

## Chapter 07 리눅스의 부팅과 종료

# 목차

01 리눅스 시스템의 부팅

02 systemd 서비스

03 리눅스 시스템의 종료

04 데몬 프로세스

05 부트 로더

# 학습목표



- 리눅스 시스템의 부팅 과정을 이해하고 부트 로더의 역할을 설명할 수 있다.
- systemd의 기능을 이해하고 사용법을 설명할 수 있다.
- 런레벨을 이해하고 변경할 수 있다.
- 리눅스 시스템 종료 방법을 설명할 수 있다.
- 데몬을 이해하고 슈퍼데몬의 역할을 설명할 수 있다.
- root 계정의 암호를 복구할 수 있다.

# 00 Preview

# 00 Preview

## ■ 7장의 내용 구성

- 리눅스 시스템의 부팅과 종료에 대해 배워보기
- 리눅스의 부팅 과정은 PC의 전원을 켜는 순간 바이오스가 동작하는 것으로 시작하여 로그인 프롬프트가 화면에 출력되는 것으로 완료됨
- 로키 리눅스는 systemd 서비스를 도입하여 시스템의 서비스를 시작하고 종료하며 상태를 확인하는 기능을 보강함
- 리눅스는 하드디스크의 추가 같은 유지·보수 작업을 위해 시스템을 종료해야 할 때가 있으므로 시스템 종료 방법을 알아야 함



# 01 리눅스 시스템의 부팅

# 01 리눅스 시스템의 부팅

## ■ 리눅스 시스템의 부팅

- 리눅스의 부팅은 PC의 전원을 켜는 순간부터 리눅스가 완전히 동작하여 로그인 프롬프트가 출력될 때까지를 말함
- 리눅스 시스템의 부팅 과정은 크게 PC 부팅과 리눅스 부팅으로 나뉨.  
즉, 리눅스가 설치된 하드웨어의 부팅과 리눅스 운영체제의 부팅 절차로 구분할 수 있음

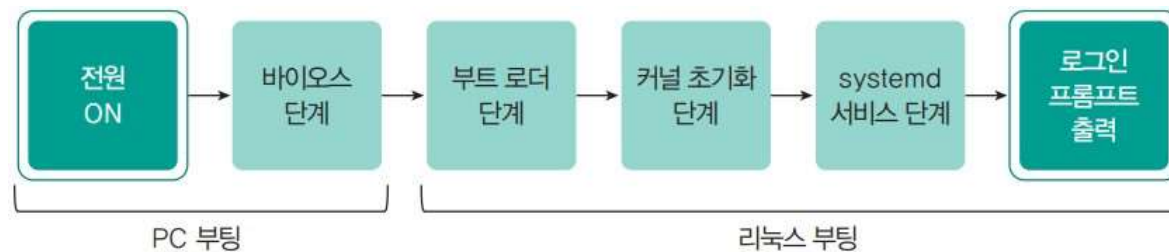


그림 7-1 리눅스의 부팅 과정

# 01 리눅스 시스템의 부팅

## ■ 바이오스 단계

- PC의 전원 스위치를 켜면 제일 먼저 바이오스BIOS, Basic Input/Output System가 동작
- 바이오스는 보통 ROM에 저장되어 있어 흔히 ROM-BIOS라고 부름
- 바이오스는 PC에 장착된 기본적인 하드웨어(키보드, 디스크 등)의 상태를 확인한 후 부팅 장치를 선택하여 부팅 디스크의 첫 섹터에서 512B를 로딩함
- 512B를 마스터 부트 레코드MBR라고 하며, 여기에는 디스크의 어느 파티션에 2차 부팅 프로그램(부트 로더)이 있는지에 대한 정보가 저장되어 있음. MBR은 부트 로더를 찾아 메모리에 로딩하는 작업까지 수행

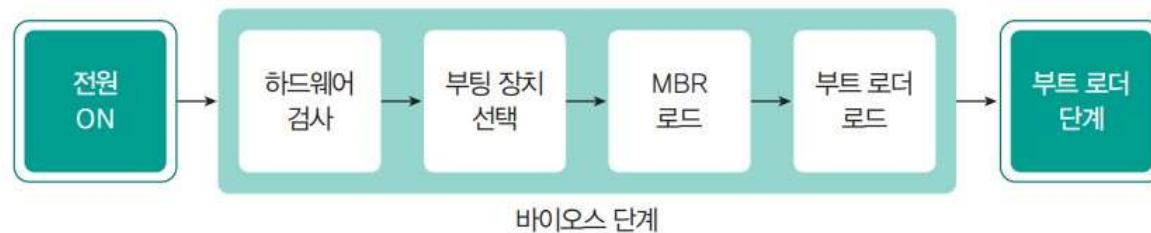


그림 7-2 바이오스 단계의 세부 동작



# 01 리눅스 시스템의 부팅

## ■ 바이오스 단계

- 바이오스 단계에서 MBR은 부트 로더를 찾아 메모리에 로딩함
- 부트 로더는 일반적으로 여러 운영체제 중에서 부팅할 운영체제를 선택할 수 있도록 메뉴를 제공함

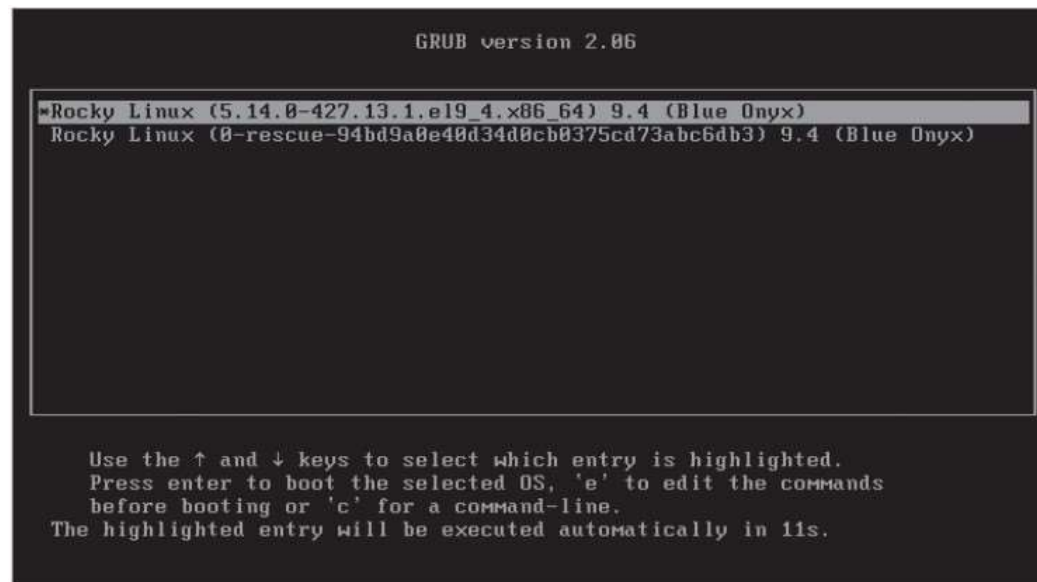


그림 7-3 부트 로더 시작 화면

# 01 리눅스 시스템의 부팅

## ■ 바이오스 단계

- 부트 로더는 리눅스 커널을 메모리에 로딩하는 역할을 수행함. 리눅스 커널은 /boot 디렉터리 아래에 'vmlinuz-버전명'의 형태로 제공됨
- 로키 리눅스의 경우, 처음 설치한 후 업데이트하면 커널이 추가로 생성됨
- rescue 버전은 응급 상황에서 시스템을 복구하는데 사용

```
[user1@localhost ~]$ ls /boot/vm*  
/boot/vmlinuz-0-rescue-94bd9a0e40d34d0cb0375cd73abc6db3*  /boot/vmlinuz-5.14.0-  
427.13.1.el9_4.x86_64
```

