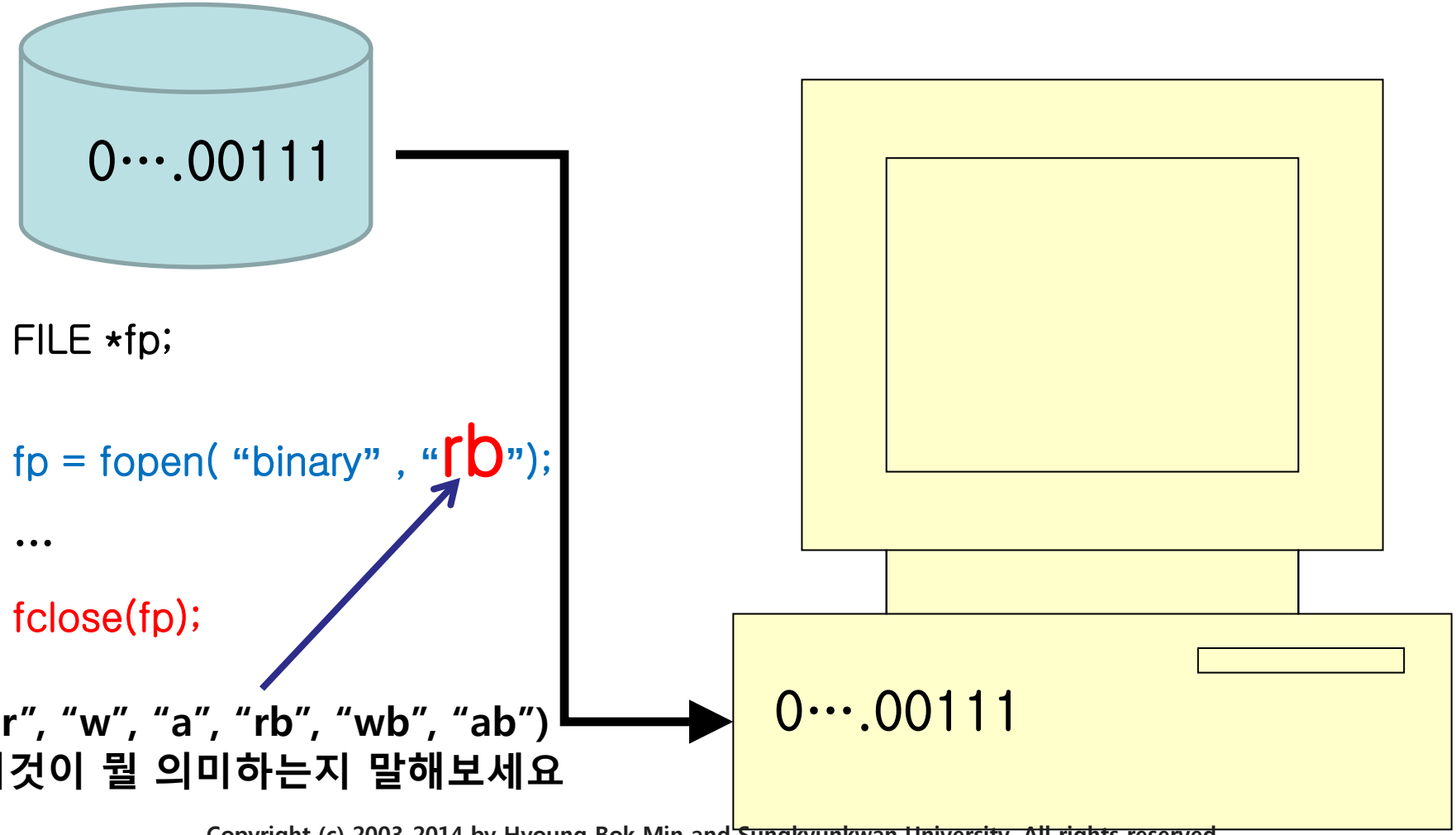
실습

Handling Binary Files

For copyright and license information,
http://class.icc.skku.ac.kr/~min/program/license.html

파일 열기(file open), 파일의 닫기(file close)

