Shell 관리

셸(Shell)

- 사용자 명령어 해석기

  - 사용자가 프롬프트에 입력한 명령을 해석해서 운영체제에 전달

- 셸(Shell)은 커널(Kernel)과 사용자간의 다리역할을 하는 것
- 사용자로부터 명령을 받아 그것을 해석하고 프로그램을 실행하는 역할
- 셸은 사용자가 시스템에 로그인(login)을 하게 되면 각 사용자에게 설정된 셸이 부여되면서 다양한 명령 수행
- 셸을 부여하지 않게 되면 시스템에 로그인하더라도 명령을 수행할 수 없게 되므로 로그인을 막는 효과와 동일

셸(Shell) 확인

- 현재 사용중인 셸 확인 : echo $SHELL
- 사용 가능한 shell 리스트 확인

  ‘**chsh –l**’ 또는 ‘**cat /etc/shells**’

```
[posein@localhost ~]$ echo $SHELL
/bin/bash
[posein@localhost ~]$ chsh -l
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
/bin/tcsh
/bin/csh
[posein@localhost ~]$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
/bin/tcsh
/bin/csh
[posein@localhost ~]$
```

셸(Shell) 변경

- 로그인 셸 명령어 : chsh
  - 암호를 입력한 후 변경하려는 셸을 절대경로로 입력
  - 변경한 셸의 적용은 다음 로그인부터 유효

```
[posein@localhost ~]$ echo $SHELL
/bin/bash
[posein@localhost ~]$ chsh
Changing shell for posein.
New shell [/bin/bash]: /bin/csh
Password:
Shell changed.
[posein@localhost ~]$
```

```
[root@localhost ~]# cat /etc/shells
/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
[root@localhost ~]# echo $SHELL
/bin/bash
```

#cat /etc/shells

#chsh gildong

/bin/sh

#su - gildong

```
[root@localhost ~]# chsh sora
Changing shell for sora.
New shell [/bin/bash]
/bin/sh
Shell changed.
[root@localhost ~]# grep sora /etc/passwd
sora:x:1000:1000:sora:/home/sora:/bin/sh
[root@localhost ~]# su sora
sh-4.4$ pwd
/root
sh-4.4$ exit
exit
[root@localhost ~]# chsh sora
Changing shell for sora.
New shell [/bin/sh]
/bin/bash
Shell changed.
[root@localhost ~]# grep sora /etc/passwd
sora:x:1000:1000:sora:/home/sora:/bin/bash
```

1) 셸 변수

- 특정한 셸(Shell)에서만 적용되는 변수
- 변수명 : 문자, 숫자, _(언더바)로 구성 가능
- 명령행에서 '변수명=값' 형태로 지정
- 변수값을 출력할 때는 변수명 앞에 $을 붙이고 echo명령으로 확인

변수 선언	varname=value	name=gildong score=90
변수 확인	echo, set	echo $name set
변수 제거	unset varname	unset name

```
[root@localhost ~]#
[root@localhost ~]# name=gildong
[root@localhost ~]# score=90
[root@localhost ~]#
[root@localhost ~]# echo $name
gildong
[root@localhost ~]# echo $score
90
[root@localhost ~]# unset name
[root@localhost ~]# unset score
[root@localhost ~]#
[root@localhost ~]# echo $name

[root@localhost ~]# echo $score
```

```
[root@localhost ~]#
[root@localhost ~]# name=gildong
[root@localhost ~]# score=90
[root@localhost ~]#
[root@localhost ~]# echo $name
gildong
[root@localhost ~]# echo $score
90
[root@localhost ~]# set | grep score
score=90
[root@localhost ~]# set | grep name
name=gildong
```

```
[root@localhost ~]# echo $score
90
[root@localhost ~]# unset score
[root@localhost ~]#
[root@localhost ~]# set | grep score
_=score
[root@localhost ~]#
```

2) 환경변수

- 동작되는 프로그램에 영향을 주는 변수
- 프롬프트 변경, PATH 변경 등과 같이 셸의 환경을 정의
- 환경변수는 미리 예약된 변수명 사용
- bash에서는 PATH, SHELL 등과 같이 대문자로 된 변수로 구성되어 있음
- 현재 설정된 전체 환경변수의 값은 env 명령으로 확인 가능
- 특정 환경변수의 값 확인과 설정은 일반 셸 변수 설정과 같음

환경변수 선언	export varname=value	export NAME=gildong echo $NAME
변수 확인	env	env

2) 환경변수

```
[root@localhost /]# env | tail
HISTCONTROL=ignoredups
SHLVL=1
XDG_SEAT=seat0
HOME=/root
LOGNAME=root
LESSOPEN=||/usr/bin/lesspipe.sh %s
DISPLAY=:0
XAUTHORITY=/root/.xautheToDBs
_=/bin/env
OLDPWD=/etc/pam.d
[root@localhost /]#
```

