리눅스의 神

: 우분투 리눅스 입문편

리눅스의 神 : 우분투 리눅스 입문편

지은이 정준석
1판 1쇄 발행일 2016년 1월 25일

펴낸이 임성춘
펴낸곳 로드북
편집 장미경
디자인 허진하(표지), 박진희(본문)

주소 서울시 관악구 신림로29길 8 101-901호
출판 등록 제 2011-21호(2011년 3월 22일)
전화 02)874-7883
팩스 02)6280-6901
정가 20,000원
ISBN 978-89-97924-18-9 93000

이메일 chief@roadbook.co.kr
블로그 www.roadbook.co.kr

기초 명령어 요약집

- 터미널 종료하기
  `$ exit`
- 현재 콘솔에서 자신의 위치 확인하기
  `$ pwd`
- 상위 디렉토리로 이동하기
  `$ cd ..`
- 홈디렉토리로 이동하기
  `$ cd ~`
- 디렉토리 이동하기
  예 `$ cd jundols`
- 현재 디렉토리 파일과 서브 디렉토리 목록 보기
  `$ ls`
- 현재 디렉토리 파일과 서브 디렉토리 목록 자세히 보
  `$ ls -l`
- 파일 복사
  `$ cp 원본_파일 대상_파일`
  예 `$ cp /home/jundols/examples.desktop /home/jundols/test.desktop`
  `$ cp ./examples.desktop ./test.desktop`
- 파일 이동
  예 `$ mv ./examples.desktop ./test.desktop`
- 디렉토리 복사
  예 `$ cp -r ./Music ./MusicCopy`
- 파일 삭제
  예 `$ rm ./test.desktop`
- 디렉토리 삭제
  예 `$ rm -rf ./MusicCopy`
- 디렉토리 생성과 삭제
  예 `$ mkdir newdir`
  `$ rmdir dirname`
- 명령어 자동 완성 기능
  예 `$ rm ex<tab>`
- 특수기호 활용하기
  예 `$ cp test* ./subdir`
  예 `$ mv * ./subdir`
  예 `$ cp test? ./subdir`
- 프로그램을 콘솔에서 설치하
  예 `$ sudo apt-get install docky`
- 패키지 정보를 최신으로 갱신하기
  `$ sudo apt-get update`
- 우분투 버전 업그레이드
  `$ sudo apt-get dist-upgrade`
- 갱신된 패키지 업그레이드
  `$ sudo apt-get upgrade`
- 패키지 정보 보기
  `$ apt-cache show 패키지명`
- 패키지를 새로 설치하기
  `$ sudo apt-get install 패키지명`
- 패키지 제거하기
  `$ sudo apt-get remove 패키지명`
- 사용하지 않는 패키지 설치파일 자동 제거하기
  `$ sudo apt-get autoclean`
- 설치된 패키지 목록 보기
  `$ apt-cache pkgnames`
- 네트워크 연결에 문제 없나 확인하기
  예 `$ ping google.com`
- 네트워크 장치 목록 보여주기
  `$ ifconfig`
- 네트워크 연결 상태 모니터링하기
  `$ netstat -at`
- 인터넷 라우팅 테이블 정보 보기
  `$ route`
- 시스템 종료하기
  `$ sudo reboot`
- 시스템 재시작하기
  `$ sudo shutdown now`
  또는
  `$ sudo halt`
- 일정 시간 뒤 종료하기
  예 `$ sudo shutdown -h 10` (10분 뒤에 종료)
  예 `$ sudo shutdown -h 8:30` (8시 30분에 종료)
- 일정 시간 뒤에 재시작하기
  예 `$ sudo shutdown -r 10` (10분 뒤에 재시작)
  예 `$ sudo shutdown -r 8:30` (8시 30분에 재시작)
- 종료 또는 재시작 예약 취소
  `$ sudo shutdown -c`
- 저장 장치의 다양한 정보 보기
  `$ df`
- df 명령어보다 세부적으로 보기
  `$ mount`
- 파일 실행권한 변경하기
  예 `$ chmod u-x Music`
  (Music 디렉토리의 소유자 권한 중에서 실행권한 제거)
  예 `$ chmod go+rwx Music`
  (Music 디렉토리의 소유 그룹과 제3자에게 읽기 쓰기 실행권한 모두 부여)
  예 `$ chmod o-xw Music`
  (Music 디렉토리의 제3자에게 실행과 쓰기 권한 제거)
- 숫자로 실행권한 변경하기
  예 `$ chmod 755 Music`
  (소유자에게는 권한 전부를 부여하고 소유그룹과 제3자에게는 읽기와 실행 권한만 부여)
  예 `$ chmod 300 Music`
  (소유자는 쓰기와 실행 권한, 소유 그룹과 제 3자에게는 어떤 권한도 없음)

