FaS
Zip file 분석 발표

정보보안암호수학과 20182209 김수빈

디지털 포렌식이란?

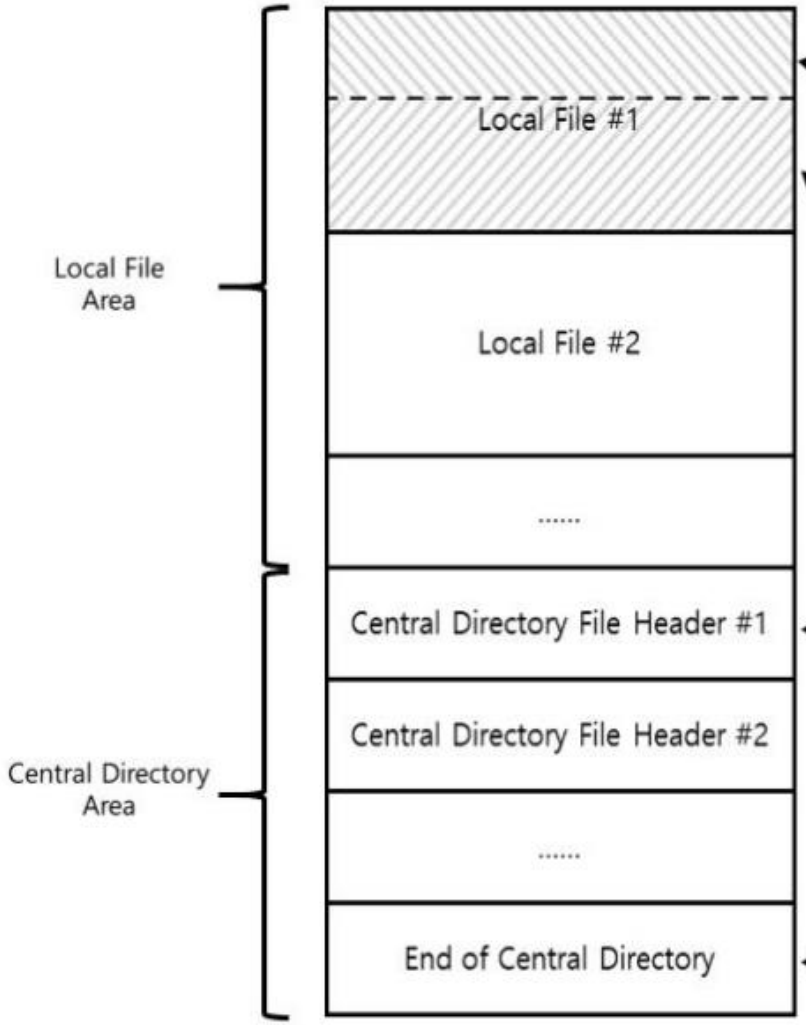
디지털 포렌식 이란?

디지털 기기를 매개로 이루어지는 범죄에 대한 법적 증거자료 확보를 위해
컴퓨터 저장매체와 네트워크로부터 자료를 수집, 분석 및 보존하여
법정 증거물로써 제출할 수 있도록 하는 일련의 절차와 행위

디지털 포렌식 종류 | 분석 대상에 따라

분석 대상	설명
디스크 포렌식	대용량의 비휘발성 저장매체(하드디스크, SSD, USB, CD 등)로부터 자료를 획득, 분석, 복구하는 분야
네트워크 포렌식	네트워크를 통하여 전송되는 데이터, 암호 등을 특정도구를 이용하여 가로채거나 서버에 로그 형태로 저장된 것을 접근하여 분석하거나 네트워크 형태 등을 조사하여 단서를 찾아내는 분야
인터넷 포렌식	인터넷으로 서비스되는 WWW, FTP등 인터넷 응용 프로토콜을 사용하는 분야
모바일 포렌식	휴대폰, PDA, 디지털카메라, 캠코더, 휴대용 메모리카드 등 휴대용 기기에서 필요한 정보를 입수하여 분석하는 분야
데이터베이스 포렌식	DB로부터 데이터를 추출/분석 하여 증거를 획득하는 분야
암호학 포렌식	문서나 시스템에서 암호를 찾아내는 분야
메모리 포렌식	메모리에 로드 되는 정보들을 분석 할 때 활용