#env | tail

◆ 주요 환경변수

- 배시셸에서는 다양한 환경변수를 제공

변 수	내 용
HOME	사용자의 홈 디렉터리
PATH	실행 파일을 찾는 디렉터리 경로
LANG	셸 사용 시 기본으로 지원되는 언어
TERM	로그인한 터미널 종류
PWD	사용자의 현재 작업 디렉터리
SHELL	사용자의 로그인 셸
USER	사용자의 이름
DISPLAY	X에서 프로그램 실행 시 출력되는 창

PS1	프롬프트(Prompt) 변수
PS2	2차 프롬프트 변수
HISTFILE	히스토리(History) 파일의 절대경로
HISTSIZE	히스토리 파일에 저장되는 명령어의 개수(줄 기준)
HISTFILESIZE	히스토리 파일의 파일 크기
HOSTNAME	시스템의 호스트명
MAIL	도착한 메일이 저장되는 경로
TMOUT	사용자가 로그인한 후 일정시간 동안 작업을 하지 않을 경우에 로그 아웃 시키는 시간으로 단위는 초(second).
UID	사용자의 UID

```
[gildong@localhost ~]$ echo $HOME
/home/gildong
[gildong@localhost ~]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/hom
bin:/home/gildong/bin
[gildong@localhost ~]$ echo $LANG
ko_KR.UTF-8
[gildong@localhost ~]$ echo $TERM
xterm-256color
[gildong@localhost ~]$ echo $USER
gildong
[gildong@localhost ~]$
```

```
[gildong@localhost ~]$ mkdir $HOME/data
[gildong@localhost ~]$
[gildong@localhost ~]$ PATH=$PATH:$HOME/data
[gildong@localhost ~]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/gildong/.loca
bin:/home/gildong/bin:/home/gildong/data
[gildong@localhost ~]$
[gildong@localhost ~]$
```

```
[gildong@localhost ~]$ echo $PS1
[₩u@₩h ₩W]₩$
[gildong@localhost ~]$ PS1="[₩u@₩t]₩$ "
[gildong@12:35:41]$ PS1="[₩u@₩t ₩w]₩$ "
[gildong@12:35:55 ~]$ ^C
[gildong@12:36:03 ~]$ ^C
[gildong@12:36:15 ~]$ echo $PS1
[₩u@₩t ₩w]$
[gildong@12:36:35 ~]$ PS1="[₩u@₩h ₩W]₩$ "
[gildong@localhost ~]$
```

PS1="[\u@\t]\$ "
PS1="[\u@\t \w]\$ “
PS1=“[\u@\h \W]\$”
echo $PS1ls

```
[gildong@localhost ~]$ alias l='ls -il'
[gildong@localhost ~]$ l
합계 20
139501616 -rw-r--r--. 1 root root 19 10월 23 20:49 1.txt
139501618 -rw-r--r--. 1 root root  0 10월 23 20:49 1.tzt
[gildong@localhost ~]$
[gildong@localhost ~]$ unalias l
[gildong@localhost ~]$
```

```
[gildong@localhost ~]$ cat .bash_history
exit
/bin/csh
clear
exit
chsh -s /bin/bash
exit
[gildong@localhost ~]$ echo $HITSIZE

[gildong@localhost ~]$ echo $HISTSIZE
1000
[gildong@localhost ~]$ HISTSIZE=10
[gildong@localhost ~]$ echo $HISTSIZE
10
```

