

---

# Linux 운영체제

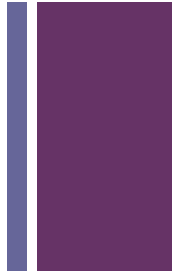
---

Chungbuk National University, Korea  
Intelligent Robots Lab. (IRL)

Prof. Gon-Woo Kim



# 운영 체제(Operating System)



## ■ 운영 체제

- 컴퓨터 시스템의 각종 자원을 효율적으로 관리하고 운영되도록 사용자에게 편리성 제공, 컴퓨터 하드웨어와 사용자간의 인터페이스 역할을 하는 시스템 프로그램
- 자원 관리자 역할과 응용 소프트웨어에게 컴퓨팅 자원에 대한 서비스 제공자 역할

## ■ 커널

- 운영체제를 작동시키는 핵심 프로그램
- 커널은 시스템의 구동에 필요한 환경 설정과 수행되는 프로그램들을 스케줄링 하는 소프트웨어
- 커널이 관리하는 컴퓨팅 자원
  - **물리적 자원:** 프로세서, 메모리, 디스크, 터미널, 네트워크 등과 같은 시스템 구성 요소들과 주변 장치
  - **추상적 자원:** 태스크와 쓰레드(프로세스 추상화), 페이지(page)와 세그먼트(segment) (메모리 추상화), 파일 및 inode(디스크 추상화), 통신 프로토콜 및 패킷(네트워크 추상화) 등, 보안 혹은 사용자 계정에 따른 접근 제어 등



# 운영 체제 (Operating System)



## ■ 모놀리딕 커널

- 전통적인 운영체제 설계 방법
- 프로세스 관리, 메모리 관리, 파일시스템 같은 커널의 모든 기능을 통합한 구조로, 커널 모드에서 모든 기능을 수행
- 구성요소들이 자료구조와 기능을 공유하므로 시스템 자원을 효율적으로 이용
- 커널 코드에 하드웨어와 관련된 저수준 상호 작용이 포함되어 있어 구현된 커널을 다른 시스템에 이식하기 어려움
- 커널의 기능을 확장할 수록 크기가 방대해져 관리가 어려움
- Solaris, AIX, HP-UX 등.

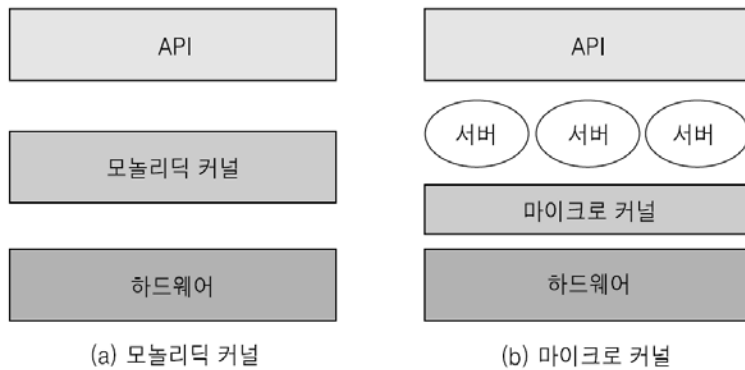
## ■ 마이크로 커널

- 운영체제의 기능을 프로세스 관리 서비스, 메모리 관리 서비스, 파일시스템 서비스와 같이 다수의 소규모 서버 프로세스로 분할하여 설계하는 방식
- 서버 프로세스를 관리하는 최소한의 기능만을 구현한 커널
- 기능 확장이 쉽고 개발된 소규모 서버 모듈을 재사용 가능
- 모놀리딕 커널에 비해 구현이 어려우며, 커널에서 전달하는 기능을 메시지 전달 방식으로 사용하기 때문에 오버헤드가 큼
- 시스템 자원을 효율적으로 사용하기 어려움
- CMU Mach, Cray UNICOS/mk , QNX 등

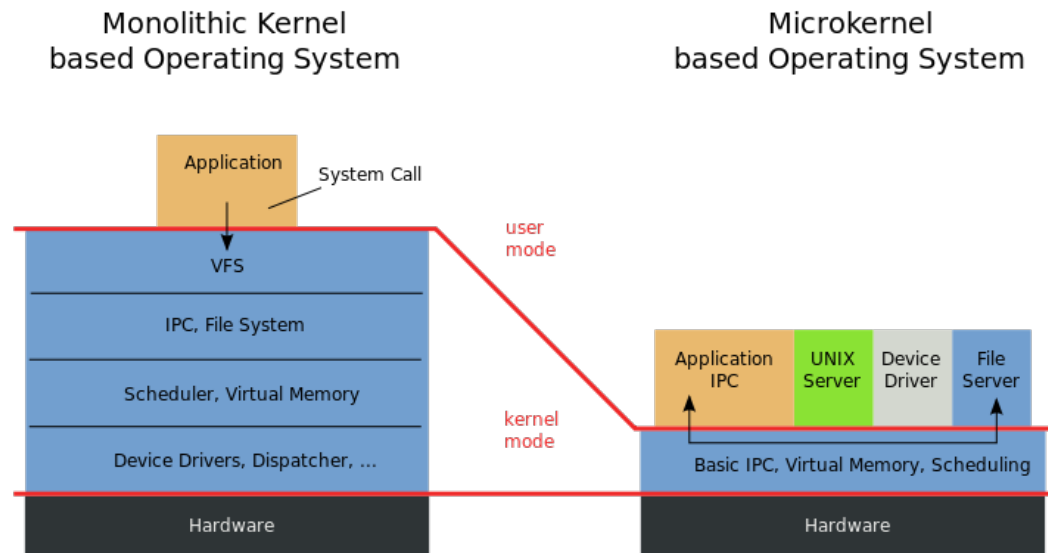


# + 운영 체제 (Operating System)

- 커널 접근 방식: 모놀리딕 커널과 마이크로 커널



[그림 3-2] 커널 접근 방식 구조



- 리눅스의 경우 모놀리딕 커널 방식을 사용하는데, 마이크로 커널의 확장성과 재사용성에 대한 장점을 모듈(module)이라는 개념을 도입해 보완

# + Linux



## ■ 리눅스의 특징

- 무료인 **OS**이며 공개된 운영체제
- **UNIX**와의 호환성
- 강력한 네트워킹기능 지원
- 강력한 멀티태스킹, 멀티유저 동작 지원
- 공개된 소스로 최적화된 작고 안정된 **OS**를 만들어 사용할 수 있음
- 다양한 응용 소프트웨어 제공
- 가상파일시스템을 사용하여 **Ext2, Ext3, Ext4, FAT32, NTFS, ISO9660**등은 물론 저널링파일시스템인 **JFFS, YAFFS, UBIFS** 등의 다양한 파일시스템을 지원
- **IA-32A, IA64, Alpha, ARM, Xscale, PowerPC, MIPS, Sparc**등 다양한 플랫폼에서 동작할 수 있음

