컴퓨터시스템의 구조

생성일	@2022년 3월 19일 오후 1:55
태그	

운영체제란 무엇인가?

운영체제(Operating System, OS)

- 컴퓨터 하드웨어 바로 위에 설치되어 사용자 및 다른 모든 소프트웨어와 하드웨어를 연결하는 소프트웨어 계층
- 협의의 운영체제(커널)
  - 운영체제의 핵심 부분으로 메모리에 상주하는 부분
- 광의의 운영체제
  - 커널 뿐 아니라 각종 주변 시스템 유틸리티를 포함한 개념

목적

- 컴퓨터 시스템을 편리하게 사용할 수 있는 환경을 제공
- 컴퓨터 시스템의 자원을 효율적으로 관리
  - 프로세서, 기억장치, 입출력장치 등의 효율적 관리
    - 사용자간의 형평성 있는 자원 분배
    - 주어진 자원으로 최대한의 성능을 내도록
  - 사용자 및 운영체제 자신의 보호
  - 프로세스, 파일, 메세지 등을 관리

분류

- 동시 작업 가능 여부
  - 단일 작업: 한 번에 하나의 작업만 처리 ex) MS-DOS
  - 다중 작업: 동시에 처리 가능 ex) UNIX, MS Windows, Apple MacOS
- 사용자의 수
  - 단일 사용자: MS-DOS, MS Windows
  - 다중 사용자: UNIX, NT Server
- 처리 방식
  - 일괄 처리 : 작업 요청의 일정량을 모아서 한꺼번에 처리하여 작업이 종료될 때까지 기다려야 함 ex) 초기 Punch Card 처리 시스템
  - 시분할(time sharing) : 여러 작업을 수행할 때 컴퓨터 처리 능력을 일정한 시간 단위로 분할하여 사용하는 interactive한 방식 ex) UNIX(일괄 처리 시스템보다 짧은 응답시간)
  - 실시간(Realtime OS): 정해진 시간 안에 어떠한 일이 반드시 종료됨이 보장되어야 하는 실시간 시스템을 위한 OS ex) 원자로,공장 제어, 미사일 제어, 반도체 장비 등
    - Hard realtime system, Soft realtime system

<참고> Multitasking, Multiprogramming, Time sharing, Multiprocessing 구분

- 이 용어들은 컴퓨터에서 여러 작업을 동시에 수행하는 것을 뜻함
- Multiprogramming은 여러 프로그램이 메모리에 올라가 있음을 강조

- Time sharing은 CPU의 시간을 분할하여 나누어 쓴다는 의미를 강조
- Multiprocessor는 하나의 컴퓨터에 CPU가 여러개를 붙어있음을 의미