◆ 환경변수의 사용 및 변경

- 환경변수를 사용하여 각 사용자 고유의 셸 환경을 구축
- 명령어와 결합하여 이용 가능

[사용 예] $ mkdir $HOME/data → 해당 사용자 홈디렉터리안에 data라는 디렉터리를 만듦

$PATH=$PATH:$HOME/data → PATH 변수에 $HOME/data라는 경로를 추가

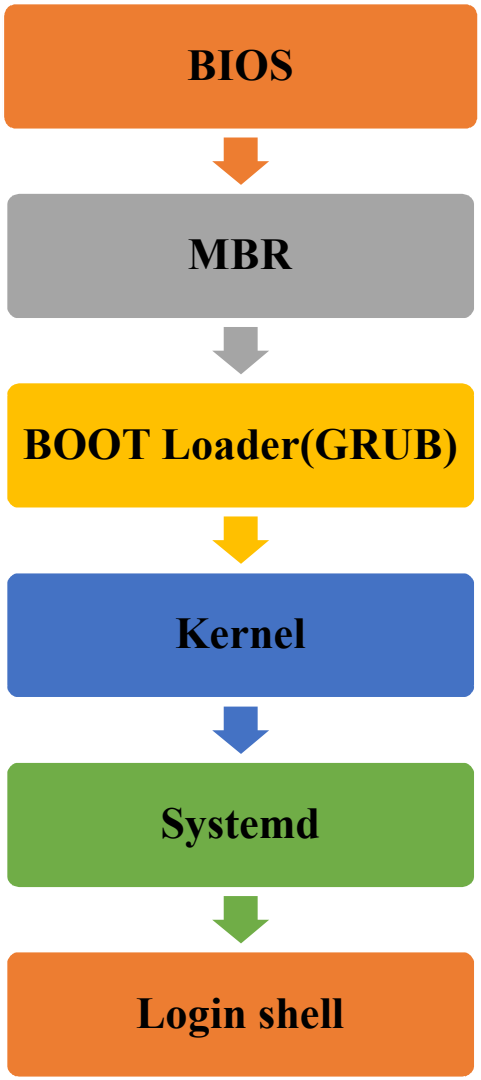
```
[posein@localhost ~]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/posein/.
:/home/posein/bin
[posein@localhost ~]$ PATH=$PATH:$HOME/data
[posein@localhost ~]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/posein/.
:/home/posein/bin:/home/posein/data
[posein@localhost ~]$
```

→ 현재 설정된 언어를 확인 후 영어로 변경

→ 설정된 프롬프트를 확인 후 변경 PS1=“[\u@t \W]\$”

PS1=“[\u@t \w]\$”

