프로세스와 스레드의 차이

프로세스	스레드
운영체제로부터 자원을 할당받는 단위	자원을 사용하는 실행 흐름의 단위
더 많은 자원을 사용하기 때문에, heavyweight processes라고도 한다.	자원을 공유하기 때문에, lightweight processes라고도 한다.
생성과 종료 시간이 더 오래 걸린다.	생성과 종료 시간이 프로세스보다 더 빠르다.
프로세스는 독립적인 code, data, file 영역을 가진다.	프로세스 안에 있는 code, data, file 을 공유한다.
프로세스 간의 커뮤니케이션이 더 느리다.	스레드 간의 커뮤니케이션이 더 빠르다.
프로세스 간의 컨텍스트 스위칭 시간이 더 느리다.	스레드 간의 컨텍스트 스위칭 시간이 더 빠르다.
프로세스들은 서로 독립적으로 존재한다.	스레드는 상호의존적이다.(서로의 데이터를 읽고 쓰고 변경할 수 있다.)

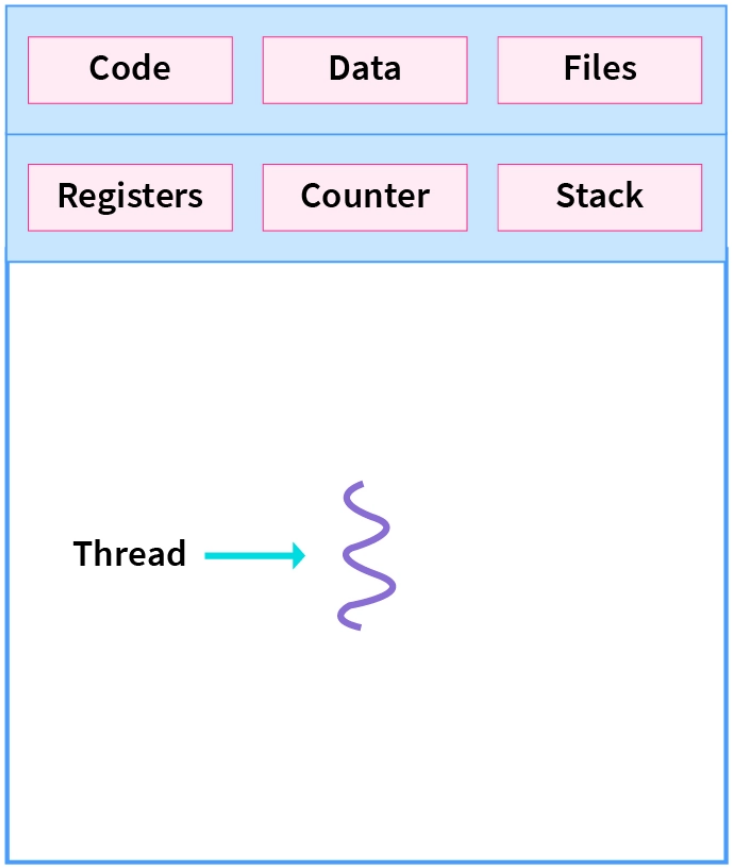
Q 프로세스와 스레드의 차이는 왜 알아야 할까?

- 프로세스와 스레드 각각의 개념과 작동방식을 알고 있는지 파악하기 위함
- 멀티 프로세스와 멀티스레드를 선택할 때 차이점을 알고 있어야 더 적합한 방식 선택이 가능함

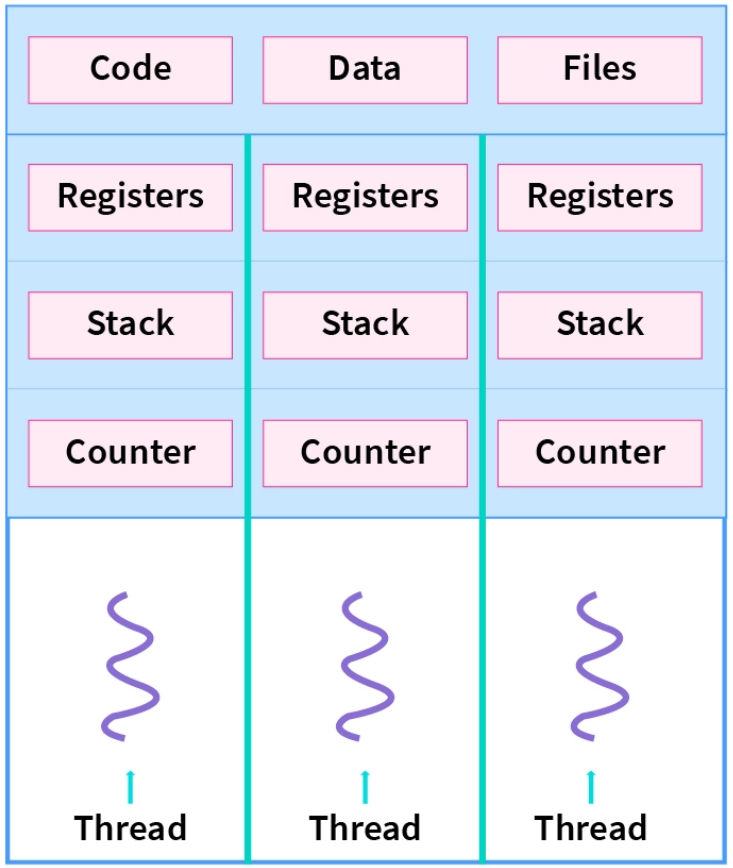
Q 스레드는 왜 stack만 독립적으로 할당받을까?

Q 프로세스는 어떤식으로 생겨날까?

Q 스레드 생성과 종료가 프로세스보다 빠른 이유는 뭘까?



