1. 운영체제(Operating System: OS)

1-1. 운영체제의 정의

운영체제는 하드웨어와 직접적으로 연관되어 시스템 자원(컴퓨터 시스템 자원: CPU, 주기억장치, 보조기억장치, 프린터, 파일 및 정보 등)를 효율적으로 관리하며, 드라이버와 같은 응용 소프트웨어를 실행하는 등 사용자가 컴퓨터를 쉽게 사용할 수 있도록 환경을 제공하는 시스템 소프트웨어이다.

※ 자원(Resource) : 컴퓨터를 이루는 모든 요소들로 본체, 키보드, 모니터, 마우스, 프린터, 네트워크 카드들을 말하며, 본체에는 CPU, 메모리, 하드디스크 등이 포함된다.

1-2. 운영체제의 목적(운영체제를 사용하는 이유)

운영체제의 목적에는 처리 능력 향상, 사용 가능도 향상, 신뢰도 향상, 반환 시간 단축 등이 있으며, 운영체제의 성능을 평가하는 기준이 된다.

처리능력(Throughput)	일정 시간 동안 시스템이 처리하는 일의 양
반환 시간 (Turn Around Time)	시스템에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간
사용 가능도(Availability)	시스템을 사용할 필요가 있을 때 즉시 사용 가능한 정도
신뢰도(Reliability)	시스템이 주어진 문제를 정확하게 해결하는 정도

1-3. 운영체제의 주요 기능

가. 프로세서(처리기, Processor), 기억장치(주기억장치, 보조기억장치), 입 · 출력장치, 파일 및 정보 등의 자원 관리

나. 자원을 효율적으로 관리하기 위한 자원의 스케쥴링 기능 제공
※ 스캐줄링 : 어떤 자원을 누가, 언제, 어떤 방식으로 사용할지를 결정해 주는 것

다. 사용자와 시스템 간의 편리한 인터페이스 제공.
라. 시스템의 각종 하드웨어 관리, 제어

마. 네트워크 관리, 제어
바. 데이터의 관리 및 데이터, 자원의 공유 기능 제공
사. 시스템의 오류 검사, 복구
아. 자원 보호 기능 제공
자. 입·출력에 대한 보조 기능 제공

1-4. 운영체제의 주요 자원 관리

운영체제에서는 다음과 같은 자원의 관리 기능을 수행한다.

자 원	기 능
프로세스	- 프로세스 스케줄링 및 동기화 관리 - 프로세스 생성과 제거, 시작과 정지, 메시지 전달
기억장치	- 프로세스에게 메모리 할당 및 회수 관리
주변장치	- 입·출력장치 스케줄링 및 전반적인 관리
파일	- 파일의 생성과 삭제, 변경, 유지 등의 관리

1-5. 운영체제의 종류

- 개인용 컴퓨터 운영체제 : DOS, Windows, LINUX, Mac OS
- 서버용 운영체제 : Windows NT, UNIX, LINUX SERVER

- 단일 작업 처리 시스템 : DOS
- 다중 작업 처리 시스템 : Windows, UNIX, LINUX

단일 처리 시스템	컴퓨터 시스템을 한 개의 작업이 독점하여 사용하는 방식으로, 예를 들어 워드 작업을 하다가 PC 통신을 하려면 워드 작업을 종료해야 한다.
다중 처리 시스템	여러 개의 프로그램을 열어 두고 다양한 작업을 동시에 진행하는 방식이다.

가. MS-DOS(Microsoft-Disk Operating System)

디스크, 출력장치, 입력장치, 인쇄장치 등 퍼스널컴퓨터의 여러 장치들을 함께 제어해주는 마이크로소프트사(社)에서 개발한 범용 운영체제로 개인용 컴퓨터(PC)의 가장 대표적인 운영체제로, 1981년에 IBM사(社)가 16비트 PC를 발매할 무렵에 마이크로소프트사가 IBM PC용으로 개발한 단일 이용자용 및 단일 태스크용의 운영체제이다. 16비트뿐만 아니라 32비트 PC용의 대표적인 OS로 1983년에 발표된 MS-DOS 2.0판에서는 유닉스적인 기능을 도입하였다. 계층 구조에 의한 파일 관리, 명령의 파이프·필터 기능, 네트워크 기능, MS-Windows와의 조합에 의한 GUI 환경 등이 도입되고, 기억 용량이 640KB로 확장되어 활용도가 향상되었다.