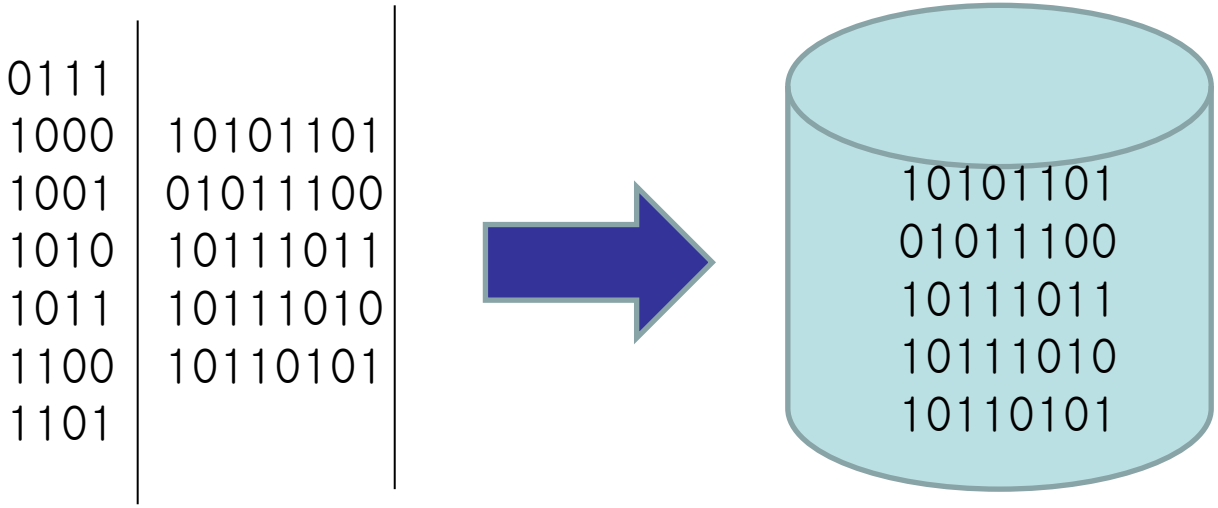


Text File

- 모든 data(char, int, short, double, …)는 computer 내에서 binary number로 표현된다
- Text file에서는 **모든 data가 character로 변환되어 저장**된다

