실습

Handling Text Files

FILE

- File의 성질
  - they have a name
  - must be opened and closed (system resources are limited)
  - can be written to, or read from, or appended to
  - after a file has been opened, the data stream can be accessed with file-handling functions in the standard library

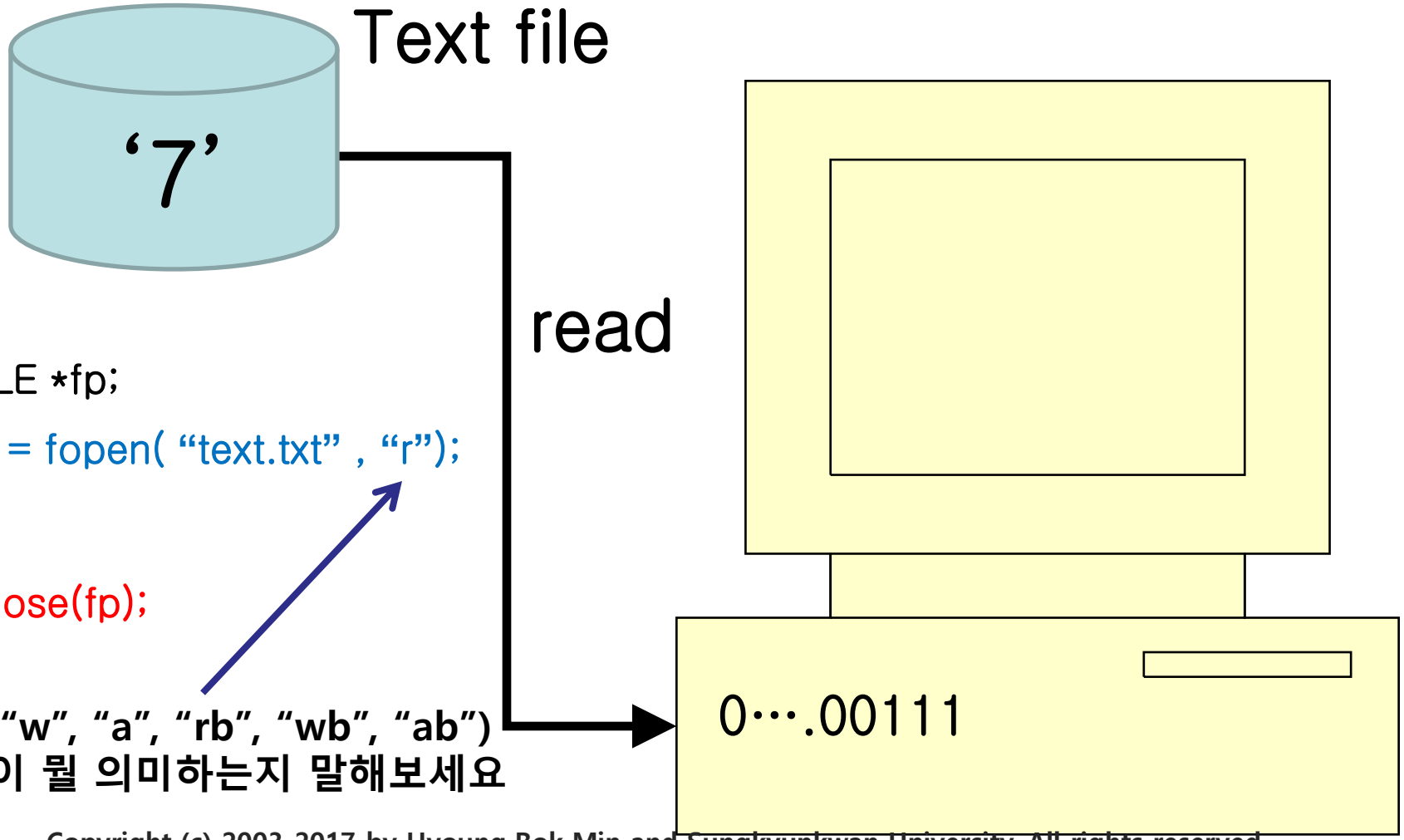
fopen( )

fprintf( )
fscanf( )

fclose( )

파일 열기(file open), 파일의 닫기(file close)

```
FILE *fopen( const char *filename, const char *mode );
```



