<http://baekhorang.tistory.com/entry/%EC%A0%84%EC%82%B0%ED%95%99%EA%B0%9C%EB%A1%A005-%EC%9A%B4%EC%98%81%EC%B2%B4%EC%A0%9COperating-System-%EA%B0%9C%EB%85%90-%EB%B0%8F-%EC%9A%A9%EC%96%B4%EC%A0%95%EB%A6%AC>

프로세스의 메모리 영역 : heap, stack, static, code

프로세스 : 실행중인 프로그램 / IPC로 프로세스간 통신

쓰레드 : 프로세스를 구성하고 실행하는 흐름 / stack 빼고 공유함

code영역을 공유해서 쓰레드가 프로세스의 함수 호출가능

별도의 장치 없이 쓰레드끼리 통신가능

Time, resource sharing, scalability

Thread mode

Kernel : 커널이 프로세스 내부의 스레드 관리, 생성시 커널에게 요청, non block

User : user level에서 관리됨, 커널은 1개로 인식, 하나 블락되면 다 블락됨

쓰레드와 프로세스 모두 스케줄링의 대상이라 컨텍스트 스위칭 필요

But 오버헤드가 덜함

OS : 하드웨어 리소스를 유저가 쉽게 사용할 수 있게 해줌 (어플리케이션 프로그램과 하드웨어 사이에 존재)

Os는 interrupt driven

* 프로세스 관리 : 프로세스 생성 제거 정지, 프로세스 싱크와 통신
* 메모리 관리 : 어떤 메모리가 사용되는지 어떤 프로세스를 메모리에 올릴지 얼마나 할당할지 결정
* 저장공간 관리 : 디스크 스케줄링, 파일관리 등
* Ui 제공

Device controller – cpu : 인터럽트를 통해 통신

인터럽트가 오면 cpu는 하던거 멈추고 인터럽트 처리함

인터럽트가 발생하면

* Os는 cpu상태를 레지스터에 저장
* 어떤 인터럽트인지 판단해서 처리 (인터럽트 벡터를 보고 해당 인터럽트에 대한 서비스 루틴을 실행)

Trap : 소프트웨어 인터럽트 / 에러나 유저의 요청에 의해 발생

인터럽트 : 프로그램 외부의 이벤트 <-> 익셉션 : 프로그램 내부 이벤트

Dual mode operation : 유저모드(1)와 커널모드(0)로 나눠서 한 프로그램이 다른 프로그램에 영향을 못 끼치게 함

Mode bit을 통해 구분하고 privileged instruction 은 커널모드에서만 실행 할 수 있는거

만약 유저모드에서 실행하려하면 trap발생시켜서 커널에서 처리

유저모드 <-> 커널모드 이동은 system call을 통해서

운영체제가 지원하는 서비스에 대한 프로그래밍 인터페이스 but 비쌈

Context switch : cpu사용을 다른 프로세스에게 전환시키는 작업

Context 정보는 pcb에 저장됨

Cpu 사용량을 최대로 하기위한 스케줄링

오버헤드 : 스위칭 할 때 cpu가 놈 / 컨텍스트 스위치 시간은 하드웨어에 의존적임(느림)

Time for context switching dependent on hw

한번에 하나의 프로세스만, dispatcher가 할당

Job queue : 모든 프로세스

Ready queue : ready상태

스케줄러

롱텀(잡 스케줄러) : job queue에서 어떤놈을 메모리로 올릴지 결정 (실행 프로세스 수도 결정<-안정성)

숏텀(cpu 스케줄러) : memory에 올라온 얘들 중 누굴 cpu에 올릴지 결정

Io bound와 cpu bound의 조화가 필요 : 한쪽 queue가 안 비게끔

밋텀 스케줄러 : 스왑핑, 한계를 초과했을 때 실행 보류시키는 기능 (suspended)

임계영역 : 한순간에 하나의 접근만 허용하는 공유 리소스에 접근하는 코드 블럭

동기화 : 임계영역에 접근하는 쓰레드의 순서가 잘 지켜지는

동기화 방법 : critical section / mutex / semaphore

임계영역 기반 동기화 : 다른 쓰레드가 현재 사용 중 이라는 것을 명시

뮤텍스 : mutex를 얻은 얘만 접근가능한

세마포어 : 뮤텍스도 세마포어의 하나 / 세마포어는 여러 개의 임계영역에 대한 것