ZIP 파일 구조



Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 03 04)				Decomp Version		General Bit		Comp Method		Mod Time		Mod Date		CRC32 -	
0x10	- CRC32				Compression Size				Decompression Size				Name Length		Extra Length	Name -
0x20	- Name (Variable)										Extra (Variable)					

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Compressed Data (Variable)															

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 01 02)				Comp Version		Decomp Version		General Bit		Comp Method		Mod Time		Mod Date	
0x10	CRC32				Compression Size				Decompression Size				Name Length		Extra Length	
0x20	Comment Length		Disk Number Start		Internal Attribute		External Attribute				Relative Local Offset				Name -	
0x30	- Name (Variable)				Extra (Variable)						Comment (Variable)					

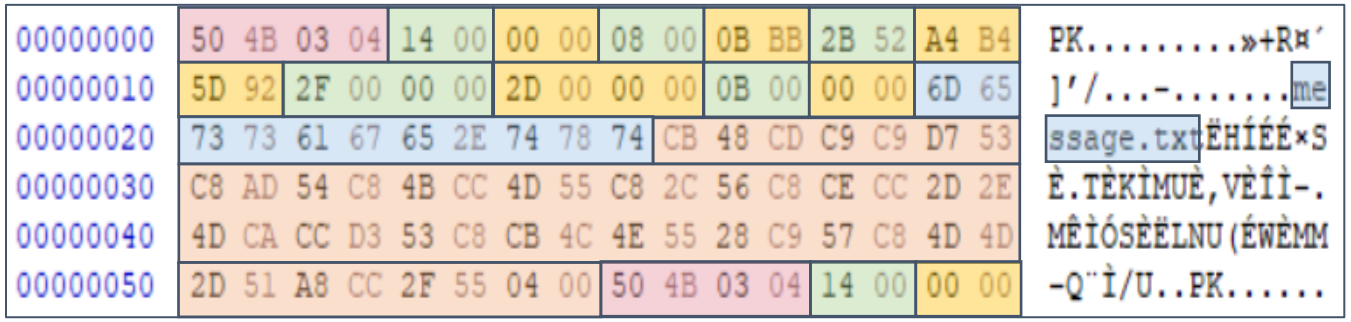
Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 05 06)				Disk number		CD Start Number		CD Count		Total CD Count		CD Size			
0x10	CD Start Offset				Comment Length		Comment (Variable)									

ZIP 파일 구조 | Local File

- 헤더영역 : 압축된 파일의 압축 정보와 같은 메타데이터를 저장
- 데이터 영역 : 압축 알고리즘으로 압축된 데이터를 저장

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 03 04)				Decomp Version		General Bit		Comp Method		Mod Time		Mod Date		CRC32 -	
0x10	- CRC32		Compression Size				Decompression Size				Name Length		Extra Length		Name -	
0x20	- Name (Variable)								Extra (Variable)							

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Compressed Data (Variable)															

