



2020 혁신성장 청년인재 집중양성 사업

빅데이터 수업 1주차

# 리눅스 1차

#Hyper-V #WSL2 #CentOS #Ubuntu

시작하기에 앞서

## 1. Windows Version Check

- Winkey + R >> winver
- if(OS build 19041 이하면) {  
    windows update;  
}

## 2. CentOS 8 Download(CentOS Stream DVD ISO)

<https://www.centos.org/>

## 3. Send to Email

[remaper.ku@gmail.com](mailto:remaper.ku@gmail.com)

제목 : [프로젝트기반빅데이터전문가과정] 이름

내용 : 연락처

## 4. 강사소개

왜 리눅스죠?

# >>>> 리눅스를 사용하는 이유

- 무료
- 용도에 상관없이 언제 어디든지 다른 OS보다 훨씬 더 쉽게 설치할 수 있다
- 유용하고 창의적인 일을 할 수 있다
- 유닉스의 거의 모든 도구를 동일하게 제공 : 안정성과 보안성이 뛰어나다
- 오픈 소스(open source) : 누구든 코드 베이스를 가져다 원하는 대로 바꿀 수 있다

## 사용 목적별 리눅스 배포판

목적	배포판
보안/해킹 방지	칼리(Kali) 리눅스 패럿(Parrot)
일반 사용자용 데스크톱	민트(Mint) 엘리멘트리(Elementary) OS
경량 OS(오래된 하드웨어, 진단용)	퍼피(Puppy) 리눅스 LXLE
사물 인터넷 관리	스냅피(Snappy) 우분투 코어(Ubuntu Core)
기업용 서버	CentOS(레드햇 기업용 리눅스의 커뮤니티 버전) 오픈수세(openSUSE)(수세의 커뮤니티 버전)
클라우드 컴퓨팅	아마존 리눅스(AWS AMI) 우분투 서버(AWS AMI)
범용(경량 OS 제외)	우분투



## 리눅스를 사용하는 이유

- 무료(기업 및 기관 서버에서 많이 사용 함)
- 다중 사용자에게 다중 접속
- 사용자별 권한 컨트롤
- 도커와 같은 컨테이너를 이용한 자유로운 운영체제 적용
- 터미널을 이용한 일괄 작업 수행
- RHEL과 같은 유료라이선스를 사용할 경우 **기술지원**, 지속적 관리, 보안업데이트를 받을 수 있음.

시작



# A table of Contents

1

Windows VM Machine :: Hyper-V and WSL2

2

리눅스 시작하기

3

원격 연결





# Part 1

Windows VM Machine :: Hyper-V and WSL2

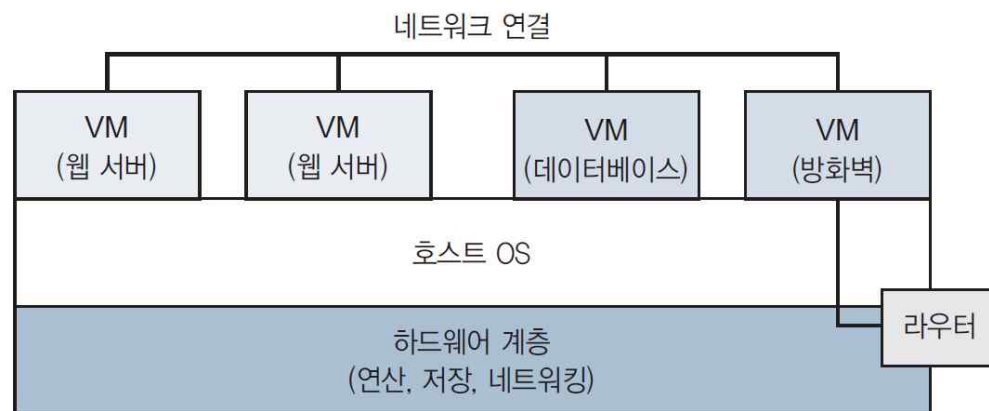


# Windows VM Machine :: Hyper-V and WSL2

## 가상화(virtualization)

- 리눅스가 가상 공간의 절대 우위를 차지한다
- 가상화를 이용하면 어떤 기술이든 더 쉽게 배울 수 있다

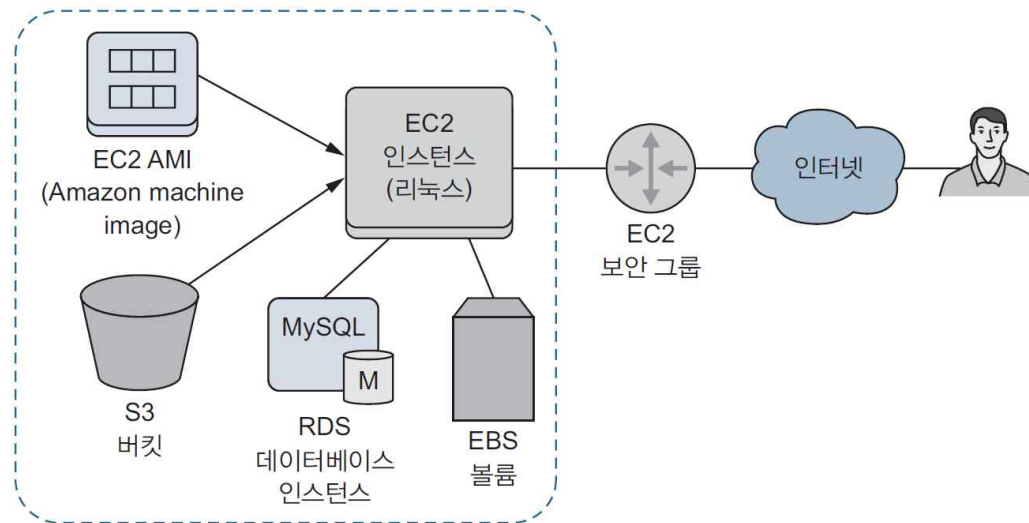
하드웨어 호스트 안에서 서로 연결되고 외부 라우터를 통해 더 큰 네트워크로 연결되는 VM 클라이언트들



# Windows VM Machine :: Hyper-V and WSL2

- 아마존 웹 서비스(AWS, Amazon Web Services) : 수많은 유명 온라인 서비스를 포함해 무수한 작업량을 소화해내는 수백만 대의 VM을 호스팅하는 (리눅스) 서버 용량을 고객에게 임대한다

AWS의 EC2 VM을 중심으로 한 전형적인 클라우드 컴퓨팅 구성

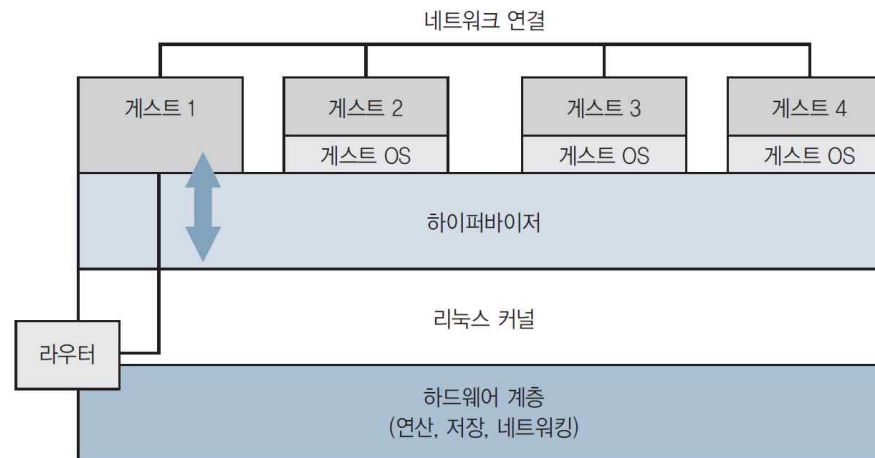


# Windows VM Machine :: Hyper-V and WSL2

## 가상화 기법 1 : 하이퍼바이저(hypervisor)

- 호스트 시스템 하드웨어를 몇 단계로 제어해 각 게스트 OS에 필요한 리소스를 제공
- AWS, KVM, VMWare ESXi, KVM, Hyper-V

특별한 관리 임무를 맡은 게스트 1과 완전한 OS가 설치된  
각 게스트를 함께 보여주는 타입 2 하이퍼바이저 아키텍처

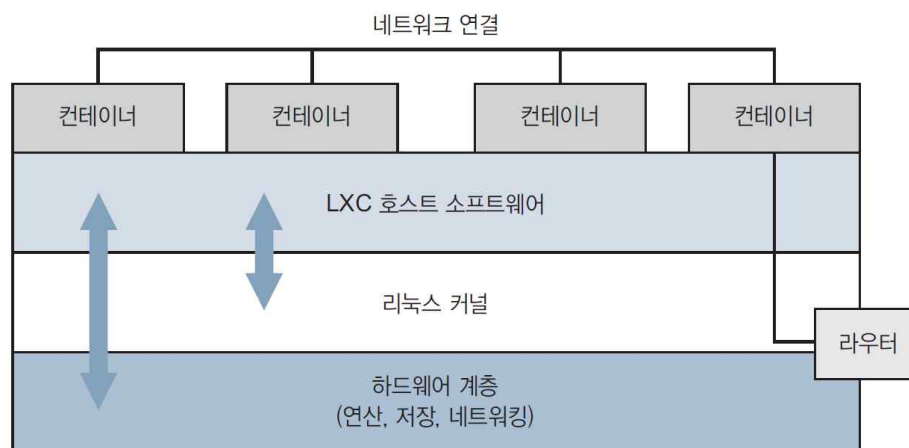


# Windows VM Machine :: Hyper-V and WSL2

## 가상화 기법 2 : 컨테이너(container)

- 초경량 가상 서버로, 완전히 독립적인 OS를 실행하는 대신 호스트 OS의 커널을 공유
- 도커(Docker)

