

운영체제는 실행되는 응용 프로그램들이 메모리와 CPU, 입출력 장치 등의 자원들을 사용할 수 있도록 만들어 준다. 더불어, 이들을 추상화하여 파일 시스템 등의 서비스를 제공한다. 또한 멀티태스킹을 지원하는 경우, 여러 개의 응용 프로그램을 실행하고 있는 동안, 운영체제는 이러한 모든 프로세스들을 스케줄링하여 마치 그들이 동시에 수행되는 것처럼 보이는 효과를 낸다.

또한 운영체제는 컴퓨터 과학의 연구 분야이기도 하다.

종류

싱글태스킹 운영체제 / 멀티태스킹 운영체제

싱글 태스킹 운영체제는 한번에 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. 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.

단일 사용자 운영체제 / 다중 사용자 운영체제

단일 사용자 운영체제는 사용자 구별이 없으나 여러 프로그램이 나란히 실행하는 것은 허용한다.^[4] 다중 사용자 운영체제는 디스크 공간과 같은 리소스와 프로세스를 식별하는 기능을 갖춘 멀티태스킹의 기본 개념을 확장하며, 여러 사용자에게 속해 있으면서 여러 사용자가 동시에 시스템과 상호 작용할 수 있게 한다. 시분할 운영체제들은 시스템의 효율적인 이용을 위해 태스크를 스케줄링하며, 프로세서 시간, 기억 공간, 인쇄, 기타 자원을 여러 사용자에게 비용적으로 할당하기 위한 회계 소프트웨어를 포함할 수 있다.

분산 운영체제

분산 운영체제는 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 수행된다. 하나의 그룹에 속하는 컴퓨터들이 협업을 할 때 분산 시스템을 형성하게 된다.^[5]

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. 이 기법은 가상화와 클라우드 컴퓨팅 관리에 둘 다 사용되며, 대형 서버 웨어하우스 환경에서 흔히 볼 수 있다.^[6]

임베디드 운영체제

임베디드 운영체제는 임베디드 컴퓨터 시스템에서 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.

실시간 운영체제

실시간 운영체제는 특정한 짧은 시간 내에 이벤트나 데이터의 처리를 보증하는 운영체제이다. 실시간 운영체제는 싱글태스킹일 수도 있고, 멀티태스킹일 수도 있으며 멀티태스킹의 경우 특수한 스케줄링 알고리즘을 사용한다.

라이브러리

라이브러리 운영체제는 네트워크 등 일반적인 운영체제가 제공하는 서비스들이 라이브러리 형태로 제공되는 것을 의미한다.

운영체제의 구성

운영체제는 많은 부분을 이룬다. 가장 중요한 요소 가운데 하나가 커널인데, 커널은 일반인이 일반적으로 보지 못하는 낮은 수준의 프로세스를 제어한다. 얼마나 메모리를 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 운영체제의 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 운영체제 그 자체라고 할 수도 있다.

운영체제에 따라 이러한 구성 요소들 가운데 다수가 실질적인 부분으로 취급되지 않을 수도 있다. 이를테면 윈도우는 사용자 인터페이스를 운영체제의 일부로 여기는데 반해 수많은 버전의 리눅스는 그렇지 않다.

목적

운영체제의 중요한 목적은 이를테면 다음과 같다.


- 사용자에게 컴퓨터의 프로그램을 쉽고 효율적으로 실행할 수 있는 환경을 제공한다.
- 컴퓨터 시스템 하드웨어 및 소프트웨어 자원을 여러 사용자 간에 효율적 할당, 관리, 보호하는 것
- 운영체제는 제어 프로그램으로서 사용자 프로그램의 오류나 잘못된 자원 사용을 감시하는 것과 입출력 장치 등의 자원에 대한 연산과 제어를 관리한다.

시스템 구성

일반적인 응용 프로그램들을 제외한 컴퓨터 시스템은 개념적으로 세 개의 구성 요소로 구분하기도 한다. 그 세 부분은 운영체제, 셸, 그리고 (낮은 수준의) 필수 유틸리티들이다. 셸은 사용자가 시스템을 운용할 수 있게끔 해주는 기본적인 응용 프로그램이다. 이런 셸의 행동들은 결국 운영체제에 명령을 내리는 일이 된다. `bash` 등의 이러한 셸은 그러나 엄밀히 운영체제의 일부가 아니며 운영체제 입장에서는 셸이나 필수 유틸리티들이나 모두 응용 프로그램일뿐 구별하지 않는다. 다만 이러한 구분은 사용자의 입장에서 필수적인 응용 프로그램이라는 뜻이다.



역사

 이 부분의 본문은 운영체제의 역사입니다.

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.^[7] 하드웨어 기능에 런타임 라이브러리, 인터럽트, 병렬 처리가 추가되었다. 개인용 컴퓨터가 애플, 아타리, IBM, 아미가와 같은 기업 덕택에 1980년대에 유명해졌다. 이 업체들은 한때 메인프레임과 미니 컴퓨터에 널리 쓰였던 운영체제 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. 이것이 현대의 운영체제의 시발점이다. 그러나 여전히 컴퓨터는 한 번에 하나의 일만 할 수 있었다.



