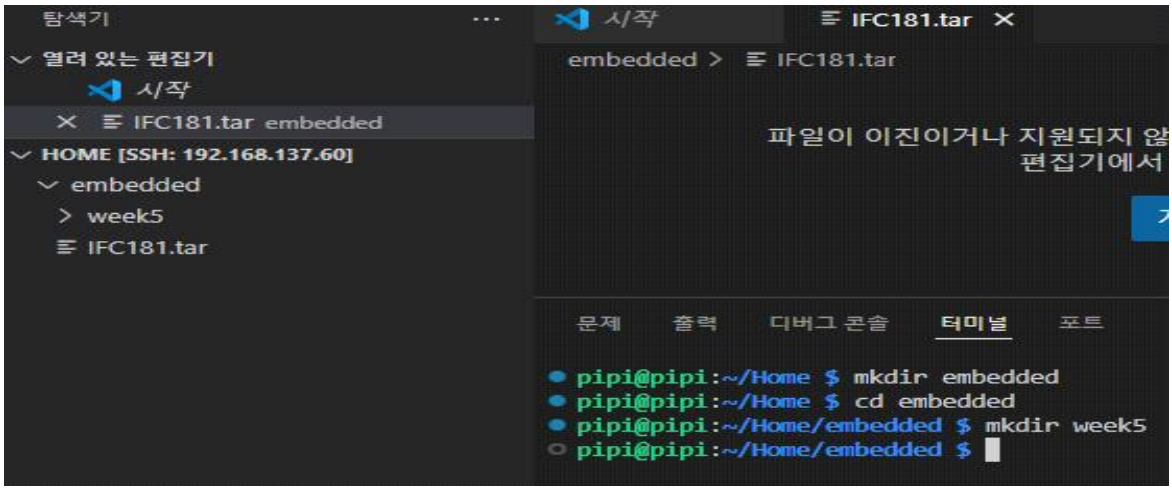
임베디드실습 및 응용

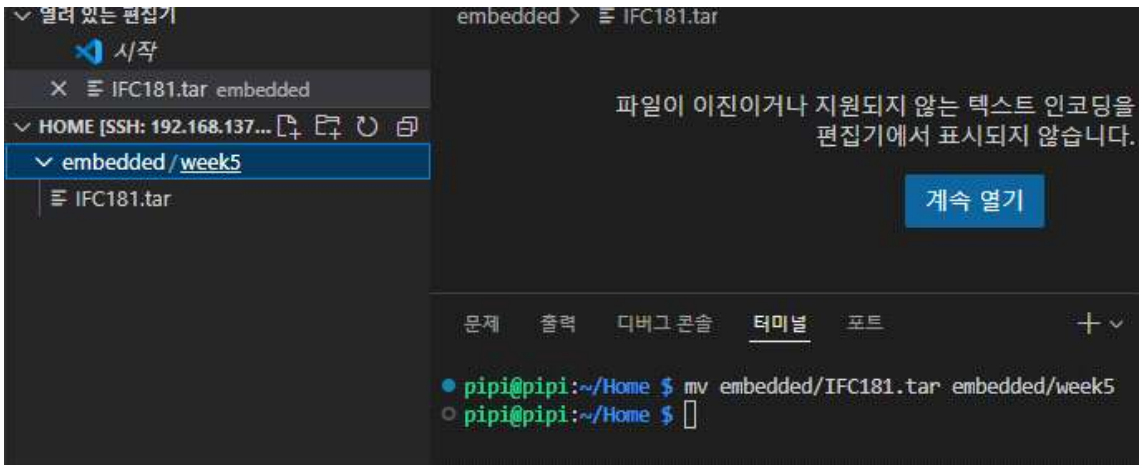
(5주차 과제)

