공개/비공개 소프트웨어

수업내용 요약

운영체제의 의미와 활용

> 운영체제는 컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고, 효과적으로 사용할 수 있도록 환경을 제공하며 대용량의 클라우드 서비스, 데스크탑, 노트북, 스마트 폰, 태블릿 등에서 사용이 됨

운영체제 중 대표적인 공개 / 비공개 소프트웨어

- 공개 소프트웨어

> 리눅스, 안드로이드
>
> 소스코드가 공개되있는 운영체제임

- 비공개 소프트웨어

> windows

- 공개 / 비공개 소프트웨어의 차이

> 라이선스 요금이 무료이면서 소스코드가 공개되어 있는 소프트웨어로서 누구나 자유롭게 사용할 수 있고, 활용할 수 있으며 배포할 수 있는지의 유무
>
> 소스코드를 만들고 그것을 컴파일할때 바이너리가 생성되는데 개발자가 아니라면 생성된 이진파일을 보고 어떻게 프로그래밍을 했는지 알수 없음 이를 무료로 열람하거나 수정할 수 있는지의 여부가 공개/비공개 소프트웨어의 차이

공개 소프트웨어

- 탄생 배경

> 소프트웨어라는 상품은 시장에서 특이한 성질을 갖고있는데, 시장의 대부분의 상품의경우 제조원가와 유지비용이 들지만, 소프트웨어의 경우 개발자 인건비를 제외하면 최소한이 인프라 구축 비용을 제외하면 추가적인 재료비가 들지 않는다.
>
> 또한 독점체제가 가능한 시장이기 때문에 1등을 하는 소프트웨어 개발자는 시장 점유율이 높다.
>
> 이때 손해를 보는 2등 이하의 소프트웨어 업체들은 소스코드를 공개함으로써 여러가지 플랫폼, 교육 시스템, 컨설팅 등 네임벨류가 올라가게 되어 점유율을 차지할 수 있었음, 또한 오픈소스 개발자들이 해당 운영체제의 버그또한 같이 개선을해 시장 점유율을 올릴 수 있었다.\
>
> 결국 2등 이하의 소프트웨어 개발자들이 망했기 때문에 공개 소프트웨어가 탄생하게 되었다,

추가자료

- 공개 소프트웨어(Open Source Software): 소스코드를 공개하며, 소스코드를 일정한 라이선스에 따라 이용할 수 있는 공개된 소프트웨어를 나타내는 용어

- 비공개 소프트웨어(Close Source Software): 소스코드를 공개하지 않으며, 프로그램을 실행할 수 있는 실행파일이나 실행환경만 제공하는 소프트웨어를 나타내는 용어
- 자유 소프트웨어(Free Software): 공개 소프트웨어와 같은 의미이지만, 소프트웨어에 대한 책임과 사용자의 권리를 더욱 강조하는 용어
    - 자유 소프트웨어 운동 : 리처드 스톨먼이 1980년대에 소프트웨어의 본래 생산 유통 방식인 정보 공유의 방식을 복원하고자 한 운동 (1950년대에서 1970년대 초반까지는 대부분의 소프트웨어가 자유롭게 공유되었지만 80년대부터 모든 소프트웨어가 저작권에 의해 공유가 제한)
    - 자유 소프트웨어 원칙 :
        1. 소프트웨어의 작동 원리를 연구하고 이를 자신의 필요에 맞게 변경시킬 수 있는 자유
        2. 소프트웨어를 이웃과 함께 공유하기 위해서 이를 복제하고 배포할 수 있는 자유
        3. 소프트웨어를 향상시키고 이를 공동체 전체의 이익을 위해서 다신 환원시킬 수 있는 자유
- 공개 소프트웨어 라이센스 종류 (라이센스는 소프트웨어의 지적 재산권을 의미 - 음악 저작권과 비슷한 개념)

운영체제

1. 운영체제란?

> 컴퓨터 하드웨어 바로 위에 설치되어 사용자 및 다른 모든 소프트웨어와 하드웨어를 연결하는 소프트웨어 계층

- 좁은 의미 : 운영체제의 커널

  (메모리에 항상 상주하는 부분; 컴퓨터 부팅~종료시까지)

- 넓은 의미 : 운영체제의 커널 뿐만 아니라 메모리에 상주하지 않는 부분까지도 포함한 개념

  (컴퓨터 시스템을 관리하는데 전반적으로 필요한 부분)

2. 목적

여러 사용자가 여러 프로그램을 편리하게 사용할 수 있도록 환경을 제공 → 독자적으로 수행하는 것과 같은 환상을 보여줌

1. 자원을 효율적으로 관리

   주어진 자원으로 최대한의 성능을 내는 효율성,

   형평성 있는 자원 분배의 문제

   기계어를 빠르게 실행(계산) → CPU

   기억 → 메모리

   판단 → 메모리에 올라간 기계어(in 소프트웨어_made by 사람)가 판단

3. 분류

1. 동시 작업 가능 여부

   - 단일 작업 (single tasking)