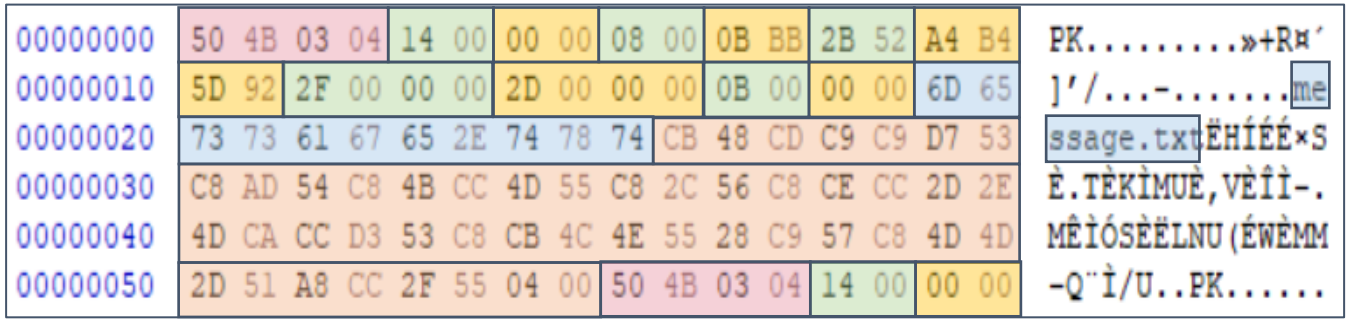


필드	설명
Signature	로컬 파일 헤더 시그니처 50 4B 03 04
Decomposition Version	압축 해제 시 필요한 버전 0x0014 = 20 → 2.0 version
General Bit	범용 비트 플래그 (암호화) Bit 00: encrypted file Bit 01: compression option Bit 02: compression option Bit 03: data descriptor Bit 04: enhanced deflation Bit 05: compressed patched data Bit 06: strong encryption Bit 07-10: unused Bit 11: language encoding Bit 12: reserved Bit 13: mask header values Bit 14-15: reserved 0x0000 = 0b0000000000000000 → 암호화 되지 않음

ZIP 파일 구조 | Local File

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 03 04)				Decomp Version		General Bit		Comp Method		Mod Time		Mod Date		CRC32 -	
0x10	- CRC32				Compression Size				Decompression Size				Name Length		Extra Length	Name -
0x20	- Name (Variable)												Extra (Variable)			

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Compressed Data (Variable)															

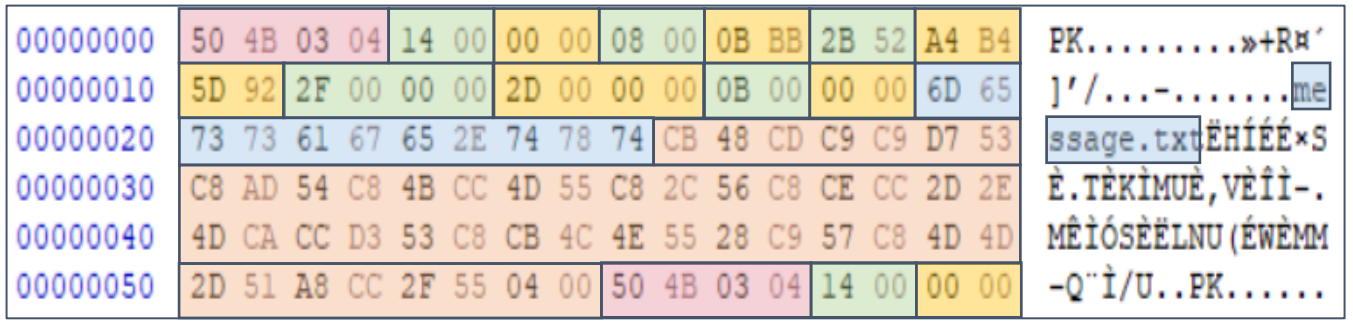


필드	설명
Compression method	압축 방법 00: no compression 01: shrunk 02-04: reduced with compression factor 1-4 06: imploded 07: reserved 08: deflated – 보통 이 방식 사용 09: enhanced deflated 10: PKWare DCL imploded 11: reserved 12: compressed using BZIP2 13: reserved 14: LZMA 15-17: reserved 18: compressed using IBM TERSE 19: IBM LZ77 z 98: PPMd version I, Rev 1 0x0008 = 8 → deflated

ZIP 파일 구조 | Local File

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 03 04)				Decomp Version		General Bit		Comp Method		Mod Time		Mod Date		CRC32 -	
0x10	- CRC32				Compression Size				Decompression Size				Name Length		Extra Length	Name -
0x20	- Name (Variable)															
	Extra (Variable)															

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Compressed Data (Variable)															

