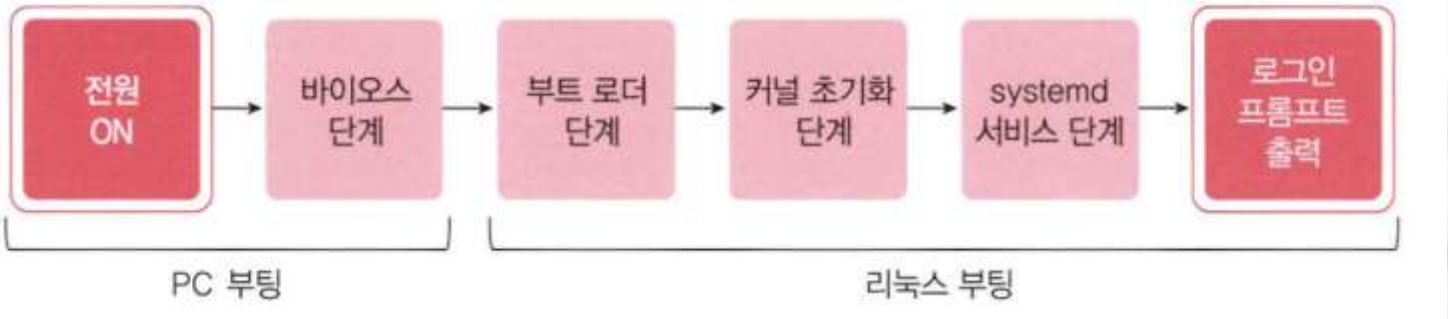
# 리눅스 부팅 시스템

# 리눅스 시스템의 부팅

❖ 리눅스의 부팅

✓ 개요

- 리눅스의 부팅은 PC 전원을 켜는 순간부터 리눅스가 완전히 동작하여 로그인 프롬프트가 출력될 때까지를 의미
- 시스템 관리자는 리눅스의 부팅 과정을 이해하고 부팅에 필요한 서비스가 시작되도록 설정해야 하며 부팅 과정에서 문제가 발생할 경우 이를 해결해야 함
- 리눅스 시스템의 부팅 과정은 크게 PC 부팅과 리눅스 부팅으로 나누는데 리눅스가 설치 된 하드웨어 부팅과 리눅스 운영체제의 부팅 절차로 구분할 있음
- 부팅 과정



# 리눅스 시스템의 부팅

- ❖ 리눅스의 부팅 과정
  - ✓ 바이오스 단계
    - PC 전원 스위치를 켜면 제일 먼저 BIOS(basic input/output system)가 동작
    - 바이오스는 보통 ROM 저장되어 있어 흔히 ROM-BIOS라고 부름
    - 바이오스는 PC에 장착된 기본적인 하드웨어 키보드, 디스크 상태를 확인한 부팅 장치를 선택하여 부팅 디스크의 섹터에서 512B 로딩하는데 512B를 마스터 부트 레코드(master boot record. MBR)라고 하며 여기에는 디스크의 어느 파티션에 2차 부팅 프로그램(부트 로더)이 있는지에 대한 정보가 저장되어 있으며 메모리에 로딩된 MBR 부트 로더를 찾아 메모리에 로딩하는 작업까지 수행


바이오스 단계

# 리눅스 시스템의 부팅

- 리눅스의 부팅 과정
  - ✓ 부트 로더 단계
    - 바이오스 단계에서 MBR 부트 로더를 찾아 메모리에 로딩
    - 부트 로더는 일반적으로 운영체제 중에서 부팅할 운영체제를 선택할 있도록 메뉴를 제공
    - 우분투에서는 부트 로더로 GRUB를 사용하는데 기본적으로 멀티 부팅이 아닐 경우 GRUB 메뉴를 출력하지 않고 바로 부팅 작업을 진행
    - 부팅할 GRUB 메뉴를 출력하려면 /etc/default/grub 파일을 수정
      - ◆ 편집기로 /etc/default/grub 파일을 열고 GRUB_TIMEOUT_STYLE=hidden 앞에 # 추가하고 GRUB_TIMEOUT=0 에서 0을 10으로 수정하는데 이 값은 GRUB 메뉴가 출력되는 시간으로 시간이 지나면 부팅 과정이 자동으로 진행됨
      - ◆ sudo update-grub 명령 수행
    - 부트 로더는 리눅스 커널을 메모리에 로딩하는 역할을 수행
      - ◆ 리눅스 커널은 /boot 디렉토리 아래에 vmlinuz-버전명의 형태로 제공
      - ◆ 우분투를 처음 설치한 후 업데이트를 하면 커널이 추가로 생성
      - ◆ ls /boot/vm*

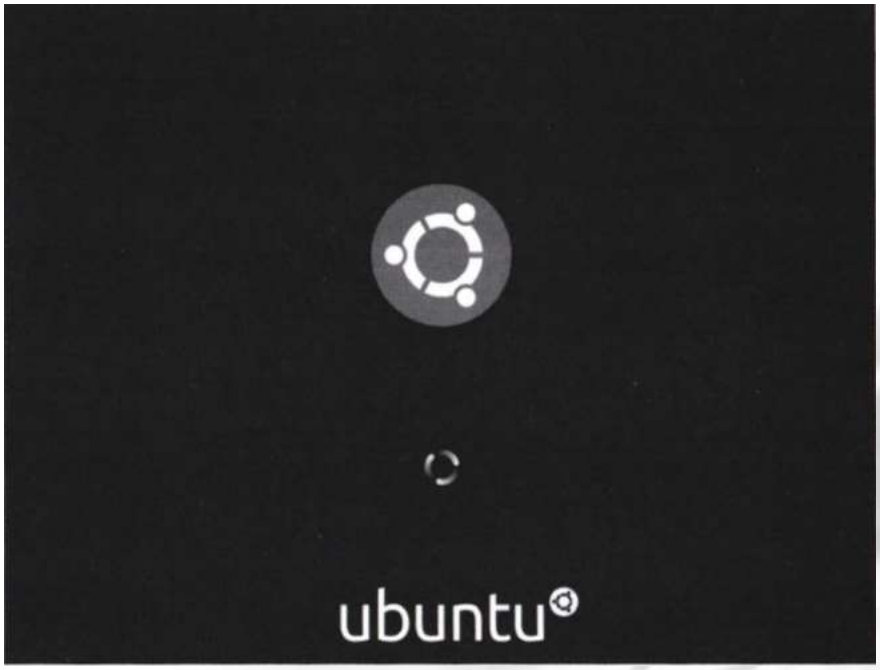
      /boot/vmlinuz  /boot/vmlinuz-6.14.0-15-generic  /boot/vmlinuz.old