운영 체제의 예

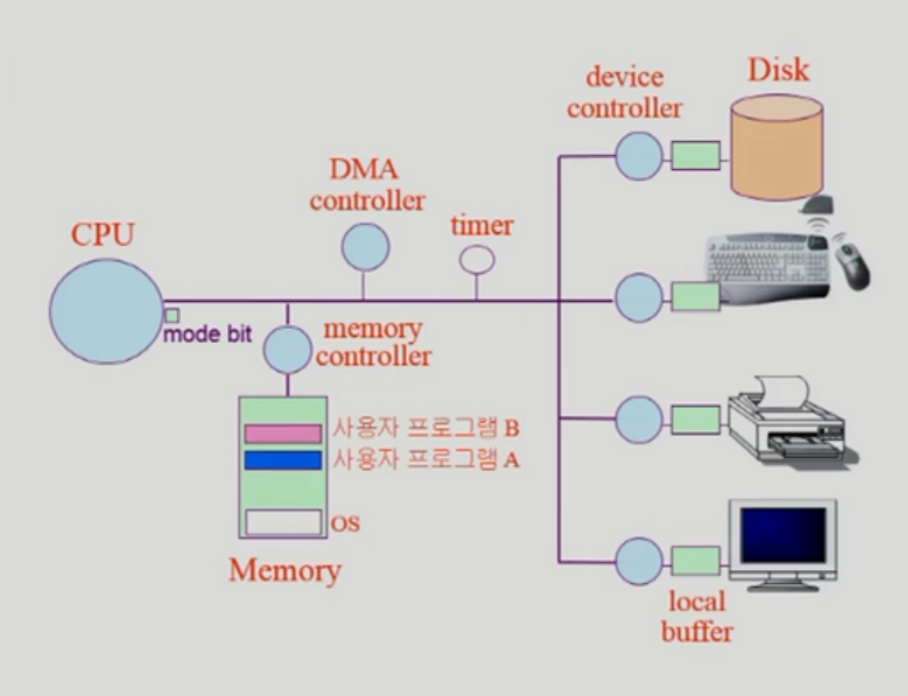
- 유닉스(UNIX)
  - 코드 대부분을 C언어로 작성
  - 높은 이식성, 최소한의 커널 구조, 확장 용이, 소스코드 공개, 프로그램 개발 용이
  - 다양한 버전 존재 (System V, FreeBSD, SunOS, Solaris, Linux)
- DOS
  - MS사가 IBM-PC를 위해 개발
  - 단일 사용자용 운영체제, 메모리 관리능력 한계
- MS Windows
  - MS사의 다중 작업용 GUI 기반 운영체제
  - Plug and Play, 네트워크 환경 강화, 호환성 제공, 불안정성, 풍부한 자원 SW
- Handheld device를 위한 OS
  - PalmOS, Pocket PC, Tiny OS

구조

- CPU스케줄링: 누구한테 CPU를 줄까?
- 메모리 관리: 한정된 메모리를 어떻게 쪼개어 쓰지?
- 파일 관리: 디스크에 파일을 어떻게 보관하지?
- 입출력 관리: 각기 다른 입출력장치와 컴퓨터 간에 어떻게 정보를 주고 받게 하지?
- 프로세스 관리: 프로세스의 생성과 삭제, 자원 할당 및 변환, 프로세스 간 협력
- 보호 시스템, 네트워킹, 명령어 해석기(Command line interpreter 등)

OS 개발자의 관점에서 바라보자! : 대부분의 알고리즘은 OS 프로그램 자체의 내용이므로 이를 이해하도록 노력하자

컴퓨터 시스템의 구조



- Mode bit
  - 사용자 프로그램의 잘못된 수행으로 다른 프로그램 및 운영체제에 피해가 가지 않도록 하기 위한 보호 장치가 필요 → Mode bit
    - 1 - 사용자 모드 (사용자 프로그램 수행)
    - 0 - 모니터 모드 (OS 코드 수행)
  - 보안을 해칠 수 있는 중요한 명령어는 모니터 모드에서만 수행 가능한 '특권명령'으로 규정
  - Interrupt나 Exception 발생 시 하드웨어가 mode bit을 0으로 바꿈
  - 사용자 프로그램에게 CPU를 넘기기 전에 mode bit을 1로 세팅

  → 모니터 모드 (=커널 모드, 시스템 모드)
- Timer
  - 정해진 시간이 흐른 뒤 운영체제에게 제어권이 넘어가도록 인터럽트를 발생시킴
  - 타이머는 매 클럭 틱 때마다 1씩 감소
  - 타이머 값이 0이 되면 타이머 인터럽트 발생
  - CPU를 특정 프로그램이 독점하는 것으로부터 보호
  - 타이머는 time sharing을 구현하기 위해 널리 이용됨
  - 타이머는 현재 시간을 계산하기 위해서도 사용
- 시스템 콜
  - 사용자 프로그램이 운영체제의 서비스를 받기 위해 커널 함수를 호출하는 것

Interrupt

- 인터럽트 당한 시점의 레지스터와 program counter를 save한 후 CPU의 제어를 인터럽트 처리 루틴에 넘긴다.
  - Interrupt(하드웨어 인터럽트): 하드웨어가 발생시킨 인터럽트
  - Trap(소프트웨어 인터럽트)
    - Exception: 프로그램이 오류를 범한 경우