- 파일 소유자 바꾸기
  예 $ sudo chown newuser:newgroup target.file
  (파일 소유자는 newuser로 소유 그룹은 newgroup으로 바꾸기)
- 현재 및 하위에 있는 모든 파일과 디렉토리 소유자 바꾸기
  예 $ sudo chown aaa:bbb -r *
  (유자를 aaa로 바꾸고 소유 그룹을 bbb로 바꾸기)
- 로그인한 계정명 보기
  $ whoami
- 로그인한 계정과 관계없이 root로 변경하기
  $ sudo whoami
- 계속 루트 계정으로 사용하기
  $ sudo su -
  (exit 명령어를 치면 원상복귀됨)
- 루트권한 탈출하기
  # exit
- 시스템 모든 파일 삭제하기(주의! 주의!)
  # rm -rf /
  (악의적으로 루트권한 얻어서 이런 명령을 벌인다면, 끔찍!!)
- 프로세스 확인하기
  $ ps
- 프로세스 죽이기
  예 $ kill -9 3290
- 모든 프로세스 보기
  $ ps -aux
- 프로세스 모니터링하기
  $ top
  (종료는 q를 누른다)
- 현재 남은 메모리 크기 확인하기
  $ free -m
- 저장장치 목록과 사용량 보기
  $ df -h
- 각각의 폴더 용량 확인
  예 $ du -h /home/aaa
- zip으로 압축하기
  예 $ zip test.zip source
- zip으로 압축 풀기
  예 $ unzip test.zip
- tar.gz로 압축하기
  예 $ tar -czvf test.tar.gz *.jpg
- tar.gz로 압축 풀기
  예 $ tar -xzvf test.tar.gz
  예 $ tar -xvf test.tar
  (압축은 하지 않고 tar로 파일만 묶어놓은 파일 형식의 경우)
- 링크 파일 만들기
  예 $ ln -s src.txt dest.txt
  (src.txt 파일이 원본 파일, dest.txt 파일이 새로 생성될 링크 파일)
- 특정 파일 찾기
  예 $ find . -name '*.jpg'
  (현재와 하위 디렉토리 검색해서 모든 jpg 파일 찾기)
  예 $ find / -name 'Music' -type d
  (루트 디렉토리부터 전부 검색해서 Music이라 폴더 찾기)
  예 $ whereis vi
  (vi 실행 파일이 어디 있는지 찾기)
  예 $ locate test.txt
  (해당 파일명이 포함된 파일들 검색)
- 파일 내용 검색하기
  예 $ find . -name '*.txt' | xargs grep -n test
  (확장자가 txt인 파일 내용 중에 test 문자열을 포함하는 파일 찾기)
  예 $ grep -r test ./
  (현재, 하위 디렉토리 검색 후 test라는 문자열을 가지고 있는 파일 찾기)
  예 $ cat test.txt
  (파일 내용 살펴보기)
  예 $ tail -n 50 test.txt
  (파일 내용의 끝 부분 50줄만 보여주기, 반대는 head 명령어)
- 파이프 명령어(명령어끼리 조합하기)
  $ ls | more
  예 $ cat test.txt | more
  (파일 내용을 한 줄씩 키보드 커서를 이용해서 살펴본다. 종료는 q를 누름)
  $ ps -aux | grep init
  (프로세스 목록을 얻어와 init이라는 단어를 찾아 해당 라인만 보여줌)
- 리다이렉션
  예 $ ls > result.txt
  (ls 명령 결과를 result.txt라는 파일에 저장)
  예 $ grep init < text.txt
  (파일을 읽어서 init이라는 단어가 있는 줄만 화면에 출력)
- 환경변수 설정하기
  예 $ export JAVA_HOME=/usr/bin/jvm/java
- 환경변수 목록 보기
  $ env
- 특정 환경변수 보기
  $ echo $PATH
- PATH 환경 변수에 디렉토리 추가하기
  예 $ export PATH=/home/jundols:$PATH

지은이의 글

저자가 리눅스를 처음 접한 지도 10년이 훌쩍 넘었다. 대학교 동아리에서 리눅스로 웹 프로그래밍을 공부하기 위해서 접하게 된 게 첫 만남이었다. 그 당시 리눅스는 설치하는 것부터가 일이었다. 같이 공부하던 동기들 중에는 리눅스 때문에 포기하고 다른 분야로 넘어간 동기도 있을 정도였다. 리눅스 깔면 반은 배운 거라는 시절이었다.

그 이후로 리눅스는 참 많이도 변했다. 특히 우분투가 나오고 나서 PC용으로 리눅스를 사용하기에 충분할 정도로 편해졌다(물론 그전에도 가능했지만 뭔가 부족했다). 우분투를 처음 설치할 때 마우스 클릭 몇 번만으로 리눅스가 설치되는 걸 보고 감동과 함께 우분투로 리눅스를 배웠다면 그렇게 고생할 필요가 없었겠다라는 안타까움이 밀려왔다. 우분투로 리눅스를 배우는 여러분들이 부럽다.

우분투를 이용해 리눅스를 공부하는 것은 시작점으로 좋은 선택이다. 뭐든지 처음엔 쉽고 재미있어야 포기하지 않고 계속 갈 수 있다. 단지 우분투를 배우는 게 목표라면 이 책으로 가능하게 하려고 노력했다. 하지만 리눅스 전문가가 되고자 한다면 이 책에서 다루는 내용을 넘어 더 많은 공부를 해야 한다.

어떤 분야의 전문가가 되느냐에 따라서 같은 리눅스지만 배워야 하는 지식과 방향이 달라지게 된다. 다양한 서버의 운영과 관리 경험을 쌓아서 안정적인 서버 운영을 책임지는 서버 관리자가 될 수도 있다. 리눅스 보안 분야를 공부해서 악의적인 해커로부터 서버를 지키는 보안 전문가가 될 수도 있다. 임베디드 분야는 리눅스를 여러 부분에서 사용하고 있다. 리눅스에서 프로그램을 작성하는 리눅스 시스템 프로그래머가 될 수도 있다. 리눅스로 하드웨어 장치를 제어하는 리눅스 디바이스 드라이버 개발자나 새로운 장치에 리눅스 커널을 포팅하는 개발자가 될 수도 있다.

어느 분야 하나 쉬운 게 없지만 첫술에 배부를 수 없다. 욕심내지 말고 쉽고 재미있는 부분부터 차근차근 하나씩 배워가면서 흥미를 잃지 않는다면 어느샌가 고수가 되어 있을 것이다. 리누즈 토발즈가 리눅스 커널을 만들면서 그랬듯(Just for Fun, 2001, Linus Torvalds&David Diamond) 재미가 첫번째다.

지은이의 글

재미가 있으려면 이론적인 내용만 계속 파는 것보다 뭐라도 직접 해봐야 한다. 그런 이유로 이 책에서 가장 재미있는 부분은 '5장 나만의 서버 구축하기'라고 생각한다. 여러분도 눈으로만 읽지 말고 꼭 우분투로 서버를 구축해보길 바란다. 한번에 성공할 리 없다(할 수도 있지만). 그래도 고생하면서 원인을 찾아 해결해보자. 서버에 접속이 성공하는 순간 뭔가 뿌듯함이 느껴질 것이다. 눈으로만 보면 모를 맛이다. 리눅스는 실전과 경험이 중요하다.

이 책이 나오는 동안 많은 분들이 고생했다. 계속 지연되는 원고 일정에도 묵묵히 응원해준 임성춘 편집장님께 고마움을 전한다. 책 쓰는 동안 첫째 서윤이가 태어났는데 많이 못 도와줘서 고생한 아내 지은이와 언제나 아들을 응원해주는 부모님, 항상 배려해주고 반겨주는 장모님께도 고마움을 전한다. 이렇게 사랑하는 분들 덕분에 힘을 얻는다.

2016년 1월 7일 한밤중에
정준석

편집자이자 베타테스터의 글

오래 전 당시에도 상당히 희귀한(?) '임베디드 파일시스템'이란 주제로 이 책의 저자와 편집자로서 만난 적이 있다. 그로부터 10여 년이 흘러 이젠 리눅스 입문서로 다시 재회를 하였으니 감회가 남다르다.

사실 처음에 쓰기로 한 주제는 '리얼타임 리눅스 커널'이었다. 이 책을 보고 있을 독자에게는 조금은 멀게 느껴지는(?) 용어일 것 같다. 커널을 다루는 일이 저자의 본업이기도 했으니 안성맞춤이겠거니 했다. 상당 부분 집필하였으나 개인적인 사정으로 책을 완성하지는 못했다. 어느 날 미완의 원고를 만지작거리다 문득 드는 생각, "이러한 고수가 입문서를 써야 하지 않나?"라는 생각에 이르렀고 제안을 하게 된 것이 이 책의 시작이다.