- 리눅스의 대표적인 부트 로더는 GRUB

# 01 리눅스 시스템의 부팅

## ■ 커널 초기화 단계

- 부트 로더에 의해 메모리에 로딩된 커널은 가장 먼저 시스템에 연결된 장치를 검사함
- 리눅스를 처음 시스템에 설치할 때 사용 가능한 하드웨어의 정보를 미리 확인해 두고 부팅할 때 이러한 장치들이 사용 가능한 상태로 유지되고 있는지 확인하는 것임
- 장치 검사 등 기본적인 초기화 과정이 끝나면 커널은 일반적으로 프로세스를 만드는 방식인 fork를 사용하지 않고 프로세스와 스레드를 생성함. 이 프로세스들은 메모리 관리 같은 커널의 여러 가지 동작을 수행함
- 커널 프로세스의 개수와 종류는 리눅스의 버전과 종류에 따라 다름. 이 프로세스들은 일반적인 프로세스와 구분되도록 대괄호([ ])로 표시하며, 주로 PID 번호가 낮게 배정되어 있음

# 01 리눅스 시스템의 부팅

## ■ 커널 초기화 단계

```
[user1@localhost ~]$ ps -ef | more
UID      PID    PPID    C   STIME   TTY   TIME      CMD
root      1        0    0   8월21   ?    00:00:14  /usr/lib/systemd/systemd rhgb
--switched-root --system --deserialize 31
root      2        0    0   8월21   ?    00:00:00  [kthreadd]
root      3        2    0   8월21   ?    00:00:00  [rcu_gp]
root      4        2    0   8월21   ?    00:00:00  [rcu_par_gp]
root      5        2    0   8월21   ?    00:00:00  [slub_flushwq]
root      6        2    0   8월21   ?    00:00:00  [netns]
(생략)
```

- 커널 프로세스가 생성되면 커널이 수행할 작업이 끝남. 이제 systemd 서비스를 동작시킴

# 01 리눅스 시스템의 부팅

## ■ systemd 서비스 단계

- systemd 서비스 단계에 이르면 리눅스의 여러 서비스가 활성화됨. 로키 리눅스에서 systemd 서비스는 기존의 init 스크립트를 대체한 것으로 다양한 서비스를 동작시킴
- 각 서비스가 시작되는 과정은 화면에 메시지로 출력되는데, 로키 리눅스에서 기본적으로 이 메시지가 보이지 않도록 이미지를 대신 출력함
- 이 이미지를 부트 스플래시라고 함

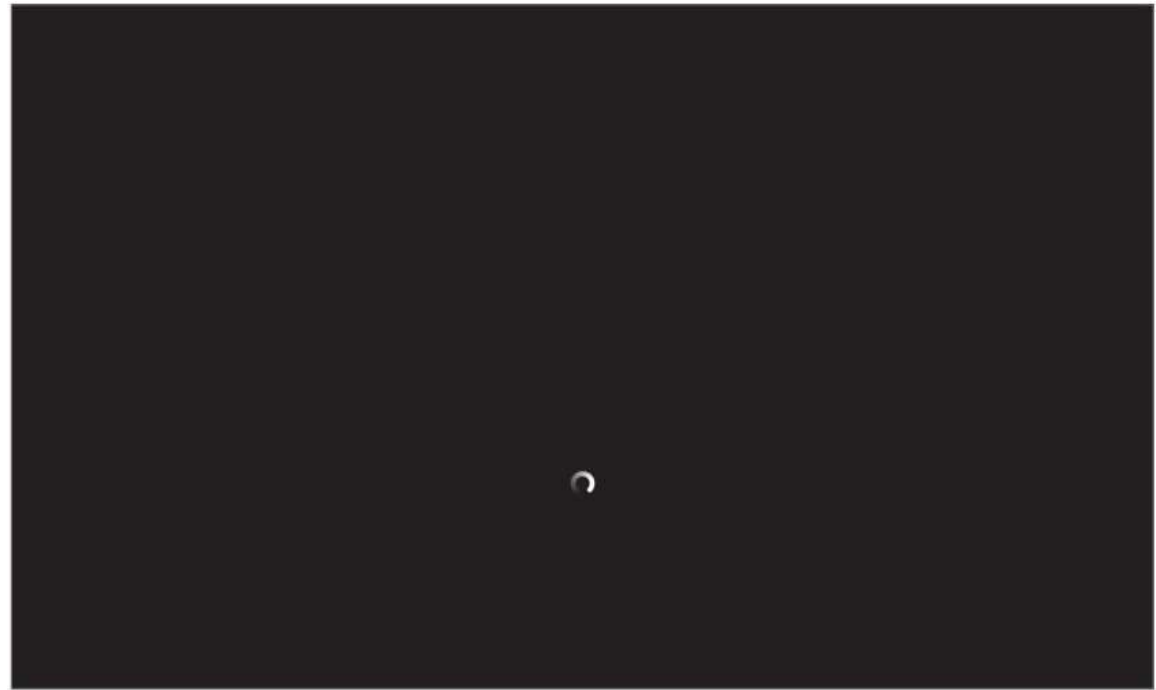


그림 7-4 로키 리눅스의 부트 스플래시 화면

# 01 리눅스 시스템의 부팅

## ■ systemd 서비스 단계

- 부트 스플래시 화면이 진행되고 있을 때 Alt +d를 누르면 메시지가 출력되는 화면으로 전환됨

```
Starting Hostname Service...
[ OK ] Started Hostname Service.
[ OK ] Started Switcheroo Control Proxy service.
[ OK ] Started Manage Sound Card State (restore and store).
[ OK ] Reached target Sound Card.
Starting Load Kernel Module dm...
[ OK ] Finished Load Kernel Module dm.
[ OK ] Started System Logging Service.
[ OK ] Started User Login Management.
[ OK ] Started Daemon for power management.
[ OK ] Started NTP client/server.
[ OK ] Started Authorization Manager.
Starting Modem Manager...
Starting firewalld - dynamic firewall daemon...
[ OK ] Started Accounts Service.
[ OK ] Started Power Profiles daemon.
[ OK ] Started Disk Manager.
[ OK ] Started Modem Manager.
[ OK ] Started firewalld - dynamic firewall daemon.
[ OK ] Reached target Preparation for Network.
Starting Network Manager...
[ OK ] Started Network Manager.
[ OK ] Reached target Network.
Starting Network Manager Wait Online...
Starting CUPS Scheduler...
Starting GSSAPI Proxy Daemon...
Starting OpenSSH server daemon...
[ OK ] Started OpenSSH server daemon.
Starting Network Manager Script Dispatcher Service...
[ OK ] Started CUPS Scheduler.
[ OK ] Started Network Manager Script Dispatcher Service.
[ OK ] Started GSSAPI Proxy Daemon.
[ OK ] Reached target NFS client services.
[ OK ] Finished Network Manager Wait Online.
[ OK ] Reached target Network is Online.
[ OK ] Reached target Preparation for Remote File Systems.
[ OK ] Reached target Remote File Systems.
Starting Crash recovery kernel arming...
Starting Notify NFS peers of a restart...
Starting Permit User Sessions...
[ OK ] Finished Permit User Sessions.
[ OK ] Started Deferred execution scheduler.
[ OK ] Started Command Scheduler.
Starting GNOME Display Manager...
Starting Hold until boot process finishes up...
[ OK ] Started Notify NFS peers of a restart.
[ OK ] Started GNOME Display Manager.
```