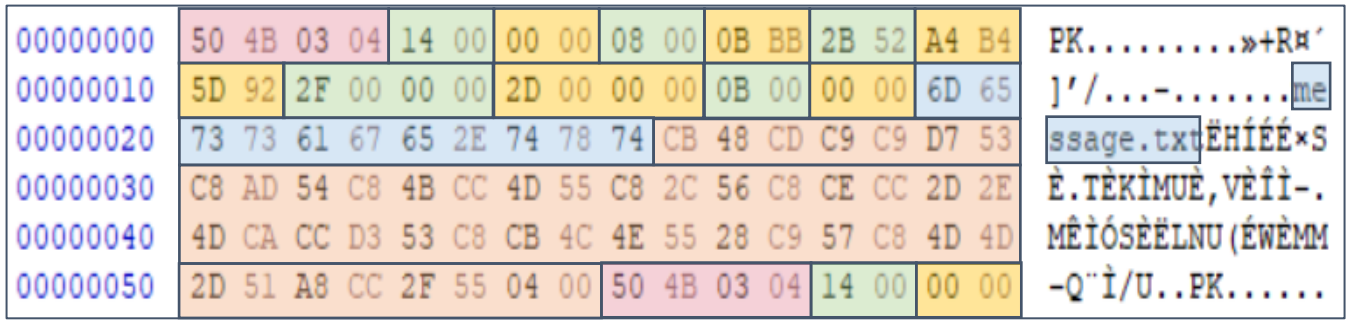


필드	설명
File modification time	마지막으로 파일 수정한 시간 Bits 00-04 : second divided by 2 Bits 05-10 : minute Bits 11-15 : hour 0xBB0B = 0b1011101100001011 second = 01011 = 11 → 22 minute = 011000 = 24 hour = 10111 = 23 23시24분22초
File modification date	마지막으로 파일 수정한 날짜 Bits 00-04 : day Bits 05-08 : month Bits 09-15 : year from 1980 0x522B = 0b0101001000101011 day = 01011 = 11 month = 0001 = 1 year = 0101000 = 41 → 1980+41 2021/01/11

ZIP 파일 구조 | Local File

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 03 04)				Decomp Version		General Bit		Comp Method		Mod Time		Mod Date		CRC32 -	
0x10	- CRC32				Compression Size				Decompression Size				Name Length		Extra Length	Name -
0x20	- Name (Variable)														Extra (Variable)	

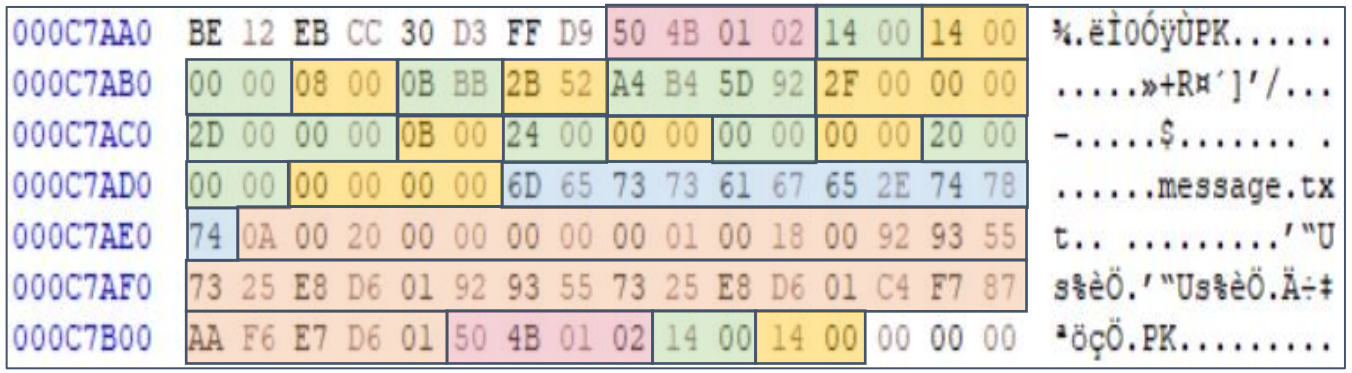
Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Compressed Data (Variable)															