# 리눅스 시스템의 부팅

- ❖ 리눅스의 부팅 과정
  - ✓ 커널 초기화 단계
    - 부트 로더에 의해 메모리에 로딩된 커널은 가장 먼저 시스템에 연결된 메모리, 디스크, 키보드, 마우스 등의 장치를 검사
    - 리눅스를 처음 시스템에 설치할 때 사용 가능한 하드웨어의 정보를 미리 확인했다가 부팅할 때 이러한 장치들이 사용 가능한 상태로 유지되고 있는지 확인
    - 장치 검사 등 기본적인 초기화 과정이 끝나면 커널은 일반적으로 프로세스를 만드는 방식인 fork를 사용하지 않고 프로세스와 스레드를 생성하는데 이 프로세스들은 메모리 관리 같은 커널의 여러 가지 동작을 수행
    - 커널 프로세스의 개수와 종류는 리눅스의 버전과 종류에 따라 다름
    - 이 프로세스들은 일반적인 프로세스와 구분되도록 대괄호([ ])로 표시되며 주로 PID 번호가 낮게 배정되어 있음
    - ps 명령으로 확인

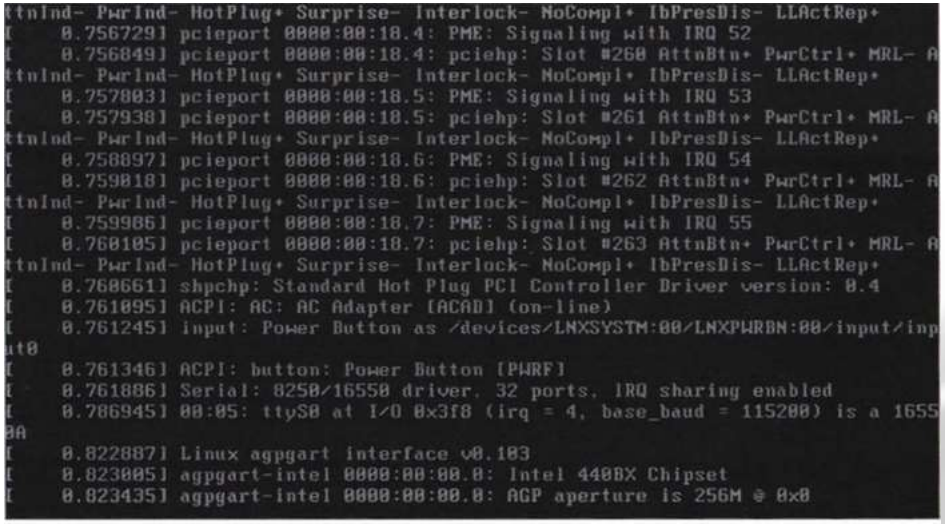
# 리눅스 시스템의 부팅

- 리눅스의 부팅 과정
  - ✓ systemd 서비스 단계
    - systemd 서비스 단계에 이르면 리눅스가 본격적 동작하기 시작한 것
    - 우분투에서 systemd 서비스는 기존의 init 스크립트를 대체한 것으로 다양한 서비스를 동작 시킴
    - 우분투에서는 하드디스크를 점검하는 부팅 메시지를 몇 줄 출력한 후 이미지를 출력
    - 부트 스플래시라고 하는 이 이미지는 리눅스 배포판과 버전에 따라 다르게 출력



# 리눅스 시스템의 부팅

- 리눅스의 부팅 과정
  - ✓ systemd 서비스 단계
    - 부팅할 때 메시지가 출력되게 하려면 /etc/default/grub 파일의 GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"에서 quiet를 삭제하고 sudo update-grub를 실행하여 변경된 내용을 적용
    - 부팅 메시지가 출력되는데 각종 서비스가 정상적으로 시작되는지(OK) 아니면 실패인지(FAIL)를 나타냄



# 리눅스 시스템의 부팅

- ❖ 리눅스의 부팅 과정
  - ✓ systemd 서비스 단계
    - 메시지는 부팅 후 dmesg 명령이나 more /var/log/bootstrap.log 명령으로 확인할 수 있음
    - dmesg 명령으로 출력되는 메시지에는 데몬의 시작과 관련된 것 뿐 아니라 하드웨어 검사와 관련된 것도 모두 포함되어 있음

```
[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x610f0000]
[    0.000000] Linux version 6.14.0-15-generic (buildd@bos03-arm64-008) (aarch64
-linux-gnu-gcc-14 (Ubuntu 14.2.0-19ubuntu2) 14.2.0, GNU ld (GNU Binutils for Ubu
ntu) 2.44) #15-Ubuntu SMP PREEMPT_DYNAMIC Sun Apr  6 14:37:51 UTC 2025 (Ubuntu 6
.14.0-15.15-generic 6.14.0)
[    0.000000] KASLR disabled due to lack of seed
[    0.000000] Machine model: linux,dummy-virt
[    0.000000] efi: EFI v2.7 by EDK II
[    0.000000] efi: MEMATTR=0xba518018 MOKvar=0xb82d0000 INITRD=0xb82e6a18 MEMRE
SERVE=0xb82e6398
[    0.000000] secureboot: Secure boot disabled
[    0.000000] OF: reserved mem: Reserved memory: No reserved-memory node in the
 DT
[    0.000000] NUMA: Faking a node at [mem 0x0000000040000000-0x00000000bfffffff
]
[    0.000000] NODE_DATA(0) allocated [mem 0xbfc08280-0xbfc0dcbf]
[    0.000000] Zone ranges:
[    0.000000]   DMA      [mem 0x0000000040000000-0x00000000bfffffff]
[    0.000000]   DMA32    empty
[    0.000000]   Normal   empty
[    0.000000]   Device   empty
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
--More--
```

