<Chapter 2. 프로세스와 스레드 관리>

프로세스 : 실행되려고하는 프로그램(실행준비상태). cpu에 할당되면 cpu에서 실행된다. (메모리 많이 필요하도 자원소비 많다.)

프로세스의 다양한 정의 : 실행중인 프로그램, pcb를 지닌 프로그램, pc를 지닌 프로그램, 능동적 개체로 순차적으로 수행하는 프로그램

운영체제가 프로세스 관리와 관련해 수행하는 기능들 : 사용자 프로세스와 시스템 프로세스의 생성과 삭제, 프로세스의 일시 중지와 재수행, 프로세스 스케줄링, 프로세스의 동기화, 프로세스 간 통신, 교착 상태(작은 자원가지고 여러 프로세스들이 차지하려 싸우는 것) 처리

CPU스케줄링 : 프로세스들에게 cpu/processor를 할당하기 위한 정책 설정. 즉 어떤 프로세스가 할당될 것인가를 정하는 방법.

목적 : 효율, 처리율을 높이고 반환시간을 최소화 시킨다.

프로세서 vs 프로세스 : 프로세서는 cpu, 프로세스는 실행되거나 실행준비 상태에 있는 프로그램이다. (메모리에 있는 프로그램은 cpu에 의해서 실행되거나 실행 준비상태에 있다)

프로세스 구성요소: 그림 참고 필수 (p.50) 메모리안에 있는 하나의 프로세스의 구성요소.

1. 코드영역 -> 실행 명령어 코드 (소스코드)
2. 데이터영역 -> 전역 변수/ 정적변수
3. 힙영역 -> 동적할당 ex)malloc()
4. ,,,,
5. 스택영역 -> 지역변수/ 매개변수

프로세스의 상태 : 그림 참고 필수 (p.51). 하나의 작업이 접수되어 완료될 때까지 다음과 같은 여러 상태의 변화를 거친다.

1. 실행상태(running) : 프로세스가 cpu를 차지하고있는상태
2. 준비완료상태(ready) : cpu가 사용가능하게 될 때 그것을 할당받을 수 있는 상태
3. 보류상태(block) : 프로세스가 cpu를 차지하고 처리하다가 입출력처리등을 하게 되면 cpu를 양도하고 입출력 처리가 완료될때까지 기다림

프로세스 제어 블록(PCB)(Process Control Block) : 프로세스의 호적이라고 보면 됨. 프로세스마다 존재하고 stack/queue에 link로 연결해서 관리.

p.53 그림 참고 필수

프로세스 생성 : 프로그램에서 다른 프로그램 생성해 새로운 프로그램 생성 unix OS(유닉스 운영체제)에서는 fork()에 의한 프로세스 생성 -> 부모, 자식 구분가능. 프로그램이 실행하는 도중에 fork()명령어 실행하게 되면 동시에 두 개의 똑같은 프로그램으로 나누어져서 실행된다. 이때 새로운 프로그램을 생성시키는 게 부모프로세스, 생선된 프로그램은 자식프로세스이다. (부모와 똑같은 자식 프로세스 생성!)

fork()명령어 : p.55 표 참고

fork()명령어 프로그램 예 : p.56 참고

프로세스 스케줄링 : 디스패치하면서 여러개중에 선택하는 방법을 프로세스 스케줄링 가지고 한다.

목적 :

1. 공정성 : 가급적 골고루. 어떤 프로세스도 무한 대기 상태 되지 않아야한다.

2. 처리능력최대화: 일을 최대한 많이 할 수 있도록한다.

3. 응답시간의 최소화(=turnaround time): 응답시간을 최대한 빠르게한다.

4. 예측가능 : 거의 같은 비용, 같은 시간안에 수행

5. 오버헤드(본업외에 추가적으로 필요한 잡다한 어떤 일, 낭비된 자원) 최소화

6. 자원 사용의 균형 유지 : 스케줄링 메커니즘들은 해당 시스템의 자원들을 쉬게해서는 안되고 사용되지 않는 자원들을 사용하도록 해야한다.