그림 7-5 부팅 메시지 출력 화면

# 01 리눅스 시스템의 부팅

## ■ systemd 서비스 단계

- 부팅 시 출력되는 메시지는 각종 서비스가 정상적으로 시작되는지(OK) 아니면 실패인지 (FAIL)를 나타냄. 이 메시지는 부팅 후 `dmesg` 명령이나 `more /var/log/boot.log` 명령으로 확인할 수 있음
- `dmesg` 명령으로 출력되는 메시지에는 데몬의 시작과 관련된 메시지뿐 아니라 하드웨어 검사와 관련된 메시지도 모두 포함되어 있음

```
[user1@localhost ~]$ dmesg | more
[    0.000000] Linux version 5.14.0-427.13.1.el9_4.x86_64 (mockbuild@iad1-prod-build001.bld.equ.rockylinux.org) (gcc (GCC) 11.4.1 20231218 (Red Hat 11.4.1-3), GNU ld version 2.35.2-43.el9) #1 SMP PREEMPT_DYNAMIC Wed May 1 19:11:28 UTC 2024
[    0.000000] The list of certified hardware and cloud instances for Enterprise Linux 9 can be viewed at the Red Hat Ecosystem Catalog, https://catalog.redhat.com.
[    0.000000] Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-5.14.0-284.11.1.el9_2.x86_64 root=/dev/mapper/rl-root ro crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M resume=/dev/mapper/rl-swap rd.lvm.lv=rl/root rd.lvm.lv=rl/swap rhgb quiet
[    0.000000] Disabled fast string operations
(생략)
```

# 01 리눅스 시스템의 부팅

## ■ systemd 서비스 단계

- 전통적으로 유닉스에서는 init 프로세스가 서비스를 실행했음
- init 프로세스는 처음 생성된 프로세스로서 PID가 1번
- 로키 리눅스에서는 init 대신 시스템과 서비스 관리자로 systemd를 사용하므로 systemd 프로세스가 1번 프로세스임

```
[user1@localhost ~]$ ps -ef | more
UID    PID  PPID  C  STIME TTY  TIME      CMD
root     1     0   0   8월21  ?    00:00:14  /usr/lib/systemd/systemd rhgb --switched-
root --system --deserialize 31
root     2     0   0   8월21  ?    00:00:00  [kthreadd]
root     3     2   0   8월21  ?    00:00:00  [rcu_gp]
(생략)
```



# 01 리눅스 시스템의 부팅

## ■ systemd 서비스 단계

- 데몬을 모두 실행한 뒤 그래픽 로그인 시스템인 GDMGnome Display Manager을 동작시키고, 로그인 프롬프트 화면을 출력함

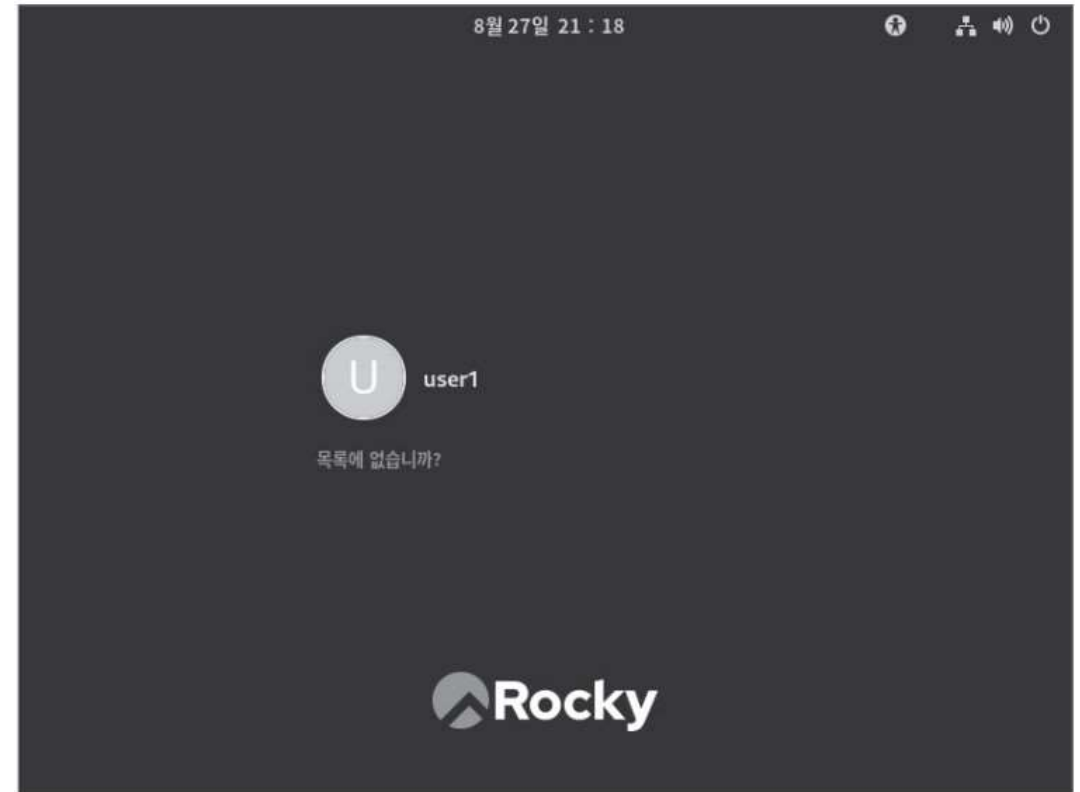


그림 7-6 로그인 프롬프트 화면

## 02 systemd 서비스

## 02 systemd 서비스

### ■ init 프로세스와 런레벨

- init 프로세스는 PID가 1번인 프로세스로 모든 프로세스의 조상 역할을 함
- init 프로세스는 부팅 과정에서 각종 서비스를 제공하는 셸 스크립트 파일을 실행
- init 프로세스와 관련된 설정 파일은 /etc/inittab이며, 현재 로키 리눅스의 /etc/inittab 파일 내용은 다음과 같음

```
[user1@localhost ~]$ cat /etc/inittab
# inittab is no longer used.
#
# ADDING CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
#
# Ctrl-Alt-Delete is handled by /usr/lib/systemd/system/ctrl-alt-del.target
#
# systemd uses 'targets' instead of runlevels. By default, there are two main
targets:
#
# multi-user.target: analogous to runlevel 3
# graphical.target: analogous to runlevel 5
#
# To view current default target, run:
# systemctl get-default
#
# To set a default target, run:
# systemctl set-default TARGET.target
```