Single-threaded process

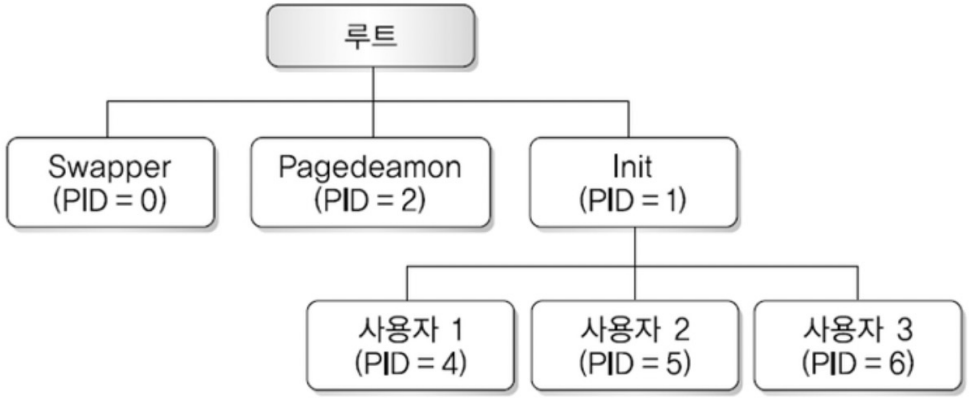


Multithreaded process

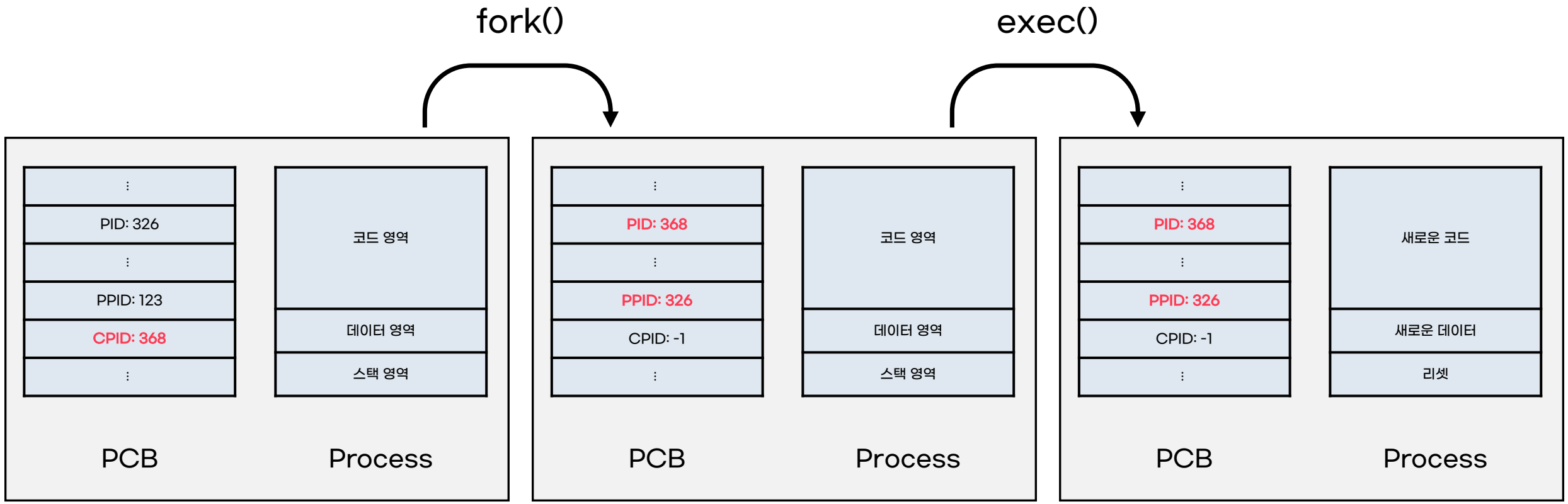
프로세스는 1개 이상의 스레드를 가지고 있다.

프로세스 (Process = Job = Task)

- 배치 시스템에서는 Job이란 용어로 통용되며, 시분할 시스템에서는 독립적인 실행 단위의 표현으로 Task라는 용어를 사용한다.
- 초창기 컴퓨터 시스템은 한 번에 하나의 프로그램만 실행 가능 → 프로그램 하나가 컴퓨터 시스템의 자원 독차지
- “프로세스”라는 용어는 1960년대 멀틱스 시스템(Multics System)을 설계한 사람들이 처음 사용했다.
- 가장 일반적인 정의는 “실행 중인 프로그램”이다.
- 실행중인 프로그램이란 디스크에 저장되어 있던 실행 가능한 프로그램이 메모리에 적재되어 운영체제의 제어를 받는 상태를 말한다.
- 프로세스가 작업을 수행하려면 프로세서 점유시간, 메모리, 파일, 입출력장치 같은 자원이 필요하다.
- 부모 프로세스로부터 자식 프로세스가 만들어질 때 부모와 모든 자원을 공유하는 모델 / 일부를 공유하는 모델 / 전혀 공유하지 않는 모델이 있다.