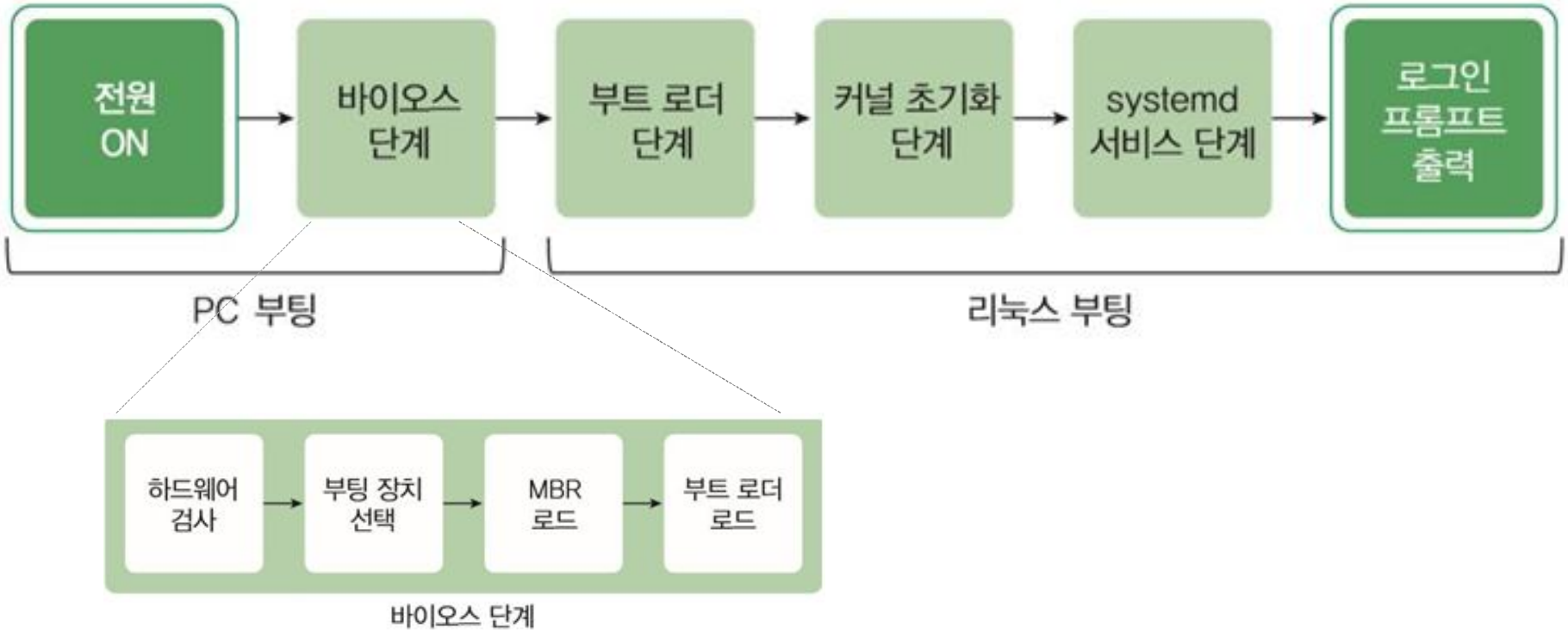
시스템 부팅과 환경설정



- **부팅(booting)**

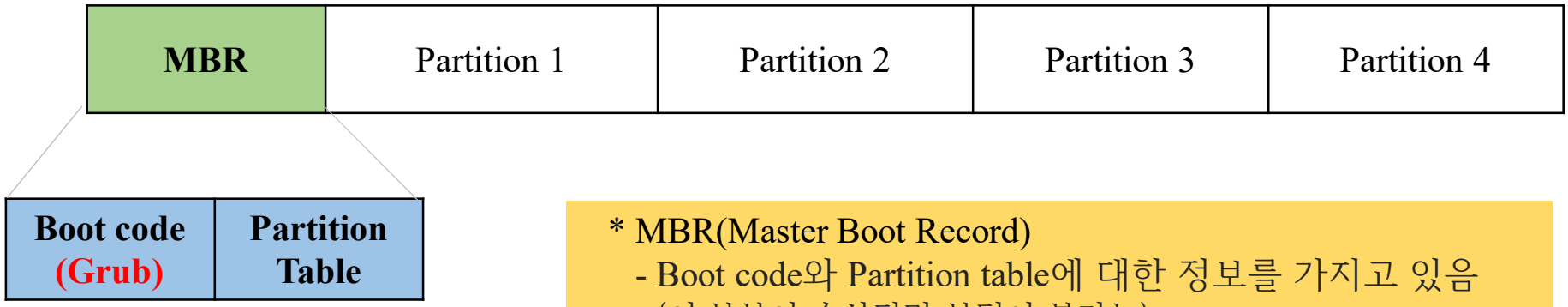
  - PC의 전원을 켜는 순간부터 리눅스가 완전히 동작하여 로그인 프롬프트가 출력될 때까지 임

  - PC 부팅과 리눅스 부팅으로 나뉨


전원
ON
바이오스
단계
부트 로더
단계
커널 초기화
단계
systemd
서비스 단계
로그인
프롬프트
출력
PC 부팅
리눅스 부팅
하드웨어
검사
부팅 장치
선택
MBR
로드
부트 로더
로드
바이오스 단계

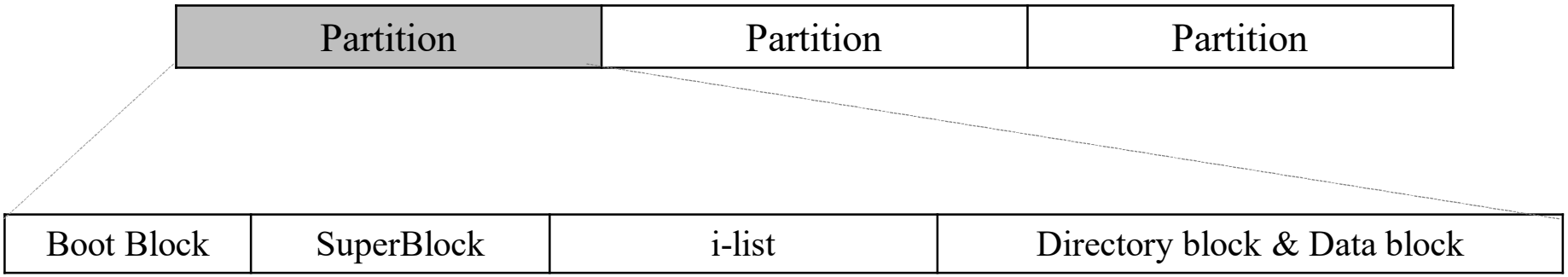
❶ 바이오스(BIOS) 단계

- PC의 전원 스위치를 켜면 제일 먼저 동작
- 보통 ROM에 저장되어 있어 흔히 ROM-BIOS라고 부름
- PC에 장착된 기본적인 하드웨어(키보드, 디스크 등)의 상태를 확인
- 하드디스크의 0번섹터(MBR, Master Boot Record)에 있는 부트로더(GRUB)을 읽어 메모리에 로딩



