3주차 과제

1731017 김민경

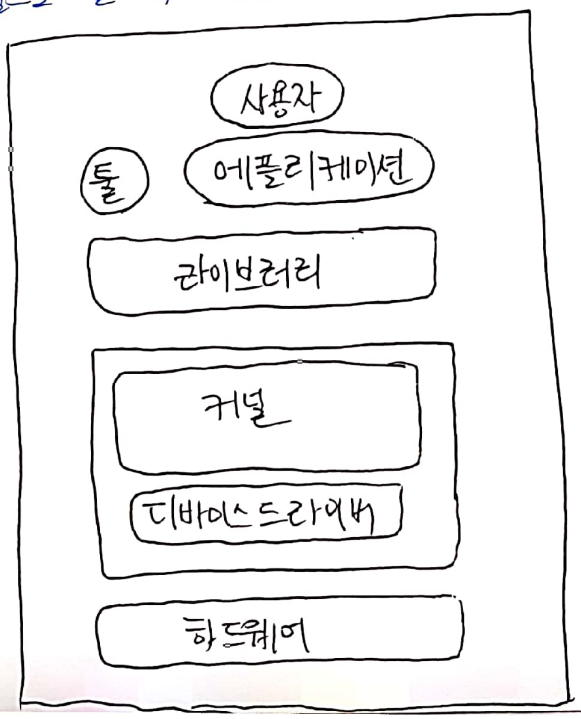
6. 사용자 공간과 커널 공간에 대해 간단히 설명하라. 그리고 사용자 공간과 커널 공간을 분리하는 이유는 무엇인가?

사용자 공간은 응용프로그램들이 탑재되고 사용되는 공간이며, 커널 공간은 디바이스 드라이버를 포함하여 커널 코드가 탑재되고 사용되는 공간이다. 사용자 공간과 커널 공간을 분리하는이유는 커널 공간에 있는 코드와 사용자 영역의 코드를 분리시켜 독립성을 보장하고 데이터를 지키기 위해서이다. 사용자 프로그램이 직접 컴퓨터 자원에 접근하면 자원충돌이 되거나 데이터가 훼손될수 있다. 메모리에 잘못 접근할 수 있고, 파일을 실수로 삭제하는 등 커널 영역 내 데이터를 훼손시키면 시스템에 큰 문제를 발생시킬 수 있다. 따라서 자원에 대한 모든 기능은 사용자 영역에서 다루지 못하고 커널 영역에만 존재하도록 하였다. 즉 메모리를 사용자 공간과 커널 공간으로 분리하게 되었다. 이와 같이 메모리를 사용자 공간과 커널공간으로 나누는것은 운영체제의 역할이다. 예를 들어 32Bit CPU를 가진 윈도우 운영체제라고 한다면 최대 4GB의 주소공간을 사용할 수 있는데, 2중 2GB는 사용자 공간으로 사용하고 나머지 2GB는 커널 공간으로 사용한다. 물론, 구체적으로 여기서 4GB는 가상 공간으로 커널 내의 매핑 테이블을 통해 논리/가상 주소가 실제 물리주소로 바뀌어서 실제 메모리에 할당되거나 커널 영역 내의 명령으로 하드디스크에 메모리가 저장된다. 가상 메모리 공간에도 마찬가지로 사용자 영역과 커널 영역이 분리되어있으며 매핑 테이블을 통해 매핑되는 실제 메모리 공간에서도 커널 영역이 분리되어있다. 운영체제의 커널 영역은 공유되는 메모리이고 사용자 영역은 메모리를 분할해서 사용하고있다.

7. 소프트웨어에서 라이브러리 함수들의 역할은 무엇이며, 라이브러리 함수들은 2가지 유형으로 나누면 어떻게 나눌 수 있는가?

