3️⃣ 프로세스 관리

태그	1주차	CSS

✨ 프로그램 실행

📌 과정

- File System의 실행 파일
- Virtual memory: 그 프로세스만의 독자적인 메모리 주소 공간
- 논리적 주소→ 물리적 주소 변환
- 당장 필요한 부분은 메모리(물리적)에 올라가서 Process가 됨
    - 그렇지 않은 부분은 Swap area

📌 Virtual Memory

✅ code

✅ data

- 프로그램 실행하다가 메모리 데이터 쓰는 경우
- 전역 변수

✅ stack

- 지역 변수

📌 커널의 주소 공간

- 운영체제: 시스템 자원을 효율적으로 관리

✅ code

- 시스템콜, 인터럽트 처리, 자원 관리, 서비스 제공 코드

✅ data

- 모든 hardware들을 관리하기 위한 자료구조
- 실행 중인 모든 process들을 관리하기 위한 자료 구조(Process Control Block)

✅ stack

- 실행 중이던 프로세스의 커널 스택

📌 함수

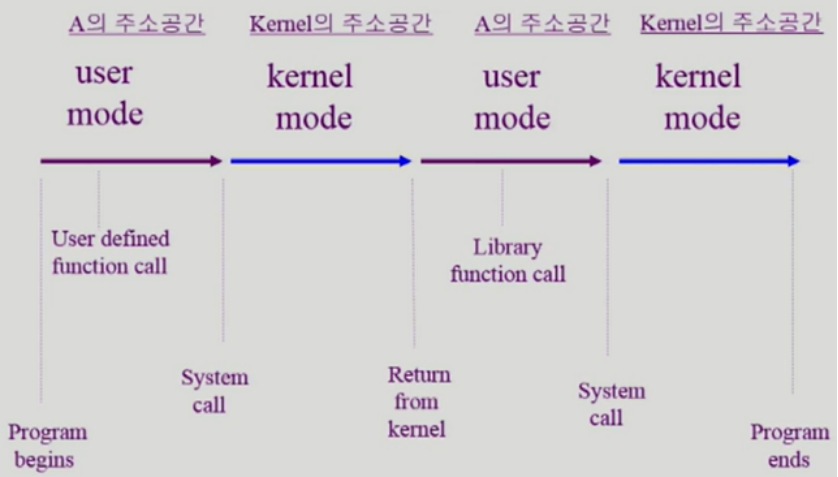
✅ 사용자 정의 함수

✅ 라이브러리 함수

✅ 커널 함수

- 시스템 콜

📌 예시



✨ 프로세스

📌 개념

- 실행 중인 프로그램

✅ 프로세스의 문맥(context)

- CPU 수행 상태를 나타내는 하드웨어 문맥
  - Program Counter, 각종 register
- 프로세스의 주소 공간
  - code, data, stack
- 프로세스 관련 커널 자료 구조
  - PCB, Kernel stack

📌 상태

- 운영체제는 PCB를 통해서 상태를 관리

✅ Running

✅ Ready

- 나머지 다른 조건 다 만족하고 cpu 쓰겠다고 기다리는 상태
- ready queue

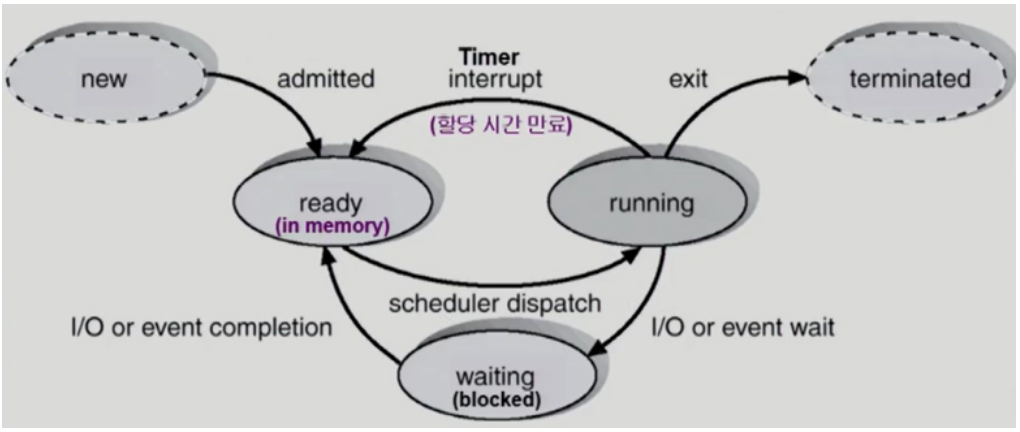
✅ Blocked(waiting)

