운영체제의 종류에 대한 조사

지도교수 : 이 호
작 성 자 : 서 진 호

< 목 차 >

요 약

오퍼레이팅 시스템(OS)라고도 불리는 운영체제는 사용자가 컴퓨터를 사용할 수 있도록 중재 역할을 해주는 소프트웨어이다. 운영체제는 어떤 사람이 시스템을 사용하고 어떤 식으로 사용하는지를 관리하므로, 컴퓨터 시스템을 총괄하는 머리라고 볼 수 있다. 이를 쉽게 요약하자면 사용자가 컴퓨터를 쉽게 다룰 수 있도록 해주는 인터페이스이다.

주요어 : OS, 운영체제, 프로그램, 소프트웨어

1. 서론

오퍼레이팅 시스템(OS)라고도 불리는 운영체제는 컴퓨터의 하드웨어를 제어하고 응용 소프트웨어를 위한 기반 환경을 제공하여, 사용자가 컴퓨터를 사용할 수 있도록 중재 역할을 해주는 소프트웨어를 말한다. 일련의 작업 순서를 정하고 중앙처리장치(CPU), 주기억장치, 주변장치 등의 여러 하드웨어 시스템에 이를 할당하는 일련의 매우 복잡한 명령으로서, 프로그램 실행은 물론 파일 접근, 응용 프로그램 구동, 모니터 및 메모리 저장장치 제어, 글자판 명령 해석과 같은 특별한 임무를 수행하도록 CPU에 지시한다. 또한 여러 사용자가 동시에 작업을 수행할 때에는 이른바 시분할(time-sharing) 방식으로 작업의 우선순위를 정해 시간과 자원을 효율적으로 배분하며, 네트워크상에서는 다른 컴퓨터와 상호 작용하는 일을 제어하기도 한다. 운영체제의 종류에는 일괄처리(batch processing) 운영체제, 대화형 운영체제, 실시간 운영체제, 하이브리드 운영체제가 있는데, 일괄처리 운영체제는 작업을 모아서 처리, 사용자와 상호작용 없이 순차적으로 실행한다. 대화형 운영체제는 시분할시스템이라고도 하며 일괄처리 시보다 반환시간이 빠르고 이용자에게 즉각적인 피드백(feedback)을 제공한다. 응답시간은 사용자 수에 따라 수 분, 혹은 수십 초가 걸리기도 한다. 실시간 운영체제는 모든 시스템 중 가장 빠른 응답시간을 보이며, 결과 값이 현재의 결정에 영향을 받으며, 데이터의 처리가 매우 빠르고 반환시간이 매우 중요한 환경에 적합하다. 하이브리드 운영체제는 일괄처리와 대화형 처리를 합성한 운영체계로 대화형 작업이 많지 않을 경우 백그라운드에서 배치 프로그램을 실행한다. 현재 사용되고 있는 대부분의 컴퓨터 시스템은 하이브리드 시스템이다.

2. 운영체제의 개요

2.1 운영체제의 역사

초기의 컴퓨터들은 계산기처럼 일련의 단일 작업들을 수행하기 위하여 만들어졌다. 여러 프로그램들을 연속으로 자동 실행하여 처리 속도를 높일 수 있었던 레지던트 모니터와 같이, 1950년대에는 기본적인 운영체제의 기능들이 개발되었다. 운영체제는 1960년대 초까지만 하여도 현대의 운영체제와 같이 더 복잡한 형태로 존재하지 않았다. 하드웨어 기능에 런타임 라이브러리, 인터럽트, 병렬 처리가 추가되었다. 개인용 컴퓨터가 애플, 아타리, IBM, 아미가와 같은 기업 덕택에 1980년대에 유명해졌다. 이 업체들은 한때 메인프레임과 미니컴퓨터에 널리 쓰였던 운영 체제 기능을 추가하였다. 나중에 그래픽 사용자 인터페이스와 같은 수많은 기능들이 개인용 컴퓨터 운영체제를 위해 개별적으로 개발되었다. 1950년대 초에 컴퓨터는 한 번에 하나의 프로그램만 실행할 수 있었다. 각 사용자는 컴퓨터만을 사용하여 예약된 시간에 천공 카드와 테이프의 프로그램과 데이터에 접근하여야 했다. 프로그램이 컴퓨터에 적재되면 컴퓨터는 프로그램이 끝나거나 충돌을 일으킬 때까지 계속 동작하였다. 토글 스위치와 패널 불빛을 이용하여 앞면 패널을 통해 프로그램을 디버깅할 수 있었다. 그 뒤에 나온 컴퓨터는 인간이 알아들을 수 있는 어셈블리어로부터의 기계어 발생이나 입출력과 같은 기능을 도와주기 위하여 사용자 프로그램을 연결해 놓은 소프트웨어 라이브러리와 함께 등장하였다. 이때가 바로 현대 운영체제의 탄생 시기이다. 그러나 여전히 컴퓨터는 한 번에 하나의 일만 할 수 있었다.

