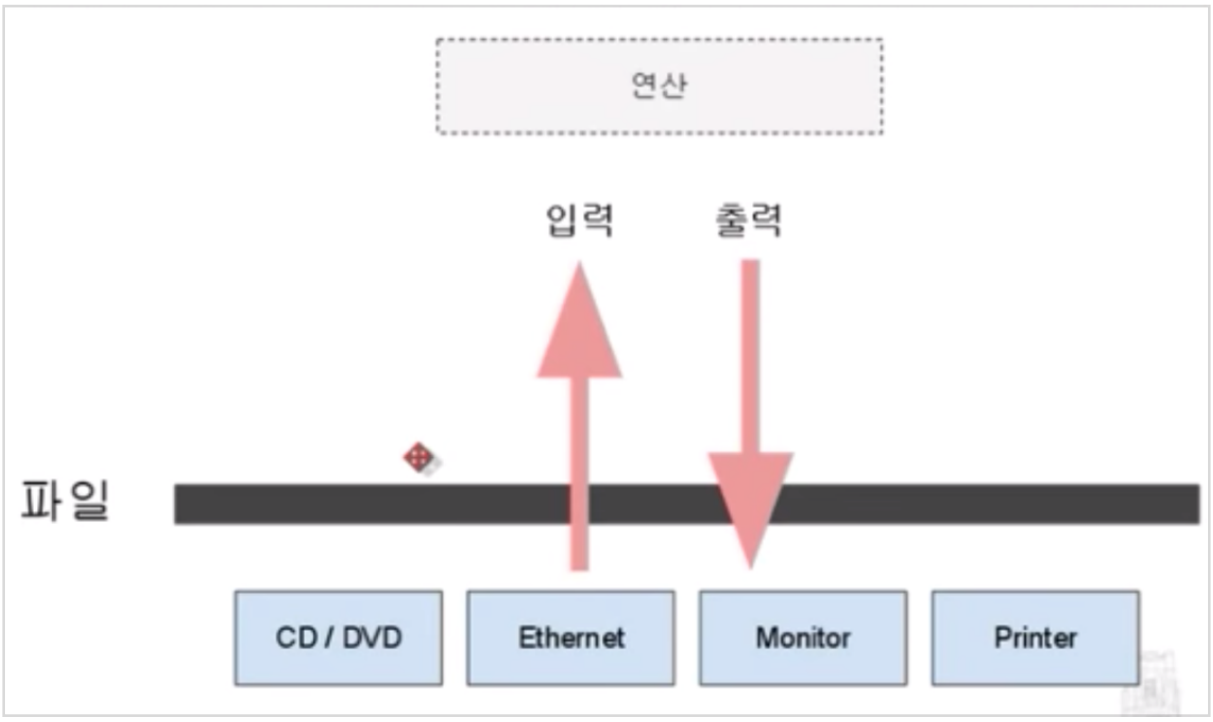
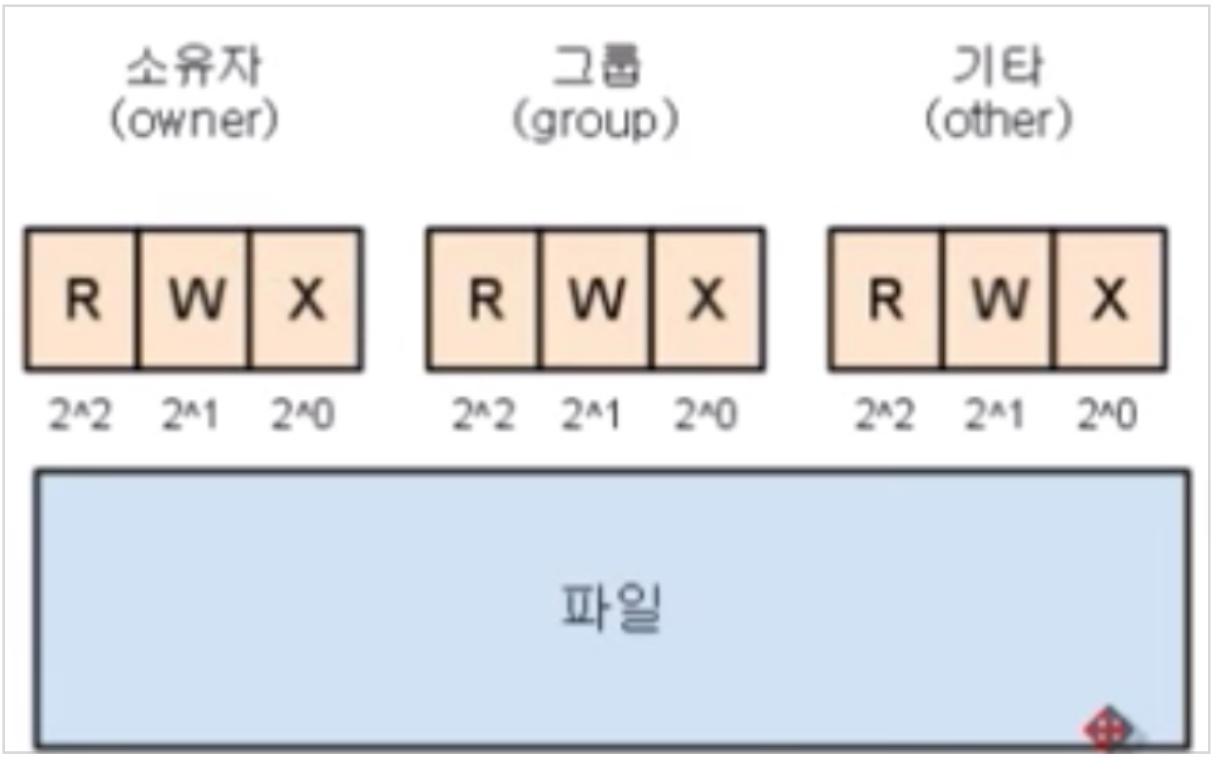
[TCP / IP 강의] 5. 파일과 소켓