나. 윈도우즈(Windows)

도스를 대체한 Microsoft의 운영체제로 1983년 버전 1.0이 최초로 탄생되었다. 그후 90년 3.0, 92년

3.1이 이어지고 95년 윈도우 95, 98년 윈도우 98이 출시되었다. 윈도우 3.1의 경우 완벽한 운영체제가 아니었지만 윈도우 95부터 완전한 운영체제로 도스를 대처할 수 있게 되었다. 윈도우 3.1은 반쪽짜리 운영체제로 도스가 없이는 동작하지 못한다. 즉 윈도우 3.1 만으로는 운영체제가 설치조차 되지 않을 뿐 아니라 도스로 부팅한 후에 윈도우 3.1을 실행해야 한다. 전체적인 구조도 도스에 의존하는 바가 너무 컸다. 반면 윈도우 95는 도스 없이 사용이 가능하며 오히려 도스가 윈도우에 흡수되었는데 이렇게 포함된 도스는 7.0 버전이다. 윈도우 2000, 윈도우98, 윈도우ME, 윈도우XP, 윈도우7, 윈도우 비스타, 윈도우8, 윈도우10 이 있다.

다. 리눅스(LINUX)

워크스테이션이나 개인용 컴퓨터에서 주로 사용되는 유닉스(UNIX)와 유사한 운영체제로 1989년 핀란드 헬싱키대학에 재학중이던 리누스 토르발스(Linus Torvalds)가 유닉스를 기반으로 개발한 공개용 운영체제이다. 1991년 11월 버전 0.10이 일반에 공개되면서 확대 보급되기 시작하였다. 유닉스가 중대형 컴퓨터에서 주로 사용되는 것과는 달리, 리눅스는 워크스테이션이나 개인용 컴퓨터에서 사용할 수 있다. 리눅스는 소스 코드를 완전 무료로 공개하여 전 세계적으로 약 5백만 명이 넘는 프로그램 개발자 그룹을 형성하게 되었으며, 이들에 의해 단일 운영체제의 독점이 아닌 다수를 위한 공개라는 원칙하에 지속적인 업그레이드가 이루어지고 있다. 파일구성이나 시스템기능의 일부는 유닉스를 기반으로 하면서, 핵심 커널 부분은 유닉스와 다르게 작성되어 있다.
인터넷 프로토콜인 TCP/IP를 강력하게 지원하는 등 네트워킹에 특히 강점을 지니고 있으며, 유닉스와 거의 유사한 환경을 제공하면서 무료라는 장점 때문에 프로그램 개발자 및 학교 등을 중심으로 급속히 사용이 확대되고 있다. 리눅스는 각종 주변기기에 따라 혹은 사용하는 시스템의 특성에 맞게 소스를 변경할 수 있으므로 다양한 변종이 출현하고 있다.

*** LINUX 시스템의 구성**

유 틸 리 티 (Utility)	- 일반 사용자가 작성한 응용프로그램을 처리하는 데 사용한다. - DOS에서의 외부 명령어에 해당된다. - 유틸리티 프로그램에는 에디터, 컴파일러, 인터프리터, 디버거 등이 있다.
쉘(Shell)	- 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기이다. - 시스템과 사용자 간의 인터페이스를 담당한다. - DOS의 COMMAND와 같은 기능을 수행한다. - 주기억장치에 상주하지 않고, 명령어가 포함된 파일 형태로 존재하며 보조 기억장치에서 교체 처리가 가능하다.
커널(Kernel)	- 가장 핵심적인 부분으로 컴퓨터가 부팅될 때 주기억장치에 적재된 후 상주하면서 실행한다. - 하드웨어를 보호하고, 프로그램과 하드웨어 간의 인터페이스 역할을 담당한다. - 프로세스(CPU 스케줄링) 관리, 기억장치 관리, 파일 관리, 입 · 출력 관리, 프로세스간 통신, 데이터 전송 및 변환 등 여러 가지 기능을 수행한다.
하드웨어	- CPU, 메모리, 스토리지, 입출력장치

라. 유닉스(UNIX)

