرویس‌های یک سیستم را مدیریت کنیم. به طور کلی، دو سیستم راه‌اندازی معروف در دنیای لینوکس وجود دارد:

- SysV

در قدیم، SysV یا SysVinit که بر پایه‌ی سیستم راه‌اندازی Unix System V نوشته شده بود، به عنوان سیستم راه‌انداز اصلی بر روی اکثر توزیع‌های لینوکسی قرار داشت. امروزه اکثر توزیع‌های لینوکسی دیگر از این سرویس استفاده نمی‌کنند، اما این راه‌انداز هنوز هم برخی از سرورهای قدیمی‌تر موجود می‌باشد.

- systemd

سیستم راه‌انداز systemd در سال ۲۰۱۰ نوشته شد و کم‌کم تبدیل به معروف‌ترین سیستم راه‌انداز و مدیریت سرویس‌ها شد و امروزه اکثر توزیع‌ها از آن استفاده می‌کنند. systemd، زمان مورد نیاز برای راه‌اندازی را با راه‌اندازی سرویس‌ها به صورت موازی، کاهش می‌دهد.

تشخیص نوع سیستم راه‌انداز

قبل از این که در مورد هر کدام از این سیستم‌های راه‌اندازی صحبت کنیم، بد نیست که نگاهی به برنامه‌ی `init` بیاندازیم. به طور کلی، استارت زدن سرویس‌های موجود در سیستم توسط برنامه‌ی `init` انجام می‌شود. این برنامه می‌تواند در دایرکتوری `/etc`، `/bin` یا `/sbin` قرار گرفته باشد و همچنین معمولا Process ID (یا PID) برابر با ۱ را دارد.

برای این که ببینیم سیستم‌عامل ما از راه‌اندازی SysV یا systemd استفاده می‌کند، به صورت زیر عمل می‌کنیم.

ابتدا موقعیت برنامه‌ی `init` را با استفاده از دستور `which` پیدا می‌کنیم:

```
[root@localhost ~]# which init
/usr/sbin/init
```

حال که موقعیت برنامه‌ی `init` را می‌دانیم، می‌توانیم با استفاده از دستور `readlink -f` ببینیم که آیا دستور `init` به برنامه‌ی دیگری لینک شده است یا نه:

```
[root@localhost ~]# readlink -f /usr/sbin/init
/usr/lib/systemd/systemd
```

همانطور که می‌بینید، سیستم‌عامل ما که CentOS 7 می‌باشد، از سیستم systemd جهت راه‌اندازی سیستم استفاده می‌کند. ما می‌توانیم این امر را با نگاه کردن به پراسس دارای PID برابر با ۱ نیز تایید کنیم:

```
[root@localhost ~]# ps -p 1
  PID TTY          TIME CMD
    1 ?        00:00:07 systemd
```

همانطور که می‌بینید، پراسس `systemd`، دارای PID برابر با ۱ می‌باشد، پس می‌توانیم مطمئن باشیم که سیستم CentOS 7 ما از systemd جهت راه‌اندازی استفاده می‌کند.

بیایید همین دستور را روی CentOS 6 اجرا کنیم:

```
[root@localhost ~]# ps -p 1
  PID TTY          TIME CMD
    1 ?        00:00:04 init
```

همانطور که می‌بینید، پراسس دارای PID برابر با ۱ در CentOS 6، پراسس `init` می‌باشد. این یعنی که CentOS 6 از SysV جهت راه‌اندازی سیستم استفاده می‌کند.

نکته‌ی جالب در مورد systemd و همچنین SysV این است که این پراسس‌ها، والد کلیه‌ی سرویس‌های موجود بر روی سیستم می‌باشد. ما می‌توانیم این امر را با استفاده از دستور `pstree` ببینیم:

```
[root@localhost ~]# pstree -p 1
systemd(1)─┬─NetworkManager(931)─┬─{NetworkManager}(941)
           │                     └─{NetworkManager}(946)
           ├─VGAuthService(863)
           ├─agetty(756)
           ├─auditd(680)───{auditd}(681)
           ├─crond(753)
           ├─dbus-daemon(710)───{dbus-daemon}(749)
           ├─firewalld(763)───{firewalld}(1077)
           ├─irqbalance(705)
           ├─lvmetad(550)
           ├─master(1397)─┬─pickup(1408)
           │              └─qmgr(1409)
           ├─polkitd(708)─┬─{polkitd}(754)
           │              ├─{polkitd}(755)
           │              ├─{polkitd}(757)
           │              ├─{polkitd}(758)
           │              ├─{polkitd}(760)
           │              └─{polkitd}(761)
           ├─rsyslogd(1255)─┬─{rsyslogd}(1263)
           │                └─{rsyslogd}(1264)
           ├─sshd(1253)───sshd(1283)───bash(1495)───pstree(1564)
           ├─systemd-journal(524)
           ├─systemd-logind(750)
           ├─systemd-udevd(559)
           ├─tuned(1254)─┬─{tuned}(1520)
           │             ├─{tuned}(1521)
           │             ├─{tuned}(1528)
           │             └─{tuned}(1532)
           └─vmtoolsd(834)───{vmtoolsd}(899)
```

همانطور که می‌بینید، با استفاده از دستور `pstree`، ارائه‌ی آپشن `p-` و وارد کردن PID مورد نظر، دیدیم که `systemd`، والد کلیه‌ی سرویس‌های موجود بر روی سیستم می‌باشد. این امر برای SysV نیز صدق می‌کند، اما ما تصدیق این حرف با استفاده از دستور `pstree` را به خودتان می‌سپاریم.

استفاده از سیستم راه‌انداز SysV

با این که امروزه بسیاری از توزیع‌های جدید، دیگر از سیستم SysV استفاده نمی‌کنند، اما آشنایی با SysV بسیار مهم می‌باشد، چون در بسیاری از شرکت‌ها، هنوز سرورهای قدیمی‌تری وجود دارند که از SysV استفاده می‌کنند و در کل، آشنایی با SysV که معماری ساده‌تری دارد، ما را در درک چگونگی راه‌اندازی سیستم، یاری می‌دهد.

نکته: جهت درک بهتر موارد موجود در این بخش، باید از توزیعی که از سیستم راه‌انداز SysV استفاده می‌کند، استفاده کنیم. برای پیداکردن چنین توزیعی، می‌توانید به سایت DistroWatch (که قبلا در مورد آن صحبت کردیم) مراجعه کنید. ما از سیستم‌عامل CentOS 6 استفاده خواهیم کرد.

برای درک عملکرد SysV، باید ابتدا با runlevelها آشنا شویم. برنامه‌ی `init` با نگاه کردن به runlevel سیستم، تصمیم می‌گیرد که چه سرویس‌هایی را استارت بزند (یا متوقف کند). به طور کلی، runlevelها از ۰ تا ۶

شماره‌گذاری شده‌اند و به هر کدام از آنها، مجموعه‌ای از سرویس‌ها اختصاص داده شده است که باید در آن runlevel خاص، استارت زده شوند. آشنایی با runlevelها و درک مفهوم هر کدام، درک ما از فرآیند راه‌اندازی یک سیستم لینوکسی را کامل‌تر می‌کند. به طور کلی، ما باید هدف runlevelها، سرویس‌های فعال در یک runlevel و چگونگی تغییر و مدیریت آنها را به صورت کامل بلد باشیم.

runlevelها

به طور کلی، ما هفت runlevel داریم که از ۰ تا ۶ شماره‌گذاری می‌شوند. از این میان، runlevelهای صفر، یک و شش، به صورت رزرو شده برای انجام عملیات خاص می‌باشند و سایر runlevelها، در دسترس ما یا توسعه‌دهندگان یک توزیع برای انجام عملیات مورد نظر ما می‌باشند. در جدول ٤، هدف و عملکرد معمول همه‌ی runlevelها را مشاهده می‌کنید.

