2장 파일 시스템

- 파일 시스템의 구조
- 파일
- 디렉터리와 경로명
- 새로운 파일 생성
- 소유권과 허가
- 파일 유형과 일반화

한빛미디어(주)

리눅스의 파일 시스템

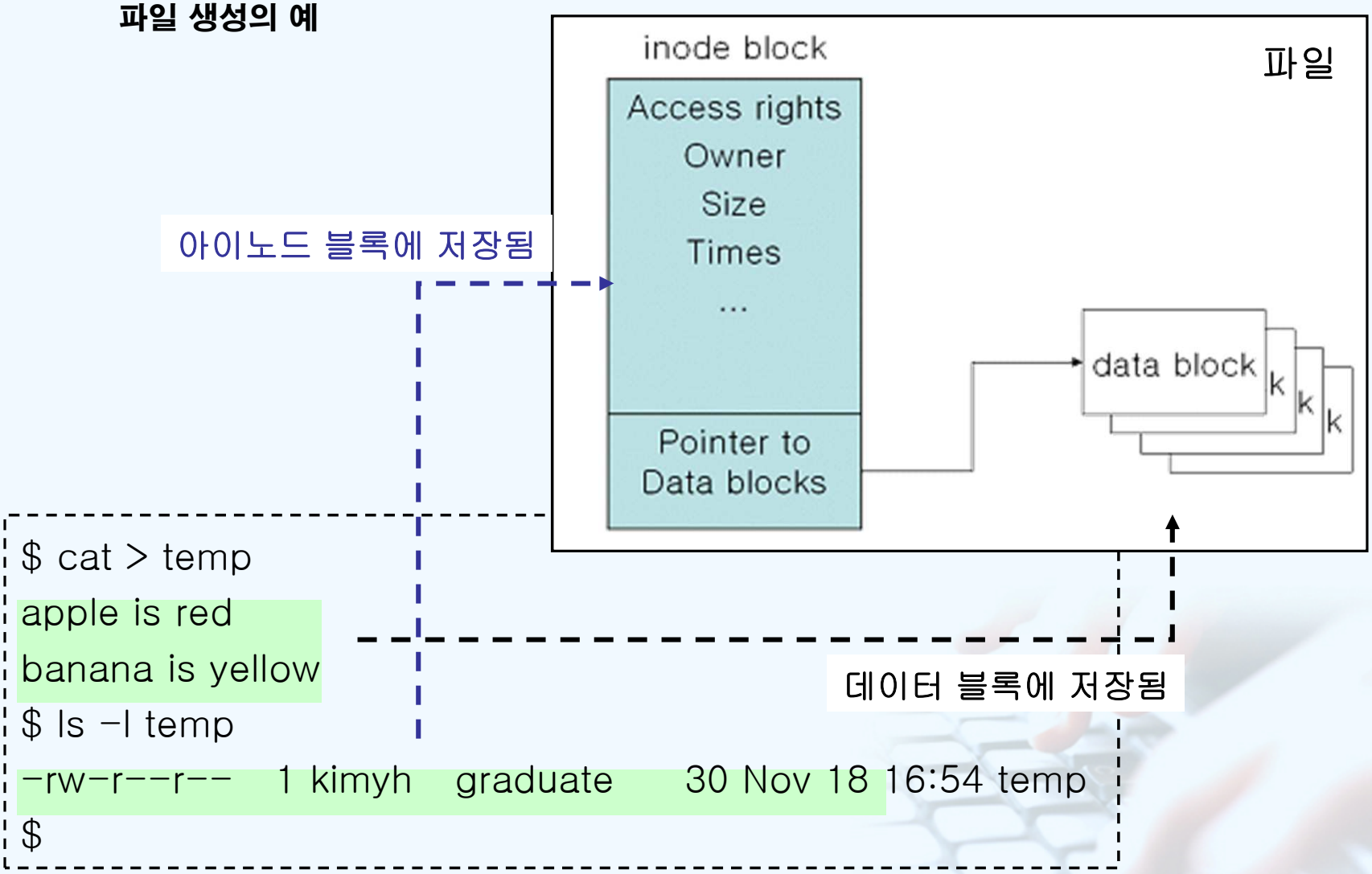
- 크게 네 가지 부분으로 구분할 수 있다.

