[week2] Quiz

[Chapter3] Process

프로세스의 개념

> 프로세스: ( )

프로세스의 문맥(context)

: ( )

- Program Counter: ( )
- 프로세스의 주소 공간
  - code, data, stack 에 어떤 내용들이 들어있는가
- 프로세스 관련 커널 자료 구조
  - PCB(Process Control Block)
  - Kernel Stack
    - 프로세스가 시스템콜을 실행한경우 Kernel에서 함수호출이 이루어지고 Kernel Stack에 프로세스별로 스택을 별도로 두고 있다.

시분할 프로그래밍이기때문에 현재 문맥을 제대로 알고있어야 다음 시점부터 instruction을 실행할 수 있다.

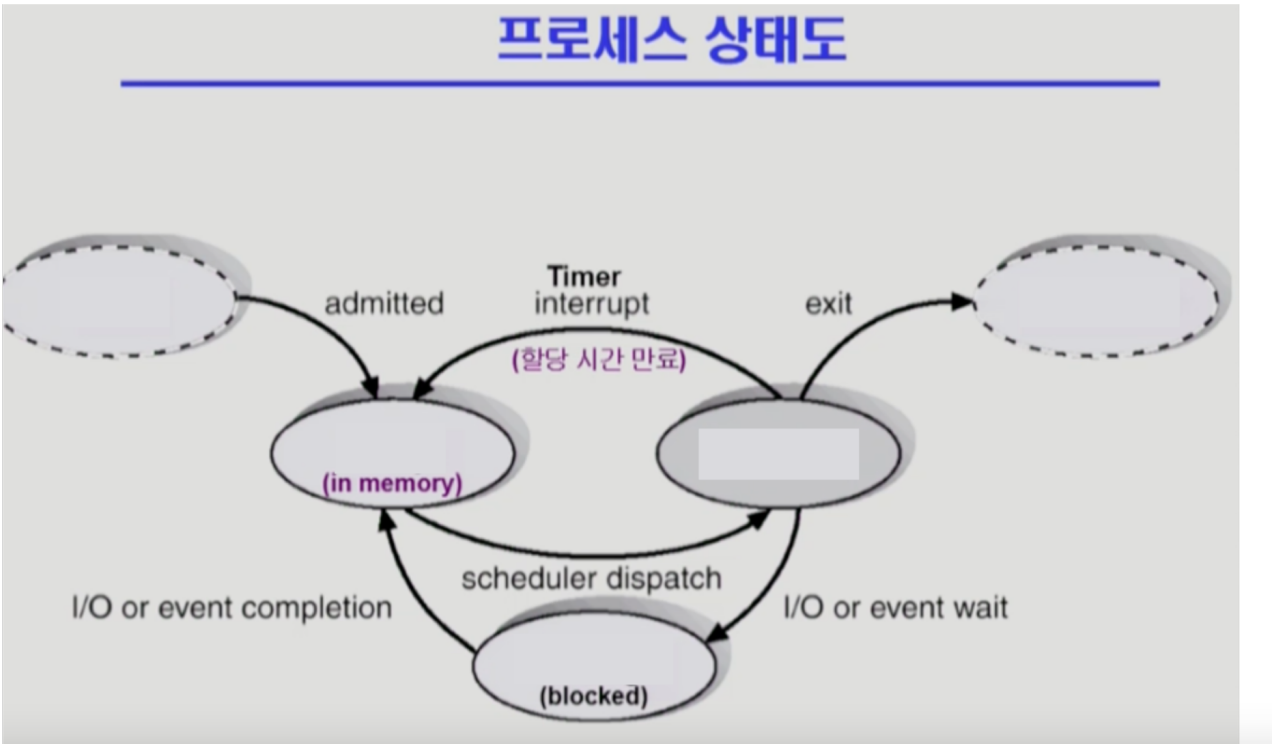
프로세스의 상태

프로세스는 상태(state)기 변경되며 수행된다.