2.2 운영체제의 정의

운영체제는 실행 관리자라고 하는 모든 하드웨어와 소프트웨어를 관리하는 컴퓨터시스템이라 볼 수 있다. 운영체제는 어떤 사람이 시스템을 사용하고 어떤 식으로 사용하는지를 관리하므로, 컴퓨터 시스템을 총괄하는 머리라고 볼 수 있다. 이를 쉽게 요약하자면 사용자가 컴퓨터를 쉽게 다룰 수 있도록 해주는 인터페이스이다. 운영체제는 OS라는 단어를 주로 사용한다. OS는 Operating System의 약자로 사용자가 컴퓨터를 보다 쉽게 다룰 수 있게 해주는 인터페이스를 뜻한다. OS에 대한 정의는 전공 서적에 따라 조금씩 차이가 있지만 컴퓨터 자원을 효율적으로 관리하기 위한 시스템, 소프트웨어 플랫폼, 컴퓨터 응용 프로그램 관리자 등 공통점을 지니고 있다. 운영체제는 하드웨어와 소프트웨어를 관리하는 소프트웨어 그 자체라고 할 수 있다. 이러한 운영체제는 어떠한 형태로도 나타날 수 있다. PC용 윈도우만이 운영체제가 아니고, MP3 플레이어를 켜면 전원이 들어와 장치를 깨우고 사용자의 명령에 따라 음악을 재생하는 동작들을 관리하는 것들도 전부 운영체제라 할 수 있다. 단, 이런 식으로 전자기기에 공장 출고 시 설치되며 애플리케이션 설치를 통한 기능 추가를 할 수 없는 것은 보통 펌웨어(firmware)라고 부른다.

2.3 운영체제의 기능

운영체제는 크게 두 가지 기능을 수행하는데, 가상적인 컴퓨터의 제공과 컴퓨터 시스템 자원의 관리다. 이 두 기능은 독립적이기보다는 상호 연관을 갖는 개념이다. 가상적인 컴퓨터의 제공이란 사용자에게 복잡한 하드웨어가 아닌 쉽게 이용할 수 있는 컴퓨터 환경을 제공한다는 뜻이다. 한 가지 예를 살펴보면 문서를 작성해서 보조기억장치에 저장하기 위해서는 우선 이 문서를 저장할 수 있는 보조기억장치의 빈 영역을 찾아 저장한다. 그리고 저장만 하고 끝나는 것이 아니라 이 문서가 어느 트랙의 어느 섹터에 저장되었는지에 대한 정보도 일일이 기억하고 있어야 한다. 그래야 다음에 이 문서를 읽을 수 있기 때문이다. 그러나 다행히도 운영체제가 이런 복잡한 일을 책임지고 있으며, 사용자는 단지 저장 명령만 내리면 된다. 이처럼 운영체제는 사용자가 복잡한 하드웨어를 몰라도 쉽게 이용할 수 있도록 가상적인 컴퓨터 환경을 제공한다.

2.4 운영체제의 구성 요소

[사진 1] 운영체제의 구성

운영 체제는 많은 부분을 이룬다. 가장 중요한 요소 가운데 하나가 커널인데, 커널은 일반인이 일반적으로 보지 못하는 낮은 수준의 프로세스를 제어한다. 메모리를 얼마만큼 읽고 쓸 것인지, 어느 프로세스를 실행할 것인지, 모니터, 키보드, 마우스와 같은 장치를 통해 어떠한 정보를 주고받을 것인지, 네트워크를 통해 받은 정보를 어떻게 해석할 것인지를 제어한다. 사용자 인터페이스는 컴퓨터 사용자가 직접 프로그램을 제어하고 사용할 수 있게 하는 운영 체제의 기능이다. 사용자 인터페이스는 아이콘과 바탕 화면을 지닌 그래픽이나 명령 줄을 지닌 문자를 이룰 수 있다. 이와 비슷한 기능으로 API가 있는데 이것은 응용 프로그램이 다른 프로그램과 상호 작용할 수 있게 하는 서비스와 코드 라이브러리가 한데 모여 있으며 운영 체제 그 자체라고 할 수도 있다. 운영 체제에 따라 이러한 구성 요소들 가운데 다수가 실질적인 부분으로 취급되지 않을 수도 있다. 이를테면 윈도우는 사용자 인터페이스를 운영 체제의 일부로 여기는 데 반해 수많은 버전의 리눅스는 그렇지 않다.

2.5 운영체제의 주요 용어

1) 매크로(macro)와 서브프로그램(sub program): 반복되는 부분을 분리시켜 놓고 이름을 부여하고 이름을 호출하여 실행할 수 있도록 하는 것으로 매크로는 컴파일 이전에 확장 과정을 거쳐 실행 시 빠른 장점이 있으며 서브프로그램은 확장 과정을 거치지 않아 저장 공간을 줄일 수 있다.