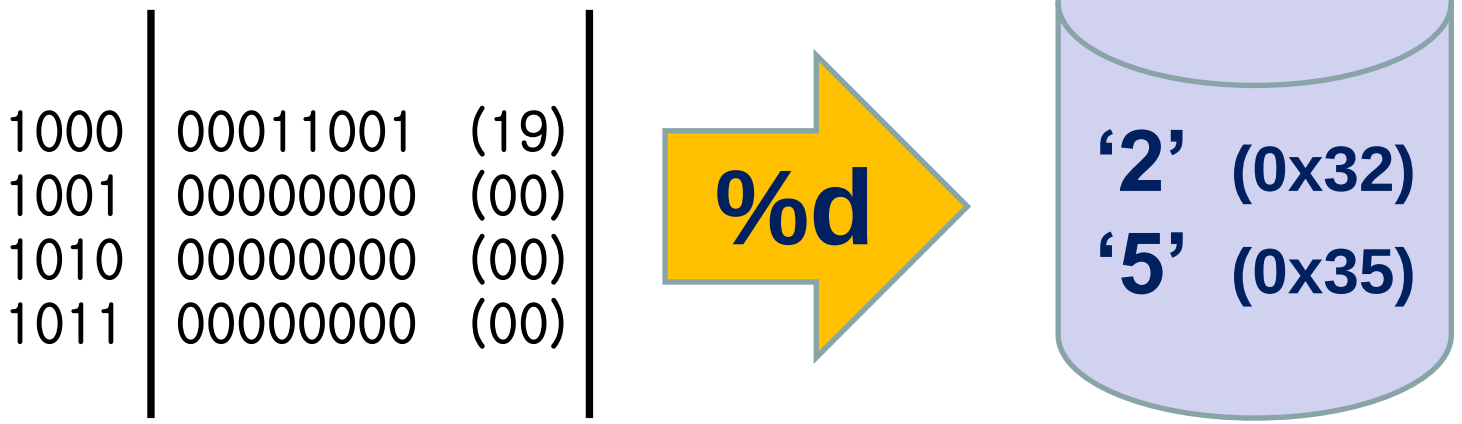
Open Mode of a FILE

Mode Strings for Text/Binary Files	
String	Meaning
"r"	open text file for reading
"w"	open text file for writing
"a"	open text file for appending
"rb"	open binary file for reading
"wb"	open binary file for writing
"ab"	open binary file for appending
"r+"	open text file for reading and writing
"rb+"	open binary file for reading and writing

Text File

int number = 25;
fprintf(fp, "%d", number);

25 = 000…..00000011001 (binary)
= 00000019 (hex)



convert binary numbers to characters

ASCII Code

- ASCII
- CP949 (euc-kr)
- Unicode (UTF-8)

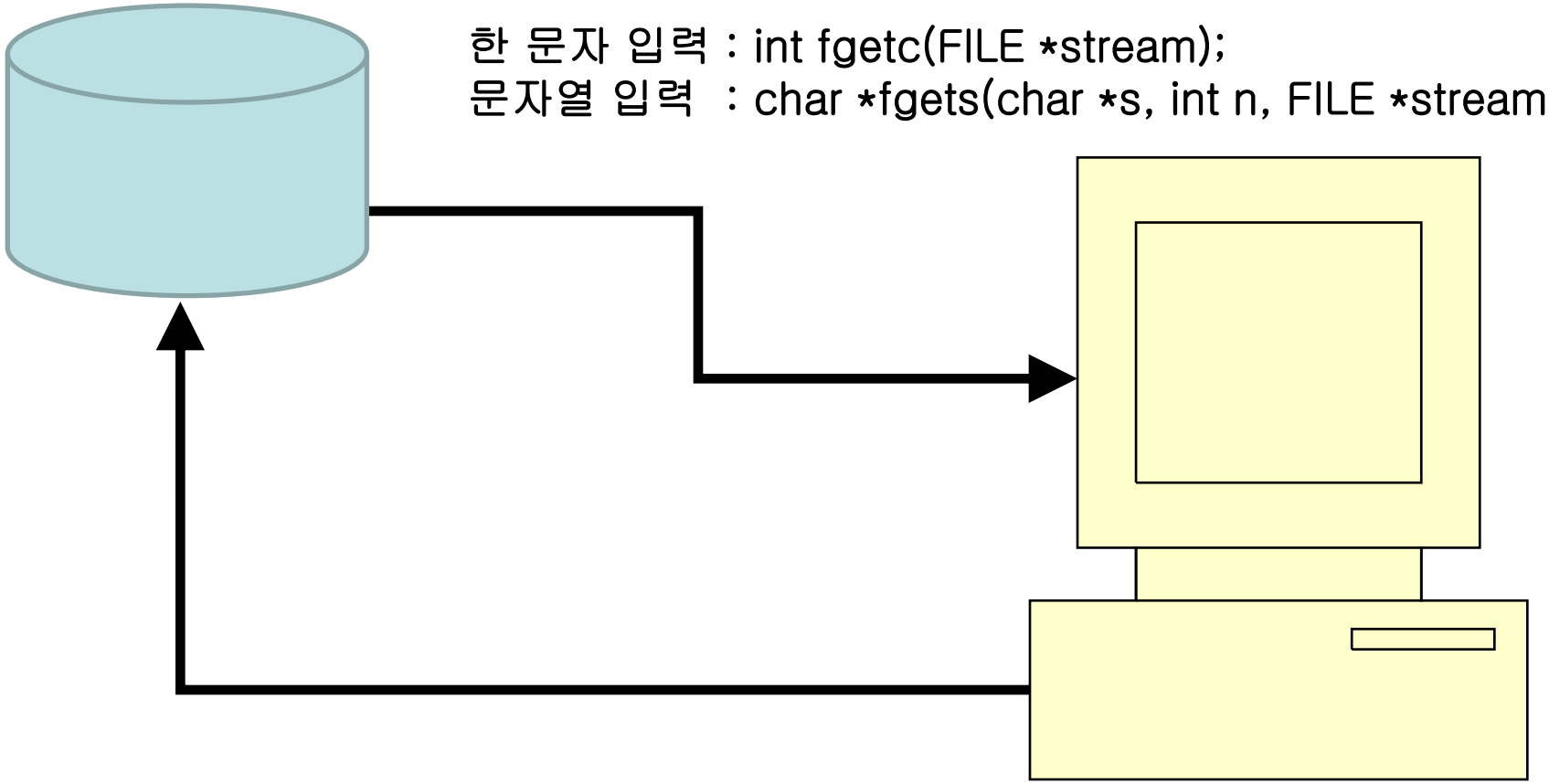
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Text File

