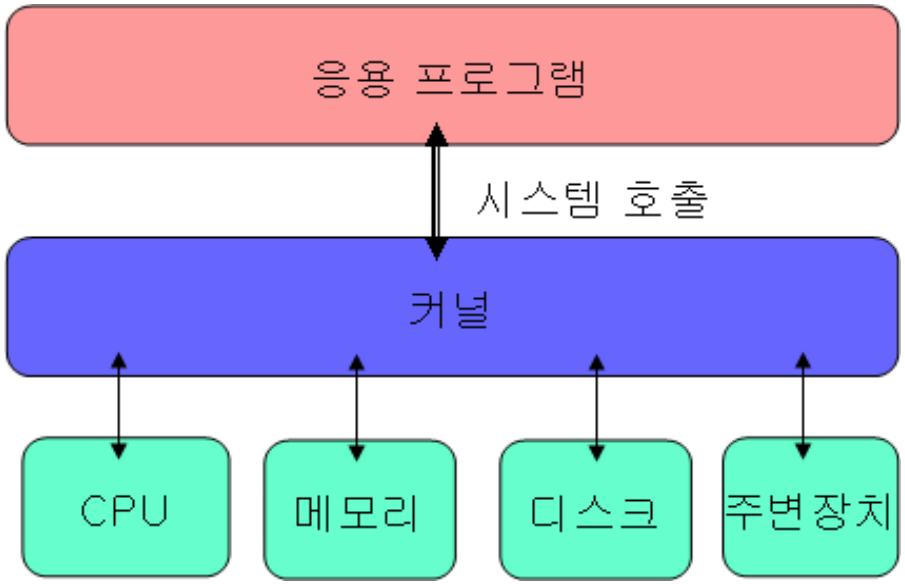
Linux Programming
5장. 파일 입출력

sisong@ut.ac.kr
한국교통대학교 컴퓨터공학전공
송석일

5.1 시스템 호출

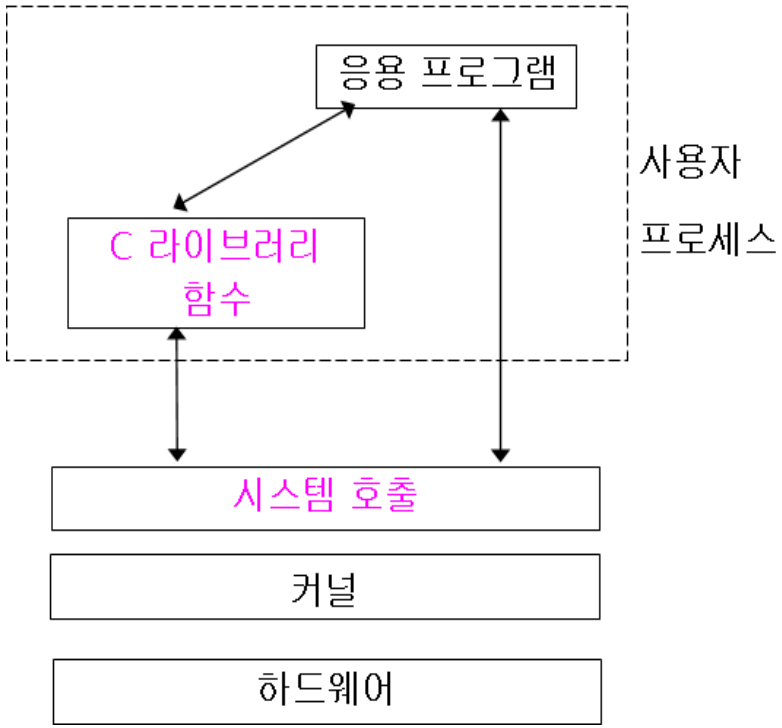
컴퓨터 시스템 구조

- 유닉스/리눅스 커널(kernel)
  - 파일 관리(File management)
  - 프로세스 관리(Process management)
  - 메모리 관리(Memory management)
  - 통신 관리(Communication management)
  - 주변장치 관리(Device management)