LXC 환경과 리눅스 커널 그리고 그 아래 하드웨어 계층 간 연결을 보여주는 LXC 아키텍처





# Windows VM Machine :: Hyper-V and WSL2

## WSL2 설치

### Windows PowerShell

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

### Windows PowerShell

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

### Windows PowerShell

```
wsl --set-default-version 2
```

Microsoft Store 열기 > Ubuntu 선택 및 설치



# Windows VM Machine :: Hyper-V and WSL2

## Hyper-V and CentOS 8

### Windows PowerShell

Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All

### Windows PowerShell

DISM /Online /Enable-Feature /All /FeatureName:Microsoft-Hyper-V

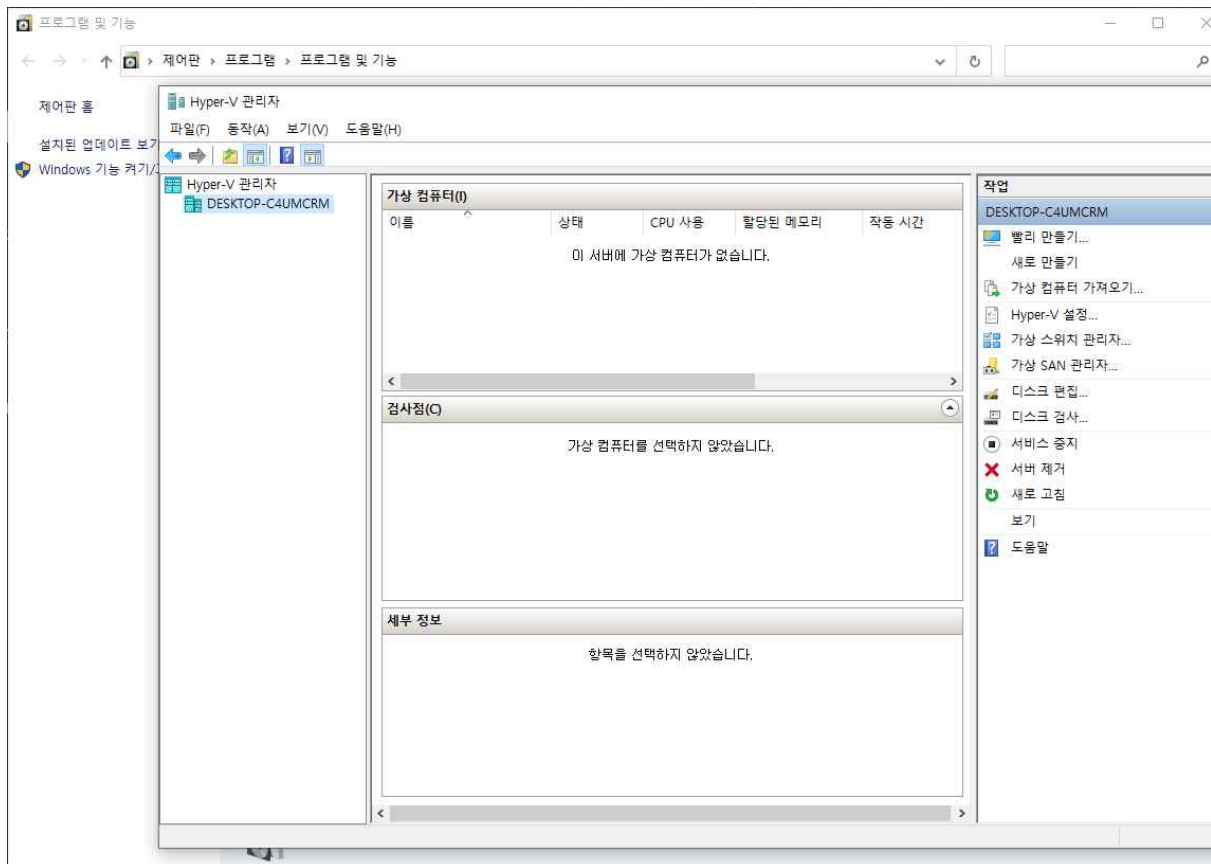
- 1.Windows 단추를 마우스 오른쪽 단추로 클릭하고 '앱 및 기능'을 선택합니다.
- 2.오른쪽의 관련 설정에서 **프로그램 및 기능**를 선택합니다.
- 3.**Windows** 기능 사용/사용 안 함을 선택합니다.
- 4.**Hyper-V**를 선택하고 **확인**을 클릭합니다.





# Windows VM Machine :: Hyper-V and WSL2

## Hyper-V and CentOS 8







# Windows VM Machine :: Hyper-V and WSL2

## Hyper-V and CentOS 8

새 가상 컴퓨터 마법사

이름 및 위치 지정

시작하기 전  
이름 및 위치 지정  
세대 지정  
메모리 할당  
네트워킹 구성  
가상 하드 디스크 연결  
설치 옵션  
요약

이 가상 컴퓨터의 이름과 위치를 선택하십시오.


이름은 Hyper-V 관리자에 표시됩니다. 게스트 운영 체제나 작업의 이름과 같이 이 가상 컴퓨터를 쉽게 식별할 수 있는 이름을 사용하는 것이 좋습니다.

이름(M):

폴더를 만들거나 기존 폴더를 사용하여 가상 컴퓨터를 저장할 수 있습니다. 폴더를 선택하지 않으면 가상 컴퓨터가 이 서버에 대해 구성된 기본 폴더에 저장됩니다.

☒ 가상 컴퓨터를 다른 위치에 저장(S)

위치(L):

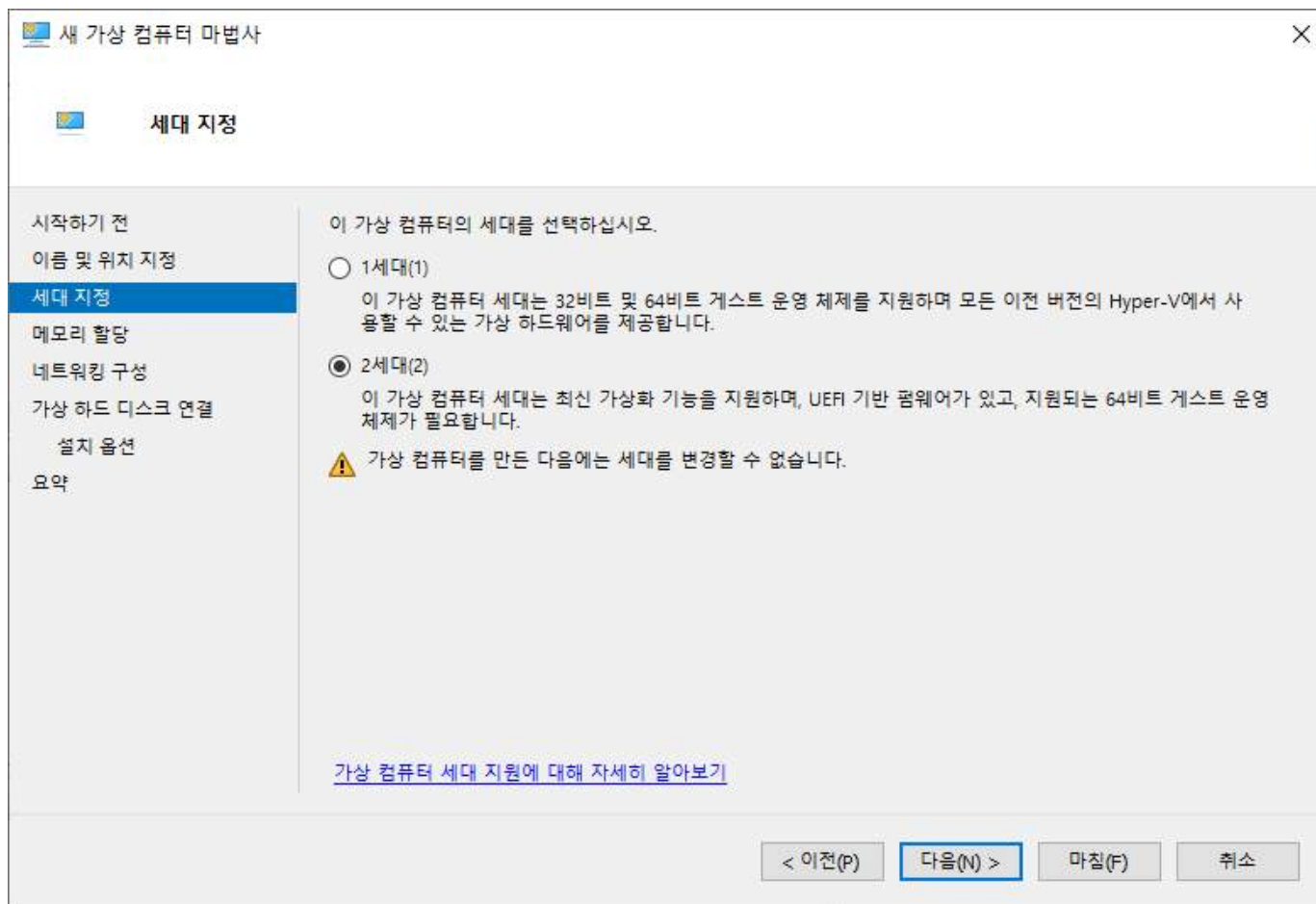
 이 가상 컴퓨터의 검사점을 만들려면 사용 가능한 공간이 충분한 위치를 선택합니다. 검사점에는 가상 컴퓨터 데이터가 포함되며 많은 공간이 필요할 수 있습니다.