7. 응답과 이용간의 균형 유지 : 응답시간과 자원간의 균형이 필요하다.

8. 실행의 무한지연을 피한다 : 선택 못받아서 무한대기가 되면 공정성이 깨지게된다. 이로인한 문제는 에이징(aging)(프로세스가 오래 기다릴수록 우선순위가 높이는 방법)으로 해결가능하다.

9. 우선순위제 실시 : 가장 높은 순위의 프로세스를 먼저 실행시킨다.

10. 주요 자원을 점유하고있는 프로세스에게 우선권을 줘서 빨리 끝나게 도와준다.

11. 좀 더 바람직한 동작을 보이는 프로세스에게 우선권 부여 : 여기서 바람직한 동작이란 page fault가 낮음을 의미한다.

12. 과부하를 완만하게 줄인다 : 시스템과부하를 줄이기 위해 새 프로세스 막 들이지 않고 조절한다.

기준 :

1. 입출력위주의 프로세스인가?
2. 연산위주의 프로세스인가?
3. 프로세스가 일괄처리형인가 대화형인가?
4. 긴급한 응답이 요구되는가?
5. 프로세스의 우선순위
6. 프로세스가 페이지부재를 얼마나 자주 발생시키는가?
7. 높은 우선순위를 지니는 프로세스에 의해 얼마나 자주 프로세스가 선점되는가?
8. 프로세스가 받은 실행 시간은 얼마나 되는가?
9. 프로세스가 완전히 처리되는데 필요한 시간은 얼마나 더 요구되는가?

단계별 분류(=중앙처리장치 스케줄링의 세가지 중요한 단계) :

1. 상위단계 스케줄링(=작업스케줄링)(=장기 스케줄링)(=승인 스케줄링) : 메모리 내어줄지 아닐지 요청을 심사해서 승인 혹은 반려한다. (시스템이 바쁠때 승인 안하는경우있다.)
2. 하위단계 스케줄링(=단기 스케줄링) : 어떤 준비완료 프로세스에게 cpu를 할당할것인가를 결정한다.(디스패처에 의해 매초 여러번 작동함)
3. 중간 단계 스케줄링(=장기 스케줄링) : 입출력 발생해서 보류됐을 때 대기상태로 가는데(입출력끝날때까지) 어떤 프로세스가 입출력장치 차지할것인가를 결정

알고리즘이 실행되는 주기로 장기, 단기, 중기로 나눈것.

방법, 환경별 분류 :

1. 선점/비선점(preemptive/nonpreemptive)스케줄링 : 선점방식 = 응답시간이 중요하고 대화형이다. 다른 프로세스가 현재 수행중인 프로세스를 중지시키고 자기자신이 cpu를 차지한다. 오버헤드 주의해야한다. Ex)webex. 비선점방식 = 일괄처리, 질서,공정. 현재 수행중인 프로세스 다 끝날 때까지 기다려야한다.
2. 기한부(deadline)스케줄링(=실시간스케줄링) : 실시간 시스템, 선점방식. 작업들이 명시적 시간이나 기한내에 완료된다. 오버헤드 많이 일어날 수 있다. 실시간 시스템의 종류 : 경성 실시간 시스템 = 중요한 task 정한 시간내에 무조건 지켜야함. , 연성 실시간시스템 = 시간적 제약 비교적 약함. Ex)가상현실, 탐사. 실시간 스케줄링 방식 : 정적 스케줄링 방식 = task집합 미리 정해짐, 주기적인 연성 시스템에 유리, RM알고리즘(주기 짧을수록 높은 우선순위), 동적 스케줄링 방식 = task 발생시간/특성 예측불가할때 사용. Task발생이 가변적. 경성 실시간 시스템에 요구됨. EDF알고리즘(마감시간에 가장 근접한 task 먼저 처리)
3. 우선순위 스케줄링(priority) : 각 프로세스에게 우선순위 부여하여 우선순위 높은 순대로 처리함. 우선순위 기법 : 정적 우선순위 = 한 번 정해지면 잘 안바뀜. 동적 우선순위 = 우선순위가 바뀔 수 있음. (성능, 처리양 좋지만 오버헤드 많고 어려움)
4. 다중 프로세서(multiple processor) 스케줄링 : 프로세서는 동질이거나 이질이다. 이질 시스템일경우 각 프로세서는 자신의 큐가 있고 자신의 스케줄링 알고리즘을 가진다. 프로세서가 동질이면 부하공유를 하게된다(별개의 큐 가짐). 2가지 스케줄링 방식 : 1. 각 프로세서가 스스로 스케줄링, 공동 준비큐 조사하여 실행할 프로세스를 선택 2. 한 프로세서가 다른 프로세서를 위한 스케줄러로서 지정-> 주/종 구조 구성.