     초창기 운영체제로, 한 번에 하나의 작업만 처리

- **다중 작업 (multi tasking)**

  여러 프로그램을 동시에 실행하는 경우

  ⇒ CPU 스케줄링, 메모리 스케줄링 등의 문제가 발생

2. 사용자의 수
   - 단일 사용자

     ex) MS-DOS, MS Windows

   - **다중 사용자**

     한 대의 컴퓨터에 여러 사용자가 터미널 형태로 동시 접속하는 경우 → 보안 문제 및 제약 조건 필요할 것

     유닉스, 리눅스는 다중 사용자의 동시 사용을 지원

3. 처리 방식
   - 일괄 (batch processing)

     작업 요청의 일정량 모아서 한꺼번에 처리

     작업이 완전히 종료될 때까지 기다려야 함

   - **시분할 (time sharing)**

     여러 작업을 수행할 때 컴퓨터 처리 능력을 일정한 시간 단위로 분할하여 사용

     일괄 처리 시스템에 비해 짧은 응답 시간

   - 실시간 (Realtime OS)

     deadline이 정해져 있어, 반드시 보장되어야 함 (deadline 꼭 만족)

     - Hard realtime system : 원자로/공장제어, 미사일 제어 등
     - Soft realtime system : 사용자 많아 동영상이 끊기는 경우

4. 추가 용어 설명

- multitasking
- multiprogramming : 메모리 측면을 강조한 것으로, 메모리에 여러 프로그램이 동시에 올라가 있는 상황
- time sharing : cpu를 강조, CPU의 시간을 분할하여 나누어 쓴다는 의미
- multiprocessor : 하나의 컴퓨터에 CPU가 여러 개 붙어있음을 의미

(충돌 등의 문제가 있을 수 있음)

⇒ 모두 컴퓨터에서 여러 작업을 동시에 수행하는 것을 뜻함

5. 운영 체제의 예

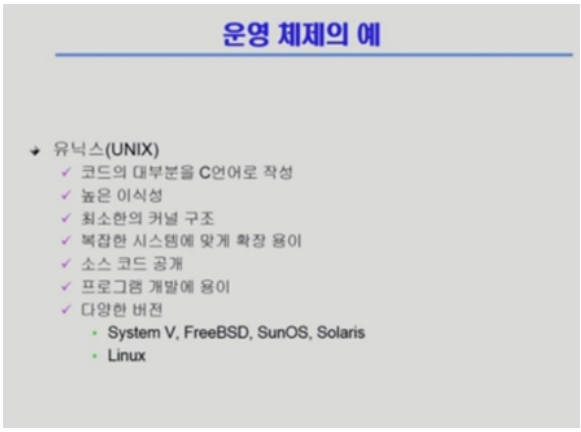
1. 유닉스

> 서버를 위한 운영체제

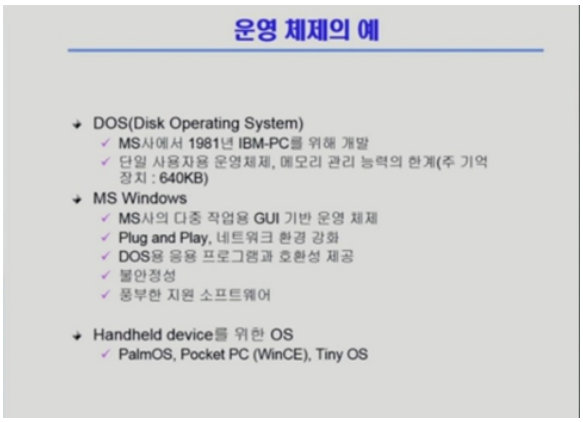
→ *여러 사용자, 여러 프로그램*을 위한 운영체제 필요

⇒ c언어로 작성 (어셈블리어에 비해 크게 비효율적이지는 않음)

⇒ 사람이 이해하기 쉬움(c언어) - 높은 이식성 (호환이 높기 때문), 유지보수 간단



2. DOS : MS사에서 개발한 개인 사용자를 위한 운영체제 (단일 사용자-단일작업)
3. MS Windows : DOS에서 발전시켜옴



6. 운영체제의 구조


운영 체제의 구조
누구한테
CPU를 줄까?
CPU 스케줄링
CPU
한정된 메모리를
어떻게 쪼개어 쓰지?
메모리 관리
memory
Disk
I/O device
디스크에 파일을
어떻게 보관하지?
파일 관리
각기 다른 입출력장치와
컴퓨터 간에 어떻게
정보를 주고 받게 하지?
입출력 관리
프로세스관리
• 프로세스의 생성과 삭제
• 자원 할당 및 반환
• 프로세스 간 협력
그 외
• 보호 시스템
• 네트워킹
• 명령어해석기
(command line interpreter)
1/3