멀티쓰레드, 멀티프로세스

1.

문맥이 나뉘어서, 동시에 두개 이상의 쓰레드가 실행되면, 이를 **멀티 쓰레드 프로그램**이라고 한다. .

2. 단점

모든 도구가 그러하듯이 Multi Thread 프로그램이라고 해서 장점만 가진 것은 아니다. Multi Thread 프로그램은 Multi Process 프로그래밍 방식에 비해서 다음과 같은 단점을 가진다.하나의 쓰레드에서 발생된 문제가 전체 프로세스에 영향을 미친다.

멀티 프로세스의 경우에는 프로세스하나가 문제가 생기더라도 단일 프로세스로 문제가 제한된다. 그러나 멀티쓰레드 프로그램의 경우 하나의 쓰레드에 생긴 문제가 다른 쓰레드에까지 영향을 줄 수 있다. 예를 들어 쓰레드 하나가 다른 프로세스의 메모리 영역을 침범할 경우 프로세스 자체가 죽어버림으로써, 프로세스에 생성된 다른 모든 쓰레드도 프로세스와 함께 죽어버리게 된다. - 이 문제는 해결 가능하지만 여기에서는 다루지 않도록 하겠다. 시그널(:12)을 잘 활용하면 된다. 관심있으면 한번 고민해 보기 바란다. -디버깅이 어렵다. 문맥이 서로 교환되므로 추적하기가 까다롭다.

이러한 단점이 있음에도 불구하고 멀티쓰레딩 프로그래밍 기법을 선호하고 있다.

1.

프로세스를 생성하는 유일한 방법은 execl 함수를 이용하는 것이라고 배웠다. 문제는 execl 함수는 원본 프로세스의 이미지를 덮어써 버린다는 것으로, 이렇게 되면 운영체제는 동시에 단지 하나의 프로세스만을 가질 수 있게 될 것이다.

유닉스 운영체제는 fork를 이용해서 이 문제를 해결할 수 있다. fork는 원본프로그램의 복사판을 만드는 함수다. fork와 execl 함수는 분명히 다르다는 점을 인지하도록 하자. execl은 다른 프로세스를 생성하지만 fork는 자기자신을 복제한다. 즉 유닉스 운영체제에서 새로운 프로세스를 생성시키는 유일한 방법은 여전히 execl 함수를 사용하는 것이다.

유닉스 운영체제는 fork & execl 을 통해서 생성된 수많은 프로세스를 **시분할**방식으로 동시에 수행함으로써, 멀티 프로세스 환경을 제공한다.

2.

데몬프로세스가 되려면 다음과 같은 조건을 갖추어야 한다.

-일단 고아 프로세스가 되어야 한다.

-데몬 프로세스는 완전히 독립된 프로세스다. 그러므로 고아 프로세스가 되어야 한다. 표준입력, 표준출력,

표준에러을 닫는다.

-터미널을 가지지 않는다.

-

1.

프로세스는 기본적으로 code, data, stack, file I/O, 그리고 signal table의 5가지 요소로 구성이 된다. fork()를 이용해서 새로운 프로세스를 생성하게 되면, 이러한 5가지 구성요소가 모두 복사가 된다. 그러하다 보니 프로세스를 생성하는데 많은 비용이 소비될 것이다. 대게의 경우에는 프로세스를 새로 생성시킬때 발생하는 성능저하가 문제가 되지는 않겠지만 웹서비스처럼 대량의 접근이 발생하는 영역에서는 문제가 될 수 있다.

fork의 이러한 방식은 상당히 효율이 떨어지는 측면이 있다. 어떤 프로그램을 병렬로 실행시킨다고 했을 때, 실제 우리가 병렬로 실행되기를 원하는 영역은 코드의 일부분이지 프로그램 전체는 아니기 때문이다.

2.

**Thread**를 이용하면 fork()를 이용한 프로세스 기반의 병렬처리의 문제점의 많은 부분을 해결할 수 있다. **Thread**는 새로운 프로세스를 생성시키지 않고, 특정 문맥(코드)만을 병렬로 실행할 수 있도록 허용한다. 새로운 프로세스를 생성시키지 않기 때문에 그만큼 자원을 아낄 수 있으며, 더 효율적으로 빠르게 움직일 수 있다. 또한 같은 프로세스이기 때문에, 데이터를 공유하기가 쉽다는 장점도 가진다.

process:

pid\_t fork(void);

자기자신을 복제한다.

int execl ( const char \*path, const char \*arg, … );

자기자신을 해당프로그램으로 변경한다.

thread:

int pthread\_create(pthread\_t \* thread, pthread\_attr\_t \*attr, void \* (\*start\_routine)(void \*), void \* arg);

**thread** : 쓰레드가 성공적으로 생성되었을 때, 넘겨주는 쓰레드 식별 번호.

**attr** : 쓰레드의 특성을 설정하기 위해서 사용한다. NULL(:12)일 경우 기본 특성

**start\_routine** : 쓰레드가 수행할 함수로 함수포인터(:12)를 넘겨준다.

**arg** : 쓰레드 함수 start\_routine를 실행시킬 때, 넘겨줄 인자

int pthread\_join(pthread\_t th, void \*\*thread\_return);

**th** : pthread\_create에 의해서 생성된, 식별번호 **th**를 가진 쓰레드를 기다리겠다는 얘기다.

**thread\_return** : 식별번호 **th**인 쓰레드의 종료시 리턴값이다.

int pthread\_detach(pthread\_t p\_thread);

자식 쓰레드를 부모쓰레드와 완전히 분리해 버리는 방법이다.