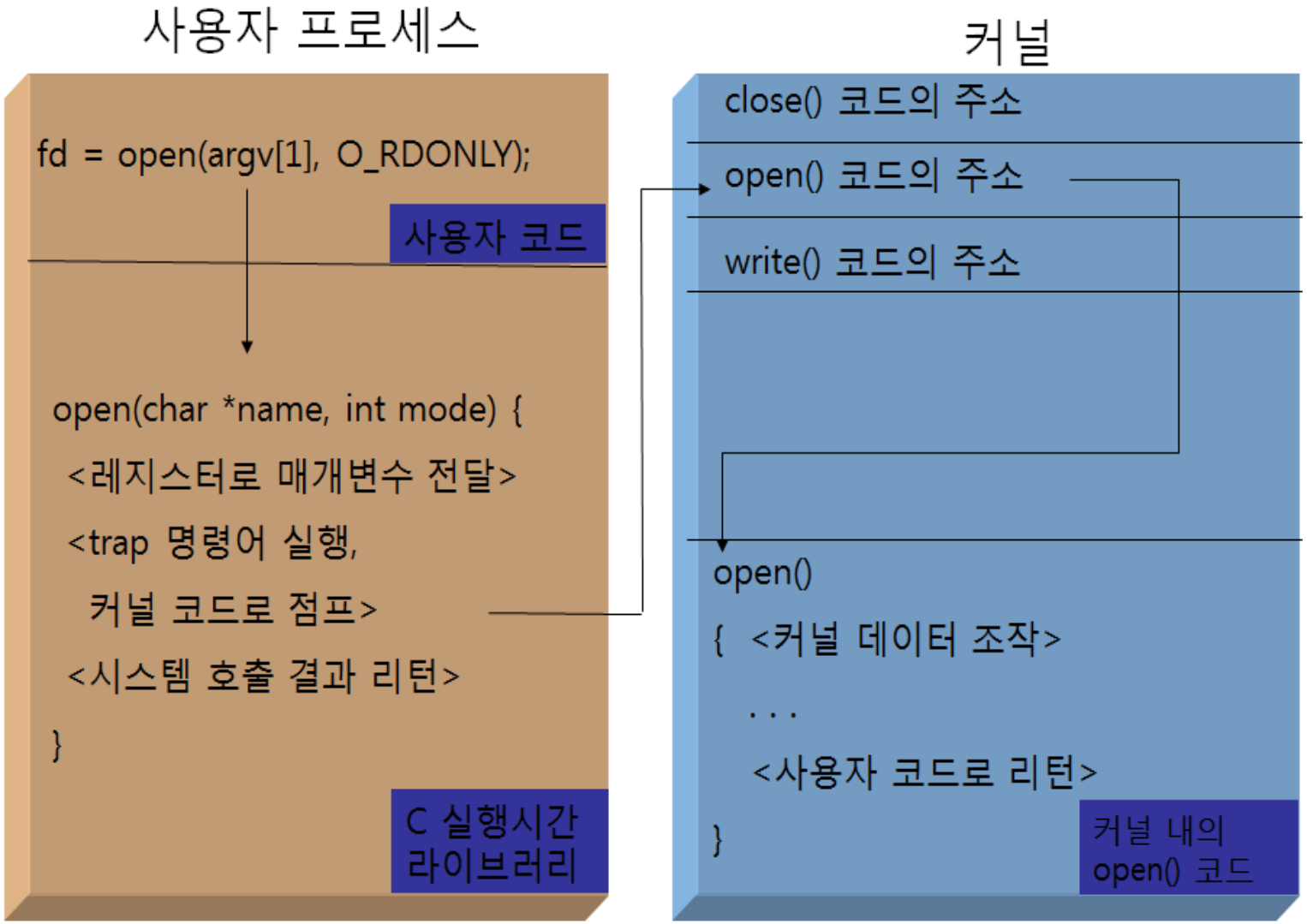


시스템 호출

- 개념
  - 커널에 서비스 요청을 위한 프로그래밍 인터페이스
  - 응용 프로그램은 시스템 호출을 통해서 커널에 서비스 요청 (운영 체제가 제공하는 기능 사용)