جدول ٤- runlevelها، هدف و عملکرد آنها

runlevel	عملکرد و هدف
0	سیستم را خاموش می‌کند. یعنی همه‌ی سرویس‌های روی سیستم را متوقف و سپس کلیه‌ی سخت‌افزارهای سیستم را نیز خاموش می‌کند.
1، s یا S	سیستم را در حالت تک-کاربره (Single User Mode) قرار می‌دهد (حالتی که فقط کاربر root می‌تواند وارد سیستم شود). این حالت، بسیار شبیه به حالت Safe Mode در ویندوز می‌باشد. در این حالت، حداقل‌ترین سرویس‌ها برای عملکرد صحیح سیستم استارت می‌شوند. این که دقیقا چه سرویس‌هایی در این حالت اجرا می‌شوند، با توجه به توزیع‌های متفاوت، دچار تغییر می‌شود. معمولا از این runlevel برای تعمیر و نگهداری سیستم استفاده می‌شود.
2	سیستم را در حالت چند-کاربره (Multi User Mode) قرار می‌دهد. در توزیع‌های Debian-based، این runlevel به کاربر یک رابط گرافیکی نیز می‌دهد. در سیستم‌های Red Hat-based سیستم را در حالت چند کاربره بدون ظاهر گرافیکی و بدون شبکه قرار می‌دهد.
3	در توزیع‌های Red Hat-based و برخی دیگر از توزیع‌ها، سیستم را در حالت چند-کاربره (Multi-User Mode) بدون ظاهر گرافیکی، اما با قابلیت‌های شبکه قرار می‌دهد.
4	این runlevel در اکثر سیستم‌ها، عملکرد خاصی ندارد و ما می‌توانیم آن را شخصی‌سازی کنیم.
5	در سیستم‌های Red Hat-based، سیستم را در حالت چند-کاربره با ظاهر گرافیکی قرار می‌دهد.
6	سیستم را ریبوت می‌کند.

نکته: در برخی از سیستم‌ها، ممکن است که runlevelهایی فراتر از ۶ نیز داشته باشیم؛ اما این امر بسیار نادر می‌باشد. در صورت مشاهده‌ی runlevelهای عجیب و فراتر از ۶، بهتر است به فایل `/etc/inittab` نگاهی بیاندازید، چون این فایل معمولا توضیحاتی در مورد runlevelها و عملکرد آنها را درون خود دارد.

تشخیص runlevel کنونی سیستم

برای این که ببینیم سیستم ما در حال حاضر در چه runlevel می‌باشد، از دستور `runlevel` استفاده می‌کنیم:

```
[root@localhost ~]# runlevel
N 3
```

در خروجی این دستور، کاراکتر یا عدد اول، نشان دهنده‌ی runlevel قبلی سیستم می‌باشد. در اینجا، N یعنی سیستم به تازگی بوت شده و runlevel قبلی وجود ندارد. عدد دوم، مشخص کننده‌ی runlevel کنونی ما می‌باشد. همانطور که می‌بینید، سیستم CentOS 6 ما، اکنون در runlevel شماره‌ی ۳ قرار دارد.

پیدا کردن و تغییر runlevel پیش‌فرض

برای این که بفهمیم runlevel پیش‌فرض یک سیستم چه شماره‌ای دارد، باید به فایل `/etc/inittab` نگاهی بیاندازیم. در قدیم، این فایل بسیار طولانی بود و استارت زدن همه‌ی سرویس‌ها بر عهده‌ی این فایل بود، اما با گذر زمان، این فایل محدود به مشخص کردن runlevel پیش‌فرض و استارت زدن برخی از سرویس‌های حیاتی شد. این فایل با توجه به نوع توزیع، هنوز هم ممکن است دارای تنظیمات و موارد زیادی باشد؛ مواردی که در حال حاضر با آنها کاری نداریم.

برای پیدا کردن runlevel پیش‌فرض، کافی است در فایل `/etc/inittab`، به دنبال خطی که درون خود عبارت `:initdeafult:` را دارد، بگردیم. برای این کار، از `grep` استفاده می‌کنیم:

```
[root@localhost ~]# grep ":initdefault:" /etc/inittab
id:3:initdefault:
```

عدد موجود بین `id:` و `:initdefault:`، مشخص کننده‌ی runlevel پیش‌فرض سیستم می‌باشد. همانطور که می‌بینید، در اینجا عدد ۳ بین این دو عبارت قرار گرفته است؛ پس runlevel پیش‌فرض ما، ۳ می‌باشد؛ یعنی سیستم به محض روشن شدن، وارد runlevel سوم می‌شود.

برای تغییر runlevel پیش‌فرض، کافی است فایل `/etc/inittab` را با یک ادیتور نظیر `vi` باز کرده و عدد موجود بین دو عبارت `id:` و `:initdefault:` را به runlevel دلخواه خود تغییر دهیم. نکته‌ای که باید به آن توجه کنیم این است که ما نباید هیچگاه runlevel پیش‌فرض را برابر با صفر یا شش قرار دهیم، چرا که در آن حالت، سیستم به محض روشن شدن خاموش یا ریبوت می‌شود (جدول ٤).

تغییر runlevel

خیلی از اوقات مکن است بخواهیم runlevel یک سیستم در حال اجرا را تغییر دهیم. برای تغییر runlevel، از دستورهای `init` (یا `telinit`)، `shutdown`، `halt`، `reboot` و `poweroff` استفاده می‌کنیم. ما در این قسمت در مورد دستور `init` (و `telinit`) و همچنین `shutdown` صحبت خواهیم کرد و بررسی عملکرد `halt`، `reboot` و `poweroff` را به خودتان می‌سپاریم.

تغییر runlevel با *init* یا *telinit*

برای تغییر runlevel با `init` یا `telinit`، کافی است دستور `init` و شماره‌ی runlevel که قصد داریم به آن برویم را وارد کنیم. بیایید ابتدا به runlevel کنونی سیستم نگاهی بیاندازیم:

```
[root@localhost ~]# runlevel
N 3
```

همانطور که می‌بینید سیستم اکنون در runlevel شماره‌ی ۳ قرار دارد. حال بیایید سیستم را به runlevel پنجم ببریم. برای رفتن به runlevel شماره‌ی ۵، به شکل زیر از `init` استفاده می‌کنیم:

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

بیایید بار دیگر runlevel کنونی سیستم را بررسی کنیم:

```
[root@localhost ~]# runlevel
3 5
```

همانطور که می‌بینید، runlevel سیستم به شماره‌ی ۵ تغییر یافت.

استفاده از `telinit` نیز دقیقا مانند `init` می‌باشد. بیایید با استفاده از `telinit` به runlevel شماره ۳ بازگردیم:

```
[root@localhost ~]# telinit 3
[root@localhost ~]# runlevel
5 3
```

همانطور که می‌بینید، استفاده از `telinit` نیز به سادگی استفاده از `init` می‌باشد و این دو دستور، تفاوت زیادی با هم ندارند.

برای خاموش کردن سیستم با `init` یا `telinit`، کافی است به صورت زیر عمل کنیم:

```
[root@localhost ~]# init 0
```

ریبوت کردن سیستم با `init` یا `telinit` نیز بسیار ساده می‌باشد (ما قبل نیز این کار را کرده‌ایم!):

```
[root@localhost ~]# init 6
```

تغییر runlevel (به ۰، ۱ یا ۶) با *shutdown*

زمانی که چندین کاربر دیگر نیز در حال استفاده از سیستم هستند، استفاده از دستور `init` برای خاموش یا ریبوت کردن سیستم، می‌تواند مشکل‌ساز شود؛ چرا که دستور `init`، بدون هیچ هشداری و به صورت کاملا ناگهانی اقدام به خاموش کردن سیستم می‌کند و به سایر کاربران سیستم، مجالی برای ذخیره کردن کار خود نمی‌دهد. به همین دلیل، زمانی که چندین کاربر از یک سیستم استفاده می‌کنند، بهتر است از دستور `shutdown` برای خاموش یا ریبوت سیستم استفاده کنیم.

دستور `shutdown`، به همه‌ی کاربرانی که در حال استفاده از سیستم هستند یک اطلاعیه ارسال می‌کند و سایر کاربران را از login کردن در سیستم باز می‌دارد. به علاوه، `shutdown` به ما امکان می‌دهد که بتوانیم یک تایمر برای خاموش کردن سیستم فعال کنیم و بدین ترتیب، به کاربران سیستم مهلتی برای ذخیره کردن کارهایشان بدهیم.

در ساده‌ترین حالت، از دستور shutdown به صورت زیر استفاده می‌کنیم:

```
[root@localhost ~]# shutdown now
```

دستور بالا، معادل اجرای دستور `init 1` می‌باشد و سیستم را بلافاصله به حالت Single User Mode می‌برد.