왜 모든 것을 파일로 다루나



- 유닉스 개발자는 컴퓨터가 다루는 모든 장치들을 파일로 읽고 쓰는 일로 추상화 할 수 있음
  - 추상화는 내용의 복잡함을 숨기고 몇 개의 인터페이스만으로 기기를 다룰 수 있게 함

파일 권한



- 유닉스는 다중 사용자 운영 체제

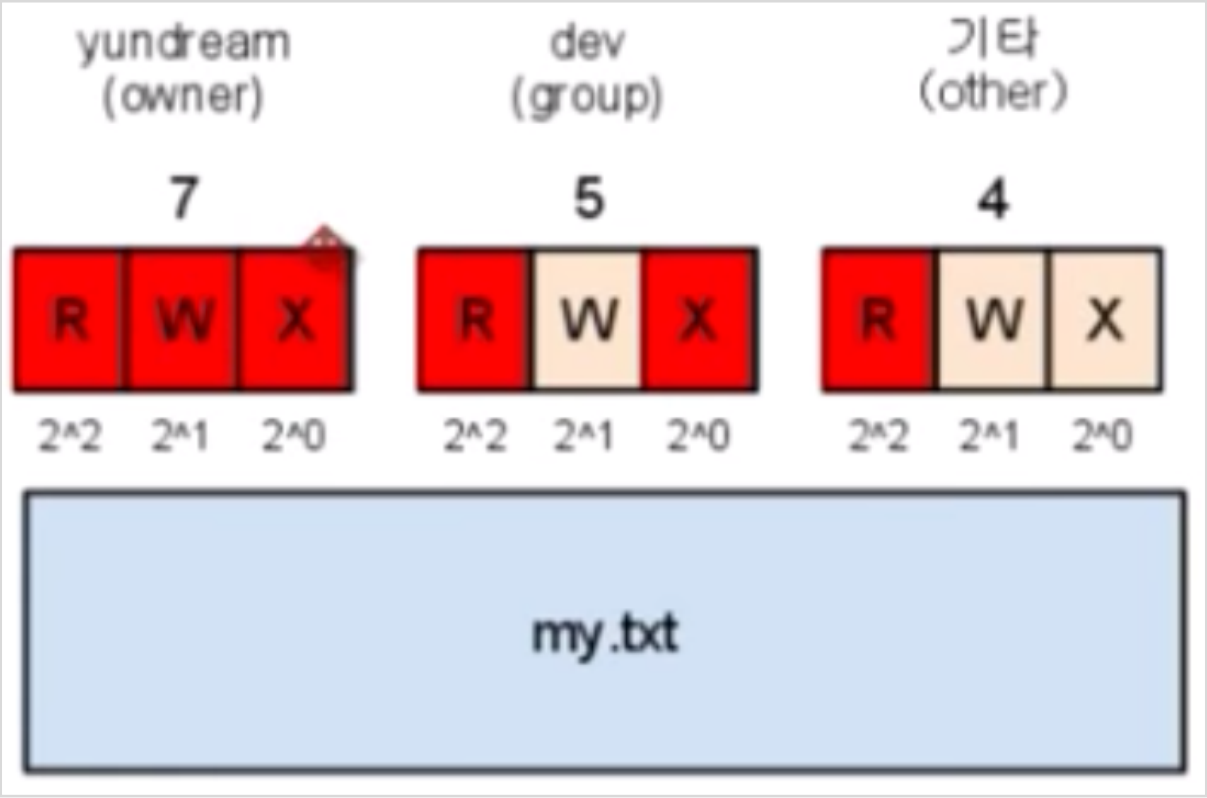
- 파일당 권한 = 자원의 권한

유닉스 파일 유저

1. 소유자 (owner)
2. 그룹 (group)
3. 기타 사용자 (other)