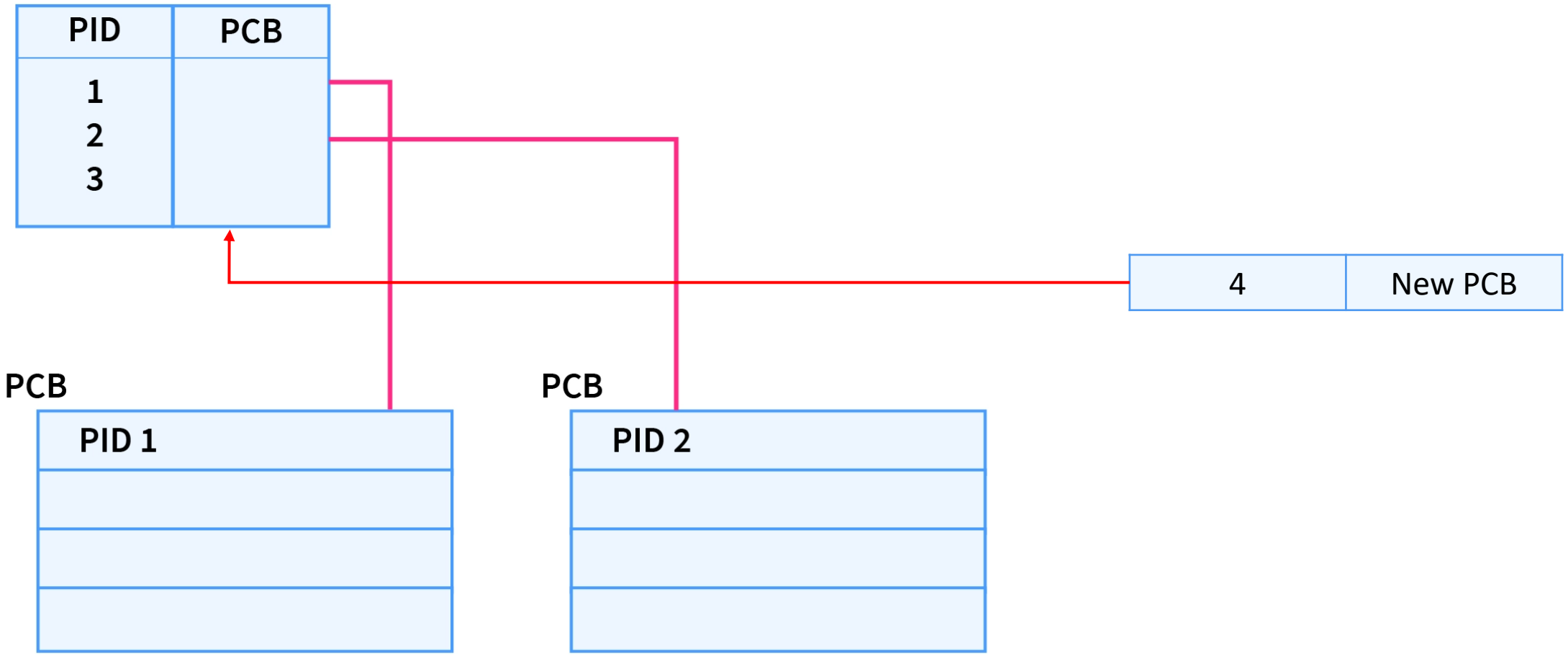
프로세스 계층 구조 예(유닉스 시스템)



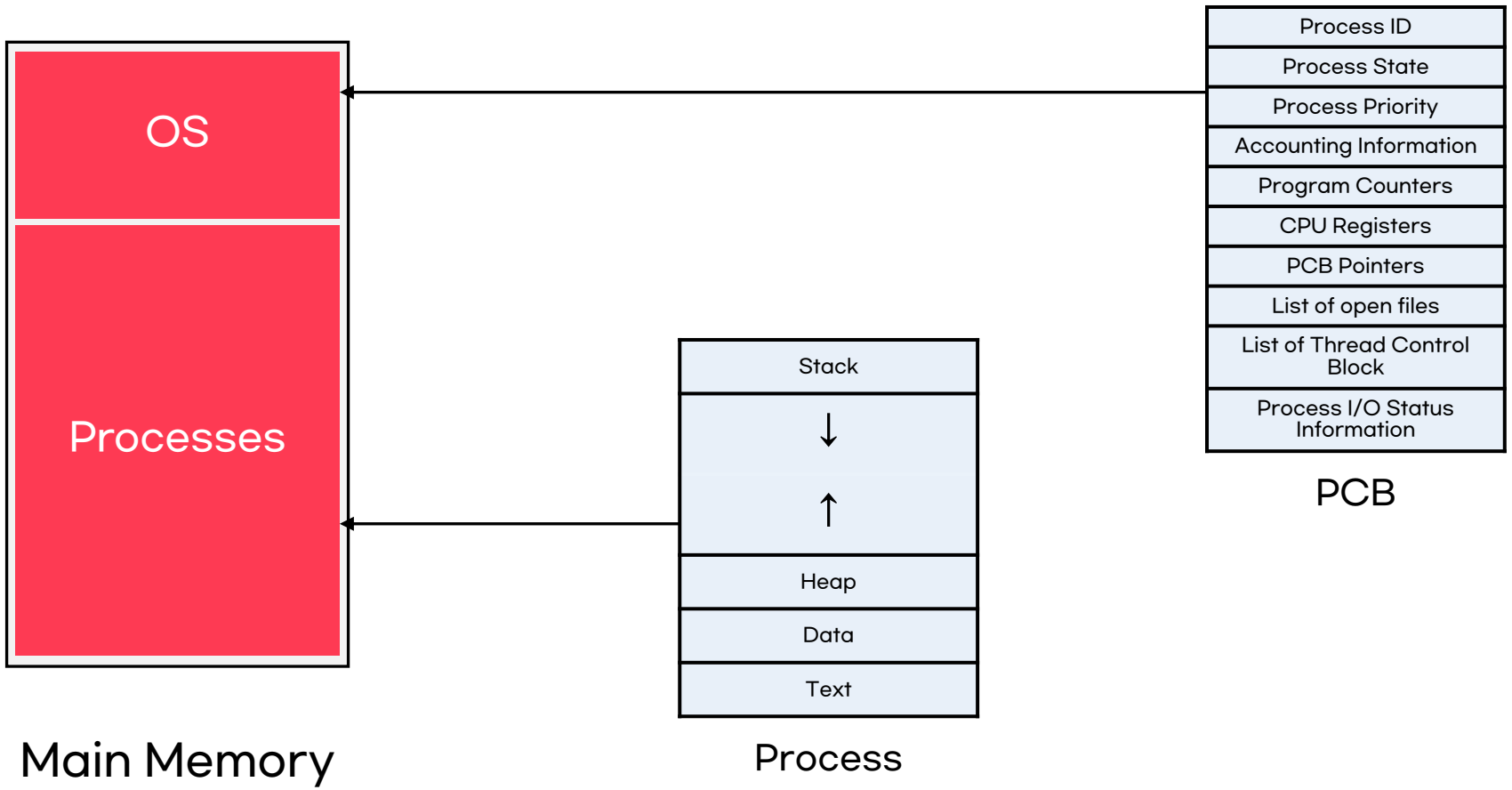
fork(): 기존의 프로세스를 복제하여 새로운 프로세스를 생성한다.
exec(): 이미 확보된 메모리 공간은 그대로 두고 그 안의 내용을 바꿔 다른 프로세스로 전환한다.

→ fork()와 exec()을 함께 사용하면 새로운 프로세스를 만들 수 있다.

