< 운영체제 기말고사 대체 과제 2 > 20201081g 조정현

1. 바쁜 대기는 한 프로세스가 이미 임계영역에 있으면 이 임계영역에 들어가려는 프로세스는 진입 코드에서 계속 반복 순환하는 것이다.
바쁜 대기는 세마포를 사용하여 피할 수 있다. 세마포는 다양한 동기화 문제들을 해결하기 위해 사용한다.

2.
1) 무한 대기 : 상대방 프로세스의 상태를 모르고 실행을 멈추고 계속하여 기다리고 있는 상태
2) 제한된 대기 . 한 프로세스가 임계영역에 대한 요청 후부터 요청이 수락되기까지의 기간 내에 다른 프로세스가 임계영역을 수행할 수 있는 횟수에 제한이 있어야 한다.
3) 바쁜 대기 : 한 프로세스가 임계영역에 있으면 이 임계 영역에 들어가려는 프로세스는 진입 코드에서 계속 반복, 순환하는 것이다.

3. 바쁜 대기를 완전히 피할 수 없다. 바쁜 대기는 공유 자원을 여러개의 프로세스들이 사용하길 원하기 때문에 발생하는 것이다. 공유 자원을 복제해 각각의 프로세스에게 할당해주면 바쁜 대기를 피할 수 있을 것이다. 하지만 자원을 복제하는 것은 메모리의 효율성을 떨어트린다. 메모리 사이즈를 무한정 증가시켜 복제된 자원을 할당할 공간을 확보할 수 있지만 비용과 크기 측면에서 메모리를 무한으로 늘리는 것은 불가능하다.

5. 임계 영역이란 임계 자원에 접근하고 실행하는 프로그램 코드 부분이다.
문제점은 반드시 한번에 하나의 프로세스만 임계 영역에 접근이 가능한 것과
특정 프로세스가 임계 영역에 오래 머물거나 무한 루프에 빠지지 않도록 관리해야한다는 것이다.

6. 상호배제 , 진행, 유한대기이다.

7. wait() 과 signal()이 있다. 이는 각각 P( )와 S( )와 같은 표현이다.
wait()는 프로세스를 준비 큐에 추가하여 대기시키는 연산이고,
signal()은 준비 큐에서 대기 중인 프로세스를 준비 큐에서 제거하고 공유 자원에 접근해도 된다는 신호를 보내는 연산이다.

8. wait 연산 혹은 signal 연산이 생략되면 상호배제 문제와 wait 연산 때문에 준비 큐에서 대기하고 있는 프로세스들이 교착 상태에 빠질 수 있다.
세마포가 사용되는 동안 wait 연산이 시작되면 프로세스는 다른 경로를 선택할 수 없다.
프로세스는 한 번에 오직 한 세마포만 대기할 수 있으므로 자원을 할당하는 과정에서 교착 상태를 가져올 수 있다.

9.
1) 교착 상태에 있다.
2) 안전하지 않다.
3) 안전하다.

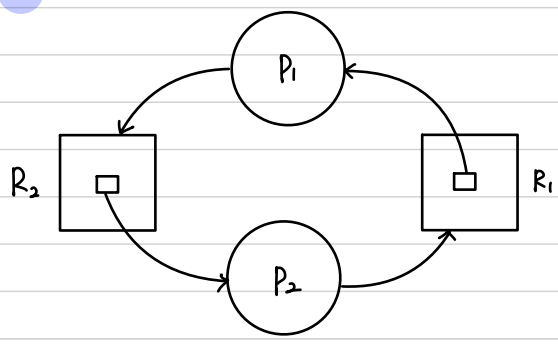
10. 프로세스를 한 개 이상 중단한다. 즉, 정지된 프로세스에 할당된 모든 자원의 해제를 요청하는 것이다.
교착상태의 프로세스들에서 자원을 선점한다. 교착 상태가 해결될 때까지 선점한 자원을 다른 프로세스에 할당해야한다.

11.
1) 프로세스가 수행된 시간과 앞으로 종료하는데 필요한 시간
2) 프로세스가 사용한 자원 형태와 수
3) 프로세스를 종료하는데 필요한 프로세스 수
4) 프로세스를 종료하는데 필요한 자원 수
5) 프로세스가 대화식인지, 일괄식인지 여부

12.
1) 사거리에서 교통마비
2) 주차장의 차들이 꽉 막힌 경우
3) 두 명의 사람이 각각 사다리의 위쪽과 아래쪽에 있다고 가정했을 때,
아래에 있는 사람은 위로, 위에 있는 사람은 아래로 이동하려고 하는 경우

13.
(i) 교착 상태가 자원을 자유롭게 할당한 자원 부족의 결과라면, 기아 상태는 교착 상태를 예방하려고 자원을 할당할 때 발생하는 결과이다.
(ii) 교착 상태는 둘 이상의 프로세스가 서로 남이 가진 자원을 요구하면서 양쪽 모두 대기 상태에 있는 상태이다.
경쟁 상태는 프로세스들이 하나의 자원을 가지기 위해 서로 싸우는 상태이다.
경쟁 상태에서 고려해야할 자원은 1개 이지만 교착 상태에서는 그게 이상이다.