시스템 호출 과정


사용자 프로세스
커널
fd = open(argv[1], O_RDONLY);
사용자 코드
open(char *name, int mode) {
<레지스터로 매개변수 전달>
<trap 명령어 실행,
커널 코드로 점프>
<시스템 호출 결과 리턴>
}
C 실행시간
라이브러리
close() 코드의 주소
open() 코드의 주소
write() 코드의 주소
open()
{ <커널 데이터 조작>
. . .
<사용자 코드로 리턴>
}
커널 내의
open() 코드

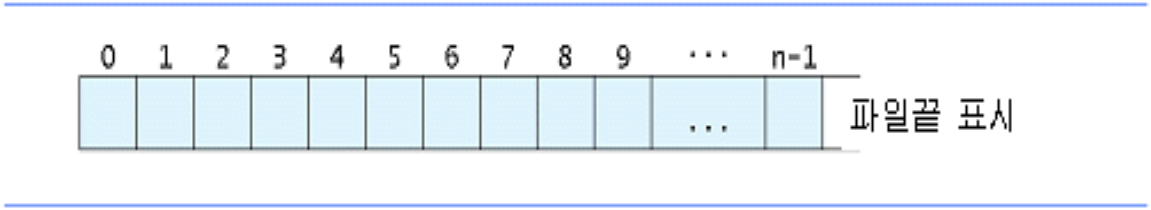
시스템 호출 요약

주요 자원	시스템 호출
파일	open(), close(), read(), write(), dup(), lseek() 등
프로세스	fork(), exec(), exit(), wait(), getpid(), getppid() 등
메모리	malloc(), calloc(), free() 등
시그널	signal(), alarm(), kill(), sleep() 등
프로세스 간 통신	pipe(), socket() 등

5.2 파일

파일

- 커널의 파일
  - 연속된 바이트의 나열
  - 특별한 다른 포맷을 정하지 않음
  - 디스크 파일뿐만 아니라 외부 장치에 대한 인터페이스
  - C 라이브러리 함수에서 처럼 텍스트/바이너리 구분 없음



파일 열기: open()

- open() 시스템 호출

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open (const char *path, int oflag, [ mode_t mode ]);
```

- 파일 열기에 성공하면 파일 디스크립터를, 실패하면 -1을 반환

- 파일을 사용하기 위해서는 먼저 open() 시스템 호출을 이용하여 파일을 열어야 함
- 파일 디스크립터는 열린 파일을 나타내는 번호

파일 열기: open()

- oflag
  - O_RDONLY : 읽기 모드, read() 호출은 사용 가능
  - O_WRONLY : 쓰기 모드, write() 호출은 사용 가능
  - O_RDWR : 읽기/쓰기 모드, read(), write() 호출 사용 가능
  - O_APPEND : 데이터를 쓰면 파일끝에 추가
  - O_CREAT : 해당 파일이 없는 경우에 생성, mode는 생성 파일 사용 권한
  - O_TRUNC : 파일이 이미 있는 경우 내용 삭제
  - O_EXCL : O_CREAT와 함께 사용되며 해당 파일이 이미 있으면 오류
  - O_NONBLOCK : 넌-블로킹 모드로 입출
  - O_SYNC : write() 시스템 호출시 디스크에 물리적으로 쓴 후 종료 (버퍼 사용 X)

파일 열기: 예

```
fd = open("account",O_RDONLY);

fd = open(argv[1], O_RDWR);

fd = open(argv[1], O_RDWR | O_CREAT, 0600);

fd = open("tmpfile", O_WRONLY|O_CREAT|O_TRUNC, 0600);

fd = open("/sys/log", O_WRONLY|O_APPEND|O_CREAT, 0600);

if ((fd = open("tmpfile", O_WRONLY|O_CREAT|O_EXCL, 0666))==-1)
```

파일 열기: fopen.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <unistd.h>
4.  #include <fcntl.h>
5.  int main(int argc, char *argv[])
6.  {
7.    int fd;
8.
9.    if ((fd = open(argv[1], O_RDWR)) == -1)
10.     printf("파일 열기 오류\n");
11.   else printf("파일 %s 열기 성공 : %d\n", argv[1], fd);
12.
13.   close(fd);
14.   exit(0);
15. }
```