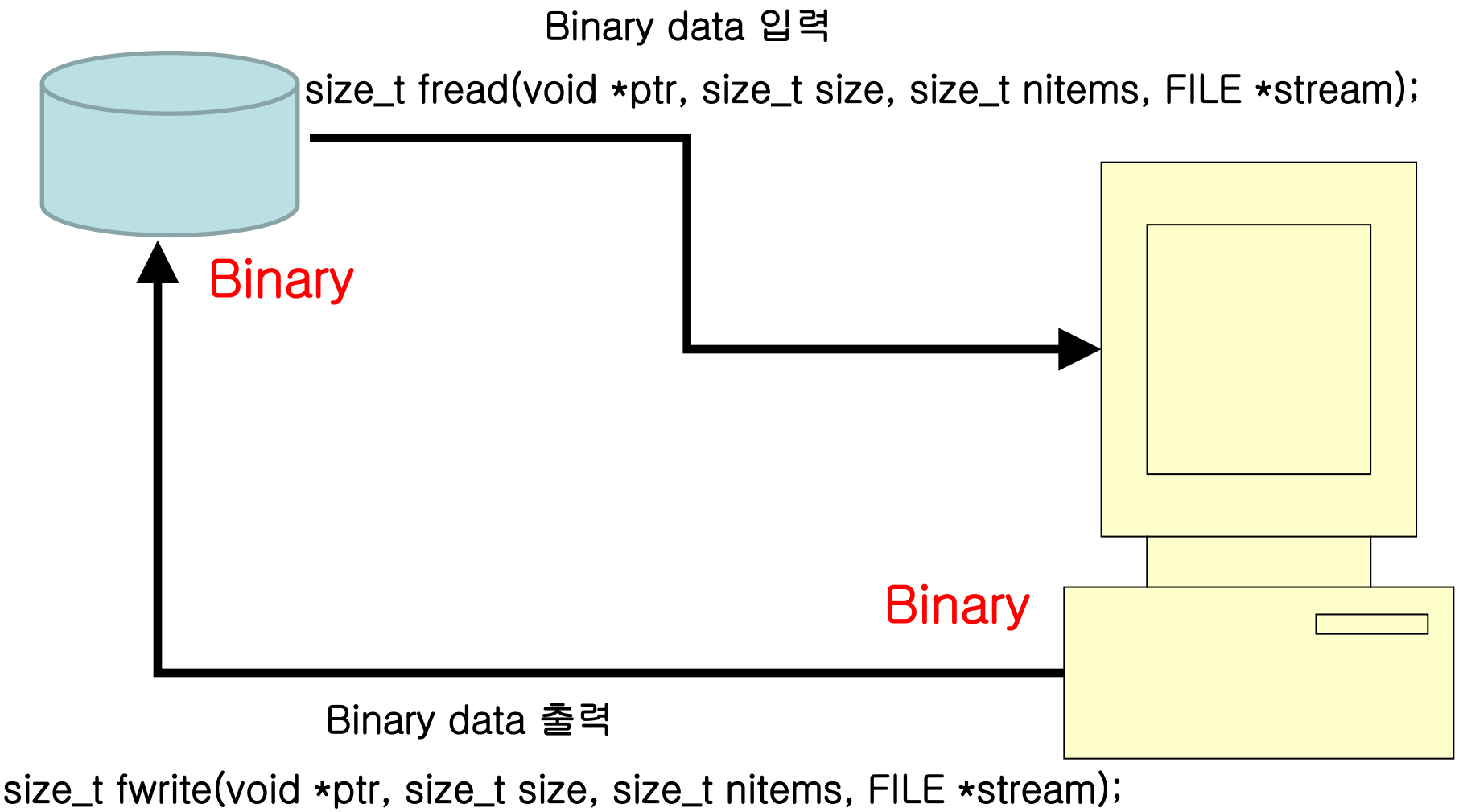
Binary File

- Data are written into file **the same as memories and registers** of a computer
  - 1 byte takes an address
- (example) Image files such as .bmp, .jpg, etc., video files, .zip, .pdf, .lnk, …



Binary data 입/출력

(컴퓨터 내의 표현 그대로 저장)

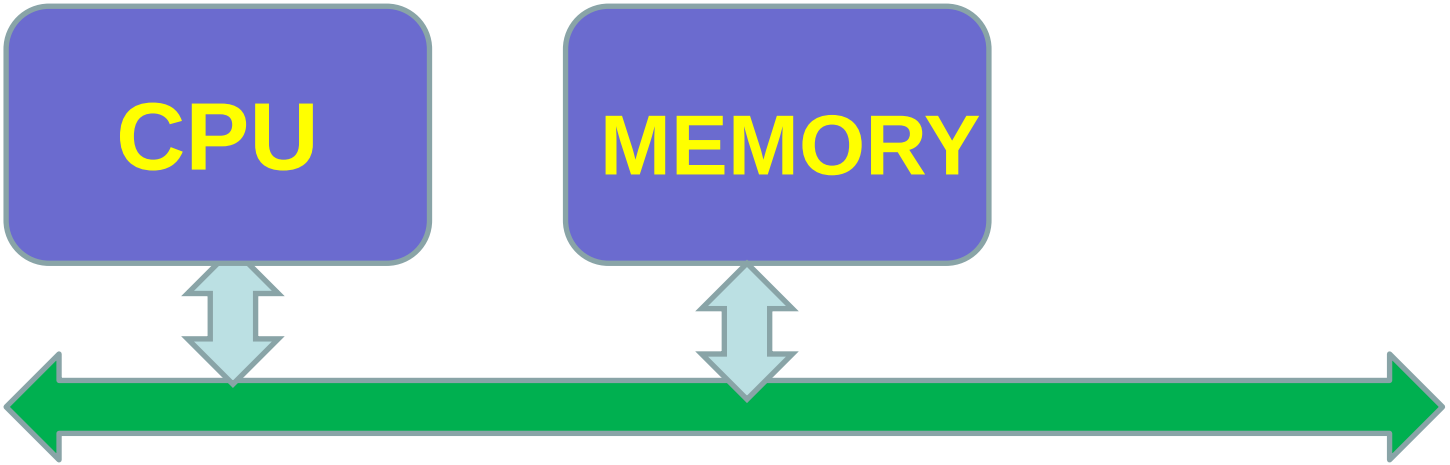


Binary File - Issue

- 32-bit integer type, for example, holds 4 bytes, which takes 4 addresses.
- Which byte takes lower address in memory ?
- Which byte is written first? (compatibility)
  - Big-endian : MSB first
    - Motorola 68000, IBM z/Architecture, etc
  - Little-endian : LSB first
    - Intel x86
  - Bi-endian : Both possible (selected by OS)
    - ARM, PowerPC, MIPS