< 이전(P)   다음(N) >   마침(F)   취소



# Windows VM Machine :: Hyper-V and WSL2

## Hyper-V and CentOS 8





# Windows VM Machine :: Hyper-V and WSL2

## Hyper-V and CentOS 8

새 가상 컴퓨터 마법사

메모리 할당

시작하기 전  
이름 및 위치 지정  
세대 지정  
**메모리 할당**  
네트워킹 구성  
가상 하드 디스크 연결  
설치 옵션  
요약

이 가상 컴퓨터에 할당할 메모리 용량을 지정하십시오. 32MB에서 251658240MB까지 지정할 수 있습니다. 성능을 높이려면 운영 체제에 대해 권장되는 최소 용량보다 크게 지정하십시오.

시작 메모리(M):  MB

☒ 이 가상 컴퓨터에 동적 메모리를 사용합니다(U).

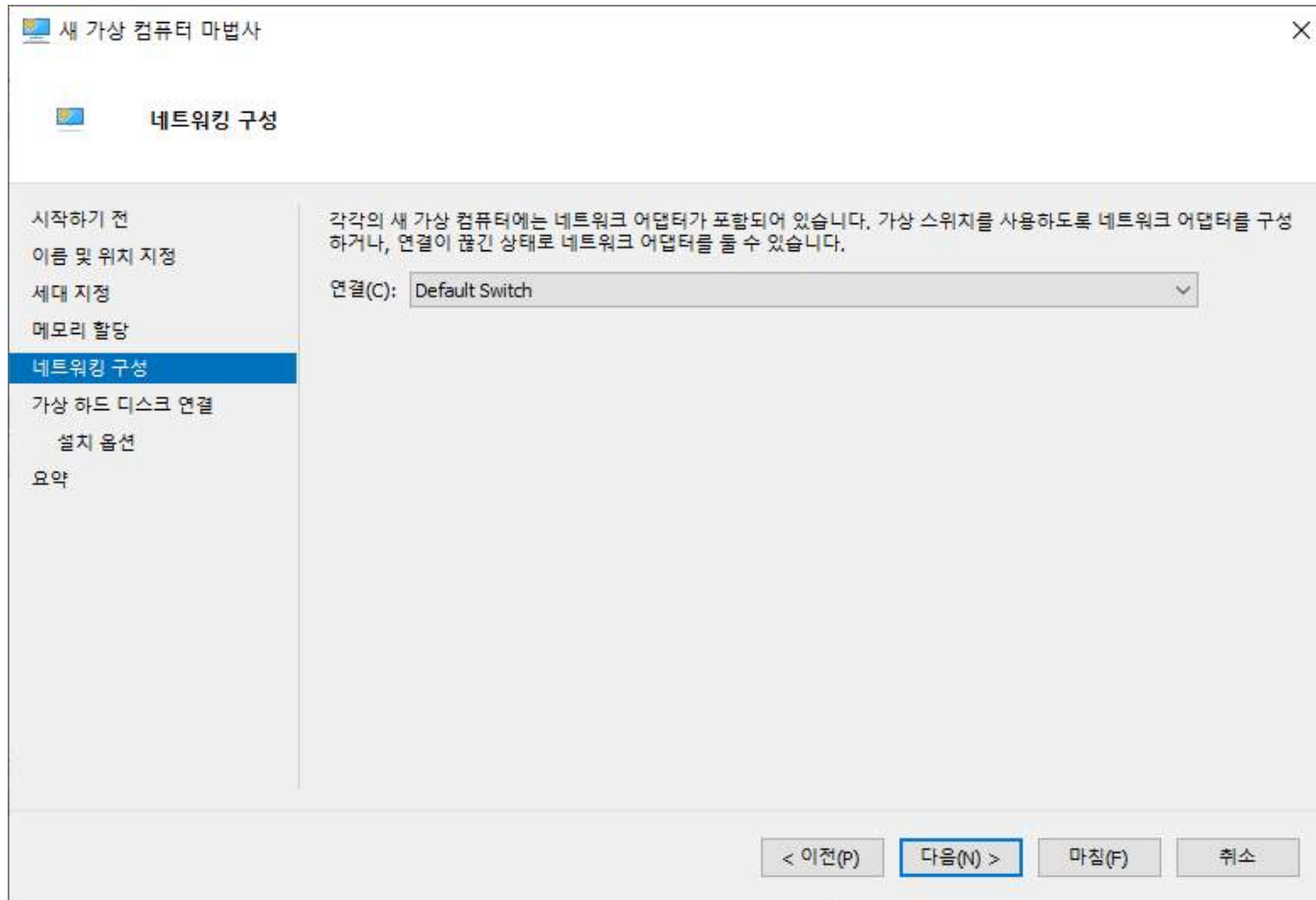
**i** 가상 컴퓨터에 할당할 메모리 양을 결정할 때 가상 컴퓨터를 사용할 목적과 실행할 운영 체제를 고려하십시오.

< 이전(P)   다음(N) >   마침(F)   취소



# Windows VM Machine :: Hyper-V and WSL2

## Hyper-V and CentOS 8





# Windows VM Machine :: Hyper-V and WSL2

## Hyper-V and CentOS 8

새 가상 컴퓨터 마법사

가상 하드 디스크 연결

시작하기 전  
이름 및 위치 지정  
세대 지정  
메모리 할당  
네트워킹 구성  
**가상 하드 디스크 연결**  
설치 옵션  
요약

가상 컴퓨터에는 운영 체제를 설치할 수 있는 저장소가 있어야 합니다. 저장소를 지금 지정하거나 가상 컴퓨터의 속성을 수정하여 나중에 구성할 수 있습니다.

☒ 가상 하드 디스크 만들기(C)  
VHDX 형식의 동적으로 확장되는 가상 하드 디스크를 만들려면 이 옵션을 사용하십시오.

이름(M):   
위치(L):  [찾아보기\(B\)...](#)  
크기(S):  GB(최대: 64TB)

☐ 기존 가상 하드 디스크 사용(U)  
기존 VHDX 가상 하드 디스크를 연결하려면 이 옵션을 사용하십시오.

위치(L):  [찾아보기\(B\)...](#)