پس از اجرای این دستور، کلیه‌ی کاربرانی که در حال کار با سیستم هستند، با اطلاعیه زیر مواجه می‌شوند و بلافاصله دسترسی آنها به سیستم قطع می‌شود:

```
Broadcast message from root@localhost.localdomain
      (/dev/tty1) at 2:17 ...

The system is going down for maintenance NOW
```

اگر به دستور بالا نگاه کنید، می‌بینید که ما پس از دستور shutdown، از عبارت now استفاده کردیم. عبارت now باعث می‌شود که سیستم بدون هیچ درنگی به وضعیت Single User Mode برود. ما می‌توانیم به جای now، از ساعت و دقیقه‌ی مورد نظر جهت اعمال shutdown استفاده کنیم. ما ساعت را در فرمت ۲۴ ساعته و به صورت hh:mm به کار می‌بریم. مثلا اگر بخواهیم سیستم در ساعت ۲۲:۱۵ دقیقه به وضعیت Single User Mode برود، به صورت زیر از shutdown استفاده می‌کنیم:

```
[root@localhost ~]# shutdown 22:15
```

به محض وارد کردن این دستور، کلیه‌ی کاربران سیستم با پیامی نظیر زیر مواجه می‌شوند:

```
Broadcast message from root@localhost.localdomain
      (/dev/pts/1) at 2:26 ...

The system is going down for maintenance in 1189 minutes!
```

اگر بخواهیم سیستم پس از سپری شدن چندین دقیقه به حالت Single User Mode برود، کافی است پس از دستور shutdown، یک علامت + قرار دهیم و بلافاصله پس از آن، دقایق مورد نظر تا اعمال shutdown را قرار دهیم. یعنی:

```
[root@localhost ~]# shutdown +10
```

دستور بالا، سیستم را پس از ۱۰ دقیقه به وضعیت Single User Mode می‌برد و به محض وارد کردن این دستور، کلیه‌ی کاربران سیستم با پیامی نظیر زیر مواجه می‌شوند:

```
Broadcast message from root@localhost.localdomain
      (/dev/tty1) at 2:27 ...

The system is going down for maintenance in 10 minutes!
```

این پیام، در هر دقیقه، تکرار می‌شوند.

ما می‌توانیم با ارائه‌ی آپشن r- به shutdown، کاری کنیم که سیستم پس از مدتی یا به صورت بلادرنگ، ریبوت شود. با ارائه‌ی آپشن P- به shutdown، می‌توانیم کاری کنیم که سیستم پس از مدتی یا به صورت بلادرنگ، خاموش شود. مثلا اگر بخواهیم سیستم پس از ۵ دقیقه خاموش شود:

```
[root@localhost ~]# shutdown -P +5
```

پس از وارد کردن این دستور، کلیه‌ی کاربران سیستم با پیامی نظیر زیر مواجه می‌شوند:

```
Broadcast message from root@localhost.localdomain
      (/dev/tty1) at 2:34 ...

The system is going down for power off in 5 minutes!
```

دقت کنید که این پیام بر خلاف گذشته به کاربران می‌گوید که سیستم پس از ۵ دقیقه Power Off یا خاموش خواهد شد.

اگر بخواهیم هنگام ارائه‌ی دستور shutdown، پیام خاصی به کاربران ارسال شود، کافی است پس از مشخص کردن زمان مورد نظر جهت اعمال shutdown، پیام خود را بین دو علامت "" قرار دهیم. فرض کنید می‌خواهیم سیستم را پس از ۱۵ دقیقه ریبوت کنیم و یک پیام ویژه نیز به همه‌ی کاربران ارسال کنیم:

```
[root@localhost ~]# shutdown -r +15 "I messed up, gonna reboot and then kill myself."
```

پس از وارد کردن این دستور، کلیه‌ی کاربران سیستم پیامی نظیر زیر دریافت می‌کنند:

```
Broadcast message from root@localhost.localdomain
      (/dev/pts/1) at 2:40 ...

The system is going down for reboot in 15 minutes!
I messed up, gonna reboot and then kill myself.
```

همانطور که می‌بینید، پیامی که بین دو علامت "" قرار دادیم دقیقا به همه‌ی کاربران نشان داده شدند.

اگر از تصمیم خود برای shutdown سیستم (خاموش کردن، ریبوت کردن یا بردن سیستم به حالت Single User) منصرف شده باشیم، می‌توانیم با استفاده از آپشن c-، دستور shutdown اعمال شده را کنسل کنیم و دقیقا مانند قبل، می‌توانیم دلیلی برای کنسل کردن shutdown، ارائه کنیم. یعنی:

```
[root@localhost ~]# shutdown -c "nah never mind lol"
```

به محض وارد کردن این دستور، shutdown زمان‌بندی شده‌ی قبلی کنسل شده و کلیه‌ی کاربران پیامی نظیر زیر دریافت می‌کنند:

```
Broadcast message from root@localhost.localdomain
      (/dev/tty1) at 2:43 ...

nah never mind lol
```

Startup Scriptهای SysV

یادگیری مفهوم، مشخص کردن runlevel پیش‌فرض و تغییر runlevelها، اولین قدم در درک و تنظیم چگونگی استارت شدن سرویس‌های سیستم می‌باشد. هر سرویس موجود در سیستم، باید یک اسکریپت، که به آن اسکریپت راه‌اندازی یا Startup Script می‌گویند، داشته باشد. این اسکریپت‌ها، معمولا در دایرکتوری /etc/init.d قرار دارند. بیایید نگاهی به محتویات این دایرکتوری بیاندازیم:

```
[root@localhost ~]# ls -l /etc/init.d/
total 168
-rwxr-xr-x. 1 root root  3580 Mar 22  2017 auditd
-r-xr-xr-x. 1 root root  1362 Nov 15  2017 blk-availability
-rwxr-xr-x. 1 root root  2826 Aug 23  2016 crond
...
-rwxr-xr-x. 1 root root  2571 Jan 26  2017 mdmonitor
-rwxr-xr-x. 1 root root  2011 Dec  1  2017 rsyslog
-rwxr-xr-x. 1 root root  1698 Nov 18  2016 sandbox
...
-rwxr-xr-x. 1 root root   647 Apr 27  2018 single
-rwxr-xr-x. 1 root root  4621 Aug 31  2017 sshd
-rwxr-xr-x. 1 root root  2294 Sep  6  2016 udev-post
```

همانطور که می‌بینید، این دایرکتوری پر از اسکریپت‌های متفاوت می‌باشد. این اسکریپت‌ها، وظیفه‌ی استارت‌زدن، متوقف کردن، restart کردن، reload کردن و نشان دادن وضعیت یک سرویس را بر عهده دارند.

نکته: در صورت نصب سرویس با یک Package Manager نظیر yum، اسکریپت راه‌اندازی مربوطه به صورت اتوماتیک در دایرکتوری init.d قرار می‌گیرد. اگر سرویسی را به صورت دستی نصب کنیم، باید اسکریپت راه‌اندازی آن برنامه را نیز به صورت دستی در این دایرکتوری قرار دهیم (اسکریپتی که خودمان نوشته‌ایم، یا از توسعه دهندگان برنامه گرفته‌ایم).

کلیه این اسکریپت‌ها، توسط یک اسکریپت دیگر به نام rc، اجرا می‌شوند. متاسفانه موقعیت قرارگیری اسکریپت rc از توزیع به توزیع متفاوت می‌باشد، اما این اسکریپت معمولا در دایرکتوری /etc/init.d یا /etc/rc.d قرار خواهد داشت. اسکریپت rc، اسکریپت‌های راه‌اندازی را با توجه به runlevel کنونی، اجرا می‌کند. اما rc از کجا می‌فهمد که در هر runlevel، باید کدامین اسکریپت‌های راه‌اندازی را اجرا کند؟