필드	설명
CRC-32 checksum	파일 내용의 오류 체크 이 필드가 작성되지 않을 경우 손상된 파일로 간주하여 압축해제를 거부함. 0x925DB4A4
Compression size	압축된 데이터의 바이트 크기 0x2F = 47
Decompression size	원본 데이터의 바이트 크기 0x2D = 45
File name length	파일 이름의 길이 0x0B = 11
Extra field length	추가 필드 길이 0x00 = 0
File name	상대 경로를 포함하는 파일의 이름 “message.txt”
Extra field	추가 정보를 저장하는 데 사용됨.
Compressed Data	압축된 데이터

ZIP 파일 구조 | Central Directory Header

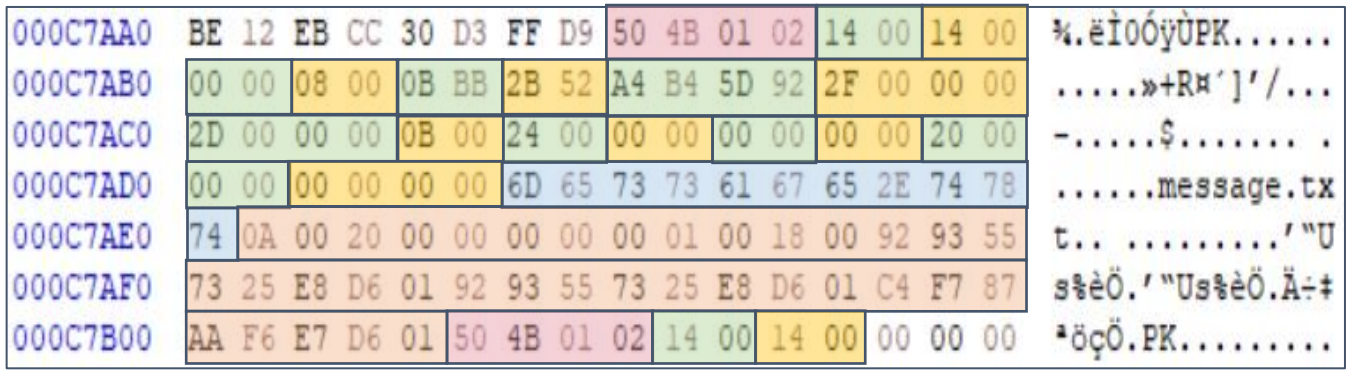
Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 01 02)				Comp Version		Decomp Version		General Bit		Comp Method		Mod Time		Mod Date	
0x10	CRC32				Compression Size				Decompression Size				Name Length		Extra Length	
0x20	Comment Length		Disk Number Start		Internal Attribute		External Attribute				Relative Local Offset				Name -	
0x30	- Name (Variable)				Extra (Variable)						Comment (Variable)					



필드	설명
Signature	로컬 파일 헤더 시그니처 50 4B 01 02
composition Version	압축 생성 버전 상위 바이트 : 0x00 → MS-DOS and OS/2 (FAT / VFAT / FAT32 file systems) 하위 바이트 : 0x14 = 20 → 2.0 version
decomposition Version ~ Name Length	Local File 필드와 동일한 값
Extra Length	Extra 길이 0x24 = 36

ZIP 파일 구조 | Central Directory Header

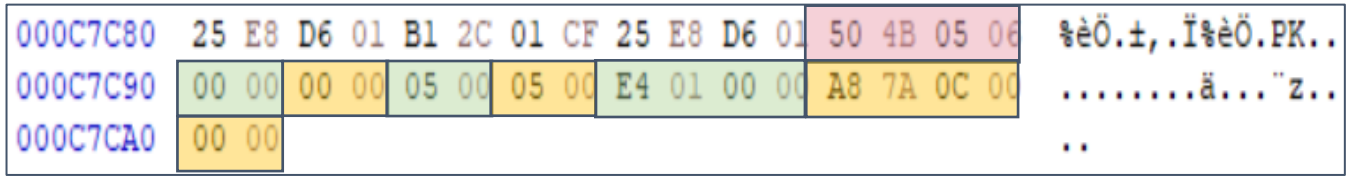
Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 01 02)				Comp Version		Decomp Version		General Bit		Comp Method		Mod Time		Mod Date	
0x10	CRC32				Compression Size				Decompression Size				Name Length		Extra Length	
0x20	Comment Length		Disk Number Start		Internal Attribute		External Attribute				Relative Local Offset				Name -	
0x30	- Name (Variable)				Extra (Variable)						Comment (Variable)					