2) 로더(loader): 외부기억장치로부터 정보들을 주기억 장치로 옮기기 위하여 메모리 할당 및 연결, 재배치, 적재를 담당하는 서비스 프로그램이다.

3) 링커(linker): 여러 오브젝트(Object)를 한 개의 실행 가능한 형태의 파일로 작성한다. 일반적으로 다른 곳에서 프로그램루틴이나 컴파일, 어셈블 된 루틴들을 모아 실행 가능한 루틴을 작성한다.

4) 오버레이(overlay): 큰 프로그램을 작은 단위의 모듈로 잘라서 필요한 부분만 램

(RAM)에 적재하여 실행할 수 있도록 하는 기법으로 프로그래머가 모듈의 자르는 단위 및 연결 순서를 명시하여야 하므로 사용이 불편하다. 이러한 오버레이의 단점은 가상메모리 기법을 이용해 해결된다.

5) 스와핑(swapping): 여러 개의 프로그램을 하나의 메모리에서 실행될 수 있도록 하기 위해 사용하는 기법으로 실행 도중 스와프 아웃(swap out)이 되어 보조기억 장치로 옮겨졌다가 실행 시 스와프 인(swap in)이 되는 것을 반복하며 실행하는 것을 말한다.

6) 버퍼링(buffering): 주기억 장치의 일부를 큐 방식(FIFO: First In First Out)으로 동작하는 버퍼로 이용하여 하나의 프로그램에서 CPU 연산과 I/O 연산을 중첩시켜 처리할 수 있게 하는 방식으로 CPU의 효율을 높이는 방식이다. 버퍼를 2개 또는 그 이상 사용하는 방식을 이용하여 버퍼링의 효과를 높일 수도 있다.

7) 스풀링(spooling): 주기억 장치를 이용하는 버퍼링(buffering)은 공간이 작으므로 보조기억장치를 이용하여 여러 개의 프로그램에 대하여 입력과 CPU 작업을 중첩시켜 처리할 수 있게 하는 방식으로 보조기억장치 중에서 직접접근 및 저장(DASD: Direct Access Storage Device)이 가능한 장치인 디스크를 사용한다.

8) 프로세스: 실행 중인 프로그램, 주기억장치에 저장된 프로그램, PCB(Process Control Block)와 결합된 형태의 코드를 말하는데, PCB란 프로세스가 작업 도중 필요한 정보나 스케줄에 필요한 여러 가지 정보를 기억하고 있는 구조체이다. 작업 스케줄러(job scheduler)에 의해서 생성되어 주기억장치에 진입한다.

9) 셸(shell): 사용자의 명령어를 번역하여 실행을 지시하고 결과를 사용자에게 돌려주는 역할을 하는 부분으로 커널과 사용자 사이의 인터페이스 역할을 하는 부분이다. 도스(DOS)에서 command.com이 셸의 기능을 하며 유닉스에서는 여러 종류의 셸이 존재한다. 이러한 셸은 주기억장치에 상주하지 않고 사용자가 요구할 때 메모리로 적재되어 실행이 된다.

10) 커널(kernel): 자원을 관리하는 모듈의 집합으로 운영체제 기능의 핵심적인 부분을 모아 놓은 부분이다. 메모리 관리 및 스케줄링 인터럽트 처리 등의 기능을 담당하며 사용자는 직접 커널의 기능을 제어할 수 없으며 단지 셸에 의해 의뢰할 뿐이다. 커널은 항상 필요로 하는 부분이므로 메모리에 적재되어 있다.

11) 스레드(thread): 프로세스는 실행 환경 부분과 제어부분의 두 부분으로 나눌 수 있는데 스레드란 프로세스의 제어부분만을 말한다. 스레드는 실행에 필요한 최소한의 정보만 가지고 자신에 속해 있는 프로세스의 기억장치나 파일과 같은 실행환경을 다른 스레드와 공유하여 프로세스의 생성과 문맥교환 등의 오버헤드를 줄여 운영체제의 성능을 개선할 수 있게 된다.

12) 선점(preemptive) 스케줄링: 특정 프로세스가 중앙처리장치를 효율적으로 사용할 수 없는 시점에 이를 때마다 중앙처리장치의 사용권이 다른 프로세스로 옮겨지는 방식으로 높은 우선순위의 프로세스들이 급하게 실행해야 할 경우에 유용하다.

13) 비선점(nonpreemptive) 스케줄링: 어떤 자원이 어떤 프로세스에 할당이 되면 실행을 종료할 때까지 그 프로세스가 중앙처리장치의 사용권을 독점하여 사용하는 것으로 짧은 작업들을 기다리게 되는 경우가 있지만 모든 프로세스 관리에 공정하다.

