**A comparison of f-versions and non-f-versions of the I/O functions**

**201420881 소프트웨어학과 박규동**

소프트웨어를 다룰 때 핵심적인 요소는 여럿 있을 수 있다. 그 요소들 중에 ‘파일’이 반드시 파일이 포함되어야 한다는 주장에 반대 의견을 내놓는 사람은 없을 것이다. 파일은 컴퓨터의 저장 장치에 데이터를 저장하기 위한 도구로 사용된다. 우리는 왜 파일을 사용할까? 프로그램이 종료 될 때 우리는 모든 데이터를 잃어버린다. 이러한 데이터들을 파일로 저장하면 프로그램이 종료되어도 데이터를 보존할 수 있다. 데이터를 저장하지 않으면 데이터의 입력을 처음부터 컴퓨터를 켤 때 마다 해야 하고 이는 당연하지만 비효율적인 일이다. 파일에 저장함으로써 작업을 수월하게 이어 나갈 수 있다. 그리고 파일로 저장함으로써 다양한 데이터의 관리를 용이하게 할 수 있다.

파일을 다루는 다양한 함수들이 있다. 파일을 열 때 사용하는 fopen(), open(), 파일을 닫을 때 사용하는 fclose(), close(), 파일을 읽고 쓸 때 사용하는 fread(), read(), fprintf(), fscanf() 등등 매우 많은 함수들을 우리는 사용한다. 그런데 이 함수들은 f가 붙은 버전과 안 붙은 버전이 있는데 이것들의 차이는 무엇일까?

f가 붙은 입출력 함수는 fopen(), fclose(), fwrite(), fread(), fprintf(), fscanf(), fseek() 등이 있고, 이들의 공통점은 Library call function이다. 반대로 f가 안 붙은 함수들은 System call function이다. System call function은 Unix/Linux와 같은 운영체제의 커널에 서비스를 요청할 때에 호출하는 함수를 말한다. 이들은 주로 하드웨어와 관련된 서비스나 프로세스의 생성/종료, 파일의 입출력 등을 처리하며 System call시에는 프로그램이 사용자 모드가 아닌 커널 모드로 실행된다. System call을 자주 사용하면 시스템 성능에 영향을 주는데 잦은 System call은 리소스를 효율적으로 사용할 수 없기 때문이다. 반면 f가 안 붙은 함수들은 Library call function으로 Library call function들에는 프로그래밍 언어에서 제공하는 기본 함수들과 컴파일러 별로 추가적으로 제공하는 API들이 속한다. 때문에 Library call function은 프로그램 내부에서 메모리의 할당/해제가 가능하다. 따라서 할당과 해제만 제대로 해주면 시스템 성능에는 크게 영향을 주지 않고 사용이 가능하다.

두 함수들의 차이는 우선 리턴 타입에 있다. Library call function의 리턴 타입은 pointer, void, int 등 함수의 목적에 따라 다양하다. 내부적으로 System call function을 호출하다가 오류가 발생한 경우에 대해서는 errno에 오류 코드가 설정되기도 한다. System call function의 리턴 타입은 int이며 오류일 경우 -1이고 정상이면 0 혹은 1이상의 값을 리턴한다. 또한, 오류가 발생할 경우 즉, 리턴값이 -1이면 상세 오류의 내용에 대해서는 errno라는 전역변수에 오류 코드가 저장된다. 오류 내용을 문자열로 표시하기 위해서는 strerror(errno)를 통해서 확인 할 수 있다. Unix/Linux 매뉴얼의 섹션 번호도 차이가 난다. Unix/Linux의 매뉴얼은 명령어 man(manual)을 통해서 제공되고 있으며, manual은 manual의 영역에 따라 섹션 번호를 제공하는데, System call function은 섹션 번호 2번을 사용하고 Library call function은 섹션 번호 3번을 사용한다.

일반적으로 개발자 C언어의 개발자 입장에서 두 f-version과 f가 붙지 않은 함수의 차이는 거의 없다. 그러나 앞서 언급하였듯이 꼭 필요한 경우가 아니면 (운영체제에서 특별한 자원을 가져와야하는 등) System call function보다는 Library call function을 사용하는 것이 좋다. fopen과 open을 비교해 보자. Library call function인 fopen은 버퍼링 I/O를 제공하여 System call function인 open이 수행하는 것보다 훨씬 빠를 수 있다. 그러나 반복적으로 함수를 호출하면 커널에서 바로 호출하는 System call이 어플리케이션의 라이브러리에서 함수를 호출해야 하는 Library call보다 빠르다. 반복하면 할수록 시간의 차이는 크게 벌어진다. 따라서 하나의 긴 파일을 I/O할때는 Library, 반복적인 호출이 사용될 때에는 System call이 좋다. 그러나 System call은 반복적으로 호출할 경우 시스템에 무리가 가고 개발자 입장에서 일정 수 이상의 호출을 하는 경우는 드물기 때문에 일반적으로 Library를 사용한다. 그리고 파일이 바이너리 모드로 열리지 않으면 fopen은 줄, 끝 변환을 수행하므로 프로그램이 Unix 이외의 환경일 경우 유용하게 사용할 수 있다. 애초에 System call은 운영체제에서 불러오기 위해 사용하는 함수이므로 개발 환경에 따라 open기능을 지원하지 않을수도 있다.

아래는 fopen()과 open()의 간단한 코드이다.





처리 속도를 비교해봤을 때 System Call은 반복적으로 사용할 때 처리 시간면에서 이점을 보였고 Library Call은 한번 사용할 때는 System Call보다 이점을 보였다.