파일 생성: creat()

- creat() 시스템 호출

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int creat (const char *path, mode_t mode );
```

- 파일 생성에 성공하면 파일 디스크립터를, 실패하면 -1을 반환

- path가 나타내는 파일을 생성하고 쓰기 전용 오픈
- 생성된 파일의 사용권한은 mode로 정함
- 기존 파일이 있는 경우에는 그 내용을 삭제
- 다음 시스템 호출과 동일
  open(path, WRONLY | O_CREAT | O_TRUNC, mode);

파일 닫기: close()

- close() 시스템 호출

```
#include <unistd.h>
int close( int fd );
```

- fd가 나타내는 파일을 닫는다.
- 성공하면 0, 실패하면 -1을 반환한다.

데이터 읽기: read()

- read() 시스템 호출

```
#include <unistd.h>

ssize_t read ( int fd, void *buf, size_t nbytes );
```

- 파일 읽기에 성공하면 읽은 바이트 수, 파일 끝을 만나면 0,
- 실패하면 -1을 반환한다.

  - fd가 나타내는 파일에서 nbytes 만큼의 데이터를 읽어서 buf에 저장

파일 크기 계산: fsize.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <unistd.h>
4.  #include <fcntl.h>
5.  #define BUFSIZE 1024

6.  /* 파일 크기를 계산 한다 */
7.  int main(int argc, char *argv[])
8.  {
9.    char buffer[BUFSIZE];
10.   int fd;
11.   ssize_t nread;
12.   long total = 0;
13.   if ((fd = open(argv[1], O_RDONLY)) == -1)  perror(argv[1]);

14.   /* 파일의 끝에 도달할 때까지 반복해서 읽으면서 파일 크기 계산 */
15.   while( (nread = read(fd, buffer,    BUFSIZE)) > 0)
16.     total += nread;

17.   close(fd);
18.   printf ("%s 파일 크기 : %ld 바이트 \n", argv[1], total);
19.   exit(0);
20. }
```

데이터 쓰기: write()

- write() 시스템 호출

```
#include <unistd.h>
ssize_t write (int fd, void *buf, size_t nbytes);
```

- 파일에 쓰기를 성공하면 실제 쓰여진 바이트 수 반환
- 실패하면 -1을 반환

  - buf에 있는 nbytes 만큼의 데이터를 fd가 나타내는 파일에 쓰기

파일 복사: copy.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <unistd.h>
4.  #include <fcntl.h>
5.  /* 파일 복사 프로그램 */
6.  main(int argc, char *argv[])
7.  {
8.    int fd1, fd2, n;
9.    char buf[BUFSIZ];

10.   if (argc != 3) {
11.     fprintf(stderr,"사용법: %s file1 file2\n", argv[0]);
12.     exit(1);
13.   }

14.  if ((fd1 = open(argv[1], O_RDONLY)) ==   -1) {
15.     perror(argv[1]);
16.     exit(2);
17.   }

18.   if ((fd2 =open(argv[2], O_WRONLY |
19.     O_CREAT|O_TRUNC 0644)) == -1) {
20.     perror(argv[2]);
21.     exit(3);
22.   }
23.   while ((n = read(fd1, buf, BUFSIZ)) > 0)
24.     write(fd2, buf, n);   // 읽은 내용을 쓴다.
25.   exit(0);
26. }
```

파일 디스크립터 복제

- dup()/dup2() 호출

#include <unistd.h>

int dup(int oldfd);

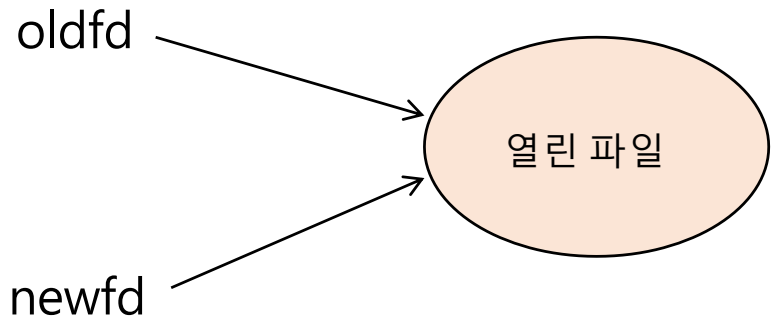
- oldfd에 대한 복제본인 새로운 파일 디스크립터를 생성하여 반환한다.
- 실패하면 –1을 반환한다.

int dup2(int oldfd, int newfd);

- oldfd을 newfd에 복제하고 복제된 새로운 파일 디스크립터를 반환한다.
- 실패하면 –1을 반환한다.

  - 기존의 파일 디스크립터를 복제
  - oldfd와 복제된 새로운 디스크립터는 하나의 파일 공유

파일 디스크립터 복제



파일 디스크립터 복제: dup.c