1969년 미국의 전신전화회사 AT&T사의 벨연구소에서 개발한 오퍼레이팅시스템으로 켄 톰슨(Ken

Thompson)이 DEC사의 미니 컴퓨터 PDP-7에서 어셈블리언어로 단일 사용자용 유닉스의 제1버전을 개발하였으며, 1972년 데니스 리치(Dennis Ritchie)가 고급언어인 C언어로 다시 작성하였다. 유닉스를 탑재한 워크스테이션의 발매와 함께 보급되어 현재는 개인용 컴퓨터, 대형 컴퓨터, 마이크로 컴퓨터에 이르기까지 많은 종류의 컴퓨터에서 사용되고 있으며, 거의 어떤 컴퓨터에도 이식이 가능하다. 유닉스는 멀티태스킹, 멀티유저를 지원하는 운영체제로 프로그램개발, 문서처리, 전자우편 등의 기능이 뛰어나다. 지금까지 유닉스 시스템은 크게 두 부류로 발전되어 왔다. 하나는 AT&T사의 유닉스 시스템 시리즈이고, 다른 하나는 버클리 대학에서 만든 BSD(Berkeley Software Distribution) 유닉스이다. 이들은 각각 독자적인 기능을 부여하여, 같은 유닉스지만 호환성이 없고 이식성이 떨어지는 혼란을 가져왔다. 이러한 유닉스의 혼란을 방지하기 위하여 미국의 유닉스 사용자 모임은 1984년 표준화 위원회를 설립하여 유닉스의 표준화를 시도하였다. 이러한 표준화작업으로 유닉스 사용자는 시스템마다 프로그램을 변경하지 않고 실행할 수 있으며, 사용자 인터페이스를 공유하여 공통된 환경을 사용할 수 있는 등 많은 이익을 얻게 되었다.

1-6. 운영체제의 특징

가. MS-DOS

- CUI(Character User Interface, 문자 중심의 사용자 인터페이스) : 작업을 위한 실행 명령을 키보드를 이용해서 명령 문자(Character)로 직접 입력하여 실행
- 싱글 유저(Single-User): 하나의 컴퓨터를 한 사람만이 사용한다.
- 싱글 태스킹(Single-Tasking): 한 번에 하나의 프로그램만을 수행한다.
- 파일 시스템의 디렉터리 구조는 트리 구조이다.

나. 윈도우(Windows)

- GUI(Graphic User Interface, 그래픽 사용자 인터페이스) : 키보드로 명령어를 직접 입력하지 않고, 마우스로 아이콘이나 메뉴를 선택하여 모든 작업을 수행

- 선점형 멀티 태스킹(Preemptive Multi-Tasking) : 동시에 여러 개의 프로그램을 실행하는 멀티태스킹을 하면서 운영체제가 각 작업의 CPU 이용 시간을 제어하여 응용 프로그램 실행중 문제가 발생하면 해당 프로그램을 강제 종료시키고 모든 시스템 자원을 반환하는 방식이다.

- 메시지 구동 시스템 : 응용 프로그램은 입력 장치로부터 키나 마우스의 입력을 직접 읽을 수 없으며 운영체제가 대신 입력을 받아 전달해 주는 방식을 사용한다. 이때 입력 장치로부터의 입력 신호를 메시지라 하며 응용 프로그램은 메시지를 받아 동작한다. 이런 동작 방식을 메시지 구동 시스템 또는 이벤트 드리븐 시스템 이라고 한다.

- 장치에 독립적 : Windows OS는 디바이스 드라이버(Device Driver)에 의해 다양한 주변 장치들을 제어하고 관리한다. 장치가 바뀌면 디바이스 드라이버를 교체하여 응용 프로그램과 연결하도록 설계되어 있어 응용 프로그램에는 영향을 미치지 않는다.

- 일관성 : 사용자가 프로그램에게 명령을 내리는 인터페이스 구성이 표준화되어 있다. 대화상자나 메뉴, 툴바도 운영체제가 직접 지원한다. 응용 프로그램 개발자들은 이런 인터페이스를 구현하지 않고 운영체제가 제공하는 표준 인터페이스를 활용하기만 하면 된다.

- 리소스가 분리되어 있다. : 리소스(Resource)란 프로그램에서 필요로 하는 메뉴, 비트맵, 아이콘 등을 말한다. 윈도우즈 응용 프로그램은 코드와 리소스가 분리되어 있어 개발자와 디자이너가 분담 작업을 쉽게 할 수 있다.