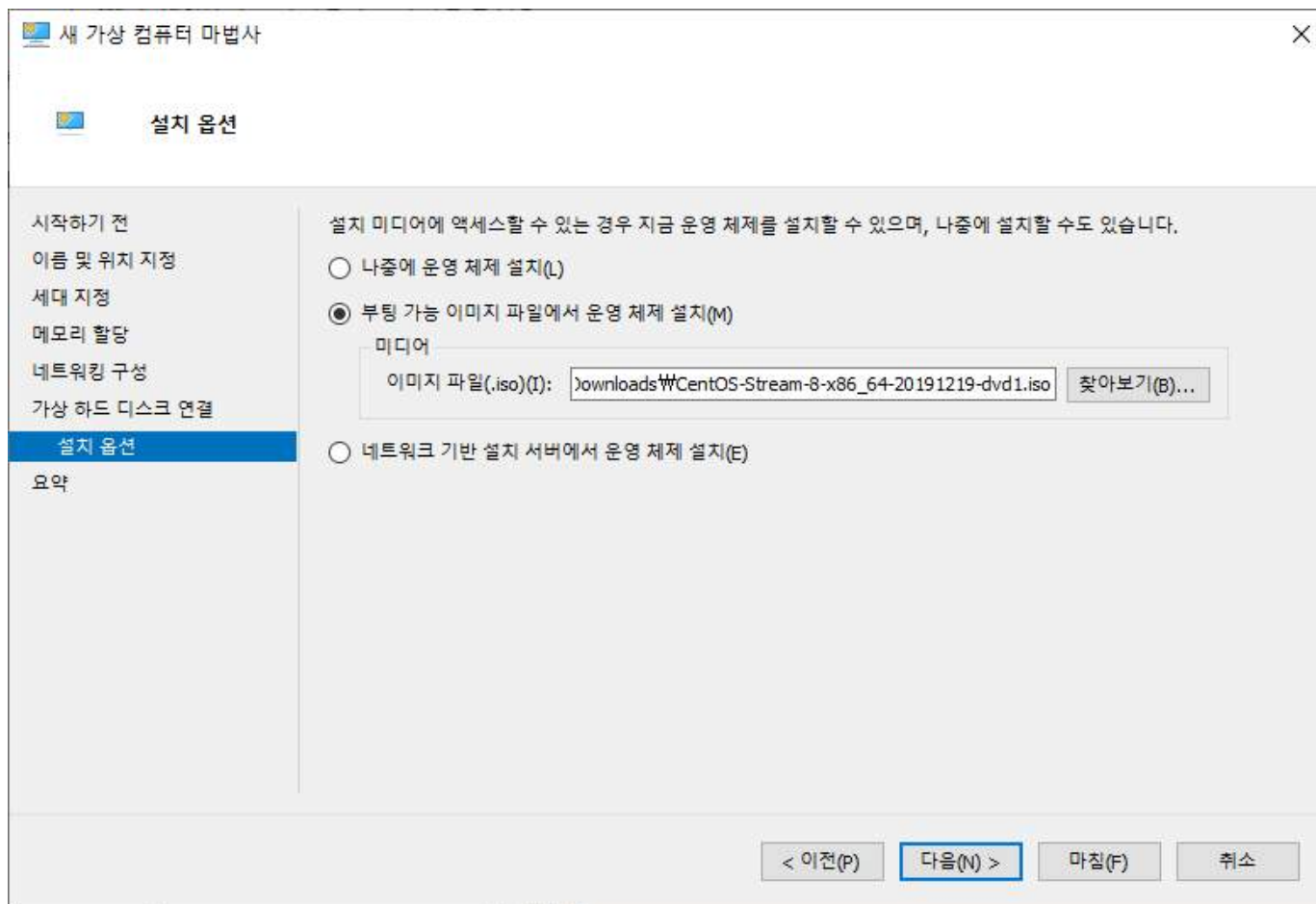
☐ 나중에 가상 하드 디스크 연결(A)  
지금 이 단계를 건너뛰고 나중에 기존 가상 하드 디스크를 연결하려면 이 옵션을 사용하십시오.

< 이전(P)   다음(N) >   마침(F)   취소



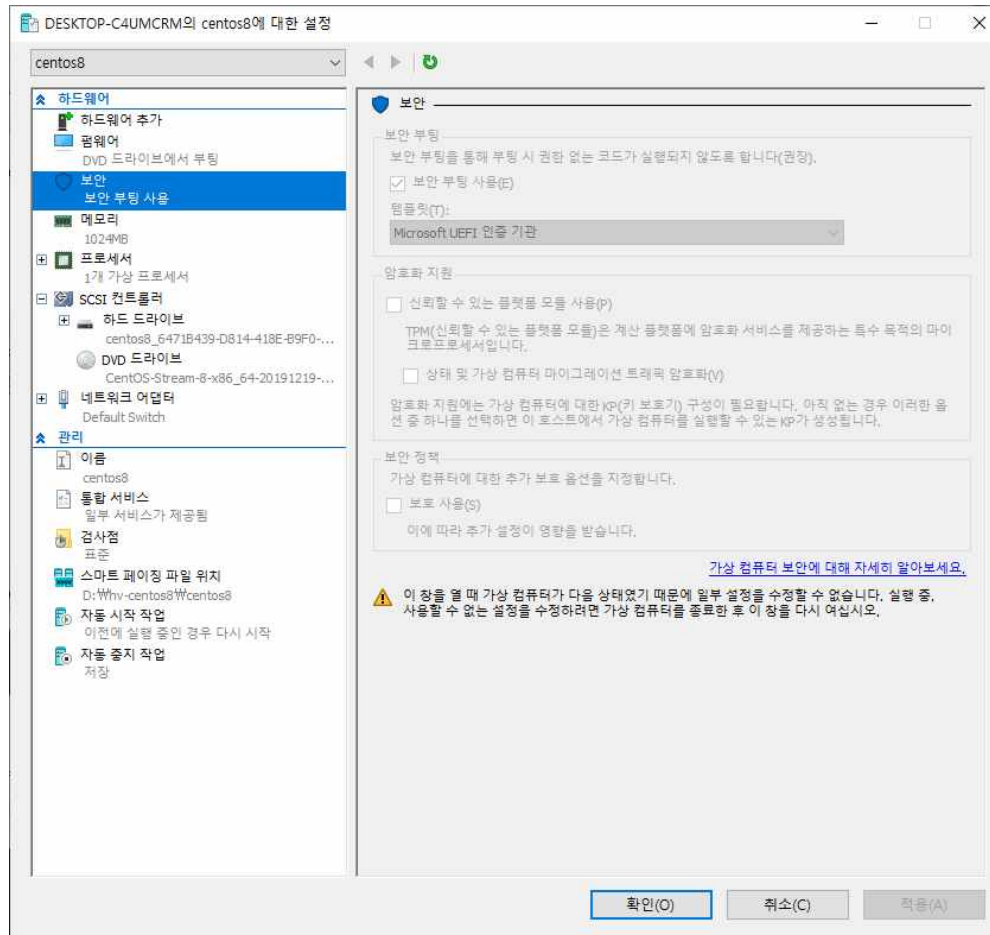
# Windows VM Machine :: Hyper-V and WSL2

## Hyper-V and CentOS 8



# Windows VM Machine :: Hyper-V and WSL2

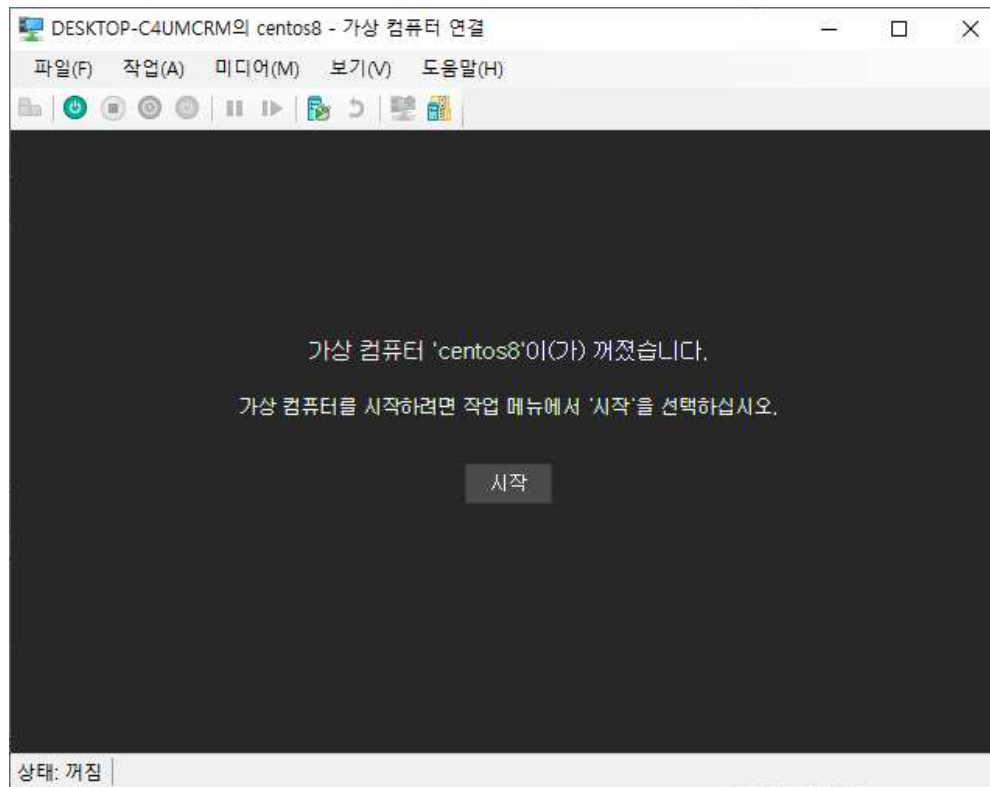
## Hyper-V and CentOS 8





# Windows VM Machine :: Hyper-V and WSL2

## Hyper-V and CentOS 8







# Windows VM Machine :: Hyper-V and WSL2

## CentOS 8 한국어 설치

-설정>지역 및 언어>입력소스>"한국어(hangul)"

-"한국어(hangul)" 없으면

-yum update

-update 과정에서 실패하면 한번더 yum update

\*우분투일 경우 apt를 쓴다. 패키지 관리자 yum(or dnf)은 레드햇 계열, apt는 데비안 계열

-재부팅 shutdown -r now

-설정>지역 및 언어>입력소스>"한국어(hangul)"

-"한국어(hangul)" 없으면

-dnf install ibus-hangul

-재부팅 shutdown -r now

-설정>지역 및 언어>입력소스>"한국어(hangul)"