나사가 달에 사람을 착륙시키는 데 도움을 준 컴퓨터를 포함하여 OS/360은 1966년부터 대부분의 IBM 메인프레임 컴퓨터에 쓰였다.

메인프레임

 이 부분의 본문은 메인프레임입니다.

1950년대를 통해 일괄 처리, 입출력 인터럽트, 버퍼링, 멀티태스킹, 스펀링, 런타임 라이브러리, 파일 정렬을 위한 프로그램을 포함한 수많은 주요 기능들이 운영체제 분야에 포함되었다. 이러한 기능들은 프로그래머의 취향에 따라 응용 소프트웨어에 포함되어 있기도 했고 포함되지 않기도 했다. 1959년에 IBM 704, 709, 7090 메인프레임 컴퓨터를 위한 세어 운영체제(SHARE)가 통합 유틸리티로 출시되었다.

마이크로컴퓨터

최초의 마이크로컴퓨터는 메인프레임과 미니컴퓨터를 위해 개발해 둔 운영체제에 대한 필요성도 그만큼 용량도 없었다. 모니터스(*Monitors*)라는 이름의 매우 작은 운영체제가 개발되었으며 룸에서 불러들였다. 눈에 띄는 초기의 디스크 기반 운영체제로 CP/M이 있었는데 수많은 초기 마이크로컴퓨터에서 지원되었으며 IBM PC에 널리 쓰였던 MS-DOS(IBM 버전의 것은 IBM DOS, 곧 PC-DOS로 불렸다)와 매우 비슷하였다. 1980년대에 애플 컴퓨터사(지금의 애플사)가 애플 II 시리즈의 마이크로컴퓨터를 버리고, 혁신적인 그래픽 사용자 인터페이스를 맥 OS 운영체제에 갖춘 애플 매킨토시 컴퓨터를 도입하였다.


32비트 아키텍처에 페이징 기능을 갖춘 인텔 80386 CPU 칩이 도입되면서 개인용 컴퓨터가 초기의 미니컴퓨터와 메인프레임 컴퓨터에서 실행할 수 있었던 멀티태스킹 운영체제를 사용할 수 있게 되었다. 마이크로소프트는 DEC용 VMS 운영체제를 개발하였던 데이비드 커틀러를 해고함으로써 이러한 진행에 응하였다. 그는 마이크로소프트의 운영체제 기반을 다룬 윈도우 NT 운영체제 개발을 이끌었다. 애플의 공동 창립자 스티브 잡스는 NeXT 컴퓨터사를 차렸고 유닉스 계열 넥스트스텝 운영체제를 개발하였다. 넥스트스텝은 훗날 애플에 인수되었으며 FreeBSD 코드가 맥 OS X의 중심이 되었다.




PC-DOS는 명령 줄 인터페이스 기능을 제공하였던 초기의 개인용 컴퓨터 운영체제였다.

프로그래머 리처드 스톨만이 사유 유닉스 운영체제를 대체하는 자유 소프트웨어를 목표로 GNU 프로젝트를 시작하였다. 이 프로젝트가 유닉스 여러 곳의 기능을 복제하는 데 큰 성공을 이룩하자 GNU 허드 커널 개발은 비생산적인 것으로 입증되었다. 1991년 핀란드 컴퓨터 과학부 학생인 리누스 토르발스는 최초의 리눅스 커널 버전을 출시하였다. 곧 GNU 유저랜드와 시스템 소프트웨어에 병합되면서 컴퓨터 운영체제를 이루게 되었다. BSD는 유닉스 계열로서 1970년대에 시작하여 수많은 미니컴퓨터에 포팅되었으며 마침내 FreeBSD, NetBSD, OpenBSD와 같이 개인용 컴퓨터에도 쓰이게 되었다.

운영체제의 예

 이 부분의 본문은 [운영체제 목록](#)입니다.

유닉스 및 유닉스 계열 운영체제

 이 부분의 본문은 [유닉스 및 유닉스 계열](#)입니다.

유닉스는 처음에는 어셈블리어로 작성되었다.^[8] 켄 톰프슨은 BCPL에 기반을 둔 B를 작성하였다. 이것으로 말미암아 그는 유닉스를 작성하기도 하였는데, 이는 멀틱스 프로젝트에서의 경험을 바탕으로 한다. C는 B를 대체하였으며 유닉스는 현대의 모든 운영체제에 영향을 미쳤던 크고 복잡한 계열의 상호 관련 운영체제로 발전하였다. (역사 참조) 유닉스 계열 운영체제는 유닉스에서 파생한 계열이며 시스템 V, BSD, GNU/리눅스,와 같은 주된 하부 분류가 몇 가지 있다. 유닉스라는 이름은 어느 운영체제와도 사용할 수 있게 한다는 것을 표명한 오픈 그룹의 상표이다. 유닉스 계열은 원래의 유닉스를 닮은 커다란 집합의 운영체제들을 가리키는 데 흔히 쓰인다.




