운영체제

Operating System

* 컴퓨터 사용자와 하드웨어 간의 인터페이스를 담당하는 소프트웨어
* 컴퓨터를 편하게 사용할 수 있도록 도와준다
* 자원 관리를 통해 하드웨어를 조금 더 효율적으로 사용할 수 있도록 도와준다

Kernel

* 운영체제의 핵심부분
* 메모리에 상주한다
* 응용 프로그램 또는 운영체제의 다른 부분이 수행에 필요한 환경을 설정할 수 있다

운영체제의 유형

1. 다중 프로그래밍 시스템 : 1개의 프로세서가 여러 개의 일을 수행
2. 시분할 시스템 : 1개의 프로세서가 주어진 시간을 나눠서 여러 개의 일을 수행
3. 분산처리 시스템 : 여러 개의 프로세서가 하나의 일을 나눠서 수행
4. 다중처리 시스템 : 여러 개의 프로세서가 다수의 일을 동시에 수행

다중 프로그래밍과 다중처리의 차이점은 프로세서를 몇 개 두고 일을 수행할 수 있는지

운영체제의 기능

메모리, 프로세스, 파일 등 시스템 구성요소를 제공하는데, 이 시스템 구성요소를 자원이라 하며 자원을 관리하는데 특화되어 있다

메모리 관리

* 현재 메모리의 어느 부분이 사용되고, 누가 사용하는 지를 점검
* 기억 공간에 어떤 프로세스를 저장할 지를 점검
* 기억 공간을 할당하고 회수하는 방법 결정

보조 기억장치 관리

* 비어 있는 공관 관리
* 저장 장소 할당
* 디스크 스케줄링

프로세스 관리

* 프로세스와 스레드 스케줄링
* CPU와 프로세스의 상태를 추적 저장한다 이는 트래픽 제어기가 수행
* CPU 할당 및 회수

파일 관리

* 파일 생성과 제거
* 디렉터리 생성과 제거
* 보조 기억에 있는 파일의 맵핑
* 안전한 저장 매체에 파일 저장

입출력

* 중앙처리장치와 비동기적으로 수행
* 수행상의 시간관계 규정이 없이 병행적으로 수행

버퍼링

* 입출력장치는 중앙처리장치와 비교할 때 매우 느린 속도
* 입출력장치의 느린 속도를 보완하는 방법
* 레코드가 읽혀 중앙처리장치가 연산을 시작할 때 버퍼링 수행
* 다음에 필요한 레코드를 미리 읽어서 주기억장치에 저장
* 주기억장치의 일부를 버퍼라 하고 일력의 과정을 버퍼링이라 한다

스풀링

* 버퍼링은 주기억장치를 버퍼로 사용한다
* 스풀링은 디스크를 매우 큰 버퍼처럼 사용하는 방법
* 다중 프로그래밍 환경에서 발견할 수 있다
* 다수의 프로세스가 입출력장치를 요구하나 제한 되어있을 때 공유를 가능하게 하기 위함

Interrupt

* 시스템에 예기치 않은 상황이 발생하였을 때, 운영체제에 알리기 위한 메커니즘

Context Switching

* 실행 중인 프로그램의 정보는 PCB에 저장된다.
* 실행할 프로그램의 정보는 레지스터로 적재된다.
* 이와 같은 정보의 교환을 Context Switching이라 한다

PCB

* 운영체제가 프로세스에 대한 중요한 정보를 저장하는 메모리 블록
* 프로세스의 상태, 이름, 우선 순위, 프로그램 카운터 등을 저장한다
* 프로세스가 생성되면 주기억장치의 시스템 영역에 생성되며 종료 시 없어진다

프로세스와 스레드 관리

프로세스

* 실행 중인 프로그램
* PCB를 지니는 프로그램
* 메모리 영역(코드, 데이터, 힙, 스택)을 독립적으로 운영

스레드

* 프로세스에서 실행 흐름을 구성하는 한 단위
* 스택 영역을 제외한 나머지 메모리 영역을 공유하고 있다
* 공유하고 있는 영역의 데이터들에 대한 동기화 수행 필요

프로세스 스케줄링

* 처리율(throuput)의 최대화와 응답 시간(response time)의 최소화를 고려
* 실행의 무한 지연을 피해야 하며 발생할 경우 aging을 통해 해결

비선점 스케줄링

1. FCFS First Come First Served
   * 호위 효과(burst 시간이 긴 하나의 프로세스가 cpu양도할 때까지 기다리는 현상)
2. SJF Shortest Job First
   * 기아 상태 유발
3. HRN High Response ratio Next

선점 스케줄링

1. SRT Shortest Remaining Time First
   * 기아 상태 유발
2. Round Robin
3. Multilevel Queue
   * 기아 상태 유발
   * 해결하기 위한 Multilevel Feedback Queue

반환 시간 = 대기 시간 + CPU 사용 시간

대기 시간 = 작업이 처리되기 전까지 기다린 시간

기억장치 관리

* 여러 개의 프로세스들이 적재되어 있어야 한다

기억장치의 계층 구조

1. 레지스터
2. 캐시 : 주기억장치에서 자주 참조되거나 실행되는 데이터나 프로그램 저장
3. 주기억장치
4. 보조기억장치

주소 바인딩

* 프로그램은 실제 실행될 주기억장치의 물리적 주소를 사용하지 않는다
* 주소를 변수로 표현하는데 이를 논리적 주소라 한다.
* 실행되기 위해서는 논리적 주소를 물리적 주소로 Mapping 해야 한다.