در سیستم‌هایی که از SysV استفاده می‌کند، برای هر runlevel، یک دایرکتوری مجزا با نام‌های rc?.d وجود دارد؛ به طوری که علامت ?، نشان دهنده‌ی شماره‌ی runlevel (یعنی ۰ تا ۶) می‌باشد. یعنی rc0.d نشان دهنده‌ی دایرکتوری مربوط به runlevel شماره‌ی صفر می‌باشد و به همین ترتیب. این دایرکتوری‌ها، درون دایرکتوری /etc/rc.d قرار دارند. بیایید نگاهی به محتویات دایرکتوری /etc/rc.d بیاندازیم:

```
[root@localhost ~]# ls -l /etc/rc.d/
total 60
drwxr-xr-x. 2 root root  4096 Sep 20 23:35 init.d
-rwxr-xr-x. 1 root root  2617 Jun 19  2018 rc
drwxr-xr-x. 2 root root  4096 Sep  5 17:57 rc0.d
drwxr-xr-x. 2 root root  4096 Sep  5 17:57 rc1.d
drwxr-xr-x. 2 root root  4096 Sep  5 17:57 rc2.d
drwxr-xr-x. 2 root root  4096 Sep  5 17:57 rc3.d
drwxr-xr-x. 2 root root  4096 Sep  5 17:57 rc4.d
drwxr-xr-x. 2 root root  4096 Sep  5 17:57 rc5.d
drwxr-xr-x. 2 root root  4096 Sep  5 17:57 rc6.d
-rwxr-xr-x. 1 root root   220 Jun 19  2018 rc.local
-rwxr-xr-x. 1 root root 20199 Jun 19  2018 rc.sysinit
```

درون هر کدام از دایرکتوری‌های دارای نام rc?.d، یک سری سافت‌لینک به اسکریپت‌های راه‌اندازی که در /etc/init.d دیدیم، وجود دارد. مثلا بیایید محتویات دایرکتوری rc3.d را نگاه کنیم:

```
[root@localhost ~]# ls -l /etc/rc.d/rc3.d/
total 0
lrwxrwxrwx. 1 root root 19 Sep  5 17:56 K10saslauthd -> ../init.d/saslauthd
lrwxrwxrwx. 1 root root 20 Sep  5 17:57 K87multipathd -> ../init.d/multipathd
...
lrwxrwxrwx. 1 root root 14 Sep  5 17:57 S55sshd -> ../init.d/sshd
lrwxrwxrwx. 1 root root 17 Sep  5 17:56 S80postfix -> ../init.d/postfix
lrwxrwxrwx. 1 root root 15 Sep  5 17:56 S90crond -> ../init.d/crond
```

همانطور که می‌بینید، این دایرکتوری شامل یک سری سافت‌لینک به اسکریپت‌های موجود در /etc/init.d می‌باشد. rc، با توجه به runlevel کنونی، کلیه‌ی اسکریپت‌های موجود در دایرکتوری مربوط به آن runlevel را اجرا می‌کند. مثلا اگر ما در runlevel شماره‌ی ۳ باشیم، rc، کلیه‌ی اسکریپت‌های موجود در دایرکتوری /etc/rc.d/rc3.d را اجرا می‌کند.

اگر به نام سافت‌لینک‌ها نگاه کنید، می‌بینید که همه‌ی آنها با حرف K یا S شروع می‌شوند. هنگامی که وارد یک runlevel می‌شویم، اسکریپت rc، پارامتر start را به کلیه‌ی اسکریپت‌هایی که با حرف S شروع می‌شوند و پارامتر stop را به کلیه‌ی اسکریپت‌هایی که نام آنها با K شروع می‌شود، می‌دهد. شماره‌ی موجود پس از K و S، ترتیب start یا stop شدن اسکریپت‌ها را مشخص می‌کند. rc ابتدا کلیه‌ی اسکریپت‌هایی که نامشان با K شروع می‌شود را stop می‌کند و سپس کلیه‌ی اسکریپت‌هایی که نامشان با S شروع می‌شود را استارت می‌زند.

پس به طور خلاصه، به محض ورود به یک runlevel جدید، اسکریپت rc، به سراغ stop کردن و سپس start کردن اسکریپت‌های موجود در دایرکتوری مربوط به آن runlevel خاص می‌رود و بدین ترتیب، سرویس‌های مورد نیاز در هر runlevel، اجرا خواهند شد.

نکته: اگر بخواهیم یک دستور یا یک سری اسکریپت به محض راه‌اندازی سیستم اجرا شود، کافی است آن دستور یا اسکریپت را در فایل /etc/rc.local وارد کنیم. این فایل، مانند startup در ویندوز می‌باشد. نکته‌ای که باید به آن توجه کرد این است که دستورات و اسکریپت‌های موجود در این فایل، پس از اجرای کلیه‌ی اسکریپت‌های مربوط به SysV اجرا خواهند شد.

برای تست عملکرد این فایل، می‌توانیم از دستوری نظیر echo به صورت زیر استفاده کنیم:

```
[root@localhost ~]# vi /etc/rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/locail
echo hello >> /root/greetings
```

همانطور که می‌بینید، ما دستور مشخص شده را به انتهای فایل rc.local اضافه کردیم. حال اگر سیستم را ریبوت کنیم، باید فایلی به نام greetings در دایرکتوری /root ایجاد شده باشد و درون آن فایل، باید رشته‌ی hello نوشته شده باشد. بیایید این امر را امتحان کنیم:

```
[root@localhost ~]# init 6
[root@localhost ~]# ls -l
total 24
-rw-------. 1 root root 1193 Sep  5 17:58 anaconda-ks.cfg
-rw-r--r--. 1 root root    6 Sep 22 08:29 greetings
-rw-r--r--. 1 root root 8901 Sep  5 17:58 install.log
-rw-r--r--. 1 root root 3384 Sep  5 17:57 install.log.syslog
[root@localhost ~]# cat greetings
hello
```

مدیریت سرویس‌های موجود در یک runlevel با *chkconfig*

همانطور که در بخش قبل دیدیم، بررسی این که چه سرویس‌هایی در یک runlevel اجرا می‌شوند، کار دشواری می‌باشد؛ اگر بخواهیم کاری کنیم که یک سرویس جدید در یک runlevel خاص استارت یا متوقف شود، کارمان دشوارتر نیز خواهد شد، چرا که باید در دایرکتوری مربوط به آن runlevel، سافت‌لینکی به اسکریپت مربوط به آن سرویس بزنیم و نام آن سافت‌لینک را با حرف S یا K به اضافه‌ی یک عدد برای مشخص کردن اولویت، درست کنیم. طبیعی است که این کار بسیار دشوار می‌باشد.

ما با استفاده از ابزار chkconfig، می‌توانیم سرویس‌ها و اسکریپت‌های مربوط به آنها را مدیریت کنیم. برای بررسی این که سرویس‌های سیستم در هر runlevel چه وضعیتی دارنداز آپشن --list دستور chkconfig استفاده می‌کنیم:

```
[root@localhost ~]# chkconfig --list
...
network         0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

```
postfix         0:off  1:off  2:on   3:on   4:on   5:on   6:off
rdisc           0:off  1:off  2:off  3:off  4:off  5:off  6:off
restorecond     0:off  1:off  2:off  3:off  4:off  5:off  6:off
rsyslog         0:off  1:off  2:on   3:on   4:on   5:on   6:off
saslauthd       0:off  1:off  2:off  3:off  4:off  5:off  6:off
sshd            0:off  1:off  2:on   3:on   4:on   5:on   6:off
udev-post       0:off  1:on   2:on   3:on   4:on   5:on   6:off
```

همانطور که می‌بینید، خروجی این دستور وضعیت هر سرویس در هر runlevel را نشان می‌دهد. برای مثال، postfix در runlevelهای ۲، ۳، ٤ و ۵ فعال می‌باشد، اما در runlevelهای ۱، ۰ و ۶ غیر فعال می‌باشد.

اگر فقط وضعیت یکی از سرویس‌ها برایمان مهم باشد، می‌توانیم نام سرویس را پس از آپشن `--list` اضافه کنیم. مثلا بیایید وضعیت سرویس `sshd` را در runlevelهای متفاوت بررسی کنیم:

```
[root@localhost ~]# chkconfig --list sshd
sshd            0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

برای این که وضعیت یک سرویس در یک runlevel را تغییر دهیم، به صورت زیر از دستور `chkconfig` استفاده می‌کنیم. مثلا بیایید کاری کنیم که سرویس `rdisc` در runlevelهای ۲ و ۳، فعال باشد:

```
[root@localhost ~]# chkconfig --level 23 rdisc on
```

برای فعال کردن سرویس، از `on` استفاده می‌کنیم، برای غیر فعال کردن از `off` و برای رفتن به وضعیت پیش‌فرض، از `reset` استفاده می‌کنیم. دستور بالا، سرویس `rdisc` را در runlevelهای ۲ و ۳، فعال می‌کند. بیایید نگاهی به وضعیت سرویس `rdisk` بیاندازیم:

```
[root@localhost ~]# chkconfig --list rdisk
rdisc           0:off  1:off  2:on   3:on   4:off  5:off  6:off
```

همانطور که می‌بینید، اکنون `rdisc` در runlevelهای ۲ و ۳ فعال شده است.