```
1.  #include <unistd.h>
2.  #include <fcntl.h>
3.  #include <stdlib.h>
4.  #include <stdio.h>
5.  int main()
6.  {
7.    int fd1, fd2;
8.    if((fd1 = creat("myfile", 0600)) == -1)
9.      perror("myfile");
10.
11.   write(fd1, "Hello! Linux", 12);
12.   fd2 = dup(fd1);
13.   write(fd2, "Bye! Linux", 10);
14.   exit(0);
15. }
```

```
$ dup
$ cat myfile
Hello! LinuxBye! Linux
```

5.3 임의 접근 파일

임의 접근과 파일 위치 포인터

- 파일 위치 포인터
  - 파일 내에 읽거나 쓸 위치인 현재 파일 위치(current file position)

- 임의 접근 파일(random access file)
  - 파일 내의 원하는 지점으로 바로 이동하여 읽기/쓰기 수행

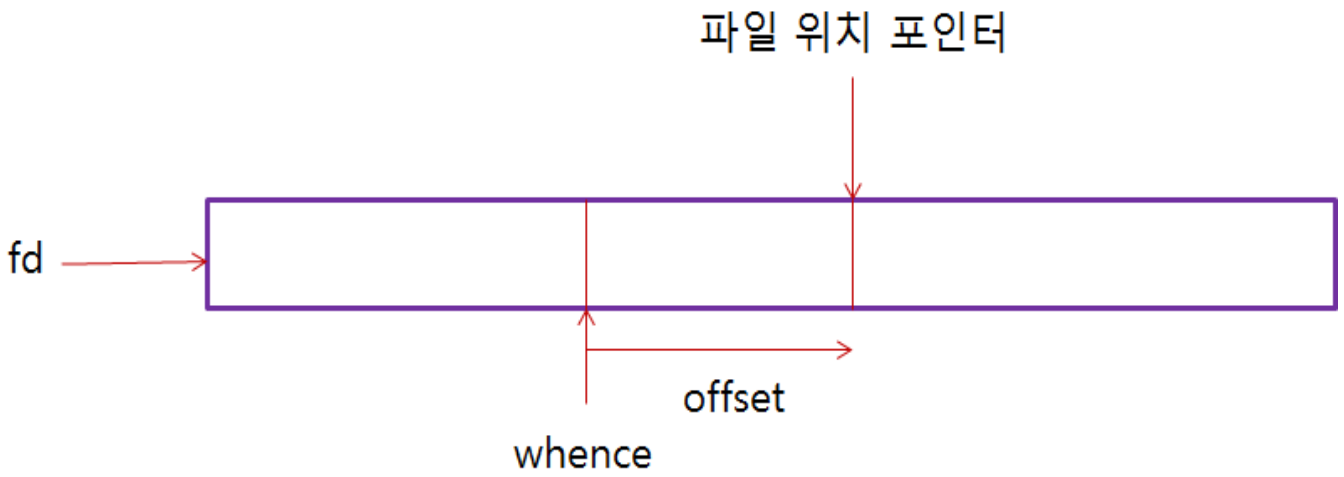
임의 접근: lseek()

- lseek() 시스템 호출