- 모든 data(char, int, short, double…)는 computer 내에서 binary number로 표현된다
- Text file에서는 **모든 data가 character로 변환되어 저장**된다
- (예) integer 17
  - Computer내의 Binary representation :
    0…010001 (0x00000011)
  - Text file : character (ASCII code) '1'과 '7'로 저장
    ('1' = 0x31, '7' = 0x37)
  - 오직 character만이 text file에도 그대로 저장됨

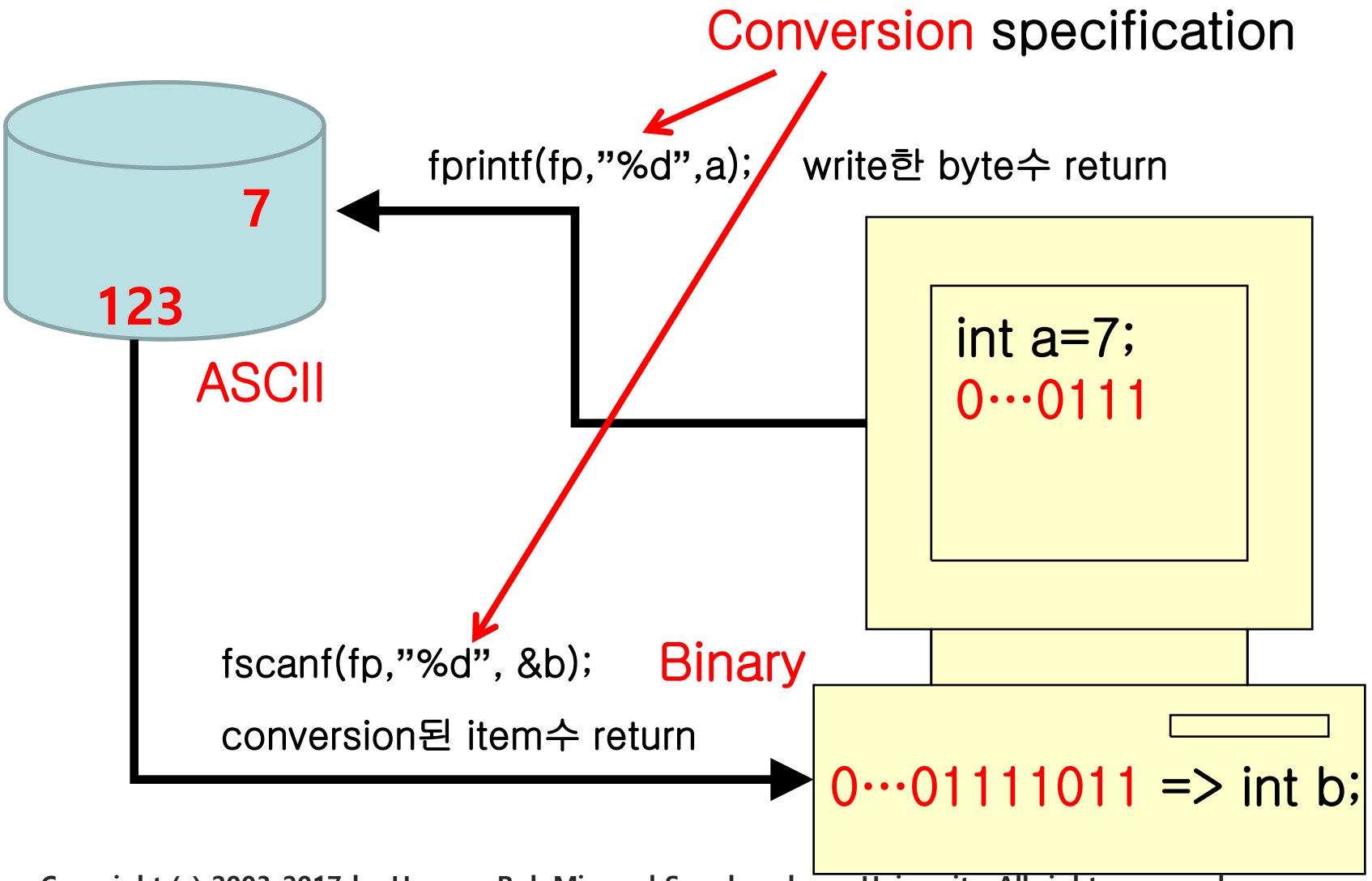
파일 문자 1개 , 문자열 입출력 (text file)