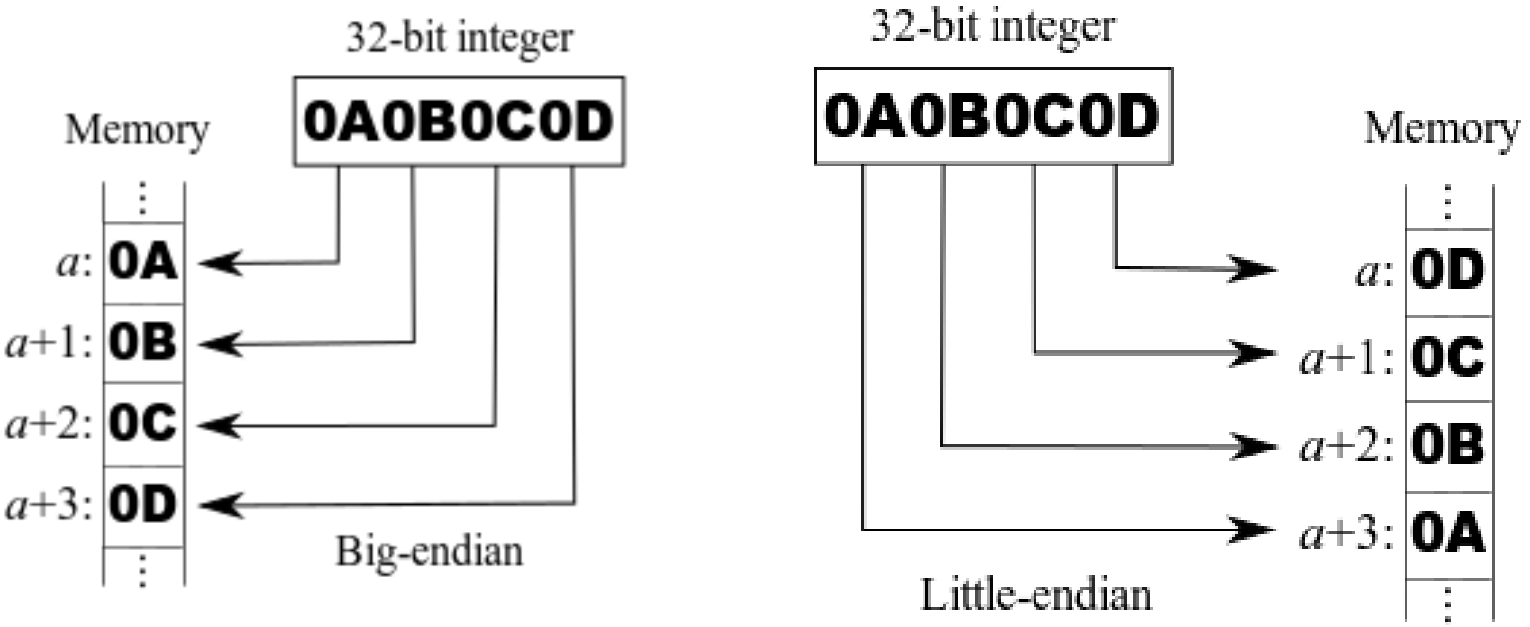
Big-endian, Littile-endian

- Memory is byte-addressable,
- while CPU registers are words of multiple bytes



Big-endian, Littile-endian

- Memory is byte-addressable,
- while CPU registers are words of multiple bytes



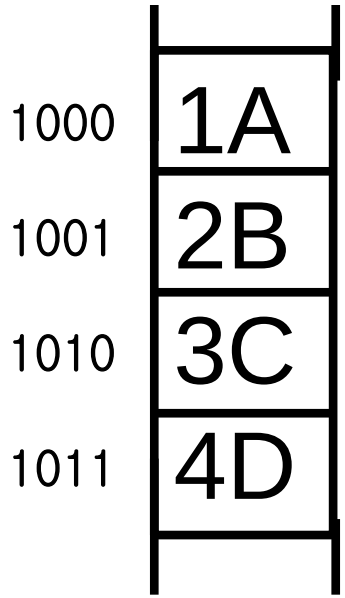
Images, courtesy of R.S. Shaw who published in public domain