# Part 2

리눅스 시작하기



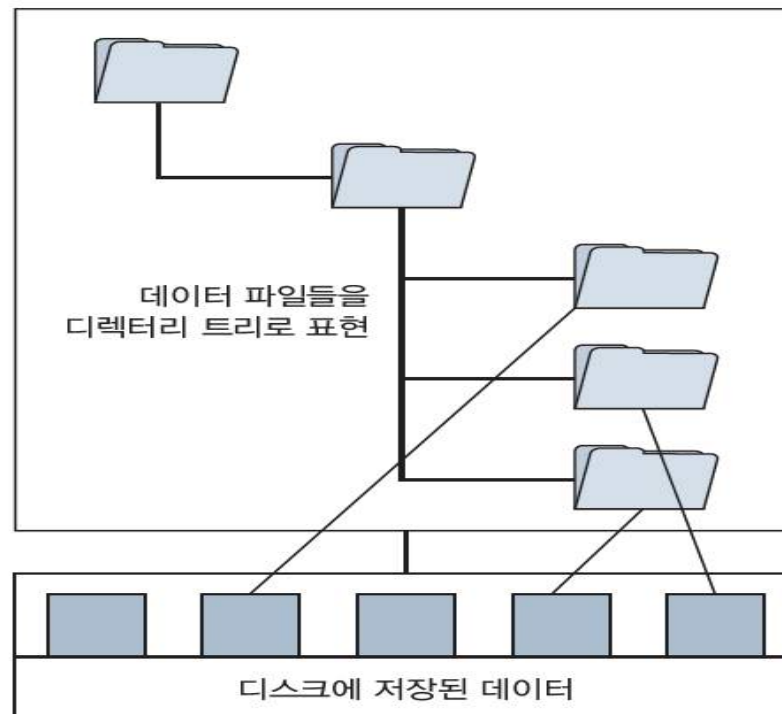


# 리눅스 시작하기

## 파일 시스템

- 물리적 디스크의 식별할 수 있는 고유 위치에 저장된 데이터와 논리적 개념 단위인 파일을 연결하는 일종의 데이터 테이블이나 인덱스(index)

OS는 저장 장치에 있는 원시 데이터를 구조화된 디렉터리 계층 구조로 시각화해 표현한다





# 리눅스 시작하기

## 인덱스가 필요한 이유

- 디지털 저장 장치에서 파일의 물리적인 위치가 만들어진 순서와 일관성이 없으며 시간이 지나면 바뀔 수도 있다
- 데이터를 신뢰성 있게 읽으려면 찾으려는 리소스를 일관되게 가리킬 수 있는 일종의 인덱스가 필요하다
- 파일 시스템은 이 인덱스를 사용해 파티션(partition)이라는 단일 디스크 분할 영역 안에서 구성된 디렉터리와 파일들을 제공한다

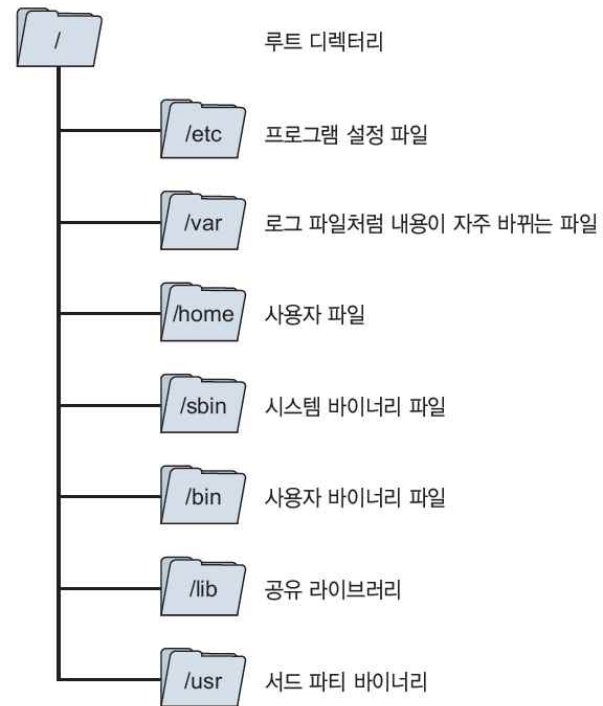


# 리눅스 시작하기

## 디렉토리를 구성하는 방법

- 유닉스 파일 시스템 계층 구조 표준(FHS, Filesystem Hierarchy Standard)을 따른다
- 리눅스 배포판, 유닉스 또는 macOS 모두 FHS를 따르므로 기본적인 구조는 서로 상당히 비슷하다
- 루트 디렉터리는 슬래시(/)로 표현한다
- 루트 디렉터리 바로 아래 디렉터리들을 최상위 디렉터리라고 하며 etc, var, home 등이 있다

## 유닉스 FHS에서 정의한 최상위 디렉터리





# 리눅스 시작하기

## ls(리스트)

현재 디렉터리에 있는 파일과 그 하위 디렉터리들의 이름이 나열

\$ ls -l : 파일명뿐만 아니라 접근 권한, 소유자, 그룹, 파일 크기, 최종 수정 날짜도 출력

\$ ls -l /var : /var과 같이 특정 디렉터리를 지정하면 해당 디렉터리의 내용이 출력

\$ ls -l : 파일 크기를 바이트 단위로 출력

\$ ls -lh : 킬로바이트나 메가바이트 또는 기가바이트 단위처럼 읽기 쉬운 형식으로 출력

\$ ls -R : 현재 디렉터리 아래에 어떤 디렉터리가 있는지 확인

## pwd(현재 작업 디렉터리)

- 현재 작업 디렉터리를 화면에 출력

\$ pwd

## cd(디렉터리 변경)

- 명령줄 인터프리터(일반적으로 Bash를 사용한다)가 현재 작업 디렉터리를 지정한 디렉터리로 바꿈
- \$ cd .. : 상위 디렉터리로 이동
- \$ cd /home/<사용자 계정> : 현재 디렉터리에서 아주 멀리 떨어져 있는 곳으로 가려면 절대 경로를 사용하는 편이 좋다



# 리눅스 시작하기

## Bash란 무엇인가?

Bash : 유닉스에서 가장 인기 있는 셸

셸(shell) : 명령줄 인터페이스(CLI, Command-line Interface)나 그래픽 사용자 인터페이스(GUI, Graphical User Interface)로 제공되며, 사용자의 명령을 해석해 실행하는 일을 한다

## cat(파일 내용 출력)

- 연결한다는 뜻의 concatenate에서 유래, 파일 내용을 터미널에 출력(편집할 수는 없음)
- `$ cat /etc/fstab` : etc 디렉터리에 있는 fstab 파일 출력(이처럼 짧은 파일을 볼 때 좋음)

## less(파일 내용 표시)

- 파일 내용을 조금씩 보여주어 텍스트 내용을 읽기 쉽게 함
- `$ less /etc/services` : less 명령 뒤에 읽으려는 파일의 이름을 지정





# 리눅스 시작하기

## touch(빈 파일 생성)

- 기존 파일에 touch 명령을 실행하면 파일 내용은 그대로 두고 타임스탬프만 갱신
- 파일을 나열하거나(ls) 파일 내용을 출력하는(cat) 등의 다양한 명령이 작동하는 방법을 변경할 때 유용

## 편집기의 종류

- 일반 텍스트 편집기(plain text editor) : gedit(우분투에서는 텍스트 편집기). 코드와 스크립트를 보기 좋게 해주는 텍스트 강조 도구도 아주 많은데, 실제 파일에는 텍스트만 저장
- 명령줄 편집기(command-line editor) : nano나 Pico처럼 직관적인 인터페이스를 제공
- Vim 편집기





# 리눅스 시작하기

## 디렉터리의 생성과 삭제

- `inode` : 메타데이터의 집합
- `$ stat` : `inode`와 같은 더 자세한 정보를 볼 때 사용. 파일명, 속성, 타임스탬프뿐만 아니라 `inode ID` 번호도 보여줌
- `$ mkdir` : 디렉터리 생성
- `$ rmdir` : 디렉터리 삭제
- `$ rm -r` : “Directory not empty(디렉터리가 비어 있지 않다)” 오류 메시지가 뜰 때
  - 주의 : 디렉터리 아래에 필요한 파일이 없다고 100% 자신할 수 있을 때 사용할 것



# 리눅스 시작하기

## 파일 복사와 이동

- \$ cp : 파일이나 디렉터리 사본 생성
- \$ mv : 파일을 한 장소에서 다른 장소로 이동

## 파일 글로빙(globbing)

- 명령이 처리할 파일명에 와일드카드 문자를 적용하는 것을 말함
- 별표(\*) : 여러 파일을 이동하거나 복사할 때 파일명을 일일이 입력하고 싶지 않을 때
- 물음표(?) : 임의의 문자 하나에 대응. 예를 들어, 파일명이 file1, file2, ..., file15인 파일들이 있을 때 file1에서 file9까지만 옮기고 싶은 경우

## 파일 삭제

- \$ rm : 파일 삭제. 원래대로 되돌릴 수 없음
- \$ rm -r \* : 이 명령을 지정한 경로 아래에 있는 모든 하위 디렉터리에 반복 적용해 내용 삭제