- Running
  - ( )
- Ready
  - ( )
- Blocked (wait, sleep)
  - ( )
- New: 프로세스가 생성중인 상태
- Terminated: 수행이 끝난 상태

프로세스 상태도

(빈칸 채워넣기)



PCB

PCB는 무엇인가

: ( )

문맥교환(Context Switch)

- CPU를 한 프로세스에서 다른 프로세스로 넘겨주는 과정
- CPU가 다른 프로세스에게 넘어갈때 운영체제는 아래와 같은 일을 함
  - CPU를 내어주는 프로세스의 상태를 그 프로세스의 PCB에 저장
  - CPU를 새롭게 얻는 프로세스의 상태를 PCB에서 읽어옴
- **System call이나 Interrupt 발생시 반드시 context switch가 일어나는 것은 아님**
  - 사용자 프로세스 A ------**kernel mode** (interrupt 또는 system call) ----------> 사용자 프로세스 A

        - Context switch 아님
    - 사용자 프로세스 A ----- **kernel mode** (timer interrupt 또는 IO 요청으로 blocked 상태)------> 사용자 프로세스 B
        - Context switch 맞음

프로세스 스케줄링을 위한 큐

- Job queue
    - 현재 시스템 내에 있는 모든 프로세스의 집합
- Ready queue
    - 현재 Ready 상태에 있는 프로세스의 집합
- Device queue
    - IO 디바이스의 처리를 기다리는 프로세스의 집합

스케줄러

- Long-term scheduler (Job scheduler)
    - 시작 프로세스(new 상태의 프로세스)가 메모리에 올라오는걸 관리
    - 시작 프로세스 중 어떤 것들을 ready queue로 보낼지 결정
    - degree of multi-programming을 제어
        - multi-programming:

          2개 이상의 프로그램을 주기억장치에 기억시키고, 중앙처리장치를 번갈아 사용하는 처리기법
    - time sharing system에는 보통 장기 스케줄러가 없음 (**무조건 ready**)

- Short-term scheduler (CPU scheduler)
    - 짧은 시간 안에 스케줄이 이루어져야함

      --> 충분히 빨라야함 (millisecond) 단위
    - 어떤 프로세스를 다음번에 running시킬지 결정

      --> 프로세스에 CPU를 주는 문제
- Medium-term scheduler (Swapper)
    - 여유공간 마련을 위해 프로세스를 통째로 메모리에서 디스크로 쫓아냄
    - 프로세스에게서 memory를 빼앗는 문제
    - **degree of multi-programming을 제어**

프로세스의 상태 (시분할 시스템에서)

- Running
- Ready
- Blocked
- Suspended (stopped)
    - ( )