fwrite() writes memory to file as is

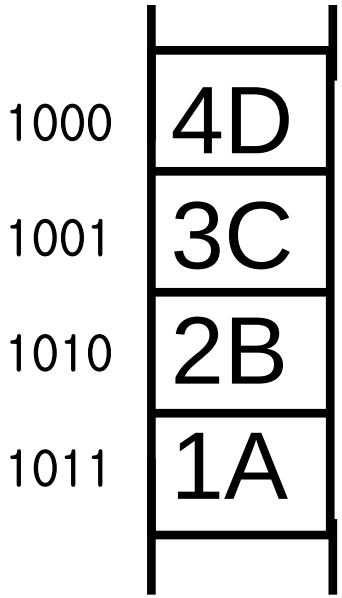
(exampe) int number = 0x1A2B3C4D;

fwrite(&number, sizeof(int), 1, fp);

- Big-endian



Little-endian

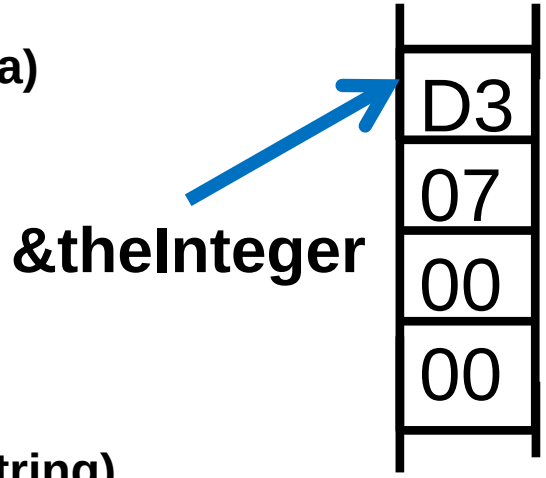


Binary Data 출력 (컴퓨터 내의 표현 그대로 저장)

size_t fwrite(void *ptr, size_t size, size_t nitems, FILE *stream);
Returns number of items actually written

-정수 출력 (int, long, float, double, char 등 1개 data)

```
int theInteger = 2003;
If (fwrite(&theInteger, sizeof(int), 1, fp) < 1) {
        fprintf(stderr, "fail to write integer);
        exit(1);
}
```



-Array 출력 (갯수가 변동되는 여러 개의 data, <예> string)

```
char *string = "This is my book";
int len = strlen(string);
fwrite(&len, sizeof(int), 1, fp);
If (fwrite(string, sizeof(char), len, fp) < len) {
        fprintf(stderr, "fail to write\n");
        exit(1);
}
```

Binary Data 입력 (표현 그대로 저장된 데이터 읽기)

```
size_t  fread(void *ptr, size_t size, size_t nitems, FILE *stream);
        Returns number of items actually read
```

-정수 입력 (int, float, double, char 등, 1개 data)

```
int theInteger;
If (fread(&theInteger, sizeof(int), 1, fp) < 1) {
        fprintf(stderr, “fail to read integer);   exit(1);
}
```

-Array 입력 (갯수가 변동되는 여러 개의 data, <예> string)

```
char *string;      int len;
fread(&len, sizeof(int), 1, fp);
string = (char *)maloc((len+1)*sizeof(char));
if(fread(string, sizeof(char), len, fp) < len) {
        fprintf(stderr, “fail to read\n”);
        exit(1);
}
string[len] = (char)0;
```

ftell(), fseek()

- long ftell(FILE *stream)
  - Returns position (in bytes) from the beginning of the file
- int fseek(FILE *stream, long offset, int whence)
  - Set position of file pointer (returns 0 if sucessful)
  - whence = SEEK_SET : offset from the beginning
  - SEEK_CUR : current position + offset
  - SEEK_END : EOF + offset

```
long  value;
fseek(fp, 0L, SEEK_END);
value = ftell(fp);      /* value = ? */
```

