커널

운영체제의 핵심 구성요소 중 하나로 하드웨어를 추상화하여 하드웨어와 응용프로그램 간의

인터페이스 역할을 수행함.

커널의 목적

사용자 수준의 소프트웨어, 하드웨어 간 통신을 관리하는 것.

Windows는 할당된 프로세스의 주소를 직접 제공하는 것은 보안적으로 위험한 행위임으로 규정하고 해당 프로세스의 할당 주소를 가르키는 핸들을 통하여 통신을 수행한다.

커널의 유형

Monolithic

Application을 제외한 모든 System 서비스를 커널이 직접처리하는 방식

각 서비스들은 커널의 여러 계층에서 관리된다. 이와 같은 방법은 커널의 크기가 상대적으로 커진다는 단점이 있으나 커널 내부 서비스들이 서로 시스템 자원을 공유하며 효율적인 관리가 가능하게 해준다.

OS 서비스들을 시스템 콜 형태로 사용할 수 있게 한다.

커널 내부 서비스를 직접 수행함으로써 빠른 처리속도를 가짐. 하지만 SSO와 동일하게 하나의

오류가 전체 시스템에 영향을 준다.

Ex) 임베디드 리눅스, UNIX

Micro Kernel

Micro Kernel은 다양한 OS 서비스를 Kernel Mode가 아닌 User Mode에서 처리하는 커널 구조

Micro kernel 서버는 커널의 권한을 가지고 직접 하드웨어에 접근할 수 있다는 것을 제외하고 데몬프로그램처럼 실행된다.

Micro kernel은 서버를 추가하는 방식으로 기능을 확장시킬 수 있다.

여러 개의 File System 하드웨어 등을 커널을 변경하지 않은 상태에서 추가할 수 있으며

다양한 OS를 추가하여 선택사용이 가능하다. [개발 용이, 디버깅 쉬움]

프로세스 간 통신이 IPC를 통하여 수행되기 때문에 메시지 전송과 컨텍스트 스위칭이 많이 발생한다. 이는 시스템 복잡도가 증가될수록 시스템 부하, 오버헤드가 급격히 증가한다.

하지만 프로세스가 각 서버 영역에서 수행되기 때문에 하나의 서비스가 에러를 일으켜도 시스템 전체가 다운되는 문제는 발생되지 않는다.

Ex) Windows NT , Mac OS

Windows NT는 커널모드에서 실행되도록 하는 방식은 Monolithic 방식과 닮아 있다.

그래서 Hybrid 커널로 소개되고 있다.

유저모드와 커널모드

컴퓨터 프로세서는 여러 개의 모드를 가짐으로써 전체 시스템을 보호하는 역할을 수행한다.

System Call

Usermode의 Application에서 Kernel mode 서비스를 사용하기 위해서는 정해진 규칙인 System Call을 사용해야만 Trap [ 악성행위] 로 받아들이지 않는다.

즉 System call은 커널 서비스를 사용하기 위한 인터페이스이다.

고급 언어는 System Call을 사용할 수 없기 때문에 API를 사용하여 서비스를 요청한다.

User Mode