[운영체제의 실제] ex3-2와 ex3-3 개선하기 Assignment Report - **20171718 한정택**

1. Problem

[상황]

- 내용이 있는 data.txt 파일이 있을 때 ex3-3(수신 프로그램)을 ex3-2(송신 프로그램)보다 먼저 실행함
- data.txt 파일이 존재할 때 일반 파일이 아닌 디렉터리 혹은 특수파일일 수 있음
- 다른 사용자나 그룹이 data.txt를 읽으면 안 되는 경우가 있음

[현상]

- 송신 프로그램에서 초기에 전송하는 일부 문자가 수신 프로그램에서 수신되지 않는 오류 발생함
- 특수파일 혹은 디렉터리에 대해서는 데이터를 읽거나 쓰면 안 됨
- 권한이 없는 사용자가 데이터를 볼 수 있으면 안 됨

ex3-2.c

```c
int main(void) {
    int fd,n ;
    char buf[256];
    mode_t mode;

    mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;

    fd = open("data.txt", O_CREAT | O_WRONLY | O_TRUNC, mode);
    if (fd == -1) {
        perror("Open data.txt");
        exit(1);
    }

    while(1) {
        write(1, ">> ", 3);
        n = read(0, buf, 255);
        buf[n] = '\0';
        if( n > 0 ) {
            if( write(fd, buf, n) != n ) {
                perror("Write error");
            }
        } else if (n==-1) {
            perror("Read error");
        }
        if(n == 1 && buf[0] == 'q') {
            write(1, "Terminate\n", 10);
            break;
        }
        write(1, buf, n);
    }

    close(fd);

    return 0;
}
```

ex3-3.c

```c
int main(void) {
    int fd,n ;
    char buf[256];

    fd = open("data.txt", O_RDONLY);
    if (fd == -1) {
        perror("Open data.txt");
        exit(1);
    }

    while(1) {
        n = read(fd, buf, 255);
        buf[n] = '\0';
        if (n==-1) {
            perror("Read error");
        } else if (n == 0) continue;
        write(1, "Recv>> ", 7);
        write(1, buf, n);
        if(n == 1 && buf[0] == 'q') {
            write(1, "Terminate\n", 10);
            break;
        }
    }

    close(fd);

    return 0;
}
```

2. 원인

- 초기에 ex3-3이 먼저 실행되면 기존에 존재하는 data.txt 파일을 읽음
- 따라서 ex3-3(Reader)의 file descriptor offset 위치는 파일의 끝에 있음
- 이후 ex3-2(Writer)가 실행되면 O_TRUNC 옵션에 의해 기존의 data.txt의 내용이 삭제됨
- ex3-2의 file descriptor offset은 파일의 시작 지점에 있음
- ex3-2의 offset이 ex3-3의 offset 위치에 도달하기 전까지 ex3-3은 데이터를 읽어오지 않음

3. Opportunity (문제 해결을 위한 기회)

- ex3-2(송신 프로그램)가 실행되면 data.txt의 크기가 N byte에서 0으로 변경됨
- data.txt의 현재 변경기록시간이 잠시 후의 data.txt 변경기록시간보다 이른 경우
- 현재 파일 offset의 위치보다 파일 크기가 큰 경우 data.txt가 수정되었음을 유추할 수 있다.
- **전략** : 파일 변경시간 및 크기를 구하여 문제를 해결해보자!

4.Solution (Core Logic)

1. ex3-2 개선하기

Requirement
가) data.txt 파일이 일반 파일이 아니면, 에러 메시지 출력 후 종료
나) data.txt 파일이 존재하고, GROUP/OTHER에 읽기/쓰기 권한이 있을 때 "data.txt must be protected"와 같은 메시지 출력 후 종료
다) data.txt 파일 생성할 때, 권한은 사용자에 대해서만 [읽기/쓰기] 할 수 있도록 정하기

- fstat을 사용하여 data.txt 파일을 지칭하는 file descriptor로부터 inode 정보를 stat struct로 불러온다.

```
struct stat sb;
// 파일의 inode 정보를 가져오기 위한 fstat
if (fstat(fd, &sb) == -1) {
     perror("fstat error");
     exit(1);
}
```

- stat 구조체의 st_mode 프로퍼티를 통해 담긴 파일의 종류 및 파일 접근 권한 정보를 얻을 수 있다.
- **ISREG() C 매크로 함수를 사용**하면 복잡한 비트 연산자 없이 **일반 파일 여부를 확인**할 수 있다.

```
// data.txt가 일반 파일이 아니라면 오류 출력
if (!S_ISREG(sb.st_mode)) {
      perror("data.txt : Not regular file");
      exit(1);
}
```

- 8진수로 표현된 mode 비트를 직접 쓰거나 주어진 상수를 활용하는 방법이 있다.
- 다음은 **Group 과 Other에 대해서 Read Write 권한**이 주어지면 **프로그램을 종료**시키는 예이다.

```
if (S_IRGRP & sb.st_mode | S_IWGRP & sb.st_mode |
   S_IROTH & sb.st_mode | S_IWOTH & sb.st_mode) {
       perror("data.txt must be protected");
       exit(1);
}
```

- data.txt 파일을 생성할 때 mode 값을 통해 사용자에 대해서만 읽기 쓰기가 가능하도록 설정한다.

```
mode = S_IRUSR | S_IWUSR;

fd = open("data.txt", O_CREAT | O_WRONLY | O_TRUNC, mode);
if (fd == -1) {
     perror("Open data.txt");
     exit(1);
}
```

2. ex3-3 개선하기

Requirement
• ex3-2의 가), 나) 항을 동일하게 적용 • data.txt가 이미 있는 경우, 기존의 내용은 읽지 않기 • 앞서 언급한 ex3-2가 나중에 실행되는 경우의 문제 해결하기 • ex3-3 실행 중 ex3-2가 재실행(혹은 나중에 실행)되는 경우에도 • ex3-2로부터 정상적으로 데이터 수신하도록 만들기

- ex3-2에서 적용된 st_mode를 사용한 파일 정보 및 권한 관리는 동일한 코드를 적용함
- ex3-3에서 처음으로 data.txt를 열었을 때 O_TRUNC 옵션을 주어 파일의 내용을 지우는 설정 진행

```
fd = open("data.txt", O_CREAT | O_TRUNC);
if (fd == -1) {
    perror("Open data.txt");
    exit(1);
}
```

- ex3-3을 진행하던 중에 ex3-2가 재실행되어서 파일의 내용이 지워질 수 있음
- 파일의 크기와 offset 사이에 동기화 문제가 발생할 수 있음
- 따라서 ex3-3은 read를 통해 읽어오는 바이트가 0인 동안 파일 크기를 감시하는 전략을 사용함
- data.txt를 지칭하는 file descriptor를 통해 offset의 위치를 파일의 끝으로 유지 시킴.
- 위 기능을 구현하기 위해 lseek 함수를 사용하였음.

```
while (1) {
    n = read(fd, buf, 255);
    buf[n] = '\0';
    if (n == -1) {
        perror("Read error");
    } else if (n == 0) {
        lseek(fd, 0, SEEK_END);
        continue;
    }
    write(1, "Recv>> ", 7);
    write(1, buf, n);
    if (n == 1 && buf[0] == 'q') {
        write(1, "Terminate\n", 10);
        break;
    }
}
```