# 리눅스 시작하기

## 맨 페이지

\$ man : 명령줄에서 man 뒤에 알고 싶은 명령을 입력하면 해당 명령의 맨 페이지(man page)를 보여줌

## Info

\$ info : 명령 이름을 모를 때 명령줄에서 info 명령을 입력하면 Bash 표준에 따른 대화형 환경으로 넘어감

## 인터넷

서버를 사용하는지, 브라우저에 어떤 에러 메시지가 나왔는지, 크래시가 어떤 로그를 남겼는지 등 구체적으로 검색

# journalctl : 시스템 로그에서 오류 정보 가져오기



# 리눅스 시작하기

## Vim 사용법

- 명령 모드 (command mode)
- 입력 모드 (insert mode)
- EX 모드 (ex mode)
- 비주얼 모드 (visual mode)

## Vim 시작하기

\$ vi filename



# 리눅스 시작하기

## Vim 사용법 - EX 모드

KEY	동작
:w	현재 변경사항 저장
:wq	현재 변경사항 저장 후 나가기
:q!	저장하지 않고 나가기
:q	나가기 (변경사항이 있다면 오류가 발생하고 나가지지 않음)
:!bash	(편집 상태를 그대로 두고 bash 셸 실행. 재 진입하려면 exit 후 엔터)
(대문자) ZZ	현재 변경사항 저장 후 나가기



# 리눅스 시작하기

## Vim 사용법 - 입력 모드

KEY	동작
i	명령 모드에서 입력 모드로 전환 (현재 커서의 위치에)
l	명령 모드에서 입력 모드로 전환 (현재 커서 줄의 맨앞 위치에)
a	명령 모드에서 입력 모드로 전환 (현재 커서의 바로 뒤 위치에)
A	명령 모드에서 입력 모드로 전환 (현재 커서 줄의 맨뒤 위치에)



# 리눅스 시작하기

## Vim 사용법 - 입력 모드

KEY	동작
i	명령 모드에서 입력 모드로 전환 (현재 커서의 위치에)
l	명령 모드에서 입력 모드로 전환 (현재 커서 줄의 맨앞 위치에)
a	명령 모드에서 입력 모드로 전환 (현재 커서의 바로 뒤 위치에)
A	명령 모드에서 입력 모드로 전환 (현재 커서 줄의 맨뒤 위치에)

## Vim 사용법 - 입력모드 1단계 되돌리기

KEY	동작
u	한 단계 되돌리기 (반복입력 가능)



# 리눅스 시작하기

## Vim 사용법 - 주요 명령어

KEY	동작
dw	커서 위치로부터 다음 단어 첫글자 전까지 지우기

KEY	동작
dd	커서 위치의 줄 삭제
2dd	커서 위치로부터 2줄 삭제
D	커서 위치로부터 줄 끝까지 삭제

KEY	동작
yy	커서 위치의 줄 버퍼로 복사
2yy	커서 위치로부터 2줄 버퍼로 복사
p	버퍼에 담긴 내용을 커서 다음 줄에 삽입
P	버퍼에 담긴 내용을 커서 이전 줄에 삽입





# 리눅스 시작하기

## Vim 사용법 - 주요 명령어

KEY	동작
/helloworld	문서 내의 helloworld 를 찾아서 이동
n	(검색 이동 상태에서) 다음 일치 항목으로 이동
N	(검색 이동 상태에서) 이전 일치 항목으로 이동

**:%s/HHH/byebye/ig**

뒤에 i 는 대소문자 구분안함(case Insensitive). 없을 경우 대소문자 일치가 일치 한 것만 변경.  
뒤에 g 는 줄 내 모두변경(Global). 없을 경우 라인에서 첫번째 일치 항목만 변경.



# 리눅스 시작하기

## CentOS의 bash 셸

- 기본 셸은 bash(Bourne Again SHell : ‘배시 셸’)
- bash 셸의 특징
  - Alias 기능(명령어 단축 기능)
  - History 기능(위/아래 화살표키)
  - 연산 기능
  - Job Control 기능
  - 자동 이름 완성 기능(탭키)
  - 프롬프트 제어 기능
  - 명령 편집 기능
- 셸의 명령문 처리 방법
  - (프롬프트) 명령어 [옵션...] [인자...]
  - 예) `# rm -rf /mydir`

# >>>> 리눅스 시작하기

## 환경 변수

- “echo \$환경변수이름” 으로 확인 가능
- “export 환경변수=값” 으로 환경 변수의 값을 변경
- 주요 환경변수

환경 변수	설명	환경 변수	설명
HOME	현재 사용자의 홈 디렉터리	PATH	실행 파일을 찾는 디렉터리 경로
LANG	기본 지원되는 언어	PWD	사용자의 현재 작업 디렉터리
TERM	로그인 터미널 타입	SHELL	로그인해서 사용하는 셸
USER	현재 사용자의 이름	DISPLAY	X 디스플레이 이름
COLUMNS	현재 터미널의 컬럼 수	LINES	현재 터미널 라인 수
PS1	1차 명령 프롬프트 변수	PS2	2차 명령 프롬프트(대개는 `>`)
BASH	bash 셸의 경로	BASH_VERSION	bash 버전
HISTFILE	히스토리 파일의 경로	HISTSIZE	히스토리 파일에 저장되는 개수
HOSTNAME	호스트의 이름	USERNAME	현재 사용자 이름
LOGNAME	로그인 이름	LS_COLORS	ls 명령어의 확장자 색상 옵션
MAIL	메일을 보관하는 경로	OSTYPE	운영체제 타입



# 리눅스 시작하기

## 셸 스크립트 프로그래밍

- C언어와 유사하게 프로그래밍이 가능
- 변수, 반복문, 제어문 등의 사용이 가능
- 별도로 컴파일하지 않고 텍스트 파일 형태로 바로 실행
- vi나 gedit으로 작성이 가능
- 리눅스의 많은 부분이 셸 스크립트로 작성되어 있음
  - ① “sh <스크립트 파일>”로 실행
  - ② “chmod +x <스크립트 파일>” 명령으로 실행 가능 속성으로 변경한 후에,
  - ③ “./<스크립트파일>”명령으로 실행



# 리눅스 시작하기

## 셸 스크립트 프로그래밍

- C언어와 유사하게 프로그래밍이 가능
- 변수, 반복문, 제어문 등의 사용이 가능
- 별도로 컴파일하지 않고 텍스트 파일 형태로 바로 실행
- vi나 gedit으로 작성이 가능
- 리눅스의 많은 부분이 셸 스크립트로 작성되어 있음
  - ① “sh <스크립트 파일>”로 실행
  - ② “chmod +x <스크립트 파일>” 명령으로 실행 가능 속성으로 변경한 후에,
  - ③ “./<스크립트파일>”명령으로 실행



# 리눅스 시작하기

## 변수의 기본

- 변수를 사용하기 전에 미리 선언하지 않으며, 변수에 처음 값이 할당되면서 자동으로 변수가 생성
- 모든 변수는 '문자열(String)'로 취급
- 변수 이름은 대소문자를 구분
- 변수를 대입할 때 '='좌우에는 공백이 없어야 함



# 리눅스 시작하기

## 변수의 입력과 출력

- '\$' 문자가 들어간 글자를 출력하려면 ' '로 묶어주거나 앞에 '\ '를 붙임.
- “ ”로 변수를 묶어줘도 된다.

```
01 #!/bin/sh
```

```
02 myvar="Hi Woo"
```

```
03 echo $myvar
```

```
04 echo "$myvar"
```

```
05 echo '$myvar'
```

```
06 echo \ $myvar
```

```
07 echo 값 입력 :
```

```
08 read myvar
```

```
09 echo '$myvar' = $myvar
```

```
10 exit 0
```



# 리눅스 시작하기

## 숫자 계산

- 변수에 대입된 값은 모두 문자열로 취급
- 변수에 들어 있는 값을 숫자로 해서 +, -, \*, / 등의 연산을 하려면 `expr`을 사용
- 수식에 괄호 또는 곱하기(\*)는 그 앞에 꼭 역슬래시(\) 붙임

```
01 #!/bin/sh
02 num1=100
03 num2=$num1+200
04 echo $num2
05 num3='expr $num1 + 200'
06 echo $num3
07 num4='expr \($num1 + 200\) / 10 \* 2'
08 echo $num4'
09 exit 0
```





# 리눅스 시작하기

## 파라미터(Parameter) 변수

- 파라미터 변수는 \$0, \$1, \$2...의 형태를 가짐
- 전체 파라미터는 \$\*로 표현

예)

```
01 #!/bin/sh
```

```
02 echo "실행파일 이름은 <$0>이다"
```

```
03 echo "첫번째 파라미터는 <$1>이고, 두번째 파라미터는 <$2>다"
```

```
04 echo "전체 파라미터는 <$*>다"
```

```
05 exit 0
```