한 문자 입력 : int fgetc(FILE *stream);
문자열 입력 : char *fgets(char *s, int n, FILE *stream);



한 문자 출력 : int fputc(int c, FILE *stream);
문자열 출력 : int fputs(const char *s, FILE *stream);

파일 서식화 입출력 (text file)



Line 별 (1줄씩) 입출력 (text file)

1줄 입력 : **char *fgets(char *s, int n, FILE *fp);**

- s: file에서 1줄을 읽어 여기 저장 (마지막에 NULL을 추가)
- n : 입력 버퍼 s의 크기 (byte)
- fp : pointer to file
- s를 return (읽을 것이 없으면, NULL을 return)

1줄 출력 : **int fputs(char *s, FILE *fp);**

- s : 여기 저장된 1줄 (NULL 직전까지)을 fp에 쓴다
- fp : pointer to file
- 실제로 write된 byte 개수를 return

Line 별 (1줄씩) 입력 예 (text file)

```
char readbuf[1024];  // line buffer
while(fgets(readbuf, 1024, fp)) {
    // do anything you want
}
```

- 읽어들일 장소를 마련해야 한다 (array / malloc)
- 안전한 읽기 방법 : 1줄이 1023 문자를 넘으면 1023 문자만 읽는다
- File을 모두 읽으면 while을 빠져나옴
- 1줄에 대하여 sscanf(…)를 이용하여 data분리 가능

읽어들인 1줄의 분리

int sscanf(const char *s, char *format, …);

(example)

```
int age, english, math;
char gender, name[128];
char *buf = “Min 23 M 12 34”;
int num = sscanf(buf, “%s%d %c %d%d”,
                      name, &age, &gender,
                      &english, &math);
```

Text File - Issue

- Line breaks
  - Special character to denote end-of-line
  - In C, line-break character is '\n'

    (Example) printf("Hello, world!\n");
- In text file
  - Use different line breaks depending on OS
  - **Use '\r' and '\n' for Windows**
  - **Use '\n' for UNIX (e.g., macOS) / LINUX**
  - Use '\r' for old MacOS

Write/Read on Windows

```
printf("Hello\n");
printf("Min\n");
printf("Wow\n");
```

Text Editor

```
Hello
Min
Wow
```

In File,

```
Hello\r\nMin\r\nWow\r\n
```

Read a line from file,

```
Hello\n
```

Write/Read on Non-Windows

```
printf("Hello\n");
printf("Min\n");
printf("Wow\n");
```

Text Editor

```
Hello
Min
Wow
```

In File,

```
Hello\nMin\nWow\n
```

Read a line from file,

```
Hello\n
```

Write on Windows
Read on Non-Windows

```
printf("Hello\n");
printf("Min\n");
printf("Wow\n");
```