1. 고유한 프로세스 아이디와 Process Control Block(PCB) 생성 후 process table에 삽입



2. PCB를 포함하여 프로세스와 관련된 데이터, 코드가 들어갈 메모리 공간 확보



3. PCB값 초기화

3-1. PCB안의 Process ID 값에 첫 번째 단계에서 할당받은 PID를 채운다.

3-2. stack pointer와 program counter를 제외한 레지스터 값들은 대부분은 0으로 채워진다.

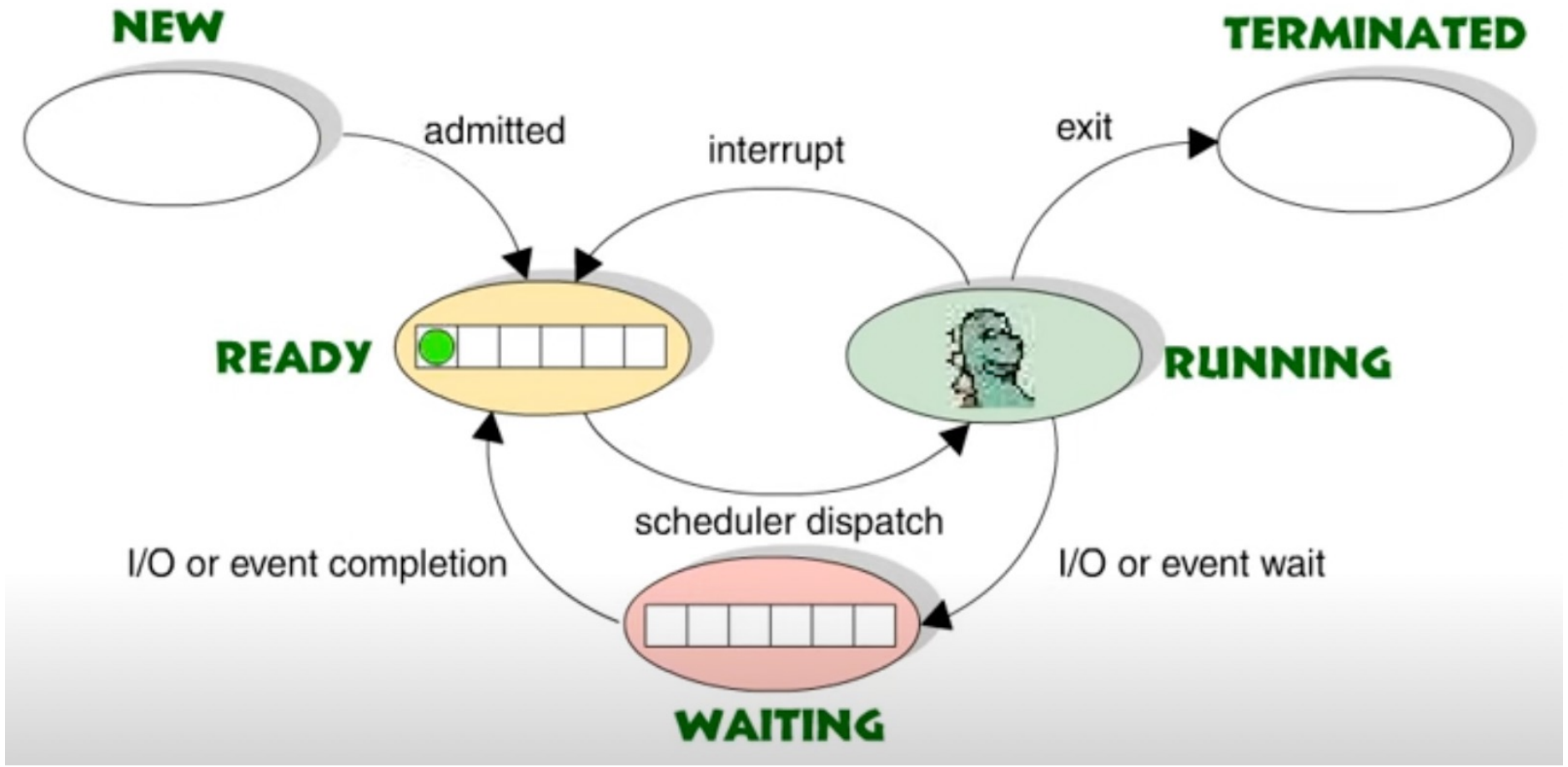
(stack pointer는 두 번째 단계에서 할당된 스택 공간의 주소로 채워지고, program counter는 프로그램의 시작 포인트의 주소로 채워진다.)

3-3. 프로세스 상태 정보는 'New'로 세팅한다.

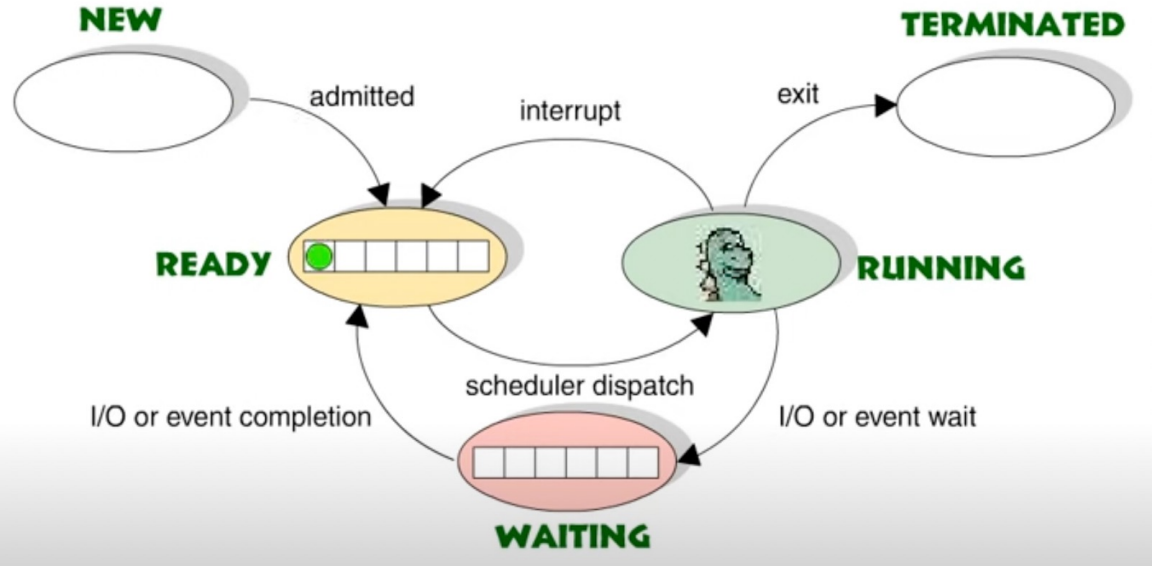
3-4. 우선순위는 디폴트로 가장 낮은 단계로 세팅되지만, 유저가 생성 과정에서 우선순위를 지정할 수 있다.