## 02 systemd 서비스

### ■ init 프로세스와 런레벨

- init 프로세스가 실행하는 스크립트 파일은 /etc/rc.d/init.d 디렉터리에 위치
- 현재는 init 프로세스의 기능이 systemd로 대체되었으므로 이 디렉터리에는 README 파일만 있음
- README 파일에는 init 스크립트가 systemd 기반으로 바뀌었다는 내용이 들어있음

```
[user1@localhost ~]$ ls /etc/rc.d/init.d
```

```
README
```

```
[user1@localhost ~]$ cat /etc/rc.d/init.d/README
```

```
You are looking for the traditional init scripts in /etc/rc.d/init.d,  
and they are gone?
```

```
Here's an explanation on what's going on:
```

```
You are running a systemd-based OS where traditional init scripts have  
been replaced by native systemd services files. Service files provide  
very similar functionality to init scripts. To make use of service  
files simply invoke "systemctl", which will output a list of all  
currently running services (and other units). Use "systemctl  
list-unit-files" to get a listing of all known unit files, including  
stopped, disabled and masked ones. Use "systemctl start  
foobar.service" and "systemctl stop foobar.service" to start or stop a  
service, respectively. For further details, please refer to  
systemctl(1).
```

```
(생략)
```

## 02 systemd 서비스

### ■ init 프로세스와 런레벨

- init는 시스템의 상태를 일곱 개로 정의하여 구분하고 각 상태에 따라 셸 스크립트를 실행하는데, 이러한 상태를 런레벨이라고 함

표 7-1 init의 런레벨

런레벨	유닉스	로키 리눅스
0	시스템 종료	시스템 종료
1, S	단일 사용자 모드	단일 사용자/복구 모드
2	다중 사용자 모드(NFS 실행 안 함)	다중 사용자 모드
3	다중 사용자 모드(NFS 포함)	
4	사용하지 않음	
5	GUI 상태로 부팅	GUI 상태로 부팅
6	재시작	재시작

## 02 systemd 서비스

### ■ systemd 기본 개념

- systemd는 리눅스의 서비스 관리와 관련된 복잡한 구조로 되어 있음
- systemd는 init 방식에 비해 다음과 같은 장점이 있음
  - 소켓 기반으로 동작하여 inetd와 호환성을 유지한다
  - 셸과 독립적으로 부팅이 가능하다
  - 마운트 제어가 가능하다
  - fsck 제어가 가능하다
  - 시스템 상태에 대한 스냅샷을 유지한다
  - SELinux와 통합이 가능하다
  - 서비스에 시그널을 전달할 수 있다
  - 쉼다운 전에 사용자 세션의 안전한 종료가 가능하다.

## 02 systemd 서비스

### ■ systemd 유닛

- systemd는 전체 시스템을 시작하고 관리하기 위해 유닛이라는 구성 요소를 사용함
- systemd는 유닛들을 '서비스명.유닛 종류'의 형태로 구분하여 관리함
- 각 유닛은 같은 이름과 종류로 구성된 설정 파일과 동일한 이름을 사용함

표 7-2 systemd 유닛의 종류

유닛의 종류	기능	예
service	시스템 서비스 유닛으로 데몬을 시작 · 종료 · 재시작 · 로드한다.	atd.service
socket	소켓을 관리하는 유닛으로 AF_INET, AF_INET6, AF_UNIX 소켓 스트림과 데이터그램, FIFO를 지원한다.	dbus.socket
target	유닛을 그루핑한다. (예 multi-user.target → 런레벨 5에 해당하는 유닛)	basic.target
device	리눅스의 커널 장치를 관리한다.	sys-module-fuse.device
mount	파일 시스템의 마운트 포인트를 관리한다.	tmp.mount
automount	파일 시스템에서 자동 마운트 포인트를 관리한다.	proc-sys-fs-binfmt_misc.automount
timer	타이머와 관련된 기능을 관리한다.	fstrim.timer
swap	스왑 파티션이나 파일을 관리한다.	x2dswap.swap
path	파일 시스템의 개체가 변경되면 다른 서비스를 동작시킨다.	cups.path
slice	시스템 프로세스를 계층적으로 관리한다.	user.slice
scope	외부에서 생성된 프로세스를 관리한다.	init.scope

## 02 systemd 서비스

### ■ systemd 관련 명령

#### systemctl

- **기능** systemd 서비스를 제어한다.
- **형식** systemctl [옵션] [명령] [유닛명]
- **옵션**
  - a: 상태와 관계없이 유닛 전체를 출력한다.
  - t 유닛 종류: 지정한 종류의 유닛만 출력한다.
- **명령**
  - start: 유닛을 시작한다.
  - stop: 유닛을 정지한다.
  - reload: 유닛의 설정 파일을 다시 읽어온다.
  - restart: 유닛을 재시작한다.
  - status: 유닛 상태를 출력한다.
  - enable: 부팅 시 유닛이 시작되도록 설정한다.
  - disable: 부팅 시 유닛이 시작하지 않도록 설정한다.
  - is-active: 유닛이 동작하고 있는지 확인한다.
  - is-enabled: 유닛이 시작되었는지 확인한다.
  - isolate: 지정한 유닛 및 이와 관련된 유닛만 시작하고 나머지는 정지한다.
  - kill: 유닛에 시그널을 전송한다.
- **사용 예**
  - systemctl
  - systemctl -a
  - systemctl start atd.service

- systemd 기반으로 서비스를 시작하거나 종료할 때 사용하는 명령은 systemctl
- systemctl 명령에서 유닛을 지정할 때 유닛의 종류는 제외해도 됨



## 02 systemd 서비스

### ■ 동작 중인 유닛 출력하기

- 옵션이나 명령 없이 systemctl 명령만 사용하면 현재 동작 중인 유닛이 출력됨

```
[user1@localhost ~]$ systemctl
```

UNIT	LOAD	ACTIVE SUB	DESCRIPTION
(생략)			
atd.service	loaded	active running	Deferred execution scheduler
auditd.service	loaded	active running	Security Auditing Service
avahi-daemon.service	loaded	active running	Avahi mDNS/DNS-SD Stack
bluetooth.service	loaded	active running	Bluetooth service
chronyd.service	loaded	active running	NTP client/server
(생략)			

- 다음 실행 결과를 보면 ACTIVE 항목이 모두 active인 것만 출력되었음

## 02 systemd 서비스

### ■ 전체 유닛 출력하기: -a

- systemctl 명령에 -a 옵션을 지정하면 전체 유닛이 출력됨. 다음 출력 결과를 보면 inactive인 것도 포함되어 있음

```
[user1@localhost ~]$ systemctl -a
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
(생략)				
boot.mount	loaded	active	mounted	/boot
dev-hugepages.mount	loaded	active	mounted	Huge Pages File System
dev-mqueue.mount	loaded	active	mounted	POSIX Message Queue File System
● home.mount	not-found	inactive	dead	home.mount
proc-fs-nfsd.mount	loaded	inactive	dead	NFSD configuration filesystem
(생략)				

## 02 systemd 서비스

### ■ 특정 유닛 출력하기: -t

- 특정 종류의 유닛만 출력하려면 -t 옵션을 사용함. 다음은 scope 유닛만 출력한 것

```
[user1@localhost ~]$ systemctl -t scope
```

UNIT	LOAD	ACTIVE SUB	DESCRIPTION
init.scope	loaded	active running	System and Service Manager
session-2.scope	loaded	active running	Session 2 of User user1
session-c1.scope	loaded	active running	Session c1 of User gdm

(생략)

## 02 systemd 서비스

### ■ 유닛의 상태 확인하기: status

- 유닛의 상태를 확인하려면 status 명령을 사용. 다음은 crond.service의 상태를 출력한 것

