- **15.** 운영체제 사례
  - 개관
    - 개관
    - 학습목표
      - **1.** 리눅스의 특징을 이해한다.
      - **2.** 임베디드 리눅스의 차이점을 이해한다.
      - **3.** 윈도우의 특징을 이해한다.
      - **4.** 안드로이드의 특징을 이해한다.
    - 주요용어
      - **POSIX**
      - 임베디드 시스템
      - 경성 실시간 시스템**(hard real-time system)**
      - 연성 실시간 시스템**(soft real-time system)**
  - **1. Linux**
    - **1.** 리눅스 개요
      - 리누스 토발즈가 **MINIX** 기반으로 생성한 오픈소스 운영체제
      - 다양한 **CPU** 지원 **:** 인텔 **CPU, ARM** 등
    - **2.** 리눅스 장단점
      - 장점
        - 무료로 사용할 수 있다.
        - 유닉스와 완벽하게 호환 가능하다.
          - 유닉스 표준인 **POSIX** 거의 모두 준수
        - 안정성이 높다.
        - 낮은 성능의 하드웨어에서도 동작 가능하다.
        - 개인용 컴퓨터를 서버 형태로 동작 가능하다
      - 단점
        - 교육, 유지보수 문제가 발생한다.
        - 보안 문제가 상대적으로 심각할 수 있다.
        - 보급률이 떨어진다.
        - 특정한 하드웨어는 잘 지원되지 않을 수 있다.
    - **3.** 리눅스 커널
      - 리눅스 커널은 토발즈가 일체형 커널**(monolithic kernel)**로 바꾸어 설계
        - 임베디드 시스템의 경우 필요 없는 부분은 제거할 수 있어서 일체형 커널의 단점을 피할 수 있다.
      - 특징
        - 멀티태스킹**(multitasking), 멀티유저(multiuser)** 시스템
          - 프로세서가 하나더라도 시간을 쪼개어 나누어 사용하는 시분할**(time sharing)**을 이용하여 동시에 여러 사용자가 실행하는여러러 개의 프로세스를 지원함
        - 멀티코어**(multicore), 멀티프로세서(multiprocessor)** 지원
          - 멀티 프로세서 **:** 하나의 시스템에 **2**개 이상의 **CPU**가 들어 잇음
          - 멀티코어**:** 하나의 프로세서에 계산을 담당하는 코어**(core)**가 둘 이상
        - 여러 가지 하드웨어 지원
          - 리눅스의 대부분 **C** 언어로 작성**. C** 언어 컴파일러가 있는 플랫폼에서는 리눅스를 이식하는 데 부담이 크게 줄어듦**.** 인텔 계열 **CPU**를 비롯하여 **ARM** 기반 **SoC, ...** 등 현존 마이크로프로세서 대부분 지원
        - **POSIX** 표준 지원
        - 프로세스 간 통신 지원
          - 세마포어**(semaphore),** 메시지 큐**(message queue),** 공유 메모리**(shared memory)** 등 유닉스에서 사용하는 프로세스 간 통신방법 지원
        - 다양한 파일 시스템 지원
          - 현재 리눅스 대부분의 경우 **ext4** 팡리 시스템 기본 지원**, but FAT, NTFS, HPFS** 등 지원 간으
        - 모듈**(module)**

- 새로운 하드웨어를 지원하는 디바이스 드라이버나 추가기능을 제공하는 서비스를 모듈로 만들어, 커널을 교체하거나 시스템을 재시동하지 않고 모듈을 삽입하거나 삭제하는 방식으로 목적 달성
- 리눅스는 일체형 커널 사용 -> 원칙적으로는 새로운 하드웨어가 추가되거나 원하는 기능을 추가하려면 커널을 새로 컴파일한 후 시스템을 재시동해야 함.
- 파일 형태의 주변장치 접근
  - 유닉스 시스템은 전통적으로 주변장치를 파일로 인식하여 처리하며, 주변장치에서 데이터를 읽고 쓰는 것을 파일에서 읽고 쓰는 것으로 간주
    - 디바이스 드라이버가 해야할 일은 파일 시스템에서 읽고 쓰는 함수를 디바이스에서 데이터를 읽고 쓰는 작업에 대응시키는 것