# 리눅스 시스템의 부팅

- 리눅스의 부팅 과정
  - ✓ systemd 서비스 단계
    - 유닉스에서는 init 프로세스가 서비스를 실행했는데 따라서 init 프로세스는 처음 생성된 프로세스로서 PID가 1번
    - 우분투는 15.04 버전부터 init 대신 시스템과 서비스 관리자로 systemd를 사용하기 시작
    - ps -ef | more

```
root         1     0  0 09:23 ?        00:00:02 /sbin/init
root         2     0  0 09:23 ?        00:00:00 [kthreadd]
root         3     2  0 09:23 ?        00:00:00 [pool_workqueue_release]
root         4     2  0 09:23 ?        00:00:00 [kworker/R-rcu_gp]
root         5     2  0 09:23 ?        00:00:00 [kworker/R-sync_wq]
root         6     2  0 09:23 ?        00:00:00 [kworker/R-kvfree_rcu_reclai
m]
```

    - ls -l /sbin/init

```
lrwxrwxrwx 1 root root 22  3월 28 15:48 /sbin/init -> ../lib/systemd/system
```

    - systemd 서비스 단계에서는 데몬을 모두 실행한 뒤 로그인 프롬프트 화면을 출력

# systemd service

- 개요
  - ✓ systemd는 리눅스의 시스템의 서비스 관리자로서 유닉스의 init 프로세스가 하던 작업을 대신 수행
  - ✓ systemd는 다양한 서비스 데몬을 시작하고 프로세스들의 상태를 유지하며 시스템의 상태를 관리

# systemd service

- init 프로세스
  - 기존에는 init 프로세스가 스크립트를 순차적으로 실행하여 다른 프로세스를 동작시켰는데 현재 init 서비스는 systemd 서비스로 대체됨
  - man init로 확인해보면 systemd에 대한 설명이 출력됨
  - man init

```
SYSTEMD(1)                    systemd                    SYSTEMD(1)
NAME
       systemd, init - systemd system and service manager
SYNOPSIS
       /usr/lib/systemd/systemd [OPTIONS...]
       init [OPTIONS...] {COMMAND}
DESCRIPTION
       systemd is a system and service manager for Linux operating systems.
       When run as first process on boot (as PID 1), it acts as init system
       that brings up and maintains userspace services. Separate instances are
       started for logged-in users to start their services.
       systemd is usually not invoked directly by the user, but is installed as
       the /sbin/init symlink and started during early boot. The user manager
       instances are started automatically through the user@.service(5)
       service.
       For compatibility with SysV, if the binary is called as init and is not
 Manual page init(1) line 1 (press h for help or q to quit)
```

# systemd service

- init 프로세스
  - ✓ init 과 관련된 스크립트 파일은 /etc/init.d 디렉토리에 있으며 아직 일부 서비스의 스크립트 파일이 남아 있음
  - ✓ init 스크립트 방식의 서비스와 systemd 서비스가 아직 공존하는 과도기
  - ✓ ls /etc/init.d

alsa-utils cups open-vm-tools spice-vdagent
anacron dbus openvpn ssh
apparmor gdm3 plymouth sssd
apport grub-common plymouth-log sysstat
bluetooth iscsid procps ufw
console-setup.sh kdump-tools rsync unattended-upgrades
cron keyboard-setup.sh saned uuidd
cryptdisks kmod screen-cleanup whoopsie
cryptdisks-early open-iscsi speech-dispatcher x11-common

# systemd service

- systemd는 init 방식에 비해 다음과 같은 장점이 있음
  - ✓ 소켓 기반으로 동작하여 inetd와 호환성을 유지
  - ✓ 셸과 독립적으로 부팅이 가능
  - ✓ 마운트 제어가 가능
  - ✓ fsck(file system check) 제어가 가능
  - ✓ 시스템 상태에 대한 스냅숏을 유지
  - ✓ 서비스에 시그널을 전달할 수 있음
  - ✓ 셧다운 전에 사용자 세션의 안전한 종료가 가능

# systemd service

- systemd unit
  - ✓ systemd는 전체 시스템을 시작하고 관리하는 데 유닛이라 부르는 구성 요소를 사용
  - ✓ systemd는 관리 대상의 이름을 서비스명.유닛 종류 의 형태로 관리
  - ✓ 각 유닛은 같은 이름과 종류로 구성된 설정 파일과 동일한 이름을 사용
  - ✓ 종류
    - service: 시스템 서비스 유닛으로 데몬을 시작, 종료, 재시작 및 로드
    - target: 유닛을 그루핑
    - automount: 디렉토리 계층 구조에서 자동 마운트 포인트를 관리
    - device: 리눅스 장치 트리에 있는 장치를 관리
    - mount: 디렉토리 계층 구조의 마운트 포인트를 관리
    - path: 파일 사스템의 파일이나 디렉토리 등 경로를 관리
    - scope: 외부에서 생성된 프로세스를 관리
    - slice: 시스템의 프로세스를 계층적으로 관리
    - socket: 소켓을 관리하는 유닛으로 AF_INET, AF_INET6, AF_UNIX Socket Stream, Datagram, FIFO 지원
    - swap: 스왑 장치 관리
    - timer: 타이머 와 관련된 기능을 관리

# systemd service

- systemctl 명령
  - ✓ 형식: systemctl [옵션] [명령] [유닛명]
  - ✓ 옵션
    - -a: 상태와 관계없이 유닛 전체를 출력
    - -t 유닛 종류: 지정한 종류의 유닛만 출력
  - ✓ 명령
    - start: 유닛을 시작
    - stop: 유닛을 정지
    - reload 유닛의 설정 파일을 다시 읽어옴
    - restart: 유닛을 재시작
    - status: 유닛 상태를 출력
    - enable: 부팅 시 유닛이 시작되도록 설정
    - disable: 부팅 시 유닛이 시작하지 않도록 설정
    - is-active: 유닛이 동작하고 있는지 확인
    - is-enabled: 유닛이 시작되었는지 확인
    - isolate: 지정한 유닛 및 이와 관련된 유닛만 시작하고 나머지는 정지
    - kill :유닛에 시그널을 전송