Process ID: 4
Process State: 'New'
Process Priority
Accounting Information
Program Counters
CPU Registers
PCB Pointers
List of open files
List of Thread Control Block
Process I/O Status Information

4. 프로세스를 스케줄링 큐에 링크하고 프로세스 상태를 'New'에서 'Ready'로 변경



프로세스 상태



1	생성(New)	프로세스가 메모리에 올라와 실행 준비를 완료한 상태
2	준비(Ready)	CPU를 얻을 때까지 기다리는 상태
3	실행(Running)	CPU를 얻어 실제 작업을 수행하는 상태
4	대기(Waiting)	이벤트(입출력 종료와 같은 외부 신호)를 기다리는 상태
5	종료(Terminated)	프로세스가 작업을 완료한 상태

프로세스 상태 변화

준비 → 실행

- 준비 큐의 맨 앞에 있는 프로세스가 프로세서를 할당받아 실행된다.

실행 → 준비

- 특정 프로세스가 프로세스를 독점하는 것을 막기 위해 인터럽트 클록(Interrupt Clock)을 두어 프로세서 점유 시간을 지정한다.
- 프로세스가 일정 시간이 지나도 프로세서를 반환하지 않으면 클록이 인터럽트를 발생시켜 운영체제가 제어권을 갖게 한다.
- 운영체제가 제어권을 가지면 실행 중인 프로세스는 준비 상태로 전환된다.

실행 → 대기(보류)

- 실행 프로세스가 지정 시간 전에 입출력 연산 등이 필요하거나 새로운 자원 요청과 같은 문제가 발생하면 스스로 프로세스를 양도하고 대기 상태가 된다.

대기(보류) → 준비

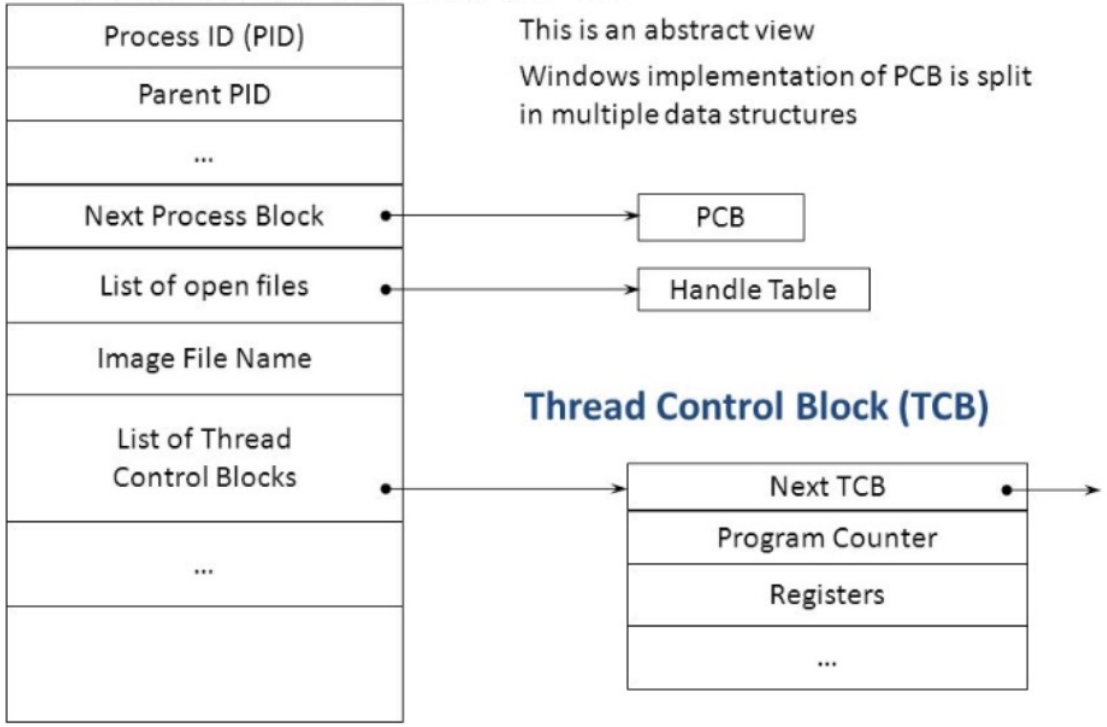
- 입출력이 끝나고 Wake Up이 발생하면 준비 상태로 전환된다.

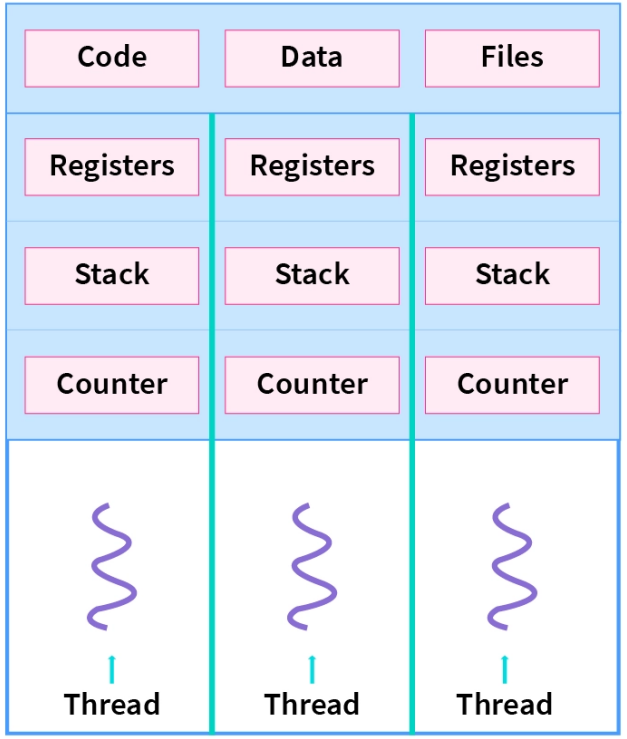
실행 → 종료

- 프로세스가 마지막 명령의 실행을 마치면 종료되고 운영체제에 프로세스 삭제를 요청한다.
- 자식 프로세스가 할당된 자원을 초과해서 사용하거나 자식 프로세스에 할당된 작업(Task)가 없으면 부모 프로세스가 자식 프로세스를 종료시킬 수 있다.