14) 병행(concurrent) 프로세스: 시스템 내에 다수의 프로세스들이 동시에 실행되는 것으로 프로세스들이 시스템 내에 동시에 존재하나 어느 한순간에 단지 한 프로세스만 CPU에서 실행된다.

15) 교착상태(deadlock)(=무한대기): 다중 프로그래밍 환경에서 두 개의 프로세스가 서로 다른 프로세스가 가지고 있는 자원을 기다리고 있으며 자신이 차지하고 있는 자원을 내놓지 않는 현상으로 이 두 프로세스에게는 영원히 처리기를 줄 수 없게 된다.

16) 단편화(fragmentation): 연속으로 기억장치를 할당하여 사용할 경우 크기가 맞지 않아서 사용되지 못하는 공간이 생길 수 있는데, 이러한 공간을 단편화 공간이라고 한다.

2.6 운영체제의 연구영역

2000년대의 주요 운영체제로는 범용컴퓨터 운영체제, 중형컴퓨터 운영체제, 소형컴퓨터 운영체제, 모바일 운영체제, 임베디드 운영체제로 구분되고 있다. 범용컴퓨터 운영체제는 VSE(Virtual Storage Extended), OS/390(IBM), MSP/EX (FUJITSU)가 대표적이며, 중형컴퓨터 운영체제는 UNIX, Linux, IRIX, AIX, HP-UX가 대표적이다. 또한 소형컴퓨터 운영체제는 Window XP/Vista/7, Mac OX가 대표적이다. 모바일 운영체제는 Windows CE, Palm OS, Cellvic OS가 대표적이다. 임베디드 운영체제는 Embedded Linux, pSOS, VxWorks, VRTX 등이 대표적으로 새로운 운영체제 개발을 위해 지속적으로 연구되고 있다.

3. 운영체제의 종류

3.1 싱글태스킹/멀티태스킹 운영체제

싱글태스킹 운영체제는 한 번에 오직 하나의 프로그램만 실행할 수 있으나 멀티태스킹 운영체제는 하나 이상의 프로그램이 동시에 실행할 수 있게 한다. 이는 운영체제의 작업 스케줄링 하부 시스템에 의해 제각기 반복적으로 인터럽트 처리되는 여러 프로세스 사이에서 이용 가능한 프로세서 시간을 쪼개는 시분할을 통해 이루어진다, 여기서 시분할이란 주어진 시간을 나누어서 다수의 사용자에게 그 시간을 분배하여 작업을 가능하게 하는 것을 말한다. 멀티태스킹의 경우 선점형과 협동형(비선점형)이 있다. 선점형 멀티태스킹의 경우 운영체제는 CPU 시간을 쪼개어 프로그램들 각각에 슬롯을 할당해준다. 솔라리스, 리눅스, 아미가OS와 같은 유닉스 계열 운영체제들은 선점형 멀티태스킹을 지원한다. 협동형 멀티태스킹은 정해진 방식에 따라 다른 프로세스들에 시간을 제공하기 위해 각 프로세스에 의존함으로써 수행된다. 16비트 버전의 마이크로소프트 윈도우는 협동형 멀티태스킹을 사용하였다. 32비트 버전의 윈도우 NT, 윈도우 9x의 경우 선점형 멀티태스킹을 사용하였다.

3.2 단일/다중 사용자 운영체제

단일 사용자 운영체제는 사용자 구별이 없으나 여러 프로그램이 나란히 실행하는 것은 허용된다. 다중 사용자 운영체제는 디스크 공간과 같은 리소스와 프로세스를 식별하는 기능을 갖춘 멀티태스킹의 기본 개념을 확장하며, 여러 사용자에 속해 있으면서 여러 사용자가 동시에 시스템과 상호 작용할 수 있게 한다, 여기서 다중 사용자란 문자 그대로의 여러 명의 사람을 가리키는 용어가 아니라 여러 명의 컴퓨터 사용자에 의한 동시 접근을 허용하는 운영체제나 소프트웨어를 정의하는 용어이다. 시분할 운영 체제들은 시스템의 효율적인 이용을 위해 태스크를 스케줄링하며, 프로세서 시간, 기억 공간, 인쇄, 기타 자원을 여러 사용자에게 비용적으로 할당하기 위한 회계 소프트웨어를 포함할 수 있다.

3.3 분산 운영 체제

분산 운영 체제는 분산 시스템 연구하는 컴퓨터 과학의 한 분야로, 인터넷에 연결된 여러 컴퓨터들의 처리 능력을 이용하여 메시지를 하나에서 다른 하나로 보냄으로써 거대한 계산 문제를 해결하려는 분산처리 모델이다. 구별된 컴퓨터 그룹을 관리하고 이들이 마치 하나의 컴퓨터인 것처럼 보이게 만들어 준다. 서로 연결되어 통신하는 네트워크화된 컴퓨터들이 개발되면서 분산 컴퓨팅이 활성화되었다. 분산되는 연산들은 하나 이상의 컴퓨터에서 수행되고, 하나의 그룹에 속하는 컴퓨터들이 협업을 할 때 분산 시스템을 형성하게 된다.