2020161047 박종혁

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

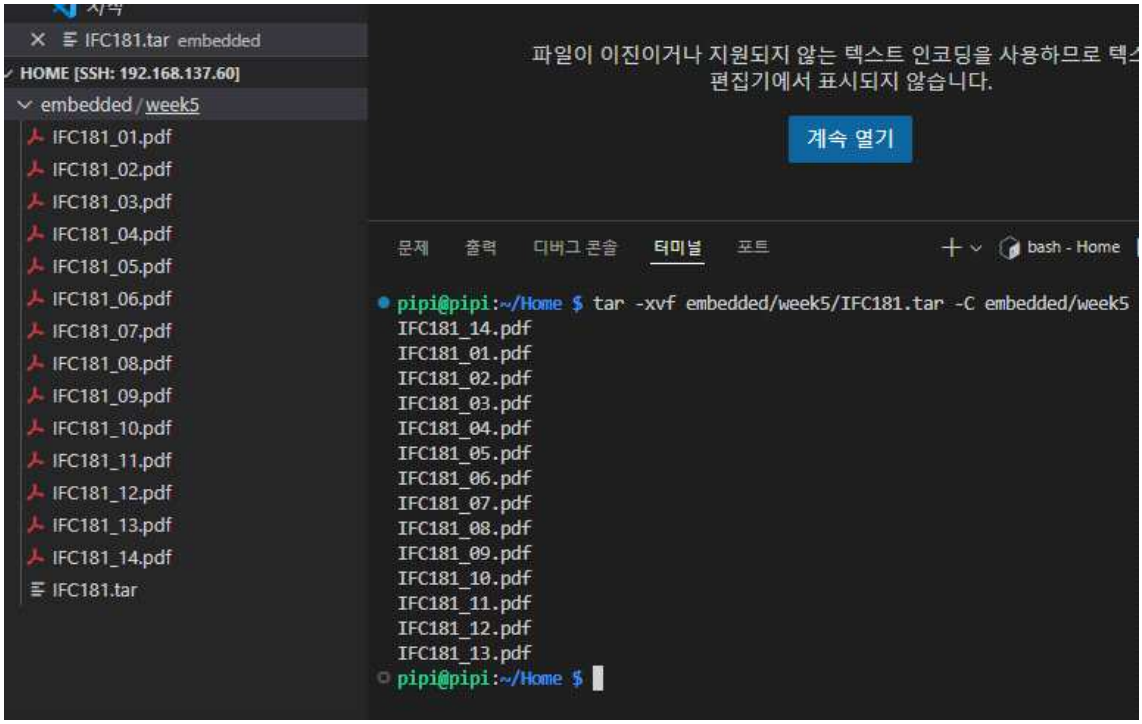


현재 IFC181.tar파일이 embedded에 들어가있어서 다음과 같이 week5로 파일을 옮겼습니다.



2. tar 명령으로 압축을 해제한다.