다. 리눅스(LINUX)

- 시분할 시스템을 위해 설계된 대화식 운영체제로 소스가 공개된 개방형 시스템(Open System)이다.
- C 언어로 작성되어 있어 이식성이 높으며 장치, 프로세스 간의 호환성이 높다.

- 다중 사용자(Multi-User), 다중 작업(Multi-Tasking)을 지원하며, 크기가 작고 이해하기가 쉽다.

※ 다중 사용자는 여러 사용자가 동시에 시스템을 사용하는 것이고, 다중 작업은 여러 개의 작업(프로그램)을 동시에 수행하는 것을 말한다.

- 뛰어난 네트워킹 기능을 제공하므로 통신망(Network) 관리용 운영체제로 적합하다.
- 트리 구조의 파일 시스템을 갖는다.

- 전문적인 프로그램 개발에 용이하고 다양한 유틸리티 프로그램들이 존재한다.

※ 리눅스는 유닉스와 완벽하게 호환된다.

리눅스의 보급이 다른 운영체제보다 빨랐던 이유 중 하나는 유닉스와 호환된다는 점이다. 유닉스는 워크스테이션용 운영체제로 대학이나 기업, 연구기관에서 주로 사용되었다. 하지만 가격이 비싸 개인이 사용하기에는 무리가 있었다. 리눅스는 높은 성능의 유닉스용 프로그램을 동작시키고, 활용할 수 있어 유닉스의 성능을 무료로 사용할 수 있다.

※ 리눅스는 공개 소스부터 공개되어 있는 운영체제이며, 무료이다.

리눅스의 개방성 또한 큰 장점이라고 할 수 있다. 많은 우수 인력이 확보되어 있기 때문에 우수한 소프트웨어 개발이 가능하고 여러 배포판 개발 업체들이 있기 때문에 사용자에게 선택권이 주어진다. 소스코드가 공개되기 때문에 우수한 코드만이 살아남을 수 있게 된다.

※ 리눅스는 PC용 OS(Window) 보다 안정적이다.

PC는 업무가 끝나면 전원을 끄지만 리눅스는 네트워크 사용을 전제로 항상 가동되도록 설계되었기에 안정적이라고 할 수 있다.

2. 운영체제의 운용 기법

2-1. 운영체제 운용 기법의 발달 과정

가. 일괄 처리 시스템(Batch Processing System)

일괄 처리 시스템은 일정량 또는 일정한 시간(하루, 일주일, 한달 등) 동안 데이터를 모아서 한꺼번에 처리하는 방식이다.

나. 다중 프로그래밍 시스템(Multi-Programming System)

다중 프로그래밍 시스템은 하나의 CPU와 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식으로 CPU의 사용률과 처리량이 증가한다.

다. 시분할 시스템(Time Sharing System)

시분할 시스템은 여러 명의 사용자가 사용하는 시스템에서 컴퓨터가 사용자들의 프로그램을 번갈아가며 처리해 줌으로써 각 사용자에게 독립된 컴퓨터를 사용하는 느낌을 주는 것으로 라운드 로빈(Round Robin) 방식이라고도 한다. 하나의 CPU는 같은 시점에서 여러 개의 작업을 동시에 수행할 수 없기 때문에, CPU의 전체 사용 시간을 작은 작업 시간 량으로 나누어서 그 시간 량 동안만 번갈아 가면서 CPU를 할당하여 각 작업을 처리한다. 시스템의 전체 효율은 좋아지나 개인별 사용자 입장에서는 반응 속도가 느려질 수 있다.

라. 다중 처리 시스템(Multi-Processing System)

다중 처리 시스템은 여러 개의 CPU와 하나의 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식으로 하나의 CPU가 고장 나더라도 다른 CPU를 이용하여 업무를 처리할 수 있으므로 시스템의 신뢰성과 안정성이 높다. 여러 CPU는 하나의 메모리를 공유하며 단일 운영체제에 의해 관리된다. 프로그램의 처리 속도는 빠르지만 기억장치, 입 · 출력장치 등의 자원 공유에 대한 문제점을 해결해야 한다.

마. 실시간 처리 시스템(Real Time Processing System)

실시간 처리 시스템은 데이터 발생 즉시, 또는 데이터 처리 요구가 있는 즉시 처리하여 결과를 산출하는 방식이다. 처리 시간이 단축되고 처리 비용이 절감된다.

