1. **메모리 관리 개관**

개관

**메모리(주기억장치, main memory)의 구성과 관리**

시스템 성능 학습 목표

=> 사용 가능한 메모리의 용량과 프로세스 처리 중 메모리를 얼마나 효과적으로 사용하는가?

1. **프로세스와 메모리의 관계를 이해한다.**
2. **기억장치 계층구조를 이해한다.**
3. **다중 프로그래밍의 의미를 이해한다.**
4. **고정 분할과 동적 분할을 이해한다.**
5. **메모리 배치기법을 이해하고 적용할 수 있다. 주요 용어**

기억장치 계층구조 연속 메모리 할당 다중 프로그래밍

**단편화(fragmentation) 통합**

집약

* 1. **프로세스와 메모리 [113]**

메모리

프로세스 동작을 위한 자원 필수 요소

**CPU, 메모리 선택 요소**

입출력장치, 파일 등 프로세스가 실행상태에서 하는 동작

프로그램 카운터(PC)를 참조하여 수행될 명령어를 읽어 와서 CPU의 해당 명령을 수행하는 것 프로그램 카운터가 가리키는 주소 = 메모리상의 특정 위치

프로세스를 메모리에 다 가져올 수 없는 경우

1) SSD나 하드 디스크 같은 보조기억장치를 활용하여 조기억장치로 옮기는 형태

**캐시 메모리 vs. 메모리 vs. 보조기억장치 비교**

* 1. **단일 프로그래밍 환경 [115]**

초기 시스템

2) 필요한 순간에만 메모리로 가져오고 3) 그렇지 않은 때는 다시 보

**1) 오직 하나의 프로세스만 전용으로 사용 2) 나머지 사용자는 기다림 연속 메모리 할당 (contiguous memory allocation)**

하나의 연속된 블록으로 메모리에 할당 단점

메모리의 용량을 초과하는 프로세스는 실행 될 수 없다.

메모리의 낭비가 심하다. : 지속적으로 사용되지 않는 프로세스도 메모리에 계속 적재된다. 자원의 낭비가 심하다 : 한 명의 사용자가 메모리를 전용하므로 주변장치 등 낭비가 심하다.

**CPU 작업 중에는 입출력장치가 대기, 입출력을 수행하는 중에는 CPU가 대기해야 한다.**

* 1. **다중 프로그래밍 환경**

**다중 프로그래밍 (멀티 프로그래밍, multi programming)**

여러 개의 프로세스가 메모리에 동시에 적재되는 것

현재 실행 중인 프로세스가 입출력 대기를 해야 하면 실행을 기다리고 있는 다른 프로세스에 CPU를 할당할 수 있다. CPU 연산과 입출력을 동시에 함

**CPU 이용도 증가**

**시스템 처리량(throughput) 증가**

* + 1. **메모리 분할 메모리 분할**

여러 프로세스를 메모리에 적재하기 위한 방법으로, 하나의 분할에 하나의 프로세스가 적재되는 방식 고정 분할

메모리를 여러 개의 고정된 크기 영역으로 분할

**분할 1은 주소 a부터 25MB, 분할 2는 주소 b부터 40MB, ...**

분할 영역에 프로세스를 배치하는 방법

분할영역마다 큐를 두고 큐에 들어온 프로세스는 해당 분할 영역에만 적재 프로그램 컴파일 시 프로그램 내의 주소를 "절대주소"로 번역

특징

구현 용이

큐가 빈 분할이 있어도 다른 큐의 프로세스는 적재할 수 없어 효율성 떨어짐

메모리 전체에 하나의 큐만 두고 모든 프로세스를 하나의 작업 큐에 넣어서 어느 분할에서든지 실행 가능하게 하는 것 프로그램 컴파일 시 프로그램 내의 주소를 "상대주소"로 번역하여 어디든 적재가 가능한 방식

분할에 적재될 작업의 주소 지정 : 재배치 컴파일러, 어셈블러, 로더 등으로 해결 특징

기억 장치 낭비 줄임

**재배치 가능 번역기와 로더는 절대 번역기보다 복잡 (구현 어려움) 내부 단편화(fragmentation)**

고정 분할 방식에서 프로세스의 크기가 적재된 분할영역의 크기보다 작아서 분할영역 내에 남게 되는 메모리는 결국 낭비되 는 부분이 된다.