실습

- Windows Shortcut file (바로가기)

  filename**.lnk**

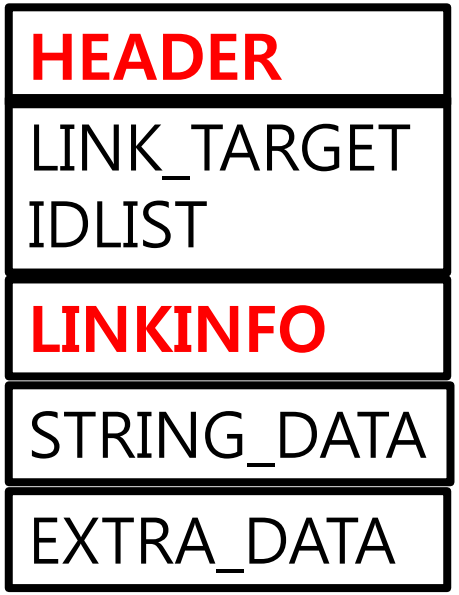
- Given a shortcut file, identify the 2 items inside the shortcut file.

  - Is this for file or folder ?

  - Full path name of the target file to which this shortcut points

    C:\Users\John\Documents\hello.txt

Windows Shortcut File Format



HasLinkTargetIDList bit @ 2.1.1

HasLinkInfo bit @ 2.1.1

hasName bit @ 2.1.1

All fields may not exist except HEADER

Header

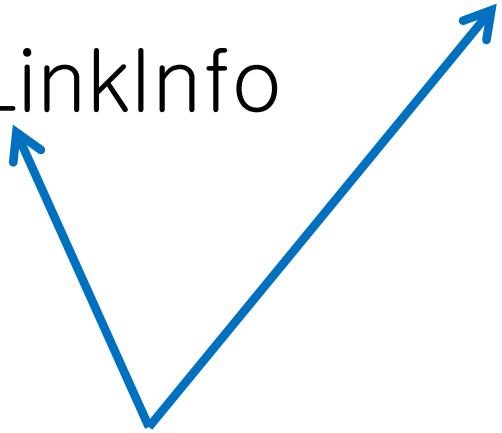
- **HeaderSize** (4 bytes,DWORD): The size, in bytes, of this structure. This value MUST be 0x0000004C.
- **LinkCLSID** (16 bytes): A class identifier (CLSID). This value MUST be 01-14-02-00-00-00-00-00-C0-00-00-00-00-00-00-46
- **LinkFlags** (4 bytes,DWORD): A LinkFlags (section 2.1.1) that specifies information about the shell link and the presence of optional portions of the structure.
- **FileAttributes** (4 bytes,DWORD): A FileAttributesFlags (section 2.1.2) that specifies information about the link target.
- **and more …**

LinkFlags

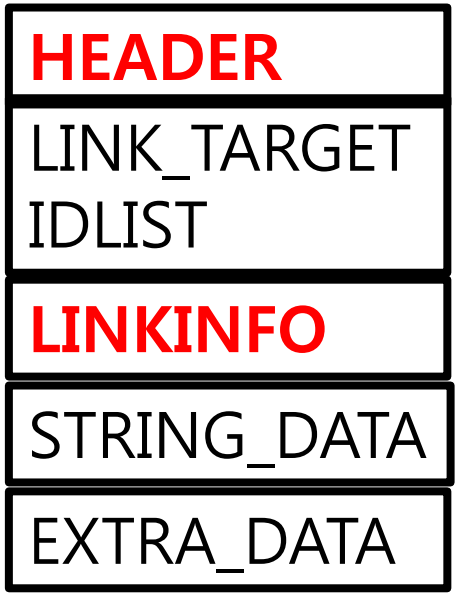
- 4 bytes (DWORD) in header, i.e., 32-bit integer

Bit 0 : hasLinkTargetIDList

Bit 1 : hasLinkInfo



- 1 if exist, 0 otherise
- Bit 0 means LSB



FileAttrbutes (from slide 13)