필드	설명
comment Length	Comment 길이 0x00 = 0
Disk Number Start	디스크 수 (거의 항상 0) 0x00 = 0
Internal Attribute	내부 파일 속성 0x00 = 0
External Attribute	확장 파일 속성 0x20 = 32
Relative Local Offset	Local File Header 구조의 시작 주소 0x00 = 0
Name	상대 경로를 포함하는 파일의 이름 “message.txt”
Extra	추가 정보를 저장하는 데 사용됨.
Comment	파일 코멘트

ZIP 파일 구조 | End of Central Directory

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 05 06)				Disk number		CD Start Number		CD Count		Total CD Count		CD Size			
0x10	CD Start Offset				Comment Length		Comment (Variable)									



필드	설명
Signature	로컬 파일 헤더 시그니처 50 4B 05 06
Disk Number	디스크 수
CD Start Number	Central Directory 시작되는 디스크번호
CD Count	Central Directory에 있는 항목의 총 수 0x5 = 5
Total CD Count	모든 항목의 총 수 0x5 = 5
CD Size	Central Directory의 바이트 크기 0x01E4 = 484 bytes
CD Start Offset	Central Directory 시작되는 오프셋 주소 Offset = 000C7AA8
Comment Length	코멘트 필드의 길이 0x00 = 0
Comment	파일 코멘트

Zip 파일 파싱 프로그램 생성 | 파일명, 파일 데이터 offset, 데이터 영역 구하기

파일명, 파일 데이터 offset, 데이터 영역 구하기

로컬 파일 영역에
파일명, 파일 데이터 offset, 데이터 영역의 정보가 모두 있기 때문에,
로컬 파일 영역만 살펴보면 된다.

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 03 04)				Decomp Version		General Bit		Comp Method		Mod Time		Mod Date		CRC32 -	
0x10	- CRC32		Compression Size				Decompression Size				Name Length		Extra Length		Name -	
0x20	- Name (Variable)										Extra (Variable)					

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Compressed Data (Variable)															

1. Local File Signature Offset 찾기
2. File name과 Data Offset 찾아서 출력
3. 데이터 영역 파일로 저장하기

Zip 파일 파싱 프로그램 생성 | Local File Signature offset 찾기

```python
f = open('testzipfile.zip', 'rb+')

# local file signature offset 찾기
LF_sig = b'\x50\x4B\x03\x04'
CF_sig = b'\x50\x4B\x01\x02'
LF_sig_offset = []
offset = 0

while True:
    f.seek(offset)
    fr =  f.read(4)

    if fr == LF_sig:
        LF_sig_offset.append(offset)
    elif fr == CF_sig:
        CF_sig_start_offset = offset
        break
    offset += 1
```

Local File Signature offset 찾기

1. Hex값을 4개씩(4byte) 읽으면서 Local file signature 값과 같으면 그 위치(offset)를 리스트에 추가
2. Central File signature 값과 같으면 Local file을 모두 읽은 것이기 때문에 반복문을 빠져나온다.

Zip 파일 파싱 프로그램 생성 | File Name과 Data Offset 찾기

```python
# 파일 이름과 data offset 찾기
name = []
data_offset = []
for i in range(len(LF_sig_offset)):
    # Name Length
    nameLen_offset = LF_sig_offset[i] + 26
    f.seek(nameLen_offset)
    nameLen_hex = f.read(2)
    nameLen = little2(nameLen_hex)

    # Extra Length
    extraLen_offset = nameLen_offset + 2
    f.seek(extraLen_offset)
    extraLen_hex = f.read(2)
    extraLen = little2(extraLen_hex)

    # Name
    name_offset = extraLen_offset + 2
    f.seek(name_offset)
    name_hex = f.read(nameLen)
    name.append(name_hex.decode())

    # data
    dataOffset = name_offset + nameLen + extraLen
    data_offset.append(dataOffset)

print('file name :', name)
print('data offset :', data_offset)
```

```python
# little endian, 10진수로 변환
def little2(hex):
    return struct.unpack('<H', hex)[0] # 2byte
```