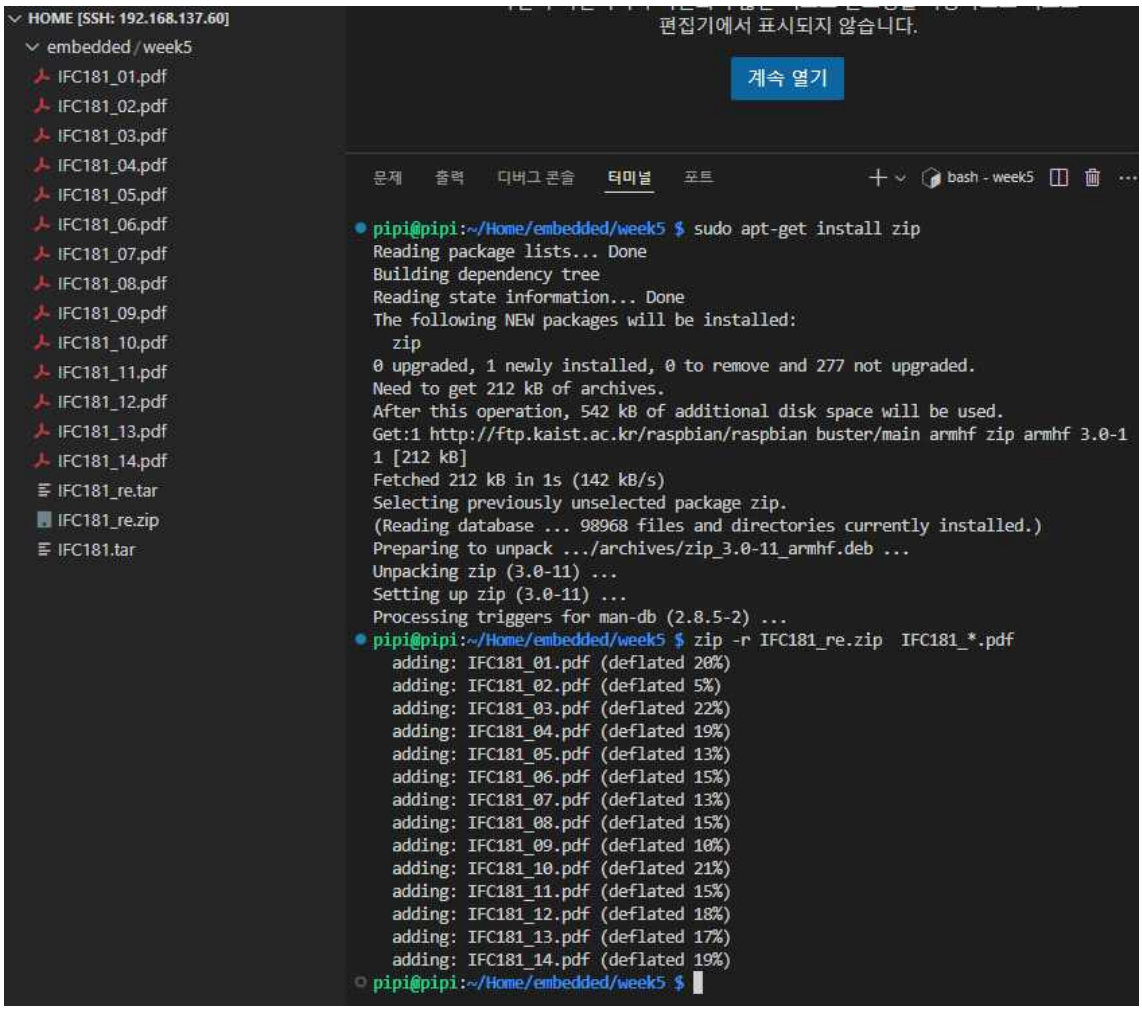
다음과 같이 tar 명령으로 week5에 있는 tar파일을 압축해제 하였습니다.



3. 2에서 압축해제된 파일들을 IFC181_re.tar로 압축한다.