3.4 판형 운영체제

운영체제에서, 배포 형식 및 클라우드 컴퓨팅 환경에서 판형은 하나의 가상 머신 이미지를 게스트 운영체제로 만드는 것을 가리키며, 실행 중인 여러 개의 가상 머신을 위한 도구로 이를 저장한다, 여기서 클라우딩 컴퓨팅이란 사용자의 직접적인 활발한 관리 없이 특히, 데이터 스토리지와 컴퓨팅 파워와 같은 컴퓨터 시스템 리소스를 필요시 바로 제공하는 것을 말한다. 이 기법은 가상화와 클라우드 컴퓨팅 관리에 둘 다 사용되며, 대형 서버 웨어하우스 환경에서 흔히 볼 수 있다.

3.5 임베디드 운영체제

임베디드 운영체제는 임베디드 시스템(embedded system, 내장형 시스템)에서 사용할 수 있게 설계되어 있다. PDA처럼 조그마한 기계에 동작하도록 설계되어 있으며, 제한된 수의 자원으로 동작한다. 매우 크기가 작고 극히 효율적으로 설계되어 있다. 임베디드 운영체제의 예로 윈도우 CE와 미닉스 3이 있다.

3.6 실시간 운영체제

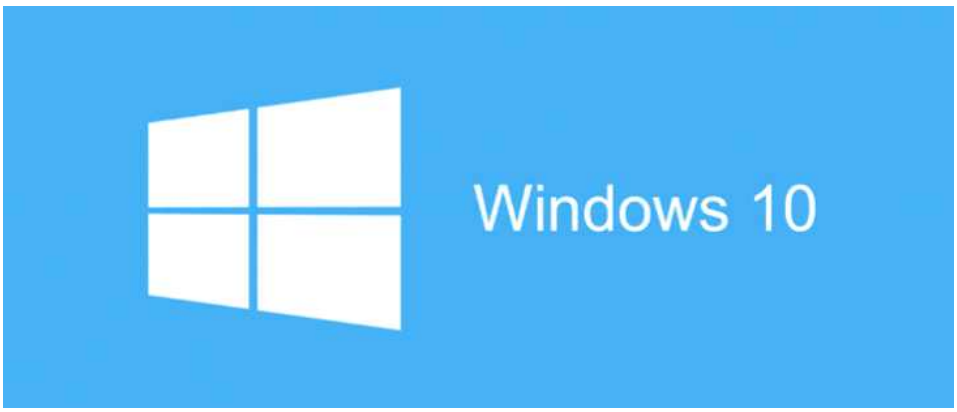
실시간 운영체제는 특정한 짧은 시간 내에 이벤트나 데이터의 처리를 보증하는 실시간 응용 프로그램을 위해 개발된 운영체제이다. 운영체제의 기능 중 CPU 시간 관리 부분에 초점을 맞추어 설계되었다. 실시간 운영 체제는 프로그래머가 프로세스 우선순위에 더 많은 제어를 할 수 있게 한다. 응용 프로그램의 우선순위가 시스템 프로그램의 우선순위를 넘어설 수도 있다. 시스템 코드의 임계 구역을 최소화하였으며, 이를 통하여 응용 프로그램의 처리 요청을 정해진 시간 안에 처리해 줄 수 있다. 실시간 운영 체제는 싱글태스킹일 수도 있고, 멀티태스킹일 수도 있으며 멀티태스킹의 경우 특수한 스케줄링 알고리즘을 사용한다. 두 가지의 기본적인 설계 방식이 존재하는데 이벤트 구동 방식과 시분할 구동 방식이다. 이벤트 구동 방식은 우선순위 기반 스케줄링 또는 선점형 스케줄링이라고도 불리는데 태스크 전환이 현재 태스크보다 높은 우선을 갖는 이벤트가 서비스를 요청할 경우 일어난다. 시분할 구동 방식은 클럭 인터럽트나 라운드 로빈과 같은 주기적인 이벤트가 발생할 때 태스크의 전환이 일어난다. 시분할 스케줄링 방식은 실제 필요한 것보다 더 자주 태스크 전환이 일어난다. 그러나 이러한 점 덕분에 좀 더 자연스럽고, 예측하기 쉬운 멀티태스킹을 제공하며, 하나의 프로세스나 한 명의 사용자가 장치를 독점적으로 사용하는 것과 같은 효과를 제공한다. 때문에 이 방식이 좀 더 나은 멀티태스킹 방식처럼 보일 수 있다.

3.7 라이브러리

라이브러리 운영 체제는 네트워크 등 일반적인 운영 체제가 제공하는 서비스들이 라이브러리 형태로 제공되는 것을 의미한다.