라이브러리는 응용프로그램이 사용할 수 있도록 미리 만들어놓은 프로그램 코드들이다. 직접 구현하기 어렵거나 반복되는 구현과정을 단순화하여 여러 동작들을 미리 정리해놓았기 때문에 편리성이 높고 재사용성도 높다. 라이브러리 함수는 표준 라이브러리 함수, 시스템호출 함수로 나눌 수 있다. 표준 라이브러리와 시스템호출 라이브러리 함수는 모두 함수 호출을 통해 활용된다. 예를 들어서 printf()는 표준 라이브러리 함수, write()는 시스템 호출 라이브러리 함수이다. 표준라이브러리 함수의 경우 운영체제나 하드웨어의 종류와 관계없이 라이브러리와 사용법이 동일하다. 의미와 표현이 동일하여 컴파일 역시 잘된다. 이와 같이 표준 라이브러리를 제공하는 이유는 사용자가 직접 작성하기 힘든 함수를 제공하여 편리하기 때문이다. 시스템 호출 라이브러리는 커널 코드 안에 있는 커널 함수들을 처리하는 시스템 호출을 진행하기 위한 라이브러리이다. write를 사용하면 시스템호출이 진행되는데, 이는 운영체제가 가지고 있는 기능을 반드시 쓰고자 할 때 사용하는것이다. 시스템 호출 라이브러리는 운영체제마다 시스템 호출 함수의 이름이 다르다는 차이점이 있다. 예를 들어 linux에서는 fork()가 windows에서는 createprocess()가 된다. 이를 통해 시스템호출은 운영체제 환경에 따라 고유의 기능을 가지고 있기 때문에 언어의 표현이 다르다는 것을 확인할 수 있다.

8. 사용자, 툴, 에플리케이션, 라이브러리, 커널, 디바이스 드라이버, 하드웨어 등으로 구성되는 전체 컴퓨터 시스템을 그림으로 그리고 각 요소의 기능을 간단히 설명하라. 이 그림으로 볼 때, 운영체제는 어떤 목적이라고 생각이 드는가?



사용자는 에플리케이션이나 툴을 이용하는 주체이다. 툴은 사용자가 컴퓨터를 편리하게 사용할 수 있도록 운영체제 패키지에 포함되어 제공하는 프로그램이다. 에플리케이션은 특정 목적을 위해 제작된 프로그램을 의미한다. 운영체제의 시스템 프로그램을 이용하고 응용해서 특정 기능만 하도록 새로 만들어진 프로그램이다. 라이브러리는 응용 프로그램이 사용할 수 있도록 미리 만들어진 프로그램 코드이다. 디바이스 드라이버는 장치를 직접 제어하는 프로그램 코드이다. 하드웨어는 컴퓨터를 구성하는 기계적 장치이다.

운영체제는 사용자의 계정을 관리하고 사용자의 컴퓨터 사용 시간을 계산한다. 그리고 사용자에게 컴퓨터 사용을 돕는 여러 도구 응용 프로그램을 제공하여 편리한 인터페이스를 제공하고 하드웨어에 관한 자세한 지식이 없어도 하드웨어에 접근하거나 변경하는 등의 조작을 쉽게할 수 있도록 돕는다. 즉, 운영체제는 하드웨어를 제어하는 일은 전적으로 운영체제가 하고 사용자와 하드웨어 사이의 매개체 역할을 한다. 응용 프로그램의 실행 순서를 제어하고 신호 발생이나 데이터를 전달한다. 이와 같은 본리는 응용 프로그램 혹은 사용자가 직접 하드웨어를 다루지 못하게 함으로서 하드웨어 사용 충돌을 막는다.

9. 사용자 모드와 커널 모드에 대해 간단히 설명하라. 커널 모드가 있는 이유가 무엇인가?