```
[user1@localhost ~]$ systemctl status crond.service
● crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; preset: enabled)
   Active: active (running) since Sun 2023-08-27 21:18:11 KST; 3 days ago
     Main PID: 1150 (crond)
        Tasks: 2 (limit: 10804)
       Memory: 1.5M
          CPU: 143ms
      CGroup: /system.slice/crond.service
              └─1150 /usr/sbin/crond -n
                 └─2648 /usr/sbin/anacron -s

(생략)
```

- crond.service의 상태 정보를 보면 데몬의 이름이 crond이고, PID가 1150이며, 현재 active 상태임을 알 수 있음

## 02 systemd 서비스

### ■ 유닛 서비스 정지하기: stop

- 유닛 서비스를 정지하려면 stop 명령을 사용
- crond 유닛을 정지한 후 다시 status 명령으로 상태를 확인하면 inactive(dead) 상태
- 서비스 정지는 root의 권한이므로 root로 사용자를 전환하여 실행하기

```
[user1@localhost ~]$ su -  
암호:  
[root@localhost ~]# systemctl stop crond.service  
[root@localhost ~]# systemctl status crond  
○ crond.service - Command Scheduler  
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; preset: enabled)  
   Active: inactive (dead) since Wed 2023-08-30 21:24:04 KST; 8s ago  
 Duration: 3d 5min 52.916s  
 Process: 1150 ExecStart=/usr/sbin/crond -n $CRONDARGS (code=exited, status=0/SUCCESS)  
 Main PID: 1150 (code=exited, status=0/SUCCESS)  
(생략)
```

## 02 systemd 서비스

### ■ 유닛 서비스 시작하기: start

- 유닛 서비스를 시작하려면 start 명령을 사용함
- crond 유닛을 시작한 후 is-active 명령으로 동작 여부를 확인해 보면 active 상태임을 알 수 있음

```
[root@localhost ~]# systemctl start crond.service  
[root@localhost ~]# systemctl is-active crond.service  
active
```

## 02 systemd 서비스

### ■ systemd와 런레벨

- 런레벨은 현재 시스템의 상태를 나타내는 한 자리 숫자(문자 S 포함)
- systemd의 target 유닛의 파일들은 /usr/lib/systemd/system 디렉터리에 있음
- 각 런레벨에 해당하는 runlevelX.target 파일이 제공되는데, 이 파일은 런레벨에 익숙한 사용자의 편의를 위한 심볼릭 링크임

표 7-3 런레벨과 target 유닛의 관계

런레벨	target 파일(심볼릭 링크)	target 원본 파일
0	runlevel0.target	poweroff.target
1	runlevel1.target	rescue.target
2	runlevel2.target	multi-user.target
3	runlevel3.target	
4	runlevel4.target	
5	runlevel5.target	graphical.target
6	runlevel6.target	reboot.target

## 02 systemd 서비스

### ■ 현재 target과 런레벨 확인하기

- 현재 target을 확인하려면 다음과 같이 get-default 명령을 사용

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

- 런레벨을 확인하려면 runlevel 명령을 사용. runlevel 명령으로 확인했을 때 출력된 'N 5'는 런레벨 5로 부팅했다는 의미

```
[root@localhost system]# runlevel  
N 5
```



## 02 systemd 서비스

### ■ 기본 target 지정하기

- 예전에는 부팅할 때 동작하는 기본 런레벨을 /etc/inittab 파일에 지정했으나 지금은 default.target이 가리키는 target 유닛으로 바뀌었음
- default.target은 /etc/systemd/ system 디렉터리 아래에 있으며 심볼릭 링크 파일
- graphical.target의 심볼릭 링크

```
[root@localhost ~]# ls -l /etc/systemd/system/def*  
lrwxrwxrwx. 1 root root 40  7월  9 10:45 /etc/systemd/system/default.target -> /usr/  
lib/systemd/system/graphical.target
```

- default.target이 가리키는 기본 target은 다음과 같은 형식으로 지정함

```
systemctl set-default <name of target>.target
```

## 02 systemd 서비스

### ■ 기본 target 지정하기

- /etc/systemd/system 디렉터리 아래의 심볼릭 링크인 default.target이 가리키는 target 파일을 변경함
- 현재 target인 graphical.target에서 multi-user.target으로 바꾸려면 다음과 같이 작성함

```
[root@localhost ~]# systemctl set-default multi-user.target
Removed "/etc/systemd/system/default.target".
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/multi-user.target.
[root@localhost ~]# ls -l /etc/systemd/system/def*
lrwxrwxrwx. 1 root root 41  8월 30 21:35 /etc/systemd/system/default.target -> /usr/lib/systemd/system/multi-user.target
```

## 02 systemd 서비스

### ■ 기본 target 지정하기

- 런레벨 0이나 런레벨 6에 해당하는 target을 기본으로 지정하면 안 됨
- 기본 target은 graphical.target(런레벨 5)으로 해놓는 것이 좋음
- 앞에서 바꾼 기본 target을 다시 graphical.target으로 변경함

```
[root@localhost ~]# systemctl set-default graphical.target
Removed "/etc/systemd/system/default.target".
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/graphical.target.
```

## 02 systemd 서비스

### ■ target 변경하기

- 현재의 target을 다른 target으로 바꿔야 할 때 systemd의 isolate 명령으로 간단히 해결할 수 있음
- multi-user.target(runlevel3)으로 변경하려면 다음 명령 중 하나를 입력함

```
systemctl isolate multi-user
```

```
systemctl isolate runlevel3
```

- graphical.target(runlevel5)으로 변경하려면 다음 명령 중 하나를 사용

```
systemctl isolate graphical
```

```
systemctl isolate runlevel5
```

## 02 systemd 서비스

### ■ 런레벨 변경하기: telinit, init

- init는 1번 프로세스의 이름이자 init 프로세스의 런레벨을 바꿀 때 사용하는 명령이기도 함
- 현재 init는 systemd에 대한 심볼릭 링크

```
[root@localhost ~]# ls -l /sbin/init
lrwxrwxrwx. 1 root root 22  5월  9 17:48 /sbin/init -> ../lib/systemd/systemd
```

- init 명령만 입력하면 다음과 같이 출력됨

```
[root@localhost ~]# init
init: required argument missing.
```

## 02 systemd 서비스

### ■ 런레벨 변경하기: telinit, init

- `init --help`로 사용법을 알아보면 다음과 같이 출력됨
- 기존 `init` 프로세스에 익숙한 사용자들을 위해 명령을 유지하고 있는 것으로, 예를 들어 `init 3`을 입력하면 런레벨 3으로 변경됨

```
[root@localhost ~]# init --help
init [OPTIONS...] COMMAND

Send control commands to the init daemon.