유닉스 계열의 역사

유닉스 계열 운영체제는 다양한 컴퓨터 아키텍처에서 돌아간다. 이들은 비즈니스 분야의 서버와 학술 및 공학 환경에서의 워크스테이션에서도 쓰인다. GNU/리눅스와 BSD와 같은 자유 유닉스들은 이러한 분야에서 널리 쓰인다.

HP의 HP-UX와 IBM의 AIX는 제조업체의 하드웨어에만 동작하도록 설계되어 있다. 솔라리스와 같은 것들은 x86 서버와 PC를 포함한 여러 종류의 하드웨어에서 돌아간다. 넥스트스텝, 마하, FreeBSD에서 파생한 하이브리드 커널 기반의 BSD류인 애플의 맥 OS X은 유닉스 계열이 아니었던 애플의 초기 맥

OS를 대체하였다. 유닉스의 정보 처리 상호 운용성은 POSIX 표준을 확립함으로써 드러난다. POSIX 표준은 이것이 비록 다양한 유닉스 계열을 위해서 만들어지기는 하였으나 다른 어떠한 운영체제에라도 적용할 수 있다.

BSD 및 BSD 계열

 이 부분의 본문은 BSD입니다.

유닉스 계열의 하부 집합 가운데 하나로 BSD 계열이 있다. 여기에는 FreeBSD, NetBSD, OpenBSD를 포함한다. 이러한 운영체제들은 웹 서버에서 가장 쉽게 찾을 수 있지만 개인용 컴퓨터 운영체제로의 역할도 한다. 인터넷은 BSD와도 많은 관련이 있는데, 네트워크에 연결하여 데이터를 주고 받는 데 흔히 쓰이는 현재의 프로토콜들 가운데 다수가 BSD에서 정의하여 널리 쓰이게 된 것이다. 또, 월드 와이드 웹은 넥스트스텝이라는 BSD 기반 운영체제를 실행하는 수많은 컴퓨터에서 처음 입증되었다.




월드 와이드 웹을 위한 최초의 서버는 BSD 기반의 넥스트스텝에서 실행하였다.

BSD는 유닉스에 뿌리를 두고 있다. 1974년에 캘리포니아 대학교 버클리는 최초의 유닉스 시스템을 설치하였다. 시간이 지나 컴퓨터 과학부의 학생들과 직원들은 그곳에서 문서 편집기와 같은 새로운 프로그램을 이러한 것들에 일찍이 추가하기 시작하였다. 버클리 대학교가 최초의 유닉스 시스템이 설치된 새로운 VAX 컴퓨터를 1978년에 도입하였을 때 대학생들은 컴퓨터 하드웨어의 가능성을 이용하기 위하여 유닉스를 수정하였다. 미국 국방부의 방위고등연구계획국이 이에 관심을 가져 프로젝트에 투자하기로 결정하였다. 수많은 학교와 회사, 정부 단체들은 이를 알아채고 AT&T에서 제공하는 공식적인 것이 아닌 버클리 버전의 유닉스를 사용하기 시작하였다. 1985년에 애플을 떠났던 스티브 잡스는 넥스트스텝이라 불리는 BSD류를 실행하는 고성능 컴퓨터를 제조하였던 기업 NeXT사를 세웠다. 이러한 컴퓨터들 가운데 하나는 팀 버너스 리가 최초의 웹 서버로 월드 와이드 웹을 만드는 데 사용하였다.

OS X

 이 부분의 본문은 macOS입니다.


 맥 OS 문서를 참고하십시오.

맥 OS X은 매킨토시 컴퓨터에 미리 최신으로 설치되어 있으면서도 애플이 개발하여 판매한 사유 그래픽 운영체제이다. 맥 OS X은 1984년 이후로 애플의 사유 운영체제였던 원래의 맥 OS의 뒤를 잇는 것이다. 전작과 달리 맥 OS X은 1980년대 2/4 분기부터 애플이 1987년 초에 이 회사를 사들일 때까지 NeXT에서 개발한 기술로 만든 유닉스 운영체제이다.

이 운영체제는 1999년에 맥 OS X 서버 1.0이라는 이름으로 처음 출시하였으며 그 뒤 2001년 3월에 데스크톱 지향 버전인 맥 OS X v10.0을 출시하였다. 그 뒤로 맥 OS X의 "클라이언트", "서버" 에디션 여섯 가지가 공개되었으며, 최신 제품은 2020년에 출시된 OS X 빅서이다.


서버 에디션인 맥 OS X 서버는 구조적으로 데스크톱의 것과 비슷하지만 일반적으로 애플의 매킨토시 서버 하드웨어에서 돌아간다. 맥 OS X은 메일 전송 에이전트, samba, LDAP 서버, DNS 등을 비롯한 네트워크 서비스에 접근할 수 있게 하는 워크 그룹 관리 및 관리 소프트웨어 도구를 포함하고 있다.

