chapter.5 프로세스 관리

1. 프로세스의 개념

프로세스(Process)란 실행 중인 프로그램(program in execution)을 뜻한다.

일반적으로 잡(job)이라는 용어와 프로세스를 혼용해 사용하기도 한다.

프로세스 문백(context)

프로세스 문맥이란 프로세스가 현재 어떤 상태에서 수행되고 있는지 정확히 규명하기 위해 필요한 정보를 의미한다.

cpu를 다시 획득해 명령의 수행을 재개하는 시점이 되면 이전의 cpu 보유 시기에 어느 부분까지 명령을 수행했는지 직전 수행 시점의 정확한 상태를 재현할 필요가 있다.

프로세스 문맥에 담기는 정보

주소공간(코드, 데이터, 스택 상태)

레지스터에 어떤 값을 가지고 있었는지

시스템 콜 등을 통해 커널에서 수행한 일의 상태

프로세스에 관해 커널이 관리하고 있는 각종 정보

- 하드웨어 문맥
- 프로세스의 주소 공간
- 커널상의 문맥 상태

2.프로세스의 상태

프로세스의 상태

- 실행 (running)
- 준비 (ready)
- 봉쇄 (blocked, wait, sleep)

실행 상태(running)

- 프로세스가 cpu를 보유하고, 기계어 명령을 실행하고 있는 상태를 가리킨다.

- 일반적인 컴퓨터 시스템 내에 cpu는 하나 이므로 컴퓨터 내에서 여러 프로세스가 동시에 수행된다고 해도 실제로 실행 상태에 있는 프로세스는 매 시점 하나이다.

준비 상태(ready)

- 프로세스가 cpu만 보유하면 당장 명령을 실행할 수 있지만 cpu를 할당받지 못한 상태를 가리킨다.

봉쇄 상태(blocked, wait, sleep)

- cpu를 할당받더라도 당장 명령을 실행할 수 없는 프로세스의 상태를 말한다.
- 예시) 프로세스가 요청한 입출력 작업이 진행 중인 경우를 들 수 있다

시작 상태(new)

- 생성 중이거나
- 프로세스가 시작되어 그 프로세스를 위한 각종 자료구조는 생성되었지만 아직 메모리 획득을 승인받지 못한 상태

완료 상태(terminated)

- 종류 중인 일시적 상태
- 프로세스가 종료되었으나 운영체제가 그 프로세스와 관련된 자료구조를 완전히 정리하지 못한 상태

문맥교환(context switch)

- 실행시킬 프로세스를 변경하기 위해 원래 수행 중이던 프로세스의 문맥을 저장하고 새로운 프로세스의 문맥을 세팅하는 과정
- 타이머 인터럽트 발생
- 입출력 용청 등으로 봉쇄 상태로 바뀌는 경우

cpu 디스패치

- 준비 상태에 있는 프로세스들 중에서 cpu를 할당받을 프로세스를 선택한 후 실제로 cpu의 제어권을 넘겨받는 과정

3. 프로세스 제어블록

프로세스 제어블록(Process control Block: PCB)이란 운영체제가 시스템 내의 프로세스들을 관리하기 위해 프로세스마다 유지하는 정보들을 담는 커널 내의 자료구조를 뜻한다.

- 프로세스의 상태 - cpu를 할당해도 되는지 여부
- 프로그램 카운터 - 다음에 수행할 명령의 위치
- CPU 레지스터의 값 -cpu연산을 위해 현 시점에 레지스터에 어떤 값을 갖고 있는지
- CPU 스케줄링 정보 -cpu 스케줄링
- 메모리 관리 정보 - 메모리 할당
- 자원 사용 정보 (accounting information) - 사용자에게 자원 사용 요금을 계산해 청구한느 등의 용도로 사용
- 입출력 상태 정보 - 프로세스가 오픈한 파일 정보 등 프로세스의 입출력 관련 상태 정보

4. 문맥교환

문맥교환(context switch)이란 하나의 사용자 프로세스로부터 다른 사용자 프로세스로 cpu의 제어권이 이양되는 과정

- 프로세스가 실행 상태일 때 시스템 콜이나 인터럽트가 발생하면 cpu의 제어권이 운영체제로 넘어와 원래 실행 중이던 프로세스의 업무를 잠시 멈추고 운영체제 커널의 코드가 실행된다. 이 경우에도 cpu의 실행 위치 등 프로세스의 문맥 중 일부를 pcb에 저장하게 되지만 이러한 과정을 문맥교환 이라고 하지는 않는다.
- 이는 하나의 **프로세스의 실행모드만이 사용자모드에서 커널모드로 바뀌는 것**일 뿐 cpu를 점유하는 프로세스가 다른 사용자 프로세스로 변경되는 과정은 아님

5. 프로세스를 스케줄링하기 위한 큐

운영체제는 준비 상태에 있는 프로세스들을 줄 세우기 위해 준비 큐를 둔다.

