2. Operating System Overview

Operating System Structure

Multiprogramming ⇒ 이거 왜 나옴?

long I/O wait같은 슬픈 상황을 피하기 위해서 기본적으로 CPU는 쉬지 않게 해야 하는데 저 상황은 CPU가 멈춰버리는 슬픈 상황이 발생함

그래서 해야될 작업들을 조절해야 되는데 현재 해야될 일을 하면서도 다음에 할 일을 결정해야됨

Timesharing(multitasking)

이거는 뒤에 있는 멀티태스킹이 좀더 익숙할 텐데 보통 스타크래프트 해보면 하나만 몰두하는게 아니라 모든 작업을 조금씩 해서 균등하게 진행

그래서 중요해지는게 멀티태스킹할 때 프로그램 간의 전환 속도가 중요해짐

생각해보면 당연한게 프로그램을 진행하는 시간보다 전환하는 시간이 더 오래걸리면 그냥 하나만 하는게 낫지 뭐하러 여러개 동시에 같이해

프로세스는 좀 모호한데, 간단하게 보자면 프로그램을 담는 그릇 같은 거라고 생각하면 편해 보통 우리 Java 배울 때 heap, static(이거 있나? ㅋㅋ), stack, text 이런 것들 저장하는 거

CPU scheduling은 뭐 나중에 나오겠지만 기본적으로 자주하는 것들은 빨리빨리 해주는게 좋겠지 예를 들어 키보드 클릭 되게 자주하는데 이게 빨리 못하게 되면 사용자 속이 터지겠지

Virtual memory는 모든 프로세스가 CPU가 인식하는 범위로 올렸다고 가정하는 방법인데 나중에 자세히 할듯

Operating System Operations

아 키보드 interrupt가 바로 나오긴 하는데 이거를 interrupt로 처리해야 성능이 좋아 왜냐하면 CPU가 사용자가 키보드를 치는지 안 치는지 기다리면서 확인할 수가 없으니까 (확인하는 것도 CPU 자원을 먹으니까)

그래서 키보드를 누르는 순간 interrupt를 보내서 CPU가 키보드 입력을 먼저 처리하도록 하고 다시 원래의 작업으로 돌아가도록 하는거지

Exception or trap은 무조건 최우선적으로 처리. 이거 해결 안 했다가 프로그램 터지면 어떡하려고? 근데 이거는 Java에서 Exception handling 해봤으니까 그런 상황 생각해보면 될듯?

아 이제 Dual-mode인데 기본적으로 하드웨어에 접근할 수 있는건 OS만 가능

상황을 생각해보면 키보드의 입출력을 처리해주는 건 OS가 해주지 사람이 직접하진 않잖아(뭐 SSD나 HDD로 예를 들어봐도 우리가 거기서 데이터 직접 찾진 않잖아..)

그래서 이런 OS가 처리해 주는 작업들을 kernel mode에서 처리 가능

그러면 PC는 사용자랑 OS를 어떻게 구분할까? ⇒ bit로 구분 가능

주로 kernel레벨이 3 User level이 1임 ⇒ 0x11 이랑 AND 연산해버리면 그 값이 나와서 누군지 바로 알 수 있어

아 C가 왜 저수준의 언어냐?(물론 책에서는 고수준이긴 한데…) 기본적으로 시스템콜들이 C언어의 인터페이스 처럼 되어있어 대표적인게 open, write, read …

⇒ 이걸 잘 wrapping해서 우리가 사용하는게 printf scanf 이런것들

Q. 그럼 C언어에서 open, write, read 사용할 수 있나요?

▼ 답

ㅔ

Process Management

Process

1. 메모리에 올라온 프로그램을 프로세스라고 합니다
2. 이게 시스템 작업의 단위인데 이제 이거 개무시하고

→ 그냥 작업을 관리하기 위한 자료구조를 process라고 생각하면 편함

그러면 memory에 올라왔으니까 나중에 disk로 내려보내주는 것도 해줘야 겠지? ⇒ 그게 process termination

Single-thread ⇒ 하나의 PC(Program Counter)

multi-thread ⇒ 쓰레드당 하나의 PC(Program Counter)

Process Management Activities

process synchronization: 나중에 주구장창 할 예정…

process communication: 이거 솔직히 잘 모르겠는데, 이번 기회에 잘 정리해보죠 ^~^

deadlock handling: 이거 서로가 서로를 기다리는 상황에 발생하는데 요즘 운영체제에서는 발생 안 함 설령 발생한다해도 운영체제가 프로세스를 강제로 종료시켜버리고 다시 실행해버려(deadlock을 탐지하는 거보다 그냥 껐다 키는게 자원을 덜 잡아먹어서 이득)

Memory Management

말 그대로 메모리 영역을 관리하는 방법

fragmentation을 메꿀건지, 어떤 프로세스를 disk로 내려보낼건지, 아니면 메모리에 어떤 프로세스를 올릴 건지 이런것들을 결정하는 방법

Storage Management

File system이 나오는데 기본적으로 file은 physical 한 주소는 아님

막말로 디렉토리, 파일 이라고 하는데 이거 주소 어딘지 아는 사람?

