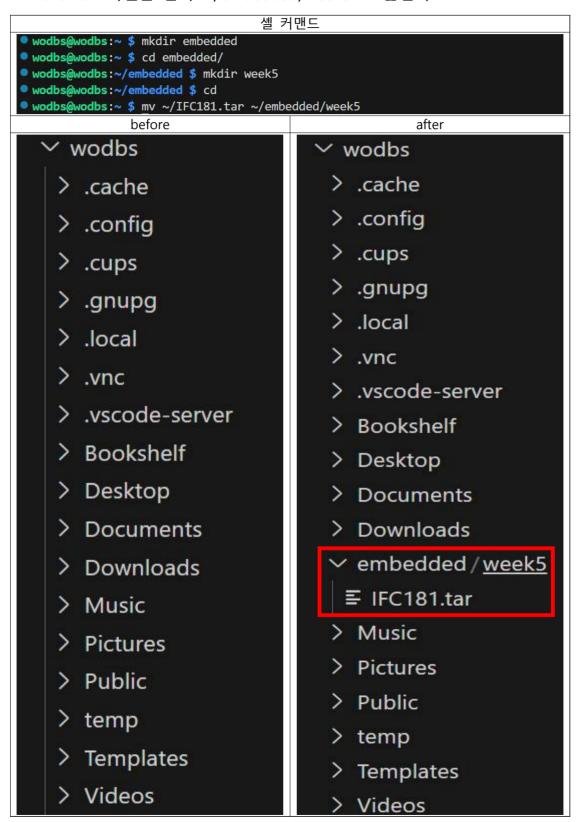
# 임베디드 응용 및 실습

- 5주차 과제 -

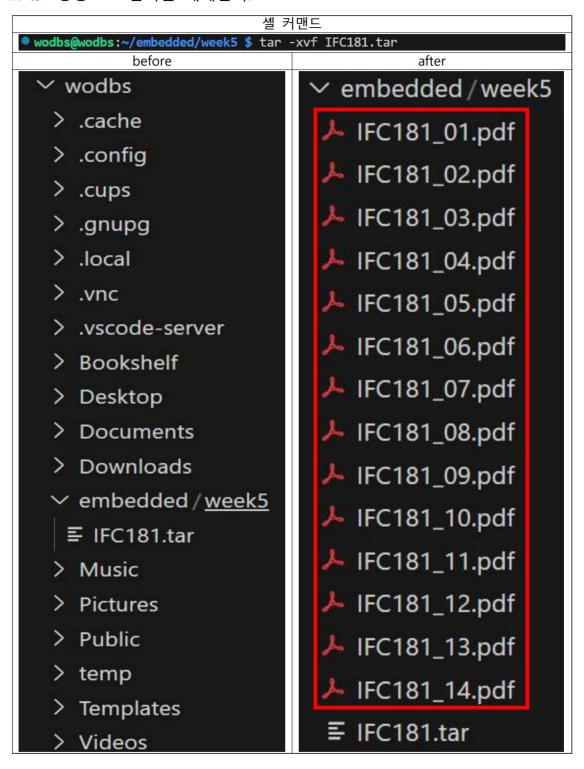
학번	2020161092
이름	이재윤

## <파일 압축 해제>

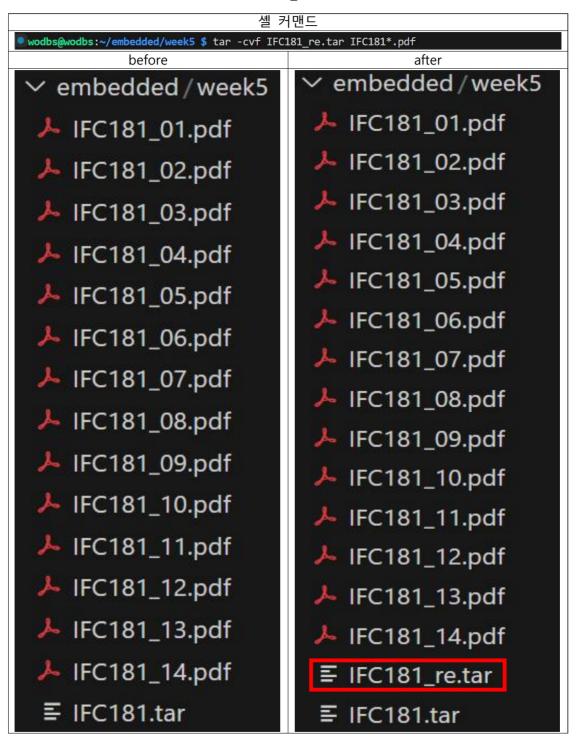
1. IFC181.tar 파일을 받아 ~/embedded/week5 로 옮긴다.