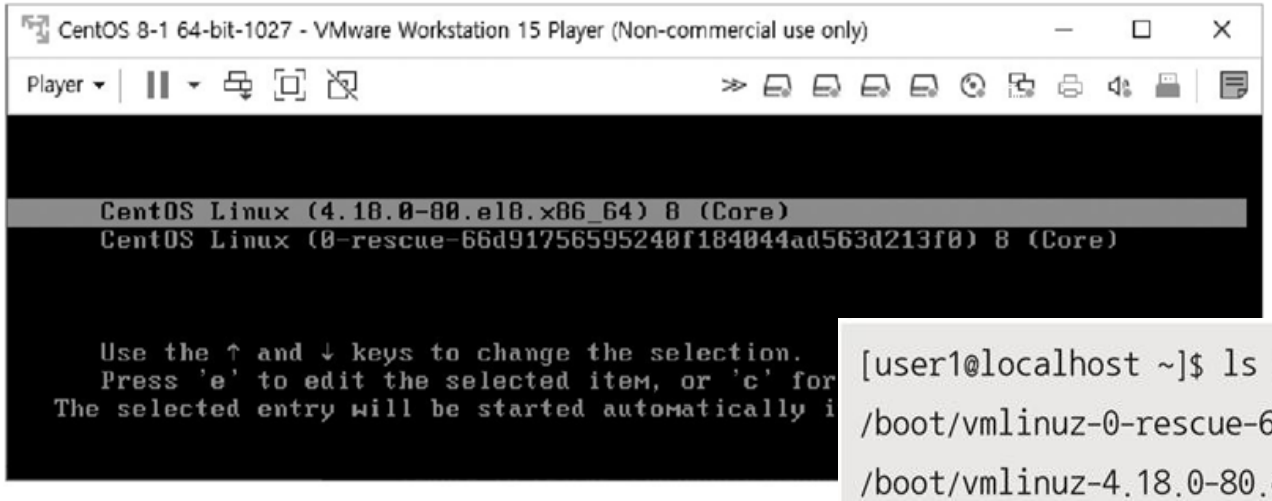
* MBR(Master Boot Record)
- Boot code와 Partition table에 대한 정보를 가지고 있음
  (이 부분이 손상되면 부팅이 불가능)

[참고] Partition 구성



❷ 부트로더(Grub) 단계

- 여러 운영체제 중에서 부팅할 운영체제를 선택할 수 있는 메뉴 제공
- 선택한 운영체제의 커널(리눅스 커널)을 메모리에 로딩하는 역할 수행
- 부트 로더는 /boot 디렉터리 아래에 'vmlinuz-버전명'의 형태로 제공
- Rescue 버전은 응급 상황에서 시스템을 복구하기 위해 사용, 윈도우의 안전모드와 유사



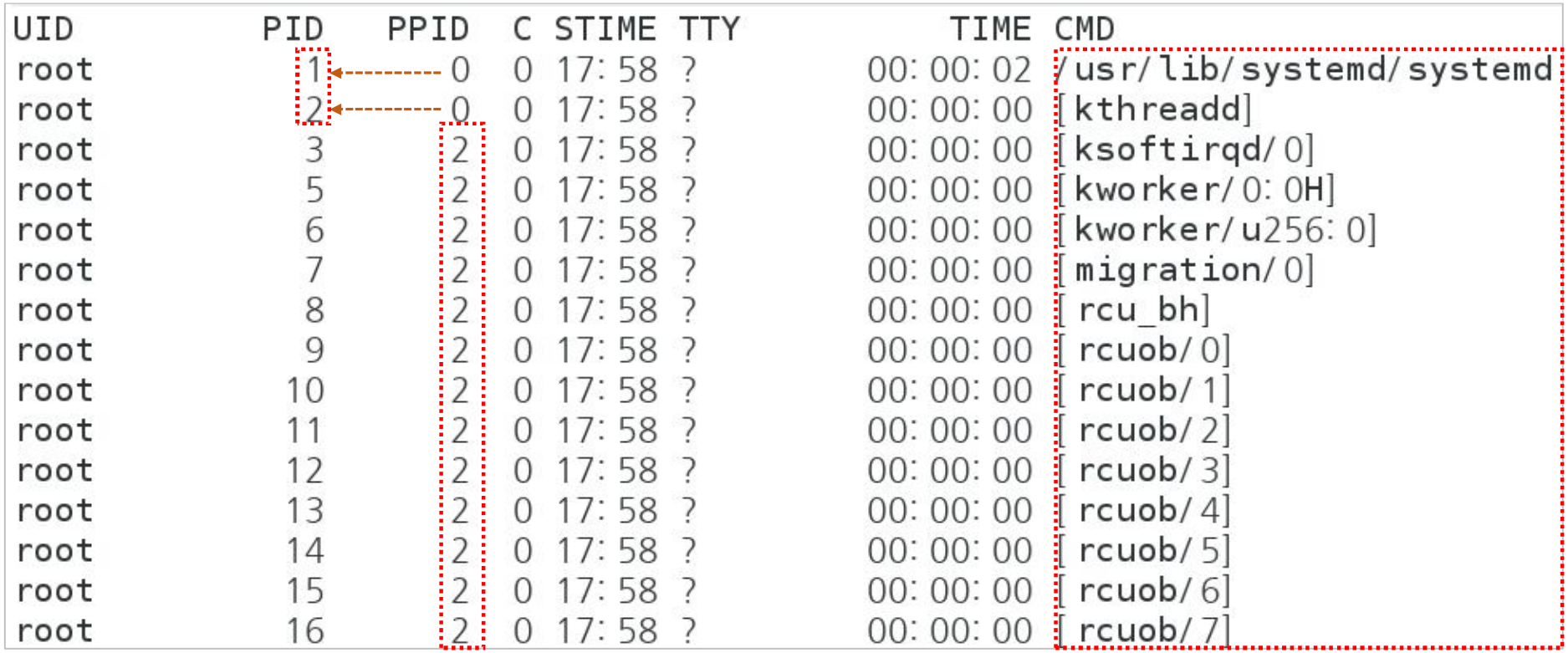

```
[user1@localhost ~]$ ls /boot/vm*
/boot/vmlinuz-0-rescue-66d91756595240f184044ad563d213f0
/boot/vmlinuz-4.18.0-80.el8.x86_64
```