```
#include <unistd.h>
off_t lseek (int fd, off_t offset, int whence );
```

- 이동에 성공하면 현재 위치를 반환하고 실패하면 -1을 반환한다.

- 임의의 위치로 파일 위치 포인터를 이동시킬 수 있다.



파일 위치 포인터이동: 예

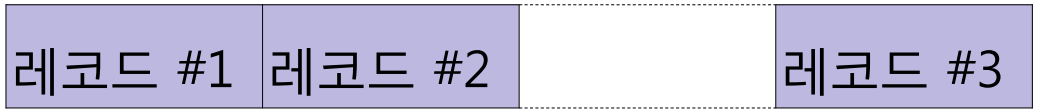
- 파일 위치 이동
  - lseek(fd, 0L, SEEK_SET); 파일 시작으로 이동(rewind)
  - lseek(fd, 100L, SEEK_SET); 파일 시작에서 100바이트 위치로
  - lseek(fd, 0L, SEEK_END); 파일 끝으로 이동(append)

- 레코드 단위로 이동
  - lseek(fd, n * sizeof(record), SEEK_SET); n+1번째 레코드 시작위치로
  - lseek(fd, sizeof(record), SEEK_CUR); 다음 레코드 시작위치로
  - lseek(fd, -sizeof(record), SEEK_CUR); 전 레코드 시작위치로 .

- 파일끝 이후로 이동
  - lseek(fd, sizeof(record), SEEK_END); 파일끝에서 한 레코드 다음 위치로

레코드 저장 예

- write(fd, &record1, sizeof(record));
- write(fd, &record2, sizeof(record));
- lseek(fd, sizeof(record), SEEK_END);
- write(fd, &record3, sizeof(record));

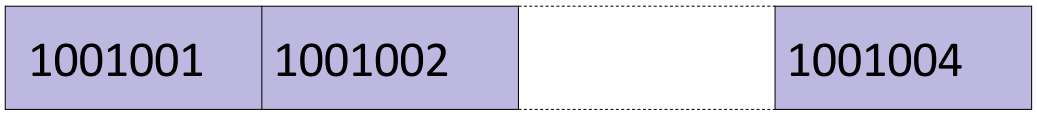


학생 레코드 파일 예제

- 임의 접근을 이용한 학생 레코드 파일
  - 저장(dbcreate.c)
  - 질의(dbquery.c)
  - 수정(dbupdate.c)

- 학생 레코드 저장 위치
  - 학번을 기준으로 학생 레코드를 해당 위치에 저장
  - 학번(rec.id)에 해당하는 학생 레코드의 위치

(rec.id – START_ID) * sizeof(rec)



임의 접근을 이용한 학생 레코드 저장: dbcreate.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <unistd.h>
4.  #include <fcntl.h>
5.  #include "student.h"
6.  /* 학생 정보를 입력받아 데이터베이스 파일에 저장한다. */
7.  int main(int argc, char *argv[])
8.  {
9.    int fd;
10.   struct student record;
11.   if (argc < 2) {
12.     fprintf(stderr, "사용법 : %s file\n", argv[0]);
13.     exit(1);
14.   }
15.   if ((fd = open(argv[1], O_WRONLY|O_CREAT|O_EXCL, 0640)) == -1) {
16.     perror(argv[1]);
17.     exit(2);
18.   }
19.   printf("%-9s %-8s %-4s\n", "학번", "이름", "점수");
20.   while (scanf("%d %s %d", &record.id, record.name, &record.score) == 3) {
21.     lseek(fd, (record.id - START_ID) * sizeof(record), SEEK_SET);
22.     write(fd, (char *) &record, sizeof(record) );
23.   }
24.   close(fd);
25.   exit(0);
26. }
```

student.h

```
#define MAX 24
#define START_ID 1001001

struct student {
  char name[MAX];
  int id;
  int score;
};
```

```
$ dbcreate stdb1
학번 이름 점수
1001001 박연아 96
1001003 김태환 85
1001006 김현진 88
1001009 장샛별 75
^D
```