프로세스 스케줄링 알고리즘 : 필기본 참고.

-스레드(thread) : 어떠한 프로그램 내에서 특히 프로세스 내에서 실행되는 흐름의 단위이다.

-스레드의 특성 : 각 스레드는 독립적이다, 스레드의 실행/종료 순서는 예측불가, 스레드들은 수행을 위해 스케줄되고 결과들은 프로세스에게 전달된다, 프로그램에 있는 스레드의 수는 다른 스레드에게 알려지지않는다. 스레드는 서로 독립적이지만 한 스레드가 취한 행동은 다른 스레드에게 영향을 미친다, 스레드는 프로세스의 일부이기 때문에 프로세스의 자원들을 공유하지만 그 자신의 처리시간과 스택과 레지스터들이 할당된다, 한 프로세스 exit()시스템을 통해 종료되면 모든 스레드들은 종료된다.

-스레드와 프로세스의 관계 : p.80 그림 참고

-단일 스레드형 프로세스와 다중 스레드형 프로세스의 차이 : p.81. 중량 프로세스(=단일스레드형프로세스) : 하나의 스레드를 가진 프로세스, 경량 프로세스(=다중스레드형프로세스) : 프로세스 내에 2개이상의 스레드를 가진 프로세스

-스레드 제어 : 1. KLT방법 2. ULT 방법 3. KLT와 ULT의 혼합. KLT = 윈도우즈, 유닉스, 리눅스 등에선 스레드들이 커널에 의해 지원된다. ULT = POSIX(portable os interface)는 사용자수준에서 라이브러리 호출 집합을 통해 커널의 상위 수준에서 지원. KLT와 ULT방법의 혼합 = sun solaris.

자바 스레드 스케줄링 : 자바 런타임 환경은 고정 우선순위 스케줄링이라 부르는 결정적 스케줄링 알고리즘 지원. 이 알고리즘은 다른 실행가능 스레드들은 관련 이선순위에 기초해 스케줄. 자바 스레드 생명주기 상태. P.84

-자바 스레드 생명주기 상태 : p.83참고

스레드가 생성될 때 스레드를 생성하는 스레드로부터 스레드의 우선순위를 상속받는다. (스레드가 생성된 후에 setPriority 메서드를 사용하여 언제든지 우선순위 수정가능.) 스레드 우선순위들은 스레드 클래스에 정의된 상수들인 MIN\_PRIORITY와 MAX\_PRIORITY 사이의 정수들이며 그 수가 클수록 우선순위 높다. 스레드를 위한 우선순위의 허용치는 1에서 10까지 범위이며 MIN\_PRIORITY의 값은 1이고 MAX\_PRIORITY는 10이다. 같은 우선순위의 2개의 스레드들이 중앙처리장치를 기다리면, 그 스케줄러는 실행을 위해 하나를 임의로 선택한다. 선택된 스레드는 다음 조건일 때까지 실행된다. -> (우선순위가 높은 스레드가 실행 가능하게 됨.스레드가 처리기 사용을 양보하거나 run 메서드를 빠져나옴. 시간 분할을 지원하는 시스템에서 시간 할당이 만료됨. )자바 런타임 시스템의 스레드 스케줄링 알고리즘은 선점 방식을 이용.

-자바 멀티스레드 우선순위 스케줄링의 예