십 년 전에도 리눅스 입문서는 있었고 지금도 나오고 있다. 그런데 왜 또 로드북에서 리눅스 입문서를? 입문서가 다양할수록 독자들에게는 이롭다고 생각한다. 여러 책 중에 골라 볼 수 있기 때문이다. 내용의 깊이, 범위, 서술 방법, 예제의 구성 등등에서 차이가 있기 때문이다. 물론, 그 나물에 그 밥이라는 평가를 받는다면 종이 밥을 먹고 사는 편집자들의 책임이 크다.

"그러면 이 책이 다른 입문서와 다른 게 뭘까?"

첫째, 많이 넣지 않았다. 실무자는 이 책을 볼 필요가 없다. 리눅스로 뭔가를 하고 싶은데 막막한 독자가 봐야 할 책이다. 이 책을 보면 고수(神)가 될 수 있다는 게 아니라 그 길을 갈 수 있도록 닦아주는 책이다. 독자 스스로 뭔가를 할 수 있겠다라는 자신감을 심어주는 게 가장 큰 목적이다.

둘째, 왜 배워야 하는가에 중점을 두었다. 물론 명령어 하나하나 어떤 때 쓰인다는 것을 모두 열거할 수 없지만, 몇 가지 사례를 들어줌으로써 단순 타이핑 실습이 아니라 생각하는 실습이 되게끔 노력하였다.

딱 이 두 가지 목적을 갖고 이 책을 기획하였다. 그리고 리눅스 책을 많이 만들어봤지만 아직도 초보자인 편집자가 직접 실습해보고 타이핑해보며 이해가 안 되면 저자에게 물어보고 수정하며 만들었다.

편집자이자 베타테스터의 글

이 책은 눈으로 읽으면 금방 읽힌다. 그만큼 쉽다. 그러나 눈으로 읽어서는 남는 게 없다. 절대 내 것이 되지 않는다. 실무에서 리눅스를 사용하는 사람이라면 sudo ~~ (블라블라) 하는 명령어들을 얼마나 많이 쳐보았겠는가? 파일을 열어 순식간에 편집하고 날쌔게 저장해서 시스템 상황을 파악해야 하는 시스템 관리자는 또 어떨까? 이런 관점으로 생각한다면 입문할 때 별거 아니라고 생각하는 것들이 새롭게 보일 것이다. 명령어에 다른 옵션도 붙여보고 책에서 언급한 자료들도 참고하며 학습한다면 이 책은 여러분에게 멋진 길라잡이가 되어줄 것이다.

여러분 중 많은 독자가 세월이 흘러 책장 한 귀퉁이에 꽂힌 이 책을 보며 "과거에 이 책으로 쉽게 입문한 적이 있었지"라는 평가를 받는다면 편집자로서 더할 나위 없는 기쁨일 것이다.

2016년 1월

담당 편집자 임성춘

이 책을 제대로 공부하는 방법

1. 명령어와 결과 표시는 아래와 같이 구분을 하였다.

```
$ pwd          ← 명령어
/home/jundols  ← 결과
```

2. 입문자를 위해 아래와 같이 본문 중간중간 Q&A를 두었다. 용어에 대한 설명일 수 있고 기타 궁금증을 유발할 만한 내용을 넣었다.

변수값을 바꾸어 실행파일을 만들 수 있다는 게 어떤 의미일까?

예를 들어 특정 프로그램은 멀티 코어를 최대한 활용하도록 설계되는데, 이 경우 해당 프로그램이 최적의 성능을 내려면 해당 컴퓨터의 프로세서 코어의 개수를 알아야 한다. 따라서 소스 컴파일 시에 코어 개수를 설정해 준 다음 컴파일을 하면 자신의 컴퓨터에 최적화된 실행 파일을 얻을 수 있다.

3. [직접해보세요] 코너는 반드시 직접 해보아야

이 코너에서는 본문에서 배운 내용들을 다시 약간 응용하여 복습하는 실습이다. 자신의 힘으로 꼭 직접 해보기 바란다.

직접 해봅시다

1. 버추얼박스에 우분투를 설치해 보자.
2. 버추얼박스의 옵션을 바꾸면서 가상머신의 성능이 어떻게 달라지는지 확인해보자.
3. Fedora나 Mint 같은 다른 리눅스 배포판을 한번 설치해보자.
4. 우분투 서버 버전을 설치해보자

4. [정리해봅시다]로 이론적인 내용을 정리

각 장에서 배운 내용을 서술 형식으로 테스트하는 과정이다. 무슨 지식을 습득하든지 연습 문제를 풀어봐야 그 지식에 더 익숙해질 수 있다.

정리 해봅시다

1. 가상머신에 우분투를 설치하는 방식의 장단점은 무엇일까?
2. 멀티 부팅으로 우분투를 설치하는 방식의 장단점은 무엇일까?
3. 버추얼 머신의 게스트 확장은 무슨 기능인가?
4. 가상 저장 장치 방식 중 동적 할당과 고정 크기는 어떤 차이가 있는가?

5. 궁금하면 여기 물어보세요

로드북 전용 Q&A 게시판이다. 질문 앞머리에 [우분투]만 붙여주면 된다. 저자에게 바로 질문 의뢰가 가고 빠른 답변을 받아볼 수 있다.

http://www.roadbook.zerois.net/qna

2장 깔아야 쓸 수 있다

3장 기본기 다지기

4장 중급으로 올라가자

서버의 기본 개념

우분투를 서버 용도로 사용하기 전에 기본적인 내용을 짚고 넘어가도록 하자. 가장 기본적으로 서버의 개념에 대해 살펴보자. 여기서 설명할 내용에 대해 이미 잘 알고 있는 독자라면 건너 뛰어도 된다.

우리는 이미 일상에서 흔하게 서버라는 단어를 많이 접하고 있다. 웹사이트에 접속할 때 접속이 느리면 "서버가 너무 느리다"라는 표현을 쓴다. 또 게임을 할 때 접속이 잘 안 되면 "서버가 불안정하다"라는 표현도 쓴다. 그 외에도 다양한 상황에서 서버라는 용어를 쓴다.

위에서 언급한 것처럼 서버는 원격지에 있는 컴퓨터이고 사용자(클라이언트)는 그 컴퓨터에 접속해서 뭔가 받아오거나 뭔가를 전송하는 작업을 한다. 사실 서버의 정확한 의미는 좀 다르지만 우선 이렇게 이해하고 넘어가자.