임의 접근을 이용한 학생 레코드 검색: dbquery.c

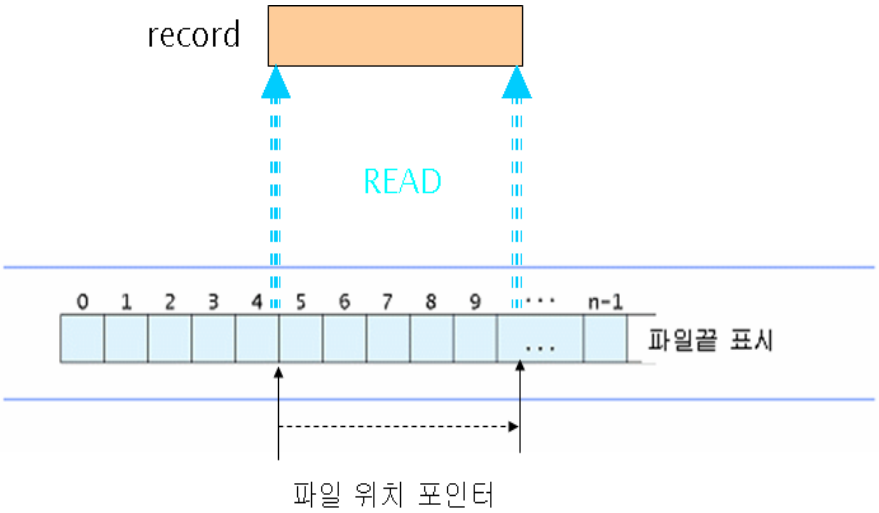
```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <unistd.h>
4.  #include <fcntl.h>
5.  #include "student.h"
6.  /* 학번을 입력받아 해당 학생의 레코드를 파일에서 읽어 출력한다. */
7.  int main(int argc, char *argv[])
8.  {
9.    int fd, id;
10.   struct student record;
11.   if (argc < 2) {
12.     fprintf(stderr, "사용법 : %s file\n", argv[0]);
13.     exit(1);
14.   }
15.   if ((fd = open(argv[1], O_RDONLY)) == -1) {
16.     perror(argv[1]);
17.     exit(2);
18.   }
19.
```

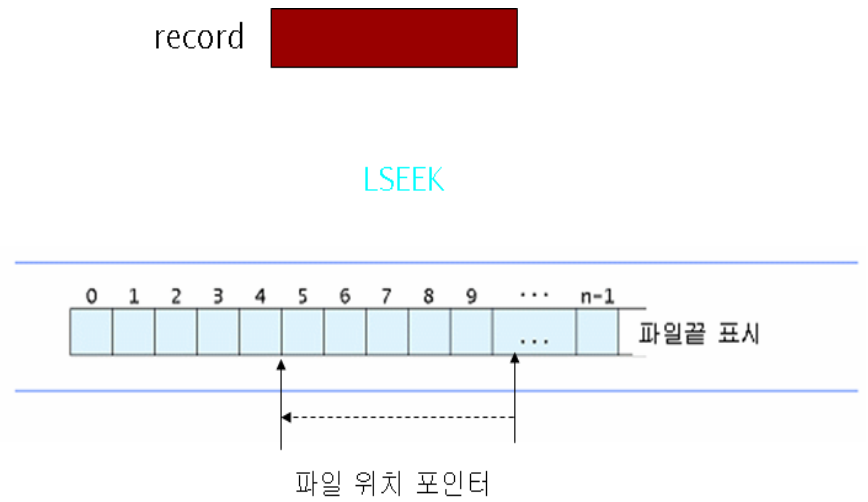
dbquery.c

```
20.  do {
21.      printf("\n검색할 학생의 학번 입력:");
22.      if (scanf("%d", &id) == 1) {
23.        lseek(fd, (id-START_ID)*sizeof(record), SEEK_SET);
24.        if ((read(fd, (char *) &record, sizeof(record)) > 0) && (record.id != 0))
25.          printf("이름:%s\t 학번:%d\t 점수:%d\n", record.name, record.id, record.score);
26.        else
27.          printf("레코드 %d 없음\n", id);
28.      }
29.      else
30.        printf("입력 오류");
31.      printf("계속하겠습니까?(Y/N)");
32.      scanf(" %c", &c);
33.    } while (c=='Y');
34.
35.    close(fd);
36.    exit(0);
37.  }
```

```
$ dbquery stdb1
검색할 학생의 학번 입력: 1001003
학번: 1001003 이름: 김태환 점수: 85
계속하겠습니까?(Y/N)Y
검색할 학생의 학번 입력: 1001006
학번: 1001006 이름: 김현진 점수: 88
계속하겠습니까?(Y/N)N
```