사용자 모드와 커널 모드를 설정하는 것은 CPU 내부의 모드 레지스터이다. 레지스터에 모드가 0 (사용자 모드) 혹은 1 (커널 모드)로 설정된다. 사용자 모드는 사용자 공간만을 접근할 수 있는 제한된 권한을 가진다. 사용자 모드에서는 system call 혹은 interrupt 를 통해서만 커널모드로 접근이 가능하다. 사용자 모드는 커널 공간에서 사용할 수 있는 특권명령어를 사용할 수 없고, 사용자 모드에서 사용자 프로그램에 오류가 발생하면 프로그램을 종료한다.

커널 모드는 사용자 공간은 물론, 특권 명령이 존재하는 커널 공간까지 접근이 가능한 권한을 가진 상태를 말한다. 커널 영역의 메모리 번지에 해당하는 메모리를 실행할 수 있다. 또한 하드웨어 접근이 제한되는 사용자 모드와 달리 커널모드에서는 모든 하드웨어 접근이 가능하고, 특권 명령어를 포함한 CPU의 모든 기계 명령어도 가능하다. 커널 모드에서 오류가 발생하면 시스템이 종료된다.

커널 모드가 존재하는 이유는 커널 코드와 데이터에 대한 보안 때문이다. 사용자 코드는 신뢰성이 떨어지기에, 실수 혹은 악의적으로 커널 코드와 데이터를 변경, 훼손 할 수 있으나, 둘을 분리 해놓을 경우 시스템을 종단시킬 수준의 심각한 문제를 미연에 차단할 수 있다.

10. 특권 명령은 어떤 유형의 명령이며, 커널 모드에서만 사용하도록 하는 이유는 무엇인가?