부트 블록 (Boot Block)	슈퍼 블록 (Super Block)	아이노드 블록 (Inode Blocks)	데이터 블록 (Data Blocks)

- 부트 블록 (boot block)
  - 운영체제를 부팅시키기 위한 코드가 저장되어 있다.
- 슈퍼 블록 (super block)
  - 파일 시스템과 관련된 정보를 저장하고 있다.
- 아이노드 블록 (inode blocks)
  - 파일에 대한 정보를 저장하고 있다.
  - 모든 파일은 반드시 아이노드 블록을 하나 가지고 있다.
- 데이터 블록 (data blocks)
  - 파일이 보관해야 하는 데이터를 저장하고 있다.
  - 보관하는 데이터의 크기에 따라 여러 개일 수 있다.

아이노드 블록과 데이터 블록(들)

파일 생성의 예



파일 관련 정보

$ ls –l 명령을 실행했을 때 보여지는 파일의 정보

-	rw-r--r--	1	kimyh	gradua te	30	Nov 18 16:54	temp
파일 유형	접근권한	하드링크 수	소유주 이름	그룹 이름	파일 크기	생성 날짜	파일명

← 아이노드 블록에 저장 →

디렉터리 파일의 데이터 블록에 저장

```
$ cat > temp
apple is red
banana is yellow
$ ls -l temp
-rw-r--r--   1 kimyh   graduate     30 Nov 18 16:54 temp
$
```

파일의 의미

- 시스템 차원에서 데이터를 저장하기 위한 가장 기본적인 단위
- 리눅스에서 파일은 단순히 바이너리(2진) 데이터의 흐름을 저장하고 있다.
  - 저장된 데이터는 논리적인 구조가 정해져 있지 않다.
  - 논리적인 구조는 프로그램에 의해 결정된다.

파일의 예

```
$ cat > data
ACD
^D
$ cat data
ACD
$
```

-data 파일은 총 4바이트의 데이터를 저장하고 있음

-텍스트 파일이라고 가정
문자 A, C, D와 개행문자를 저장하고 있음

-2진 파일이라고 가정
숫자형 값인 65, 67, 68, 10을 저장하고 있음

cat으로 바이너리 파일을 표준 출력하는 예

- cat은 지정한 파일의 내용을 문자로 인식하여 터미널 화면으로 출력한다.

  아래의 예는 실행 파일인 바이너리 파일을 cat으로 표준 출력하는 예이다.

- 바이너리 파일은 문자 코드에 해당하지 않는 2진 값도 문자로 가정하므로

  아래와 같이 의미 없는 글자들이 찍히게 된다.

```
$ cat three
d.1__do_global_dtors_aux__EH_FRAME_BEGIN__fini_dummyobject.2frame_dummy
init_dummyforce_to_data__CTOR_LIST____do_global_ctors_aux__CTOR_END____
DTOR_END____FRAME_END__one.ctwo.c_DYNAMIC__register_frame_info@@GLIBC_2
.0_fp_hw_init__deregister_frame_info@@GLIBC_2.0_start__bss_startmain__l
ibc_start_main@@GLIBC_2.0data_startprintf@@GLIBC_2.0_finiprintmsg_edata
_GLOBAL_OFFSET_TABLE__end_IO_stdin_used__data_start__gmon_start__  $
$
```

디렉터리 (directory)

- 파일의 목록을 저장하기 위한 특수한 형태의 파일이다.
- 디렉터리 파일이라고 부르기도 함
- 디렉터리 파일의 데이터 블록에 파일명이 목록으로 저장되어 있다.

