System Programming

2. File IO (1): Standard I/O Library - 3

Seung-Ho Lim

Dept. of Computer & Electronic Systems Eng.

File Offset

- Every open file has a (r/w) offset which indicates the next access position in the file
  - when a file is opened for reading/writing, the offset is set to the beginning of the file
  - when a file is opened for appending, the offset is set to the end of the file
  - While reading/writing, the offset automatically advances

File Access Methods

- Sequential access
  - sequential access by following the r/w offset
- Random access
  - moves the r/w offset to a wanted access position by calling **fseek()** library function
    - or by **lseek()** system call,
  - mainly used for record processing.

cf. Keyed access

- Access a record of a DB by a key,
- A internal index tree of a DB is necessary.

R/W offset related functions

Function Prototypes	***Input Arg.***	***Return***	
		normal	*error*
*int **fseek** (FILE *stream, long offset, int sopt)*	- *stream*: file pointer **- *offset***: distance relative to SEEK option position - *sopt*: SEEK option	0	-1
*void **rewind** (FILE *stream)*	stream: file pointer	none	none
*long **ftell** (FILE *stream)*	stream: file pointer	current offset	-1

- SEEK options
  - SEEK_SET: new r/w offset = offset
  - SEEK_CUR: new r/w offset = current_offset + offset
  - SEEK_END: new r/w offset = EOF + offset

Random access example (1)

frandom-ex.c

```
#include <stdio.h>
int main(int argc, char *argv[])
{
		FILE *fp;
		char buf[256];
		int rspn;
		long pos;

		if((fp = fopen(argv[1], "r")) == NULL) {
				perror(argv[1]);
				return 1;
		}
		rspn = fseek(fp, 8L, SEEK_SET);
		pos = ftell(fp);
```

Random access example (2)

```
        fgets(buf, 256, fp);
        printf("Position : %ld\n", pos);
        printf("%s\n", buf);

        rewind(fp);
        pos = ftell(fp);

        fgets(buf, 256, fp);
        fclose(fp);
        printf("Position : %ld\n", pos);
        printf("%s\n", buf);
        return 0;
}
```

Random access example (3)

- Execution

```
$ cat test.dat
This is a test data.

$ ./a.out test.dat
Position : 8
a test data.
Postion : 0
This is a test data.
$
```

I/O Types

- Unformatted I/O (Binary I/O)
  - I/O in binary format (memory representation).

    integer : 4 byte, signed two's complement.

    float: 4 bytes, "sign + exp(8-bit) + mantissa(23-bit)".

    double: 8 bytes, "sign + exp(11-bit) + mantissa(52-bit)".
  - a user's viewer program must be supported.
- Formatted I/O
  - output: integer, float, double → output in an ASCII string
  - input: ASCII string input → integer, float, double (scan conversion)
  - e.g.

    %5d: integer to decimal ASCII string (5 digits)

    %f: 12.43
  - file contents can be seen by "cat file".

Formatted Output

Function Prototypes	*Description*	*Return*	
		normal	*error*
*int **printf** (const char *format, /* args */ ...)*	to the console	output length	negative integer
*int **fprintf** (FILE *stream, const char *format, /* args */ ...)*	to a file		
*int **sprintf** (char *s, const char *format, /* args */ ...)*	to a string		

Formatted Input

Function Prototypes	***Description***	***Return***	
		normal	*error*
*int **scanf** (const char *format, ...)*	from the console	input length	EOF
*int **fscanf** (FILE *stream, const char *format, ...)*	from a file		
*int **sscanf** (char *s, const char *format, ...)*	from a string		

Formatted I/O example (1)

stdio-ex.c

```
#include <stdio.h>

int main(int argc, char argv[])
{
        FILE *fp;
        char buf[256];
        int num, Nnum;
        char str[30], Nstr[30];

        scanf("%d %s", &num, str);
        if((fp = fopen("test.dat", "w")) == NULL) {
                perror(test.dat);
                return 1;
        }
```

Formatted I/O example (2)

```
            fprintf(fp, "%d %s\n", num, str);

            if((fp = freopen("test.dat", "r", fp)) == NULL) {
                        perror("test.dat");
                        return 1;
            }
            fscanf(fp, "%d %s\n", &Nnum, Nstr);
            printf("%d %s\n", Nnum, Nstr);

            fclose(fp);
            return 0;
}
```

File error check

Function Prototypes	*Return*	
	error value	*when no error*
*int **ferror** (FILE *stream)*	nonzero value	0
*int **feof** (FILE *stream)*	nonzero value	0
*void **clearerr** (FILE *stream)*	none	none

File error check example (1)

ferror-ex.c

```
#include <stdio.h>

int main(void)
{           int ret;
            FILE *fp;

            fp = fopen("test.dat", "r");
            putc('?', fp);
            if(ret = ferror(fp))
                        printf("ferror() return %d\n", ret);
            clearerr(fp);
            printf("ferror() return %d\n", ferror(fp));
            fclose(fp);
            return 0;
}
```

File error check example (2)

- Execution

```
$ cat test.dat
1234 abcd
$ ./a.out
ferror() returned 1
ferror() returned 0
$
```

EOF check example (1)

feof-ex.c

```
#include <stdio.h>
int main()
{
        int stat = 0;
        FILE *fp;
        char buf[256];

        fp = fopen("test.dat", "r");
        while(!stat)
                if(fgets(buf, 256, fp))
                        printf("%s\n", buf)
                else
                        stat = feof(fp);
        printf("feof returned %d\n", stat);
        fclose(fp);
        return 0;
}
```

EOF check example (2)

- Execution

```
$ cat test.dat
1234 abcd

$ ./a.out
1234 abcd
feof returned 1
$
```

Error handling

- Important ANSI C Features:
  - function prototypes
  - generic pointers (`void *`)
  - abstract data types (e.g. `pid_t, size_t`)
- Error Handling:
  - meaningful return values
  - *`errno`* variable
    - must include <errno.h>
  - look up constant error values via two functions:

```
#include <string.h>
char *strerror(int errnum) // returns pointer to message string

 #include <stdio.h>
void perror(const char *msg) // print the last error with the msg
```

Homework

- Write your own short text file using *vim* editor or others
  - File name : test.dat
  - At least 5 lines, each lines contains 10 over characters
- Write and run following programs(in the lecture note)
  - *fileio-ex.c*
  - *filecopy.c*
  - *frandom-ex.c*
  - *feof-ex.c*
- Submission
  - Make directory in your home directory with name “HW1”
  - in the HW1 directory, submit above program files
  - Due date : 4/2