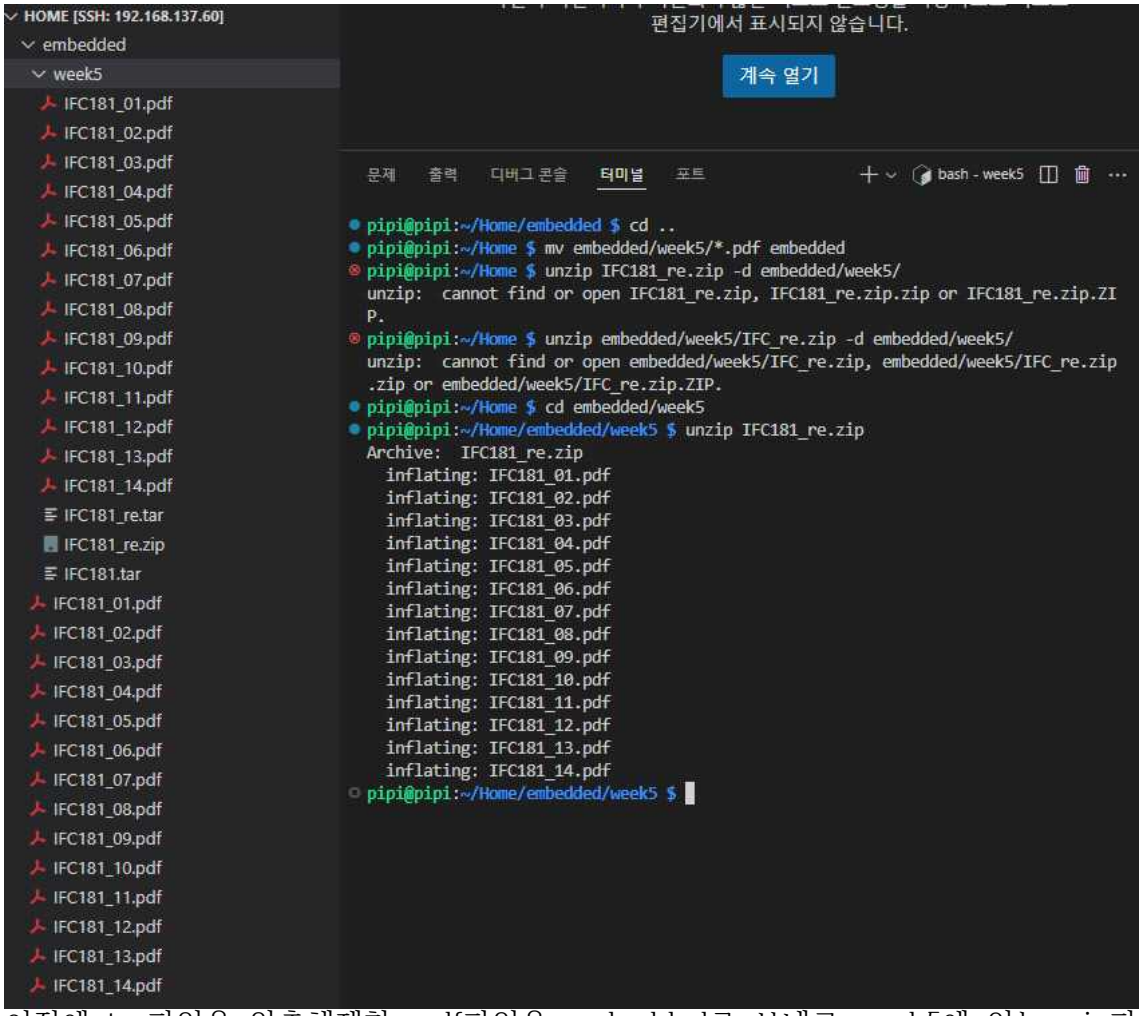
```
pipi@pipi:~/Home/embedded/week5 $ tar -cvf IFC181_re.tar IFC181_*.pdf
IFC181_01.pdf
IFC181_02.pdf
IFC181_03.pdf
IFC181_04.pdf
IFC181_05.pdf
IFC181_06.pdf
IFC181_07.pdf
IFC181_08.pdf
IFC181_09.pdf
IFC181_10.pdf
IFC181_11.pdf
IFC181_12.pdf
IFC181_13.pdf
IFC181_14.pdf
pipi@pipi:~/Home/embedded/week5 $
```

4. 2에서 압축해제된 파일들을 zip명령으로 IFC181_re.zip으로 압축한다.
- 이때 zip 명령을 수행할 수 있도록 zip을 설치한다.
: sudo apt-get install zip



