Operating system:

191103

1. process, process table & process status

프로세스는 실행중인 프로그램의 *인스턴스다. 예를 들면, 웹 브라우저와 쉘은 모두 프로세스다.

*인스턴스: 해당 클래스의 구조로 컴퓨터 저장공간에 할당된 실체

OS는 모든 프로세스를 관리하고 프로세서를 사용하도록 할당할 책임이 있다. OS는 프로세스가 필요로 하는 다른 메모리와 디스크 같은 자원들을 할당한다. 프로세스들의 상태를 계속 추적하기 위해 OS는 프로세스 테이블을 사용한다. 테이블 안에는 프로세스가 사용하는 리스트와 프로세스의 현재상태가 기록된 프로세스 리스트가 들어있다.

프로세스 상태는 세 가지다: running, ready, or waiting. 실행 상태는 프로세스가 실행에 필요한 모든 자원을 들고 있고 OS로부터 프로세서 사용을 허가받은 상태다. 오직 하나의 프로세스만이 실행 상태일 수 있다. 나머지 프로세스들은 대기 상태이거나 준비 상태이다. 대기와 준비 상태는 큐로 구현한다.

2. thread & process

쓰레드는 프로세스 안의 *single sequence stream이다. 쓰레드는 프로세스의 속성 일부를 가지고 있기 때문에 경량 프로세스로 불리기도 한다. 쓰레드는 병렬처리에 사용된다.

쓰레드는 각각의 *프로그램 카운터, 레지스터 셋, 스택 영역을 가진다. 쓰레드는 프로세스처럼 서로 독립이 아니다. 쓰레드는 서로 코드 섹션, 데이터 섹션, OS 자원 등을 공유한다.

3. multithreaded programming

시스템을 더 빨리 응답하게 하고 자원 공유를 가능하게 한다. 더 경제적이고 선호되는 구조인 멀티 프로세스 아키텍처로 이어진다.

4. scheduling algorithms

스케줄링의 목적은 다음과 같다. 최대 cpu 활용, cpu의 고른 할당, 최대 처리량, 최소 반환 시간, 최소 대기시간, 최소 응답 시간

1) FCFS(=FIFO): 요청이 들어온 순서대로 프로세스에 CPU를 할당한다. 구현에 FIFO 큐를 사용한다. *프로세스 제어 블록이 준비 큐의 마지막에 링크된다. 비선점 알고리즘이다. *콘보이 효과 등의 문제가 있다.

2) shortest job next(SJN): 최소 작업시간을 가진 프로세스를 먼저 처리한다. 작업시간이 같다면 FCFS로 처리한다. 비선점 알고리즘이다. 기아(무기한 연기) 문제가 있지만 에이징 기법으로 해결 가능하다.

3) shortest remaining time(SRT): SJN의 선점 알고리즘 방식이다. 잔여시간이 짧은 프로세스를 먼저 처리한다.

4) highest response ration next(HRRN): 기아 문제를 해결하기 위해 에이징을 도입한 알고리즘. Response ratio = (waiting time + burst time)/burst time 이 높은 순으로 프로세스를 처리한다.

5) round robin(RR): 프로세스들 사이에 우선순위를 두지 않고 순서대로 시간 단위로 CPU를 할당한다. 준비 큐가 환형 큐이다. 시간 단위 동안 수행된 프로세스는 준비 큐 끝으로 밀려난다. 선점형 알고리즘이다.

6) priority scheduling: 우선순위에 따라 프로세스를 처리한다. 비선점형 알고리즘이다.

7) multi-level queues: 우선순위에 따라 프로세스들을 다른 큐에 배치한다. 우선 순위가 높을수록 상위 큐에 배치된다. 상위 큐의 수행이 완료된 후에 하위 큐가 수행된다. 기아 문제가 있다.

8) multi-level feedback queues: 다단계 큐에서 에이징을 적용한 알고리즘. 큐 사이의 이동이 가능하다.

4.1. arrival, completion, burst, turn around & waiting time

1) arrival time: 도착시간. 프로세스가 준비 큐에 도착한 시간

2) completion time: 완료시간. 프로세스가 수행을 끝낸 시간

3) burst time: 실행시간. 프로세스가 cpu를 실행하는 데 필요한 시간

4) turn around time: 반환시간. 프로세스가 준비 큐에 도착해 수행을 끝내는 데 걸린 시간. completion time - arrival time

5) waiting time: 대기시간. 프로세스가 준비 큐에서 기다린 시간. turn around time - burst time

5. deadlock

데드락은 둘 이상의 프로세스들이 서로 끝나기를 기다리는데, 어느 프로세스도 끝나지 않는 상황을 말한다. *식사하는 철학자 문제. https://simsimjae.tistory.com/72

6. necessary conditions for deadlock

1) 상호배제: 공유되지 않는 자원이 있다.