نکته: اگر هنگام استفاده از دستور `chkconfig` برای فعال یا غیرفعال کردن یک سرویس در یک runlevel، شماره‌ی runlevelهای مورد نظر را مشخص نکنیم، این دستور به صورت اتوماتیک سرویس مورد نظر ما را در runlevelهای ۲ تا ۵، فعال یا غیر فعال می‌کند. یعنی:

```
[root@localhost ~]# chkconfig --list multipathd
multipathd      0:off  1:off  2:off  3:off  4:off  5:off  6:off
[root@localhost ~]# chkconfig multipathd on
[root@localhost ~]# chkconfig --list multipathd
multipathd      0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

استفاده از دستور service

برای مشاهده‌ی وضعیت یک سرویس در runlevel کنونی، می‌توانیم به صورت زیر از دستور `service` استفاده کنیم:

```
[root@localhost ~]# service sshd status
openssh-daemon (pid  1240) is running...
```

همانطور که می‌بینید، این دستور به ما می‌گوید که سرویس `sshd` در runlevel کنونی، با پراسس‌آی‌دی ۱۲٤۰ در حال اجرا می‌باشد.

برای این که یک سرویس را فقط در session کنونی اجرا کنیم، به صورت زیر از service استفاده می‌کنیم:

```
[root@localhost ~]# service multipathd start
Starting multipathd daemon:        [  OK  ]
```

همانطور که می‌بینید، این دستور سرویس multipathd را استارت زد. بیایید صحت این امر را بررسی کنیم:

```
[root@localhost ~]# service multipathd status
multipathd (pid  1935) is running...
```

توجه کنید که سرویس‌های استارت شده توسط service، فقط تا زمانی که در سیستم login باشیم فعال خواهند ماند و به محض logoff کردن یا تغییر runlevel کنونی، غیرفعال خواهند شد.

برای این که یک سرویس را در session کنونی غیر فعال یا متوقف کنیم، به صورت زیر از service استفاده می‌کنیم:

```
[root@localhost ~]# service multipathd stop
Stopping multipathd daemon:        [  OK  ]
```

همانطور که می‌بینید این دستور سرویس multipathd را متوقف کرد. بیایید صحت این امر را بررسی کنیم:

```
[root@localhost ~]# service multipathd status
multipathd is stopped
```

البته ما همه‌ی آپشن‌های دستور service را بررسی نکردیم. آپشن restart، یک سرویس را متوقف و سپس آن را اجرا می‌کند و آپشن reload، فایل اصلی تنظیمات یک سرویس را بدون متوقف کردن آن، از اول بارگذاری می‌کند.

توجه کنید که این تنظیمات نیز، فقط تا زمانی که در سیستم login باشیم یا در runlevel کنونی باشیم فعال خواهند بود و به محض logoff یا تغییر runlevel، به حالت پیش‌فرض خود باز می‌گردند. اگر بخواهیم این سرویس‌ها پس از ریبوت سیستم، باز هم فعال بمانند (یا غیر فعال شوند)، باید از chkconfig که در بخش قبل آن را توضیح دادیم استفاده کنیم.

استفاده از سیستم راه‌اندازی systemd

سیستم راه‌انداز systemd و رویکرد آن در راه‌اندازی سیستم، یک انقلاب (از نظر برخی مثبت و از نظر برخی منفی) در چگونگی مدیریت سرویس‌ها در لینوکس به وجود آورد. systemd به سیستم‌ها این قابلیت را می‌دهد که علاوه بر استارت زدن سرویس‌ها هنگام روشن شدن، سرویس‌ها را زمانی که یک سخت‌افزار خاص به سیستم متصل می‌شود، یک سرویس دیگر استارت می‌شود، یا پس از گذر یک مدت زمان خاص، استارت بزنند.

بهترین روش برای آشنایی با systemd، صحبت در مورد یونیت‌های systemd (systemd units) می‌باشد. هر یونیت در systemd، معرف یک سرویس، گروهی از سرویس‌ها یا یک عملکرد می‌باشد. هر یونیت، دارای یک نام، یک نوع (type) و یک فایل تنظیمات می‌باشد. در حال حاضر، یونیت‌های systemd دارای ۱۲ نوع (type) متفاوت می‌باشند:

- automount
- device
- mount
- path
- scope
- service
- slice

- snapshot
- socket
- swap
- target
- timer

دستور systemctl، وسیله‌ی اصلی ما برای مدیریت systemd و سرویس‌های موجود در سیستم می‌باشد. به طور کلی، به صورت زیر از systemctl استفاده می‌کنیم:

```
systemctl [OPTIONS...] COMMAND [NAME...]
```

ما می‌توانیم با استفاده از فرمان list-units دستور systemctl، لیستی از یونیت‌هایی که اکنون در سیستم ما بارگذاری شده‌اند را مشاهده کنیم. خروجی این دستور بسیار طولانی می‌باشد، پس ما فقط بخشی از آن را در اینجا قرار می‌دهیم:

```
[root@localhost ~]# systemctl list-units
  UNIT              LOAD      ACTIVE   SUB       DESCRIPTION
  ...
  polkit.service    loaded    active   running   Authorization Manager
  postfix.service   loaded    active   running   Postfix Mail Transport Agent
  ...
  sockets.target    loaded    active   active    Sockets
  ...
```

همانطور که می‌بینید، در خروجی این دستور، چندین یونیت متفاوت به علاوه‌ی اطلاعات جانبی آنها به ما نشان داده می‌شود. در ستون UNIT، ما نام و نوع هر یونیت را در فرمت name.type مشاهده می‌کنیم؛ یعنی در این فرمت، ابتدا نام یونیت و پس از آن نوع یونیت نوشته می‌شود. برای درک بهتر این قضیه، لازم است بدانیم که سرویس‌ها (یا daemonها) در systemd، دارای نوع service می‌باشند و در نتیجه نام یونیت‌فایل آنها، دارای پسوند service. خواهد بود. یعنی یونیت فایل سرویسی نظیر Postfix، برابر با postfix.service خواهد بود.

سرویس‌یونیت‌ها (Service Units)

بیایید ابتدا در مورد سرویس‌یونیت‌ها صحبت کنیم. هر سرویس در سیستم، یک یونیت‌فایل دارد. این فایل‌ها، اطلاعاتی نظیر این که یک سرویس در چه زمانی باید استارت شود، در کدام تارگت استارت شود و... را درون خود دارد. این یونیت‌فایل‌ها، در دایرکتوری‌های متفاوتی قرار گرفته‌اند.

موقعیت قرارگیری این یونیت‌فایل‌ها بسیار مهم می‌باشد، چرا که اگر یک فایل در دو دایرکتوری متفاوت وجود داشته باشد، یکی از دایرکتوری‌ها بر دایرکتوری دیگر تقدم خواهد داشت. دایرکتوری‌هایی که یونیت‌فایل‌ها می‌توانند در آن قرار گیرند، به ترتیب تقدم، به شرح زیر می‌باشند:

- /etc/systemd/system
- /run/systemd/system
- /usr/lib/systemd/system

برای مشاهده‌ی یونیت‌فایل‌های سرویس موجود در سیستم، می‌توانیم از فرمان list-unit-files دستور systemctl استفاده کنیم:

```
[root@localhost ~]# systemctl list-unit-files
UNIT FILE                                  STATE
proc-sys-fs-binfmt_misc.automount          static
dev-hugepages.mount                        static
```

```
dev-mqueue.mount                                static
proc-sys-fs-binfmt_misc.mount                   static
sys-fs-fuse-connections.mount                   static
...
postfix.service                                 enabled
quotaon.service                                 static
rc-local.service                                static
rdisc.service                                   disabled
...
```

همانطور که می‌بینید، در خروجی این دستور، علاوه بر نام یونیت‌فایل‌ها، وضعیت آنها را نیز در ستون STATE مشاهده می‌کنیم. به این وضعیت‌ها، Enablement States یا وضعیت فعال‌بودن می‌گویند. این وضعیت‌ها به زمانی که سرویس استارت شده است اشاره دارند. به طور کلی ۱۲ وضعیت وجود دارد؛ اما معمولا ما به این ۳ نوع وضعیت برخورد خواهیم کرد:

- enabled: سرویس هنگام بوت شدن سیستم استارت می‌شود.
- disabled: سرویس هنگام بوت شدن سیستم استارت نمی‌شود.
- static: سرویس در صورتی استارت می‌شود که یک یونیت دیگر به آن نیاز داشته باشد، یا فرمان استارت به صورت دستی به آن اعمال شود.

برای دیدن این که یک یونیت‌فایل در کدام دایرکتوری یا دایرکتوری‌ها قرار گرفته است، از فرمان cat دستور systemctl استفاده می‌کنیم. برای مثال، بیایید موقعیت یونیت‌فایل سرویس postfix را پیدا کنیم:

```
[root@localhost ~]# systemctl cat postfix.service
# /usr/lib/systemd/system/postfix.service
[Unit]
Description=Postfix Mail Transport Agent
After=syslog.target network.target
Conflicts=sendmail.service exim.service

[Service]
Type=forking
PIDFile=/var/spool/postfix/pid/master.pid
EnvironmentFile=-/etc/sysconfig/network
ExecStartPre=-/usr/libexec/postfix/aliasesdb
ExecStartPre=-/usr/libexec/postfix/chroot-update
ExecStart=/usr/sbin/postfix start
ExecReload=/usr/sbin/postfix reload
ExecStop=/usr/sbin/postfix stop

[Install]
WantedBy=multi-user.target
```

همانطور که می‌بینید، در خط اول خروجی، موقعیت قرارگیری یونیت‌فایل سرویس postfix نشان داده شده است (توجه کنید که یونیت فایل با فایل تنظیمات سرویس فرق دارد). در سایر خط‌ها، کلیه خط‌های موجود در یونیت‌فایل مشخص شده در خط اول، نشان داده شده است.

در یونیت‌فایل هر سرویس، سه بخش اصلی وجود دارد که به شرح زیر می‌باشند:

- [Unit]
- [Service]
- [Install]

در هر کدام از این بخش‌ها، مواردی که به آن Directive گفته می‌شود قرار گرفته است. Directive چیزی

است که یکی از تنظیمات را دگرگون می‌کند. مثلا در خروجی دستور بالا، After یک Directive می‌باشد.

پراستفاده‌ترین Directive موجود در بخش [Unit]، در جدول ۵ قابل مشاهده می‌باشند. در این جدول، «یونیت ما» اشاره به یونیتی دارد در حال بررسی Directiveهای آن هستیم (یعنی مثلا postfix.service).

جدول ۵- پراستفاده‌ترین Directiveهای موجود در [Unit]

Directive	توصیف
After	یونیت ما را پس از یونیت مشخص شده در این Directive اجرا می‌کند.
Before	یونیت ما را قبل از یونیت مشخص شده در این Directive اجرا می‌کند.
Description	یونیت ما را توضیف می‌کند.
Documentation	یک سری URI که به مستندات این یونیت اشاره می‌کنند را مشخص می‌کند. این URIها می‌توانند صفحات وب، فایل‌های محلی، manpage و... باشند.
Conflicts	کاری می‌کند که یونیت ما، همراه با یونیت‌های مشخص شده در این Directive، اجرا نشود. اگر یونیت مشخص شده در این Directive اجرا شود، یونیت ما اجرا نخواهد شد. (برعکس Requires)
Requires	کاری می‌کند که یونیت ما، همراه با یونیت‌های مشخص شده در این Directive، اجرا شود. اگر یونیت مشخص شده در این Directive اجرا نشود، یونیت ما نیز اجرا نخواهد شد.
Wants	یونیت ما را همراه با یونیت‌های مشخص شده در این Directive اجرا می‌کند. اگر یونیت مشخص شده در این Directive اجرا نشود، یونیت ما در هر حال اجرا خواهد شد.

در قسمت [Service] نیز یک سری Directiveهای دیگر قرار گرفته که مختص سرویس مورد بررسی می‌باشند. پراستفاده‌ترین Directiveهای موجود در [Service]، در جدول ۶ قابل مشاهده می‌باشند:

جدول ۶-پراستفاده‌ترین Directiveهای موجود در [Service]

Directive	توصیف
ExecStart	اسکریپت‌ها، دستورها و آپشن‌هایی که باید برای استارت سرویس اجرا شوند را مشخص می‌کند.
ExecStop	اسکریپت‌ها، دستورها و آپشن‌هایی که باید برای توقف سرویس اجرا شوند را مشخص می‌کند.
ExecReload	اسکریپت‌ها، دستورها و آپشن‌هایی که باید برای بارگذاری مجدد یونیت اجرا شوند را مشخص می‌کند.
Environment	متغیرهای محلی یا Environment Variableهای مربوط به یونیت را مشخص می‌کند.

EnvironmentFile	فایلی که حاوی Environment Variableهای مربوط به یونیت می‌باشد را مشخص می‌کند.
RemainAfterExit	مقدار این Directive می‌تواند no (مقدار پیش‌فرض) یا yes باشد. اگر مقدار این Directive، برابر با yes باشد، سرویس ما پس از توقف پراسس‌های استارت شده توسط ExecStart، فعال می‌ماند. اگر مقدار این Directive برابر با no باشد، وقتی پراسس‌های اجرا شده توسط ExecStart متوقف شوند، ExecStop اجرا می‌شود.
Restart	وقتی یکی از پراسس‌های اجرا شده توسط ExecStart متوقف شود، سرویس را restart می‌کند.
Type	Startup Type را مشخص می‌کند. (اطلاعات بیشتر)

در قسمت [Install] نیز Directiveهایی وجود دارد که مشخص کننده‌ی اتفاقی که برای یک سرویس هنگام فعال یا غیر فعال شدن میافتد، می‌باشد. یک سرویس فعال، هنگام بوت شدن سیستم استارت می‌شود و یک سرویس غیرفعال، هنگام بوت شدن سیستم استارت نمی‌شود. پر استفاده‌ترین Directiveهای موجود در [Install] به شرح زیر می‌باشند:

جدول ۷- پراستفاده‌ترین Directiveهای موجود در [Install]

Directive	توصیف
Alias	یک نام مستعار برای سرویس مشخص می‌کند که می‌توانیم به جای نام اصلی از آن استفاده کنیم.
Also	سایر یونیت‌هایی که باید برای این سرویس فعال یا غیرفعال شوند را مشخص می‌کند. معمولا این یونیت‌ها، از نوع socket می‌باشند.
RequiredBy	یونیت‌هایی که به این سرویس نیاز دارند را مشخص می‌کند.
WantedBy	مشخص می‌کند که کدام تارگت‌یونیت این سرویس را مدیریت می‌کند.

تارگت‌یونیت‌ها (Target Units)

درsystemd، تارگت‌یونیت‌ها عملکردی نظیر مفهوم runlevelها در SysV را دارا می‌باشند؛ همانطور که در SysV، هر runlevel گروهی از سرویس‌ها را اجرا می‌کرد، در systemd نیز گروهی از سرویس‌ها، توسط تارگت‌یونیت‌ها اجرا می‌شوند.

هنگام روشن شدن سیستم، یونیت default.target کلیه‌ی سرویس‌های حیاتی و مورد نیاز ما برای روشن شدن صحیح سیستم را اجرا می‌کند. همانطور که حدس زده‌اید، default.target عملکردی نظیر مفهوم default runlevel در SysV را دارا می‌باشد. برای این که ببنیم default.target به کدام تارگت‌یونیت اشاره دارد، می‌توانیم از فرمان get-default دستور sytemctl استفاده کنیم:

```
[root@localhost ~]# systemctl get-default
multi-user.target
```

در جدول ۵، پراستفاده‌ترین تارگت‌یونیت‌ها قابل مشاهده می‌باشند:

جدول ۸= پراستفاده‌ترین تارگت‌یونیت‌ها

نام یونیت‌فایل	توصیف
graphical.target	به چندین کاربر اجازه‌ی دسترسی و استفاده از سیستم را از طریق ترمینال‌های محلی یا ترمینال‌های تحت شبکه می‌دهد. در این حالت امکان ارائه‌ی ظاهر گرافیکی به کاربر نیز وجود دارد.
multi-user.target	به چندین کاربر اجازه‌ی دسترسی و استفاده از سیستم را از طریق ترمینال‌های محلی یا ترمینال‌های تحت شبکه می‌دهد، اما در این حالت امکان ارائه‌ی ظاهر گرافیکی وجود ندارد.
runlevel*n*.target	جهت سازگاری با سیستم‌های دارای SysV وجود دارد؛ به طوری که *n* عددی بین ۱ تا ۵، معادل runlevelهای ۱ تا ۵ در SysV، می‌باشد.

پس هدف اصلی تارگت‌یونیت‌ها، گروه‌بندی چندین سرویس برای استارت شدن هنگام بوت می‌باشد. تارگت‌یونیت پیش‌فرض یا default.target، در واقع به یکی از تارگت‌یوینت‌هایی که در جدول بالا با آن آشنا شدیم سافت‌لینک شده است.

بیایید محتویات فایل تارگت‌یونیت multi-user.target را با هم مشاهده کنیم. برای مشاهده‌ی محتویات این فایل، از فرمان cat دستور systemctl استفاده می‌کنیم:

```
[root@localhost ~]# systemctl cat multi-user.target
# /lib/systemd/system/multi-user.target
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

همانطور که می‌بینید، بسیاری از Directiveهای موجود در فایل تارگت‌یونیت multi-user.target، شبیه به Directiveهایی که در جدول ۵ توصیف کردیم می‌باشند، با این تفاوت که Directiveهای موجود در این تارگت‌یونیت، به جای اشاره به سرویس‌یونیت‌ها، به تارگت‌یونیت‌ها اشاره می‌کنند.

در خروجی بالا، یک Directive جدید به نام AllowIsolate داریم، که قبلا در مورد آن صحبت نکرده‌ایم. وجود مقدار yes برای این Directive، باعث می‌شود که بتوانیم با استفاده از دستور systemctl isolate، تارگت‌یونیت کنونی سیستم را به این تارگت‌یونیت تغییر دهیم. ما در بخش‌های بعدی با این دستور آشنا می‌شویم.

نکته: برخی از اوقات، مجبور به اعمال تغییرات در یونیت‌فایل‌های systemd خواهیم بود (چه یونیت‌فایل‌های سرویس و چه یونیت‌فایل‌های تارگت و...). توجه کنید که در چنین حالتی، بهتر است یونیت‌فایل‌های موجود در

/lib/systemd/system یا /usr/lib/systemd/system را تغییر ندهیم. پیشنهاد می‌شود که در صورت تمایل برای تغییر یک یونیت‌فایل، آن یونیت‌فایل را در دایرکتوری /etc/systemd/system کپی کنیم و تغییرات را در آن فایل به وجود آوریم، چرا که یونیت‌فایل موجود در این دایرکتوری، اولویت بالاتری نسبت به یونیت‌فایل‌های موجود در سایر دایرکتوری‌ها دارند. در این حالت اگر تغییراتی که در یک یونیت‌فایل به وجود آوردیم سبب خرابی سیستم شوند، می‌توانیم با پاک کردن این یونیت‌فایل، تنظیمات آن یونیت را به حالت اولیه بازگردانیم.

بررسی دستور systemctl

دستورات متعددی برای مدیریت systemd و سرویسهای موجود بر روی سیستم وجود دارد، اما دستور systemctl سریع‌ترین و ساده‌ترین وسیله برای مدیریت systemd می‌باشد. ما تا به اینجا کم و بیش با دستور systemctl کار کرده‌ایم، اما در این بخش می‌خواهیم به صورت دقیق‌تر با این دستور آشنا شویم.

systemctl چندین فرمان برای مدیریت سرویس‌ها به ما می‌دهد. یکی از کاربردی‌ترین این فرمان‌ها، status می‌باشد. این فرمان اطلاعات زیادی در مورد یک سرویس را به ما می‌دهد. مثلا بیایید وضعیت سرویس postfix را با استفاده از این فرمان بررسی کنیم:

```
[root@localhost ~]# systemctl status postfix
● postfix.service - Postfix Mail Transport Agent
   Loaded: loaded (/usr/lib/systemd/system/postfix.service; enabled; vendor
preset: disabled)
   Active: active (running) since Mon 2020-09-28 09:47:55 +0330; 2 days ago
  Process: 1275 ExecStart=/usr/sbin/postfix start (code=exited, status=0/SUCCESS)
  Process: 1272 ExecStartPre=/usr/libexec/postfix/chroot-update (code=exited,
status=0/SUCCESS)
  Process: 1258 ExecStartPre=/usr/libexec/postfix/aliasesdb (code=exited,
status=0/SUCCESS)
 Main PID: 1402 (master)
   CGroup: /system.slice/postfix.service
           ├─1402 /usr/libexec/postfix/master -w
           ├─1407 qmgr -l -t unix -u
           └─4488 pickup -l -t unix -u

Sep 28 09:47:53 localhost.localdomain systemd[1]: Starting Postfix Mail Transport
Age....
Sep 28 09:47:55 localhost.localdomain postfix/master[1402]: daemon started --
version ...
Sep 28 09:47:55 localhost.localdomain systemd[1]: Started Postfix Mail Transport
Agent.
```

همانطور که می‌بینید ما با استفاده از systemctl status، وضعیت سرویس postfix را در خروجی دریافت کردیم. اگر به خط دوم خروجی این دستور نگاه کنید، می‌بینید که این سرویس enabled می‌باشد، این یعنی که این سرویس هنگام بوت شدن سیستم، اجرا و فعال خواهد شد. در خط چهارم خروجی، عبارت active (running) را مشاهده می‌کنیم. این امر نشان می‌دهد که سرویس postfix در حال حاضر، اجرا می‌باشد. علاوه بر این، در خط دوم، پس از عبارت loaded، می‌توانیم موقعیت یونیت‌فایل این سرویس را نیز مشاهده کنیم.

حال بیایید وضعیت سرویس rdisc را نیز بررسی کنیم:

```
[root@localhost ~]# systemctl status rdisc
● rdisc.service - rdisc daemon which discovers routers on the local subnet
   Loaded: loaded (/usr/lib/systemd/system/rdisc.service; disabled; vendor
preset: disabled)
   Active: inactive (dead)
```

همانطور که می‌بینید، این سرویس disabled می‌باشد، یعنی هنگام بوت شدن سیستم اجرا نمیشود و همچنین این سرویس inactive می‌باشد، یعنی در حال حاضر اجرا نیست.

به طور کلی، فرمان‌هایی که می‌توانیم به systemctl برای مدیریت سرویس‌ها بدهیم را در جدول ۹ مشاهده می‌کنید. ما از این فرمان‌ها به صورت زیر استفاده می‌کنیم:

```
systemctl COMMAND UNIT-NAME
```

به طوری که UNIT-NAME، نام یونیتی می‌باشد که می‌خواهیم آن را مدیریت کنیم.

جدول ۹- فرمان‌های مدیریت سرویس در systemctl

آپشن	توصیف
daemon-reload	یونیت‌فایل سرویس مشخص شده را بدون متوقف کردن آن سرویس، مجددا بارگذاری می‌کند. این آپشن با آپشن reload تفاوت دارد.
reload	فایل تنظیمات سرویس مشخص شده را بدون متوقف کردن آن سرویس، مجددا بارگذاری می‌کند. این آپشن با آپشن daemon-reload تفاوت دارد؛ چرا که فایل تنظیمات سرویس، تنظیمات کلی یک سرویس را مشخص می‌کند اما یونیت فایل، تنظیمات آن سرویس در systemd را مشخص می‌کند.
disable	کاری می‌کند که سرویس مشخص شده، به صورت اتوماتیک هنگام بوت، راه‌اندازی نشود.
enable	کاری می‌کند که سرویس مشخص شده، به صورت اتوماتیک هنگام بوت، راه‌اندازی شود.
start	سرویس مشخص شده را استارت می‌زند.
restart	سرویس مشخص شده را متوقف و سپس بلافاصله آن را استارت می‌زند. اگر یونیت مشخص شده استارت نبوده باشد، آن را استارت می‌زند.
stop	سرویس مشخص شده را متوقف می‌کند.
status	وضعیت سرویس مشخص شده را نمایش می‌دهد.
mask	از استارت شدن سرویس مشخص شده جلوگیری می‌کند. با استفاده از این آپشن، سرویس حتی با start و یا هنگام بوت نیز استارت نمی‌شود.
unmask	تاثیر mask را از روی سرویس مشخص شده، خنثی می‌کند.