OS는 physical 하게 주소를 저장하도록 하고 사용자에게 file의 형태로 이를 보여줘

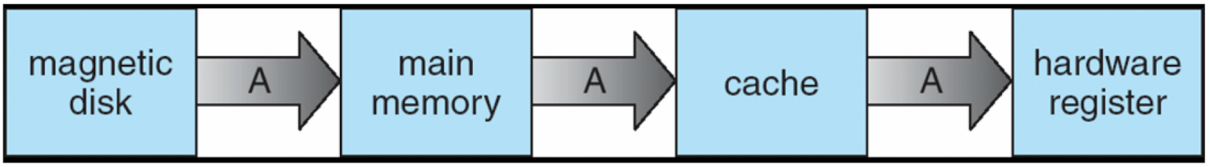
이게 수업 때 배웠는데 디렉토리 == file

물론 UNIX 기준이고 윈도우는 모름…

Migration of Integer A from Disk to Register

이전에 정리했던 컴퓨터 구조

레지스터 ⇒ 캐시 ⇒ 메모리 ⇒ 디스크



register는 되도록이면 캐시에 데이터가 있는 걸 기대할거고, 반대로 캐시는 메모리에 데이터가 있는 것을 기대

전에도 나왔지만 Multiprocessor 환경에서는 cache coherence problem을 해결해줄 수 있어야 해

(물론 난 모름… 아는 사람?)

I/O Subsystem

여기서는 spooling 정도만 알아가면 될듯?

buffering ⇒ 데이터가 전송될 때 임시적으로 저장하는 공간

caching ⇒ 프로그램 성능을 향상하기 위해 사용하는 거

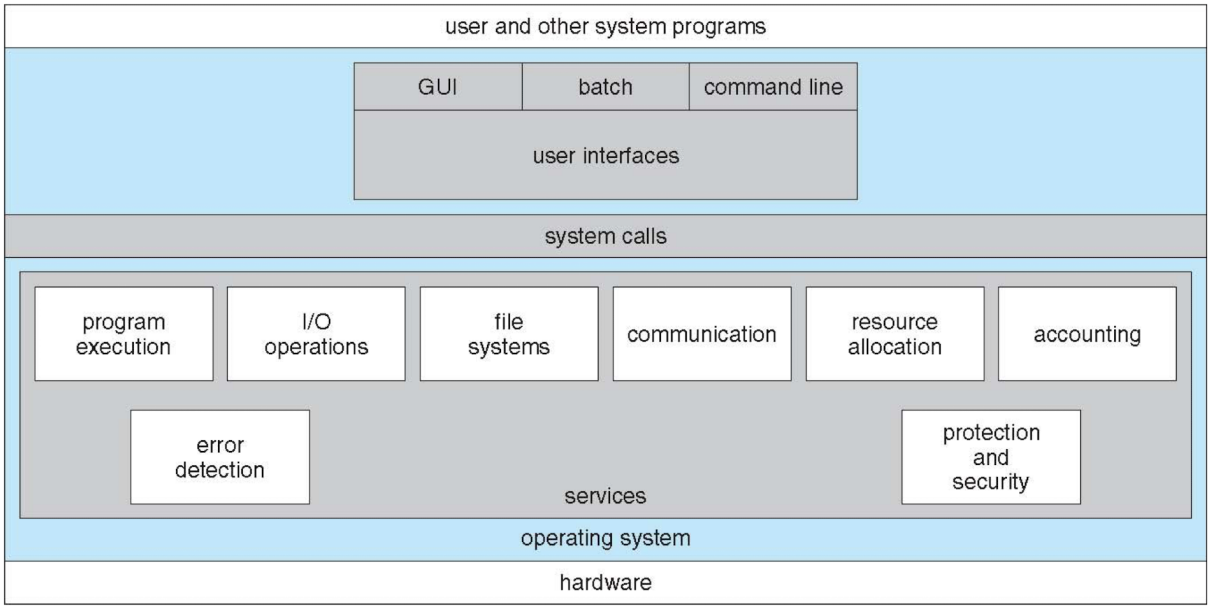
spooling

1. 이제는 문서 작업할 때 spool에다가 열심히 등록해 (이걸 OS가 날라)
2. 그러면 spool에 생성된 작업물을 다른 작업에서 읽어가
3. 이러면 장점이 뭐냐? 기다리는 시간이 확 줄어들어
4. 기본적으로 키보드 입력은 I/O 작업이라 시간이 걸리는데 이걸 spool에 써놓고 다른 작업을 하고 있으면 I/O를 처리하는 다른 작업이 이걸 읽고 이어서 작업
5. 또 이렇게 하다보니 동시에 문서작업을 할 수 있는 장점이 생겨

그럼 여기서 사용자가 쓰는 작업을 spool에 등록하는데 이거 가능하냐?

⇒ 원칙적으로는 불가능 spool은 OS권한이고 작업은 user 권한인데 접근이 불가능해 그래서 privilege escalation을 통해 일시적으로 사용자의 권한을 OS권한으로 상승시켜

A View of Operating System Services



System calls

아까 위에서 적고 와서 넘어갈게요

그러면 일반 user들이 쓰는 라이브러리랑 system call의 차이는 무엇일까요?

⇒ 솔직히 난 잘 모르겠음… 그냥 둘이 또이또이 한거 같은데….

예전에 교오수님한테 듣기로는 system call이 많이 불린애가 라이브러리라고…