# 리눅스 시작하기

## 파라미터(Parameter) 변수

- 파라미터 변수는 \$0, \$1, \$2...의 형태를 가짐
- 전체 파라미터는 \$\*로 표현

예)

```
01 #!/bin/sh
```

```
02 echo "실행파일 이름은 <$0>이다"
```

```
03 echo "첫번째 파라미터는 <$1>이고, 두번째 파라미터는 <$2>다"
```

```
04 echo "전체 파라미터는 <$*>다"
```

```
05 exit 0
```

# >>>> 리눅스 시작하기

## 기본 if 문

- 형식

```
if [ 조건 ]
then
    참일 경우 실행
fi
```
- ```
01 #!/bin/sh
02 if [ "woo" = "woo" ]
03 then
04     echo "참입니다"
05 fi
06 exit 0
```

## if~else 문

- 형식

```
if [ 조건 ]
then
    참일 경우 실행
else
    거짓인 경우 실행
fi
```
- ```
01 #!/bin/sh
02 if [ "woo" != "woo" ]
03 then
04     echo "참입니다"
05 else
06     echo "거짓입니다"
07 fi
08 exit 0
```

# >>>> 리눅스 시작하기

## 조건문에 들어가는 비교 연산자

문자열 비교	결과
"문자열1" = "문자열2"	두 문자열이 같으면 참
"문자열1" != "문자열2"	두 문자열이 같지 않으면 참
-n "문자열"	문자열이 NULL(빈 문자열)이 아니면 참
-z "문자열"	문자열이 NULL(빈 문자열)이면 참

산술 비교	결과
수식1 -eq 수식2	두 수식(또는 변수)이 같으면 참
수식1 -ne 수식2	두 수식(또는 변수)이 같지 않으면 참
수식1 -gt 수식2	수식1이 크다면 참
수식1 -ge 수식2	수식1이 크거나 같으면 참
수식1 -lt 수식2	수식1이 작으면 참
수식1 -le 수식2	수식1이 작거나 같으면 참
!수식	수식이 거짓이면 참

```
01 #!/bin/sh
02 if [ 100 -eq 200 ]
03 then
04     echo "100과 200은 같다."
05 else
06     echo "100과 200은 다르다."
07 fi
08 exit 0
```

# >>>> 리눅스 시작하기

## 파일과 관련된 조건

```
01 #!/bin/sh
02 fname=/lib/systemd/system
   /sshd.service
03 if [ -f $fname ]
04 then
05     head -5 $fname
06 else
07     echo "sshd 서버가 설치되지 않았습니다."
08 fi
09 exit 0
```

파일 조건	결과
-d 파일이름	파일이 디렉터리면 참
-e 파일이름	파일이 존재하면 참
-f 파일이름	파일이 일반 파일이면 참
-g 파일이름	파일에 set-group-id가 설정되면 참
-r 파일이름	파일이 읽기 가능이면 참
-s 파일이름	파일 크기가 0이 아니면 참
-u 파일이름	파일에 set-user-id가 설정되면 참
-w 파일이름	파일이 쓰기 가능 상태면 참
-x 파일이름	파일이 실행 가능 상태면 참

# >>>> 리눅스 시작하기

## case~esac 문 (1)

- if 문은 참과 거짓의 두 경우만 사용 (2중분기)
- 여러 가지 경우의 수가 있다면 case 문 (다중분기)

```
01 #!/bin/sh
02 case "$1" in
03     start)
04         echo "시작~~";;
05     stop)
06         echo "중지~~";;
07     restart)
08         echo "다시 시작~~";;
09     *)
10         echo "뭔지 모름~~";;
11 esac
12 exit 0
```



## 리눅스 시작하기

### case~esac 문 (2)

```
01 #!/bin/sh
02 echo "리눅스가 재미있나요? (yes / no)"
03 read answer
04 case $answer in
05     yes | y | Y | Yes | YES)
06         echo "다행입니다."
07         echo "더욱 열심히 하세요 ^^";;
08     [nN]*)
09         echo "안타깝네요. ㅠㅠ";;
10     *)
11         echo "yes 아니면 no만
                입력했어야죠"
12         exit 1;;
13 esac
14 exit 0
```



# 리눅스 시작하기

## AND, OR 관계 연산자

- and는 '-a' 또는 '&&'를 사용
- or는 '-o' 또는 '||'를 사용

```
01 #!/bin/sh
02 echo "보고 싶은 파일명을 입력하세요."
03 read fname
04 if [ -f $fname ] && [ -s $fname ] ; then
05     head -5 $fname
06 else
07     echo "파일이 없거나,
           크기가 0입니다."
08 fi
09 exit 0
```



# >>>> 리눅스 시작하기

## 반복문 - for문 (1)

- 형식  
for 변수 in 값1 값2 값3 ...  
do  
    반복할 문장  
done

```
01 #!/bin/sh
02 hap=0
03 for i in 1 2 3 4 5 6 7 8 9 10
04 do
05     hap='expr $hap + $i'
06 done
07 echo "1부터 10까지의 합: "$hap
08 exit 0
```

# >>>> 리눅스 시작하기

## 반복문 - for문 (2)

- 현재 디렉터리에 있는 셸 스크립트 파일(\*.sh)의 파일명과 앞 3줄을 출력하는 프로그램

```
01 #!/bin/sh
02 for fname in $(ls *.sh)
03 do
04     echo "-----$fname-----"
05     head -3 $fname
06 done
07 exit 0
```

# 리눅스 시작하기

## 반복문 - while문 (1)

- 조건식이 참인 동안에 계속 반복

```
01 #!/bin/sh
```

```
02 while [ 1 ]
```

```
03 do
```

```
04   echo “이것이 리눅스다. ^^”
```

```
05 done
```

```
06 exit 0
```

# >>>> 리눅스 시작하기

## 반복문 - while문 (2)

- 1에서 10까지의 합계를 출력 ('반복문 - for문 (1)' 슬라이드와 동일)

```
01 #!/bin/sh
02 hap=0
03 i=1
04 while [ $i -le 10 ]
05 do
06     hap='expr $hap + $i'
07     i='expr $i + 1'
08 done
09 echo "1부터 10까지의 합 : "$hap
10 exit 0
```



## 리눅스 시작하기

### 반복문 - while문 (3)

- 비밀번호를 입력받고, 비밀번호가 맞을 때까지 계속 입력받는 스크립트

```
01 #!/bin/sh
02 echo "비밀번호를 입력하세요."
03 read mypass
04 while [ $mypass != "1234" ]
05 do
06     echo "틀렸음. 다시 입력하세요."
07     read mypass
08 done
09 echo "통과~~"
10 exit 0
```



## 리눅스 시작하기

### 반복문 - while문 (3)

- 비밀번호를 입력받고, 비밀번호가 맞을 때까지 계속 입력받는 스크립트

```
01 #!/bin/sh
02 echo "비밀번호를 입력하세요."
03 read mypass
04 while [ $mypass != "1234" ]
05 do
06     echo "틀렸음. 다시 입력하세요."
07     read mypass
08 done
09 echo "통과~~"
10 exit 0
```

# >>>> 리눅스 시작하기

## until 문

- while문과 용도가 거의 같지만, until문은 조건식이 참일 때까지(=거짓인 동안) 계속 반복한다.
- while2.sh를 동일한 용도로 until문으로 바꾸려면 4행을 다음과 같이 바꾸면 된다.
  - `until [ $i -gt 10 ]`



# 리눅스 시작하기

## break, continue, exit, return 문

- break는 주로 반복문을 종료할 때 사용되며, continue는 반복문의 조건식으로 돌아가게 함. exit는 해당 프로그램을 완전히 종료함. Return은 함수 안에서 사용될 수 있으며 함수를 호출한 곳으로 돌아가게 함.

```
01 #!/bin/sh
02 echo "무한반복 입력을 시작합니다(b: break, c: continue, e: exit)"
03 while [ 1 ] ; do
04 read input
05 case $input in
06     b | B )
07         break ;;
08     c | C )
09         echo "continue를 누르면 while의 조건으로 돌아감"
10         continue ;;
11     e | E )
12         echo "exit를 누르면 프로그램(함수)를 완전히 종료함"
13         exit 1 ;;
14 esac;
15 done
16 echo "break를 누르면 while을 빠져나와 지금 이 문장이 출력됨."
17 exit 0
```





# 리눅스 시작하기

## 사용자 정의 함수

- 형식  
함수이름 ( ) { → 함수를 정의  
내용들...  
}  
함수이름 → 함수를 호출

```
01 #!/bin/sh
02 myFunction () {
03     echo "함수 안으로 들어 왔음"
04     return
05 }
06 echo "프로그램을 시작합니다."
07 myFunction
08 echo "프로그램을 종료합니다."
09 exit 0
```



# 리눅스 시작하기

## 함수의 파라미터 사용

- 형식  
함수이름 ( ) { → 함수를 정의  
\$1, \$2 ... 등을 사용  
}  
함수이름 파라미터1 파라미터2 ... → 함수를 호출

```
01 #!/bin/sh
02 hap () {
03     echo 'expr $1 + $2'
04 }
05 echo "10 더하기 20을 실행합니다"
06 hap 10 20
07 exit 0
```