# + Linux

## ■ 커널 버전 번호 `linux-x.y.z.tar.gz`

- **x**: 주 버전 번호로 리눅스 골격 자체의 획기적인 변화를 의미
- **y**: 부 버전 번호이며 홀수이면 개발중인 커널이고 짝수이면 안정된 커널
- **z**: 릴리스 번호로 추가된 기능은 없지만 오류 수정 등에 의한 작은 변화를 의미

## ■ 배포판

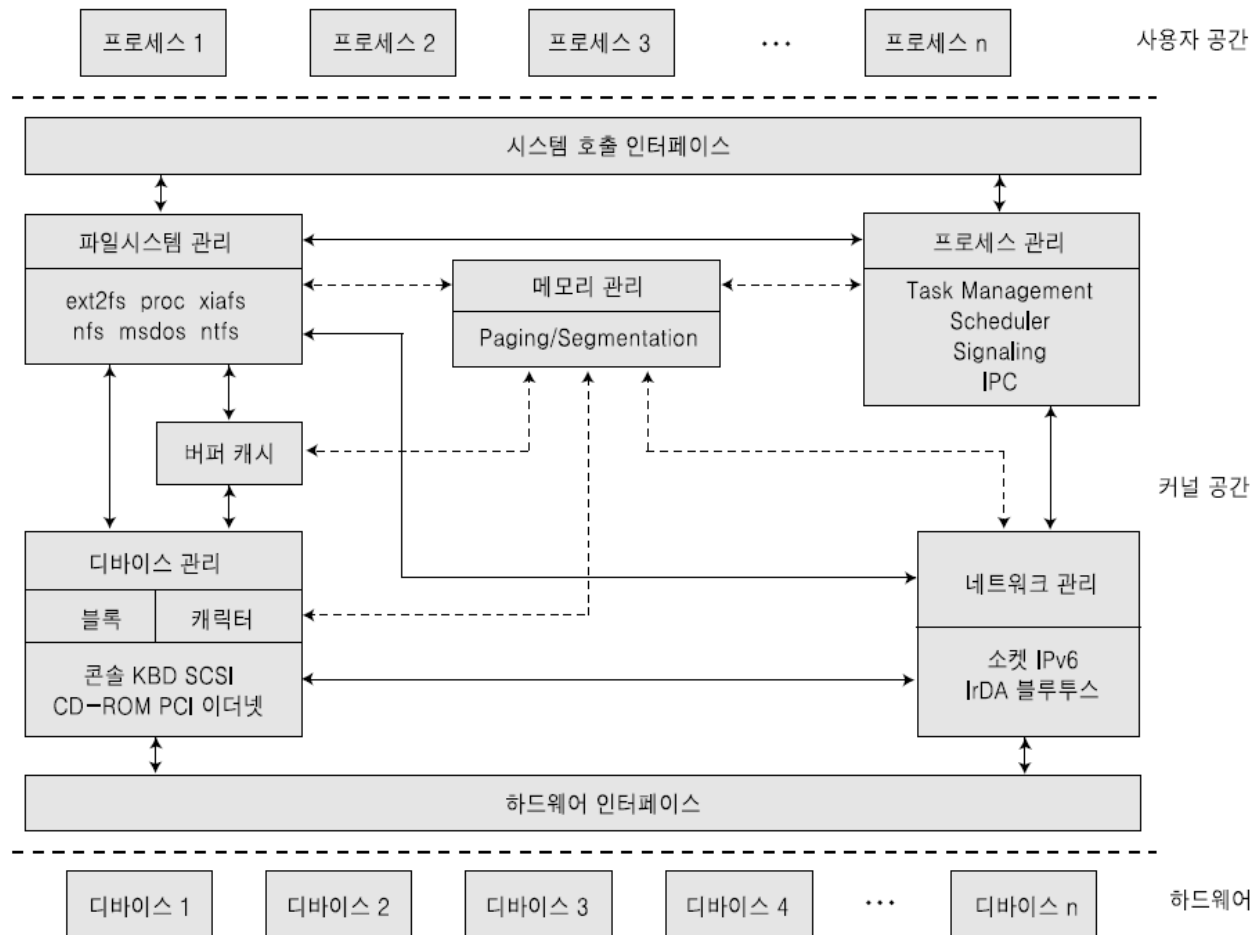
- 비상업적이거나 상업적인 소프트웨어를 통합한 리눅스 패키지
- 리눅스 커널 외에 **GNU** 유틸리티, 개발 환경, **X** 윈도우 시스템 등을 포함
- 배포판: **Redhat Linux, Fedora Core, Ubuntu**



# Linux Kernel

## ■ 리눅스 커널 및 소스 구조

- 프로세스 관리, 메모리 관리, 파일 시스템 관리, 디바이스 관리, 네트워크 관리의 5개의 기능을 블록으로 구분





# Linux Kernel



## ■ 프로세스 관리

- 리눅스 운영체제에서는 시스템이 동작 이후에 최소한 하나 이상의 프로세스가 동작
- 프로세스는 태스크라고도 하며, 주어진 일을 수행하는 기본 단위
- 커널은 프로세스 스케줄러를 이용하여 여러 프로세스가 동작할 수 있도록 각 프로세스를 생성하고 제거하며, 외부 환경과 프로세스를 연결하고 관리