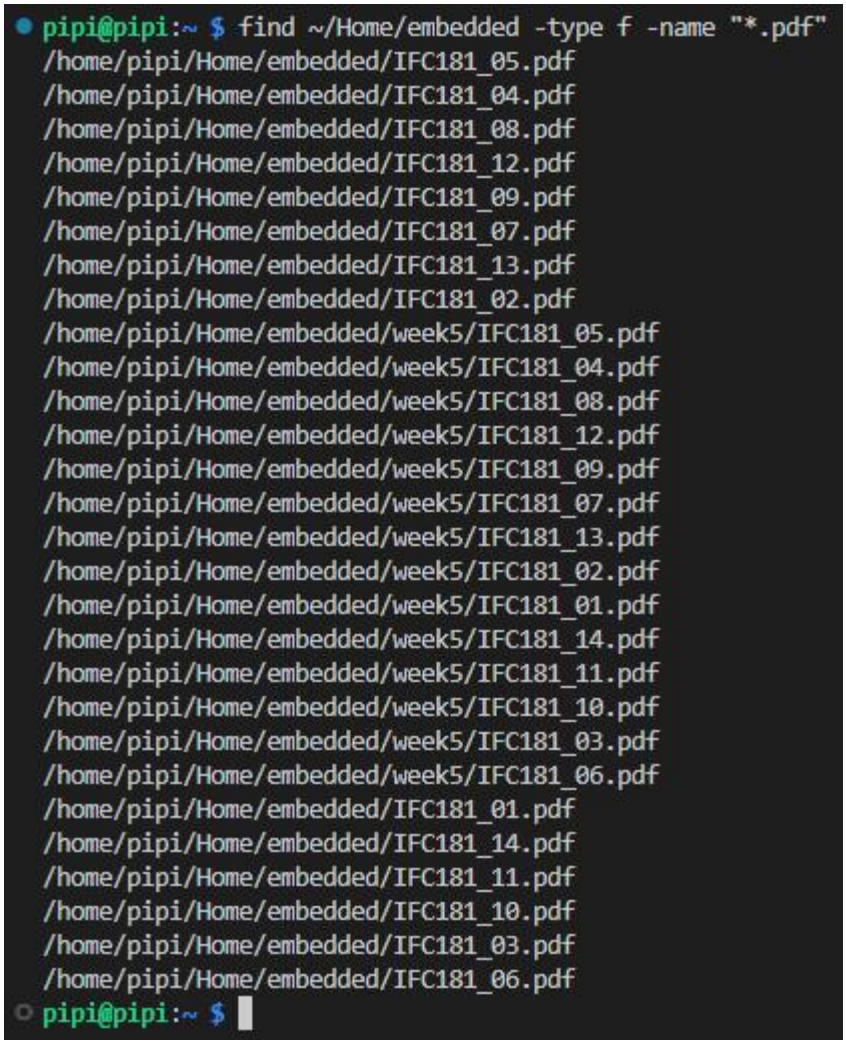
-> 먼저 zip을 설치하고 다음 명령어로 압축해제된 파일들을 IFC181_re.zip으로 압축했습니다.

5. 4에서 나온 IFC181_re.zip 파일을 unzip 명령을 통해 압축해제 한다.



이전에 tar파일을 압축해제한 .pdf파일을 embedded로 보내고 week5에 있는 .zip파일을 압축해제 하였습니다.

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



7. 아래의 명령 결과가 동일한지 아닌지 결과를 보이고 동작 결과를 설명하시오
innosm@innosm:~ $find . -name *.pdf
innosm@innosm:~ $find ./ -name *.pdf
innosm@innosm:~ $find -name *.pdf
innosm@innosm:~ $find ~ -name *.pdf
innosm@innosm:~ $find /home/innosm -name *.pdf
- 첫 번째 명령어 수행

```
pipi@pipi:~ $ find . -name *.pdf
./Bookshelf/BeginnersGuide-4thEd-Eng_v2.pdf
./Home/embedded/IFC181_05.pdf
./Home/embedded/IFC181_04.pdf
./Home/embedded/IFC181_08.pdf
./Home/embedded/IFC181_12.pdf
./Home/embedded/IFC181_09.pdf
./Home/embedded/IFC181_07.pdf
./Home/embedded/IFC181_13.pdf
./Home/embedded/IFC181_02.pdf
./Home/embedded/week5/IFC181_05.pdf
./Home/embedded/week5/IFC181_04.pdf
./Home/embedded/week5/IFC181_08.pdf
./Home/embedded/week5/IFC181_12.pdf
./Home/embedded/week5/IFC181_09.pdf
./Home/embedded/week5/IFC181_07.pdf
./Home/embedded/week5/IFC181_13.pdf
./Home/embedded/week5/IFC181_02.pdf
./Home/embedded/week5/IFC181_01.pdf
./Home/embedded/week5/IFC181_14.pdf
./Home/embedded/week5/IFC181_11.pdf
./Home/embedded/week5/IFC181_10.pdf
./Home/embedded/week5/IFC181_03.pdf
./Home/embedded/week5/IFC181_06.pdf
./Home/embedded/IFC181_01.pdf
./Home/embedded/IFC181_14.pdf
./Home/embedded/IFC181_11.pdf
./Home/embedded/IFC181_10.pdf
./Home/embedded/IFC181_03.pdf
./Home/embedded/IFC181_06.pdf
pipi@pipi:~ $
```