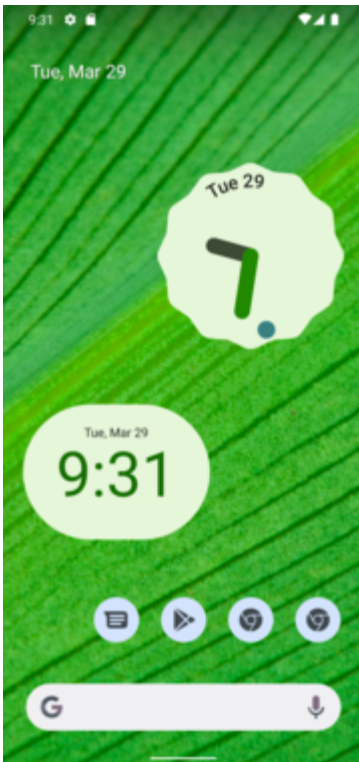
플랜 9

 이 부분의 본문은 플랜 9 (운영체제)입니다.

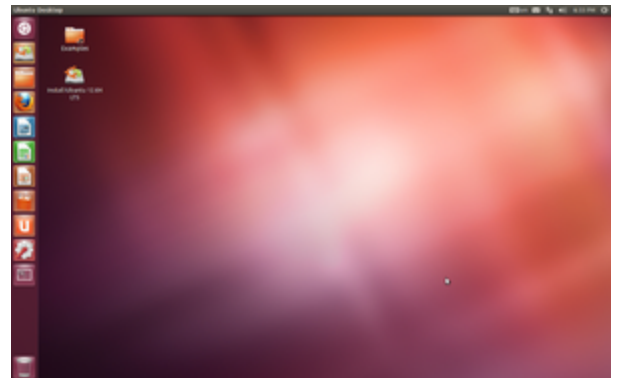
켄 톰프슨, 데니스 리치, 더글라스 맥길로이는 벨 연구소에서 유닉스 운영체제를 개발하기 위하여 C 프로그래밍 언어를 설계하고 개발하였다. 벨 연구소의 프로그래머들은 현대의 배포 환경을 위해 계획된, 플랜 9와 인페르노를 개발하기 시작하였다. 플랜 9는 네트워킹 운영체제로 발돋움하기 위하여 계획된 것이었으며 당시 그래픽이 제공되지 않았던 유닉스와 달리 그래픽을 내장하였다. Lucent 공중 허가 사용 허가서 하에 출시되었다. 인페르노는 Vita Nuova Holdings에 팔려 GPL/MIT 라이선스로 배포되고 있다.

리눅스와 GNU

 이 부분의 본문은 GNU, 리눅스 및 리눅스 커널입니다.



안드로이드는 리눅스 커널을 이용하는 대중적인 모바일 운영체제이다.




리눅스의 데스크톱 배포판인 우분투.

리눅스는 BSD 및 그 변종과 달리 실제 유닉스 코드 없이 개발된 유닉스 계열 운영체제이다. 슈퍼컴퓨터에서부터 손목시계에 이르기까지 다양한 기기에 쓰인다. 리눅스 커널은 오픈 소스 라이선스로 배포되므로 누구나 코드를 읽고 수정할 수 있다. 리눅스는 다양한 전자 기기에서 동작하도록 수정되고 있다.

GNU 프로젝트는 완전한 원래의 코드를 제외하고 유닉스와 비슷하게 완전하게 자유롭고 열려 있는 운영체제를 만들고 싶어하는 프로그래머들의 협동적인 노고 그 자체이다. 이 프로젝트는 1983년에 리처드 스톨만이 시작하였고 대부분의 리눅스의 수많은 부분을 책임지고 있다. 이 까닭에 리눅스는 GNU/리눅스로 불리기도 한다. 실질적으로 모든 운영체제를 위한 수많은 소프트웨어가 GNU 일반 공중 사용 허가서 하에 배포된다. 이 가운데 리눅스 커널은 핀란드의 대학교 학생이었던 리누스


토르발스의 부차적인 프로젝트로 시작되었다. 1991년에 토르발스는 이 작업에 착수하여 프로젝트에 대한 정보를 컴퓨터 학생과 프로그래머를 위한 뉴스그룹에 게시하였다.

구글 크롬 OS

 이 부분의 본문은 구글 크롬 OS입니다.

크롬은 구글이 리눅스 커널을 기반으로 설계한 운영체제이다. 크롬은 대부분의 시간을 인터넷으로 보내는 이용자들을 대상으로 한다. 기술적으로는 어떠한 응용 프로그램도 없는 웹 브라우저만을 이용하며 문서 작성이나 미디어 보기와 같은 작업을 위해 웹 브라우저에 쓰이는 인터넷 애플리케이션에 의존한다.

마이크로소프트 윈도우

 이 부분의 본문은 마이크로소프트 윈도우입니다.