## ■ 메모리 관리

- 시스템에서의 메모리는 프로세서와 마찬가지로 가장 핵심적이고, 중요하게 관리해야 하는 자원
- 메모리를 관리하는 정책은 시스템 성능을 결정하는 중요한 요소로, 각각의 프로세스가 독립적인 공간에서 수행할 수 있도록 가상 주소 공간을 제공
- 가상 메모리 관리를 바탕으로 보조 기억 장치와 연동하여 물리적인 한계를 극복할 수 있는 기능을 제공







# Linux Kernel



## ■ 파일 시스템 관리

- 리눅스 커널은 유닉스 시스템에서 사용하는 파일 시스템을 근간으로 설계
- 리눅스 커널에서 동작하는 응용 프로그램은 시스템에 동작하는 모든 자원을 파일처럼 다룰 수 있도록 통일된 인터페이스를 제공
- 커널은 가상 파일 시스템(VFS)를 이용하여 현존하는 대부분의 파일 시스템 형식을 지원

## ■ 디바이스 제어

- 하드웨어에 관련된 처리는 디바이스 드라이버에서 담당하며, 커널이 반드시 구현해야 하는 것 중 하나
- 리눅스 커널은 파일 시스템의 구조에 디바이스 드라이버를 연동하여 구현하며, 표준화된 형식으로 하드디스크로부터 키보드, 이더넷과 같은 모든 주변 장치를 관리



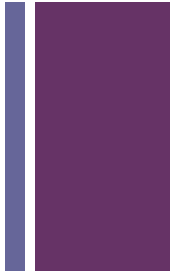
# Linux Kernel



## ■ 네트워크 관리

- 현대의 시스템은 반드시 네트워크 처리를 수반함
- 현존하는 운영체제 중에는 네트워크 처리를 수행하지 않는 운영체제도 있겠지만, 리눅스 커널은 네트워크를 필요로 하는 시스템에서 주로 개발되었기 때문에 가장 우수한 네트워크 관리 시스템을 갖추고 있음
- 리눅스 커널은 네트워크 스택을 이용하여 응용 프로그램과 네트워크 디바이스 드라이버를 연결하며, 매우 효율적인 네트워크 처리를 구현
- 또한 리눅스 커널에서 동작하는 네트워크 시스템은 암호화와 보안 특성이 연계된 매우 견고한 시스템을 구성

# + Linux Kernel



## ■ 리눅스 커널의 기능

구분	기능
프로세스 관리	프로세스의 생성 및 소멸, 프로세스 사이의 통신, 프로세스 스케줄링 동기화
메모리 관리	가상 메모리 관리 기법 제공, 메모리 하드웨어의 효율적 관리
파일시스템 관리	가상 파일시스템(VFS, Virtual File System)에 의한 다양한 파일시스템 지원, 파일 및 디렉토리 관리
디바이스 관리	입출력 요청 작업의 검증 및 스케줄링, 주변장치와 메모리간의 데이터 전송
네트워크 관리	통신 프로토콜 구현, 네트워크 라우팅 및 주소 지정