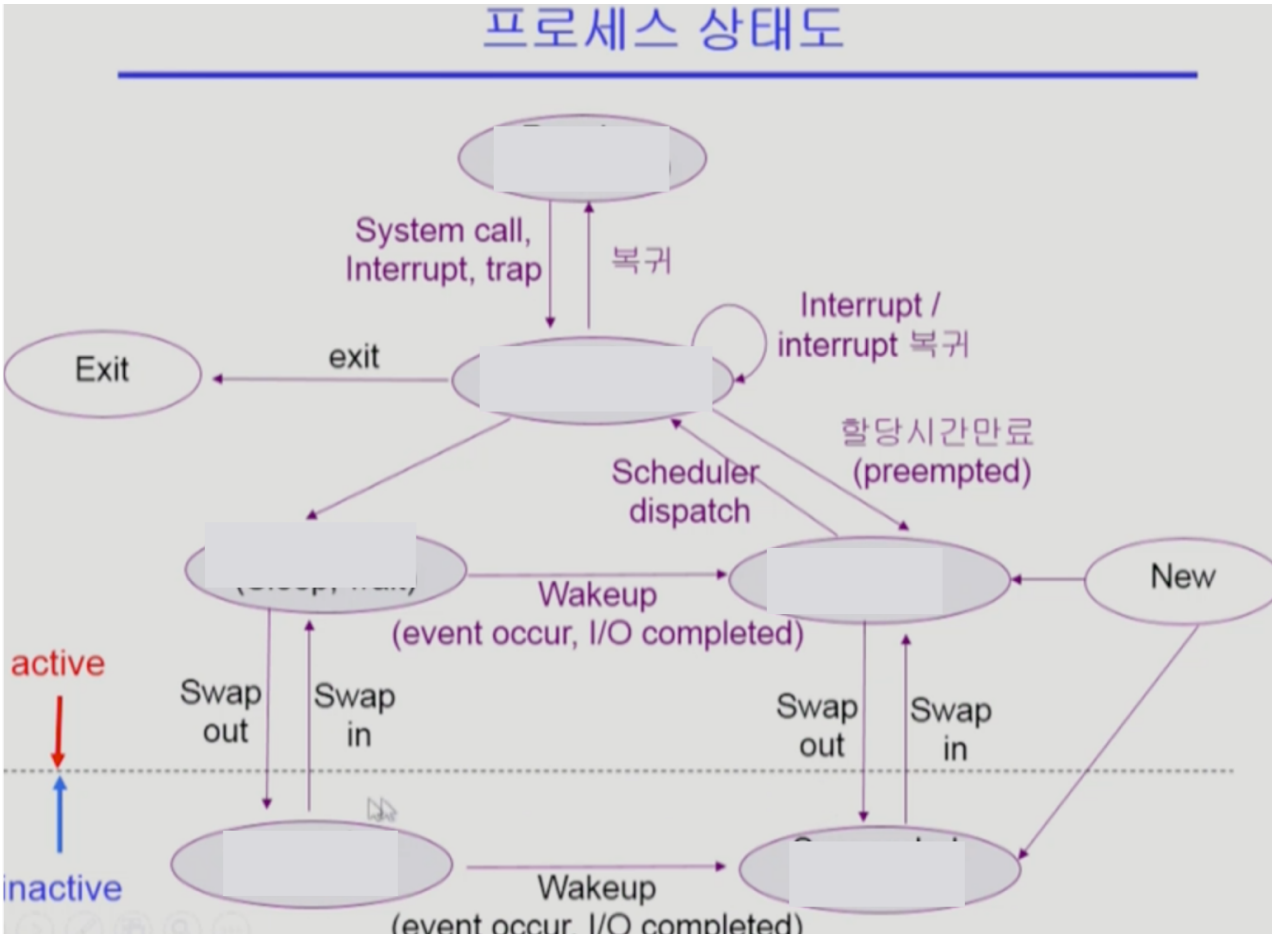
> Blocked vs Suspended

Blocked : ( )

Suspended: ( )

프로세스의 상태도 (시분할 시스템에서)

(빈칸 채워넣기)



(지난강의 질문)

동기식 입출력과 비동기식 입출력

프로세스가 입출력이 진행되는 동안에 instruction을 실행할 수 있으면 -> 비동기

입출력 완료까지 기다리면 -> 동기

- 일을 못하는 동안 CPU를 가지고 있음
- 일을 못하는 동안 CPU를 다른 프로세스에 넘겨줌

이렇게 두가지로 구현할 수 있음

Thread

스레드는 무엇인가? ( )

Tread의 구조를 그려보아라

Thread의 장점

4가지

1. ( )
2. ( )
3. ( )
4. ( )

Thread 구현

- 커널 스레드
  - ( )
- 유저 스레드
  - ( )

[chapter 4] Process Management

프로세스의 생성

부모 프로세스에서 자식 프로세스가 생성되기까지 대략적인 과정을 설명하려라

: ( )

프로세스 종료

괄호 안에 해당하는 시스템콜을 적어라

- 프로세스가 마지막 명령을 수행한 후 운영체제에게 이를 알려줌 ( )
  - 자식이 부모에게 output data를 보냄 ( )
  - 프로세스의 각종 자원들이 운영체제에게 반납됨
- 부모 프로세스가 자식의 수행을 종료시킴 ( )

  자식을 종료시키는 상황
  - 자식이 할당 자원의 한계치를 넘어섬
  - 자식에게 할당된 테스크가 더이상 필요하지 않음

  - 부모 프로세스가 종료하는 경우
    - 운영체제는 부모가 종료하는 경우 자식도 같이 종료시킨다
    - 자식 ----> 부모 순으로 단계적인 종료

fork() 시스템 콜

출력 결과를 적어라

```
int main()
{
  int pid;
  printf("\n Start Program");

  pid = fork();
  if (pid == 0)
    printf("\n Hello, I am child\n");
  else if (pid>0)
    printf("\n Hello, I am parent\n");
}
```