2) 점유대기: 프로세스는 최소 하나의 자원을 들고 있고 다른 프로세스가 들고 있는 자원을 기다린다.

3) 비선점: 프로세스가 자원을 돌려주기까지 OS는 그 자원을 가져갈 수 없다.

4) 순환대기: 다수의 프로세스가 환형으로 서로를 기다리고 있다.

조건을 하나라도 만족하지 않으면 데드락이 발생하지 않는다. 보통 데드락을 해결할 때 상호배제와 선점불가를 없애는 것으로 해결하지 않는다.

7. virtual memory

가상 메모리는 사용자가 하나 이상의 연속적인 주소공간들을 갖는 것처럼 보이게 만든다. 각 주소공간들은 0으로 시작한다. 가상 메모리의 아이디어는 디스크 공간을 램의 확장으로 쓰는 것이다. 프로세스는 메모리 관리 유닛(MMU)에 의해 메모리가 디스크에 있는지 램에 있는지 상관하지 않는다. 대용량 메모리는 가상 메모리를 작은 조각들로 분할하는 것으로 구현된다. 각 조각들은 프로세스가 필요로 할 때, 물리적 메모리로 로드된다.

8. segmentation & paging

세그멘테이션과 페이징은 MMU의 메모리 관리 기법이다. 이를 통해 가상 메모리 주소에서 물리 메모리 주소를 알아낼 수 있다.

세그멘테이션은 메모리를 *커널, 스택, 공유라이브러리, 힙, BSS, 데이터, 코드의 서로 다른 논리적 블록 단위(세그먼트)로 나누고 메모리를 할당하는 방법이다. 세그멘테이션에서는 세그먼트 시작주소와 길이 정보가 담긴 세그먼트 테이블을 운용한다. 이를 통해 각각의 세그먼트를 offset 단위로 관리할 수 있다. 또한 시작주소와 길이로 실제 물리적 메모리 주소로 변환 가능하다.

페이징은 가상 메모리를 고정된 크기로 나누어 관리하는 방법이다. 나뉜 블록을 페이지라고 하며 대응하는 물리 메모리의 블록 단위를 프레임이라고 한다. 프레임 번호에 offset을 더하는 것으로 주소 전환이 이루어진다.

8.1. thrashing

쓰래싱은 컴퓨터의 성능이 저하되거나 아예 무너지는 경우를 말한다. 쓰레싱은 시스템에서 수행 트랜잭션보다 *페이지 부재 처리에 시간이 걸리는 경우 발생한다. 페이징의 부정적 이면으로 볼 수 있다.

*페이지 부재: 페이지에 접근 요청을 했는데 물리적 메모리에 없는 상태

9. page replacing algorithm

1) optimal replacement(OPT): 가장 오랫동안 사용하지 않을 페이지를 예측해 교체한다.

2) FIFO: 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체한다.

3) least recently used(LRU): 최근에 가장 오랫동안 사용하지 않은 페이지를 교체한다.

4) least frequently used(LFU): 사용빈도가 가장 낮은 페이지를 교체한다.

5) not used recently(NUR): LRU와 유사. 교체 순서를 참조와 변형 두 기준으로 판단한다.

6) second chance replacement(SCR): FIFO로 교체하되, 사용된 페이지를 FIFO 리스트 마지막으로 옮긴다.

https://jhpop.tistory.com/34

9.1. belady's anormaly

벨라디 모순은 페이지 프레임 수가 많으면 페이지 부재율이 줄어드는 것이 일반적이지만, 프레임 수를 증가시켰는데도 페이지 부재가 더 많이 일어나는 현상이다. FIFO 알고리즘에서 발생한다.
https://stackoverflow.com/questions/4800285/cant-understand-beladys-anomaly

10. mutex & semaphore

뮤텍스는 공유 자원에 여러 쓰레드가 접근하는 것을 막는 것이다. mutual exculsion의 약자이다. 공유 자원에 대한 접근을 조율하기 위해 locking과 unlocking을 사용한다. 뮤텍스는 프로세스 범위에 존재한다. 따라서 프로세스가 사라지면 뮤텍스도 사라진다. 뮤텍스를 소유한 쓰레가 해당 뮤텍스를 해제할 수 있다.

세마포어는 공유자원에 여러 프로세스가 접근하는 것을 막는 것이다. 깃발이라는 뜻의 독일어다. 세마포어는 사용

가능한 공유자원의 개수를 나타내는 카운터다. 카운터가 0이되면 프로세스는 해당 자원에 접근할 수 없다. 이진 세마포어는 뮤텍스처럼 사용할 수 있다. 세마포어는 파일형태로 존재한다. 세마포어를 소유하고 있지 않은 쓰레드도 세마포어를 해제할 수 있다.