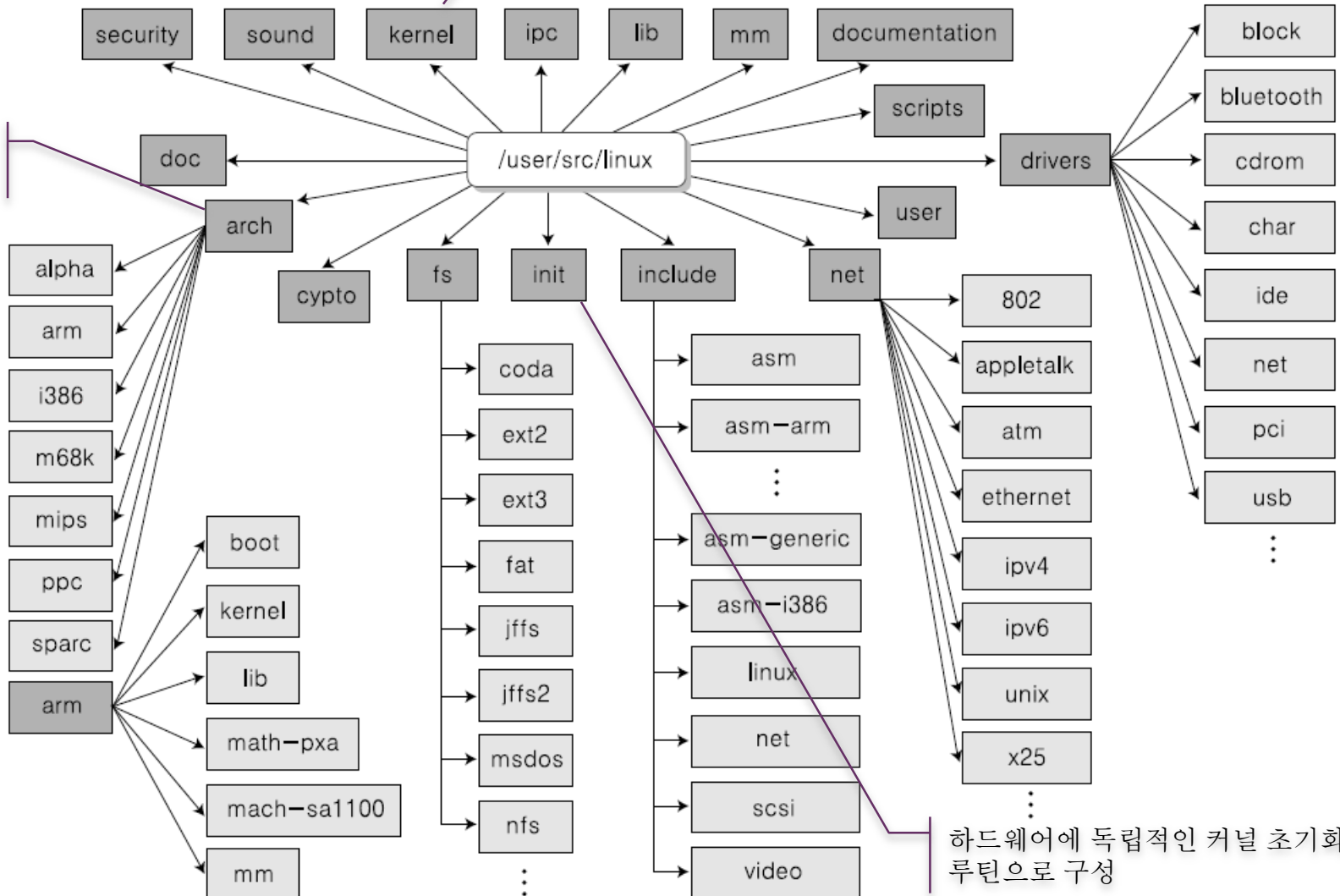


# Linux Kernel

## ■ 리눅스 커널 소스 구조

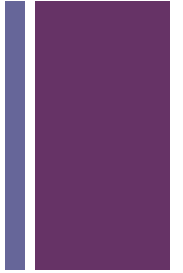
리눅스 커널의 핵심 디렉토리, 태스크의 생성, 소멸, 스케줄링, 시그널 처리 루틴 등 하드웨어 독립적인 커널의 주요 시스템 호출과 관리 루틴으로 구성

프로세서에 종속적인 코드로 구성





# Linux Kernel



## ■ 리눅스 부팅 과정

- ① **ROM BIOS**에 의하여 시스템의 이상 유무를 점검하고 하드웨어를 초기화
- ② 부트로더 프로그램을 메모리에 적재
- ③ 부트로더는 커널을 메모리에 적재
- ④ 커널 실행 후 커널은 최초의 프로세스인 **/sbin/init** 코드 실행
- ⑤ **init** 프로세스에 의해 운영체제 초기화

# + Linux

## ■ Users and Groups

- 멀티유저 시스템에서 각 사용자는 자신만의 **private space**를 갖는다.
  - 디스크 영역 할당, **Private mail messages, and so on**
  - 사용자는 다른 사용자의 **private space**를 침해할 수 없다.
- 사용자 관리
  - **User ID (UID)**
    - 사용자를 구분하기 위한 고유 번호 (**identifier**)
    - 컴퓨터는 사용자에게 **Login name and password**를 물어 본다.
  - **Groups**
    - 선택적으로 다른 사용자와 자원을 공유할 수 있도록 한 개 이상의 그룹에 속할 수 있음
    - **Group ID (GID)**: 그룹에게 부여되는 하나 고유 번호
    - 파일은 한 개의 그룹과 연관되어 있음
  - **Root, Superuser, or Supervisor**
    - 시스템 관리자 계정
    - 사용자 계정을 다루거나, 시스템 백업, 프로그램 갱신 등과 같은 시스템 유지 작업을 하려면 반드시 **root**로 **login**
    - **Root**는 시스템에 있는 모든 파일을 액세스할 수 있으며, 실행 중인 모든 사용자 프로그램에 간섭할 수 있다.

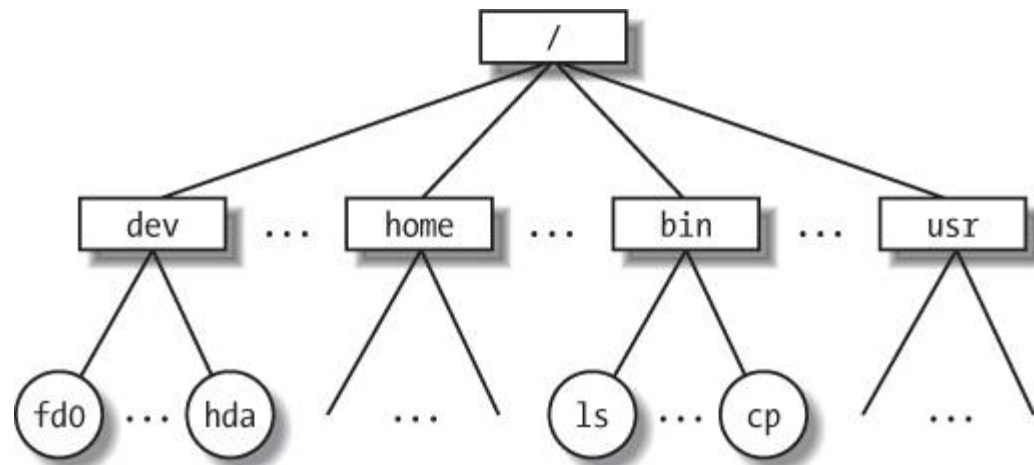
# + Linux Root Filesystem

## ■ Linux file

- An information container as a sequence of bytes
- 커널은 파일의 내용을 해석하지 않는다.
- 프로그램은 커널이 제공하는 시스템 콜에 의하여 파일을 사용한다.
- 사용자 관점에서 파일은 트리 구조의 **namespace**로 조직되어 있다.