#### 2. tar 명령으로 압축을 해제한다.



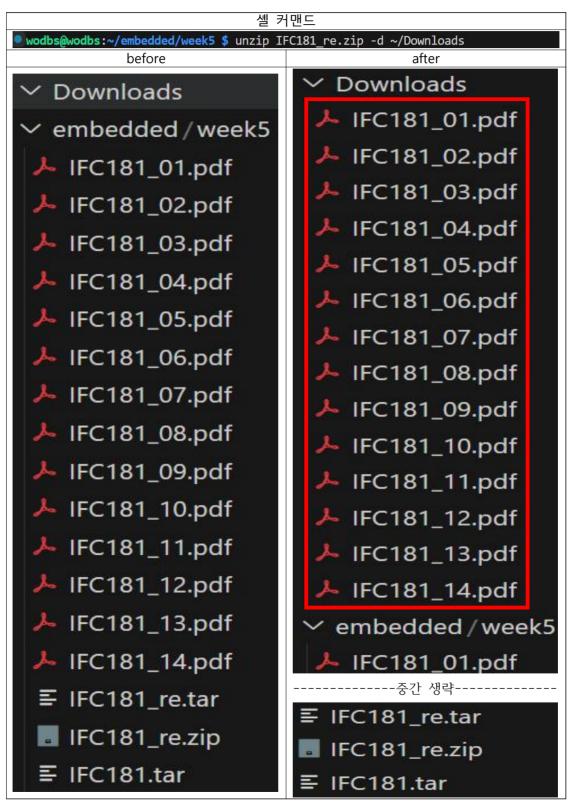
3. 2에서 압축 해제된 파일들을 IFC181 re.tar로 압축한다.



4. 2에서 압축해제된 파일들을 zip명령으로 IFC181 re.zip으로 압축한다.



- 5. 4에서 나온 IFC181 re.zip 파일을 unzip 명령을 통해 압축해제 한다.
- ~/embedded/week5 안에 IFC181\*.pdf가 이미 있기 때문에 Downloads 안에 압축 해제 하겠습니다.