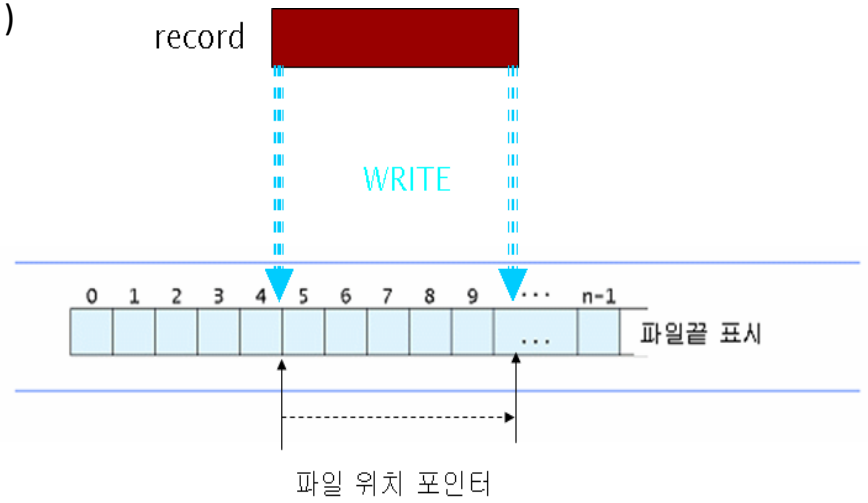
레코드 수정





(1) 파일로부터 해당 레코드 읽기 (파일 -> 변수 또는 메모리)
(2) 레코드를 수정한 후에 (변수 수정)
(3) 수정된 레코드를 다시 파일 내의 원래 위치에 쓰기

Lseek 필요



임의 접근을 이용한 학생 레코드 수정: dbupdate.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <unistd.h>
4.  #include <fcntl.h>
5.  #include "student.h"
6.  /* 학번을 입력받아 해당 학생 레코드를 수정한다. */
7.  int main(int argc, char *argv[])
8.  {
9.    int fd, id;
10.   char c;
11.   struct student record;
12.
13.   if (argc < 2) {
14.     fprintf(stderr, "사용법 : %s file\n", argv[0]);
15.     exit(1);
16.   }
17.
18.   if ((fd = open(argv[1], O_RDWR)) == -1) {
19.     perror(argv[1]);
20.     exit(2);
21.   }
```

dbupdate.c

```
22.  do {
23.      printf("수정할 학생의 학번 입력: ");
24.      if (scanf("%d", &id) == 1) {
25.        lseek(fd, (long) (id-START_ID)*sizeof(record), SEEK_SET);
26.        if ((read(fd, (char *) &record, sizeof(record)) > 0) && (record.id != 0)) {
27.          printf("학번:%8d\t 이름:%4s\t 점수:%4d\n", record.id, record.name, record.score);
28.          printf("새로운 점수: ");
29.          scanf("%d", &record.score);
30.          lseek(fd, (long) -sizeof(record), SEEK_CUR);
31.          //lseek(fd, (long) (id-START_ID)*sizeof(record), SEEK_SET); 도 가능
32.          write(fd, (char *) &record, sizeof(record));
33.        } else
34.          printf("레코드 %d 없음\n", id);
35.      } else
36.        printf("입력오류\n");
37.      printf("계속하겠습니까?(Y/N)");
38.      scanf(" %c",&c);
39.    } while (c == 'Y');
40.
41.    close(fd);
42.    exit(0);
43.  }
```

```
$ dbupdate stdb1
수정할 학생의 학번 입력: 1001009
학번: 1001009 이름: 장샛별 점수: 75
새로운 점수 입력: 85
계속하겠습니까?(Y/N)N
```