현재 디렉토리를 표현하는 .에서 name이 .pdf로 끝나는 파일 찾습니다.
현재 모든 경로에 있는 .pdf파일의 경로를 보여줍니다.

- 두 번째 명령어 수행

./는 현재 디렉토리와 해당 디렉토리 내의 경로에서 .pdf 파일을 찾습니다.
~에서 실행했으므로 .와 ./차이는 없습니다.

- 세 번째 명령어 수행

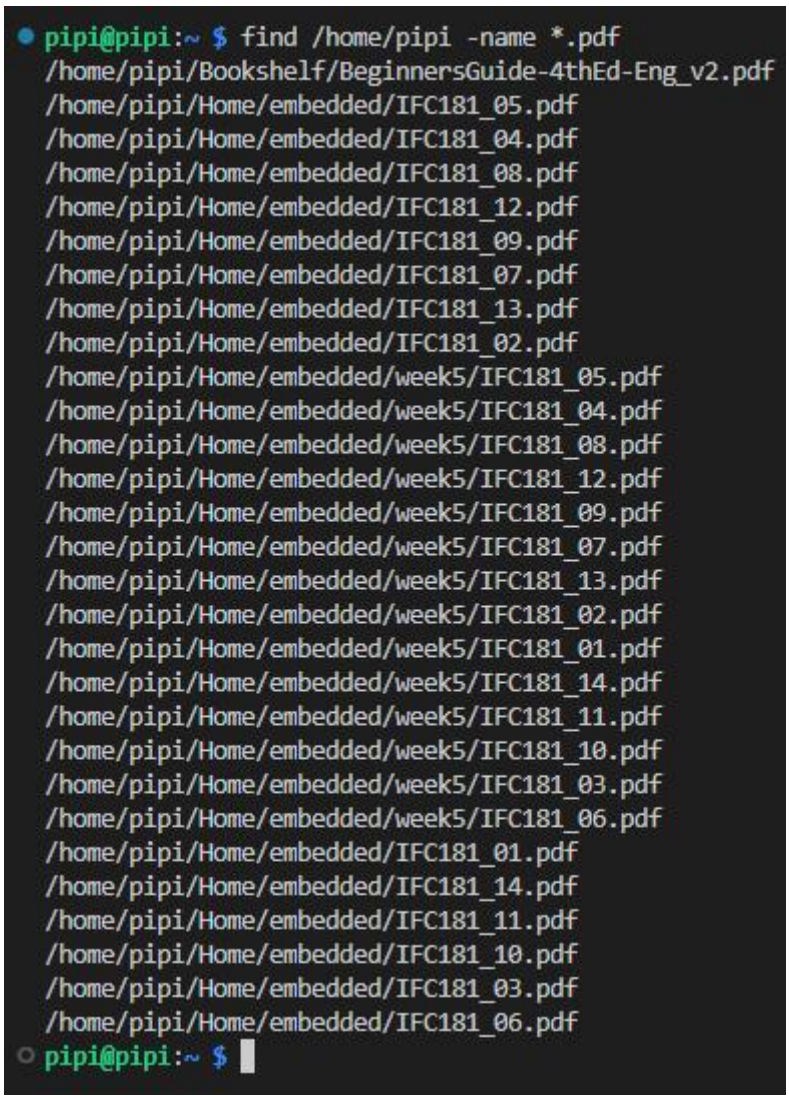
경로 표기없이 -name *.pdf 명령을 했으므로 현재 디렉토리에서 .pdf로 끝나는 파일을 찾아냅니다.