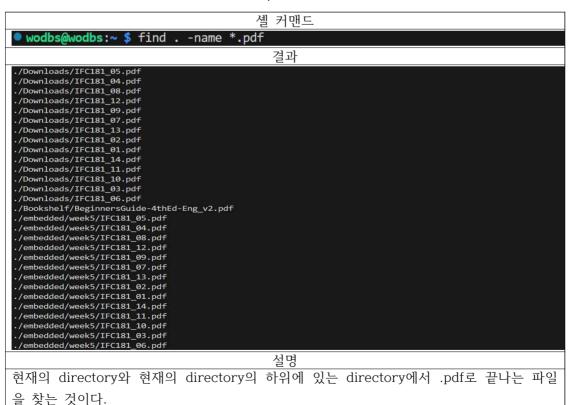


## <find 명령어 연습>

6. ~ (home) 디렉토리로 이동하고 find 명령을 통해 embedded폴더에서(하 위폴더 포함) .pdf 파일을 찾는 명령을 수행해본다.

```
셸 커맨드
 wodbs@wodbs:~ $ cd ~
wodbs@wodbs:~ $ find embedded -name *.pdf
                                           결과
embedded/week5/IFC181_05.pdf
embedded/week5/IFC181_04.pdf
embedded/week5/IFC181 08.pdf
embedded/week5/IFC181 12.pdf
embedded/week5/IFC181_09.pdf
embedded/week5/IFC181 07.pdf
embedded/week5/IFC181_13.pdf
embedded/week5/IFC181_02.pdf
embedded/week5/IFC181_01.pdf
embedded/week5/IFC181_14.pdf
embedded/week5/IFC181_11.pdf
embedded/week5/IFC181_10.pdf
embedded/week5/IFC181_03.pdf
embedded/week5/IFC181_06.pdf
```

- 7. 아래의 명령 결과가 동일한지 아닌지 결과를 보이고 동작 결과를 설명하 시오
- ① innosm@innosm:~ \$find . -name \*.pdf



#### 2 innosm@innosm:~ \$find ./ -name \*.pdf

```
| Move | Move
```

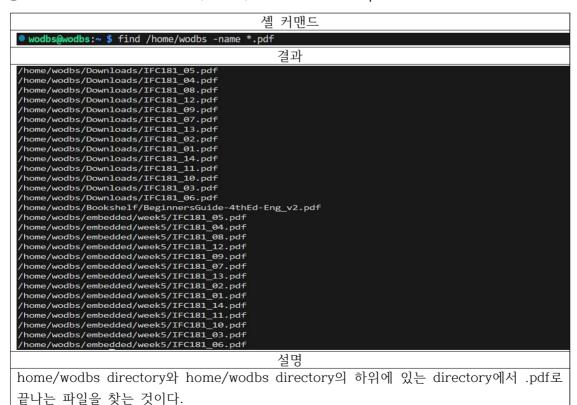
# (3) innosm@innosm:~ \$find -name \*.pdf

```
● wodbs@wodbs:~ $ find -name *.pdf

| Jownloads/IFC181_05.pdf | Jownloads/IFC181_04.pdf | Jownloads/IFC181_08.pdf | Jownloads/IFC181_09.pdf | Jownloads/IFC181_09.pdf | Jownloads/IFC181_12.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_00.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_00.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_11.pdf | Jownloads/IFC181_11.pdf | Jownloads/IFC181_11.pdf | Jownloads/IFC181_11.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_10.pdf | Jownloads/IFC181_00.pdf | Jown
```

#### 4 innosm@innosm:~ \$find ~ -name \*.pdf

#### (5) innosm@innosm:~ \$find /home/innosm -name \*.pdf



- find . -name \*.pdf, find ./ -name \*.pdf, find -name \*.pdf는 현재의 directory와 그 하위 directory에서 .pdf 파일을 찾는 것이기 때문에, 결과가 같다. 하지만 find ~ -name \*.pdf는 home directory와 home directory의 하위 directory에서 .pdf 파일을 찾는 것이기 때문에 결과가 다를 수 있다. 또한, find /home/wodbs -name \*.pdf도 home/wodbs directory와 home/wodbs의 하위 directory에서 .pdf 파일을 찾는 것이기 때문에 결과가 다를 수 있다. 하지만 현재 home directroy와 home/wodbs directory, 각각의 하위 directory에 다른 .pdf 파일이 없어 결과는 같다고 할 수 있다.
- 8. ~ (home) 디렉토리로 이동하고, week5 폴더가 있는지 검색하려고 한다. 적절한 명령을 수행하여 week5 폴더를 검색하고 결과를 보이시오.

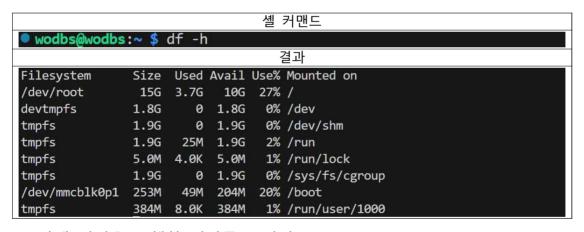
```
셸 커맨드

wodbs@wodbs:~ $ cd ~
wodbs@wodbs:~ $ find ~ -type d -name week5
결과
/home/wodbs/embedded/week5
```

#### <파일 용량 확인>

9. 아래 명령을 수행한 결과를 보이시오.

df -h



10. 아래 명령을 수행한 결과를 보이시오.