Text Editor

```
Hello\r
Min\r
Wow\r
```

In File,

```
Hello\r\nMin\r\nWow\r\n
```

Read a line from file (gcc),

```
Hello\r\n
```

Write on Non-Windows
Read on Windows

```
printf("Hello\n");
printf("Min\n");
printf("Wow\n");
```

NotePad

메모장

In File,

```
Hello↵Min↵Wow↵
```

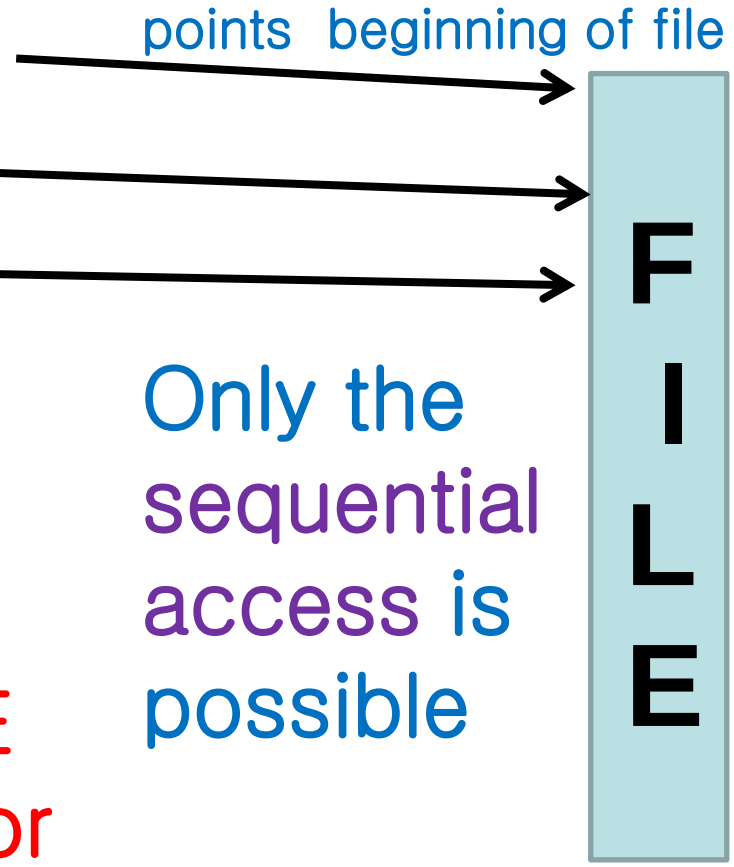
```
Hello\nMin\nWow\n
```

Read a line from file (gcc),

```
Hello\n
```

FILE pointer advances

```
FILE *fp = fopen(…);
fgets(…, fp);
fscanf(fp, …);
  // fprintf(fp, …);
fclose(fp);
```



Only the sequential access is possible

Can we move FILE pointer forward or backward ?

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

But, be careful for Windows text files

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

실습 (1)

- **Text file이 1개 주어진다. 이 파일에 여러명 학생의 이름, 나이, 성별 (M or F), 여러 과목의 성적 점수가 들어있다.**
- **이 파일의 첫 줄에는 학생의 숫자와 과목의 숫자가 있으며, 2번째 줄 이후에는 학생 정보 (이름, 아니, 성별, 여러 과복 점수)가 1줄에 1명씩 들어 있다.**
- **// 로 시작하는 모든 줄은 comment이며 건너뛴다.**
- **정보는 누락된 것이 있을 수 있다. 누락된 경우에는 적절한 error message를 준 후, 이 줄은 무시한다. 즉, 다음 page에 주어진 연산에 포함시키지 않는다.**

실습 (2)

- 이 file의 크기를 측정하여 Console 화면에 출력한다.
- 이 text file을 읽어 들여서, 모든 학생의 N과목 점수의 평균점을 double floating point number로 계산하여 소수점 2자리까지 console에 출력한다.
- M명을 성별로 숫자를 console에 출력한다.
- 이제, M명의 나이의 평균을 소수점 1자리까지 console 에 출력한다.
- Console에 출력한 것과 똑같은 것을 text file "outfile.txt"에 동시에 출력한다.

Program Spec

- 입력, 출력 file의 이름을 변경할 수 없음
- 출력은 web page의 format을 따를 것
- File size의 측정
  - fseek ()
  - ftell ()
- File 읽기
  - Read a line - use fgets ()
  - Extract data – use sscanf ()

Refer to scanf.c

- Provided at class web
- Show how to use a few format strings at scanf(), fscanf(), sscanf().