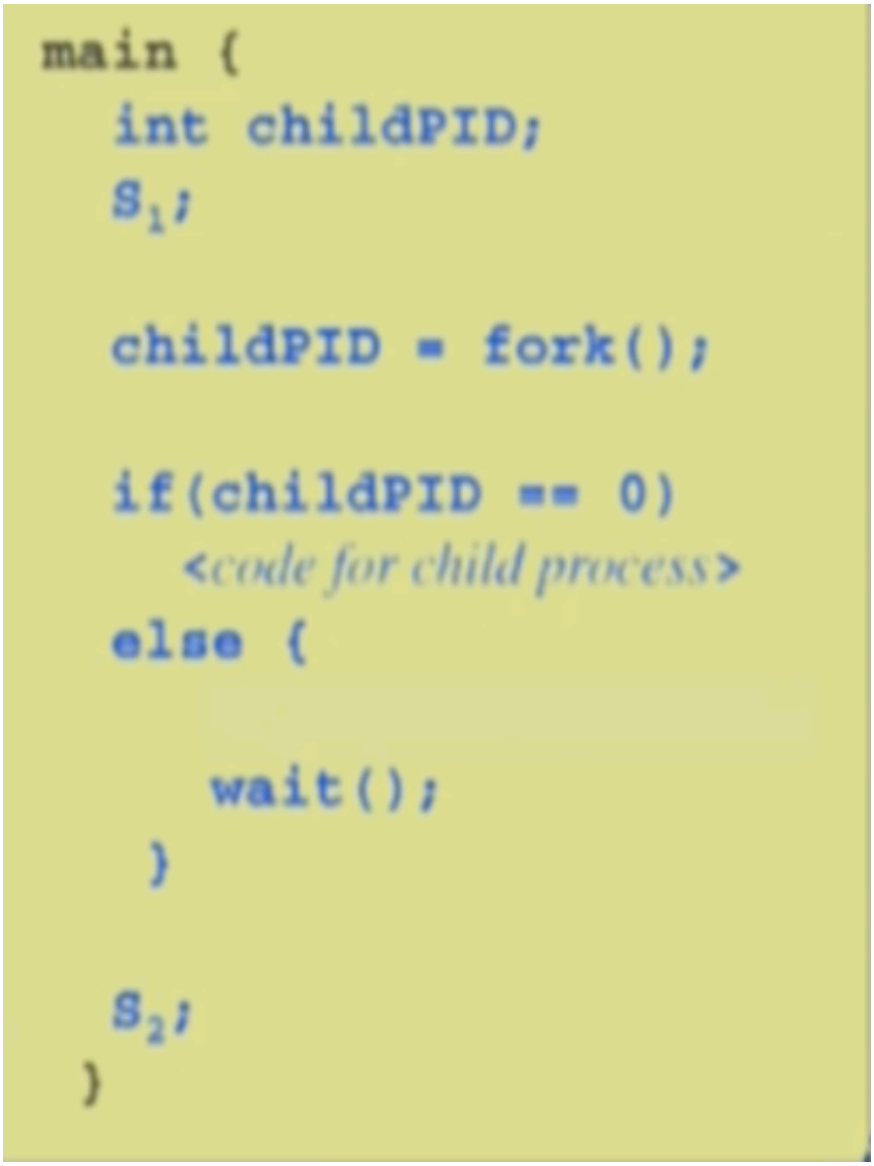
exec() 시스템 콜

출력 결과를 적어라

```
int main()
{

  printf("1");
  execlp("echo", "echo", "hello","3",(char *)0);

  print("2")

}
```

wait() 시스템 콜

아래 코드가 어떻게 실행되는지 설명하라



프로세스 간 협력

- 프로세스 간 협력 메커니즘 (IPC: Interprocess Communication)
  - 메세지를 전달하는 방법
    - ***Message passing***: ( )
  - 주소공간을 공유하는 방법
    - ***Shared memory***: ( )
  - Message passing과 shared memory가 이루어지는 과정을 그림으로 그려라