유닉스 프로그래밍 및 실습

3장. 버퍼 입출력

1. 사용자 버퍼 입출력

- 사용자 버퍼 입출력
  - 사용자 영역에서 수행하는 버퍼링
  - 응용 자체에서 직접 수행 또는 라이브러리가 투명하게 수행
- dd를 이용한 실험
- 블록 크기
  - 512, 1024, 2048, 4096
  - 블록크기의 정수배 또는 약수 단위로 수행하는 경우 성능 향상
  - 블록 크기 알아내기 – stat(7장)
  - 사용자 버퍼 입출력
    - 응용프로그램 내부에서 직접 구현
    - 표준입출력 라이브러리 활용

2. 표준 입출력

- 표준 C 라이브러리
  - 1989년 ANSI C 표준
    - 문자열 처리, 수학 루틴, 시간과 날짜, 입출력
- 파일 포인터
  - <stdio.h>
  - FILE
  - 읽기, 쓰기, 읽기/쓰기 모드

3. 파일 열기

- fopen()
- 모드

4. 파일 기술자로 스트림 열기

- fdopen()

5. 스트림 닫기

- fclose()
- fcloseall()

6. 스트림에서 읽기

- 한번에 하나씩 문자 읽기
  - fgetc()
- 문자 하나 되돌리기
  - ungetc()
- 전체 행 읽기
  - fgets()
- 행 단위가 아닌 임의 문자열 읽기
  - fgetc()를 이용하는 방법
- 이진 자료 읽기
  - fread()
  - ferror(), feof() (3.11)

7. 스트림에 쓰기

- 문자 하나 쓰기
  - fputc()
- 문자열 쓰기
  - fputs()
- 이진 자료 쓰기
  - fwrite()

8. 버퍼 입출력을 활용하는 예제 프로그램

- 예제
- 아키텍처와 ABI가 동일한 경우에만 이진 자료 쓰기/읽기 동작이 일관성을 보임

과제

1. 8절 예제 프로그램 수행
2. 신상 정보(이름, 학번, 전화번호 등)를 사용자로부터 입력 받아 구조체에 저장하고, 이것을 텍스트 파일로 저장(덧붙이기)
3. 텍스트 파일에 저장된 신상 정보를 구조체로 읽어들여 화면에 반복 출력

- 프로그램 코드, 설명, 수행 결과 캡처 화면 등을 하나의 파일로 만들어 cms.mmu.ac.kr/bear 과제 제출에 업로드
- 제출기한 : 9월 23일 자정

참고 – st2txt.c (1)

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4
 5 #define NAME_LENGTH 20
 6 #define P_NO_LENGTH 20
 7 #define PHONE_LENGTH 20
 8 #define LINE_LENGTH 80
 9 #define LONG_LINE_LENGTH 256
10
11 typedef struct {
12         char name[NAME_LENGTH];
13         char p_no[P_NO_LENGTH];
14         char phone[PHONE_LENGTH];
15 } guest_info_t;
16
17
```

참고 – st2txt.c (2)

```
18 /*************************************************************************/
19 int get_info_by_scanf(guest_info_t *info)
20 {
21         printf("Guest name ? : ");
22         if (scanf("%s", info->name) != 1)
23                 return(0);
24
25         printf("Persoanl number ? : ");
26         if (scanf("%s", info->p_no) != 1)
27                 return(0);
28
29         printf("Phone number ? : ");
30         if (scanf("%s", info->phone) != 1)
31                 return(0);
32
33         return (1);
34 }
35
```

참고 – st2txt.c (3)

```
36 /*************************************************************************/
37 int get_info_by_fgets(guest_info_t *info)
38 {
39         printf("Guest name ? : ");
40         if (fgets(info->name, NAME_LENGTH, stdin) == NULL)
41                 return (0);
42
43         printf("Persoanl number ? : ");
44         if (fgets(info->p_no, P_NO_LENGTH, stdin) == NULL)
45                 return (0);
46
47         printf("Phone number ? : ");
48         if (fgets(info->phone, PHONE_LENGTH, stdin) == NULL)
49                 return (0);
50
51         info->name[strlen(info->name)] = '\0';
52         info->p_no[strlen(info->p_no)] = '\0';
53         info->phone[strlen(info->phone)] = '\0';
54
55         return (1);
56 }
57
```

참고 – st2txt.c (4)