# systemd service

- systemctl 명령
  - ✓ 동작 중인 유닛 출력 – 명령만 사용

```
user1@myubuntu:~$ systemctl
  UNIT                      LOAD   ACTIVE SUB       DESCRIPTION
(생략)
  basic.target              loaded active active    Basic System
  bluetooth.target          loaded active active    Bluetooth
  cryptsetup.target         loaded active active    Local Encrypted Volumes
  getty.target              loaded active active    Login Prompts
  graphical.target          loaded active active    Graphical Interface
  local-fs-pre.target       loaded active active    Local File Systems (Pre)
  local-fs.target           loaded active active    Local File Systems
  multi-user.target         loaded active active    Multi-User System
  network-online.target     loaded active active    Network is Online
  network.target            loaded active active    Network
(생략)
LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
229 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

# systemd service

- systemctl 명령
  - ✓ 전체 유닛 출력

```
user1@myubuntu:~$ systemctl -a
  UNIT                          LOAD      ACTIVE   SUB       DESCRIPTION
(생략)
  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-sys-fs-binfmt_misc.mount loaded    inactive dead      Arbitrary Executable
File Formats File System
  run-vmblock\x2dfuse.mount     loaded    active   mounted   VMware vmblock fuse
mount
  snap-bare-5.mount             loaded    active   mounted   Mount unit for bare, revision 5
  snap-core-11743.mount loaded    active   mounted   Mount unit for core, revision
11743
(생략)
```

# systemd service

- systemctl 명령
  - ✓ 특정 유닛 출력

```
user1@myubuntu:~$ systemctl -t service
UNIT                          LOAD   ACTIVE SUB     DESCRIPTION
  accounts-daemon.service     loaded active running Accounts Service
  acpid.service               loaded active running ACPI event daemon
  alsa-restore.service        loaded active exited  Save/Restore Sound Card State
  apparmor.service            loaded active exited  Load AppArmor profiles
  apport.service              loaded active exited  LSB: automatic crash report
generation
  atd.service                 loaded active running Deferred execution scheduler
  avahi-daemon.service        loaded active running Avahi mDNS/DNS-SD Stack
(생략)
```

# systemd service

- systemctl 명령
  - ✓ 유닛 서비스 시작

```
adam@adam:~$ sudo systemctl start cron
[sudo] password for adam:
adam@adam:~$ systemctl is-active cron
active
```

# systemd service

- systemctl 명령
  - ✓ 유닛 서비스 상태 확인

```
adam@adam:~$ systemctl status cron.service
cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2023-09-29 01:36:48 UTC; 1h 12min ago
       Docs: man:cron(8)
   Main PID: 811 (cron)
      Tasks: 1 (limit: 4524)
     Memory: 428.0K
        CPU: 48ms
     CGroup: /system.slice/cron.service
             └─811 /usr/sbin/cron -f -P

 9월 29 01:36:48 adam systemd[1]: Started Regular background program processing
daemon.
 9월 29 01:36:48 adam cron[811]: (CRON) INFO (pidfile fd = 3)
 9월 29 01:36:48 adam cron[811]: (CRON) INFO (Running @reboot jobs)
```

# systemd service

- systemctl 명령
  - ✓ 유닛 서비스 중지

```
adam@adam:~$ systemctl status stop cron
adam@adam:~$ systemctl status cron

cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
     Active: inactive (dead) since Fri 2023-09-29 02:50:22 UTC; 9s ago
       Docs: man:cron(8)
    Process: 811 ExecStart=/usr/sbin/cron -f -P $EXTRA_OPTS (code=killed, signal=TERM)
   Main PID: 811 (code=killed, signal=TERM)
        CPU: 49ms