GRUB(Grand Unified BootLoader)

- GNU 프로젝트에서 만든 부트로더
- LILO에 비교하여 다양한 파일시스템을 지원, 부팅 시에 커널 인자를 조정하여 동적인 부팅 지원
- 메뉴 인터페이스 방식을 기본으로 사용하지만, Bash와 같은 명령행 인터페이스를 추가로 제공
- GRUB의 설정은 LILO와 유사하게 하나의 환경설정 파일을 사용
- GRUB의 부트 화면은 그래픽한 메뉴 목록형식으로 제공

❸ 커널 초기화 단계

- 커널은 가장 먼저 시스템에 연결된 메모리, 디스크, 키보드, 마우스 등 장치들을 검사
- **장치 검사 등 기본적인 초기화 과정이 끝나면 프로세스와 스레드 생성**
- 이 프로세스들은 메모리 관리 같은 커널의 여러 가지 동작을 수행
- 프로세스의 개수와 종류는 리눅스의 버전과 종류에 따라 다름
- 커널 프로세스가 생성되면 커널이 수행할 작업이 끝남

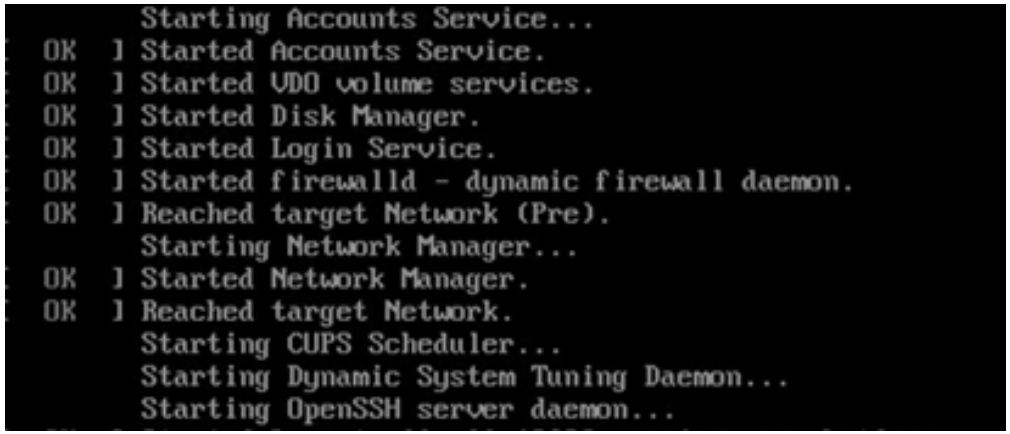


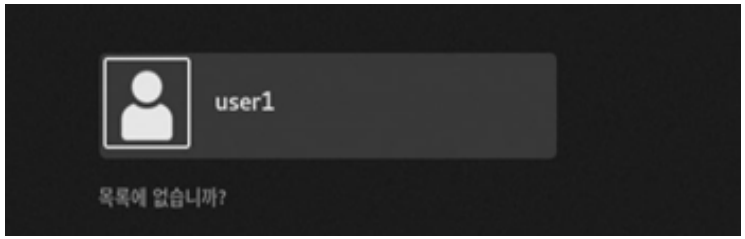
UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	17:58	?	00:00:02	/usr/lib/systemd/systemd
root	2	0	0	17:58	?	00:00:00	[kthreadd]
root	3	2	0	17:58	?	00:00:00	[ksoftirqd/0]
root	5	2	0	17:58	?	00:00:00	[kworker/0:0H]
root	6	2	0	17:58	?	00:00:00	[kworker/u256:0]
root	7	2	0	17:58	?	00:00:00	[migration/0]
root	8	2	0	17:58	?	00:00:00	[rcu_bh]
root	9	2	0	17:58	?	00:00:00	[rcuob/0]
root	10	2	0	17:58	?	00:00:00	[rcuob/1]
root	11	2	0	17:58	?	00:00:00	[rcuob/2]
root	12	2	0	17:58	?	00:00:00	[rcuob/3]
root	13	2	0	17:58	?	00:00:00	[rcuob/4]
root	14	2	0	17:58	?	00:00:00	[rcuob/5]
root	15	2	0	17:58	?	00:00:00	[rcuob/6]
root	16	2	0	17:58	?	00:00:00	[rcuob/7]