4. PC 운영 체제의 종류

4.1 윈도우



[사진 2] 마이크로소프트 윈도우10 로고

윈도우(Windows)는 마이크로소프트(MicroSoft)에서 개발한 컴퓨터 운영체제로 역사적으로도 가장 대중적으로 널리 알려져 있고 가장 많이 사용되고 있는 운영체제이다.

일정 금액을 지불하여 라이센스를 구입해야 하는 상용 소프트웨어이기 때문에 불법 복제가 가장 많이 이루어지는 운영체제이기도 하다. 95, 98, XP, VISTA, 7, 8, 10, 11등 많은 업그레이드가 이루어졌고 현재 윈도우10 라이센스를 소지하고 있는 사용자들에 한해서 윈도우11 개발자 버전을 무료로 사용이 가능하다. 초창기 윈도우 시리즈는 완전한 OS가 아닌 DOS 커널 위에서 실행되는 셸 형태여서 구조적으로 불안정했다, 그래서 서버용 윈도우 운영체제를 위해 NT커널을 사용한 윈도우 NT를 출시했다. NT커널은 기존 DOS커널보다 우수한 성능과 안정성을 자랑했으나 일반 사용자들은 DOS커널을 계속 사용해왔고 호환성과 같은 다양한 이유로 블루스크린에 시달렸다. 이 때문에 XP부터는 일반 사용자 OS에도 DOS를 통째로 들어낸 NT커널을 사용함으로써 비록 옛날 프로그램을 사용하던 사용자들은 불만을 표했지만 결과적으로 이전보다 뛰어난 성능과 안전성을 인정받고 환영받게 된다.

4.2 macOS

[사진 3] apple macOS 로고

업계 최초 GUI 환경 OS인 Apple Macintosh OS의 후속작으로 유닉스와 Darwin을 기반으로 개발한 Mac 전용 운영체제이다. 하이브리드 커널이며 윈도우와 달리, 소프트웨어가 별도로 판매되지 않고 하드웨어와 함께 완제품으로만 판매하여 소프트웨어 점유율에선 경쟁하지 않는다. macOS의 가장 큰 특징이자 단점은 사용자 레벨 아래 작업을 운영체제나 프로그램이 적절하게 처리한다는 것이다. 이로 인하여 사용자가 쾌적한 환경을 유지하기 위해 시스템 관리에 많은 시간을 투자할 필요성이 없는 점은 장점이지만 시스템 파일 권한이 꼬인다는 식으로 자가 문제를 일으킬 수 있어 우스갯소리로 시한부적 시스템이라고 한다. 또한 레지스트리가 없어 특정 프로그램에 의해서, 혹은 사용자에 의해서 운영체제에 이상 증세가 나타날 확률이 상대적으로 낮다. 이는 유닉스 기반의 운영체제 특성인데 이로 인해 리눅스 배포판을 보면 이를 모방한 테마가 적지 않게 보인다.

이로 인해 MacOS는 결국 유닉스 운영체제가 맞지 않냐는 말도 있는데 Single UNIX Specification Unix 03 인증을 받았으며 최신 버전인 Big Sur 인증 대상에 포함되어있기에 맞다고 할 수 있다. 그러나 SUS(Single UNIX Specification)는 코드 기반이 어떻게 되든 자격 요건만 맞추면 되기 때문에 AT&T System V 코드를 사용하지 않아도 되기에 오해하기가 쉽다. macOS는 BSD의 커널이 사용된 유닉스 기반의 운영체제이며 유닉스의 많은 기능들을 macOS에서도 사용이 가능하지만 다른 유닉스 혹은 리눅스 운영체제와는 많은 차이점을 가지고 있다. macOS가 영향을 받은 BSD의 경우 범용 운영체제로서 다양한 기기에서 사용이 가능하지만 MAC의 경우 애플사의 하드웨어에서만 돌아간다. 유닉스에서 사용가능하다고 알려진 패키지들은 MacOS에서도 쓸 수 있으나 MacOS에서 돌아가는 패키지들은 유닉스용으로 개발되지 않았기에 사용이 불가능하다. 유닉스 및 리눅스에서는 ROOT 유저가 모든 실권을 갖고 그 이상의 권한을 행사할 수 있지만 최근의 MacOS에서는 루트리스 기능을 통해 루트 권한을 얻더라도 시스템 운영에 근본적으로 영향을 주는 권한의 행사를 제한하는 기능이 추가되었다. macOS를 사용하면 윈도우 운영체제를 사용하지 못한다고 알려져 있는데 부트캠프를 사용하여 현재 사용되고 있는 저장장치의 파티션을 나누어 다른 파티션에 윈도우 운영체제를 설치할 수 있다.