디렉터리 항 (directory entry)

- 디렉터리 파일의 목록을 항(entry)이라고 한다.
- 모든 디렉터리는 항상 두개의 항을 가지고 있다.
  - 자기 자신을 나타내는 항 (.)
  - 부모 디렉터리를 나타내는 항 (..)

```
$ ls -la
drwxr-xr-x    2 kimyh    graduate     4096 Nov 18 17:39 .
drwxr-xr-x    3 kimyh    graduate     4096 Nov 18 17:39 ..
$
```

아이노드 블록 번호

모든 디렉터리 항은 가리키는 파일의 아이노드 블록 번호를 가지고 있다.

- $ ls –i 로 확인할 수 있음

```
$ ls -lai
2845303 drwxr-xr-x    2 kimyh    graduate      4096 Nov 18 17:44 .
3139591 drwxr-xr-x    3 kimyh    graduate      4096 Nov 18 17:39 ..
2845304 -rw-r--r--    1 kimyh    graduate        13 Nov 18 17:44 file
$
```

아이노드 블록 번호

1) 현재 디렉터리 파일의 아이노드 블록의 번호는 28453030이다.
2) 부모 디렉터리 파일의 아이노드 블록의 번호는 31395910이다.
3) 현재 디렉터리에 등록되어 있는 file 파일의 아이노드 블록의 번호는 28453040이다.

ls 명령이 수행되는 과정

ls 명령은 지정한 디렉터리의 디렉터리 항을 출력한다.

그 과정을 다음과 같다.

1. 현재 디렉터리의 데이터 블록에서 항을 하나 읽는다.
2. 읽어 들인 디렉터리 항으로부터 아이노드 번호와 파일 이름을 구한다.
3. 아이노드 번호로 아이노드 블록을 지정하여 필요한 정보를 가져온다.
4. 정보를 출력한다.
5. 현재 디렉터리의 데이터 블록에서 다음 항을 읽고 2번에서 4번의 과정을 반복한다.

※위의 모든 과정이 ls 명령에 의해서만 이루어지는 것이 아니다. 리눅스 시스템의 커널의 도움으로 파일의 정보를 가져오게 된다.

디렉터리 파일의 실제 내용

od 명령

- 지정한 파일의 데이터 블록의 내용을 바이트 단위로 표준 출력한다.
- 기본적으로 바이트의 값을 8진수로 출력한다.

```
$ ls -lai
119271 drwxr-xr-x   2 kimyh    graduate      512 Nov 18 17:55 ./
 15552 drwxr-xr-x   5 kimyh    graduate      512 Nov 18 17:54 ../
119272 -rw-r--r--   1 kimyh    graduate       13 Nov 18 17:55 file
119273 -rw-r--r--   1 kimyh    graduate       14 Nov 18 17:55 text
$ od -c .
0000000  ₩0 001 321 347  ₩0  ₩f  ₩0 001   .  ₩0  ₩0  ₩0  ₩0  ₩0   < 300
0000020  ₩0  ₩f  ₩0 002   .   .  ₩0  ₩0  ₩0 001 321 350  ₩0 020  ₩0 004
0000040   f   i   l   e  ₩0  ₩0  ₩0  ₩0  ₩0 001 321 351 001 330  ₩0 004
0000060   t   e   x   t  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0
0000100  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0
*
0001000
$
```