마이크로소프트 윈도우는 개인용 컴퓨터에 가장 흔히 쓰이는 사유 운영체제 계열이다. 개인용 컴퓨터를 위한 가장 흔한 운영체제이며 약 90%의 시장 점유율을 차지하고 있다.^{[9][10][11]} 최신 버전은 개인용 컴퓨터의 경우 윈도우 11이, 서버의 경우 윈도우 서버 2019이다.

1981년에 IBM PC용의 오래된 MS-DOS 운영체제에 추가 기능으로 나온 것이 기원이다. 1985년에 마이크로소프트는 개인용 컴퓨터의 비즈니스 분야를 지배하기 시작하여 수많은 산업 표준을 정립하기에 이르렀다. 윈도우 XP를 시작으로 현대의 모든 윈도 버전은 윈도우 NT 커널을 기반으로 하고 있다. 현재 나오는 윈도우 버전은 IA-32와 x86-64 프로세서에서 동작하지만 그 이전에 나왔던 버전들은 다른 아키텍처를 지원하기도 하였다.


기타

틈새 시장에 존재하는 오래된 운영체제로는 IBM과 마이크로소프트의 OS/2, 애플 맥 OS X의 유닉스를 이용하지 않는 이전작 맥 OS, 또 BeOS와 XTS-300이 있다. RISC OS, MorphOS, 아미가OS 4도 열성적인 커뮤니티와 전문 분야를 위한 소수 플랫폼으로 개발이 이어지고 있다. DEC의 OpenVMS도 휴렛 패커드가 계속 개발하고 있다. 도스 등의 운영체제도 FreeDOS와 같은 프로젝트를 통해 명맥을 이어가고 있다. 그 밖의 운영체제는 운영체제 교육용이나 학술용, 또는 운영체제 개념의 연구 등을 위해 예외적으로 쓰인다. 학술과 연구 역할을 모두 수행하는 시스템의 전형적인 예로 미닉스가 있다. 반면 순수 연구 목적으로 쓰이는 것으로는 싱글래터티가 있다.

구성 요소

운영체제를 이루는 요소는 컴퓨터의 다른 부분들과 함께 동작하게 만들기 위하여 존재한다. 금융 데이터베이스부터 영화 편집 프로그램에 이르기까지 소프트웨어는 모두 소프트웨어에 쓰이는 하드웨어가 단순히 마우스나 키보드만 이용하든지 아니면 인터넷 연결같이 복잡한 방식을 이용하든지 상관 없이 하드웨어를 이용하기 위하여 운영체제로 말미암아 실행해야 한다.

커널

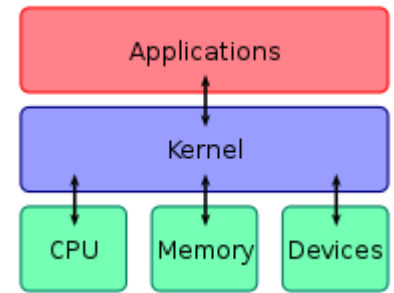
 이 부분의 본문은 커널 (컴퓨팅)입니다.

펌웨어와 장치 드라이버의 도움을 받아 커널은 모든 컴퓨터 하드웨어 장치에 대한 가장 기초 수준의 제어권을 제공한다. 커널은 램을 통해 프로그램을 위한 메모리 접근을 관리하며 어느 프로그램이 어느 하드웨어 자원에 접근할지를 결정하며 CPU의 동작 상태를 늘 최적으로 설정 및 초기화하고 디스크, 테이프, 플래시 메모리와 같은 매체의 파일 시스템을 갖춘 장시간 비휘발성 기억 장치를 위한 데이터를 정리한다. 운영체제 내에서의 커널의 영역과 그 구성에 따라 모노리딕 커널(monolithic kernel), 마이크로 커널(micro kernel) 등으로 분류한다.

프로그램 실행

 이 부분의 본문은 프로세스입니다.

운영체제는 응용 프로그램과 하드웨어 사이의 인터페이스 역할을 한다. 운영체제는 응용 프로그램 개발을 단순하게 하는 서비스의 집합이다. 프로그램을 실행하면 운영체제가 프로세스를 만든다. 커널은 메모리와 다른 자원을 할당하여 프로세스를 만들며, 이로써 멀티태스킹 환경에서 프로세스에 대한 우선 순위를 확립하고, 메모리에 프로그램 코드를 적재하며 프로그램을 실행한다. 그 뒤 프로그램은 사용자 및 장치와 상호작용한 다음 원하는 명령을 수행하게 된다.



커널이 응용 소프트웨어를 컴퓨터 하드웨어에 연결하고 있다.

운영체제는 프로세스^[12] 들을 생성하거나 삭제하고, 중단시키거나 재개시킨다. 프로세스 간의 동기화와 통신, 교착상태 처리에 관한 메커니즘을 제공한다.