    - System call: 프로그램이 커널 함수를 호출하는 경우
- 현대의 운영체제는 인터럽트에 의해 구동
- 인터럽트 벡터: 해당 인터럽트의 처리 루틴 주소를 가지고 있음
- 인터럽트 처리 루틴: 해당 인터럽트를 처리하는 커널 함수

동기식 입출력과 비동기식 입출력

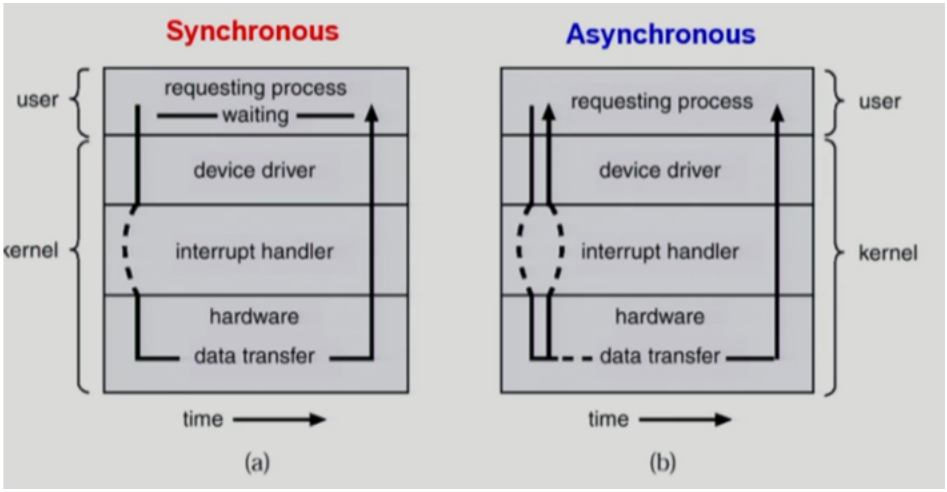
두 입출력 모두 완료는 인터럽트로 알려준다

동기식 입출력 (synchronous I/O)

- I/O 요청 후 입출력 작업이 완료된 후에야 제어가 사용자 프로그램에 넘어감
- 구현 방법 1
    - I/O가 끝날 때까지 CPU를 낭비시킴
    - 매 시점 하나의 I/O만 일어날 수 있음
- 구현 방법 2
    - I/O가 완료될 때까지 해당 프로그램에게서 CPU를 빼앗음
    - I/O 처리를 기다리는 줄에 그 프로그램을 줄 세움
    - 다른 프로그램에게 CPU를 줌

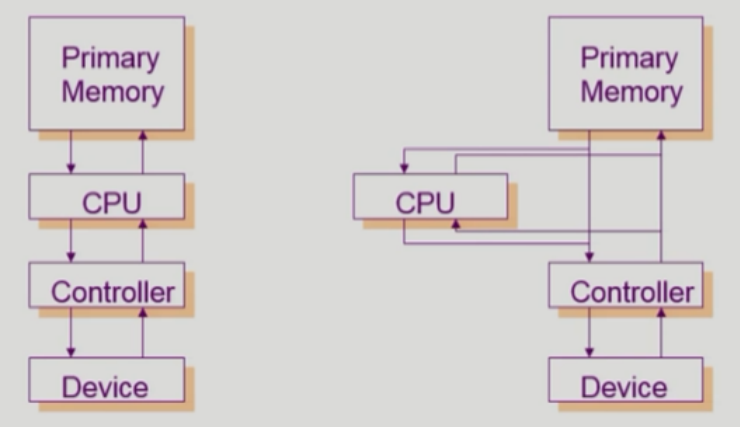
비동기식 입출력 (asynchronous I/O)

- I/O가 시작된 후 입출력 작업이 끝나기를 기다릿 않고 제어가 사용자 프로그램에 즉시 넘어감



DMA (Direct Memory Access)

- 빠른 입출력 장치를 메모리에 가까운 속도로 처리하기 위해 사용
- CPU의 중재 없이 device controller가 device의 buffer storage의 내용을 메모리에 block 단위로 직접 전송
- 바이트 단위가 아니라 block 단위의 인터럽트를 발생



저장장치 계층구조

- Caching: copying information into faster storage system