## ■ Root filesystem

- 시스템 내부의 디렉토리 구조
- Tree 형태
- FSSTND (Linux Filesystem Standard) 표준 준수



# + Linux Root Filesystem

## ■ 파일을 사용하는 사용자의 구분

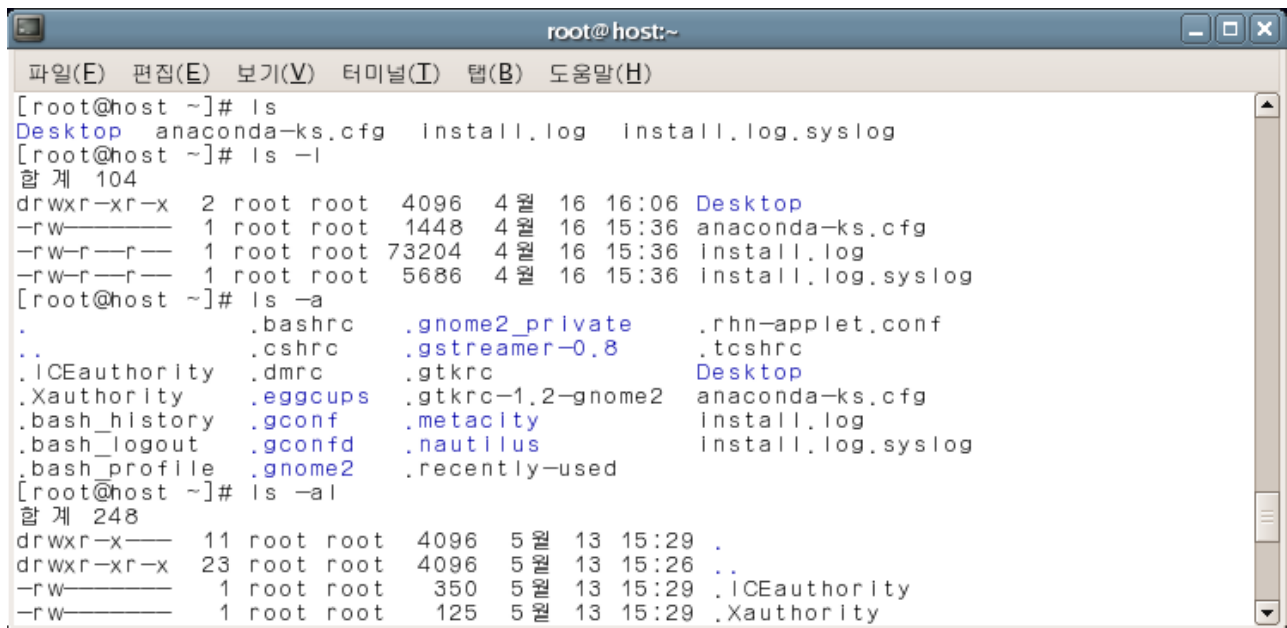
- **owner:** 파일 소유자
- **group:** 파일 소유자와 같은 그룹에 속한 사용자
- **others:** 기타 나머지

## ■ Access rights (사용 권한)

- 각 사용자 **class**마다 세 개의 **access rights**가 부여된다.

### ■ Read/Write/execute

- 각 파일에 부여되며, 사용자 클래스마다 존재하므로 모두 9 개의 **binary flag**가 할당된다.



```
root@host:~  
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)  
[root@host ~]# ls  
Desktop anaconda-ks.cfg install.log install.log.syslog  
[root@host ~]# ls -l  
합 계 104  
drwxr-xr-x  2 root root  4096  4월 16 16:06 Desktop  
-rw-r--r--  1 root root  1448  4월 16 15:36 anaconda-ks.cfg  
-rw-r--r--  1 root root 73204  4월 16 15:36 install.log  
-rw-r--r--  1 root root  5686  4월 16 15:36 install.log.syslog  
[root@host ~]# ls -la  
.  
..  
ICEauthority .bashrc .gnome2_private .rhn-applet.conf  
.cshrc .gstreamer-0.8 .tcshrc  
.dmrc .gtkrc .gnome2 .Desktop  
.Xauthority .eggccups .gtkrc-1.2-gnome2 anaconda-ks.cfg  
.bash_history .gconf .metacity install.log  
.bash_logout .gconfd .nautilus install.log.syslog  
.bash_profile .gnome2 .recently-used  
[root@host ~]# ls -al  
합 계 248  
drwxr-xr-x  11 root root  4096  5월 13 15:29 .  
drwxr-xr-x  23 root root  4096  5월 13 15:26 ..  
-rw-r--r--  1 root root   350  5월 13 15:29 .ICEauthority  
-rw-r--r--  1 root root   125  5월 13 15:29 .Xauthority
```

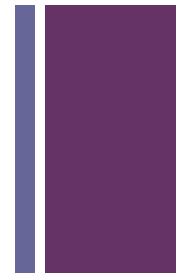


# + Linux Root Filesystem

## ■ Root filesystem의 최상위 디렉토리 구조

- `/`: “root”, 최상위 디렉토리
- `/bin`: 기본적인 명령어를 저장하고 있는 디렉토리
- `/boot`: 리눅스 부트로더를 저장하고 있는 디렉토리
- `/dev`: 시스템 디바이스 파일을 저장하고 있는 디렉토리
- `/etc`: 시스템 설정 파일을 저장하고 있는 디렉토리
- `/home`: 사용자 홈 디렉토리
- `/lib`: 커널 모듈과 라이브러리 파일을 저장하고 있는 디렉토리
- `/lost+found`: 파일 복구를 위한 디렉토리
- `/media`: 탈부착이 가능한 장치들의 마운트 포인트로 사용하는 디렉토리
- `/mnt`: 일시적인 마운트 포인트로 사용하는 디렉토리
- `/opt`
- `/proc`: 시스템 정보를 보기 위한 가상 파일시스템
- `/root`: root 사용자의 홈 디렉토리
- `/sbin`: 시스템 관리용 실행 파일을 저장하는 디렉토리
- `/tmp`: 임시 파일 생성용 디렉토리
- `/usr`: user application을 저장하는 디렉토리
- `/var`: 시스템 운영 중 생성되는 각종 임시 파일을 저장하는 디렉토리

# + Linux Root Filesystem



- Root filesystem의 주요 디렉토리
  - `/`: “root”, 최상위 디렉토리
    - 리눅스의 모든 디렉토리의 시작점
  - `/bin`: 기본적인 명령을 저장하고 있는 디렉토리
    - binaries의 약자.
    - 예: `mv`, `cp`, `rm`, `rmdir`, `df`, `sync` 등
    - `root` 사용자 이외에 일반 사용자들도 사용할 수 있다.
    - `/sbin`의 명령은 일반적으로 `root` 사용자만 사용할 수 있다.
  - `/home`: 사용자 홈 디렉토리
    - `adduser` 명령으로 새로운 사용자를 생성하면 사용자 ID와 동일한 이름의 디렉토리가 자동으로 생성된다.