**[그림 8-8] 프로세스 적재 후 19MB가 남지만, 사용할 수 없음 동적 분할**

메모리의 분할 경계가 고정되지 않고 각 프로세스에 필요한 만큼의 메모리만 할당

필요한 시점에 필요한 만큼의 메모리만 할당받아 내부 단편화 문제는 일어나지 않는다. 외부 단편화 문제가 발생한다.

외부 단편화

메모리의 할당과 반환이 계속 반복됨에 따라 작은 크기의 공백이 메모리 공간에 흩어져 생기는 것 외부 단편화를 해결하는 방법

# 통합(coalescing)

인접된 공백을 더 큰 하나의 공백으로 만드는 과정

비어 있는 메모리가 인접해 있는 부분 조사하여, 인접한 경우 빈 공간 리스트에 새로운 공백이나 기존의 공백과 합쳐 하 나의 공백을 만든다.

한계

통합된 여러 공백의 합은 자원을 요구하는 프로세스 크기보다 크지만, 외부 파편화로 공백 부분이 나뉘어 가장 큰 파편화 메모리만으로 해결 안되는 경우 존재

# 집약(compaction)

메모리 내의 모든 공백을 하나로 모으는 작업

동적 분할에서 통상적으로 발생하는 여러 개의 작은 공백을 하나의 커다란 저장공간으로 만드는 것이다.

* + 1. **메모리 보호**

여러 프로세스가 동시에 메모리에 상주해야 하므로 (각각 프로세스 메모리 경계를 구분짓는) 메모리 보호가 필요하다. 연속 메모리 할당 방식에서의 메모리 보호

프로세스가 사용할 수 있는 주소 범위를 할당영역을 침범하지 않게 한다.

1) 하한-상한 레지스터 쌍 또는 2) 하한-크기 레지스터 쌍의 값으로 제한하여 다른

프로세스가 이 제한을 넘어 운영체제를 호출하려면? 시스템 호출을 통해서만 가능하다

* 1. **메모리 배치기법 [122]**

메모리 배치 기법

동적 분할 다중 프로그래밍에서 새로 반입된 프로그램이나 데이터를 메모리의 어느 위치에 배치할 것인가를 결정하는 방법 운영체제는 "빈 공간 리스트"를 유지하고 있어서 그중 적합한 공간을 찾으면 된다.

* + 1. **최초 적합(first-fit)**

프로세스가 적재될 수 있는 빈 공간 중에서 가장 먼저 발견되는 곳을 할당

빈 공간 리스트를 메모리 주소순으로 유지하며 할당속도를 빠르게 할 수 있다.

* + 1. **후속 적합(next-fit)**

이전에 탐색이 끝난 그다음 부분부터 시작하여 사용 가능한 빈 공간 중에서 가장 먼저 발견되는 곳을 할당 위 그림에서 빈 공간 리스트에서 공백 3은 나머지 2MB를 가리키도록 업데이트 한다

* + 1. **최적 적합(best-fit)**

필요한 공간을 제공할 수 있는 빈 공간 중 가장 작은 곳을 선택하여 할당 빈 공간 리스트에서 공백 3은 공백 5를 가르키도록 업데이트함

큰 빈 공간 최대한 많이 남겨 놓기 위한 방법

* + 1. **최악 적합(worst-fit)**

필요한 공간을 제공할 수 있는 빈 공간 중 가장 큰 곳을 선택하여 할당 작은 자투리가 남아 사용되는 못하는 공간이 발생하는 것을 최소화

빈 공간 리스트에서 공백 5는 나머지 15MB를 가리키도록 업데이트함