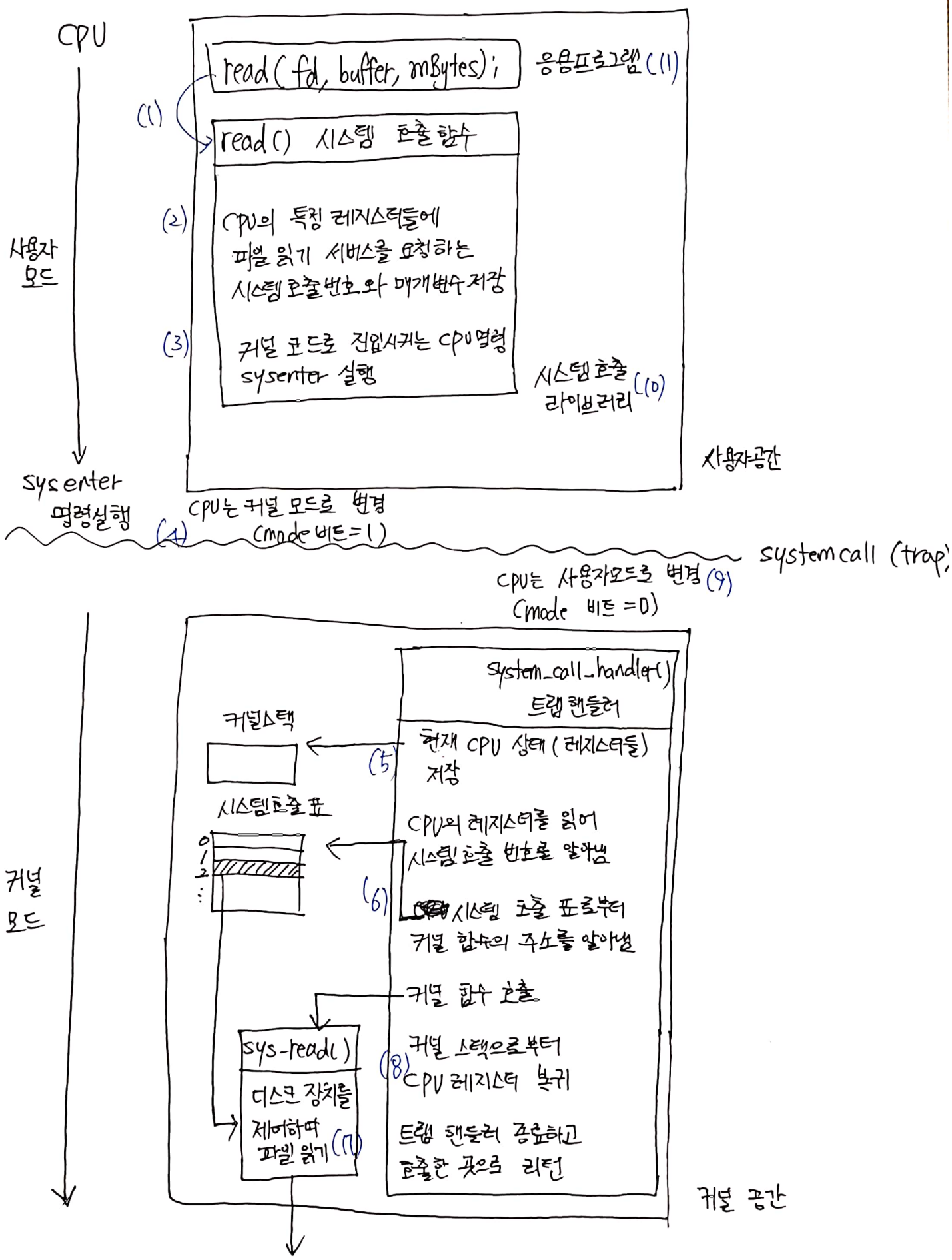
특권 명령이란 커널 모드에서 실행할 특별한 목적으로 설계된 CPU 명령이다. 커널 코드 속에서 실행하는 것이며 종류로는 I/O 명령, 하드웨어 제어 및 장치로부터의 입출력 등이 있다.
구체적으로 예를 들어 HALT 명령어가 있는데 이는 CPU의 작동을 잠시 중지시키는것이다. I/O를 관리하고 있다가 쉬고 있는 경우에 커널이 CPU 에게 HALT 라는 명령어를 준다. 그러면 HLT 라는 기계 명령어를 사용하여 저전력 상태로 바꾼다. 이와 같은 시스템콜 명령으로 인해 전력을 줄이는 방법이있다. 인터럽트 플래그 켜고 끄기 같은 경우도 특권 명령어이다. CPU 안에는 플래그 레지스터가 있는데, 만약 이 플래그 레지스터에 인터럽트를 받지 말것에 대한 명령을 넣은 후에 CLI 나 STI 와 같은 특권 명령어를 통해 인터럽트를 다시 받도록 할 수 있다. 굉장히 중요한 일을 처리해야 할 때 기계어를 이용하여 현재하고 있는 프로세스를 먼저 처리할 수 있도록 인터럽트 플래그를 끄고 킬 수있다. 이와 같은 인터럽트 처리 영역 역시 응용 프로그램이 직접 관리할수 없으므로 커널모드에서만 특권 명령이 적용되는것을 알 수 있다. 이 밖에도 특권 명령어로는 메모리 지우기, 장치 상태 테이블 수정, 컨텍스트 스위칭, 타이머 설정 등이 있다.

특권 명령어를 커널 모드에서만 사용해야하는 이유는 사용자 명령어의 신뢰성이 부족하다는것을 들수 있다. 시스템 보안적인 측면에서도 특권 명령을 사용자 모드에서도 이루어지도록 한다면 하드웨어를 마음대로 제어할 수 있기 때문에 자원이 손상될 우려가 있다. 따라서, 프로세서의 상태보기, 시스템 시간 보기, 시스템 호출 명령 등은 사용자 모드에서 명령을 내릴 수 있지만 특권명령을 처리하려면 시스템호출이나 인터럽트로 커널 모드로 변환 후 사용해야한다.