# + Linux Root Filesystem

- Root filesystem의 주요 디렉토리
  - **/proc**: 시스템 정보를 보기 위한 가상 파일시스템
    - 디스크에 존재하는 파일이 아니다.
    - 현재 메모리에 존재하는 작업들에 대한 정보를 파일 형태로 보여준다.
    - **cat** 명령을 사용하여 시스템 정보를 확인할 수 있다.
    - 예) **#cat /proc/cpuinfo**
    - **/proc/1/status**: 첫 번째 프로세스의 상태
    - **/proc/cpuinfo**: **cpu**의 타입, 제조자, 모델, 성능
    - **/proc/devices**: 현재 동작 중인 디바이스 드라이버 목록
    - **/proc/dma**: 현재 사용 중인 **DMA** 채널 목록
    - **/proc/filesystems**: 커널에 설정되어 있는 파일시스템 목록
    - **/proc/interrupts**: 사용 중인 인터럽트 목록과 사용 현황
    - **/proc/ioports**: 사용 중인 **io ports** 목록
    - **/proc/kmsg**: 커널이 출력하는 커널 메시지
    - **/proc/meminfo**: 메모리의 사용량과 가상 메모리에 대한 정보
    - **/proc/stat**: 시스템의 현재 상태
    - **/proc/uptime**: 시스템이 동작한 시간
    - **/proc/version**: 커널 버전 정보



# Linux Shell



## ■ Shell

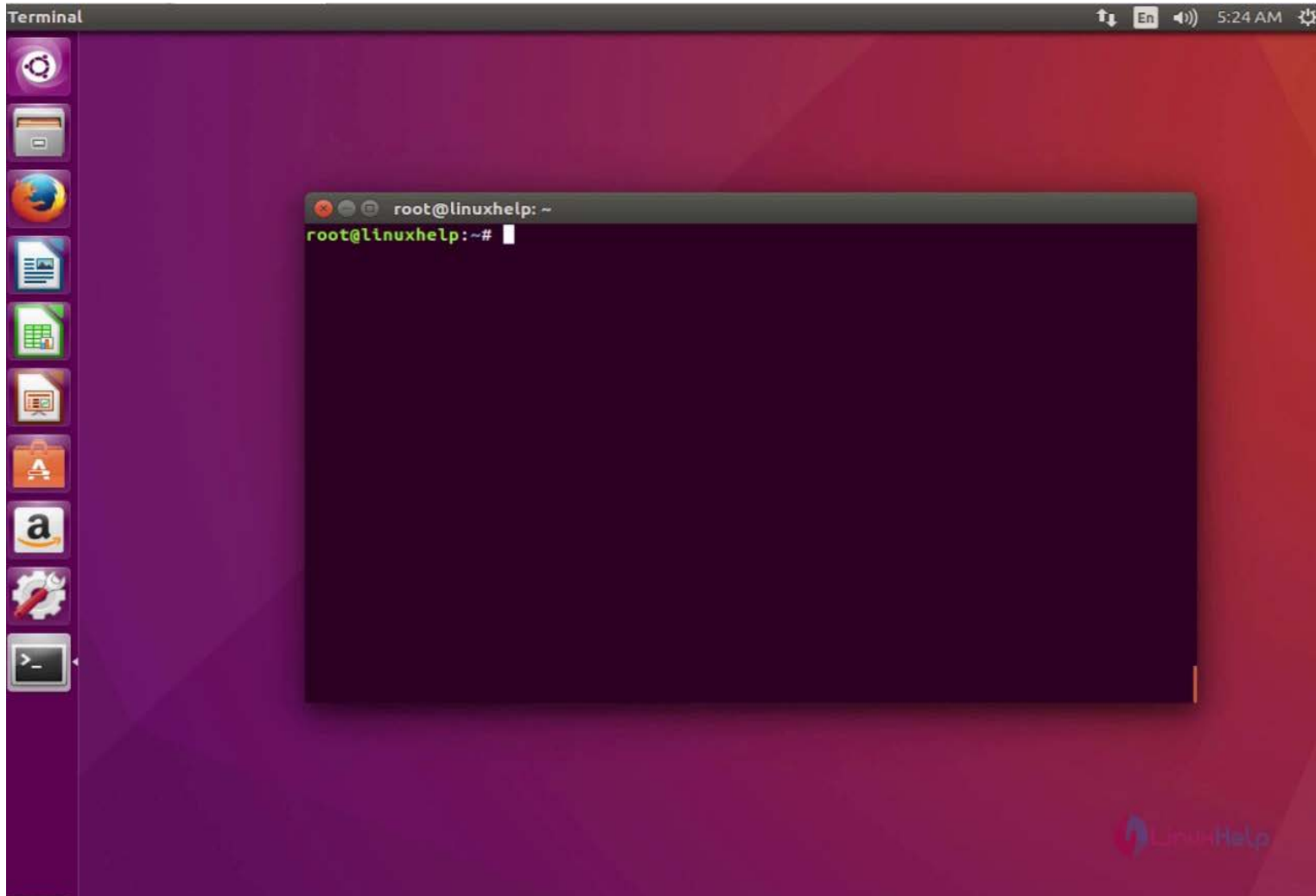
- 리눅스의 사용자 명령(**command**)를 해석하고 처리하는 계층
- 즉, **terminal program**에서 명령을 받고 처리하는 계층
- 유닉스가 진화하는 과정에서 여러 가지 **shell**이 개발됨

## ■ Shell의 종류

- **bash** (Bourne Again Shell)
  - 리눅스에서 가장 많이 사용되는 **shell**
  - POSIX 호환
- **csh** (C Shell)
  - 버클리에서 개발
  - 기능이 추가, 명령행 편집 기능을 제공하지 않음
- **ksh** (Korn Shell)
  - 일반적으로 유닉스에서 가장 많이 하는 **shell**
  - **C shell**의 기능을 추가, 명령행 편집 기능을 제공하지 않음
- **sh**: **bash**의 **original shell**.
- **tcsh**: 확장 **C shell**.
- **zsh**: 가장 최근에 나온 **shell**. **bash**와 호환되며, 명령행 편집 기능을 제공

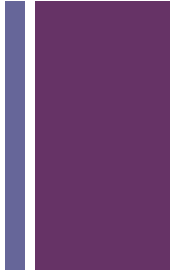


# + Linux Shell





# Linux Shell



## ■ Shell

- 현재 사용하고 있는 **shell**을 알아보는 방법

- `$ echo $SHELL`

## ■ Command Line

- Shell prompt에서 명령을 입력하면, **shell**은 해당 프로그램을 실행시킴

- 명령어 형식: **command** [options] [arg1] [arg2] ...