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 03 04)				Decomp Version		General Bit		Comp Method		Mod Time		Mod Date		CRC32 -	
0x10	- CRC32		Compression Size				Decompression Size				Name Length		Extra Length		Name -	
0x20	- Name (Variable)										Extra (Variable)					

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Compressed Data (Variable)															

File Name과 Data Offset 찾기

1. Name 길이, Extra 길이 구하기
- Name Length offset = Local File signature offset + 26
- Extra Length offset = Name Length offset + 2
- 2byte Hex 값을 Little endian으로 변환 후 10진수로 변환

Zip 파일 파싱 프로그램 생성 | File Name과 Data Offset 찾기

```python
# 파일 이름과 data offset 찾기
name = []
data_offset = []
for i in range(len(LF_sig_offset)):
    # Name Length
    nameLen_offset = LF_sig_offset[i] + 26
    f.seek(nameLen_offset)
    nameLen_b = f.read(2)
    nameLen = little2(nameLen_b)

    # Extra Length
    extraLen_offset = nameLen_offset + 2
    f.seek(extraLen_offset)
    extraLen_b = f.read(2)
    extraLen = little2(extraLen_b)

    # Name
    name_offset = extraLen_offset + 2
    f.seek(name_offset)
    name_b = f.read(nameLen)
    name.append(name_b.decode())

    # data
    dataOffset = name_offset + nameLen + extraLen
    data_offset.append(dataOffset)

print('file name :', name)
print('data offset :', data_offset)
```

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Signature (50 4B 03 04)				Decomp Version		General Bit		Comp Method		Mod Time		Mod Date		CRC32 -	
0x10	- CRC32		Compression Size				Decompression Size				Name Length		Extra Length		Name -	
0x20	- Name (Variable)										Extra (Variable)					

Offset(h)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Compressed Data (Variable)															

File Name과 Data Offset 찾기

2. File name 찾기
- Name offset = Extra Length offset + 2
- Name Length만큼 읽어 문자열을 리스트에 저장

3. Data offset 찾기
- Data offset = Name offset + Name Length + Extra Length

```
file name : ['message.txt', 'photo/', 'photo/KAKAO.jpeg', 'photo/kakao_apeach.png', 'photo/toystory.jpeg']
data offset : [41, 124, 170, 27147, 799121]
```

Zip 파일 파싱 프로그램 생성 | Data 영역 파일로 저장하기

```python
# data 길이 구하기
dataLen = []
for i in range(len(data_offset)):
    if i == len(data_offset)-1:
        dataLen.append(CF_sig_start_offset - data_offset[i])
    else:
        dataLen.append(LF_sig_offset[i+1] - data_offset[i])

# data 영역 파일로 저장하기
for i in range(len(data_offset)):
    f.seek(data_offset[i])
    data = f.read(dataLen[i])
    if len(data) == 0:
        os.makedirs(name[i], exist_ok=True)
    else:
        data = zlib.decompress(data, -zlib.MAX_WBITS) # deflate 압축 풀기
        fw = open(name[i], 'wb+')
        fw.write(data)
        fw.close()
```

Data 영역 파일로 저장하기

1. Data 영역 길이 구하기
- 다음 Local File Signature offset 에서 현재 data offset을 빼면 data 길이를 구할 수 있다.
- 맨 마지막 data 길이는 Central File Signature offset에서 data offset을 빼야 한다.

2. Data 영역 파일로 저장하기
- 구한 Data Length만큼 data 영역을 읽는다.
- Data Length가 0이면 폴더를 뜻한다. 따라서 os.makedirs 함수를 이용해서 폴더를 만들어 저장한다.
- Data Length가 0이 아니면 파일이다.
- 이때 data는 deflated로 압축되어 있는 데이터이기 때문에 zlib 라이브러리를 이용해 압축을 풀어준다.
- Write를 통해 data를 파일로 저장한다.

Zip 파일 파싱 프로그램 생성 | Data 영역 파일로 저장하기