11. CPU가 평균적으로 커널 모드에서 많이 실행된다. 그 이유는 무엇이라고 생각이드는가?

CPU가 유저모드보다 커널모드에서 시간을 더 많이 보내게 되는 이유로는 몇몇 가지가 있다. 첫번째, 운영체제 내 예외 상황이 발생하는 경우이다. 페이지 문제, 잘못 정렬된 메모리 참조, 부동 소수점 예외 같은 경우는 시스템 시간을 더 많이 필요로 한다. 이와 같은 문제가 발생하는 경우 운영체제는 적절한 패치 과정, 업그레이드, 그리고 충분한 CPU/메모리/디스크 공간을 확보해야하기 때문이다. 두번째, Virtual Machine 과 관련된 경우이다. 만약 가상 환경에서 응용 프로그램이 실행되고 있는경우 애플레이션 시스템의 특성 때문에 사용자 시간 보다 커널 단의 시스템 시간이 더 많이 소요된다. 따라서 가상화된 환경에서는 과부하가 걸리지 않도록 하고 가상 머신에서 응용프로그램을 실행할 수 있는 적절한 자원이 있는지 확인해보아야한다. 세번째, 메모리 제약의 문제가 있을 경우에 시스템 시간이 높아진다. 운영체제는 보통 2MB 정도의 큰 페이지를 요구하게 되는데 이와 같은 페이지를 할당하기 위해 연속 여유 공간을 찾을 수 없는 경우 운영체제는 실행 중인 모든 프로세스를 중지하고 연속 여유 공간을 찾기 위해 데이터를 이동한다. 이와 같이 데이터를 이동하는 과정에서 시스템 시간이 많이 소요된다. 네 번째, I/O 활동이 많을 경우이다. I/O 활동이 많으면 커널에 있는 시간이 많을 수 밖에 없다. 시스템 호출이 발생할 경우, 모드를 변경하면서 CPU 내의 레지스터에 참수 주소와 매개변수들을 넘겨주어야 하며 커널 스택에 CPU 레지스터 값을 저장하는 과정, 디스크 장치를 제어하여 파일을 읽는 과정, CPU 레지스터를 다시 복귀하고 핸들러를 종료하고 복귀하는 과정 등 상대적으로 유저 시간 보다 시스템 시간에서 보내는 시간 비용이 많이 들기 때문이다.

12. 시스템 호출이 일어나는 과정을 그림을 그리고 설명하라



(1) 응용 프로그램은 파일에서 데이터를 읽기위해 read( ) 함수를 호출한다.
read() 함수는 파일 읽기를 위해 운영체제가 제공한 시스템 호출 함수이다.

(2) 커널은 시스템 호출에 의해 불려질 함수(기능) 마다 번호를 매기고,
응용 프로그램에서는 이 번호를 사용한다. read() 함수는 커널 함수의 번호를 CPU의 특정 레지스터에 넣는다. 예를 들어 x86 계열의 CPU 에서는 eax 레지스터에 커널 함수의 번호를 넣는다. 커널 함수의 번호와 함께 매개변수로 미리 정해진 레지스터에 저장한다.

(3) read() 함수는 커널 함수 번호와 매개변수들을 레지스터에 저장한 후, sysenter / int 0x80 과 같은 CPU 명령어를 실행해 커널모드로 진입한다. 이과정을 trap 이라고 한다.

(4) 트랩 과정이 시작되면, CPU는 CPU 내부의 모든 비트를 커널 모드로 바꾸고 미리 설정된 커널 내의 함수(주소)인 system_call_handler() 로 점프한다.

(5) system_call_handler() 는 커널 함수를 호출하기전, CPU의 상태를 보존하기 위해 커널 스택에 CPU 의 모든 레지스터 값을 저장한다.

(6) system_call_handler() 는 read() 함수가 CPU 레지스터에 저장해둔 시스템 호출 번호를 알아내고, 시스템 호출 테이블 (시스템 호출 번호별로 커널 함수의 주소가 저장된 테이블)에서 함수의 주소를 알아내어 커널 함수 (sys_read()) 를 호출한다. 물론, 여기서 sys_read()를 호출한 주체, 커널 스택 모두 응용프로그램에서 관리하는 것이다.