- **Option**: 명령을 처리하는 방법을 나타냄

- 예) `ls -r`

- 예) `ls -x`

# + Linux 명령어

## ■ 시스템 관리

- **logout**: login shell 종료하기
- **reboot**: 재 부팅하기
- **df**: 파일시스템이 사용한 용량과 사용 가능한 디스크 용량을 보여줌
- **ps**: 프로세스 상태, 현재 실행되고 있는 프로세스들의 목록을 보여줌
- **kill**: 실행 중인 프로세스를 정지시킴
- **pwd**: (present working directory): 현재 작업중인 디렉토리를 보여줌



# Linux 명령어



## ■ 사용자 관리

### ■ **useradd**: 사용자 계정 만들기

- **root** 권한으로만 수행할 수 있다.
- 기본적으로 **/home** 디렉토리 아래에 사용자 계정을 만든다.
- 사용 방법: **useradd [-options] username**

### ■ **sudo**

- **ubuntu**에서 **superuser** 권한으로 명령을 실행하는 경우에 사용한다.
- 일반적으로 시스템을 관리하고자 할 때 **superuser (root)** 권한을 가져야 함

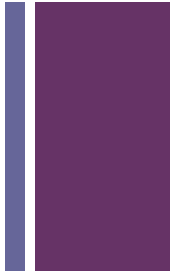
### ■ **sudo su**

- 일반 사용자에서 **superuser**로 전환함
- 정확한 암호를 입력한 후 **superuser** 상태로 들어감





# Linux 명령어



## ■ 파일 및 디렉토리 관리

### ls 명령어

- 기    능    ◉ 파일 및 디렉토리의 목록을 출력한다.
- 형    식    ◉ ls [옵션] [ {디렉토리 | 파일} ]
- 옵션    ◉ -l : 파일 크기, 저장 날짜, 허가권, 소유권 등을 포함하여 자세히 보기  
            -a : 마침표로 시작하는 숨겨진 파일을 포함한 모든 파일 보여주기  
            -t : 시간 순서대로 정렬해서 보기  
            -R : 하위 디렉토리의 파일까지 재귀적으로 보여주기



# Linux 명령어



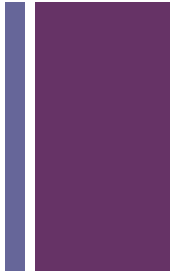
## ■ 파일 및 디렉토리 관리

### cd 명령어

- 기능 ➡ 디렉토리를 이동한다.
- 형 식 ➡ cd [디렉토리명]
- 디렉토리명 ➡ ~ : 자신의 홈 디렉토리  
~(계정 이름) : 다른 사용자의 홈 디렉토리. 예를 들어, cd ~embed일 경우  
embed의 홈 디렉토리  
. : 현재 디렉토리  
.. : 상위 디렉토리  
- : 바로 직전에 사용한 디렉토리



# Linux 명령어



## ■ 파일 및 디렉토리 관리

### cp 명령어

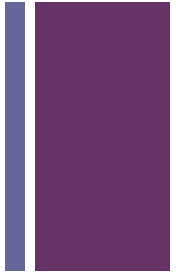
- 기    능    ➡ 파일이나 디렉토리를 복사한다.
- 형    식    ➡ cp [옵션] <소스 파일> <타겟 파일>
- 옵션   ➡ -f : 강제 복사. 복사할 타겟 파일이 존재해도 무시하고 복사한다.  
              -i : 대화형 복사. 파일 복사 전에 복사 여부를 확인한다.  
              -r : 재귀적으로 실행. 파일을 복사할 때 하위 디렉토리를 포함해 모든 파일을 복사한다.

### mv 명령어

- 기    능    ➡ 파일이나 디렉토리 이름을 변경하거나 위치를 이동한다.
- 형    식    ➡ mv [옵션] <소스 파일> <타겟 파일>
- 옵션   ➡ -f : 강제 이동. 이동할 타겟 파일이 존재해도 무시하고 이동한다.  
              -i : 대화형 이동. 이동하기 전에 이동 여부를 확인한다.



# Linux 명령어



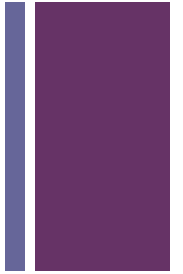
## ■ 파일 및 디렉토리 관리

### rm 명령어

- 기능 ○ 파일을 삭제한다.
- 형식 ○ rm [옵션] 파일명
- 옵션 ○
  - f : 강제 삭제, 바로 삭제한다.
  - i : 대화형 삭제, 파일 삭제 전에 삭제 여부를 확인한다.
  - r : 재귀적으로 실행, 파일을 삭제할 때 하위 디렉토리를 포함해 모든 파일을 삭제한다.