 9월 29 01:36:48 adam systemd[1]: Started Regular background program processing
daemon.
 9월 29 01:36:48 adam cron[811]: (CRON) INFO (pidfile fd = 3)
 9월 29 01:36:48 adam cron[811]: (CRON) INFO (Running @reboot jobs)
 9월 29 02:17:01 adam CRON[5230]: pam_unix(cron:session): session opened for
```

# systemd service

- 실습
  - ✓ colord.service가 동작 중인지 is-active 명령으로 확인
  - ✓ colord.service 를 중지
  - ✓ colord.service의 상태를 확인
  - ✓ colord.service를 다시 시작
  - ✓ colord.service의 상태에서 PID를 확인

# systemd service

- systemd 와 Run Level
  - ✓ Run Level은 현재 시스템의 상태를 나타내는 한 자리 숫자(문자 S, s 포함)
  - ✓ systemd의 target 유닛은 /usr/lib/systemd/system 디렉토리에 존재
  - ✓ Run Level에 해당하는 runlevelX.target이 제공되는데 이는 Run Level에 익숙한 사용자의 편의를 위한 심볼릭 링크
  - ✓ Run Level 과 target unit의 관계

| Run Level | Target File(Symbolic Link) | Target Original File |
|---|---|---|
| 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 |

# systemd service

- systemd 와 Run Level
  - ✓ 현재 target 확인: systemctl get-default

    graphical.target
  - ✓ 현재 Run Level 확인: runlevel

    N 5
  - ✓ 기본 target 지정
    - 부팅할 때 동작하는 기본 Run Level은 예전에는 /etc/inittab 파일에 지정했으나 지금은 default.target이 가리키는 target 유닛으로 변경됨
    - default.target이 가리키는 기본 target은 다음과 같은 형식으로 지정

      systemctl set-default <name of target>.target
    - 이 명령은 /etc/systemd/system 디렉토리 아래의 심볼릭 링크인 default.target이 가리키는 target 파일을 변경
    - 현재 target인 graphical.target에서 multi-user.target 으로 변경: sudo systemctl set-default multi-user.target

      created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/multi-user.target
    - 확인: ls -l /etc/systemd/system/default.target

      lrwxrwxrwx 1 root root 41 Sep 2 00:18 /etc/systemd/system/default.target -> /usr/lib/systemd/system/multi-user.target

# systemd service

- systemd 와 Run Level
  - ✓ Run Level 0 이나 Run Level 6에 해당하는 target을 기본으로 지정하면 안됨
  - ✓ 기본 target은 graphical.target(Run Level 5)으로 해놓는 것이 좋음
  - ✓ 기본 target을 다시 graphical.target 으로 변경: sudo systemctl set-default graphical.target

    Removed "/etc/systemd/system/default.target".

    Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/graphical.target.
  - ✓ systemd에서 Run Level 변경
    - systemctl isolate multi-user
    - systemctl isolate runlevel3

# systemd service

- systemd 와 Run Level
  - ✓ 단일 사용자 모드로 전환
    - 시스템에 문제가 있을 경우 시스템을 rescue.target 유닛(Run Level 1, Run Level S)으로 변경하여 점검해야 함
    - 다중 사용자 모드에서 시스템 관리자만 사용할 수 있는 단일 사용자 모드로 전환하는 것
    - 이 모드로 변환하기 전에 다른 사용자들은 로그아웃을 해야 함
    - 다음 명령 중 하나를 사용하면 단일 사용자 모드로 전환
      - systemctl isolate rescue
      - systemctl isolate runlevel1
      - init 1
      - telinit S
    - 단일 사용자 모드로 전환하면 그래픽 환경이었던 리눅스가 텍스트 모드로 변경됨
    - 단일 사용자 모드에서는 바로 root 암호를 입력하여 시스템과 관련된 점검 작업을 하고 다시 다중 사용자 모드로 전환
    - 단일 사용자 모드에서 다중 사용자 모드로 전환하려면 reboot 명령이나 systemctl default 명령을 사용

# systemd service

- 서비스 등록
  - ✓ Ubuntu 서비스 등록은 systemd 서비스를 이용하는 것이 일반적
  - ✓ .service 파일을 생성하고 systemctl daemon-reload, systemctl enable 명령어를 실행하여 서비스를 등록하고 활성화시킴
  - ✓ 서비스 파일 생성
    - 파일 경로: /etc/systemd/system/ 디렉토리에 .service 파일을 생성
    - 파일 내용: 다음과 같은 기본적인 구조를 따름

```
[Unit]
Description=My custom service
After=network.target

[Service]
ExecStart=/path/to/your/script/myservice.sh
Restart=on-failure

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

# systemd service

- ❖ 서비스 등록
  - ✓ 서비스 파일 내용
    - [Unit]
      - ◆ 서비스의 메타데이터와 의존성을 정의
      - ◆ Description으로 서비스 설명을 After로 서비스 시작 전에 네트워크가 준비되어야 함을 명시할 수 있음
    - [Service]
      - ◆ 서비스 실행 방법 및 속성을 정의
      - ◆ ExecStart: 서비스 시작 시 실행될 스크립트나 프로그램 경로를 지정
      - ◆ Restart: 서비스가 실패했을 때 재시작하는 정책을 설정
    - [Install]
      - ◆ 서비스가 부팅 시 자동으로 시작되도록 하는 부분을 정의
      - ◆ WantedBy는 서비스가 어떤 상태에서 시작되어야 하는지를 설정

# systemd service

- ❖ 서비스 등록
  - ✓ 서비스 등록 및 활성화
    - 서비스 파일을 생성한 후 systemd에게 새로운 서비스를 인식시키고 활성화
      - ◆ sudo systemctl daemon-reload: 새로운 서비스 파일을 읽어들이도록 systemd 데몬을 재시작
    - 서비스 관리
      - ◆ 서비스가 잘 등록되고 실행되는지 확인하고, 필요에 따라 시작, 중지, 상태 확인을 할 수 있음
      - ◆ 서비스가 부팅 시 자동으로 시작되도록 활성화 : sudo systemctl enable myservice.service:
      - ◆ 시작: sudo systemctl start myservice.service
      - ◆ 중지: sudo systemctl stop myservice.service
      - ◆ 재시작: sudo systemctl restart myservice.service
      - ◆ 상태 확인: sudo systemctl status myservice.service

# 종료

❖ 개요

- ✓ 시스템을 종료할 때 정상적인 절차를 거치는 것이 매우 중요
- ✓ 리눅스는 대부분 서버 운영체제로 사용되기 때문에 비정상적으로 시스템을 종료하여 문제가 발생하면 서비스를 제공하지 못할 수도 있음
- ✓ 종료 방법
  - shutdown 명령을 사용
  - halt 명령을 사용
  - poweroff 명령을 사용
  - Run Level을 0 이나 6 으로 전환
  - reboot 명령을 사용
  - 전원 끄기

# 종료

- shutdown 명령
  - ✓ 리눅스 시스템을 가장 정상적으로 종료하는 방법은 shutdown 명령을 사용하는 것
  - ✓ 시스템을 종료하는 다른 명령들과 달리 shutdown 명령은 다양한 종료 방법을 제공
  - ✓ 시스템 종료 외에 Run Level을 바꿀 때도 사용할 수 있음
  - ✓ 사용 방식
    - 형식: shutdown [옵션] [시간] [메시지]
    - 옵션
      - -k: 실제로 시스템을 종료하는 것이 아니라 사용자들에게 메시지만 전달
      - -r: 종료한 후 재시작
      - -h: 종료하고 halt 상태로 이동
      - -f: 빠른 재시작으로 이 과정에서 fsck를 생략할 수 있음
      - -c: 이전에 내렸던 shutdown 명령을 취소
    - 시간:  종료할 시간(hh:mm, now)
    - 메시지: 모든 사용자에게 보낼 메시지