Commands:
  0          Power-off the machine
  6          Reboot the machine
  2, 3, 4, 5 Start runlevelX.target unit
  1, s, S    Enter rescue mode
  q, Q       Reload init daemon configuration
  u, U       Reexecute init daemon

Options:
  --help    Show this help
  --no-wall Don't send wall message before halt/power-off/reboot

See the telinit(8) man page for details.
```

## 02 systemd 서비스

### ■ 런레벨 변경하기: telinit, init

- 런레벨을 바꾸는 명령으로 telinit도 있음
- telinit는 systemctl에 대한 심볼릭 링크
- telinit 명령을 실행하면 init 명령과 같은 결과가 출력됨

```
[root@localhost ~]# ls -l /sbin/telinit  
lrwxrwxrwx. 1 root root 16 5월 9 17:48 /sbin/telinit -> ../bin/systemctl
```

## 02 systemd 서비스

### ■ 단일 사용자 모드로 전환하기: rescue.target(런레벨 1)

- 시스템에 문제가 생기면 시스템을 rescue.target 유닛(런레벨 1, 런레벨 S)으로 변경하여 점검 해야 함  
다중 사용자 모드에서 시스템 관리자만 사용할 수 있는 단일 사용자 모드로 전환하는 것
- 이 모드로 변환하기 전에 다른 사용자들은 로그아웃해야 함

```
systemctl isolate rescue
```

```
systemctl isolate runlevel1
```

```
init 1 또는 init S
```

```
telinit 1 또는 init S
```

- 단일 사용자 모드로 전환하면 그래픽 환경이었던 리눅스가 텍스트 모드로 바뀜. 단일 사용자 모드에서는 바로 root 암호를 입력하여 시스템과 관련된 점검 작업을 하고 다시 다중 사용자 모드로 전환함
- 단일 사용자 모드에서 다중 사용자 모드로 전환하려면 reboot 명령이나 systemctl default 명령을 사용함



## 02 systemd 서비스

### ■ 서비스 정지하고 시작하기

#### 혼자해보기 서비스 정지하고 시작하기

- ① colord.service가 동작 중인지 is-active 명령으로 확인한다.
- ② colord.service를 정지한다.
- ③ colord.service의 상태를 확인한다.
- ④ colord.service를 다시 시작한다.
- ⑤ colord.service의 상태에서 PID를 확인한다.

## 03 리눅스 시스템의 종료

## 03 리눅스 시스템의 종료

### ■ 리눅스 시스템의 종료

- 리눅스에서 시스템을 종료할 때 정상적인 절차를 거치는 것이 매우 중요.  
리눅스는 비정상적으로 시스템을 종료하여 문제가 발생하면 서비스를 제공하지 못할 수도 있음
- 리눅스를 종료하는 방법. 전원 끄기는 다른 무엇도 할 수 없을 때만 사용하는 최후의 수단임
  - shutdown 명령을 사용한다
  - 런레벨을 0이나 6으로 전환한다
  - halt 명령을 사용한다
  - poweroff 명령을 사용한다
  - reboot 명령을 사용한다
  - 전원을 끈다.

## 03 리눅스 시스템의 종료

### ■ shutdown 명령

- 리눅스 시스템을 가장 정상적으로 종료하는 방법은 shutdown 명령을 사용하는 것
- shutdown 명령은 다양한 종료 방법을 제공함. 시스템 종료 외에 런레벨을 바꿀 때도 사용할 수 있음

#### shutdown

- 기능 리눅스를 종료한다.
- 형식 shutdown [옵션] [시간] [메시지]
- 옵션
  - k: 실제로 시스템을 종료하는 것이 아니라 사용자들에게 메시지만 전달한다.
  - r: 종료 후 재시작한다.
  - h: 종료하며 halt 상태로 이동한다.
  - f: 빠른 재시작으로 이 과정에서 fsck를 생략할 수도 있다.
  - c: 이전에 내렸던 shutdown 명령을 취소한다.
- 시간 종료할 시간(hh:mm, +m, now)
- 메시지 모든 사용자에게 보낼 메시지
- 사용 예

```
shutdown -h now
shutdown -r +3 "System is going down"
shutdown -c
```

## 03 리눅스 시스템의 종료

### ■ 시스템 즉시 종료하기

- shutdown 명령으로 시스템을 즉시 종료하려면 -h 옵션과 함께 now로 현재 시간을 지정함

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

- 종료 시간이 now이므로 바로 시스템 종료 절차가 시작됨

### ■ 쉿다운한다는 메시지 보내고 종료하기

- 시스템을 종료할 때 shutdown 명령으로 메시지를 보낼 수 있음
- 사용자들이 메시지를 받고 정리할 시간이 필요하므로 시간을 now로 지정하면 안되고 특정 시간을 지정해야 함
- 2분 후에 시스템이 종료되도록 설정하고 메시지를 보내기

```
[root@localhost ~]# shutdown -h +2 "System is going down in 2 min"
```

## 03 리눅스 시스템의 종료

### ■ 시스템 재시작하기

- shutdown 명령으로 시스템을 재시작하려면 -r 옵션을 사용함
- 3분 후에 시스템을 재시작 하도록 지정하려면 다음과 같이 함

```
[root@localhost ~]# shutdown -r +3
```

### ■ 명령 취소하기

- shutdown 명령을 취소하려면 -c 옵션을 사용함

```
[root@localhost ~]# shutdown -c
```

## 03 리눅스 시스템의 종료

### ■ 메시지만 보내기

- 실제로 shutdown 명령을 실행하지는 않고 사용자들에게 메시지만 보내려면 -k 옵션을 사용함

```
[root@localhost ~]# shutdown -k 2
```

- 현재 시각을 알려주고 2분 후에 시스템이 종료된다는 메시지를 보내는데, 그 시간이 되었을 때 실제로 시스템이 종료되지는 않고 자동으로 취소됨

## 03 리눅스 시스템의 종료

### ■ 런레벨 변경 명령

- 시스템을 종료하는 다른 방법은 런레벨을 변경하는 것
- telinit 명령으로 런레벨을 0으로 변경하면 시스템이 종료되고, 6으로 변경하면 재시작됨
- systemd 기능을 사용할 때는 target을 바꾸면 됨

### ■ 런레벨 변경하기

- 런레벨 0은 시스템 종료 상태. 따라서 런레벨을 0으로 바꾸면 시스템이 종료됨

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

- 시스템을 재시작하려면 런레벨을 6으로 바꾸면 됨

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



## 03 리눅스 시스템의 종료

### ■ systemd로 종료하기

- systemd에서 target 유닛을 변경하면 시스템을 종료하거나 재시작할 수 있음
- 다음과 같이 작성하면 시스템을 종료할 수 있음

```
[root@localhost ~]# systemctl isolate poweroff.target
```

```
[root@localhost ~]# systemctl isolate runlevel0.target
```

- 다음과 같이 작성하면 시스템을 재시작할 수 있음

```
[root@localhost ~]# systemctl isolate reboot.target
```

```
[root@localhost ~]# systemctl isolate runlevel6.target
```

## 03 리눅스 시스템의 종료

### ■ 기타 시스템 종료 명령

- shutdown 명령과 런레벨 변경 외에 시스템을 종료하거나 재시작하기 위해 사용할 수 있는 명령으로 halt, poweroff, reboot가 있음. 이 명령들은 모두 systemctl 명령의 심볼릭 링크