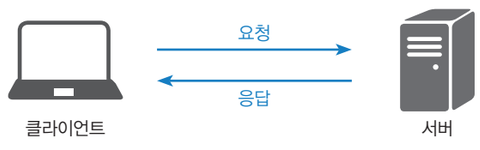

그림 5-1 서버의 개념

위 그림에서 보듯이 클라이언트가 네트워크를 통해서 서버로 서비스를 요청하면 서버는 적절한 처리를 한 후에 응답을 다시 클라이언트에게 보내게 된다. 서버는 언제 클라이언트로부터 요청을 받을지 모르기 때문에 항상 켜져 있어야 하고 네트워크에 연결되어 있어야 한다. 또한 서버에 접속하는 클라이언트가 1명이 아닐 수도 있다. 인기 있는 서비스를 제공하는 서버는 동시에 다수의 클라이언트의 요청을 처리하기도 하기 때문에 고성능 하드웨어가 필요하다.

근래에 컴퓨터를 활용하는 방식을 살펴보면 예전과 다르게 컴퓨터 단독으로 동작하는 경우보다 서버를 통해서 동작하는 경우가 더 많아지고 있다. 예를 들어서 문서 작업 같은 경우도 예전에는 MS워드나 한글처럼 컴퓨터 단독으로 동작하는 프로그램을 사용해서 문서를 작성했다면 최근에는 Google Docs처럼 서버에 문서를 저장하는 방식으로 변화하고 있다. 이렇게 서버에 데이터를 저장하게 되면 장점은 다른 컴퓨터에서도 언제든지 해당 데이터에 접근할 수 있다는 것이다. 또한 해당 데이터를 누군가와 공유하거나 복사해주는 것도 훨씬 수월하다. 또한 일반적으로 본인의 컴퓨터에 저장했을 경우보다 데이터를 잃어버리거나 여러 이유로 의도치 않게 삭제돼버릴 가능성도 더 낮다.

지금까지 서버란 무엇인지에 대해 간략하게 살펴보았다. 그런데 여기서 하나 짚고 넘어가야 할 부분이 있다. 사실 서버는 컴퓨터의 하드웨어를 지칭하는 것은 아니다. 서버는 클라이언트로부터 들어오는 다양한 요청에 대해 적절하게 서비스를 제공하는 일종의 소프트웨어를 의미한다. 혹자는 서버가 하드웨어를 의미하는 것으로 오해하기도 한다. 이는 하드웨어 제작사(HP, DELL)들이 일반 용도의 데스크탑과는 다른 서버용 컴퓨터 제품들을 따로 제작해서 팔기 때문에 생기는 오해이다. 굳이 서버용 컴퓨터들이 아니더라도 서버 소프트웨어를 설치하면 어떤 컴퓨터도 서버로 사용할 수 있다.

그림 5-2 서버용 컴퓨터

물론 서버용 컴퓨터를 사용하면 많은 장점이 있다. 일반 데스크탑 컴퓨터보다 신뢰성이 높아서 망가질 위험이 낮고, 망가지더라도 데이터를 소실하지 않도록 설계하기도 한다. 또한 하드디스크 용량이나 램 용량이 매우 크게 설계되어 있다. 프로세서도 인텔 제온Xeon 프로세서를 장착해서 성능을 극대화하기도 한다. 이렇듯 일반 컴퓨터보다 좋은 성능을 가지고 있지만 가격이 비싸다.

제온(Xeon)프로세서?

인텔이 서버용으로 만든 프로세서 모델명이다. 보통 일반 사용자들이 사용하는 컴퓨터는 프로세서로 인텔 i3, i5, i7 모델을 사용한다. 하지만 고성능 서버용 컴퓨터에서는 이런 프로세서가 아닌 서버용으로 설계된 제온 프로세서를 사용한다. 제온 프로세서도 E3, E5, E7 라인업을 가지고 있는데, E7은 최고 사양으로 i7시리즈 프로세서보다 더 많은 코어와 더 큰 캐시를 장착하고 있다. 하지만 성능에 비해 가격대가 매우 높다.

서버 소프트웨어는 목적과 용도에 따라서 그 종류가 매우 다양하다. 무료로 쓸 수 있는 공개용 서버 소프트웨어부터 고가의 상용 소프트웨어까지 수많은 소프트웨어가 있다. 따라서 자신의 환경에 따라서 그때 그때 적절한 소프트웨어를 선택해야 한다. 서버 소프트웨어의 종류와 특징에 대해서는 뒤에서 한번 살펴보도록 하자.

서버에 접속하는 동작 원리

이번에는 클라이언트가 서버에 접속하는 동작 원리에 대해 살펴보자. 원격지에 있는 서버에 접속하기 위해서는 첫 번째로 당연히 서로 물리적인 네트워크가 연결되어 있어야 한다. 컴퓨터를 네트워크에 연결하지 않으면 외부와 통신을 할 수가 없다. 두 번째로는 네트워크에 연결된 컴퓨터끼리 데이터를 주고 받기 위해서는 서로 사전에 약속한 규약이 있어야 한다. 이런 규약을 프로토콜Protocol이라고 부른다. 마지막으로 이렇게 서로 약속된 규약 안에서 클라이언트는 원격지에 있는 특정 컴퓨터에 접속해서 데이터를 요청하고 응답을 받아야 한다.

이렇게 간단하게 요약했지만 사실 클라이언트가 서버에 접속하는 원리는 상당히 복잡하다. 하지만 이 책에서는 이런 내용을 자세하게 살펴보는 건 범위를 벗어나므로 기초적인 부분만 살펴보자.

컴퓨터를 물리적으로 연결하는 LAN

세상에는 다양한 종류의 네트워크가 있고, 각각의 네트워크마다 동작 원리가 다르다. 하지만 현재 가장 대중적으로 사용되는 네트워크 방식은 이더넷ethernet이다(이더넷과 인터넷은 다르다). 이더넷은 워낙 일반적으로 사용되기 때문에 LANLocalAreaNetwork이라고 부르기도 한다. 랜선이니 랜카드니 하는 것들은 대부분 이더넷 네트워크 장비를 말한다. 이더넷 네트워크 구성의 핵심인 랜카드는 저마다 고유한 주소값을 가지고 있어서 멀리서 전송된 데이터를 정확하게 수신 받을 수 있도록 해준다. 이런 이더넷 주소값을 MAC Address라고 부른다. 이 외에도 많이 쓰이는 네트워크로는 무선으로 네트워크를 연결할 수 있는 방식인 Wireless LAN도 있다. 보통은 WIFI라고 부른다. 여기서는 WIFI는 다루지 않는다.