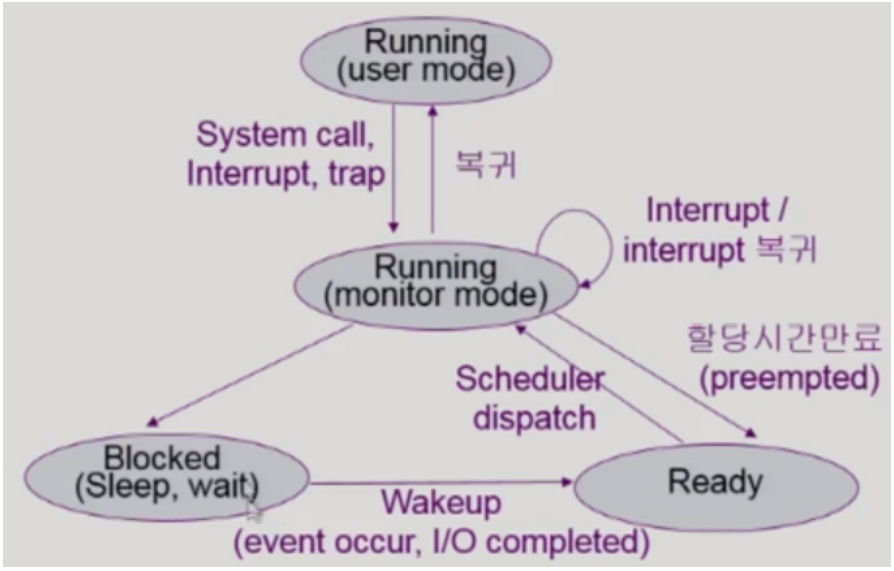
- cpu 당장 얻어봐야 무의미한 상태
- 예) 디스크에서 file을 읽어와야 하는 경우, 공유 데이터
- i/o queue
- 일을 다하고 cpu에 interrupt를 걸면 ready queue로 이동

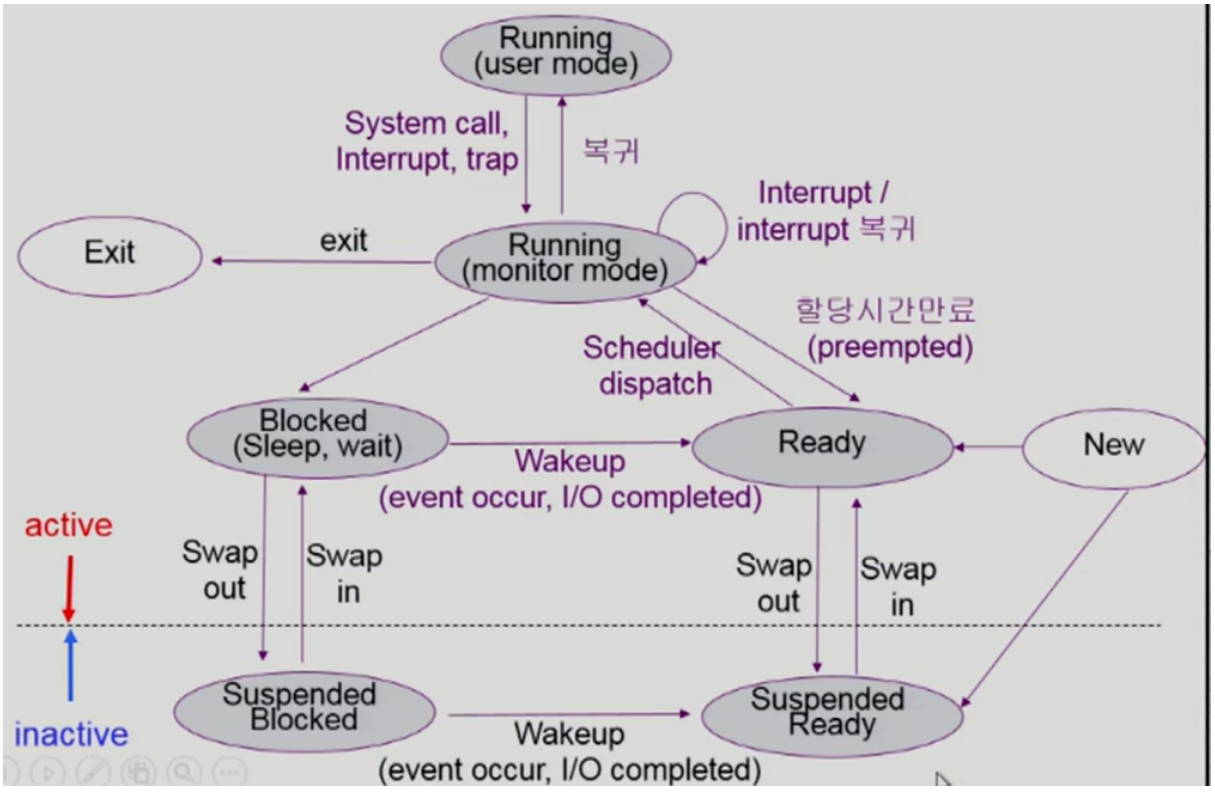
✅ Suspended(stopped)