1. Linux (2) [262]
   - 4. 임베디드 시스템과 실시간 시스템
     - 임베디드 시스템(embedded system)
       - 임베디드 시스템(embedded system)은 미리 정해진 특정한 기능을 수행하기 위해 하드웨어와 소프트웨어를 결합하여 설계된 컴퓨터 시스템
         - 한 가지 일을 잘하도록 설계된 시스템
       - 임베디드 시스템은 보통 마감시간(deadline) 등의 주어진 시간제약에 따라 동작하는 실시간 시스템(real-time system)에 이용된다.
         - 시간제약
           - 시간제약은 주어진 이벤트에 대한 반응이 시스템의 상황과 무관하게 정해진 마감시간 내에 이루어져야 함.
     - 종류
       - 기준: 정해진 마감시간 내 작업을 완수하지 못했을 때 어떻게 되는가?
         - 경성 실시간 시스템(hard real-time system)
           - 시스템은 반드시 마감시간 내 작업을 완수
           - 예) 자동차 엔진 제어, 항공기 전자제어 시스템(fly-by-wire)
         - 연성 실시간 시스템(soft real-time system)
           - 시스템이 마감시간 내 작업을 완수하면 좋지만, 그렇지 않다고 해서 시스템이 목적달성에 실패한 것은 아니다.
           - 예) 멀티미디어 재생 (음악이나 동영상은 프레임이라는 정해진 시간 단위로 재생해야 할 부분 있지만, 다음 프레임부터 정상적으로 재생할 수 잇다면 시스템 전체에 문제가 생긴 것은 아님
   - 5. 임베디드 리눅스
     - 임베디드 시스템은 목표를 달성하기 위해 필요한 최소한의 성능을 갖추는 것이 비용 대비 효율이 좋아 저성능 하드웨어로 구현
     - 임베디드 리눅스
       - 임베디드 시스템 목표가 되는 응용에 적합하게 리눅스 커널을 최적화하고 필요 없는 부분을 제거한 것
     - 필요 요건
       - 메인 메모리와 보조 메모리 모두, 임베디드 시스템은 작은 크기이기 때문에 운영체제의 크기를 최소화하며, 기능은 필요한 것만 남기고 다른 부분은 모두 제거
       - 마이크로프로세서도 ㅅ헝능 최적화되어야 함
       - 실시간 시스템의 요구사항에 대응
     - 장점
       - 무료 사용, 운영체제 응용에 적합하게 수정하여 사용 가능
       - 검증가능
       - 인베디드 시스템 간 통신에 쓰이는 TCP/IP, HTTP 등 네트워크 프로토콜 지원
       - 운영체제 최신 동향 대해 가장 빠르게 반영
     - 단점
       - 연성 실시간 시스템을 지원하며 상대적으로 요구되는 하드웨어 사양이 높다
2. Windows [265]
   - 1. 윈도우의 역사
     - 윈도우는 GUI(Graphical User Interface, 그래픽 사용자 인터페이스)를 제공하는 운영체제
     - 1990 Windows 3.0이 발표되기 전까지 윈도우는 운영체제라기보다 MS-DOS라는 텍스트 기반 운영체제에 GUI를 추가하는 응용 프로그램으로 생각됨. PC의 성능이 떨어졌기 때문에 멀티태스킹 등을 제공하는 것도 어려웠고, 그래픽 처리를 하는 데 부담이 컸음.
     - Win32가 추가된 Windows NT: POSIX Windows 3.0의 API를 지원
     - Windows 95 : 윈도우가 MS-DOS 위에서 윈도우 안 포함
     - Windows RT : ARM 마이크로프로세서에서 실행할 수 있음

