OS introduction 2

저장장치 Hierarchy

Hierarchy

- hierarachy의 기준이 되는 3가지 : 속도, 비용, 휘발성
- hierarchy ↑ 일수록 faster, more expensive, volatile
- hierarchy ↓ 일수록 slower, cheaper, non-volatile

Caching

- 더 빠른 저장 장치에 데이터를 저장하는 것
- main memory는 어떻게 보면 secondary storage의 cache임

Computer System Operation

I/O 장치와 CPU 사이 커뮤니케이션

- I/O device들과 CPU는 동시에 작동
- 각각의 device controller는 특정 타입의 device를 관리하고 local buffer를 갖고 있음
- device driver가 device controller를 이해할 수 있음
- CPU는 데이터를 controller속 local buffer에 transfer or local buffer에서 fetch
- I/O는 device에서 local buffer 까지만 존재
- device controller는 작업이 끝나면 interrupt를 발생시켜서 CPU에게 알림

DMA

Direct Memory Access

왜 DMA를 사용하는가?

- 인터럽트 기반 I/O는 대량 데이터를 처리하기엔 High overhead가 발생
- memory 속도와 비슷하게 데이터를 전달 가능
- device controller가 데이터 블록 전체를 local buffer <-> memory로 CPU의 개입 없이 옮길 수 있음
- DMA를 사용함으로써 매 바이트마다 발생하던 interrupt를 매 블록마다 interrupt가 발생하게 됨

Computer System Architecture

- 대부분 컴퓨터는 a single general-purpose processor
- 그 외 시스템도 존재

종류

- single - processor system
- multi - processor system
  - symmetric multiprocessing architecture
  - asymmetric multiprocessing architecture
  - dual-core architecture
  - ...
- distributed system
  - clustered system
  - grid computing system
  - ...

Operating System Structure

multiprogramming

- batch system
- 효율을 위해 필요
- 한 명의 사용자가 항상 CPU와 I/O를 사용하는 건 아님
- job (code & data) 를 잘 짜서 CPU가 항상 처리할 일을 만들어줌
- 처리해야할 job 들이 memory에 상주하게됨
- I/O 처럼 기다려야할 상황이 있을 때 OS는 다른 작업으로 switch
- 여러 프로그램이 한 꺼번에 메모리에 올라가서 CPU가 여러 프로그램을 처리할 수 있음

timesharing

- multitasking
- multiprogramming의 논리적 확장 개념
- CPU가 현재 실행 중인 여러 작업들을 계속, 빠르게 switch
- 여러 프로그램이 동시에 실행되는 것 처럼 보임

잠깐 용어 정리

- job : 아직 디스크에 있는 작업이면 job
- process : 메모리에서 실행 중인 프로그램. job을 메모리에 올리면 process.
- job scheduling : job을 메모리에 올려 실행 가능한 상태로 준비하는 과정
- cpu scheduling : cpu resource를 process들에게 효율적으로 분배하는 과정
- swapping : 메모리에 프로세스가 다 안 들어가면 일부를 넣었다 빼면서 swapping
- virtual memory : 프로세스가 전부 메모리에 load 되지 않아도 실행되게끔 만들어줌

Operating System Operations

- OS는 interrupt-driven
- OS와 user는 hw, sw resource를 공유하기 때문에 하나의 user program에 하나의 problem만 발생시키도록 만들어야 함
- 무한 루프에 빠지거나 A 프로세스가 B 프로세스를 수정할 수 도 있음
- 이를 방지하기 위해 dual-mode/timer, multi-mode 사용

Dual-Mode Operation

- mode bit 사용
- user를 대신하는 작업과 OS를 대신하는 작업을 구분하기 위해 필요
- kernel mode vs user mod
- hw의 도움이 필요. hw만이 mode bit를 구현할 수 있음
- 몇몇 instruction은 커널 모드에서만 실행 가능
  - 타이머 지정, 인터럽트 끄기, I/O 장치 접근, trap instruction 생성, ...
  - 클럭 일기는 유저 모드에서도 실행 가능

Timer

왜 사용하는가?

- 무한 루프, 프로세스의 자원 독점(hogging, 돼지)을 방지
- OS는 할당된 시간이 지나면 프로그램 종료시킴
- 프로세스를 스케줄링하기 전에 타이머를 설정해서 프로세스가 아직 끝나지 않았더라도 시간이 지나면 제어권을 다시 가질 수 있게 만듦 (자원 분배)

타이머 작동 원리

- 특정 시간이 지나면 timer가 interrupt를 발생
- OS가 counter를 --;
- counter가 0이 되면 interrupt 발생

OS의 역할

1. process Management

program vs process

- 실행 중인 prorgram = process
- process는 resoure (CPU time, memory, files, IO devices) 가 필요

process management?

- CPU에서 thread의 process를 scheduling
- process 생성, 제거, 중단, 계속
- process synchronization, communication 메커니즘 제공

2. Memory Management

- 무엇을, 언제 메모리에 저장할지 관리
- 모든 instruction은 실행시키기 위해 memory에 존재해야 함
- 모든 data는 processing 전/후 메모리에 존재해야 함
- 멀쩡 프로그램은 CPU utilization과 유지에 대한 컴퓨터의 response 속도를 높이기 위해 메모리에 계속 뇌둬야 함

some activities for memory management

- 현재 어느 부분이 누구에 의해 사용되는지 감시
- 어떤 process와 data를 메모리에 넣거나 뺄지 결정
- 필요 시 메모리 allocate, deallocate

3. Storage Management

- OS는 저장소의 uniform, local view를 제공
- physical한 저장 장치를 file이라는 logical 저장 단위로 abstraction
- physical 매체를 file로 mapping한 후, file을 통해 저장장치에 접근

some activities for storage management

- file-system management
  - 누가 뭘 접근할 수 있는지 결정
  - file과 directory 생성, 제거
- mass-storage management
  - 메인메모리보다 큰 데이터 저장, 긴 시간 동안 유지시켜야 하는 데이터 저장
  - 빈 공간 관리, 저장소 할당, 디스크 스케줄링
- caching과 I/O 서브시스템

Protection과 Security

Protection

- OS가 정의한 리소스에 대한 process, user의 접근을 관리
- file access contorl - UID, GID (리눅스? 윈도우와 생각하면 안 될 듯)

Security

- 외부, 내부 공격에 대한 시스템의 방어
- User Authenticataion (user name/password)

Design Goals

Abstraction

- 시스템을 편리하고 쓰기 쉽게 만들 것

High Performance

- OS의 오버헤드를 최소화시킬 것
- OS는 과도한 오버헤드 없이 virtualization을 제공해야한다.

Protection

- Isolation : 한 프로그램의 비정상 행동은 다른 프로그램과 OS에 영향을 미쳐선 안된다.

High Degree of Reliability

- OS는 쉬지 않고 실행되어야 한다.

Other issues

- 에너지 효율, 보안, 이식성, ...