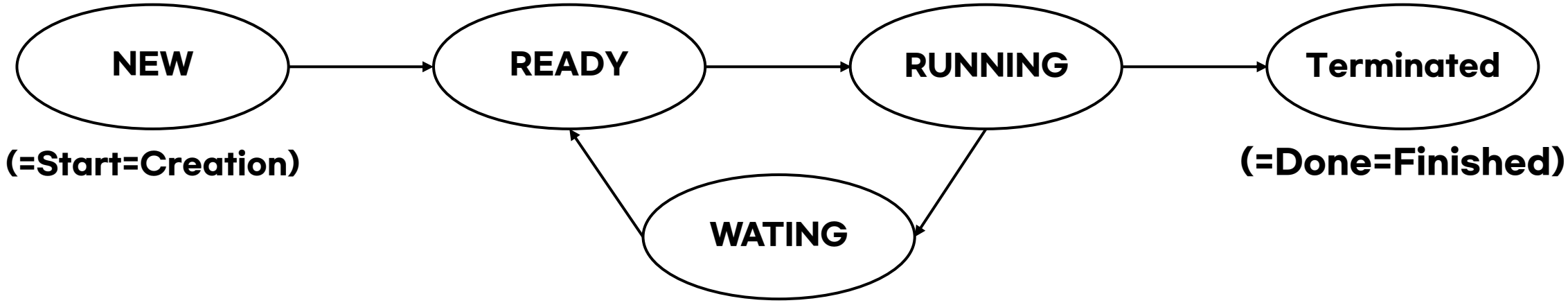
스레드 (Lightweight Process)

- 일반적으로 새로운 프로세스가 생성되면 프로세스를 위한 스레드도 함께 생성된다. 프로세스를 실행시킬 때 디폴트 스레드를 실행시킨다. (무조건 1개의 스레드는 가지게 됨)
- 스레드 생성 시스템 콜 호출(pthread_create) or 스레드 클래스를 활용한 스레드 생성 (Java나 C++과 같은 상위 수준 언어에서)
- 스레드 생성에서는 운영체제가 부모 프로세스와 공유할 자원을 초기화할 필요가 없다.
- 프로세스 내의 스레드는 해당 프로세스에서 다른 스레드를 생성하고 새로 형성된 스레드를 위한 스택과 레지스터를 제공한다.
- 프로세스의 생성과 종료 과정보다 스레드 생성과 종료 과정의 오버헤드가 훨씬 적다.





스레드 상태



1	생성(New)	스레드가 생성된 상태
2	준비(Ready)	스레드가 CPU에 의해 실행될 수 있는 상태.
3	실행(Running)	CPU를 얻어 실제 작업을 수행하는 상태
4	대기(Waiting)	이벤트를 기다리는 상태. 이 때 자신의 정보를 스택에 저장한다. 이벤트가 발생하면 준비큐에 삽입된다.
5	종료(Terminated)	스레드가 작업을 종료하면 자원을 해제하고 레지스터 문맥과 스택 할당이 제거된다.

Java에서 스레드 상태변화

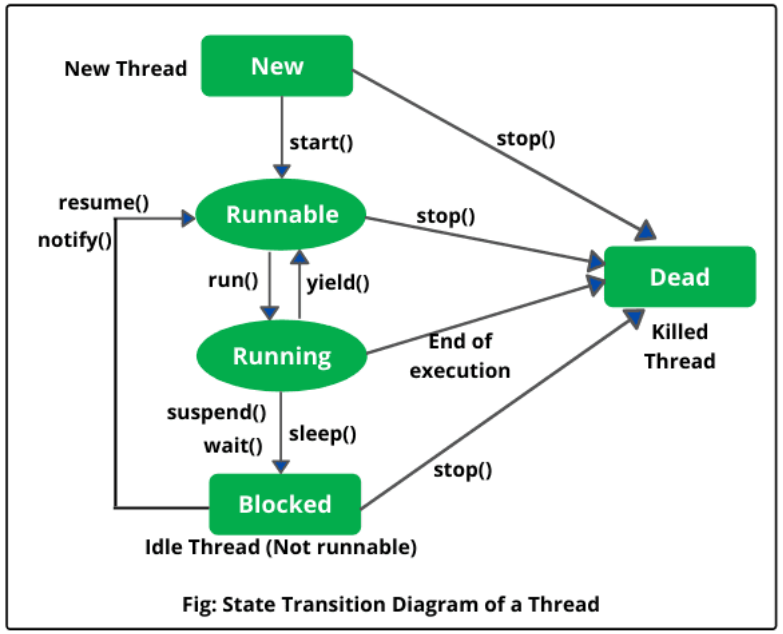
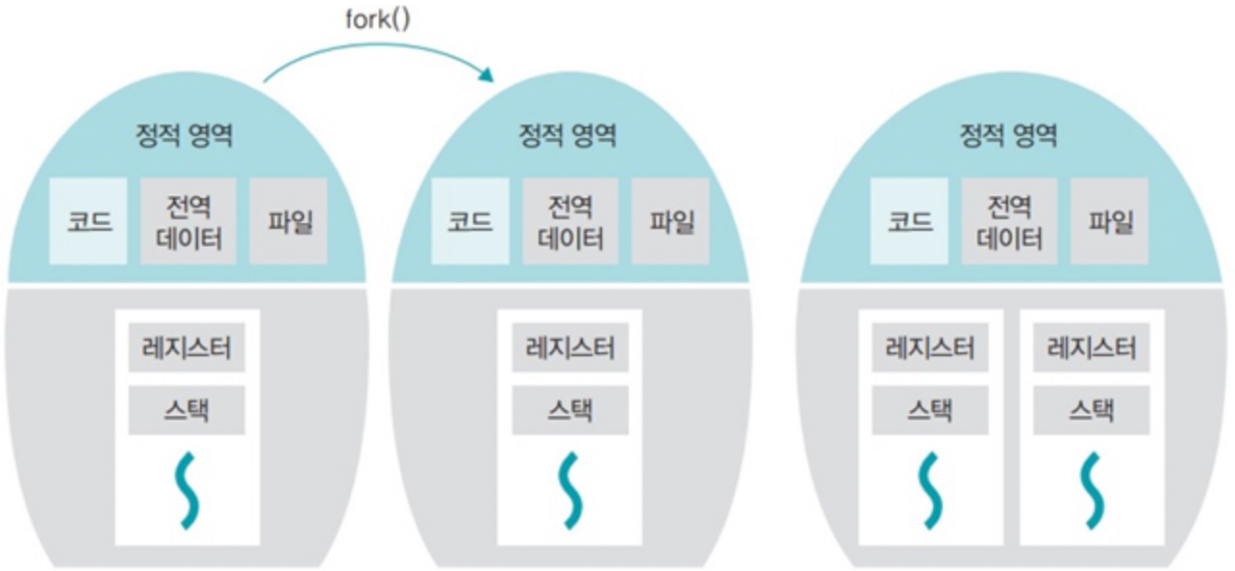