운영체제는 특정 자원을 기다리는 프로세스들을 줄 세우기 위해 자원별로 장치 큐를 둔다.

- 디스크에 입출력 서비스를 요청한 프로세스들은 디스크 입출력 큐에 줄 서게 된다.
- **공유 데이터에 동시에 접근**하려고 할 경우 공유 데이터를 **기다리는 큐에 줄 서게 하여** 현재 그 데이터를 사용하고 있는 프로세스가 **데이터를 반납하기 전까지 접근하지 못하게 하고**, 반납할 경우 큐에 줄 서 있는 **순서대로 데이터의 접근 권한**을 주는 방법을 사용하게 된다.

작업 큐

- 시스템 내의 모든 프로세스를 관리하기 위한 큐로, 프로세스의 상태와 무관하게 현재 시스템 내의 모든 프로세스가 작업 큐에 속한다.
- 작업 큐에 있다고 해서 반드시 메모리를 가지고 있는 것은 아님

작업 큐가 가장 넓은 범위 준비 큐와 장치 큐 모두 작업 큐에 속해 있다.

6. 스케줄러

스케줄러란 어떤 프로세스에게 자원을 할당할지를 결정하는 운영체제 커널의 코드를 지칭한다.

장기 스케줄러

- 어떤 프로세스를 준비 큐에 진입시킬지 결정하는 역할을 함
- 프로세스에게 메모리를 할당하는 문제에 관여함
- 시작 상태의 프로세스들 중 어떠한 프로세스를 준비 큐에 삽입할 것인지 결정하는 역할
- 수십 초 내지 수 분 단위로 가끔 호출되기 때문에 상대적으로 속도가 느린 것이 허용됨
- 메모리에 동시에 올라가 있는 프로세스의 수를 조절하는 역할을 함

현대의 시분할 시스템에서 사용되는 운영체제에는 일반적으로 장기 스케줄러를 두지 않는 경우가 대부분이다. 프로세스 시작 상태가 되면 장기 스케줄러 없이 곧바로 그 프로세스에 메모리를 할당해 준비 큐에 넣어주게 되었다.

단기 스케줄러

- 준비 상태의 프로세스 중에서 어떤 프로세스를 다음번에 실행 상태로 만들 것인지 결정함
- 준비 큐에 있는 여러 프로세스들 중 어떠한 프로세스에게 cpu를 할당할 것인가를 단기 스케줄러가 결정
- 밀리초 정도의 시간 단위로 매우 빈번하게 호출되기 때문에 수행 속도가 충분히 빨라야 함

중기 스케줄러

- 너무 많은 프로세스에게 메모리를 할당해 시스템의 성능이 저하되는 경우 이를 해결하기 위해 **메모리에 적재된 프로세스의 수를 동적으로 조절**하기 위해 추가된 스케줄러.

- 메모리에 올라와있는 프로세스 중 일부를 선정해 이들로 부터 메모리를 통째로 빼앗아 그 내용을 디스크의 스왑 영역에 저장해둔다. (**스왑아웃**)
- 프로세스당 보유 메모리양이 지나치게 적어진 경우 이를 완화시키기 위해
- 이때 스왑아웃 당하는 0순위 프로세스는 봉쇄 상태에 있는 프로세스들이다.

  중기 스케줄러로 인해 중지(suspended stopped)상태가 추가됨

7.1 프로세스의 생성

시스템이 부팅된 후 최초의 프로세스는 운영체제가 직접 생성하지만 그다음부터는 이미 존재하는 프로세스가 다른 프로세스를 복제 생성하게 된다.