- 네 번째 명령어 수행

```
pipi@pipi:~ $ find ~ -name *.pdf
/home/pipi/Bookshelf/BeginnersGuide-4thEd-Eng_v2.pdf
/home/pipi/Home/embedded/IFC181_05.pdf
/home/pipi/Home/embedded/IFC181_04.pdf
/home/pipi/Home/embedded/IFC181_08.pdf
/home/pipi/Home/embedded/IFC181_12.pdf
/home/pipi/Home/embedded/IFC181_09.pdf
/home/pipi/Home/embedded/IFC181_07.pdf
/home/pipi/Home/embedded/IFC181_13.pdf
/home/pipi/Home/embedded/IFC181_02.pdf
/home/pipi/Home/embedded/week5/IFC181_05.pdf
/home/pipi/Home/embedded/week5/IFC181_04.pdf
/home/pipi/Home/embedded/week5/IFC181_08.pdf
/home/pipi/Home/embedded/week5/IFC181_12.pdf
/home/pipi/Home/embedded/week5/IFC181_09.pdf
/home/pipi/Home/embedded/week5/IFC181_07.pdf
/home/pipi/Home/embedded/week5/IFC181_13.pdf
/home/pipi/Home/embedded/week5/IFC181_02.pdf
/home/pipi/Home/embedded/week5/IFC181_01.pdf
/home/pipi/Home/embedded/week5/IFC181_14.pdf
/home/pipi/Home/embedded/week5/IFC181_11.pdf
/home/pipi/Home/embedded/week5/IFC181_10.pdf
/home/pipi/Home/embedded/week5/IFC181_03.pdf
/home/pipi/Home/embedded/week5/IFC181_06.pdf
/home/pipi/Home/embedded/IFC181_01.pdf
/home/pipi/Home/embedded/IFC181_14.pdf
/home/pipi/Home/embedded/IFC181_11.pdf
/home/pipi/Home/embedded/IFC181_10.pdf
/home/pipi/Home/embedded/IFC181_03.pdf
/home/pipi/Home/embedded/IFC181_06.pdf
pipi@pipi:~ $
```

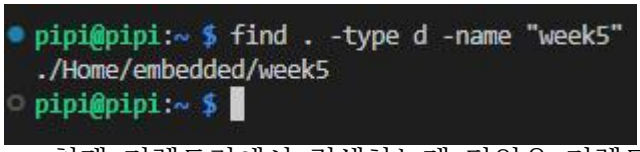
경로를 ~로 지정해주었으므로 ~에서 이름이 .pdf로 끝나는 파일을 찾아줍니다.

- 다섯 번째 명령어 수행



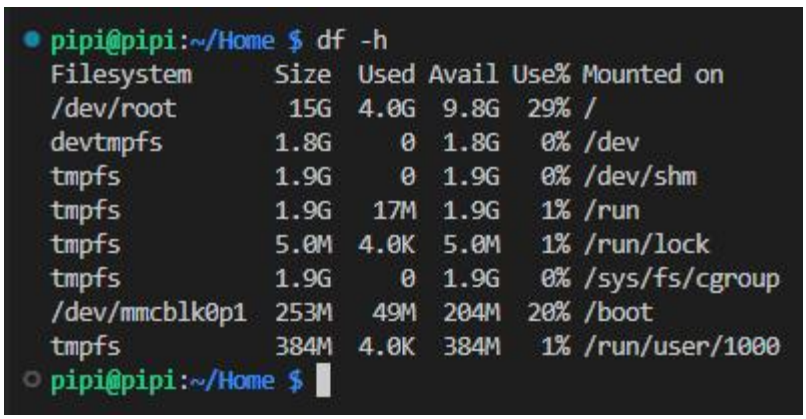
경로를 /home/pipi로 지정해주었기에 그 경로를 포함하여 이름이 .pdf로 끝나는 파일을 찾아냅니다.