랜(LAN)이란?

근거리 통신망이라 부르며, 서로 가까운 거리에 붙어 있는 컴퓨터들끼리 네트워크로 묶어 놓은 것을 의미한다. 좀 더 살펴보면, LAN으로 묶여 있는 컴퓨터들은 같은 네트워크 방식을 사용한다.

예를 들어 A라는 집의 공유기에 3개의 컴퓨터가 연결되어 있다면 해당 컴퓨터 3대는 서로 LAN으로 연결되어 있다고 말할 수 있다. 그런데 옆집 B라는 집의 컴퓨터는 A집과 가까울지라도 서로 LAN으로 연결되어 있다고 말할 수 없다. A집과 B집은 같은 네트워크 방식으로 엮여있지 않기 때문이다.

LAN으로 묶여 있는 네트워크는 외부와 통신하기 위해서는 반드시 게이트웨이 또는 라우터 장치가 존재한다. 집에서는 인터넷 공유기나 케이블 모뎀이 그 역할을 해준다. 인터넷 공유기나 케이블 모뎀을 기준으로 LAN의 외부, 내부를 구분 짓게 된다.

요즘엔 대부분 이더넷을 이용해서 LAN을 구성한다. 쉽게 이야기해서 인터넷 공유기를 이용해서 여러 컴퓨터를 공유기에 물리는 것이 바로 이더넷으로 LAN을 구성한 것이다.

LAN에 연결된 랜카드들은 각자 고유한 물리 주소값을 가지고 있는데 이를 MAC Address라고 부른다. MAC Address는 LAN 내부에서만 사용되기 때문에 IP 주소로 서로를 구분 짓는 인터넷과 방식이 다르다. 이때 이 둘을 매칭해주는 프로토콜이 ARP(Address Resolution Protocol)이다.

인터넷 공유기(게이트웨이)가 내부 컴퓨터인 123.123.123.123에게 전달할 데이터를 외부에서 수신 받았다고 하자. 그러면 공유기는 ARP Request를 LAN 내부에 있는 모든 컴퓨터에게 전송한다. ARP Request는 123.123.123.123인 IP를 사용중인 컴퓨터의 MAC Address를 알려달라는 프로토콜이다. 프로토콜을 수신 받은 모든 내부 컴퓨터 중에서 123.123.123.123을 사용중인 컴퓨터가 있다면 그에 대한 응답으로 자신의 MAC Address를 보내준다. 이를 ARP Response라고 한다.

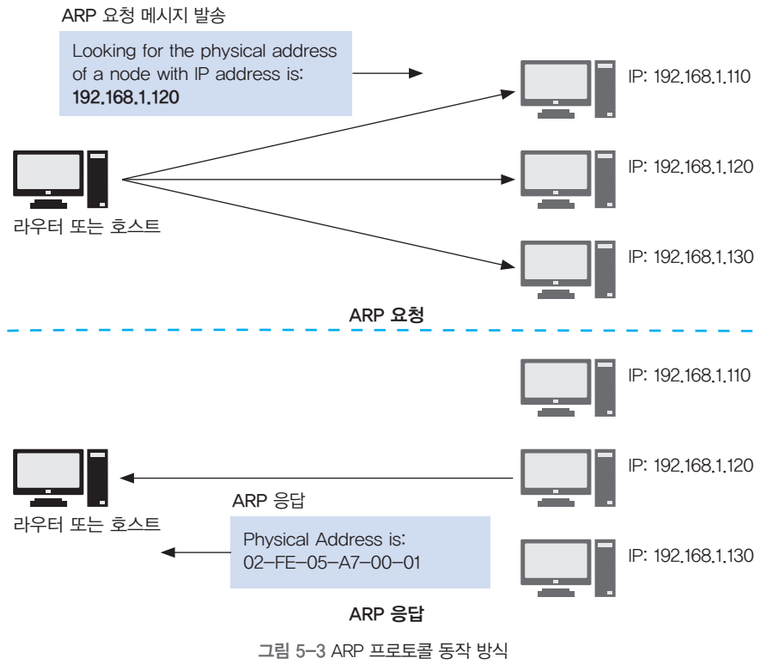

그림 5-3 ARP 프로토콜 동작 방식

컴퓨터에 장착되어 있는 랜카드에 랜선을 이용해서 인터넷 공유기 또는 케이블 모뎀과 연결을 했다면 물리적으로 네트워크가 연결된 것이다. 즉, 컴퓨터로 인터넷을 쓸 수 있는 연결을 완료한 것이다.

보통 사용자 입장에서 인터넷에 연결한다는 것은 케이블 모뎀이나 인터넷 공유기 같은 장치에 컴퓨터를 연결하는 것만 하면 된다. 하지만 실제로 그것은 단지 인터넷에 들어가는 시작점을 연결한 것에 불과하다.

컴퓨터와 케이블 모뎀까지는 이더넷 케이블로 연결되지만 케이블 모뎀 반대편은 다른 네트워크 세상이 펼쳐진다. 케이블 모뎀은 망 사업자로 연결된다. 이때 사용하는 네트워크는 이더넷 방식이 아닌 다른 네트워크 방식(예를 들면 DSL)을 사용한다.

망 사업자는 다시 인터넷 백본Backbone 망과 연결이 된다. 인터넷 백본 망은 해외에 있는 컴퓨터와 통신을 하기 위해 바다 속 해저 광케이블을 지나서 전세계로 뻗어 나가게 된다.

이렇듯 인터넷이라는 거대하고 수많은 네트워크의 조합에서 보자면 사용자가 사용하는 이더넷 네트워크는 매우 작은 부분에 불과하다.

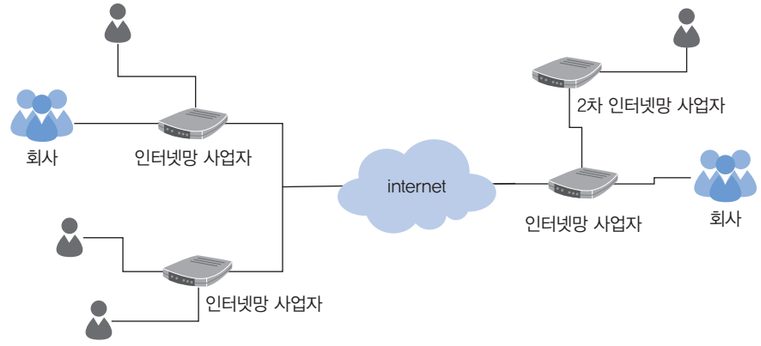

그림 5-4 인터넷에 연결된 네트워크 구조

위 그림처럼 인터넷은 사용자가 사용하는 이더넷 네트워크 외에도 수많은 종류의 다양한 네트워크가 혼재되어 있다. 이런 다양한 종류의 네트워크들이 서로 데이터를 주고 받고 통신을 원활하게 하기 위해서는 모든 네트워크가 이해하고 호환이 될 수 있는 공통의 규약이 필요하다. 인터넷을 이루는 가장 중요한 규약인 TCP/IP가 바로 그것이다.