4.3 리눅스

[사진 4] Linux 로고

리눅스(Linux)라는 이름은 Linus+ nix, 리누스의 유닉스라는 뜻으로 지어졌다. 리눅스 재단에 따르면 퍼블릭 클라우드 컴퓨팅 워크로드의 90%, 스마트폰의 82%. 임베디드 기기의 62%, 슈퍼컴퓨터 시장의 99%를 차지한다. 데스크탑, 랩탑 용도뿐만 아니라 웹 서버, 클라우드, 안드로이드 및 포터블 게이밍 콘솔 등의 모바일 기기, 가전용이나 상업용 같은 각종 임베디드 기기 등을 구동하는 운영체제이다.

4.4 유닉스

[사진 5] UNIX 로고

유닉스(Unix)는 벨 연구소에서 개발한 운영체제로 윈도우를 제외한 리눅스, 안드로이드, macOS, IOS등의 대부분의 운영체제가 유닉스를 뿌리로 두고 있다. 원래는 멀티유저용 서버 운영체제이나 현재는 개인용 데스크탑이나 임베디드용으로도 많이 쓰인다. 운영체제가 대부분 고급 언어인 C언어로 쓰여 있었고 소스 코드를 쉽게 구할 수 있어 다른 컴퓨터 하드웨어나 새로운 기종에 적은 노력으로도 쉽게 이식이 가능했다.

C언어 자체가 유닉스 시스템 프로그래밍을 하기 위해 만들어졌고 컴팩트하면서도 매우 효율적으로 이식이 쉬운 결과를 얻었다. 각종 편리한 프로그램 도구가 잘 발달해서 프로그래머들이 개발하기에 가장 편리한 환경으로 발전하였다, 그렇기에 C언어 또한 시스템 프로그래밍 언어의 업계 표준이 되었다.

5. 모바일 운영체제의 종류

5.1 안드로이드

[사진 6] 안드로이드 로고

Android는 리눅스 커널을 기반으로 Google에서 제작하고 있는 스마트폰과 같은 플랫폼의 모바일 운영 체제와 미들웨어 및 중요 애플리케이션이 포함된 소프트웨어 집합이다. Google은 새로운 운영 체제의 버전 공개와 동시에 소스를 공개하고 있다. 이렇게 공개된 소스를 AOSP라고 한다. 2019년 Android는 세계에서 가장 대표적인 오픈 소스 플랫폼이며 세계 최다 사용자를 보유한 운영 체제이며 2008년에 1.0 버전이 첫 등장하여 2018년에 10주년을 기록했다. Android의 Linux 커널은 최신 버전을 사용하지 않으며 3~5년 전의 커널을 사용하는 것이 보통이다. 그 이유는 크게 두 가지이다. 먼저 충분히 검증된 안정적인 커널을 이용해 문제 발생의 소지를 최소화하겠단 의도이다. 이러한 경향은 커널 사용이 지연되는 시기의 차이만 있을 뿐 PC 버전용 Linux 배포판 역시 비슷하다. 또한 스마트폰의 성능 동향에 따른 것이기도 하다. 스마트폰의 성능이 아무리 비약적으로 발전한다 한들 PC의 성능에는 크게 미치지 못하기 때문에 굳이 최신 기술과 기능이 포함된 최신 커널에 집착할 필요가 없다. 때문에 커널에 Android의 구조를 맞추기보다는 Android의 구조에 커널 버전을 맞추는 것이다.

5.2 iOS

[사진7] iOS 로고

iOS란 Apple이 생산하는 제품 중 모바일 기기에 탑재되는 독자 운영 체제이다. 최초에는 iPhone을 위해 만들어진 OS였고, 그러다 보니 공개 당시엔 'OS X for iPhone' 그리고 3버전까지는 'iPhone OS'라는 이름을 사용했다. 2010년 4월 이전까지는 따로 공식적으로 통일된 명칭이 없이 iPhone OS라는 이름을 사용하다가, 2010년 4월에 4번째 버전 베타가 공개되어 iPod touch, iPad, Apple TV까지 이 운영 체제를 사용하게 되면서 이름을 iOS로 바꾼다. 이후 Apple TV는 2015년에 tvOS로 분리되었고, iPad는 2019년부터 iPadOS로 분리되어 iPhone과 iPod touch만 iOS를 사용하고 있다. macOS를 이용해 개발되었으므로 Darwin OS를 베이스로 하며, Darwin은 Mach 커널과 BSD 커널로 이루어진 XNU 커널로 만들어졌기 때문에 iOS는 일종의 유닉스 기반이라고 할 수 있다. 따라서 macOS와 커널 구조가 상당히 유사하며 리눅스나 Unix-like OS 등과 시스템 구조가 비슷하다 계층 구조, 터미널 명령어 등도 거의 유사하다.