1. 사용자 및 시스템 프로세스의 생성과 종료 관리
2. 프로세스의 일시 중지와 속개
3. 프로세스 동기화를 위한 수단의 제공
4. 프로세스간 통신을 위한 수단의 제공
5. 교착 상태 처리를 위한 수단의 제공

인터럽트

이 부분의 본문은 인터럽트입니다.

인터럽트는 주변 환경에 반응하고 상호작용하는 데에 효율적인 방법을 운영체제에 제공하므로 운영체제에 핵심적인 역할을 한다고 할 수 있다. 동작을 요구하는 이벤트(폴링)를 위한 다양한 소스의 입력을 운영체제가 감시할 수 있는 다른 대안은 스택이 매우 작은 구형 운영체제에서 볼 수 있으나 스택이 큰 현대의 운영체제에서는 드문 편이다. 인터럽트 기반의 프로그래밍은 현대의 대부분의 CPU에서 직접적으로 지원된다. 인터럽트는 자동으로 로컬 레지스터 컨텍스트를 저장하고 이벤트에 반응하는 특정 코드를 실행하는 방법을 컴퓨터에 제공한다. 매우 기초적인 컴퓨터들은 모두 하드웨어 인터럽트들을 지원하며 이벤트가 발생할 때 실행될 코드를 프로그래머가 지정할 수 있게 한다.

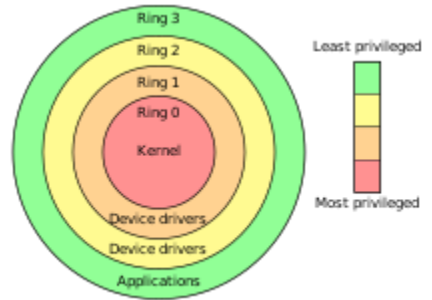
프로그램이 운영체제에 인터럽트를 발생시키는 경우도 있다. 이를테면 프로그램이 하드웨어에 접근하고자 한다면 운영체제의 커널을 가로막을 수 있으며, 이를 통해 제어권을 커널에 넘겨준다. 그 뒤 커널은 요청을 수행한다. 프로그램이 메모리(또는 공유 자원)와 같은 추가 자원이 필요하면 커널이 집중할 수 있게 인터럽트를 발생시킨다.

모드

이 부분의 본문은 보호 모드 및 수퍼바이저 모드입니다.

현대의 CPU는 여러 모드의 명령을 지원한다. 이러한 기능을 지원하는 CPU에는 두 가지 모드가 있다: 보호 모드, 수퍼바이저 모드. 수퍼바이저 모드는 메모리의 기록 및 삭제 방식을 제어하는 것과 그래픽 카드와 같은 장치와 통신하는 것과 같이 운영체제의 커널이 하드웨어에 제한 없이 액세스해야 하는 낮은 수준의 태스크를 위해 사용한다. 반대로 보호 모드는 그 밖의 거의 모든 용도로 사용된다. 응용 프로그램들은 보호 모드 안에서 동작하며, 수퍼바이저 모드의 모든 것을 제어하는 커널과 통신해야만 하드웨어를 이용할 수 있다. CPU는 오래된 프로세서를 가상으로 구현하기 위한 가상 모드와 같이 보호 모드와 비슷한 다른 모드들을 지니고 있을 수도 있다. (이를테면 32비트에서 16비트 프로세서를, 아니면 64비트에서 32비트 프로세서를 가상으로 구현할 때)

컴퓨터가 처음 시동할 때 자동으로 수퍼바이저 모드에서 실행된다. 컴퓨터를 켜자마자 먼저 실행되는 몇 안 되는 프로그램들이 바이오스와 부트로더이며 운영체제는 하드웨어에 제한 없이 접근한다. 그리고 운영체제가 다른 프로그램에 대한 제어권을 보낼 때 CPU를 보호 모드에 놓을 수 있다.



메모리 관리

이 부분의 본문은 메모리 관리입니다.

멀티프로그래밍 운영체제 커널은 현재 프로그램이 이용하는 모든 시스템 메모리를 관리해야 한다. 이로써 어느 특정한 프로그램이 다른 프로그램이 이미 사용하고 있는 메모리와 상호 작용하지 않게 한다. 프로그램이 시분할하므로 각 프로그램은 메모리에 독립적으로 접근해야 한다.

보호 모드에서 사용할 수 있는 x86용 권한 링. 운영체제는 어느 프로세스가 개별 모드에서 실행할 것인지 결정한다.