# 종료

- shutdown 명령
  - ✓ 즉시 종료
    - shutdown 명령으로 시스템을 즉시 종료하려면 -h 옵션과 함께 현재 시간(now)을 지정
    - sudo shutdown -h now
    - 종료 시간이 now이므로 바로 시스템 종료 절차가 시작됨
  - ✓ 메시지 보내고 종료
    - 현재 시스템을 사용 중인 사용자들이 있다면 시스템이 종료된다는 메시지를 보내어 작업을 저장하고 정리할 시간을 주어야 함
    - 시스템을 종료할 때 shutdown 명령으로 메시지를 보낼 수 있음
    - 사용자들이 메시지를 받고 정리할 시간이 필요하므로 시간을 now로 지정하면 안되고 특정 시간을 지정
    - 2분 후에 시스템이 종료되도록 설정하고 메시지를 보내려면 다음과 같이 지정

      user1@myubuntu : ~$ sudo shutdown -h +2 "System is going down in 2 min"

# 종료

- shutdown 명령
  - ✓ 시스템 재시작
    - shutdown 명령을 이용하고자 하는 경우는 -r 옵션을 사용
    - 3분 후에 시스템을 재시작하도록 지정

      user1@myubuntu : ~$ sudo shutdown -r +3
  - ✓ 명령 취소
    - shutdown 명령을 취소하려면 -c 옵션을 사용

      user1@myubuntu : ~$ sudo shutdown -c
  - ✓ 메시지만 보내기
    - shutdown 명령을 실행하지는 않고 사용자들에게 메시지만 보내려면 -k 옵션을 사용

      user1@myubuntu : ~$ sudo shutdown -k 2

# 종료

- Run Level 변경
  - ✓ 시스템을 종료하는 다른 방법은 Run Level을 변경하는 것
  - ✓ telinit 명령으로 Run Level을 0 으로 변경하면 시스템이 종료되고 6 으로 변경하면 재시작
  - ✓ Run Level 변경
    - 종료: sudo init 0
    - 재시작: sudo init 6
  - ✓ systemd 기능을 사용할 때는 target을 변경하면 됨
    - 종료

      sudo systemctl isolate poweroff.target

      sudo systemctl isolate runlevel0.target
    - 재시작

      sudo systemctl isolate reboot.target

      sudo systemctl isolate runlevel6.target

# 종료

- 기타 시스템 종료 명령
  - ✓ shutdown 명령과 Run Level 변경 외에 시스템을 종료하거나 재시작하기 위해 사용할 수 있는 명령으로 halt, poweroff, reboot가 있는데 systemctl 명령의 심볼릭 링크

    ls -l /sbin/halt

    ls -l /sbin/poweroff

    ls -1 /sbin/reboot
  - ✓ 모두 systemctl 명령으로 시스템을 종료하고 재시작한다는 의미
  - ✓ 이 명령들은 전통적으로 존재하던 명령으로 계속 유지되고 있는 것
  - ✓ halt, poweroff, reboot 명령은 /var/log/wtmp 파일에 시스템 종료 기록을 남기고 시스템을 종료하거나 재시작
  - ✓ 이 명령들은 Run Level이 1~5일 때 내부적으로 shutdown 명령을 호출
  - ✓ 사용할 수 있는 옵션
    - -n: 재시작이나 종료 전에 sync를 호출하지 않음
    - -w: 실제로 재시작하거나 종료하지는 않지만 wtmp 파일에 기록을 남김
    - -d: wtmp 파일에 기록을 남기지 않는데 -n: 옵션은 -d 옵션을 포함
    - -f: 강제로 명령을 실행하며 shutdown을 호출하지 않음
    - -p: 시스템의 전원 종료

# Daemon Process

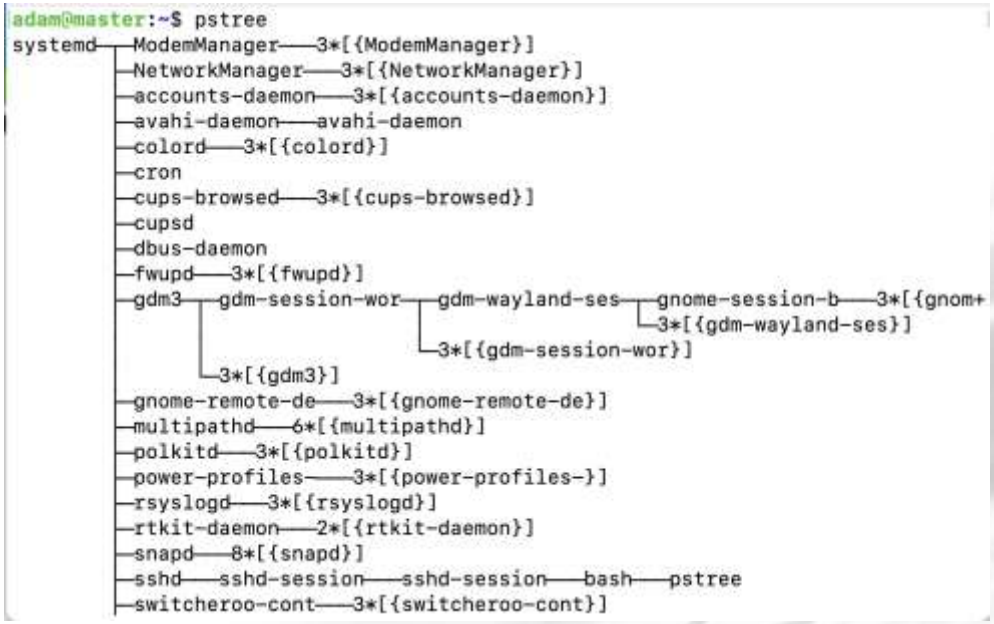
- 개요
  - ✓ 데몬은 리눅스의 백그라운드에서 동작하며 특정한 서비스를 제공하는 프로세스를 의미
  - ✓ 리눅스 시스템에서 동작하는 웹 서버나 데이터베이스 서버, 원격 접속 서버 등 각종 서비스를 제공하는 프로세스가 데몬
  - ✓ 데몬 혼자서 스스로 동작하는 독자형 과 데몬을 관리하는 슈퍼 데몬에 의해 동작하는 방식이 있음
  - ✓ 독자형의 경우 시스템의 백그라운드에서 항상 동작하고 있는데 자주 호출되는 데몬이 아니라면 시스템의 자원을 낭비할 우려가 있음
  - ✓ 슈퍼 데몬에 의한 동작 방식은 평소에는 슈퍼 데몬만 동작하다가 서비스 요청이 오면 슈퍼 데몬이 해당 데몬을 동작시키는 것으로 독자형보다는 서비스에 응답하는 데 시간이 좀 더 걸릴 수 있지만 자원을 효율적으로 사용한다는 장점이 있음
  - ✓ 독자형이든 슈퍼 데몬에 의해 동작하는 형태든 데몬이 제대로 동작하지 않으면 시스템이 서비스를 제공할 수 없음