- **2.** 윈도우 커널
  - **Windows NT :** 마이크로커널을 확장한 형태의 커널 구조
  - 윈도우를 실행하는 컴퓨터의 프로세서
    - 사용자 모드**(user mode)**
      - 가장 낮은 수준의 권한이 주어지며, 자원에 대해 직접적인 접근이 불가능
        - 사용자 모드에서 동작하는 프로그램은 자기 자신의 주소공간을 가지게 됨 다른 사용자 프로그램이나 운영체제가 사용하는 영역은 포함되어 있지 않기 때문에 접근이 봉쇄됨
    - 커널 모드**(kernel mode)**
      - 운영체제 수준의 가장 높은 수준의 권한을 가진 모드 모든 자원을 이용할 수 있음
        - 커널 모드에서 동작하는 프로그램은 다른 프로그램의 주소공간을 읽고 쓸 수 있다. 오작동 시 다른 프로그램의 동작에도 영향을 미칠 수 있음
    - 파일을 읽거나 쓰기 위해서는 파일 시스템과 이에 연결된 **HDD**나 **SSD**에 접근해야 함.
      - 시스템 호출**(API)**
        - 사용자 모드에서는 원칙적으로 불가능하기 때문에 잠시 커널 모드의 권한을 써야 하는데, 시스템 호출**(API)**를 이용하여 가능함
    - **Windows NT**의 기본 구조
      - 커널 모드에서는 마이크로커널 위에 여러 운영체제가 제공하는 서비스가 동작하고 있음
      - **POSIX** 호환을 제공하는 흐름에서 각각의 **OS**는 대응되는 하위 시스템에서 **NT API**를 이용하여 시스템에 접근한다.
      - 마이크로커널 구조
        - 운영체제가 해야 할 가장 핵심적이고 기본적인 일(주소변환, 프로세스 및 쓰레드 관리, 프로세스 간 통신)만 커널 내부에 구현하고, 다른 작업은 서비스 형태로 사용자 영역에서 동작
        - 장점
          - 커널의 규모가 작기 때문에 관리가 편함
          - 필요한 서비스를 설정하기 쉬움
          - 한 서비스에서 문제가 생기더라도 전체 운영체제에 영향을 미치지 않기 때문에 결함 허용성이 높아지는 장점이 있음.
        - 단점
          - 성능 저하
            - 커널 모드가 필요한 작업 할 때 사용자 모드와 커널 모드 사이 전환 및 프로세스 간 통신**(IPC)**로 인한 성능저하
      - 특징
        - **Windows NT** 커널에서는 운영체제의 기본 서비스는 커널 모드를 나가지 않은 상태에서 동작하며, **Win32, POSIX** 등 운영체제 환경은 사용자 모드에서 별도의 프로세스로 동작하도록 만들어서 마이크로커널 구조와 일체형 커널 구조의 장점을 모두 얻을 수 있도록 절충한 구조를 가지고 있다.
        - 하드웨어 자원
          - 추상화 계층을 거쳐 커널과 연결되고, 커널 모드에서 동작하는 여러 서비스가 관리한다.
          - 이 자원들은 커널 모드에서만 접근이 가능하기 때문에 사용자 모드에서 동작하는 프로그램들은 사용자 모드와 커널 모드를 잇는 함수를 통해 이용할 수 있다.
            - **NT**의 네이티브 **API(native API)**
        - 하드웨어를 직접 접근하지 않고 추상화 계층을 거치는 이유
          - 윈도우을 인텔 **x86 CPU** 외의 하드웨어 구조에서 동작시킬 필요가 생겼음.
          - 추상화를 거쳐 (다른 하드웨어 구조에서도) 커널이 통일된 형태로 하드웨어를 접근할 수 있게 만든다.
      - 마이크로커런릉 기반으로 커널 모드에서 동작하는 **Windows NT**의 각 서비스
        - **I/O** 관리자
          - 사용자 모드에서 동작하는 프로그램이 디바이스를 이용할 수 있게 해준다.
            - 구체적으로 읽기/쓰기 요청이 주어지면 이를 해당하는 디바이스 드라이버에 요청을 전달하는 일을 한다.
            - 캐시 관리 수행 : 디스크 장치 성능 향상
        - **Win32** 윈도우 관리자와 그래픽 장치 인터페이스
          - 사용자 입력과 화면 출력 제어
          - 그래픽 장치 인터페이스**(Graphic Device Interface: GDI)**
            - 화면에 글씨를 쓰고, 선과 곡선을 그리는 등의 일을 함.
        - 보안 참조 모니터 **(SRM: Security Reference Monitor)**
          - 자원의 접근 가능 여부를 점검

- LPC 기능 (Local Procedure Call)
  - 같은 기계에서 동작하는 프로세스 사이의 메시지 전달을 통한 정보교환
  - RPC를 윈도우 환경에 맞게 최적화한 버전
- 가상 메모리 관리자
  - 가상 메모리(virtual memory)를 관리
    - 실제 물리적 메모리보다 큰 메모리를 잡고 이를 RAM과 HDD나 SSD 사이의 페이지교환으로 고나리하는 것
    - 각 프로세스별로 물리적인 메모리와 분리된 개별적인 주소공간을 할당하는 일
- 객체 관리자
  - 윈도우의 자우너을 관리하는 서비스
    - 주변장치와 같은 하드웨어이든 파일과 같은 논리적 자원이든 모든 자원은 객체로 간주
- 프로세스 관리자
  - 프로세스와 쓰레드를 생성하고 중단하는 일