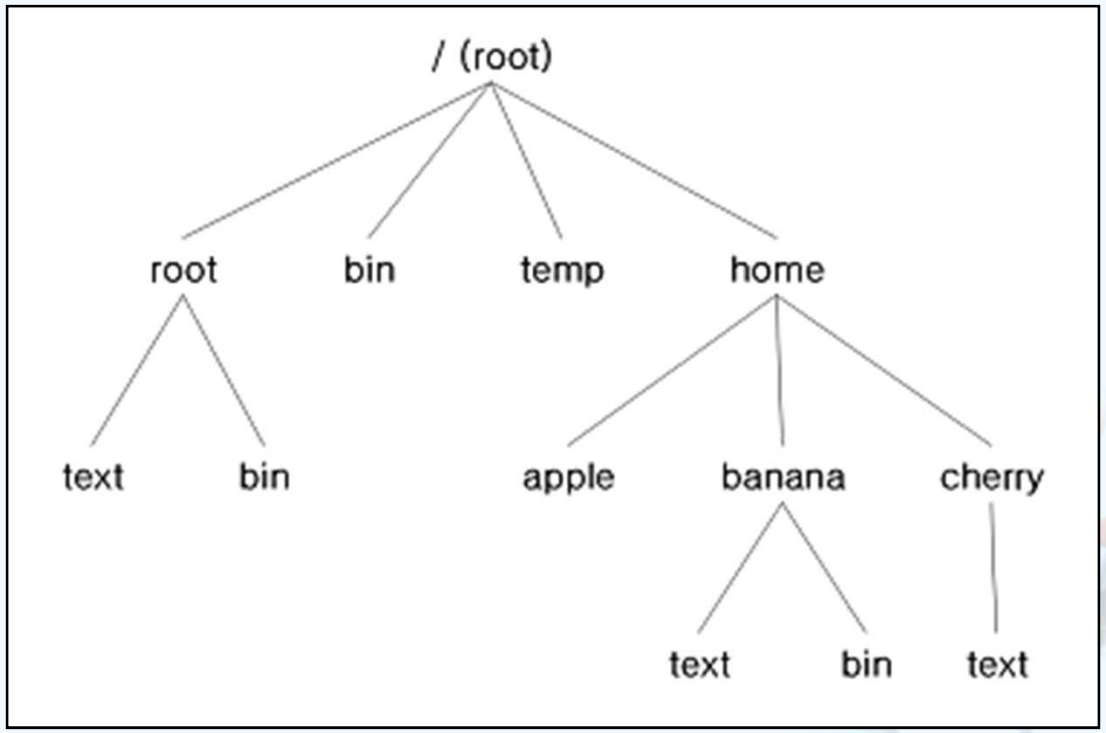
디렉터리 파일의 논리적인 구조

119271	.	₩0			
15552	.	.	₩0		
119272	f	i	l	e	₩0
119273	t	e	x	t	₩0

```
$ ls -lai
119271 drwxr-xr-x   2 kimyh    graduate       512 Nov 18 17:55 ./
 15552 drwxr-xr-x   5 kimyh    graduate       512 Nov 18 17:54 ../
119272 -rw-r--r--   1 kimyh    graduate        13 Nov 18 17:55 file
119273 -rw-r--r--   1 kimyh    graduate        14 Nov 18 17:55 text
$ od -c .
0000000  ₩0 001 321 347  ₩0  ₩f  ₩0 001   .  ₩0  ₩0  ₩0  ₩0  ₩0   < 300
0000020  ₩0  ₩f  ₩0 002   .   .  ₩0  ₩0  ₩0 001 321 350  ₩0 020  ₩0 004
0000040   f   i   l   e  ₩0  ₩0  ₩0  ₩0  ₩0 001 321 351 001 330  ₩0 004
0000060   t   e   x   t  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0
0000100  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0
*
0001000
$
```