## 리눅스 시작하기

### eval

- 문자열을 명령문으로 인식하고 실행

```
01 #!/bin/sh
```

```
02 str="ls -l eval.sh"
```

```
03 echo $str
```

```
04 eval $str
```

```
05 exit 0
```



## 리눅스 시작하기

### eval

- 문자열을 명령문으로 인식하고 실행

```
01 #!/bin/sh
```

```
02 str="ls -l eval.sh"
```

```
03 echo $str
```

```
04 eval $str
```

```
05 exit 0
```



# 리눅스 시작하기

## export

- 외부 변수로 선언해 준다. 즉, 선언한 변수를 다른 프로그램에서도 사용할 수 있도록 해줌

- exp1.sh

```
01 #!/bin/sh
02 echo $var1
03 echo $var2
04 exit 0
```

- exp2.sh

```
01 #!/bin/sh
02 var1="지역 변수"
03 export var2="외부 변수"
04 sh exp1.sh
05 exit 0
```



## 리눅스 시작하기

### printf

- C언어의 printf() 함수와 비슷하게 형식을 지정해서 출력

01 #!/bin/sh

02 var1=100.5

03 var2="재미있는 리눅스~~"

04 printf "%5.2f \n\n \t %s \n" \$var1 "\$var2"

05 exit

# >>>> 리눅스 시작하기

## set과 \$(명령어)

- 리눅스 명령어를 결과로 사용하기 위해서는 \$(명령어) 형식을 사용
- 결과를 파라미터로 사용하고자 할 때는 set과 함께 사용

01 #!/bin/sh

02 echo "오늘 날짜는 \$(date) 입니다."

03 set \$(date)

04 echo "오늘은 \$4 요일 입니다."

05 exit 0



## 리눅스 시작하기

### shift

- 파라미터 변수를 왼쪽으로 한 단계씩 아래로 쉬프트시킴

```
01 #!/bin/sh
02 myfunc() {
03     str=""
04     while [ "$1" != "" ] ; do
05         str="$str $1"
06         shift
07     done
08     echo $str
09 }
10 myfunc AAA BBB CCC DDD EEE FFF GGG HHH III JJJ KKK
11 exit 0
```



# Part 3

원격 연결





## 원격 연결

### FTP 개요

- FTP(File Transfer Protocol)는 파일을 전송하기 위한 서비스
- 웹에서 FTP의 고유 기능인 파일 전송을 편리하게 할 수 있게 되어서 예전보다 인기가 많이 떨어짐
- 파일 전송 자체를 위해서는 성능이 뛰어남
- `$ sudo dnf install vsftpd`
- `$ sudo vi /etc/vsftpd/vsftpd.conf`



## 원격 연결

### vsftpd.conf 옵션

- `anonymous_enable`: anonymous(익명) 사용자의 접속을 허가할지 설정
- `local_enable`: 로컬 사용자의 접속 허가 여부를 설정
- `write_enable`: 로컬 사용자가 저장, 삭제, 디렉터리 생성 등의 명령을 실행하게 할 것인지 설정 (anonymous 사용자는 해당 없음)
- `anon_upload_enable`: anonymous 사용자의 파일 업로드 허가 여부를 설정
- `anon_mkdir_write_enable`: anonymous 사용자의 디렉터리 생성 허가 여부를 설정
- `dirlist_enable`: 접속한 디렉터리의 파일 리스트를 보여줄지 설정
- `download_enable`: 다운로드의 허가 여부를 설정
- `listen_port`: FTP 서비스의 포트 번호를 설정(기본: 22번)



## 원격 연결

### FileZilla 설치 - Windows

<https://filezilla-project.org/>

- Download FileZilla Client

The screenshot shows the FileZilla website homepage. The header features the FileZilla logo and the tagline 'The free FTP solution'. A left sidebar contains navigation links for Home, FileZilla (Features, Screenshots, Download, Documentation), FileZilla Server (Download), Community (Forum, Project page, Wiki), General (Contact, License, Privacy Policy), Development (Source code, Nightly builds, Translations, Version history, Changelog, Issue tracker), and Other projects (libfilezilla, Octochess). The main content area is titled 'Overview' and includes a welcome message, support links (forums, wiki, bug and feature request trackers), and a section for 'Quick download links'. This section contains two buttons: 'Download FileZilla Client' (All platforms) and 'Download FileZilla Server' (Windows only). The 'Download FileZilla Client' button is highlighted with a red box. Below the buttons, there is a 'News' section with several entries, including '2016-03-16 - FileZilla Server 0.9.56.1 released' and '2016-03-16 - FileZilla Client 3.16.1 released', each with a list of fixed vulnerabilities and bugfixes. At the bottom, there are logos for PayPal and NCC.



# 원격 연결

## FileZilla로 FTP 서버 연결

FileZilla interface showing a successful connection to an FTP server at 172.26.250.100. The local site is C:\Users\wremap\ and the remote site is /root. The status bar indicates the connection is successful.

Local Site: C:\Users\wremap\

Remote Site: /root

파일명	크기	파일 유형	최종 수정	권한	소유자/그룹
..					
anaconda		파일 폴더	2019-09-19 오후 ...		
android		파일 폴더	2020-04-01 오후 ...		

28 파일 및 61 디렉터리. 총 크기: 98,624,220 바이트

파일명	크기	파일 유형	최종 수정	권한	소유자/그룹
..					
.cache		파일 폴더	2020-06-14 ...	drwx-----	root root
config		파일 폴더	2020-06-15 ...	drwx-----	root root

10 파일 및 4 디렉터리. 총 크기: 5,162 바이트

Server/Local File | Direction | Remote File | Size | Priority | Status

대기 파일 | 전송 실패 | 전송 성공

대기열: 비어있음

파일 속성 바꾸기

"공개" 디렉터리의 새로운 속성을 선택하십시오.

소유자 권한

☒ 읽기(R) ☐ 쓰기(W) ☒ 실행(E)

그룹 권한

☐ 읽기(A) ☐ 쓰기(R) ☐ 실행(X)

공개 권한

☐ 읽기(D) ☐ 쓰기(I) ☐ 실행(C)

숫자값(N)

원래 파일의 권한을 유지하려면 해당 자리에 x를 지정하면 됩니다.

☐ 하위 디렉터리로 이동(U)

☒ 모든 파일과 디렉터리에 적용(A)

☐ 파일에만 적용(F)

☐ 디렉터리에만 적용(L)

확인 취소



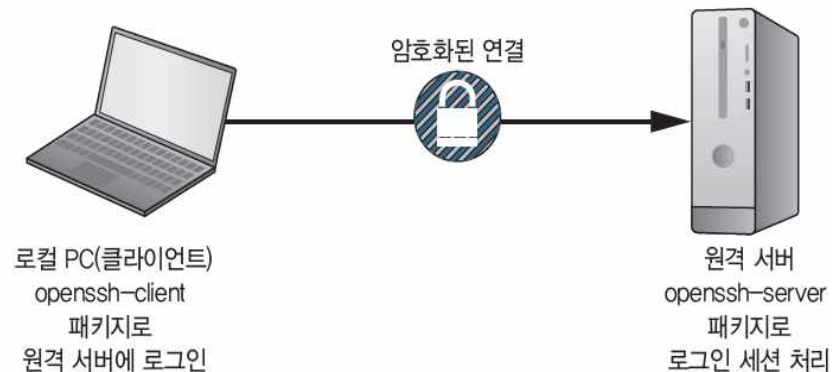
## 원격 연결

- 설치된 패키지 확인 : `rpm -qa | grep [패키지명]`  
`$ rpm -qa | grep ssh`
- 서버에서 실행되는지 확인 : `ps -ef | grep [프로그램명]`  
`$ ps -ef | grep ssh`  
`$ systemctl status sshd`
- 서버에 로그인한 상태라면 `ip addr` 명령으로 현재 컴퓨터의 공인 IP 주소를 가져옴  
`$ ip addr`
- 두 컴퓨터가 서로 통신할 수 있는지 `ping` 명령으로 확인  
`$ ping 255.255.255.255`
- Ctrl + C 를 누르면 ping 요청을 중단하고 명령줄로 돌아감



## 원격 연결

- 포트를 수정하고 싶으면 /etc/ssh/sshd\_config 파일 수정
- 포트 수정하면 방화벽 설정 해주어야 함  
\$ firewall-cmd --permanent --zone=public --add-port=[포트num]/tcp  
\$ firewall-cmd --reload
- \$ systemctl restart sshd  
\$ systemctl reload sshd
- netstat -nlpt | grep 22



# 원격 연결

## PuTTY 설치 - Windows

<https://www.chiark.greenend.org.uk/~sgtatham/putty/>

### Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

#### MSI ('Windows Installer')

32-bit: [putty-0.73-installer.msi](#) [\(or by FTP\)](#) [\(signature\)](#)

64-bit: [putty-64bit-0.73-installer.msi](#) [\(or by FTP\)](#) [\(signature\)](#)

#### Unix source archive

.tar.gz: [putty-0.73.tar.gz](#) [\(or by FTP\)](#) [\(signature\)](#)