5.3 iPadOS

[사진 8] apple iPadOS 로고

Apple이 WWDC19에서 공개한 iPad용 운영체제이다. iOS 12까지는 iPhone/iPod touch와 iPad가 제공하는 기능상의 차이일 뿐 동일한 네이밍과 버전을 공유하는 단일 운영체제를 사용하였으나, iPad가 iOS의 버전업 업데이트를 거쳐 오면서 iPad에서만 지원하는 기능이 서서히 많아지면서 2019년 가을 이후로는 iOS 13과 iPadOS의 별개의 운영체제로 분리되었다. iOS를 기반으로 제작되었기 때문에 큰 틀에서는 차이가 없고 기존 iOS에 대화면 특화 기능과 제스처가 추가된 형태를 하는 등, 세세한 부분에서 차이가

있다.

다른 운영체제라곤 하지만 기존과 동일하게 iOS의 앱도 모두 사용할 수 있고 iOS와 버전 업데이트 역시 동일하다. 하지만 iPadOS에서 구동되는 애플리케이션의 제작을 위해 iOS SDK(개발자도구)를 사용해야 하는 등 완벽하게 iOS에서 독립했다고 보기는 어렵다. 만약 별도의 SDK를 만들 경우 iOS에 추가로 SDK를 설치해야 하는 번거로움이 늘어나 iPadOS를 지원하는 앱의 개발 난이도가 상승할 것으로 보이기 때문에 결과적으로 iPadOS의 앱 지원이 기피될 가능성이 없지 않기 때문에 별도 SDK의 가능성은 거의 없기에 개발자 입장에서는 그냥 하나의 OS로 봐도 무방하다.

5.4 webOS

[사진 9] webOS 로고

Palm에서 시작하여 현재 LG전자에서 개발 중인 리눅스 기반 모바일 및 스마트TV, 사물인터넷용 운영체제이다. LG전자의 인수 이전에는 Palm OS의 한계를 벗어나기 위해 만들어졌지만, LG전자 인수 이후에는 스마트TV와 사물인터넷용 운영체제로 변하였고, LG전자는 webOS를 스마트TV와 웨어러블 기기에 응용하고 있다. webOS의 앱을 돌리기 위한 오픈 소스 클론 운영체제로 LuneOS가 있다. webOS와 생긴 것만 비슷할 뿐 코드는 완전히 다르다.

6. 결론

운영체제는 하드웨어 자원을 관리하며 시스템 응용 및 응용 프로그램 실행에 도움을 제공한다. 또한 사용자와 하드웨어 사이에서 중재자 역할 또한 맡아서 수행한다. 운영체제는 크게 두 가지 기능을 수행하는 가상적 컴퓨터의 제공과 컴퓨터 시스템 자원 관리이다. 가상적 컴퓨터의 제공의 경우에는 사용자에게 복잡한 하드웨어가 아닌 쉽게 이용할 수 있는 컴퓨터 환경을 제공하는 기능이고 컴퓨터 시스템 자원 관리는 중앙처리장치, 주기억장치, 보조기억장치, 프로그램, 파일 등 다양한 자원을 관리하는 기능을 맡는다. 또한 프로세스는 생성, 준비, 실행, 종료, 대기 상태가 있다. 운영체제는 이러한 프로세스들을 실행중인 프로그램이라고도 정의할 수 있는데 프로그램 코드뿐만이 아니라 실행에 필요한 다양한 정보 또한 포함된다. 어떠한 프로세스를 실행시킬 것인지를 결정하는 기능도 포함되어 있는데 이러한 과정을 프로세스 스케줄링이라 하며, FCFS(first-come first-served) 스케줄링, 라운드 로빈 스케줄링, 우선순위 스케줄링 이렇게 세 가지 종류로 나뉜다. FCFS(first-come first-served) 스케줄링은 먼저 도착한 프로세스를 먼저 서비스(실행)하는 방법이고, 라운드 로빈 스케줄링은 하나의 중앙처리장치를 임의의 프로세스가 종료될 때까지 차지하는 것이 아니라, 여러 프로세스들이 중앙처리장치를 조금씩 돌아가며 할당받아 실행되는 방식이다. 마지막으로 우선순위 스케줄링은 가장 높은 우선순위의 프로세스에게 먼저 중앙처리장치를 할당하는 방법이다. 운영체제는 주기억장치를 관리하는데, 현재 사용되고 있는 주기억장치 영역과 사용되지 않는 영역에 대한 정보를 유지하면서 주기억장치를 필요로 하는 프로세스에게 할당하고, 프로세스가 종료되면 사용했던 주기억장치 영역을 회수한다.

참고문헌

1) 김종훈, 김종진, 컴퓨터 개론, 운영체제의 기능, 네이버 지식백과, 2013. 03. 10
2) 이계영, 운영체제, 네이버백과, 정익사, 2009년
3) 위키피디아, 운영_체제, 2021.09.18
4) 김태달, 학문명백과 : 백과, 운영체제 Operating System 네이버 지식백과
5) A. Silberschatz, Operating System Concepts 2002년
6) 나무위키, 안드로이드(운영체제), 2021.10.24
7) 나무위키, iOS, 2021.10.24
8) 나무위키, webOS, 2021.08.17