- 커널은 메모리 관리, 스케쥴링 같은 동작을 수행하기 위해 여러 개의 프로세스(커널 프로세스)들을 생성
- 이 프로세스는 일반적인 프로세스와 구분되도록 대괄호([ ])로 표시, 주로 PID 번호가 낮게 배정

❹ Systemd 서비스 단계

- systemd 서비스는 기존의 init 스크립트를 대체한 것으로 다양한 서비스를 동작시킴
- 각 서비스가 시작하는 과정은 화면에 메시지로 출력
- CentOS는 버전 7부터 init 대신에 시스템과 서비스 관리자로 systemd를 사용하기 시작했고, systemd 프로세스가 1번 프로세스가 됨





- 마지막으로 systemd 서비스 단계에서는 데몬을 모두 실행한 뒤 그래픽 로그인 시스템인 GDM을 동작

Systemd 관련 명령어 systemctl

- Systemd는 전체 시스템을 시작하고 관리하는 unit이라 부루는 구성요소를 사용
- 관리 대상의 이름을 '서비스이름.unit'형태로 관리

#systemctl

```
rngd.service                                                                loaded active running   Hardware RNG Entropy Ga
rpcbind.service                                                             loaded active running   RPC bind service
rsyslog.service                                                             loaded active running   System Logging Service
rtkit-daemon.service                                                        loaded active running   RealtimeKit Scheduling
smartd.service                                                              loaded active running   Self Monitoring and Rep
sshd.service                                                                loaded active running   OpenSSH server daemon
sysstat.service                                                             loaded active exited    Resets System Activity
systemd-fsck-root.service                                                   loaded active exited    File System Check on Ro
systemd-fsck@dev-di...2d4e04₩x2d9eda₩x2d0c9519d02397.service                loaded active exited    File System Check on /d
systemd-journald.service                                                    loaded active running   Journal Service
systemd-logind.service                                                      loaded active running   Login Service
systemd-random-seed.service                                                 loaded active exited    Load/Save Random Seed
systemd-remount-fs.service                                                  loaded active exited    Remount Root and Kernel
systemd-sysctl.service                                                      loaded active exited    Apply Kernel Variables
```

#systemctl list-unit-files --type service

```
[root@localhost systemd]# systemctl list-unit-files --type service
UNIT FILE                                   STATE
abrt-ccpp.service                           enabled
abrt-oops.service                           enabled
abrt-pstoreoops.service                     disabled
abrt-vmcore.service                         enabled
abrt-xorg.service                           enabled
abrtd.service                               enabled
accounts-daemon.service                     enabled
alsa-restore.service                        static
alsa-state.service                          static
```

#systemctl list-unit-files --type service | grep atd.service

```
[root@localhost systemd]# systemctl list-unit-files --type service | grep atd.service
atd.service                                 enabled
[root@localhost systemd]#
```

#systemctl status atd.service

```
[root@localhost systemd]# systemctl status atd.service
atd.service - Job spooling tools
   Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled)
   Active: active (running) since 수 2023-10-25 17:58:54 KST; 1h 44min ago
 Main PID: 1021 (atd)
   CGroup: /system.slice/atd.service
           └─1021 /usr/sbin/atd -f

10월 25 17:58:54 localhost.localdomain systemd[1]: Started Job spooling tools.
[root@localhost systemd]#
```

#systemctl start atd.service

#systemctl stop atd.service

#systemctl restart atd.service

사용자 환경 설정 파일

환경 설정 파일들

파일명	설 명
/etc/profile	시스템 전체(모든 사용자)에 적용되는 환경변수와 시작관련 프로그램 설정
/etc/bashrc	시스템 전체(모든 사용자)에 적용되는 alias와 함수 설정
~/.bash_profile	• 개인 사용자의 환경설정과 시작 프로그램 설정과 관련이 있는 파일 • 로그인시 읽어 들여, 사용자의 경로, 환경변수 등의 설정이 들어있음 • 사용자가 PATH와 같은 환경변수 수정 시 사용 •/etc/profile에서 설정한 전역 변수들을 덮어 사용할 경우도 있음
~/.bashrc	• 개인 사용자가 정의한 alias와 함수들이 있는 파일 • alias를 지속적으로 사용하려면 이 파일에 설정
~/.bash_logout	• 개인사용자가 로그아웃 할 때 수행하는 설정을 지정하는 파일
/etc/profile.d	• 응용프로그램들이 시작할 때 위한 필요한 스크립트가 위치하는 디렉터리 • 보통 /etc/profile에서 호출 • 일반 사용자의 alias 설정 등과 관련된 스크립트가 존재