- 중기 스케줄러에 의해 메모리에서 쫓겨난 상태
- 디스크에 swap out
- 외부에서 resume해 주어야 active

📌 상태도


new
admitted
Timer
interrupt
(할당 시간 만료)
exit
terminated
ready
(in memory)
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting
(blocked)


Running
(user mode)
System call,
Interrupt, trap
복귀
Interrupt /
interrupt 복귀
Running
(monitor mode)
할당시간만료
(preempted)
Scheduler
dispatch
Blocked
(Sleep, wait)
Ready
Wakeup
(event occur, I/O completed)


Running
(user mode)
System call,
Interrupt, trap
복귀
Interrupt /
interrupt 복귀
Exit
exit
Running
(monitor mode)
할당시간만료
(preempted)
Scheduler
dispatch
Blocked
(Sleep, wait)
Ready
New
Wakeup
(event occur, I/O completed)
active
Swap
out
Swap
in
Swap
out
Swap
in
inactive
Suspended
Blocked
Wakeup
Suspended
Ready
(event occur, I/O completed)

📌 Process Control Block (PCB)

- 구조체

✅ OS가 관리상 사용하는 정보

- 프로세스 상태, PID, 스케줄링 정보, 우선순위

✅ CPU 수행 관련 하드웨어

- Program counter, registers

✅ 메모리 관련

- code, data, stack의 위치 정보

✅ 파일 관련

✨ 문맥 교환 (Context Switch)

- CPU를 다른 프로세스로 넘겨주는 과정
- 그 전 프로세스 PCB에 상태를 저장 - save
- CPU는 새 프로세스 상태를 읽음 - load
- kernel의 개입이 있음

1. timer interrupt
2. 오래 걸리는 I/O 요청 system call

✨ 프로세스 스케줄링 큐

- Job queue: 모든 프로세스
- Ready queue
- Device Queues

✨ 스케줄러

✅ Long-term(job) scheduler

- 어떤 프로세스를 레디큐로 보낼지, 메모리에 올릴지
- new →admitted→ ready
- 현대 os에는 없음 (바로 레디)

✅ Short-term(CPU) scheduler

- running 시킬지, cpu를 줄지
- ready → running

✅ Medium-term scheduler(Swapper)

- 메모리가 부족할 때 특정 프로세스를 쫓아냄, 메모리를 뺏음
- degree of mulitprogramming(메모리에 올라와있는 프로그램 수)를 관리

✨ 스레드

- 프로세스 중 cpu 수행 단위
- 프로세스는 하나
- 프로세스의 어느 부분을 수행하고 있는가
- 스레드 간 이동은 효율적

✅ 구성

- program counter

- register set
- stack space

✅ thread끼리 공유하는 부분(=task)

- code
- data
- os resources

✅ 장점

- 속도, 응답성이 빠르다
- 성능
- 자원 공유
- 병렬 처리 가능(멀티 프로세서 환경)

✅ 구현

- kernel threads
  - 운영체제가 알게 구현하는 것
- user threads
  - 운영체제가 볼 때 하나의 스레드로 보이게
  - 프로세스 내부에서 처리

✨ 프로세스 생성

✅ 특징

- 부모 프로세스가 자식 프로세스(모든 게 똑같음)를 생성
- fork(): os에게 만들어 달라고 요청
- 트리 형태의 계층 구조 형성
- 부모와 자식 간 자원 공유
  - 모든 자원
  - 일부
  - 전혀 x
- 수행(Execution)
  - 부모와 자식 공존하며 수행되는 모델
  - 자식이 종료(terminate)될 때까지 부모가 기다리는(wait/blocked) 모델

✅ 생성

- 자식은 부모의 주소 공간을 복사 (binary and os data)
- 그 공간에 새로운 프로그램 올림

✅ fork()

- 복제 생성
- return 값 - 부모: 자식 pid > 0, 자식: 0 으로 구분

✅ exec()

- 새로운 프로그램으로 덮어 씌어지는, 아래 부분 실행 안 됨
- 자식한테 다른 프로그램 실행하게 하고 싶을 때

✅ 종료

- exit(): 운영체제에 종료를 알려주는 함수
- abort(): 부모가 자식을 강제로 죽임