8. ~ (home) 디렉토리로 이동하고, week5 폴더가 있는지 검색하려고 한다.
적절한 명령을 수행하여 week5 폴더를 검색하고 결과를 보이시오.



-> 현재 디렉토리에서 검색하는데 타입은 디렉토리로 디렉토리만 검색한다.
-name “week5”로 이름이 week5인 항목을 검색한다.

9. 아래 명령을 수행한 결과를 보이시오.
df -h



10. 아래 명령을 수행한 결과를 보이시오.
cd ~
cd embedded
du -h

다만 홈 디렉토리에 embedded라는 폴더가 있고 home디렉토리에 embedded 폴더가 있습니다. 위에서 했던 작업은 home 디렉토리에 embedded 폴더에서 수행했기 때문에 사용량이 다릅니다.
따라서 위 작업을 한 폴더의 사용량을 보고 싶다면 다음과 같이 명령하여 실행결과를 도출하겠습니다.

```
pipi@pipi:~ $ cd Home
pipi@pipi:~/Home $ ls
embedded
pipi@pipi:~/Home $ cd embedded
pipi@pipi:~/Home/embedded $ du -h
16M     ./week5
20M     .
pipi@pipi:~/Home/embedded $
```

11. 10번에서 현재 embedded 폴더의 총 사용량은(하위폴더 포함, 단위 표시)?
-> 현재 embedded 폴더의 총 사용량은 10번의 결과로 알 수 있다시피 4.0KB입니다.
home 폴더 안에 있는 embedded폴더의 총 사용량은 현재에서 20MB이고 week5의 폴더에서의 사용량은 16MB입니다.

12. cd embedded/week5 를 수행하여 위치를 이동하고, 아래 명령을 차례로 수행하시오

1) df .
2) 이 폴더에 temp_file 파일 생성하고 파일에 1을 기록
3) stat temp_file 명령을 통해 파일의 크기를 확인
4) df .

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

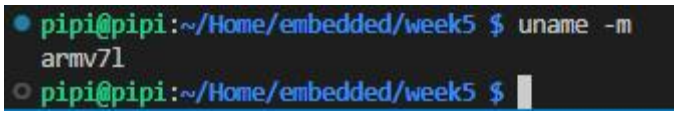
```
pipi@pipi:~/Home/embedded/week5 $ df .
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root       14986204 4096968  10227728  29% /
pipi@pipi:~/Home/embedded/week5 $
```

```
pipi@pipi:~/Home/embedded/week5 $ cat > temp_file
1
pipi@pipi:~/Home/embedded/week5 $ stat temp_file
  File: temp_file
  Size: 2               Blocks: 8          IO Block: 4096   regular file
Device: b302h/45826d    Inode: 269171      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/    pipi)   Gid: ( 1000/    pipi)
Access: 2024-10-08 05:09:16.702048723 +0100
Modify: 2024-10-08 05:09:18.702035665 +0100
Change: 2024-10-08 05:09:18.702035665 +0100
 Birth: -
pipi@pipi:~/Home/embedded/week5 $ df .
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root       14986204 4096972  10227724  29% /
pipi@pipi:~/Home/embedded/week5 $
```

1)과 4)에서 줄어든 용량은 4KB로 3)에서 확인할 수 있는 IO Block이 4096인 4KB와 같은 값을 갖게 되고 이는 temp_file을 생성하기 전과 후의 변경된 사용량과 같습니다. 따라서 temp_file이 2바이트로 작더라도 디스크는 블록 단위로 저장하므로 사용량은 4KB가 증가한 것으로 알 수 있습니다.

<cpu architecture>

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



armv7l로 실행결과가 나왔으므로 32비트 시스템임을 알 수 있습니다.