TCP/IP 프로토콜

다양한 네트워크로 연결된 컴퓨터끼리 데이터를 주고 받으려면 서로 호환이 되는 규약이 필요하다. 데이터를 주고 받을 때 사용하는 가장 대표적인 규약은 인터넷이라는 거대한 망을 이용할 때 사용하는 TCP/IP라는 프로토콜이다.

TCP/IP 규약만 호환되도록 네트워크를 설계한다면 어떠한 새로운 네트워크도 인터넷에 연결될 수가 있다. TCP/IP는 네트워크 망에 독립적인 규약이다. 즉, TCP/IP 규약은 물리적인 전송 방식에 대해서는 규정을 정해놓지 않고 있다. TCP/IP 프로토콜은 인터넷을 통해서 데이터를 주고 받을 때 사용하는 가장 대표적인 프로토콜 2개를 의미한다. 하나는 TCP[Transmission Control Protocol]이고 또 하나는 IP[Internet Protocol]이다. 하지만 이것 외에도 인터넷을 구성하는 프로토콜은 종류가 더 많다.

프로토콜이 필요한 이유

표준을 만든다는 것은 서로가 지켜야 할 규약을 만든다는 것이다. 하드웨어 표준이면 물리적인 크기, 핀 개수, 전기적 특성 같은 내용들이 표준으로 정의될 부분들일 것이다. 소프트웨어, 그 중에서도 프로토콜 표준은 서로 주고 받는 데이터의 구조와 방식에 대한 약속이다. 예를 들어 아래 IP 프로토콜의 데이터 구조를 보여주는 그림을 보자.

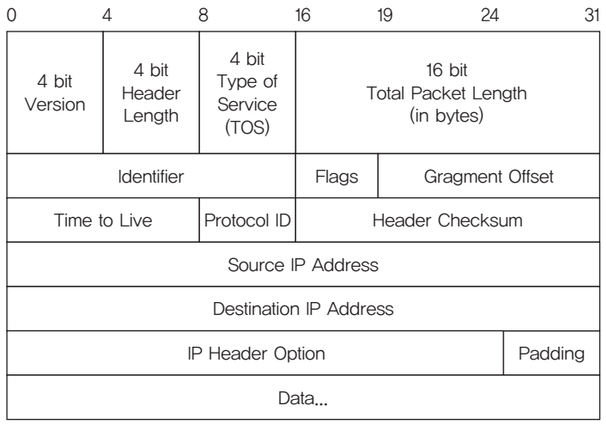

그림 5-5 IP 프로토콜 데이터 구조

그림에서 보듯이 다양한 항목이 있다. 각 항목들은 반드시 위 그림처럼 정해진 크기와 위치에 적절하게 작성되어서 전송되어야 한다. Source IP Address는 반드시 13번째 바이트(위 그림에서 한 줄이 4바이트이다)에 들어있어야 한다. 이런 규약을 지키지 않으면 해당 IP 프로토콜을 수신한 컴퓨터는 제대로 정보를 해석할 수가 없다.

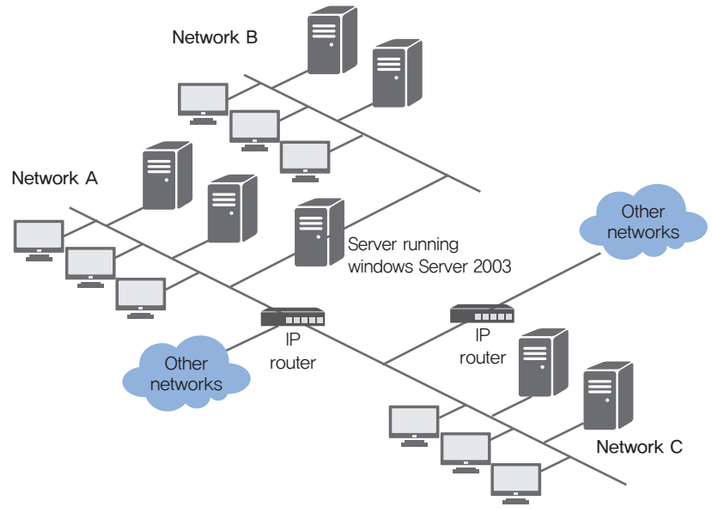

그림 5-6 IP 라우터로 연결된 인터넷[1]

우선 IP에 대해 살펴보자. IP는 자주 듣는 용어이어서 다들 익숙할 것이다. 인터넷에 연결된 컴퓨터는 반드시 고유한 IP 주소를 가지고 있는다. xxx.xxx.xxx.xxx처럼 세자리 숫자가 점(.)을 기준으로 분리되어 4부분으로 표현된다.

예를 들면 211.223.049.008(앞쪽 0은 생략 가능) 이런 식이다. 상대방의 IP 주소만 알면 상대방이 전세계 어디에 있든 접속할 수가 있다(정확하게 따지자면 이 말은 틀리다. 나중에 살펴볼 것이다). 이게 가능한 이유는 인터넷 망 중간 중간에 라우터라고 불리는 중계 장치가 붙어 있는데, 이 중계기들이 목적지 IP 주소를 살펴봐서 어디로 데이터를 보내야 할지 판단해 주기 때문이다. 이를 IP 라우팅이라고 한다. [그림 5-6]에서 보듯이 사용자가 전송한 데이터는 항상 중요한 지점에서 라우터를 만나게 되고 라우터는 해당 데이터를 적절한 네트워크로 전달해준다. 아래 [그림 5-7]에서 이런 내용에 대해 보여주고 있다.