바. 다중 모드 처리(Multi-Mode Processing)

다중 모드 처리는 일괄 처리 시스템, 시분할 시스템, 다중 처리 시스템, 실시간 처리 시스템을 한 시스템에서 모두 제공하는 방식이다.

사. 분산 처리 시스템(Distributed Processing System)

분산 처리 시스템은 여러 개의 컴퓨터(프로세서)를 통신 회선으로 연결하여 하나의 작업을 처리하는 방식이다. 각 단말장치나 컴퓨터 시스템은 고유의 운영체제와 CPU, 메모리를 가지고 있다.

3. 운영체제 기본 명령어

3-1. UNIX/LINUX 기본 명령어

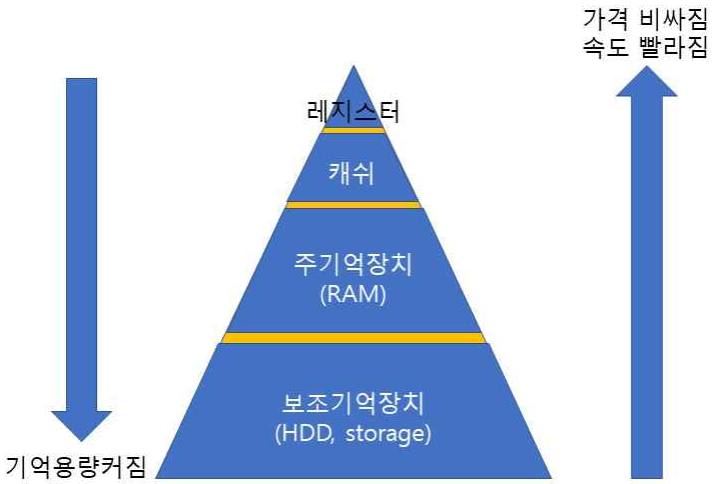
명령어	의 미
ls	현재 디렉터리 내의 파일 목록을 확인한다.
mkdir	디렉터리를 생성한다.
chdir	현재 사용할 디렉터리 위치를 변경한다.
rmdir	디렉터리를 삭제한다.
cp	파일을 복사한다.
mv	파일을 이동시키거나 이름을 변경한다.
rm	파일을 삭제한다.
cat	파일 내용을 화면에 표시한다.
chmod	파일의 보호 모드를 설정하여 파일의 사용 허가를 지정한다.
chown	소유자를 변경한다.
find	파일을 찾는다.
creat	파일을 생성시킨다.
open	파일을 사용할 수 있는 상태로 준비시킨다.
close	파일을 닫는다.
mknod	특수 파일을 생성한다.
mount/unmount	파일 시스템을 마운팅 한다./마운팅 해제한다.
mkfs	파일 시스템을 생성한다.
fsck	파일 시스템을 검사하고 보수한다.
finger	사용자 정보를 표시한다.

3-2. MS-DOS / Windows cmd 기본 명령어

명령어	의 미
DIR	파일 목록을 표시한다.
MD	디렉터리를 생성한다.
CD	디렉터리 위치를 변경한다.
RD	디렉터리를 삭제한다.
COPY	파일을 복사한다.
DEL	파일을 삭제한다.

명령어	의 미
CLS	화면의 내용을 지운다.
TYPE	파일의 내용을 표시한다.
REN	파일 이름을 변경한다.
PATH	파일의 탐색 경로를 지정한다.
VER	버전을 표시한다.

4. 기억장치 계층 구조



5. 메모리 관리 기법

모든 프로세스는 실행되기 전 메모리에 저장된다. 시스템의 성능을 최대로 높이기 위해 사용할 수 있는 주기억장치 관리 기법이다. 보조기억장치의 프로그램이나 데이터를 주기억장치에 적재시키는 시기, 적재 위치 등을 지정하여 한정된 주기억장치의 공간을 효율적으로 사용하기 위한 것으로 반입 기법(Fetch Strategy), 배치 기법(Placement Strategy), 교체 기법(Replacement Strategy)이 있다.