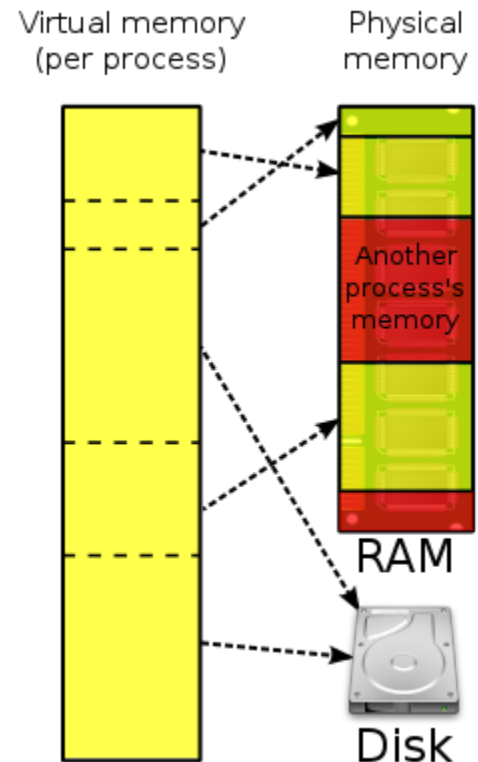
가상 메모리

이 부분의 본문은 가상 메모리입니다.

페이징이나 세그먼테이션과 같은 가상 메모리 어드레싱을 이용하면 커널은 어느 메모리를 각 프로그램이 주어진 시간에 사용할 수 있게 할지 설정할 수 있다. 그러므로 운영체제가 여러 개의 태스크에 같은 메모리 위치를 사용할 수 있게 한다.

프로그램이 접근할 수 있는 메모리 범위에 없는 메모리에 접근하려고 하지만 그곳에 할당되면 커널은 프로그램이 마치 할당된 메모리를 초과 사용한 것과 같은 방식으로 인터럽트 처리한다. 유닉스에서 이러한 종류의 인터럽트를 페이지 실패라고 부른다. 커널이 페이지 실패를 감지하면 이러한 문제를 일으킨 프로그램의 가상 메모리 영역을 수정하는 것이 일반적이다. 이로써 요청된 메모리에 프로그램이 접근할 수 있게 한다.

현대의 운영체제에서 자주 접근하지 않는 메모리는 일시적으로 디스크나 다른 매체에 저장하여 다른 프로그램에게 사용할 수 있는 공간을 제공해 준다. 이를 스왑 처리(swapping)라고 하며 이를 통해 여러 개의 프로그램이 특정한 메모리 영역을 차지할 수 있다.



수많은 운영체제는 하드 디스크와 램에 분산된 메모리를 이용하여 마치 가상 메모리로 불리는 메모리 덩어리가 연속적인 것처럼 프로그램을 속일 수 있다.


멀티태스킹

이 부분의 본문은 멀티태스킹 및 프로세스 관리입니다.

멀티태스킹은 여러 개의 독립적인 컴퓨터 프로그램을 하나의 컴퓨터에 실행시키는 것을 가리킨다. 마치 태스크들이 동시에 수행하는 것처럼 보여 준다. 대부분의 컴퓨터가 한 번에 최대 한 두개를 수행할 수 있고 이는 일반적으로 시분할을 통해 수행된다. 다시 말해 각 프로그램은 컴퓨터의 실행 시간의 일부를 사용한다.

운영체제 커널은 스케줄러라는 프로그램이 포함되어 있는데 이 프로그램은 얼마나 많은 시간을 각 프로그램이 실행에 소비하게 할 것인지를 결정하며 여기서 실행 제어권이 프로그램에 넘어갈 수 있게 한다. 제어권은 프로그램이 CPU와 메모리에 접근할 수 있게 하는 커널로 말미암아 프로세스로 넘어간다. 나중에 다른 프로그램이 CPU를 사용할 수 있게 하기 위해 제어권은 같은 메커니즘을 통하여 커널로 반환된다. 커널과 응용 프로그램 간의 제어권 이동을 이른바 문맥 교환이라고 부른다.

디스크 접근 및 파일 시스템

 이 부분의 본문은 가상 파일 시스템입니다.

디스크에 저장된 데이터로 접근하는 것은 모든 운영체제의 기본 기능이다. 컴퓨터는 더 빠른 접근, 더 높은 신뢰성을 위해, 또 드라이브의 남은 공간을 더 잘 이용하기 위한 특정한 방식으로 구조화된 파일을 이용하여 디스크에 데이터를 저장한다. 파일을 디스크에 저장하는 이러한 방식을 파일 시스템이라고 부르며 파일이 이름과 특성을 가질 수 있게 한다. 또, 이러한 파일들을 디렉터리 트리로 정렬되는 특정 계급의 디렉터리와 폴더에 저장하게 한다.




파일 시스템은 디렉터리(폴더)를 이용하면서 사용자와 프로그램이 컴퓨터 파일을 정리하고 정렬할 수 있게 해 준다.

초기의 운영체제는 일반적으로 한 종류의 디스크 드라이브와 한 종류의 파일 시스템을 지원하였다. 초기의 파일 시스템들은 용량, 속도, 또 파일 이름과 디렉터리 구조의 종류에 제한이 있었다. 이러한 제한은 설계된 운영체제의 제한에 반영되므로 특정한 운영체제가 하나 이상의 파일 시스템을 지원하는 것을 매우 어렵게 만들었다.