Fig: State Transition Diagram of a Thread

1	스레드 객체 생성(NEW)	스레드가 아직 시작되지 않은 상태
2	실행 대기(RUNNABLE)	스레드가 실행 가능한 상태
3	일시정지(WAITING)	다른 스레드가 특정 작업을 수행하기 기다리는 상태
4	일시정지(TIMED_WAITING)	지정된 대기 시간동안 대기하는 상태
5	일시 정지(BLOCKED)	모니터 락(monitor lock)을 기다리는 동안 차단(Blocked)된 상태
6	종료(TERMINATED)	스레드가 실행이 완료되어 종료(Terminated)한 상태

멀티 프로세스 vs 멀티 스레드

멀티 프로세스와 멀티 스레드는 한 어플리케이션에 대한 처리 방식이다.



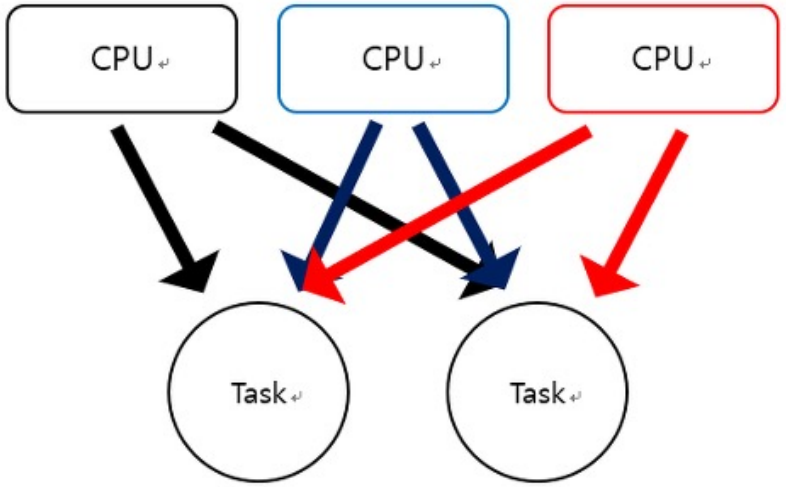
(a) 멀티태스킹 (b) 멀티스레드

멀티 프로세스	밀티스레드
운영체제로부터 자원을 할당받는 단위	자원을 사용하는 실행 흐름의 단위
더 많은 자원을 사용하기 때문에, heavyweight processes라고도 한다.	자원을 공유하기 때문에, lightweight processes라고도 한다.
생성과 종료 시간이 더 오래 걸린다.	생성과 종료 시간이 프로세스보다 더 빠르다.

멀티 프로그래밍 vs 멀티 태스킹 vs 멀티 프로세싱 vs 멀티 스레딩

- 멀티 프로그래밍: 여러 개의 프로그램을 동시에 실행시키는 것. 진행하던 프로세스가 이벤트를 처리하는 동안 다른 프로세스를 실행시킨다. 하지만 이 때 특정 프로세스의 처리 시간이 길어지면 다른 프로세스는 오래 기다려야한다.
- 멀티 태스킹: 여러 개의 프로그램을 동시에 실행시킨다는 점은 멀티 프로그래밍과 동일하다. 하지만 멀티 프로그래밍와 다르게 응답 시간을 높이기 위해 CPU 시간을 쪼개서 사용한다. 프로세스는 CPU를 할당받아 실행하다가 시간이 초과되면 준비 상태로 전환된다.
- 멀티 스레딩: 멀티 태스킹을 통해 응답 시간을 높였지만, 하나의 프로세스가 동시에 여러 작업을 수행하지는 못한다는 단점을 보완하기 위해 등장했다.