 $cd \sim \rightarrow cd \ embedded \rightarrow du -h$ 

```
셸 커맨드

wodbs@wodbs:~ $ cd ~
wodbs@wodbs:~ $ cd embedded
wodbs@wodbs:~/embedded $ du -h
결과

16M ./week5
16M .
```

- 11. 10번에서 현재 embedded 폴더의 총 사용량은(하위폴더 포함, 단위 표시)?
- 16MB
- 12. cd embedded/week5 를 수행하여 위치를 이동하고, 아래 명령을 차례로 수행하시오
- 1) df.

```
셸 커맨드

wodbs@wodbs:~ $ cd embedded/week5
wodbs@wodbs:~/embedded/week5 $ df .
결과

Filesystem 1K-blocks Used Available Use% Mounted on /dev/root 14986204 3845844 10478852 27% /
```

2) 이 폴더에 temp file 파일 생성하고 파일에 1을 기록

```
셸 커맨드
wodbs@wodbs:~/embedded/week5 $ echo 1 > temp file
                               before
wodbs@wodbs:~/embedded/week5 $ 1s
IFC181_01.pdf IFC181_06.pdf IFC181_11.pdf IFC181_re.zip
 IFC181_02.pdf IFC181_07.pdf IFC181_12.pdf IFC181.tar
 IFC181_03.pdf IFC181_08.pdf IFC181_13.pdf
 IFC181_04.pdf IFC181_09.pdf IFC181_14.pdf
 IFC181 05.pdf IFC181 10.pdf IFC181 re.tar
wodbs@wodbs:~/embedded/week5 $ 1s
  IFC181_01.pdf IFC181_06.pdf IFC181_11.pdf
                                                  IFC181_re.zip
 IFC181_02.pdf IFC181_07.pdf IFC181_12.pdf
                                                  IFC181.tar
 IFC181_03.pdf IFC181_08.pdf IFC181_13.pdf temp_file
  IFC181 04.pdf IFC181 09.pdf IFC181 14.pdf
 IFC181_05.pdf IFC181_10.pdf IFC181_re.tar
```

3) stat temp\_file 명령을 통해 파일의 크기를 확인

```
셸 커맨드
wodbs@wodbs:~/embedded/week5 $ stat temp_file
                                 결과
  File: temp_file
  Size: 2
                       Blocks: 8
                                         IO Block: 4096
                                                          regular file
Device: b302h/45826d
                       Inode: 261656
                                         Links: 1
Access: (0644/-rw-r--r--) Uid: (1000/ wodbs) Gid: (1000/
                                                                wodbs)
Access: 2024-10-05 13:15:09.672813493 +0100
Modify: 2024-10-05 13:15:09.672813493 +0100
Change: 2024-10-05 13:15:09.672813493 +0100
Birth: -
```

#### 4) df.

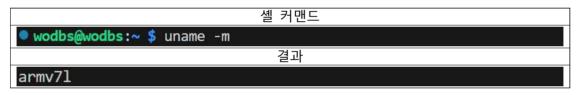


1)과 4)에서 줄어든 용량과 3)에서 확인한 용량이 다르다면 그 이유는?

- 1)과 4)에서 줄어든 용량 : 4Kbyte
- 3)에서 확인한 용량 : 2byte
- 실제 용량은 2byte지만, 파일 시스템에서 블록 크기는 일반적으로 4KB이기 때문에, 파일 이 2byte여도 파일 시스템에서는 최소 1개의 블록을 할당하기 때문에 4KB의 공간을 차지하는 것이다.

# <cpu architecture>

13. 현재 사용하는 라즈비언 OS가 몇비트 시스템인지 확인하고 결과를 첨부하시오.



- 32비트 시스템이다.