5-1. 반입 기법(Fetch Strategy)

보조기억장치에 보관중인 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지를 결정하는 전략이다. 즉, 주기억장치에 적재할 다음 프로세스의 반입 시기를 결정하는 방법

가. 요구 반입(Demand Fetch): 필요로 할 때 적재(load)하는 방법
실행중인 프로그램의 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재하는 방법으로 운영체제나 시스템 프로그램, 사용자 프로그램 등의 참조요구가 있을 경우 메인 메모리에 적재하는 방법

나. 예상 반입(Anticipatory Fetch): 미리 적재해 놓는 방법
실행중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법으로 시스템의

요구를 예측하여 미리 메모리에 적재하는 방법

5-2. 배치 기법(Placement Strategy)

반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략으로 최초 적합(First Fit), 최적 적합(Best Fit), 최악 적합(Worst Fit)가 있다.

가. 최초 적합(First Fit)
프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치 시키는 방법

나. 최적 적합(Best Fit)
프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화(주기억장치에 분할된 영역에 프로그램이나 데이터를 할당할 경우 분할된 영역이 프로그램이나 데이터 보다 작거나 커서 생기는 빈 공간)를 가장 적게 남기는 분할 영역에 배치 시키는 방법

다. 최악 적합(Worst Fit)
프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법

5-3. 교체 기법(Replacement Strategy)

교환(재배치) 전략으로 메인 메모리에서 어느 프로세스를 제거 할 것인가를 결정하는 방법으로 시기에 따라, 사용빈도에 따라 결정하는 등 여러 방법이 있다. 교체 전략에는 FIFO, OPT, LRU, LFU, NUR, SCR 등이 있다.

6. 가상기억장치

가상 메모리 또는 가상 기억 장치(문화어: 가상기억기, virtual memory, virtual storage)는 메모리 관리 기법의 하나로, 기계에 실제로 이용 가능한 기억 자원을 이상적으로 추상화하여 사용자들에게 매우 큰 (주) 메모리로 보이게 만드는 것을 말한다.

각 프로그램에 실제 메모리 주소가 아닌 가상의 메모리 주소를 주는 방식이다.

이러한 방식은 멀티태스킹 운영 체제에서 흔히 사용되며, 실제 주기억장치보다 큰 메모리 영역을 제공하는 방법으로도 사용된다.

가상적으로 주어진 주소를 가상 주소(virtual address) 또는 논리 주소(logical address) 라고 하며, 실제 메모리 상에서 유효한 주소를 물리 주소(physical address) 또는 실주소(real address)라고 한다. 가상 주소의 범위를 가상 주소 공간, 물리 주소의 범위를 물리 주소 공간이라고 한다.

가상 주소 공간은 메모리 관리 장치(MMU)에 의해서 물리 주소로 변환된다. 이 덕분에 프로그래머는 가상 주소 공간상에서 프로그램을 짜게 되어 프로그램이나 데이터가 주메모리상에 어떻게 존재하는지를 의식할 필요가 없어진다. 대부분의 현대적 아키텍처와 운영 체제는 가상 메모리 기능을 제공하며,

각 응용 프로그램에 더 적합한 메모리 관리를 위해 어도비 포토샵과 같은 일부 응용 프로그램은 스스로 가상 메모리를 관리하기도 한다.

가상 메모리의 개념은 1957년에 발표되었다. 실제 적용된 것은 맨체스터 대학교가 Atlas용으로 1961년에 개발한 것이 최초이다. 1965년에 MIT가 개발한 멀틱스 시스템 이후 본격적으로 채용되기 시작했다.

가상 메모리는 크게 나누어 세그먼트(segment) 방식과 페이징 방식의 2종류가 있다. 예를 들어 MC68000 시스템에서는 68451(세그먼트(segment) 방식)과 68851(페이징 방식) 두 가지의 MMU가 준비되어 있었다.

6 - 1. 가상기억장치의 개요

가상기억장치는 보조기억장치(하드디스크)의 일부를 주기억장치처럼 사용하는 것으로, 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법이다.
프로그램을 여러 개의 작은 블록 단위로 나누어서 가상기억장치에 보관해 놓고, 프로그램 실행 시 요구되는 블록만 주기억장치에 불연속적으로 할당하여 처리한다.

주기억장치의 용량보다 큰 프로그램을 실행하기 위해 사용한다.
주기억장치의 이용률과 다중 프로그래밍의 효율을 높일 수 있다.
가상기억장치에 저장된 프로그램을 실행하려면 가상기억장치의 주소를 주기억장치의 주소로 바꾸는 주소 변환 작업이 필요하다.
블록 단위로 나누어 사용하므로 연속 할당 방식에서 발생할 수 있는 단편화를 해결할 수 있다.