Bash shell 종류

- Login Shell

  - 시스템에 로그인을 통하여 접근하는 일반적인 쉘

- Non Login shell

  - 로그인 쉘로부터 파생되는 쉘로써 로그인이 아닌 다른 방식으로 떠있는 쉘

* Login Shell은 /etc/profile, ~/.bash_profile의 환경설정 파일에 영향을 받지만 Non Login Shell의 경우는 영향을 받지 않음
* Login Shell 은 모든 설정 파일에 영향을 다 받지만 Non Login Shell 은 /etc/bashrc, ~/.bashrc 파일만 영향을 받음

Login Shell

- 로그인 시 실행
  - 터미널을 통해 SSH를 통해 리눅스 시스템에 접근할 경우
  - "su -" 명령어를 통해 user를 바꿀 경우
- Login Shell은 logout 명령으로 쉘을 종료

Login Shell 실행 순서

❶ /etc/profile 은 /etc/profile.d 안에 있는 스크립트들 실행

❷ $HOME/ .bash_profile 실행 (or ~/.bash_login or ~/.profile)

❸ $HOME/ .bash_profile 은 $HOME/ .bashrc 파일 실행

❹ $HOME/ .bashrc 는 /etc/bashrc 파일을 실행

/etc/profile → /etc/profile.d → $HOME/.bash_profile → $HOME/.bashrc → /etc/bashrc

* root 로 로그인 시 /etc/profile 을 먼저 읽고 이후 홈 디렉토리의 실행파일들을 실행

```
[root@localhost etc]# tail -15 /etc/profile
    umask 022
fi

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        if [ "${-#*i}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null
        fi
    fi
done

unset i
unset -f pathmunge
[root@localhost etc]#
```

/etc/profile

```
[root@localhost etc]# ls -ld /etc/profile.d
drwxr-xr-x. 2 root root 4096  5월  11 03:59 /etc/profile.d
[root@localhost etc]# ls -l /etc/profile.d
합계 72
-rw-r--r--. 1 root root  771  6월  10  2014 256term.csh
-rw-r--r--. 1 root root  841  6월  10  2014 256term.sh
-rw-r--r--. 1 root root  840  6월  20  2014 PackageKit.sh
-rw-r--r--. 1 root root  461  6월  19  2014 abrt-console-notification.sh
-rw-r--r--. 1 root root  660  6월  10  2014 bash_completion.sh
-rw-r--r--. 1 root root  337  2월   4  2014 colorgrep.csh
-rw-r--r--. 1 root root  345  2월   4  2014 colorgrep.sh
-rw-r--r--. 1 root root 1435  1월  25  2014 colorls.csh
-rw-r--r--. 1 root root 1391  1월  25  2014 colorls.sh
```

/etc/profile.d

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# cat .bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH
[root@localhost ~]#
```

$HOME/.bash_profile

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# cat .bashrc
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
[root@localhost ~]#
```

$HOME/.bashrc

Non Login Shell

- login shell로부터 파생된 Shell
  - ssh로 리눅스 시스템에 접근 후, bash Shell을 실행하는 경우
  - GUI 세션에서 터미널을 띄우는 경우
- exit 명령으로 Shell 종료
- ~/.bashrc 를 실행

NoLogin Shell 실행 순서

❶ /~/.bashrc는 /etc/bashrc 를 실행

❷ /etc/bashrc 는 /etc/profile.d 내 스크립트를 실행

~/.bashrc → /etc/bashrc → /etc/profile.d

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# cat .bashrc
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
[root@localhost ~]#
```

$HOME/.bashrc

```
[root@localhost ~]# tail -15 /etc/bashrc
    # and interactive - otherwise just process them to set envvars
    for i in /etc/profile.d/*.sh; do
        if [ -r "$i" ]; then
            if [ "$PS1" ]; then
                . "$i"
            else
                . "$i" >/dev/null
            fi
        fi
    done

    unset i
    unset -f pathmunge
fi
# vim:ts=4:sw=4
[root@localhost ~]#
```

/etc/bachrc

```
[root@localhost etc]# ls -ld /etc/profile.d
drwxr-xr-x. 2 root root 4096 5월 11 03:59 /etc/profile.d
[root@localhost etc]# ls -l /etc/profile.d
합계 72
-rw-r--r--. 1 root root  771 6월 10 2014 256term.csh
-rw-r--r--. 1 root root  841 6월 10 2014 256term.sh
-rw-r--r--. 1 root root  840 6월 20 2014 PackageKit.sh
-rw-r--r--. 1 root root  461 6월 19 2014 abrt-console-notification.sh
-rw-r--r--. 1 root root  660 6월 10 2014 bash_completion.sh
-rw-r--r--. 1 root root  337 2월  4 2014 colorgrep.csh
-rw-r--r--. 1 root root  345 2월  4 2014 colorgrep.sh
-rw-r--r--. 1 root root 1435 1월 25 2014 colorls.csh
-rw-r--r--. 1 root root 1391 1월 25 2014 colorls.sh
```

/etc/profile.d