علاوه بر فرمان‌های مشخص شده در جدول بالا، چندین فرمان به‌دردبخور دیگر که از آن برای مشاهده و مدیریت وضعیت سرویس استفاده می‌کنیم به شرح زیر می‌باشند:

جدول ۱۰- فرمان‌های مشاهده‌ی وضعیت سرویس در systemctl

آپشن	توصیف
is-active	برای سرویس‌هایی که در حال اجرا می‌باشند، مقدار active و برای سرویس‌هایی که در وضعیت failed می‌باشند، مقدار failed را نشان می‌دهد.

is-enabled	برای سرویس‌هایی که هنگام بوت شدن سیستم اجرا می‌شوند، مقدار enabled و برای سرویس‌هایی که هنگام بوت شدن سیستم اجرا نمی‌شوند، مقدار disabled را نشان می‌دهد.
is-failed	برای سرویس‌هایی که وضعیت failed رسیده‌اند، مقدار failed و برای سرویس‌هایی که در حال جرا هستند، مقدار active را نشان می‌دهد.

در جدول بالا، ما چندین بار در مورد وضعیت failed صحبت کردیم. سرویس‌ها به دلایل متفاوتی در وضعیت failed قرار می‌گیرند. این دلایل، شامل مشکلات سخت‌افزار، عدم وجود یکی از dependencyها، مشکلات مجوز و... می‌باشند.

بررسی فرمان‌های ویژه‌ی systemctl

دستور systemctl، چندین فرمان بسیار کاربردی دارد که عملکردی فراتر از مدیریت سرویس‌ها دارند. ما با استفاده از این فرمان‌ها می‌توانیم مشخص کنیم که کدام تارگت‌ها (گروهی از سرویس‌ها) هنگام بوت شدن سیستم اجرا می‌شوند، وضعیت سیستم را تغییر دهیم و حتی زمانی که برای بوت شدن سیستم سپری شده را بررسی کنیم. ما در این بخش، به بررسی این فرمان‌های ویژه می‌پردازیم. یکی از مهم‌ترین این فرمان‌ها، فرمان is-system-running می‌باشد. این فرمان وضعیت کنونی سیستم را به ما نشان می‌دهد. بیایید عملکرد این فرمان را بررسی کنیم:

```
[root@localhost ~]# systemctl is-system-running
running
```

شاید خروجی این دستور به نظر به‌دردنخور بیاید، اما این خروجی بدین معنی است که سیستم ما به درستی و با سلامت کامل در حال اجرا می‌باشد. در جدول ۱۱ کلیه‌ی وضعیت‌هایی که ممکن است توسط این دستور به ما نشان داده شود و مفهوم آنها را مشاهده می‌کنیم:

جدول ۱۱- مفهوم وضعیت‌های گزارش‌شده توسط فرمان is-system-running

وضعیت	مفهوم
running	سیستم به صورت کاملا صحیح در اجرا می‌باشد.
degraded	سیستم دارای یک یا چند یونیت failed شده می‌باشد.
maintenance	سیستم در وضعیت emergency یا recovery قرار دارد.
initializing	سیستم در حال شروع فرآیند بوت شدن است.
starting	سیستم هنوز در حال بوت شدن است.
stopping	سیستم در حال خاموش شدن است.

همانطور که گفتیم، اگر سیستم ما در وضعیت degraded قرار داشته باشد، بدین معنی است که یک یا چندتا از یونیت‌های موجود در سیستم در وضعیت failed قرار دارند. در چنین حالتی، برای مشاهده‌ی یونیت‌های failed شده در سیستم، باید از آپشن --failed دستور systemctl، استفاده کنیم:

```
[root@localhost ~]# systemctl --failed
  UNIT           LOAD   ACTIVE SUB    DESCRIPTION
● kdump.service loaded failed failed Crash recovery kernel arming
```

از فرمان‌های کاربردی دیگر systemctl، می‌توان به موارد زیر اشاره کرد:

- get-default

ما از قبل با get-default آشنایی داریم؛ اما به طور کلی، این آپشن تارگت پیش‌فرض سیستم را به ما نشان می‌دهد.

- set-default

با استفاده از set-default، می‌توانیم تارگت پیش‌فرض سیستم را تغییر دهیم.

- isolate

با استفاده از isolate، می‌توانیم تارگت کنونی سیستم را تغییر دهیم (شبیه به دستور init در SysV). زمانی که از فرمان isolate به علاوه‌ی نام یک تارگت‌یونیت استفاده کنیم، کلیه‌ی سرویس‌هایی که در تارگت‌یونیت مشخص شده فعال نیستند (وضعیت enabled ندارند) متوقف می‌شوند و کلیه‌ی سرویس‌هایی که در تارگت‌یونیت مشخص شده باید فعال باشند، استارت می‌شوند (شبیه به مفهوم runlevelها در SysV).

بررسی تارگت‌یونیت‌های ویژه‌ی rescue و emergency

ما دو تارگت ویژه در systemd داریم که rescue و emergency می‌باشند. عملکرد این دو تارگت، به شرح زیر می‌باشد:

- Rescue Target

زمانی که سیستم در تارگت rescue قرار داشته باشد، سیستم فقط پارتیشن‌های محلی را لود می‌کند، فقط کاربر روت می‌تواند وارد سیستم شود و شبکه نیز غیر فعال می‌شود و فقط تعداد بسیاری کمی از سرویس‌ها استارت می‌شوند (حالی شبیه به Single User Mode در SysV). اگر در این تارگت قرار داشته باشیم، خروجی دستور systemctl is-system-running برابر با maintenance خواهد شد. از این تارگت، معمولا برای رفع مشکلات سخت‌افزاری (مشکلات هارددیسک و...) استفاده می‌کنند. برای رفتن به این تارگت، از فرمان isolate استفاده می‌کنیم:

```
[root@localhost ~]# systemctl isolate rescue
```

به محض وارد کردن این دستور، سیستم صفحه‌ای با محتویات زیر به ما نشان می‌دهد:

```
Welcome to rescue mode! after logging in, type “journalctl -xb” to
view system logs, “systemctl reboot” to reboot, “systemctl default”
or ^D to try again to boot into default mode.
Give root password for maintenance
(or press Control-D to continue):
```

در اینجا پس از وارد کردن رمز روت، می‌توانیم دستورات مورد نظر را روی سیستم اجرا کنیم. برای بازگشت به تارگت پیش‌فرض کافی است دستور زیر را اجرا کنیم:

```
[root@localhost ~]# systemctl default
```

- Emergency Target

زمانی که سیستم در تارگت emergency قرار داشته باشد، سیستم فقط پارتیشن روت را به صورت Read-Only مانت می‌کند و فقط به کاربر روت اجازه وارد شدن به سیستم را می‌دهد. به علاوه، شبکه نیز در این حالت غیرفعال می‌شود و فقط تعداد محدودی از سرویس‌ها استارت می‌شوند. اگر در این تارگت قرار داشته باشیم، خروجی دستور systemctl is-system-running برابر با

maintenance خواهد بود. اگر سیستم ما به صورت اتوماتیک به تارگت emergency برود، بدبخت خواهیم شد و این امر نشان می‌دهد که سیستم یک مشکل جدی دارد؛ چرا که سیستم زمانی به این تارگت می‌رود که حتی ورود به تارگت rescue غیرممکن بوده باشد.

برای این که به صورت دستی وارد این تارگت شویم:

```
[root@localhost ~]# systemctl isolate emergency
```

به محض وارد کردن این دستور، سیستم صفحه‌ای با محتویات زیر به ما نشان می‌دهد:

```
Welcome to emergency mode! after logging in, type “journalctl -xb”
to view system logs, “systemctl reboot” to reboot, “systemctl
default” or ^D to try again to boot into default mode.
Give root password for maintenance
(or press Control-D to continue):
```

در اینجا پس از وارد کردن رمز روت، می‌توانیم دستورات مورد نظر را روی سیستم اجرا کنیم. برای بازگشت به تارگت پیش‌فرض کافی است دستور زیر را اجرا کنیم:

```
[root@localhost ~]# systemctl default
```

نکته: برای ریبوت و خاموش کردن سیستم، علاوه بر امکان استفاده از دستور init 0، init 6 و همچنین دستورهایی نظیر shutdown و reboot، می‌توانیم از تارگت‌یونیت‌های reboot و poweroff به صورت systemctl isolate reboot و systemctl isolate poweroff نیز استفاده کنیم.