계층 구조

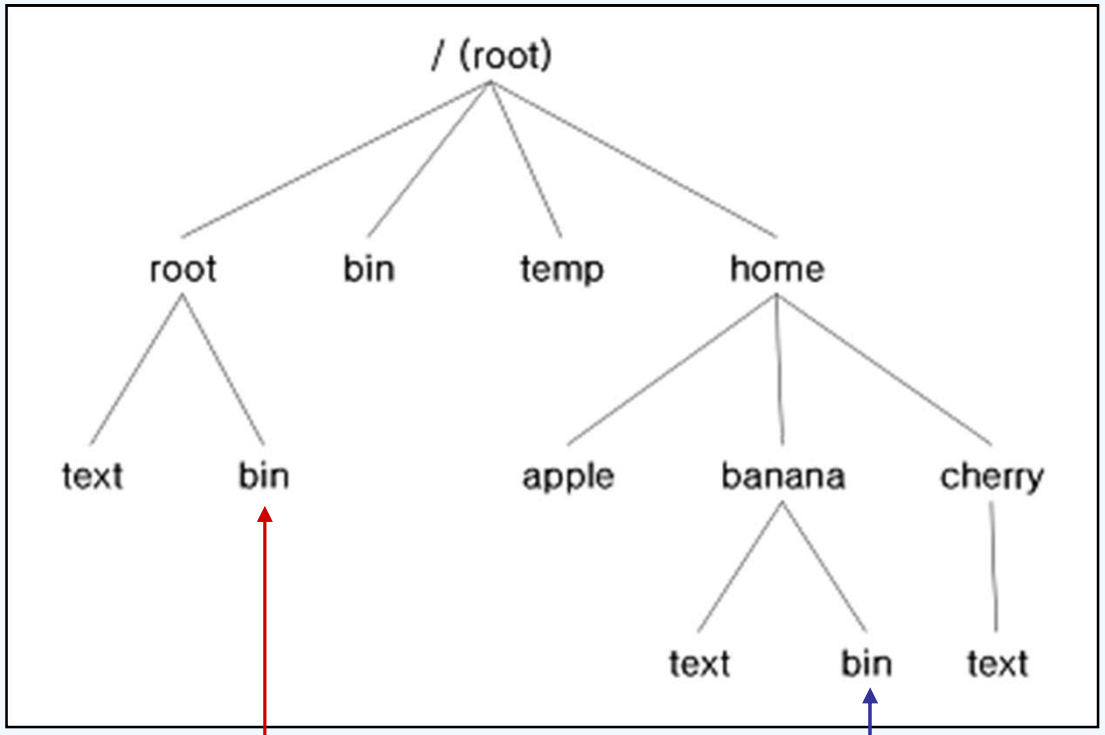
- 리눅스의 파일 시스템에는 많은 수의 디렉터리와 파일이 존재한다.
- 하나의 디렉터리 안에는 또 다른 디렉터리나 파일이 존재한다.
- 모든 디렉터리와 파일은 유일하게 존재하는 루트 디렉터리를 시작으로 트리(tree) 모양의 계층적인 구조를 이루고 있다.



절대 경로와 상대 경로

- 경로명 (pathname)
  - 파일 시스템 내에서 파일의 위치를 의미한다.
- 절대 경로
  - 루트 디렉터리를 기준으로 파일의 위치를 표현
  - 루트 디렉터리는 유일하면서 불변이기 때문에 절대 경로는 항상 같은 파일을 가리킨다.
- 상대 경로
  - 현재 디렉터리를 기준으로 파일의 위치를 표현
  - 현재 디렉터리는 바뀔 수 있기 때문에 동일한 상대 경로라도 현재 디렉터리에 다르면 가리키는 파일 역시 다르게 된다.

절대 경로와 상대 경로의 예



절대 경로

/home/apple

현재 디렉터리에 상관없이 항상 동일한 대상을 가리킴

상대 경로

./bin

현재 디렉터리에 따라 가리키는 대상이 달라짐

현재 디렉터리가 banana

현재 디렉터리가 root

※상대 경로에서 "."는 현재 디렉터리, ".."는 부모 디렉터리를 의미한다.
banana 디렉터리의 ".."는 home 디렉터리이다.

새로운 파일을 생성하는 과정

파일의 구조와 디렉터리 파일의 구조를 이해하면 새로운 파일이 생성되는 과정을 간단하게 생각해 볼 수 있다.

```
$ cat > file2
apple is red
^D
$ ls -li
2845304 -rw-r--r--    1 kimyh    graduate        13 Nov 18 17:44 file
2845305 -rw-r--r--    1 kimyh    graduate        13 Nov 18 20:10 file2
$
```

1. 새롭게 생성할 디렉터리에 동일한 이름의 항이 존재하는지 확인한다.
2. 아이노드 블록 하나를 할당 받는다.
3. 할당 받은 아이노드 블록에 파일의 정보를 저장한다.
4. 파일이 저장할 데이터의 크기에 따라 데이터 블록을 할당 받는다.

