System Call

시스템 호출(System Call)은 운영 체제와 사용자 프로그램 사이에서 인터페이스 역할을 하는 중요한 메커니즘입니다. 사용자 프로그램이 운영 체제의 기능을 사용할 수 있도록 커널 모드로 전환하는 과정입니다. 시스템 호출을 통해 프로그램은 파일 시스템 접근, 프로세스 생성, 메모리 관리, 네트워크 통신 등의 작업을 요청할 수 있습니다.

System Call vs Kernel Call

- **System Call**: 운영 체제(OS)가 제공하는 서비스에 접근하기 위한 프로그래밍 인터페이스입니다.
  - 주로 C나 C++과 같은 고수준 언어로 작성되며, API(Application Programming Interface)를 통해 접근됩니다
  - 일반적으로 애플리케이션이 직접 시스템 호출을 사용하기보다는 **POSIX API**나 **Win32 API**와 같은 고수준의 API를 사용합니다. API는 시스템 호출을 간접적으로 다루며, 개발자가 OS의 내부 동작을 알 필요 없이 해당 기능을 쉽게 사용할 수 있게 합니다
    - **POSIX(Portable Operating System Interface)**: 서로 다른 운영 체제 간의 호환성을 유지하기 위한 IEEE 표준
      - 유닉스 계열 운영 체제에서 응용 프로그램이 파일, 프로세스, 네트워크 등의 시스템 자원에 접근할 수 있도록 API를 정의
- **Kernel Call**: 시스템 내부에서 커널과 시스템 프로세스 간에 이루어지는 호출
  - 시스템 호출과 달리 사용자가 직접 호출할 수 없으며, 주로 장치 드라이버와 시스템 프로세스 간의 통신을 위해 사용됩니다.
  - 예를 들어, `sys_devio()` 커널 호출은 I/O 포트를 읽거나 쓰기 위한 호출입니다. 이러한 커널 호출은 시스템 프로세스에만 제한되어 있습니다.

항목	시스템 호출 (System Call)	커널 호출 (Kernel Call)
정의	사용자 모드 애플리케이션이 커널에 작업을 요청하는 메커니즘	커널이 내부에서 다른 커널 함수를 호출하는 메커니즘
발생 위치	사용자 모드에서 커널 모드로 전환됨	커널 내부에서만 발생
예	`open()`, `read()`, `write()`, `fork()`	`do_fork()`, `vfs_read()`, `kmalloc()`
접근 가능 여부	사용자 애플리케이션이 호출할 수 있음	커널 코드 내에서만 호출 가능, 사용자 모드에서는 불가
하드웨어 접근	직접적인 하드웨어 접근 불가능, 커널에 요청	커널 모드에서 하드웨어 자원에 직접 접근 가능

시스템 호출의 일반적인 과정:

1. **사용자 모드(User Mode)**: 일반 애플리케이션이 실행되는 모드로, 제한된 권한만을 가집니다. 프로그램이 운영 체제의 자원에 접근하려고 하면 시스템 호출을 통해 요청합니다.
2. **트랩(Trap) 발생**: 사용자 모드에서 시스템 호출이 발생하면 **트랩(Trap)**이라는 소프트웨어 인터럽트를 통해 커널 모드로 전환됩니다.
    - Trap: 특정한 조건에서 CPU가 실행을 중단하고 운영체제의 커널로 제어를 넘기는 메커니즘
3. **커널 모드(Kernel Mode)**: 운영 체제의 핵심 기능이 실행되는 모드로, 모든 시스템 자원에 접근할 수 있습니다. 트랩이 발생한 후 운영 체제는 해당 요청을 처리하게 됩니다.
4. **시스템 호출 처리**: 요청된 작업이 수행된 후, 결과가 사용자 모드로 반환됩니다. 시스템 자원에 대한 접근 허가 여부, 데이터 처리 결과 등이 이 과정에서 결정됩니다.
5. **사용자 모드로 전환**: 시스템 호출이 끝나면 다시 사용자 모드로 돌아가고, 프로그램은 결과를 받아 작업을 이어나갑니다.

시스템 호출의 주요 유형

1. **프로세스 관리**:
    - `fork()`, `exec()`, `wait()` 등
    - 새로운 프로세스 생성, 프로세스 실행, 프로세스 종료 등을 관리합니다.
2. **파일 관리**:
    - `open()`, `read()`, `write()`, `close()`
    - 파일의 생성, 읽기, 쓰기, 닫기 등의 작업을 처리합니다.
3. **장치 관리**:
    - `ioctl()`, `read()`, `write()`
    - 입출력 장치와의 통신을 제어합니다.
4. **통신**:
    - `socket()`, `connect()`, `send()`, `recv()`
    - 네트워크 통신을 위한 시스템 호출입니다.
5. **메모리 관리**:
    - `brk()`, `mmap()`
    - 메모리 할당 및 해제를 관리합니다.