# Daemon Process

- 슈퍼 데몬
  - ✓ 데몬의 종류가 늘어나자 이를 관리하기 위한 슈퍼 데몬이 등장
  - ✓ 유닉스의 슈퍼 데몬은 이름이 inetd였으나 우분투에서는 보안 기능이 포함된 xinetd를 사용
  - ✓ 슈퍼 데몬은 네트워크 서비스를 제공하는 데몬만 관리
  - ✓ 사용자가 네트워크 서비스를 요청하면 슈퍼 데몬이 이를 받아서 해당하는 서비스 데몬을 동작시키는 것

# Daemon Process

- 데몬의 조상
  - ✓ systemd 데몬
    - systemd는 init를 대체한 데몬
    - systemd는 1번 프로세스로서 프로세스 대부분의 조상 프로세스이며 시스템의 상태를 종합적으로 관리하는 역할을 수행
    - pstree 명령으로 프로세스의 실행 구조를 확인해보면 systemd가 다른 데몬들의 조상임을 더 명확하게 알 수 있음
    - pstree



# Daemon Process

- 데몬의 조상
  - ✓ kernel thread 데몬
    - 커널 일부분을 프로세스처럼 관리하는 데몬이 커널 데몬
    - ps 명령으로 확인했을 때 대괄호([ ])에 들어 있는 프로세스들
    - 예전에는 대부분 k로 시작했으나 요즘에는 이를 반드시 준수하지는 않음
    - 커널 데몬은 대부분 입출력이나 메모리 관리, 디스크 동기화 등을 수행하며 대체로 낮은 PID가 할당되어 있음
    - 일반 프로세스의 조상 데몬이 systemd라면 커널 데몬을 동작시키는 조상 데몬은 kthreadd
    - ps 명령으로 확인해보면 모든 커널 데몬의 PPID가 2번임을 알 수 있음

```
adam@master:~$ ps -ef | more
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 09:23 ?        00:00:02 /sbin/init
root           2       0  0 09:23 ?        00:00:00 [kthreadd]
root           3       2  0 09:23 ?        00:00:00 [pool_workqueue_release]
root           4       2  0 09:23 ?        00:00:00 [kworker/R-rcu_gp]
root           5       2  0 09:23 ?        00:00:00 [kworker/R-sync_wq]
root           6       2  0 09:23 ?        00:00:00 [kworker/R-kvfree_rcu_reclai
m]
```

# Daemon Process

❖ 주요 데몬

| 데몬 | 기능 |
|---|---|
| atd | 특정 시간에 실행되도록 예약한 명령을 실행 |
| cron | 주기적으로 실행되도록 예약한 명령을 실행 |
| dhcpd | 동적으로 IP 주소 부여 |
| httpd | 웹 서비스 제공 |
| lpd | 프린트 서비스 제공 |
| nfs | 네트워크 파일 시스템 서비스 제공 |
| named | DNS 서비스 제공 |
| sendmail | 이메일 서비스 제공 |
| smtpd | 메일 전송 데몬 |
| popd | 기본 편지함 서비스를 제공 |
| routed | 자동 IP 라우터 테이블 서비스를 제공 |
| smb | 삼바 서비스를 제공 |

# Daemon Process

❖ 주요 데몬

| 데몬 | 기능 |
|---|---|
| syslogd | 로그 기록 서비스를 제공 |
| sshd | 원격 보안 서비스를 제공 |
| in.telnetd | 원격 접속 서비스를 제공 |
| ftpd | 파일 송수신 서비스를 제공 |
| ntpd | 시간 동기화 서비스를 제공 |

# Boot Loader

- 개요
  - ✓ 부트 로더는 커널을 메모리에 로딩하는 역할을 수행
  - ✓ 리눅스에는 LILO와 GRUB라는 두 가지 부트 로더가 있는데 우분투에서는 GRUB를 기본으로 지원
  - ✓ GRUB는 grand unified bootloader의 약자로 리눅스의 전통적인 부트 로더인 LILO의 단점을 보완하여 GNU 프로젝트의 일환으로 개발
  - ✓ GRUB는 LILO에 비해 다음과 같은 장점이 있음
    - LILO는 리눅스에서만 사용할 수 있지만 GRUB는 윈도에서도 사용할 수 있음
    - LILO에 비해 설정과 사용이 편리
    - 부팅할 때 명령을 사용하여 수정할 수 있음
    - 멀티 부팅 기능을 지원
  - ✓ GRUB의 최신 버전은 GRUB2 인데 GRUB2는 GRUB를 대체한 것으로 우분투는 GRUB2를 기본 부트 로더로 사용
  - ✓ GRUB에 비해 GRUB2는 이식성과 모듈화가 더 좋아져서 동적으로 모듈을 로딩할 수 있음