리눅스의 특징

- 리눅스는 다중 사용자를 지원하므로 소유에 대한 구분과 권한 설정이 중요하다.

파일의 소유권

- 파일이 어느 사용자의 것인지를 나타낸다.
- 모든 파일은 시스템에 등록된 사용자 중 한 사용자의 소유가 된다.

  관리자의 소유, 일반 사용자 홍길동의 소유

허가

- 모든 파일은 읽기, 쓰기, 실행 권한을 가지고 있다.

  각 권한은 설정이 되어 있을 수도 있고 그렇지 않을 수도 있다.
- 파일에 대한 권한은 사용자 유형에 따라 다르게 적용된다.

  파일의 소유자, 파일과 같은 그룹에 속한 사용자, 기타 사용자
- 접근 권한이라고도 한다.

소유권과 허가의 예

```
$ ls -l
-rw-r-----    1 kimyh    graduate        13 Nov 18 17:44 file
$
```

- 파일의 소유자 : kimyh라는 ID의 사용자
- 파일의 그룹 : graduate
- 허가

유형	소유주에 대한 권한	동일 그룹 사용자에 대한 권한	기타 사용자에 대한 권한
권한	r w -	r - -	- - -
의미	읽기 가능 쓰기 가능 실행 불가능	읽기 가능 쓰기 불가능 실행 불가능	읽기 불가능 쓰기 불가능 실행 불가능
8진수	6	4	0

소유권과 허가의 변경

소유권의 변경

- 시스템의 관리자만 수행할 수 있다.
- chown 명령 사용

허가의 변경

- 파일의 소유자나 시스템 관리자가 수행할 수 있다.
- chmod 명령 사용

```
$ chmod 644 file
$ ls -l file
-rw-r--r--    1 kimyh    graduate        13 Nov 18 17:44 file
$
```

※허가를 나타내는 644는 “rw-r--r--”을 의미한다.

파일의 유형

- 흔히 접하는 파일의 유형은 일반 파일, 실행 파일, 디렉터리 파일 등이다.
  이러한 파일을 통틀어 "일반 파일" 이라고 부른다.
- 리눅스 시스템은 다양한 개체를 파일로 다룰 수 있게 한다.
  주기억장치, 보조기억장치, 파이프, 터미널 연결 상태 등
  이러한 파일을 통틀어 "특수 파일" 이라고 부른다.

```
...
crw-------    1 root     root      14,   4 Apr 11  2002 audio
brw-rw----    1 root     floppy     2,   0 Apr 11  2002 fd0
brw-rw----    1 root     disk       8,   0 Apr 11  2002 sda
...
```

특수 파일을 지원하는 이유

사용자(관리자를 포함)가 디바이스들을 파일처럼 쉽게 사용할 수 있다.

터미널 연결 상태를 나타내는 파일 (1)

tty 명령으로 자신의 연결 상태를 나타내는 파일 확인할 수 있음

- 아래의 예에서 /dev/pts/7
- /dev/pts 디렉터리를 확인하면 현재 연결 중인 터미널의 수를 확인할 수 있음

```
$ tty
/dev/pts/7
$ ls -l /dev/pts
total 0
crw--w----    1 fineplus tty       136,   2 Nov 18 19:26 2
crw--w----    1 kimyh    tty       136,   7 Nov 18 19:57 7
$ who
fineplus pts/2    Nov 18 19:12 (210.92.29.253)
kimyh    pts/7    Nov 18 13:39 (202.31.201.117)
$
```

터미널 연결 상태를 나타내는 파일 (2)

터미널 연결 상태를 나타내는 파일로 출력하기

```
$ cat > /dev/pts/7
apple is red
apple is red
banana is yellow
banana is yellow
^C
$
```