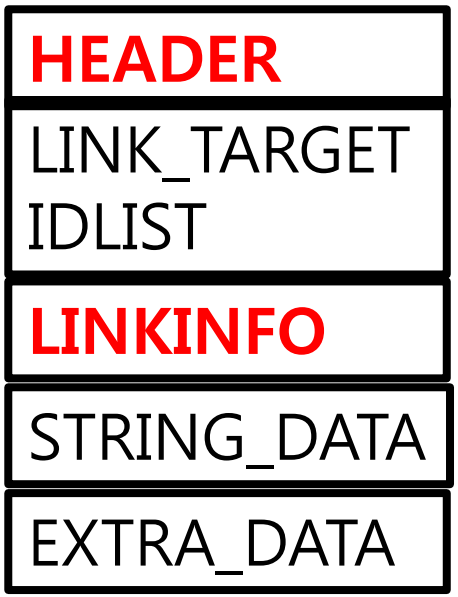
- 4 bytes (DWORD) in header, i.e. 32-bit integer

Bit 4 : FILE_ATTRIBUTE_DIRECTORY

(1 if folder, 0 if file)

LinkTargetIDList

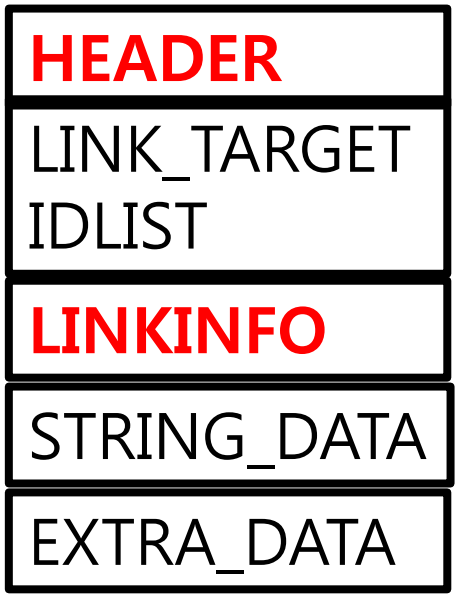
- IDListSize + IDList
- IDListSize : 2 bytes (WORD), i.e., unsigned short integer



(이는 IDList만의 size임)

(전체 크기는 IDListSIze+2)