3. Android [270]
   1. 모바일 운영체제
      - 과거에는 모바일(mobile) 환경에서 한 가지 목적을 달성하는 것이 주된 목적인 임베디드 시스템이 주, 현대에는 이전에서는 PC가 수행하던 여러 가지 기능을 모바일 단말기가 수행할 수 있도록 개발되고 잇음
      - 모바일 환경의 조건
        - 모바일 장치는 배터리를 통해 동작하기 때문에 전력 소모량을 줄이는 것이 중요한 문제
        - 대부분의 경우 유선이 아니라 무선 네트워크를 통해 인터넷에 연결
        - 작은 크기의 터치 스크린이 입력 및 출력 장치로 사용되는 등 입출력 장치가 PC 환경과 다름.
        - 저수준 운영체제와 고수준 사용자 인터페이스가 결합된 형태로 운영체제가 개발됨
      - 구글 안드로이드(오픈소스), 애플 iOS 모두 유닉스에 기반
   2. 안드로이드의 개요
      - 안드로이드 특징
        - 기본적으로는 운영체제의 소스가 공개되어 잇지만, 회사에 다라 디바이스 드라이버 비공개
      - ARM, x86 CPU를 지원
      - 리눅스에 기반한 일체형 커널구조
      - 운영체제는 C와 C++로 구현되어 있고, 이 위에 Java로 개발된 응용 프로그램이 돌아가는 형태
        - 달빅(Dalvik)
          - 구글이 자체적으로 만든 가상기계 하드웨어에서 달빅이 동작하기만 하면 응용프로그램이 동작하도록 만듦
          - 성능이 제한되는 이유: 응용 프로그램은 실제 기계에서 사용되는 코드가 아닌, 중간 코드 형태인 바이트코드(bytecode) 형태로 컴파일 되고, 이 바이트 코드들은 달빅 가상기계가 실제 기계 코드로 변환
        - 안드로이드 런타임(ART: Android Runtime)
          - 용응 프로그램을 처음 설치할 때 중간 코드를 실제 기계 코드로 번역하고, 이후부터 실행할 때는 이 번역된 코드를 실행시켜 성능을 높임
          - 다만, 이 경우 응용프로그램을 처음 설치할 때 코드를 번역하는 과정이 포함되기 때문에 설치에 시간이 더 걸림
      - 파편화(fragmentation) 문제
        - 애플의 iOS에 비해 다양한 회사의 다양한 하드웨어 지원
        - 특정 하드웨어에서만 동작하는 소프트웨어가 존재하거나, 특정 하드웨어에서는 동작하지 않는 소프트웨어가 존재할 수 있음.

요약

1. 리눅스 커널은 일체형(monolithic) 구조이며, 커널 소스는 공개되어 자유롭게 수정할 수 있다.
2. 임베디드 시스템(embedded system)은 미리 정해진 특정한 기능을 수행하기 위해 하드웨어와 소프트웨어를 결합하여 설계된 컴퓨터 시스템이다.
3. 실시간 운영체제는 주어진 마감시간 내에 필요한 결과를 얻기 위한 실시간 시스템을 위한 운영체제이다.
4. 실시간 시스템은 마감시간을 얼마나 엄격하게 지켜야 하느냐에 따라 경성 실시간 시스템과 연성 실시간 시스템으로 나뉜다.
5. 리눅스는 임베디드 시스템에서 활발히 사용되고 있다.
6. Windows 95부터 본격적인 GUI를 갖춘 운영체제의 형태를 갖추었다.
7. Windows XP 이후는 Windows NT의 구조인 수정된 마이크로커널에 기반한 운영체제이다.
8. Windows NT의 커널, 실행부 서브시스템, 커널 드라이버, 윈도우 관리자, GDI, I/O 관리자, 장치 드라이버 등은 커널 모드에서 동작한다.
9. Android는 iOS와 함께 대표적인 모바일 운영체제로, 리눅스에 기반한 일체형 커널 구조이다.