1 https://msdn.microsoft.com/ko-kr/library/cc785246(v=ws.10).aspx

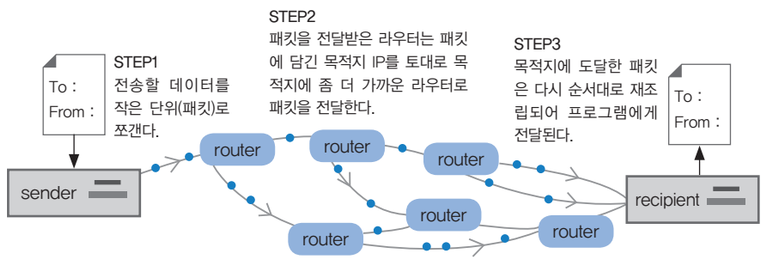

그림 5-7 네트워크를 통해 데이터가 전송되는 원리[2]

TCP는 사용자가 전송한 데이터가 인터넷을 통해서 다른 컴퓨터로 전송되는 도중에 깨지거나 변조가 되지 않도록 해주는 규약을 담고 있다. 사용자가 데이터 전송을 요청하면 컴퓨터는 해당 데이터를 적절하게 작은 크기로 쪼개서 전송하는데, 이를 패킷Packet이라고 부른다. TCP는 이런 패킷이 전송되는 도중에 전송 순서가 바뀌거나 내용이 바뀌지 않도록 모든 패킷의 앞쪽에 각각 패킷의 순서와 패킷의 에러를 체크할 수 있는 확인값을 같이 첨부해서 보내게 된다. 그러면 패킷을 수신 받는 쪽 컴퓨터는 수신된 다량의 패킷을 순서대로 다시 정렬한 다음에 에러를 체크하고 에러가 없으면 해당 데이터를 기다리고 있는 서버 소프트웨어에게 전달해 준다.

지금까지 살펴본 것처럼 TCP/IP는 단지 전송 규약이다. TCP/IP의 목적은 사용자가 보내고자 하는 데이터를 정확한 목적지까지 오류 없이 보내는 것을 목표로 하는 규약이다. 최종적으로 정확하게 목적지에 도착한 데이터는 해당 데이터를 기다리고 있는 적절한 프로그램에게 전달이 되야 하는 것이다. 예시를 하나 살펴보자.

2 http://www.billslater.com/internet

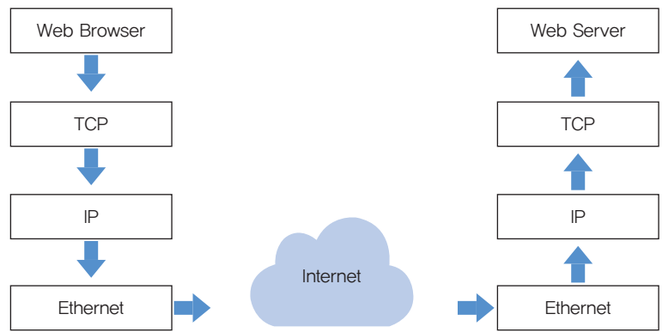

그림 5-8 인터넷 전송 계층 구조

[그림 5-8]에서 보듯이 최초에 데이터를 전송하고자 하는 소프트웨어는 웹 브라우저이다. TCP/IP는 웹 브라우저가 보내려는 데이터를 목적지에 정확하게 보내는 역할을 해준다. 수신측 컴퓨터는 웹 브라우저가 보낸 데이터를 웹 서버에게 전달해 준다. 그러면 웹 서버는 적절한 처리 후에 응답 데이터를 다시 송신측 어플리케이션인 웹 브라우저에게 전송한다. 이렇듯 인터넷은 전송 구조가 계층적으로 설계되어 있다. 물리 계층 위에 IP 계층이 있고 IP 계층 위에 TCP 계층이 있다. TCP 계층 위에는 수많은 상위 프로토콜(예를 들면 HTTP)들이 존재한다.

사실 실제 인터넷은 지금까지 설명한 것보다 더 많은 계층으로 나뉘어져 있다. 또 더 복잡하고 더 많은 프로토콜이 있지만 이런 이론적인 부분을 일일이 설명하는 것은 이 책의 주제를 벗어나므로 생략한다. 대신 어떤 프로토콜들이 있는지 아래 그림을 보면서 한번 살펴보는 것으로 하자.

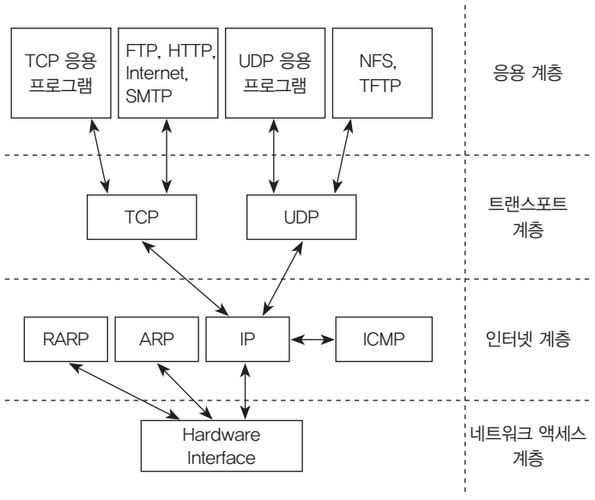

그림 5-9 인터넷 계층도[3]

[그림 5-9]는 인터넷의 계층 구조와 각각 계층에 해당하는 프로토콜 목록을 보여주고 있다. [그림 5-9]처럼 TCP 프로토콜을 기반으로 동작하는 대표적인 상위 프로토콜은 FTP, HTTP, Telnet 같은 것들이 있다.

트랜스포트 계층에는 TCP와 같은 역할을 하는 비슷한 프로토콜로 UDP라는 것이 있다. UDP도 패킷을 전송하는 목적을 가지고 있지만 TCP처럼 목적지에 데이터를 신뢰성 있게 전송하려는 특성이 없고 최대한 전달이 되도록 노력은 하되 중간에 망가지거나 사라질 수도 있는 특성을 가지고 있다. 대신에 동일 조건에서 UDP의 전송 속도가 TCP보다 빠르다. UDP에는 수신측 컴퓨터가 패킷을 잘 받았는지 확인하는 과정이 없기 때문이다. 따라서 TCP와 UDP 중 어떤 것이 좋은가는 상황에 따라서 다르기 때문에 판단하기 어렵다. UDP는 음성 전달, 영상 전달처럼 중간에 데이터가 소실돼도 서비스 제공에 큰 문제가 없는 곳에서 주로 사용된다. 대부분의 상위 프로토콜들은 TCP 기반으로 동작하는 경우가 많다. 그래서 TCP/IP라고 부르는 것이다.

3 http://jkkang.net/unix/netprg/chap1/net1_2.html

IP와 TCP란?

IP와 IP Address를 혼동하면 안 된다. IP(Internet Protocol)는 인터넷에서 데이터를 주고 받을 때 목적지와 발신지를 지정하는 규약이다. IP Address는 IP 통신을 하기 위한 숫자로된 주소 값을 말한다. IP나 TCP나 전부 소프트웨어적으로 구현이 되는 프로토콜이다. IP나 TCP 프로토콜은 최상위에 있는 어플리케이션 프로토콜이 아닌 기본에 가까운 프로토콜이고 표준이기 때문에 어느 운영체제든지 커널에 구현되어 있다. 윈도우나 리눅스가 TCP를 구현한 코드가 다르게 작성되었다 할지라도 동작 방식은 거의 동일하다.