```
58 /*************************************************************************/
59 int main()
60 {
61         FILE *astream;
62         char *filename = "datafile";
63         int going = 1;
64         guest_info_t    record;
65         char answer[LINE_LENGTH];
66         char buffer[LONG_LINE_LENGTH];
67
68         astream = fopen(filename, "a+");
69
70         if (astream) {
71                 while (going) {
72                         if (get_info_by_fgets(&record)) {
73                                 printf("save or not (y|n) ? : ");
74                                 scanf("%s", answer);
75                                 if (answer[0] == 'y' || answer[0] == 'Y') {
76                                         sprintf(buffer, "%s %s %s\n",
77                                                         record.name, record.p_no, record.phone);
78                                         fputs(buffer, astream);
79                                 }
80                         }
81
82                         printf("quit or not (y|n) ? : ");
83                         scanf("%s", answer);
84                         if (answer[0] == 'y' || answer[0] == 'Y') {
85                                 going = 0;
86                         }
87                 }
88                 fclose(astream);
89         }
90         else {
91                 perror("fopen");
92                 exit(1);
93         }
94 }
```

참고 – txt2st.c

```
17 /*************************************************************************/
18 int main()
19 {
20         FILE *stream;
21         char *filename = "datafile";
22         guest_info_t    record;
23         char buffer[LONG_LINE_LENGTH];
24
25         stream = fopen(filename, "r");
26
27         if (stream) {
28                 while (fgets(buffer, LONG_LINE_LENGTH, stream)) {
29                         sscanf(buffer, "%s %s %s", record.name, record.p_no, record.phone);
30                         printf("Guest name : %s\tPersonal No: %s\tPhone No: %s\n",
31                                 record.name, record.p_no, record.phone);
32                 }
33                 fclose(stream);
34         }
35         else {
36                 perror("fopen");
37                 exit(1);
38         }
39 }
```

9. 스트림 탐색하기

- fseek()
- fsetpos()
- rewind()
- 현재 스트림 위치 얻기
  - ftell()
  - fgetpos()

10. 스트림 강제 출력(버퍼 비우기)

- fflush()
- fsync() 참조

11. 오류와 EOF

- ferror()
- feof()
- clearerr()

12. 관련된 파일 기술자 얻기

- fileno()

13. 버퍼 제어하기

- 옵션
  - 버퍼 미사용
  - 행 버퍼
    - 터미널의 기본 버퍼 정책
  - 블록 버퍼
    - 파일과 관련된 스트림의 기본 정책
    - Full buffer라고도 지칭
- setevbuf()

14. 스레드 안전

- 스레드 사용시
  - 자료 접근 동기화에 주의를 기울여야 함
  - 상호배제를 보장하는 잠금 메커니즘 필요
- 표준입출력 함수는 본질적으로 안전하게 스레드 지원
  - 단일 함수 호출은 원자적으로 수행됨
- 넓은 원자성이 요구되는 경우 조치 필요
- 수동으로 파일 잠그기
  - flockfile()
  - funlockfile()
  - ftrylockfile()
- 잠금 기능이 없는 스트림 연산 (리눅스 전용)
  - 잠금 부하를 최소화해서 성능 향상
  - fgetc_unlocked
  - fgets_unlocked
  - …
  - 스트림에 관련된 잠금을 점검하지도 않고 획득하지도 않음

15. 표준 입출력 비판

- 표준입출력에 존재하는 결함
  - gets() 축출
- 가장 큰 불만
  - 이중 복사 과정에서 나타나는 성능 문제

16. 결론

과제

- 호텔 고객 관리 프로그램
  - 호텔 각 층은 총 20개의 객실이 있고, 객실 번호는 517과 같이 층번호와 객실번호를 붙여 사용한다.
  - 숙박 기록은 이름, 투숙객 수, 전화번호 등으로 구성됨
  - 숙박하고자 하는 손님이 있으면, 원하는 방번호를 묻고, 숙박부 파일을 검색하여 그 방이 비어있으면 배정, 이미 손님이 있는 방이면 다시 방 번호를 물어 나머지 동작을 계속함
  - 손님이 퇴실하면, 숙박부에 기재된 숙박기록을 지움
  - 특정 호실에 숙박 중인 손님 신상 정보 출력하기
  - 손님이 투숙 중인 호실 출력하기
  - 빈방 출력하기
- 프로그램 코드, 설명, 수행 결과 캡처 화면 등을 하나의 파일로 만들어 cms.mmu.ac.kr/bear 과제 제출에 업로드
- 제출기한 : 10월 4일 자정