LinkInfo



- Target path name is in this block
- 다음 2개로 구성됨

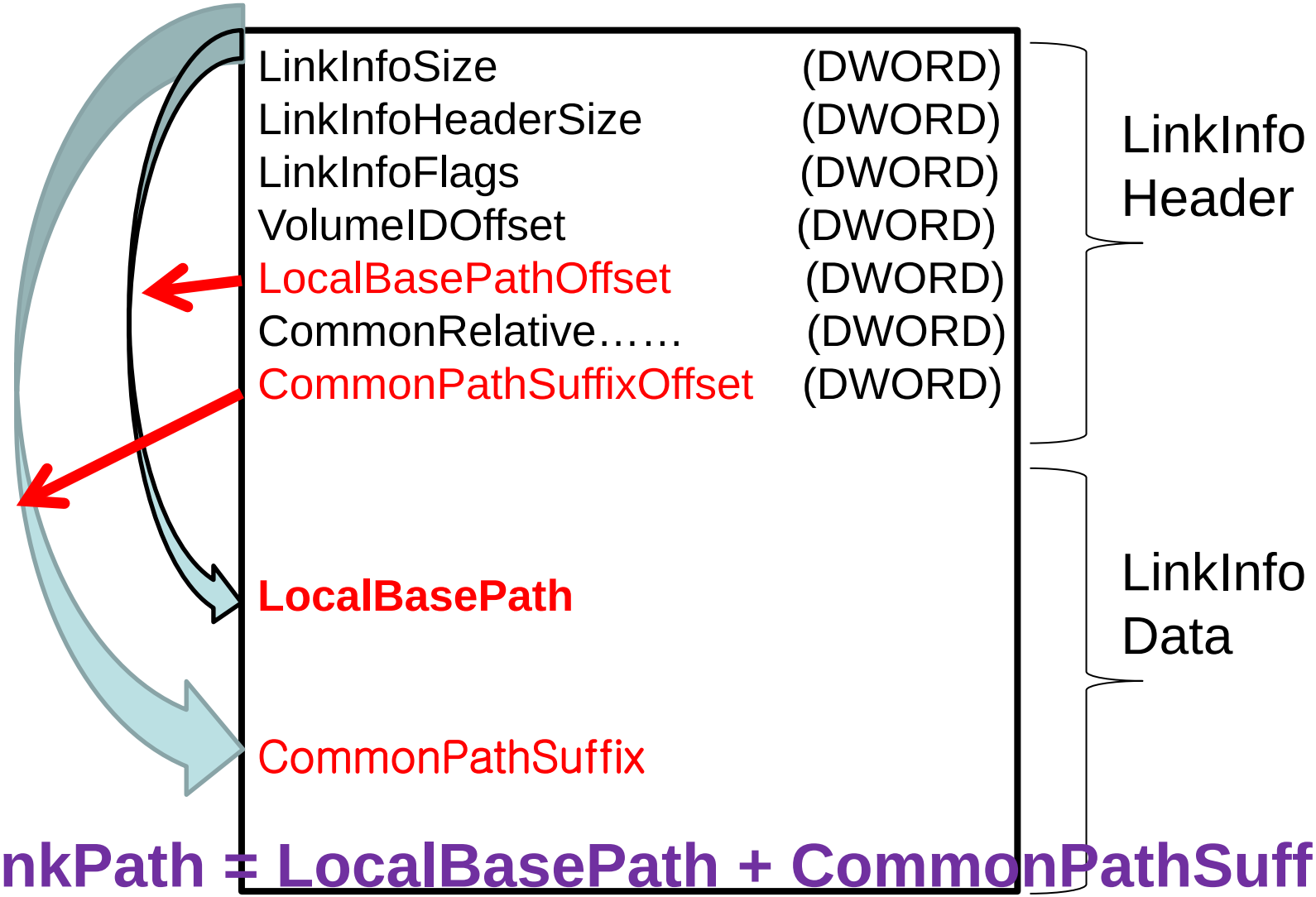
  Header & Data
- LinkInfo Header
  - LinkInfoSize (4 bytes): A 32-bit, unsigned integer that specifies the size, in bytes, of the LinkInfo structure
  - LinkInfoHeaderSize (4 bytes): A 32-bit, unsigned integer that specifies the size, in bytes, of the LinkInfo header section
  - and more (**next slide**)

LinkInfo Header

- LinkInfoSize **(4 bytes)**
- LinkInfoHeaderSize **(4 bytes)**
- LinkInfoFlags **(4 bytes)**
- VolumeIDOffset **(4 bytes)**
- **LocalBasePathOffset (4 bytes):** A 32-bit, unsigned integer that specifies the location of the LocalBasePath field. If the VolumeIDAndLocalBasePath flag is set, this value is **an offset, in bytes, from the start of the LinkInfo structure**; otherwise, this value MUST be zero.
- CommonNetworkRelativeLinkOffset **(4 bytes)**
- **CommonPathSuffixOffset (4 bytes):** A 32-bit, unsigned integer that specifies the location of the CommonPathSuffix field. This value is an **offset, in bytes, from the start of the LinkInfo structure**.

- **String의 시작점을 알려주며, 끝은 '₩0' character로 알 수 있다.**
- **Full path name은 이 두개의 string을 붙여서 얻는다.**

LinkInfo



LinkPath = LocalBasePath + CommonPathSuffix

Programming

1. 1번째 DWORD와 LinkCLSID를 읽어 들여서 이 file이 Shortcut file임을 확인한다.

   (Shortcut이 아니면, error message를 주고 종료)

1. Header에서 folder인지 file인지 확인한다
2. Header에서 LinkTargetIDList와 LinkInfo의 존재 여부를 확인한다.
3. LinkTargetIDList가 존재하면 건너뛰고, LinkInfo가 없으면, error message와 함께 종료
4. LinkInfo의 header에서 LocalBasePath와 PathSuffix의 위치를 잦아서 LinkPath를 출력한다.

int getLink (char *lnkFile, int *isFile, char *pathName, int sizePathName)

- **Returns length of string "pathName"** if read link-path successfully, 0 otherwise (failure).
- char ***lnkFile** : input : shortcut file name
- int ***isFile** : output : 1 if file, 0 if folder
- char ***pathName** : output : string buffer to hold the link path name in the shortcut file
- int **sizePathName** : size of the given string buffer 'pathName'
- You have to print proper error message(s) to console if this function returns 0