OS introduction 1

OS 개요

OS란 무엇인가?

- Operating System
- 세계에서 가장 복잡한 소프트웨어
- 시스템 소프트웨어
- 사용자가 하드웨어를 사용할 수 있도록 도와주는 인터페이스
- 응용 프로그램을 동작시킬 수 있는 환경을 제공하고 컴퓨터 하드웨어를 관리하는 프로그램
- 자원 할당자 : CPU, 메모리, 디스크 같은 모든 자원들을 관리하고 자원 공유, 사용을 적절히 분배
- 제어 프로그램 : 에러나 부적절한 컴퓨터 사용을 방지하기 위해 프로그램 실행을 제어

OS가 하는 일?

- 자원 분배
- 프로그램을 실행시키기 쉽도록 만드는 것
- 프로그램이 메모리를 공유할 수 있도록 만드는 것
- 프로그램이 하드웨어 장치들을 활용할 수 있도록 만드는 것
- 시스템이 정확하고 효율적으로 운영되도록 만드는 것

+) 모든 프로그램은 시스템 프로그램 or 응용 프로그램

커널이란?

- 컴퓨터가 켜진 동안 항상 돌아가는 프로그램

Bootstrap program?

- 컴퓨터를 power-up, reboot, start-up 시 실행
- ROM이나 EEPROM에 존재 (firmware)
- firmware?
  - 하드웨어와 소프트웨어 사이 중개자 역할을 하는 프로그램
  - 하드웨어가 제대로 작동할 수 있도록 기본적인 제어와 명령 제공
- 시스템의 모든 영역을 초기화
- 커널을 로드하고 실행
- OS는 첫 번째 프로세스를 실행하고 이벤트가 발생하길 기다림 (인터럽트)

boostrap의 어원

- bootstrap은 부츠에 달린 끈을 의미
- 19세기 영미권에서 부츠끈을 당겨 내 몸을 들어올린다는 말이 유행
- 불가능한 자력갱생을 의미
- 근데 마치 OS가 스스로를 켜는 것이 bootstrap 농담 같아서 이런 이름을 갖게 됨

인터럽트

어떻게 작동하는가?

- interrupt vector를 통해 interupt가 interrupt service routine에 제어권을 넘김
- interrupt architecture (인터럽트를 채택한 구조) 는 interrupted instruction의 주소를 반드시 저장해야함
  - OS는 레지스터와 프로그램 카운터를 저장해서 CPU 상태를 보존함

- 현재 처리 중인 인터럽트가 있는데 incoming 인터럽트가 있다면 인터럽트 손실을 막기 위해 incoming 인터럽트를 disable

trap은 소프트웨어가 생성한 인터럽트

- CPU의 흐름을 잠시 바꿔버리는? 덫을 설치
- 시스템 콜일 수 도 있고 페이지폴팅 일 수 도 있고
- trap handler가 알맞은 인터럽트 유형으로 처리
- systemcall handler는 systemcall service rountine의 주소들을 mapping한 table임
- handler는 약간 router같은 역할인 듯

OS는 인터럽트 기반

Polling vs Vectored Interrupt System

- 폴링 : 엣날 방식 (HW, SW 상황을 매 순간순간 검사함)
- vectored interrupt system : 이벤트 발생 시 interrupt 발생, cpu에 전달
- 기능이 다양, 복잡해질수록 폴링보단 인터럽트가 효율적

Interrupt Timeline

- I/O device가 I/O request를 하면 interrupt를 transfer
- CPU는 interrupt가 들어오면 하던 일을 멈추고 I/O interrupt를 processing
- 끝나면 본래 하던 일을 다시 시작
- 반복

프로그램 실행 과정

- auxiliary storage device에서 프로그램을 memory에 loading
- cpu가 memory 속 instruction을 one by one 읽음
- 이 역할을 OS가 수행

프로그램이 실행되면 어떤 일이 일어날까

typical instruction-execution cycle

1. 프로그램 로딩
   - CPU는 메모리에서만 instruction 로드가 가능
   - 어떤 프로그램이든 실행시킬려면 메모리에 적재되어야 함
2. intruction fetching
   - processor (cpu) 가 메모리에서 instruction을 fetch
3. instruction decoding
4. 연산자 fetching
5. intruction 실행
   - ex) add two numbers, 메모리 엑세스, check a condition, jump to function, ...
6. 결과 저장 후 2번부터 시작
   - 프로세서는 다음 instruction으로 이동

System call

시스템 콜이란 무엇인가?

- 유저가 OS에게 뭘 할지 알려줄 수 있음
- 보통 OS는 몇백개의 시스템 콜을 제공 (exports)
  - ex) Run program, Access memory, Access device, ...
- 사용자 프로그램이 커널에 직접 요청을 보낼 때 사용하는 인터페이스

c언어에서 사용되는 시스템 콜

- C언어에서 open(), write(), close() 도 시스템 콜임
- OS에 file system이라는 부분이 있는데 여기로 시스템 콜이 전달됨
- file system이 이 요청을 처리함

---

virtualization이란?

- 실제 hardware resource를 물리적으로 직접 사용하지 않고 software를 통해 논리적으로 추상화하여 사용하는 기술
- 모든 프로세스가 마치 각자의 메모리를 가진 듯한 착시
- OS는 physical resource를 virtual form으로 변환시킴
  - physical resource : processor, memory, disk, ...
- virtual form은 more general, powerful, easy-to-use
- 때때로 OS를 virtual machine이라고도 함
  - CPU, memory 등을 virtualizing 하기 때문
- OS는 한 번에 많은 것들을 관리
- 현대 multi-threaded 프로그램들에게서 concurrency 문제가 나타남 (어쩌라고)

---

persistence

- hw와 sw는 데이터를 영구적으로 저장해야할 때가 있음

C언어로 데이터를 persistently하게 저장

- open(), write(), close() 사용
- OS는 disk에 write하기 위해 새 데이터를 어디 저장할지 계산, storage device에 I/O request를 보냄
- file syste은 write 하는 동안 system crash를 처리

데이터가 순서대로 잘 write 되도록 관리하기

journaling (write-ahead logging, wal)

- 데이터를 디스크에 쓰기 전 로그에 기록
- 장애 발생 시 로그를 이용해 복구

copy-on-write (cow)

- A, B 프로세스가 존재한다고 가정
- 이때 A, B는 같은 데이터를 공유 (초기에는 데이터가 같으므로 독립적인 두 개의 데이터를 사용하지 않음 -> 리소스 효율 up)
- 만약 A가 해당 데이터를 수정하려 한다면 A는 해당 데이터를 복사해서 자기만의 데이터를 만들고 이 데이터를 수정
- 즉, 초기에는 데이터를 공유
- 수정하길 원하는 프로세스는 따로 떨어져 나와 데이터를 복사 후 수정