포트(Port)에 대하여

여기까지 왔다면 클라이언트로부터 전송된 데이터는 머나먼 길을 지나서 드디어 목적지 컴퓨터에 도착했을 것이다. 목적지에 도착했으니 여행이 끝난 것 같지만 아직 거쳐야 할 단계가 하나 더 남아있다. 바로 포트 번호 확인이다.

컴퓨터에 송수신되는 수많은 데이터들이 항상 하나의 프로그램에서만 전송되고 수신되는 건 아니다. 사용자는 웹 서핑을 하면서 동시에 온라인 게임을 할 수도 있고, 인터넷을 통해서 음악을 들을 수도 있다. 이렇게 다양한 프로그램이 동시에 데이터를 주고 받는다고 해도 전송에 이용되는 네트워크 라인은 하나이다. 여러 프로그램이 하나의 네트워크 라인을 공유하게 되는 것이다. 이럴 경우 어떤 데이터가 어떤 어플리케이션에서 보낸 건지, 방금 수신한 데이터를 어떤 어플리케이션에게 건네줘야 하는지를 결정하는 게 바로 포트라는 것이다.

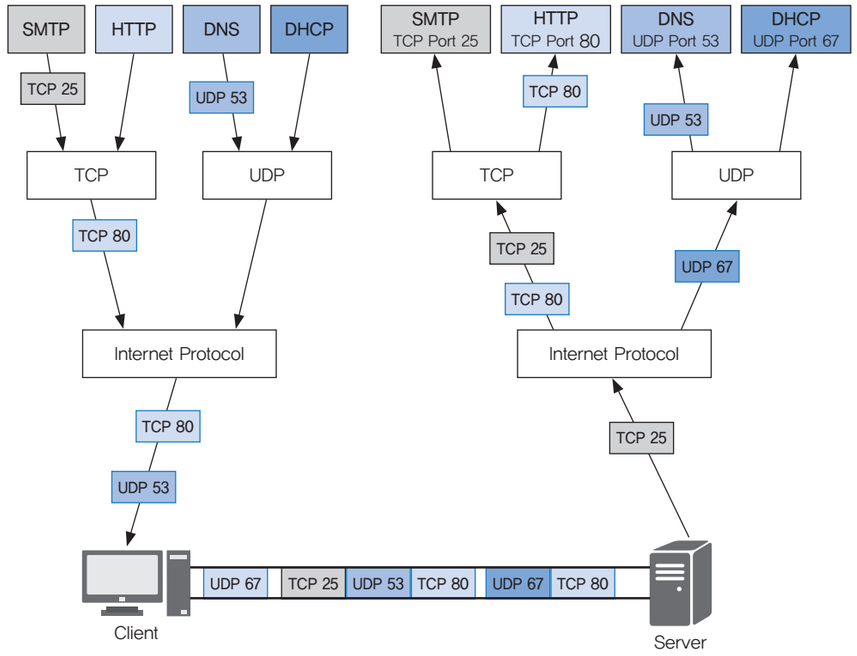

그림 5-10 포트의 개념[4]

그림 예제를 한번 살펴보자. 왼쪽에 있는 클라이언트 컴퓨터에는 4개의 프로그램이 동작 중에 있다. SMTP 프로그램은 서버의 TCP 25번 포트로 데이터를 전송하고 있다. 서버 TCP 80번 포트를 향해서 전달되고 있는 데이터도 있는데 이는 HTTP 프로그램이 전송한 것이다. DNS 프로그램은 서버 UDP 53번 포트로 전달하는 데이터를 전송 중에 있다. 클라이언트 컴퓨터에서 출발한 데이터는 목적지 서버의 IP와 포트 번호만 있으면 서버에 도착해서 적절한 서버 프로그램에게 전달되게 된다.

서버 프로그램에 전달된 데이터가 잘 처리되면 처리 결과를 다시 클라이언트에게 전달해야 한다. 이때에는 클라이언트가 서버로 데이터를 보낼 때와는 반대로 클라이언트의 IP 주소와 포트 번호를 목적지로 정확하게 설정하고 보내줘야 한다. 따라서 클라이언트는 서버로 데이터를 보낼 때 자신의 IP 주소와 해당 데이터를 수신할 자신의 포트 정보를 같이 실어서 보내준다.

4 http://www.tcpipguide.com/free/t_TCPIPPortsTransportLayerTCPUDPAddressing-2.htm

일반적으로 클라이언트측의 포트 번호는 따로 사용자가 설정해주지 않는다. 서버로 데이터를 전송할 때 자신의 수신 포트를 같이 알려주기 때문에 운영체제가 알아서 수신용으로 임의의 포트 번호를 지정한다.

하지만 서버측 포트 번호는 중요하다. 클라이언트는 서버에 접속할 때 서버 IP 외에도 서버 포트 번호를 같이 알아야 한다. 따라서 서버 프로그램이 동작할 때마다 포트 번호가 임의로 할당되게 되면 클라이언트가 제대로 서버 프로그램에 접속할 수가 없게 된다. 따라서 서버 프로그램은 프로그램 구동 시에 반드시 어떤 포트 번호를 사용할지 지정해 주게 되어 있다.

그런데 평소에 우리는 인터넷을 할 때 포트 번호를 신경 쓰지 않고 서버에 접속하는 경우가 대부분이다. 예를 들어 네이버에 접속한다고 하자. 그러면 웹 브라우저에 http://www.naver.com이라고만 적으면 네이버 웹 서버에 접속해서 정보를 얻어온다. 원래 정확하게 접속하자면 http://www.naver.com:80이라고 적어야 한다. 여기서 80이라는 숫자는 네이버 서버에 80번 포트로 접속하라는 의미이다.

왜 서버에 접속할 때 반드시 알아야 하는 포트 번호를 생략해도 문제가 없었을까? 이유는 웹 서버는 TCP 80번 포트를 쓴다는 약속이 되어 있기 때문이다. 특별한 경우가 아니면 대부분의 웹 서버는 전부 TCP 80번 포트를 사용한다. 따라서 웹 브라우저는 사용자가 포트 번호를 생략하면 자동으로 서버에 접속할 때 80번으로 접속하게 되는 것이다. 이렇게 목적이 정해져 있는 포트 번호를 Well-Known 포트 번호라고 부른다. 아래 표는 대표적인 Well-Known 포트 리스트를 보여 준다.

아래 표에 적힌 번호 외에도 상당히 많이 있다.