- 각 사용자 별로 `실행 / 읽기 / 쓰기` 를 권한으로 가짐
  - ls -alh ➡ 터미널 명령어

파일 권한 예시



- my.txt의 파일 권한
  - yundream가 소유자로서 읽기 쓰기 실행 모두 권한을 가지고 있음
  - dev는 읽기 / 실행 가능
  - 기타는 읽기만 가능

파일의 종류

- 일반파일
- 디렉토리 : 파일을 계층적으로 관리하기 위함
- 파이프 : 프로세스간의 데이터 교환
- 심볼릭 링크 : 다른 파일을 가리키기 위한 메타 정보를 가진 파일
- 블록 장치

- 문자 장치
- 소켓

표준 입력, 표준 출력, 표준 에러

1. 표준 입력 : 키보드를 이용한 입력 (0)
2. 표준 출력 : 모니터를 이용한 출력 (1)
3. 표준 에러 : 모니터를 이용한 출력 (2)
- 프로그램 실행 ➡ 프로세스 생성 시 0, 1, 2라는 파일로 만들어짐 ⬅ ➡ 모니터 / 키보드

재지향 (redirection)

```
$ cat my.txt
$ cat my.txt > ok.txt  //my.txt가 ok.txt로 재지향 (복사) 됨

$ ./stderr 2> errmsg.txt // 표준의 파일 지정번호 '2'를 지정해서 에러 메세지 저장

#include <stdio.h>
int main() {
  fprintf(stderr, "This message Error\n"); //stderr : 표준 에러
  fprintf(stdout, "This message Success\n"); //stdout : 표준 출력
}
```

파일 열기

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

// pathname 파일 이름
// flags 파일을 열기 위한 방식 (ex 읽기 전용, 쓰기 전용 등)
// mde_t mode 파일 권한 설정
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mde_t mode);

// my.txt 쓰기 전용으로 파일 생성, 0755라는 권한을 부여
fd = open("my.txt", O_WRONLY|O_CREAT, 0755);
```

Open 함수의 Flag

- `O_RDONLY` - 읽기 전용
- `O_WRONLY` - 쓰기 전용
- `O_RDWR` - 읽기와 쓰기가 모두 가능
- `O_CREAT` - 해당 파일이 없으면 생성합니다.
- `O_EXCL` - 파일이 있는지 검사
- `O_APPEND` - 추가 모드로 열기
  - 파일을 추가하여 쓰기가 되도록 open 후에 쓰기 포인터가 파일의 끝에 위치하게 됩니다.
- `O_NONBLOCK` - 비 봉쇄 모드로 열기
  - 읽을 내용이 없을 때에는 읽을 내용이 있을 때까지 기다리지 않고 바로 복귀합니다.

```
// 쓰기 전용 + 새로 만들고자 하는 파일이 이미 있는지 확인하고 싶을 때
open("my.txt", O_WRONLY| O_CREAT| O_EXCL)
```

  - 만약, my.txt가 있으면 `O_EXCL` 에서 에러 반환

파일의 매개변수와 파일 이용 순서

- 파일을 생성 할 때 권한 지정 가능
  - 소유자 / 그룹 / Other
  - 읽기 / 쓰기 / 실행 권한 지정
- 파일 열기 (open) 시에는 변경 불가

1. 파일 Open
2. Open 함수가 파일 지정번호를 통해 파일을 지정
   - 파일 지정 번호는 int 타입
3. 2의 open 함수로 만들어진 파일 지정 번호를 이용해서 읽고 쓰기

파일 이용의 주의 사항

- 안쓰는 파일은 닫아줘야 한다
  - close(int fd);
- 프로세스는 열 수 있는 파일 한계가 있다

파일 읽기 / 쓰기

- 파일의 읽기와 쓰기는 커널 버퍼를 경유함 ➜ 이후 유저 버퍼로 복사
- 만약 데이터가 커널 버퍼에 쌓이지 않는다면 read()는 계속 기둘
  - 호출 시, 데이터를 기다리면서 봉쇄 = block 입출력 ⬅ ➜ 반대는 nonblock 입출력

소켓

- 소켓도 파일 형식으로 존재
- read / write와 같은 파일 함수를 그대로 사용한다
- fcntl 함수로 제어

#Development/study