가상기억장치의 일반적인 구현 방법에는 블록의 종류에 따라 페이징 기법과 세그먼테이션 기법으로 나눌 수 있다.

6-2. 페이징 기법

프로그램을 동일한 크기로 나눈 단위를 페이지라 하며 이 페이지를 블록으로 사용하는 기법

가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램(페이지)을 동일하게 나눠진 주기억장치의 영역(페이지 프레임)에 적재시켜 실행하는 기법이다.

프로그램을 일정한 크기로 나눈 단위를 페이지(Page)라고 하고, 페이지 크기로 일정하게 나누어진 주기억장치의 단위를 페이지 프레임(Page Frame)이라고 한다.

외부 단편화는 발생하지 않으나 내부 단편화는 발생할 수 있다.
주소 변환을 위해서 페이지의 위치 정보를 가지고 있는 페이지 맵 테이블(Page Map Table)이 필요하다.

페이지 맵 테이블 사용으로 비용이 증가되고, 처리 속도가 감소된다.

6-3. 세그먼테이션 기법

프로그램을 가변적인 크기로 나눈 단위를 세그먼트라 하며, 이 세그먼트를 블록으로 사용하는 기법

가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 기법이다.

프로그램을 배열이나 함수 등과 같은 논리적인 크기로 나눈 단위를 세그먼트라고 하며, 각 세그먼트는 고유한 이름과 크기를 갖는다.

기억장치의 사용자 관점을 보존하는 기억장치 관리 기법이다.

세그먼테이션 기법을 이용하는 궁극적인 이유는 기억공간을 절약하기 위해서이다.
주소 변환을 위해서 세그먼트가 존재하는 위치 정보를 가지고 있는 세그먼트 맵 테이블(Segment Map Table)이 필요하다.
세그먼트가 주기억장치에 적재될 때 다른 세그먼트에게 할당된 영역을 침범할 수 없으며, 이를 위해 기억장치 보호키(Storage Protection Key)가 필요하다.

내부 단편화는 발생하지 않으나 외부 단편화는 발생할 수 있다.

7. 프로세스 스케줄링(Process Scheduling)의 기법

7-1. 프로세스(Process)의 정의

프로세스는 일반적으로 프로세서(처리기, CPU)에 의해 처리되는 사용자 프로그램, 시스템 프로그램, 즉 실행중인 프로그램을 의미하며, 작업(Job), 태스크(Task)라고도 한다. 프로세스의 상태는 제출, 접수, 준비, 실행, 대기 상태로 나눌 수 있으며, 이 중 주요 세 가지 상태는 준비, 실행, 대기 상태이다.

7-2. 스케줄링(Scheduling)의 개요

스케줄링은 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업으로 CPU나 자원을 효율적으로 사용하기 위한 정책이다. 프로세스가 생성되어 완료될 때까지 프로세스는 여러 종류의 스케줄링 과정을 거치게 된다. 스케줄링의 종류에는 장기 스케줄링, 중기 스케줄링, 단기 스케줄링이 있다.

7-3. 프로세스 스케줄링의 기법

가. 비선점(Non-preemptive) 스케줄링
- 한 프로세스가 CPU를 할당받으면 다른 프로세스가 강제로 CPU를 점유 못하는 스케줄링 기법이다.
- 짧은 작업을 수행하는 프로세스가 긴 작업이 종료될 때까지 기다려야 한다.
- 모든 프로세스들에게 공정하고 응답시간의 예측이 가능하다.
- 일괄 처리 방식에 적합하다.

나. 선점(Preemptive) 스케줄링

- 한 프로세스가 CPU를 차지하고 있을 때 우선순위가 높은 다른 프로세스가 현재 프로세스를 중지시키고 자신이 CPU를 차지할 수 있는 스케줄링 기법이다.

- 높은 우선순위를 가진 프로세스들이 빠른 처리를 요구하는 시스템에서 유용하다.
- 빠른 응답시간을 요구하는 대화식 시분할 시스템에 유용하다.

- 높은 우선순위 프로세스들이 들어오는 경우 많은 오버헤드(Overhead: 간접적인 처리 시간)를 초래한다.