```
[root@localhost ~]# ls -l /sbin/telinit
lrwxrwxrwx. 1 root root 16 5월 9 17:48 /sbin/telinit -> ../bin/systemctl
[root@localhost ~]# ls -l /sbin/halt
lrwxrwxrwx. 1 root root 16 5월 9 17:48 /sbin/halt -> ../bin/systemctl
[root@localhost ~]# ls -l /sbin/poweroff
lrwxrwxrwx. 1 root root 16 5월 9 17:48 /sbin/poweroff -> ../bin/systemctl
[root@localhost ~]# ls -l /sbin/reboot
lrwxrwxrwx. 1 root root 16 5월 9 17:48 /sbin/reboot -> ../bin/systemctl
```

- 결국 모두 systemctl 명령으로 시스템을 종료하고 재시작한다는 의미. 이 명령들은 전통적으로 존재하던 명령이라 계속 유지되고 있는 것

## 03 리눅스 시스템의 종료

### ■ 기타 시스템 종료 명령

- halt, poweroff, reboot 명령은 /var/log/wtmp 파일에 시스템 종료 기록을 남기고 시스템을 종료하거나 재시작함
- 사용할 수 있는 옵션은 다음과 같음
  - -n: 재시작하거나 종료하기 전에 sync를 호출하지 않는다
  - -w: 실제로 재시작하거나 종료하지는 않지만 wtmp 파일에 기록을 남긴다
  - -d: wtmp 파일에 기록을 남기지 않는다
  - -n 옵션은 -d 옵션을 포함한다
  - -f: 강제로 명령을 실행하며 shutdown을 호출하지 않는다
  - -p: 시스템의 전원을 끈다

## 04 데몬 프로세스

## 04 데몬 프로세스

### ■ 데몬 프로세스

- 데몬: 리눅스의 백그라운드에서 동작하며 특정 서비스를 제공하는 프로세스
- 리눅스 시스템에서 동작하는 웹 서버나 데이터베이스 서버, 원격 접속 서버 등 각종 서비스를 제공하는 프로세스

### ■ 데몬의 동작 방식

- 독자형: 데몬 혼자서 스스로 동작. 시스템의 백그라운드에서 항상 동작하고 있음. 자주 호출되는 데몬이 아니라면 시스템의 자원을 낭비할 우려가 있음
- 슈퍼데몬에 의해 동작: 평소 슈퍼데몬만 동작하다가 서비스 요청이 오면 슈퍼데몬이 해당 데몬을 동작시키는 것. 독자형보다는 서비스에 응답하는데 시간이 좀 더 걸릴 수 있지만 자원을 효율적으로 사용한다는 장점이 있음
- 데몬이 제대로 동작하지 않으면 시스템이 서비스를 제공할 수 없음

## 04 데몬 프로세스

### ■ 슈퍼데몬

- 데몬의 종류가 늘어나자 이를 관리하기 위한 슈퍼데몬이 등장
- 유닉스의 슈퍼데몬은 이름이 inetd였으나 로키 리눅스에서는 보안 기능이 포함된 xinetd를 사용하고 있음
- 슈퍼데몬은 사용자의 요청을 받아서 해당하는 서비스 데몬을 동작시킴

## 04 데몬 프로세스

### ■ systemd 데몬

- systemd는 init를 대체한 데몬. systemd는 1번 프로세스로서 대다수 프로세스의 조상 프로세스이며, 시스템의 상태를 종합적으로 관리하는 역할을 수행

```
[user1@localhost ~]$ pstree
systemd├─ModemManager──3*[{ModemManager}]
      │├─NetworkManager──2*[{NetworkManager}]
      │├─VGAuthService
      │├─accounts-daemon──3*[{accounts-daemon}]
      │├─alsactl
      │├─at-spi-bus-laun├─dbus-daemon
      ││               └─3*[{at-spi-bus-laun}]
      │├─at-spi2-registr──2*[{at-spi2-registr}]
      │├─atd
      │├─auditd├─sedispatch
      ││       └─2*[{auditd}]
```

(생략)

## 04 데몬 프로세스

### ■ 커널 스레드 데몬

- 커널 스레드: 커널 일부를 프로세스처럼 관리하는 데몬. ps 명령으로 확인했을 때 대괄호([ ])에 들어 있는 프로세스. 대부분 입출력이나 메모리 관리, 디스크 동기화 등을 수행하며 대체로 낮은 PID가 할당되어 있음
- 커널 스레드를 동작시키는  
조상 데몬은 커널 스레드 데몬
- ps 명령으로 확인해 보면 모든  
커널 스레드의 PPID가 2번임을  
알 수 있음
- PID 2번은 kthreadd 데몬

```
[user1@localhost ~]$ ps -ef | more
UID    PID  PPID  C  STIME TTY  TIME      CMD
root     1      0  0  17:33  ?    00:00:04  /usr/lib/systemd/systemd rhgb --switched-
root --sys
tem --deserialize 31
root     2      0  0  17:33  ?    00:00:00  [kthreadd]
root     3      2  0  17:33  ?    00:00:00  [rcu_gp]
root     4      2  0  17:33  ?    00:00:00  [rcu_par_gp]
root     5      2  0  17:33  ?    00:00:00  [slub_flushwq]
root     6      2  0  17:33  ?    00:00:00  [netns]
(생략)
```



## 04 데몬 프로세스

### ■ 주요 데몬

- 리눅스에서 각종 서비스를 제공하기 위한 주요 데몬은 [표 7-4]와 같음
- 이외에도 매우 다양 한 데몬이 존재하며, 사용자가 필요에 따라 만들 수도 있음

표 7-4 리눅스의 주요 데몬

데몬	기능
atd	특정 시간에 실행하도록 예약한 명령을 실행한다(at 명령으로 예약).
crond	주기적으로 실행하도록 예약한 명령을 실행한다.
dhcpcd	동적으로 IP 주소를 부여하는 서비스를 제공한다.
httpd	웹 서비스를 제공한다.
lpd	프린트 서비스를 제공한다.
nfs	네트워크 파일 시스템 서비스를 제공한다.
named	DNS 서비스를 제공한다.
sendmail	이메일 서비스를 제공한다.
smtpd	메일 전송 서비스를 제공한다.
popd	기본 편지함 서비스를 제공한다.
routed	자동 IP 라우터 테이블 서비스를 제공한다.
smb	삼바 서비스를 제공한다.
syslogd	로그 기록 서비스를 제공한다.
sshd	원격 보안 접속 서비스를 제공한다.
in.telnetd	원격 접속 서비스를 제공한다.
ftpd	파일 송수신 서비스를 제공한다.
ntpd	시간 동기화 서비스를 제공한다.

## 05 부트 로더

## 05 부트 로더

### ■ GRUB의 개요

- GRUB는 'Grand Unified Bootloader'의 약자로, 리눅스의 전통적인 부트 로더인 LILO의 단점을 보완하여 GNU 프로젝트의 일환으로 개발되었음
- GRUB는 LILO에 비해 다음과 같은 장점이 있음
  - LILO는 리눅스에서만 사용할 수 있지만 GRUB는 윈도우에서도 사용할 수 있다
  - LILO에 비해 설정과 사용이 편리하다
  - 부팅할 때 명령을 사용하여 수정할 수 있다
  - 멀티 부팅 기능을 지원한다
- GRUB의 최신 버전은 GRUB2. GRUB를 대체한 것으로, 로키 리눅스는 GRUB2를 기본 부트 로더로 사용하고 있음. GRUB2는 이식성과 모듈화가 더 좋아져서 동적으로 모듈을 로딩할 수 있음

## 05 부트 로더

### ■ /boot/grub2/grub.cfg 파일

- /boot/grub2/grub.cfg 파일은 사용자가 직접 수정할 수 없음
- 이 파일은 /etc/default/grub 파일과 /etc/grub.d 디렉터리 아래에 있는 스크립트를 읽어서 생성됨
- 이 파일의 내용을 수정하려면 /etc/default/grub 파일과 /etc/grub.d 디렉터리 아래에 있는 스크립트를 수정해야 함
- /etc/default/grub 파일이 수정되면 grub2-mkconfig -o /boot/grub2/grub.cfg 명령으로 적용해야 함

## 05 부트 로더

### ■ /boot/grub2/grub.cfg 파일

```
[root@localhost ~]# more /boot/grub2/grub.cfg
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by grub2-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#