(7) sys_read() 함수는 디스크로 부터 파일 데이터를 읽는다.

(8) system_call_handler() 는 작업을 끝낸 후 응용프로그램을 돌아가기 전에, 커널 스택에 저장해둔 레지스터 값들을 CPU 레지스터에 복귀시킨다.

(9) system_call_handler() 함수에서 sysexit 명령을 실행하여, CPU를 커널 모드에서 사용자모드로 바꾼다.

(10) read() 함수로 리턴한다.

(11) 응용 프로그램으로 리턴한다.

13. 응용 프로그램이 시스템 호출을 많이 사용할수록 좋은가 나쁜가, 아니면 관계없는지 간단히 결론과 이유를 말하고, 표준 라이브러리 함수 fread()와 시스템호출 함수 read()를 사용하는 사례를 들어 설명하라.

응용프로그램이 시스템 호출을 많이 사용할수록 나쁘다. 그 이유는 시간 비용이 들기 때문이다. 시스템 호출 과정에서 사용자모드가 커널 모드로 바뀌고, CPU 레지스터를 커널에 있는 스택에 상태를 저장하는 과정, 커널의 함수를 호출하는 과정 등이 일어나기에 상대적으로 응용 시스템보다 시간 비용이 더 든다. 표준 라이브러리 함수 fread()와 시스템 호출함수 read()를 예로 들어 설명해보도록 하겠다. 두 경우 모두 매개변수에 파일의 이름과 위치, 사용자버퍼(buf[])[1], 그리고 100 바이트 크기만큼의 데이터를 받겠다는 단위를 넘겨준다고 해보자. 총 사용자 버퍼의 크기는 1000 이고 100 바이트 단위의 버퍼 값을 10번 반복해서 넣어서 총 1KB값을 응용 프로그램 내 파일이 읽을 수 있도록한다. 위와 같은 상황에서 fread()의 경우, 처음 fread()를 호출하면 fread()는 표준 라이브러리 내의 버퍼를 살펴본다. fread()는 버퍼가 비어있으므로 read() 함수를 호출하여 디스크 블록 크기 만큼 파일에서 한 번에 읽도록 한다. read() 함수는 시스템 호출을 통해 커널의 파일 읽기함수인 sys_read()를 호출하고, 커널의 이 함수 (sys_read())는 요청했던 바이트 (4kB) 만큼 파일에서 읽고 리턴한다. 즉. 요청한 1000Byte (1KB) 만큼의 데이터를 표준 라이브러리에 돌려주는 역할을 한다. fread()로 돌아오면 fread()는 표준 라이브러리의 버퍼에서 100 바이트를 사용자 버퍼 (buf[])로 복사하고 리턴한다. fread()는 read()에서 버퍼 복사가 이미 되었고 사용자 버퍼까지 한번 저장이 되었기 때문에, 표준 라이브러리 내 버퍼에서 사용자 버퍼로 총 9번의 버퍼 복사가 일어난다. 그리고 시스템 호출은 총 1번 일어난것을 확인할수있다. 반면, 응용 프로그램에서 read() 시스템 호출 함수를 직접 호출하는 경우를 보자. read()를 호출하면 시스템 호출을 통해 커널의 파일 읽기 함수 sys_read()가 호출된다. 커널의 이 함수는 (sys_read())는 요청받은 100 바이트를 파일에서 읽어서 직접 사용자 버퍼로 읽어들인다. 즉 100 byte씩 10번 반복하므로 모두 10번의 시스템 호출이 일어난다. 이와 같은 사례를 통해 볼때, fread()를 이용하는 경우 시스템 호출은 1번만 일어나지만 read()를 이용하는 경우 매번 시스템 호출이 일어나며, 시간 비용이 많이 드는 것을 확인할 수 있다.