커널

커널(Kernel)

정의

운영체제의 핵심 부분으로, 하드웨어와 응용 프로그램 사이에서 중재자 역할을 수행하며, 시스템 자원을 효율적으로 관리하는 핵심 소프트웨어.

특징

- 운영체제의 중요한 역할을 담당
- 부팅 시 메모리에 로드되어 항상 실행 상태 유지
- 응용 프로그램이 하드웨어에 직접 접근하지 못하게 하고, 안정성과 보안성 보장
- 커널 모드에서만 실행 가능 (특권 명령 수행)

핵심 기능

1. **프로세스 관리**
    - 프로세스 생성, 스케줄링, 종료, 전환 등 관리
    - CPU 자원 분배 및 동기화 조정
2. **메모리 관리**
    - 메모리 할당 및 해제
    - 가상 메모리, 페이징, 페이지 교체 등 효율적 관리
3. **파일 시스템 관리**
    - 파일/디렉터리 생성, 삭제, 접근 제어
    - 디스크 I/O 최적화 및 버퍼 관리
4. **입출력(I/O) 관리**
    - 입출력 장치 제어 및 드라이버 관리
    - 인터럽트 기반 I/O 처리
5. **시스템 콜(System Call) 제공**
    - 사용자 프로그램이 OS 기능을 요청할 수 있는 인터페이스 제공
    - ex) read(), write() 등

6. **인터럽트 및 예외 처리**
    - 하드웨어 인터럽트, 소프트웨어 예외, 시스템 콜 등을 처리
    - 안정적인 시스템 동작 유지
7. **보안 및 접근 제어**
    - 사용자 권한, 파일 접근 제어, 메모리 보호
    - 시스템 무결성 유지
8. **네트워크 관리**
    - 프로토콜 스택 관리 (TCP/IP 등)
    - 소켓 통신 지원

커널 모드(Kernel Mode)

정의

운영체제의 커널이 동작하는 특권 실행 모드로,

모든 시스템 자원과 하드웨어에 직접 접근이 가능한 상태.

특징

- CPU의 명령 중 일부는 커널 모드에서만 실행 가능 (특권 명령)
- 커널 코드, 장치 드라이버, 인터럽트 처리 루틴 등이 실행
- 시스템 콜 발생 시 유저 모드 → 커널 모드로 전환
- 커널 주소 공간은 보호되어 있으며, 일반 프로세스는 접근 불가

기능

1. **시스템 콜 처리**
    - 유저 모드 프로그램이 요청한 시스템 자원 접근을 대신 수행
2. **하드웨어 제어**
    - 디바이스 컨트롤러, I/O 장치 제어
3. **메모리 및 프로세스 관리**
    - 페이지 테이블, 스케줄러, 디바이스 큐 관리
4. **인터럽트/예외 처리**
    - 하드웨어 이벤트(입출력, 타이머 등)에 즉각 대응

유저 모드(User Mode)

정의

사용자 응용 프로그램이 실행되는 일반적인 모드로,

시스템 자원에 직접 접근할 수 없는 보호된 실행 환경.

특징

- 특권 명령 실행 불가
- 각 프로세스는 독립된 가상 주소 공간에서 동작 (메모리 격리)
- 시스템 리소스 접근은 반드시 시스템 콜을 통해 커널에 요청
- 한 응용 프로그램의 오류가 시스템 전체로 확산되지 않음

기능

1. **응용 프로그램 실행**
    - 사용자 코드, 라이브러리, 런타임 환경 등이 동작
2. **프로세스 간 보호**
    - 메모리 접근 격리로 안정성 확보
3. **시스템 콜 요청**
    - I/O, 파일 접근 등은 커널 모드에 요청

폴링(Polling)

정의

CPU가 주기적으로 입출력 장치의 상태를 직접 확인(poll) 하여

이벤트 발생 여부를 감지하는 방식.

동작 과정

1. CPU가 일정 주기마다 I/O 장치 상태 확인
2. 장치가 준비되면 데이터를 읽거나 씀
3. 준비되지 않았다면 다시 확인 루프로 복귀

장점

- 구현이 단순하고 하드웨어 지원이 필요 없음

- 소규모 또는 짧은 입출력에서는 응답성이 빠름

단점

- CPU가 “계속 물어보는” 방식이므로 비효율적
- 불필요한 CPU 자원 낭비 및 전력 소비 증가
- 응답 지연 또는 데이터 손실 가능

활용 예시:

초기 임베디드 시스템, 단순 키보드 입력 감시 등

인터럽트(Interrupt)

정의

CPU가 다른 작업을 수행하던 중,

외부나 내부에서 발생한 사건(이벤트)을 처리하기 위해 현재 작업을 잠시 중단하고,

해당 사건을 처리하는 루틴(Interrupt Service Routine, ISR)을 실행하는 메커니즘.

인터럽트의 종류

- **하드웨어 인터럽트:** 키보드 입력, 디스크 완료, 네트워크 패킷 수신 등
- **소프트웨어 인터럽트:** 시스템 콜, 예외(0으로 나누기 등)

동작 과정

1. 이벤트 발생 → 인터럽트 신호 발생
2. CPU는 현재 상태(레지스터, PC 등)를 저장
3. 인터럽트 벡터 테이블을 참조 → ISR 주소 확인
4. ISR(인터럽트 서비스 루틴) 실행
5. 처리 완료 후 이전 작업으로 복귀

장점

- CPU가 불필요하게 기다리지 않음 (효율적)
- 즉각적인 반응 가능 (실시간성 높음)
- 여러 입출력 장치 동시 처리 가능