- 멀티 프로세싱: 다수의 프로세서가 협력적으로 작업을 동시에 처리하는 것



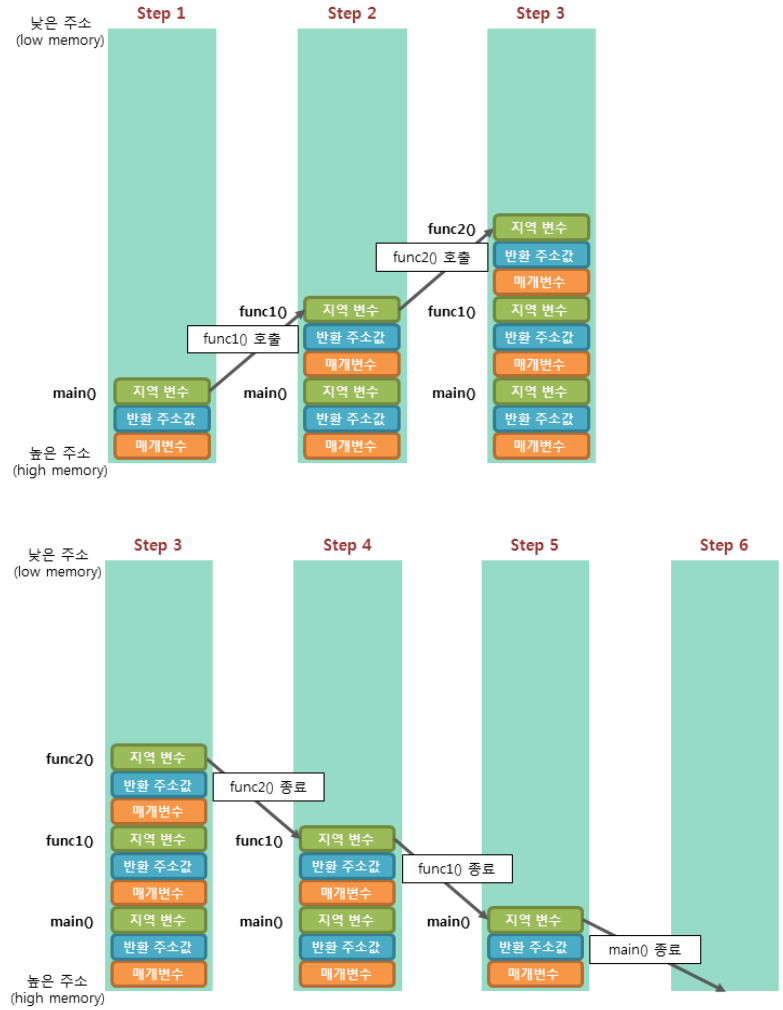
왜 스택만 스레드에 독립적으로 할당해주는걸까?

C fun-test.c

```c
int main(void)
{
    int d = func1();  // func1() 호출
    return d;
}

void func1()
{
    int c = func2(1, 2);  // func2() 호출
    return c;
}

void func2(int a, int b)
{
    return a + b;
}
```



독립적인 스택을 가진다 = 독립적으로 함수 실행이 가능하다

독립적인 실행흐름을 추가하기 위해 스택 공간을 스레드별로 할당한 것

퀴즈

1. 프로세스와 스레드 대한 설명으로 알맞지 않은 것은? (1개 이상)

1️⃣ 스레드는 자원을 공유하고 있고, 프로세스에 비해 가볍기 때문에 Lightweight process라고도 한다.

2️⃣ Process의 상태에는 New, Ready, Runnable, Run, Wait, Terminated가 있다.

3️⃣ 프로세스는 운영체제로부터 자원을 할당받는 단위이다.

4️⃣ 프로세스가 실행(Run)이 되다가 주어진 시간이 끝나면 대기(Wait) 상태로 전환된다.

2. 스레드에 스택을 독립적으로 할당하는 이유는?

3. 프로세스 제어 블록에 저장되는 정보로 올바르지 않은 것은?

1️⃣ 프로세스 상태 2️⃣ Thread control block리스트 3️⃣ 형제 프로세스 아이디 4️⃣ 프로세스 우선순위

퀴즈

1. 프로세스와 스레드 대한 설명으로 알맞지 않은 것은? (1개 이상)

1️⃣ 스레드는 자원을 공유하고 있고, 프로세스에 비해 가볍기 때문에 Lightweight process라고도 한다.

2️⃣ Process의 상태에는 New, Ready, Runnable, Run, Wait, Terminated가 있다.

→ Runnable은 Java의 Thread 에서 지원하는 Thread 상태 종류 중 하나이다.

3️⃣ 프로세스는 운영체제로부터 자원을 할당받는 단위이다.

4️⃣ 프로세스가 실행(Run)이 되다가 주어진 시간이 끝나면 대기(Wait) 상태로 전환된다.

→ 대기가 아니라 준비큐에 삽입되고 준비(Ready) 상태로 전환된다.

2. 스레드에 스택을 독립적으로 할당하는 이유는?

스택은 함수와 관련된 값과 돌아갈 위치가 저장되어 있는 메모리 공간이기 때문에, 독립적인 실행 흐름을 추가하기 위해 스택을 독립적으로 할당한다.

3. 프로세스 제어 블록에 저장되는 정보로 올바르지 않은 것은?

1️⃣ 프로세스 상태 2️⃣ Thread control block리스트 3️⃣ 형제 프로세스 아이디 4️⃣ 프로세스 우선순위

참고

1. 스택 구조 - “시스템 해킹 강좌 6강 - 스택 프레임(Stack Frame) 이해하기”
https://www.youtube.com/watch?v=ZFOHvzXcao0&t=3s
2. 스택 구조 - " 스택 프레임 ”
http://www.tcpschool.com/c/c_memory_stackframe
3. 프로세스와 스레드 개념 -
https://inpa.tistory.com/entry/%F0%9F%91%A9%E2%80%8D%F0%9F%92%BB-%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4-%E2%9A%94%EF%B8%8F-%EC%93%B0%EB%A0%88%EB%93%9C-%EC%B0%A8%EC%9D%B4#thankYou
4. 프로세스 컨트롤 블럭 - “What is Process Control Block in OS?”
https://www.scaler.com/topics/operating-system/process-control-block-in-os/
5. 프로세스의 생성과 종료 과정 - “process-creation-and-deletions-in-operating-systems”
https://www.geeksforgeeks.org/process-creation-and-deletions-in-operating-systems/
6. 스레드 생성 - “Thread creation using POSIX”
https://www.linkedin.com/pulse/thread-creation-using-posix-amit-nadiger
7. Java에서 스레드 상태 변화 - “Life Cycle of Thread in Java | Thread State”, “[Java]쓰레드 상태(Thread State)와 메서드에 대하여
”
https://www.scientecheasy.com/2020/08/life-cycle-of-thread-in-java.html/
https://jongwoon.tistory.com/14
https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.State.html
8. 프로세스의 생성과 전반적인 개념 - [책] 운영체제 개정판 - 그림으로 배우는 원리와 구조

https://www.cs.fsu.edu/~baker/opsys/notes/threads.html
thread 생성
https://www.d.umn.edu/~gshute/os/processes-and-threads.html