14.



20201081 9 조정현 2

15. 교착 상태라고 단정할 수 없다.

16. 교착 상태가 발생할 수 없다. 교착 상태는 2개 이상의 프로세스가 필요하기 때문이다.

17. 감지할 수 없다. 기아 문제는 우선 순위가 낮은 스레드 혹은 프로세스에서 발생하므로 이들의 우선 순위를 높여주면 된다.
대기 시간이 지날 수록, 낮은 우선순위를 가진 프로세스의 우선 순위를 높여주는 방법인 에이징으로 해결할 수 있다.
우선 순위 계승 프로토콜 (priority inheritance protocol) 으로도 해결할 수 있다.

18. 논리적 주소는 데이터가 디스크 혹은 SSD 등 보조 기억 장치에 있을 때 CPU가 주는 것이지만
물리적 주소는 load time 동안 데이터를 주기억 장치에 올리면서 메모리 단위로 표시되는 실제 주소이다.
물리적 주소는 CPU에 의해 만들어지지 않는다.

19.
1) 13 비트. $1024 = 2^{10}$ 이고 $8 = 2^3$ 이므로 $10+3 = 13$이기 때문이다.
2) 15 비트. $32 = 2^5$ 이고 $1024 = 2^{10}$ 이므로 $5+10 = 15$ 이기 때문이다.

20.
1) 최초 적합 : 사용 가능 공간 리스트에서 충분히 큰 첫 번째 공백 분할 공간에 할당한다.
2) 최적 적합 : 사용 가능한 공간 중에서 가장 작은 크기의 단편화가 생기는 곳에 할당한다.
3) 최악 적합 : 사용 가능한 공간 중에서 가장 큰 사용 가능 공간에 작업을 할당한다.

21.
1) 최초 적합 알고리즘
① 212 KB ⟶ 500KB 부분에 할당
② 417 KB ⟶ 600KB 부분에 할당
③ 112 KB ⟶ 500KB 부분에 할당
④ 426 KB 는 들어갈 공간이 없으므로 다른 프로세스 끝날 때까지 대기한다.

2) 최적 적합 알고리즘
① 212 KB ⟶ 300KB 부분에 할당
② 417 KB ⟶ 500KB 부분에 할당
③ 112 KB ⟶ 200KB 부분에 할당
④ 426 KB ⟶ 600KB 부분에 할당

3) 최악 적합 알고리즘
① 212 KB ⟶ 600KB 부분에 할당
② 417 KB ⟶ 500KB 부분에 할당
③ 112 KB ⟶ 300KB 부분에 할당
④ 426 KB 는 들어갈 공간이 없으므로 다른 프로세스 끝날 때까지 대기한다.

⇒ 최적 적합 알고리즘이 기억장치를 가장 적절히 사용한다.

22. 내부단편화 : 페이징에서 페이지 단위로 메모리에 적재시킬 때 프로그램의 사이즈가 정해진 단위의 블록으로 나뉘게 되는데,
이때 정해진 단위의 사이즈보다 조금 큰 부분을 하나의 블록에 저장하게 되어 블록 안에 생기는 빈 공간을 의미한다.
외부단편화 : 세그멘테이션에서 세그먼트 단위로 메모리에 적재시키는데 세그먼트와 세그먼트 사이의 공간이 다른 세그먼트는 들어갈 수 없을 정도의 빈 공간이 생기는 것을 의미한다.

23.
1) 400 나노초 ∵ 메모리 참조 200 나노초 + 페이지로 기억 장치 참조 200 나노초 = 400 나노초
2) 250 나노초 ∵ $(0.75 \times 200) + (0.25 \times 400) = 150 + 100 = 250$ 나노초

24. 항목 2개가 메모리의 동일한 페이지 프레임을 가리키면 사용자들은 코드와 데이터를 공유할 수 있다.
많은 양의 메모리 복사는 서로 다른 테이블들의 동일한 메모리 할당을 가능하게 하여 시간을 감소시킬 수 있다.
한 페이지를 다른 페이지로 업데이트하면 그 코드를 사용할 수 있는 모든 사용자의 그 코드 변경과 복사가 가능하다.

25.
1) 64개.
2) 18비트
3) 페이지 번호 6비트, 페이지오프셋 12비트

26. 페이지의 논리적 주소인 페이지 번호와 이것에 대응하는 물리적 주소인 페이지 프레임 주소를 포함하여 별도의 레지스터로 구성하거나 메인 메모리에 배치하는 기능을 한다.

27. 논리적 주소이다. 함수 호출과 순환 부분, 그리고 데이터를 참조하는 주소를 생성하려면 페이지 테이블을 참조해야하기 때문이다.

28.

논리주소(10진수)	페이지 번호(10진수)	오프셋(10진수)
2375	1	327
19366	18	934
30000	29	304
256	0	256
16385	16	1

29.

1) 프로세가 원하는 정보가 담긴 그 페이지가 페이지 테이블에 없기 때문이다.

2) 운영체제가 다른 메모리에 접근할 수 있다.

운영체제가 다른 메모리에 접근할 수 없게 해야한다.