물리적 주소

* 기억장치가 취급하는 주소

논리적 주소

* CPU가 생성하는 주소
* 가상 주소
* 프로그램 수행 중에 물리적 주소로 변환해야 하는데 MMU에 의해 수행

Overlay 기법

* 주기억장치보다 큰 프로그램의 실행을 수행할 수 있게 도와주는 방법

메모리 분할

1. 고정 분할 기억장치 할당
2. 가변 분할 기억장치 할당
   * 고정 단위 : 미리 여러 개의 크기로 자른 후 적당한 크기의 메모리 할당
   * 버디 시스템 : 메모리 분할 시 클 경우 절반만큼 잘라가며 메모리 할당
   * 가변 단위 : 프로그램 크기에 맞추어 메모리 할당

단편화의 발생

* 프로그램의 크기에 딱 맞추어 매번 할당할 수 없어 단편화 발생
* 내부 단편화 외부 단편화 존재
* 내부 단편화란 할당한 메모리 크기가 클 경우 비사용 공간이 생긴 현상
* 외부 단편화란 이용 가능하지만 메모리가 띄엄띄엄 존재하여 메모리 할당 불가 상태

단편화의 해결

1. 통합(coalescing) 인접한 공백 공간을 하나의 큰 공간으로 합치는 과정
2. 압축(compaction) 사용한 공간을 한쪽으로 모음으로써 공백 공간을 합치는 과정

가상 메모리 관리

* 주기억장치의 이용 가능한 기억 공간보다 훨씬 큰 주소 지정을 할 수 있도록 도와준다
* 주기억장치보다 크기가 큰 프로세스를 수행시킬 수 있다
* 가상 주소 공간을 구성하는 것을 가상 메모리
* 실제 주소 공간을 구성하는 것을 실제 메모리 또는 주기억장치라 한다.
* 가상 주소와 실제 주소를 매핑하기 위한 대표적인 메커니즘으로 동적 주소 변환이 있다

블록 사상

* 같은 크기로 할 때 이를 페이지, 다른 크기로 할 때 세그먼트라 한다
* 페이지와 관련된 가상 메모리 구성을 페이징이라 한다
* 세그먼트와 관련된 가상 메모리 구성을 세그먼테이션이라고 한다

페이징

* 실제 주소 공간을 페이지 크기와 같은 페이지 프레임으로 나누어 사용한다
* 페이지 프레임은 주기억장치에 자리잡는다

세그먼테이션

* 고정된 단위가 아닌 서로 논리적으로 관련있는 정보의 단위로 분할

Page fault

* 페이지를 참조하는 과정에서 원하는 페이지가 주기억장치 내에 없는 경우

페이지 교체 알고리즘

1. FIFO
2. Optimal 비현실적
3. LRU(Least Recently Used)
4. LFU(Least Frequently Used)

Thrashing

* 페이지 부재가 계속적으로 발생되어 프로세스 수행시간 보다 페이지 교체시간이 긴 경우

Locality

1. Temporal locality : 최근에 참조된 메모리가 다시 참조될 가능성이 높음을 의미
2. Spatial locality : 메모리 레퍼런스 시 특정 메모리 주위를 집중적으로 참조함을 의미

Working set

* 프로세스에 의해 자주 참조되는 페이지들의 집합체

디스크 관리

디스크 구조

* 데이터들은 헤드에 의하여 기록되거나 읽혀진다
* 디스크 구조에서 부채꼴 모양으로 자른 것을 섹터
* 중심축에 대해 동심원으로 나누어진 것을 트랙
* 섹터와 트랙의 교차점으로 둘러싸인 구역을 블록
* 헤드가 한 번에 판독/기록할 수 있는 원통형의 트랙군을 실린더

디스크에서 데이터 처리

* 헤드가 해당 위치로 옮겨져야 한다
* 실제 원하는 실린더를 찾는데 소요되는 시간을 탐색 시간 Seek Time
* 헤드가 옮겨진 후 회전하면서 데이터를 찾는 시간을 회전 지연 시간 Latency Time
* 데이터를 주기억장치 까지 전달하는 시간을 전송 시간 Transmission time
* 디스크 접근 시간 = Seek Time + Latency Time + Transmission Time

디스크 스케줄링

* 탐색 시간이 많이 소요되기 때문에 이를 최소화 하는데 중점

1. FCFS
2. SSTF
3. SCAN 및 LOOK
4. C-SCAN 및 C-LOOK SCAN 방식을 같은 방향으로 계속 처리하는 방법