- 프로세스를 생성한 프로세스를 부모 프로세스
- 새롭게 생성된 프로세스를 자식 프로세스라고 함

부모와 자식이 공존하며 수행되는 모델,

- 자식과 부모가 같이 cpu를 획득하기 위해 경쟁하는 관계

자식이 종료될 때까지 부모가 기다리는모델

- 자식 프로세스가 종료될 때까지 부모 프로세스는 아무 일도 하지 않고 봉쇄상태에 머물러 있다가, 자식 프로세스가 종료되면 그때 부모 프로세스가 준비 상태가 되어 cpu를 얻을 권한이 생김
- 프로세스 종료를 기다리며 봉쇄 상태에 머물러서 기다리고 있음

```
main() {
    if (fork() == 0)
        printf("\n hello, I am child! \n");
    else
        printf("\n Hello, I am parent!\n");
}
```

> 자식 프로세스는 부모 프로세스의 처음부터 수행을 시작하는 것이 아니라 부모 프로세스가 현재 수행한 시점 부터 수행하게 된다는 것이다. 현재 그 라인까지 수행했다는 기억조자도 똑같은 자식 프로세스가 생성되는 것

8. 프로세스 간의 협력

IPC

- 하나의 컴퓨터 안에서 실행중인 서로 다른 프로세스간 에 발생하는 통신을 말함
- 프로세스들 간의 통신과 동기화를 이루기 위한 메커니즘을 의미
- 공유데이터를 서로 다른 두 프로세스가 사용 할 수 있다고 하면 데이터가 불일치 하게됨

메시지 전달

- 공유 데이터를 사용 안함
- 메시지를 주고받으면서 통신하는 방식
- 프로세스의 주소 공간이 다르므로 메시지 전달을 직접 할 순 없음
- 커널이 그 역할을 대신해줌
- 커널에 의해 send(message)와 receiver(message)라는 두 가지 연산을 제공 받음
- 시스템 콜 방식으로 요청함
- 특권명령임

직접통신

- 연산의 인터페이스만 다름
- 프로세스의 이름을 명시적으로 표시
- send(P, message) 프로세스 P에게 메시지를 전송
- receive(Q, message) 프로세스 Q로부터 메시지를 전달받는 것
- 링크는 자동적으로 생성
- 하나의 링크는 정확히 한 쌍의 프로세스에게 할당됨
- 단방향성 일 수 있으나 대부분의 경우 양방향성

간접통신

- 메일박스 또는 포트로부터 전달받음
- 메일박스에는 고유의 id가 있음
- 메일박스를 공유한느 프로세스들만 서로 통신 할 슁ㅆ음
- 하나의 링크가 여러 프로세스들에게 할당될 수 있으며

- 각 프로세스의 쌍은 여러 링크를 공유할 수 있다
- 간접통신에서는 새로운 메일박스를 생성하는 연산
- send(A, message)라는 A라는 메일박스에 메시지를 전송
- receive(A, message)라는 A라는 메일박스로 부터 메시지를 전달 받음
- 메일박스를 공유하기 때문에 다음과 같은 의문 상황이 발생할 수 있음
- p1, p2, p3가 메일박스 A를 공유하는 경우
- p1이 메시지를 보냈다면
- p2,p3 어느 프로세스가 메시지를 받게 되는가
- p2, p3에게 각각 따로 링크를 생성하는 것
- 혹은 링크에 대한 receive() 연산을 매 시점 하나의 프로세스만 수행할 수 있도록 하는 방법
- 시스템이 메시지 수신자를 임의로 결정해 누가 메시지를 받았는지 송신자에게 통신해주는 방식

공유 메모리

- 공유 데이터를 사용함
- 원칙적으로 서로 다른 프로세스 A와 B는 각자 독립적인 주소 공간을 가짐
- 프로세스 A가 자신의 주소 공간에 특정한 내용을 쓸 경우 자신만 볼 수 있게됨
- 공유 메모리를 사용하는 시스템 콜을 지원함
- 서로 다른 프로세스들이 그들의 주소 공간 중 일부를 공유할 수 있도록 함
- 공유 메모리 영역은 각자의 주소 공간에 공통적으로 포함되는 영역
- 프로세스가 읽고 쓰는 것이 가능 함
- 프로세스 A, B가 독자적인 주소 공간을 가지고 있지만 이 주소 공간이 물리적 메모리에 매핑될 때 공유메모리 주소 영역에 대해서는 동일한 물리적 메모리 주소로 매핑 됨
- 통신을 수월하게 만드는 인터페이스를 제공
- 서로의 데이터 일관성 문제가 유발됨
- 커널이 책임지지 않기에 프로세스들끼리 직접 공유메모리 접근에 대한 동기화 문제를 책임져야 함