# Linux 명령어



## ■ 파일 및 디렉토리 관리

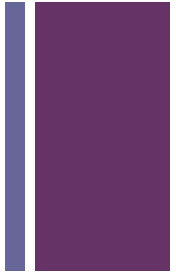
### mkdir 명령어

- 기능 ○ 디렉토리를 생성한다. 생성된 디렉토리는 명령어를 수행한 사용자의 소유가 된다.
- 형식 ○ mkdir 디렉토리명

### rmdir 명령어

- 기능 ○ 디렉토리를 삭제한다.
- 형식 ○ rmdir 디렉토리명

# + Linux 명령어



## ■ 파일 및 디렉토리 관리(계속)

### cat 명령어

- 기      능 ➡ 텍스트 파일의 내용을 출력한다.
- 형      식 ➡ cat 파일명

### more 명령어

- 기      능 ➡ 텍스트 파일의 내용을 화면에 한 페이지씩 출력한다.
- 형      식 ➡ more 파일명

# + Linux 명령어

## ■ 파일 및 디렉토리 관리(계속)

### touch 명령어

- 기능 ➡ 빈 파일을 생성하거나 파일의 생성 시간을 현재로 변경한다.
- 형식 ➡ touch 파일명

# + Linux 명령어

## ■ 파일 및 디렉토리 관리

### ln 명령어

- 기      능    ➡ 파일 사이의 링크를 생성한다.
- 형      식    ➡ ln [-s] <링크 대상 파일명> <링크 파일명>





# Linux 명령어



## ■ 마운트/언마운트

### mount 명령어

- 기능 ○ 디바이스를 디렉토리와 연관시켜 파일과 같이 사용할 수 있도록 인식시킨다.
- 형식 ○ mount <장치 디렉토리> <마운트 디렉토리>

### umount 명령어

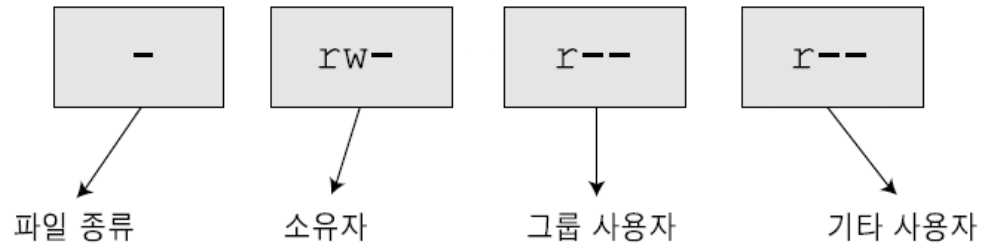
- 기능 ○ 마운트된 디바이스를 디렉토리에서 해제한다.
- 형식 ○ umount <마운트 디렉토리>

# + Linux 명령어

## ■ 허가권 관리 명령어

### chmod 명령어

- 기능 ○ 허가권을 변경한다.
- 형식 ○ chmod 권한 파일명



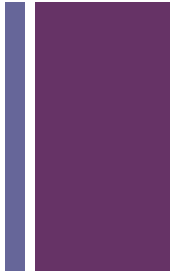
[그림 3-15] 파일의 허가권 표기

## ■ 허가권의 의미와 허가권 값

문자	허가권	값(8진수)
<b>R</b>	읽기(Read)	<b>4</b>
<b>W</b>	쓰기(Write)	<b>2</b>
<b>X</b>	실행하기(eXecute)	<b>1</b>



# Linux 명령어



## ■ 허가권 관리 명령어(계속)

### ■ 기호 모드의 구성요소

권한 변경 - 기호 모드



### ■ 기호 모드에서 사용하는 문자와 기호

구분	문자/기호	의미
사용자 카테고리	u	파일 소유자
	g	소유자가 속한 그룹
	o	소유자와 그룹 이외의 기타 사용자
연산자	+	권한 부여
	-	권한 제거
권한	r	읽기 권한
	w	쓰기 권한
	x	실행 권한

# + Linux 명령어

## ■ 네트워크 관리 명령어

### ping 명령어

- 기능 ➡ 네트워크에 연결되어 있는지를 확인한다.
- 형식 ➡ ping {<IP 주소> | URL}

# + Linux 명령어

## ■ 네트워크 관리 명령어

### ifconfig 명령어

- 기능 ➡ 네트워크 인터페이스를 설정 및 확인한다.
- 형식 ➡ `ifconfig [dev] [ip] [netmask] [broadcast_address]`

# + Linux 명령어

## ■ 기타 명령어

### gzip 명령어

- 기능 ➡ 파일을 압축한다.
- 형식 ➡ gzip [옵션] 파일명

# + Linux 명령어

## ■ 기타 명령어

### gunzip 명령어

- 기능 ➡ .gz로 압축된 파일의 압축을 푼다.
- 형식 ➡ gzip [옵션] 파일명

# + Linux 명령어

## ■ 기타 명령어(계속)

### tar 명령어

- 기    능    ➡ 파일이나 디렉토리를 하나로 묶거나 푼다.
- 형    식    ➡ tar 옵션 파일명 [위치]
- 옵션    ➡
  - c : 하나의 파일로 묶기(compress)
  - x : 묶인 파일 풀기(extract)
  - v : 파일을 묶거나 풀 때 진행 과정을 자세히 보여줌(verbose)
  - f : 묶음 파일명, tar 명령어를 사용할 때 반드시 사용(file)
  - z : gzip과 관련하여 압축/복원을 동시에 수행





# Summary

- 커널은 운영체제를 작동시키는 핵심으로, 응용 프로그램과 컴퓨팅 자원 사이의 인터페이스 역할을 하며, 자원을 관리
- 커널을 접근하는 방법에 따라 모놀리딕 커널(**monolithic kernel**)과 마이크로 커널(**micro kernel**)로 구분
- 임베디드 리눅스 운영체제는 저성능의 프로세서와 소용량의 메모리를 가진 제한된 컴퓨팅 자원하에서 특정 응용 프로그램의 수행에 필요한 요구 사항을 충족시킨 최적화된 리눅스 커널을 의미

## 주요 리눅스 명령어

명령어	의미
ls	파일과 디렉토리의 목록 출력
cd	디렉토리 이동
cp	파일이나 디렉토리를 복사
mv	파일이나 디렉토리 이름을 변경하거나 다른 디렉토리로 이동
rm	파일 삭제
mkdir/rmdir	디렉토리 생성/디렉토리 삭제
cat	텍스트 파일의 내용 출력
more	텍스트 파일의 내용을 화면에 한페이지씩 출력
touch	빈 파일의 생성 혹은 파일의 생성 시간을 현재로 변경
ln	파일사이의 링크 생성
rpm	패키지 설치
gzip/gunzip	파일 압축/파일 압축 해제
chmod	소유권 변경
tar	파일 묶기, 풀기