### BEGIN /etc/grub.d/00_header ###
set pager=1

if [ -f ${config_directory}/grubenv ]; then
  load_env -f ${config_directory}/grubenv
elif [ -s $prefix/grubenv ]; then
  load_env
fi
(생략)
```

## 05 부트 로더

### ■ /etc/grub.d 디렉터리

- 이 디렉터리는 GRUB 스크립트를 가지고 있음
- 이 스크립트들은 GRUB의 명령이 실행될 때 순서대로 실행되어 grub.cfg 파일을 생성함

```
[root@localhost ~]# ls /etc/grub.d
00_header  08_fallback_counting  12_menu_auto_hide  20_ppc_terminfo  35_fwupd README
00_tuned   10_linux              14_menu_show_once  30_os-prober     40_custom
01_users   10_reset_boot_success 20_linux_xen       30_uefi-firmware 41_custom
```

## 05 부트 로더

### ■ /etc/default/grub 파일

- 이 파일에는 GRUB 메뉴 설정 내용이 저장되어 있음
- GRUB 스크립트가 이 파일을 읽어서 grub.cfg에 기록함

```
[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=1G-4G:192M,4G-64G:256M,64G-:512M resume=/dev/mapper/
rl-swap rd.lvm.lv=rl/root rd.lvm.lv=rl/swap rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
GRUB_ENABLE_BLSCFG=true
```

## 05 부트 로더

### ■ root 암호 복구하기

- GRUB2 부트 로더와 관련된 사용법 중에서 꼭 알아 두어야 할 것은 root 암호를 복구하기 위해 단일 사용자 모드로 부팅하는 방법. 이는 root 계정의 암호를 잊어버렸을 때 암호를 복구하는데 필요함

#### ① 시스템 재시작하기

- 일단 시스템을 재시작해야 함. 부팅할 때 [그림 7-7]과 같이 GRUB 메뉴 초기 화면이 출력됨



그림 7-7 GRUB 메뉴 초기 화면




## 05 부트 로더

### ■ root 암호 복구하기

#### ② GRUB 편집 모드로 전환하기

- GRUB Boot Menu가 출력될 때 신속하게 e 키를 눌러서 편집 모드로 전환해야 함
- 편집 모드가 되면 [그림 7-8]과 같이 편집 가능한 상태가 됨



```
GRUB version 2.06

load_video
set gfxpayload=keep
insmod gzio
linux ($root)/vmlinuz=5.14.0-284.11.1.el9_2.x86_64 root=/dev/mapper/rl-root\
ro crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M resume=/dev/mapper/rl-swap\
rd.lvm.lv=rl/root rd.lvm.lv=rl/swap rhgb quiet
initrd ($root)/initramfs-5.14.0-284.11.1.el9_2.x86_64.img $tuned_initrd

Minimum Emacs-like screen editing is supported. TAB lists
completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for
a command-line or ESC to discard edits and return to the GRUB menu.
```


그림 7-8 GRUB 편집 화면

## 05 부트 로더

### ■ root 암호 복구하기

#### ③ 단일 사용자 모드로 수정하기

- 커서를 아래로 이동하여 리눅스 커널 정보가 있는 행에서 ro crachkernel부터 quiet까지 삭제하고 rd.break enforcing=0을 추가함



```
GRUB version 2.06

load_video
set gfxpayload=keep
insmod gzio
linux ($root)/vmlinuz-5.14.0-284.11.1.el9_2.x86_64 root=/dev/mapper/r1-root\
rd.break enforcing=0
initrd ($root)/initramfs-5.14.0-284.11.1.el9_2.x86_64.img $tuned_initrd

Minimum Emacs-like screen editing is supported. TAB lists
completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for
a command-line or ESC to discard edits and return to the GRUB menu.
```

그림 7-9 root 암호 복구를 위한 리눅스 커널 항목 수정 화면

- rd.break는 부팅 단계에서 초기 단계만 수행한다는 뜻이며, enforcing=0은 selinux를 강제로 적용하지 않도록 한다는 뜻
- selinux는 보안 설정 도구. 강제로 적용하지 않도록 한다는 것은 보안을 해제한다는 의미

## 05 부트 로더

### ■ root 암호 복구하기

#### ④ 재시작하기

- Ctrl +x를 눌러 재시작 함
- rd.break enforcing=0을 추가했다면 단일 사용자 모드로 부팅하여 바로 root 계정으로 동작함

```
[ OK ] Reached target Initrd Root Device.
[ OK ] Finished dracut initqueue hook.
[ OK ] Reached target Preparation for Remote File Systems.
[ OK ] Reached target Remote File Systems.
Starting File System Check on /dev/mapper/r1-root...
[ OK ] Finished File System Check on /dev/mapper/r1-root.
Mounting /sysroot...
[ 31.843888] SGI XFS with ACLs, security attributes, scrub, quota, no debug enabled
[ 31.847888] XFS (dm-1): Mounting U5 Filesystem
[ 32.113153] XFS (dm-1): Ending clean mount
[ OK ] Mounted /sysroot.
[ OK ] Reached target Initrd Root File System.
Starting Mountpoints Configured in the Real Root.
Type '!' to finish. To override mounts configured in the Real Root,
Type '!' to finish. To override mounts configured in the Real Root,
You might want to save "/run/ramfs/rdsosrep" to a USB stick or /boot
after mounting them and attach it to a bug report.

switch_root:/# [ 34.891775] usb 2-2.1: new full-speed USB device number 4 using uhci_hcd
[ 34.991989] usb 2-2.1: New USB device found, idVendor=0e0f, idProduct=0000, bcdDevice=1.00
[ 34.992629] usb 2-2.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 34.993128] usb 2-2.1: Product: Virtual Bluetooth Adapter
[ 34.994098] usb 2-2.1: Manufacturer: VMware
[ 34.994893] usb 2-2.1: SerialNumber: 000650268328

switch_root:/# _
```

그림 7-10 root 암호 복구를 위한 화면

- [그림 7-10]과 같이 부팅된 경우 루트 파일 시스템이 읽기 전용으로 마운트되므로 읽기/쓰기 모드로 다시 마운트해야 함
- 시스템의 메시지가 기본적으로 한글로 출력되는데 콘솔 모드에서는 한글이 출력되지 않기 때문에 언어를 영어로 바꿔야 함

## 05 부트 로더

### ■ root 암호 복구하기

- [그림 7-10]과 같이 부팅된 경우 루트 파일 시스템이 읽기 전용으로 마운트되므로 읽기/쓰기 모드로 다시 마운트해야 함
- 시스템의 메시지가 기본적으로 한글로 출력되는데 콘솔 모드에서는 한글이 출력되지 않기 때문에 언어를 영어로 바꿔야 함
- passwd 명령으로 암호를 변경할 수 있음
- root 암호를 수정한 후 exit 명령을 두 번 입력하면 시스템이 다시 부팅됨

```
switch_root:/# mount -o remount,rw /sysroot → /sysroot를 rw 모드로 다시 마운트한다.
switch_root:/# chroot /sysroot → /sysroot 디렉터리를 root 디렉터리로 설정한다.
sh-4.4# LANG=en_US.UTF-8 → 언어 변경
sh-4.4# passwd → 암호 재설정
Changing password for user root.
New password:
Retype new password:
passwd: all authentication tokens updated succesfully.
sh-4.4# touch /.autorelabel → SELinux enable
sh-4.4# exit
exit
switch_root:/# exit → 재부팅
```

# Thank you!