# Boot Loader

- GRUB2 관련 디렉토리와 파일
  - ✓ /boot/grub/grub.cfg 파일
    - /boot/grub/grub.cfg 파일은 기존의 menu.lst 파일을 대체하는 기본 설정 파일로 menu.lst 파일과 달리 사용자가 직접 수정할 수 없음
    - 이 파일은 /etc/default/grub 파일과 /etc/grub.d 디렉토리 아래에 있는 스크립트를 읽어서 생성
    - 파일의 내용을 수정하려면 /etc/default/grub 파일과 /etc/grub.d 디렉토리 아래에 있는 스크립트를 수정해야 함
    - 확인: sudo more /boot/grub/grub.cfg
  - ✓ /etc/grub.d 디렉토리
    - 이 디렉토리는 GRUB 스크립트를 가지고 있음
    - 스크립트들은 GRUB의 명령이 실행될 때 순서대로 실행되어 grub.cfg 파일을 생성
    - ls /etc/grub.d
  - ✓ /etc/default/grub 파일
    - 이 파일에는 GRUB 메뉴 설정 내용이 저장되어 있으며 GRUB 스크립트가 이 파일을 읽어서 grub.cfg에 기록
    - menu.lst 파일과 비슷한 역할을 수행
    - cat /etc/default/grub

# Boot Loader

- 암호 복구
  - ✓ 시스템을 재시작해서 GRUB 메뉴 초기 화면을 출력
  - ✓ 메뉴 초기 화면에서 E를 눌러서 편집 모드로 전환한 후 커서를 아래로 내려서 수정
    ro splash $vt_handoff 를 rw init=/bin/bash 로 수정
  - ✓ 재시작
  - ✓ root 계정으로 접속되므로 이 상태에서 원하는 작업을 수행

# Namespaces & Cgroup

- Container
  - ✓ 컨테이너 기술은 애플리케이션을 효율적이고 독립적으로 실행할 수 있는 경량화된 환경을 제공
  - ✓ 컨테이너의 근간이 되는 리눅스 기술
    - Control Group (Cgroup)
    - Namespaces
    - Union Mount Filesystem

# Namespaces & Cgroup

- Control Group
  - ✓ 프로세스들이 사용하는 시스템 자원의 사용 정보를 수집 및 제한시키는 커널의 기능
  - ✓ cgroup 사용 가능한 서브시스템
    - CPU: 스케줄러를 이용해 cgroup에 속한 프로세스의 CPU 사용시간을 제어
    - Memory: cgroup에 속한 프로세스의 메모리 사용량 제어
    - Freezer: cgroup의 작업을 일시 중지하거나 다시 시작
    - blkio: cgroup에 블록 장치에 대한 I/O 제한을 설정
    - net_cls: 트래픽 컨트롤러가 특정 cgroup 적업에서 발생하는 패킷을 식별하게 하는 네트워크 패킷 태그를 지정
    - cpuset:개별 CPU 메모리 노드를 cgroup에 할당
    - cpuacct: cgoup이 사용한 CPU자원 보고서 생성
    - devices: cgroup의 작업 단위로 장치에 대한 엑세스를 허용하거나 거부
    - ns: - namespace 서브시스템

# Namespaces & Cgroup

- Control Group
  - ✓ cgroup은 /sys/fs/cgroup 디렉토리에 가상파일로 존재하고 있으며 해당 파일들에 설정들을 변경해가면서 사용 가능
  - ✓ cgroup으로 cpu 사용 제한
    - /sys/fs/cgroup/cpu 내에 컨트롤 그룹 디렉토리를 만들고 해당 디렉토리에 들어가면 cgroup 관련 각종 값들이 자동으로 생성됨
    - 해당 컨트롤 그룹에 프로세스를 등록하려면 task파일에 프로세스 ID를 등록하면 됨
    - 사용량 제한
      - 50000 100000 → 100ms 중 50ms만 사용 (즉 50%): echo "50000 100000" | sudo tee /sys/fs/cgroup/mygroup/cpu.max
      - 무제한: echo "max 100000" | sudo tee /sys/fs/cgroup/mygroup/cpu.max
    - 프로세스 추가: echo $$ | sudo tee /sys/fs/cgroup/mygroup/cgroup.procs

# Namespaces & Cgroup

- Namespace
  - ✓ Ubuntu에서 namespace는 프로세스를 격리하기 위해 사용하는 Linux 커널 기능
  - ✓ 도커(Docker), LXC 같은 컨테이너 기술의 핵심 기반
  - ✓ 대표적인 네임스페이스
    - PID: 프로세스 ID 공간 분리
    - NET: 네트워크 인터페이스, IP, 라우팅 분리
    - UTS: 호스트 이름 분리
    - MNT: 마운트 포인트/파일시스템 분리
    - IPC: 메시지 큐, 세마포어 같은 IPC 자원 분리
    - USER: UID/GID 분리 (권한 격리)

# Namespaces & Cgroup

- Namespace
  - ✓ 호스트 네임 변경
    - sudo unshare --uts /bin/bash
    - hostname test-namespace
    - hostname
  - ✓ PID 네임스페이스
    - sudo unshare --pid --fork --mount-proc /bin/bash
    - ps aux
  - ✓ NET 네임스페이스
    - sudo ip netns add mynet
    - sudo ip netns exec mynet bash
  - ✓ 여러 네임스페이스 조합
    - sudo unshare --uts --ipc --net --pid --mount --fork /bin/bash

# Namespaces & Cgroup

- ❖ Namespace
  - ✓ 현재 네임스페이스 확인
    - ls -l /proc/$$/ns

```
total 0
lrwxrwxrwx 1 adam adam 0 Sep  2 02:28 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 adam adam 0 Sep  2 02:28 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 adam adam 0 Sep  2 02:28 mnt -> 'mnt:[4026531841]'
lrwxrwxrwx 1 adam adam 0 Sep  2 02:28 net -> 'net:[4026531840]'
lrwxrwxrwx 1 adam adam 0 Sep  2 02:28 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 adam adam 0 Sep  2 03:35 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 adam adam 0 Sep  2 02:28 time -> 'time:[4026531834]'
lrwxrwxrwx 1 adam adam 0 Sep  2 03:35 time_for_children -> 'time:[4026531834]'
lrwxrwxrwx 1 adam adam 0 Sep  2 02:28 user -> 'user:[4026531837]'
lrwxrwxrwx 1 adam adam 0 Sep  2 02:28 uts -> 'uts:[4026531838]'
```