더 단순한 수많은 운영체제들은 기억 장치의 시스템에 접근하기 위한 제한된 옵션들을 지원하였는데, 유닉스와 GNU/리눅스와 같은 운영체제들은 가상 파일 시스템(VFS)이라는 기술을 지원한다. 유닉스와 같은 운영체제는 공통 API를 통해 접근하는 파일 시스템이나 디자인에 관계 없이 다양한 기억 장치를 지원한다. 그러므로 프로그램을 개발할 때 장치 접근에 대한 정보를 공부하지 않아도 되게 한다. VFS는 다양한 파일 시스템에, 특정한 장치 드라이버와 파일 시스템 드라이버를 사용하여 프로그램들이 무제한의 장치에 접근할 수 있는 기능을 운영체제에 제공한다.

연결된 하드 드라이브와 같은 기억 장치들은 장치 드라이버를 통해 접근한다. 장치 드라이버는 드라이브의 특정한 언어를 이해하고 이 언어를 운영체제가 모든 디스크 드라이브에 접근할 때 사용하는 표준 언어로 번역할 수 있다. 유닉스에서 이를 블록 장치의 언어라고 한다.

장치 드라이버


 이 부분의 본문은 장치 드라이버입니다.

장치 드라이버는 하드웨어 장치들과 상호 작용할 수 있도록 개발된 특정한 종류의 컴퓨터 소프트웨어이다.

장치 드라이버의 주된 설계 목적은 추상화이다. 하드웨어의 모델은, 특히 같은 종류의 장치라 하더라도 각기 다르다. 제조업체들은 더 신뢰할만한, 더 나은 성능을 제공하기 위해 더 새로운 모델들을 출시하고 이러한 새로운 모델들은 다르게 동작하기도 한다. 컴퓨터들과 운영체제들은 현재든 앞으로든 모든 장치를 어떻게 제어할 것인지 예측하는 것은 불가능하다. 이러한 문제를 해결하기 위해 운영체제들은 반드시 어떠한 종류의 장치가 제어될 것인지를 지시하여야 한다. 그러면 장치 드라이버의 기능이 이러한 운영체제의 함수 호출을 장치 특유의 호출로 번역하게 된다. 이론적으로 새로운 방식으로 제어되는 새로운 장치는 적절한 드라이버를 사용할 수 있는 상황이라면 올바르게 동작한다. 새로운 드라이버는 운영체제의 관점에서 장치가 평소처럼 동작하고 있음을 보증하게 된다.

비스타 이전의 윈도우, 2.6 미만의 리눅스 버전에서는 모든 드라이버 실행이 협동적이어서, 드라이버가 무한 루프에 진입하면 시스템이 정지하는 일이 발생한다. 더 최근에 나온 운영체제들은 커널 선점을 사용한다. 이 경우 태스크 제공을 위해 커널이 드라이버를 간섭하며 장치 드라이버로부터 응답을 받을 때까지 자신을 프로세스로부터 분리시키며 더 많은 태스크를 부여할 수 있다.

네트워킹

 이 부분의 본문은 컴퓨터 네트워크입니다.

현재 대부분의 운영체제는 다양한 통신 프로토콜, 하드웨어, 응용 프로그램을 지원한다. 다시 말해, 서로 비슷하지 않은 운영체제를 실행하는 컴퓨터가 자원(유무선 연결을 이용한 연산, 파일, 프린터, 스캐너)을 공유하기 위해 같은 망에 참여할 수 있다. 네트워크는 컴퓨터의 운영체제가 원격 컴퓨터의 자원에 접근하는 데 필수적이다. 마치 리소스가 로컬 컴퓨터에 바로 연결되어 있는 것처럼 보이게 만들어 준다. 여기에는 컴퓨터의 그래픽, 사운드 하드웨어를 공유하거나 네트워크 파일 시스템을 이용하는 등 단순한 통신에서 나오는 모든 것을 포함한다. 일부 네트워크 서비스는 컴퓨터의 자원을 투명하게 접근할 수 있게 한다. 이를테면 SSH는 네트워크로 이어진 사용자들이 컴퓨터의 명령 줄 인터페이스에 직접 접근할 수 있게 한다.

보안

 이 부분의 본문은 컴퓨터 보안입니다.

컴퓨터의 보안은 수많은 기술이 올바르게 동작하고 있는 지에 달려 있다. 또, 운영체제는 보안을 위하여 특정 환경에 대한 권한을 사용자나 프로그램에 개별적으로 설정하고 인증 프로세스를 제공한다. 인터넷 보안은 특히 여러 사용자가 사용하는 시스템에 적절하다. 시스템을 사용하는 각 사용자는 개인 파일을 다른 사용자가 읽을 수 없게 할 수 있다.

사용자 인터페이스

 이 부분의 본문은 사용자 인터페이스입니다.

어떠한 종류의 입력을 받는 모든 컴퓨터는 사람이 컴퓨터와 소통할 수 있게 하는 사용자 인터페이스가 필요하다. 키보드, 마우스와 같은 장치들이 이러한 역할을 하지만 사용자 인터페이스는 이를 위한 소프트웨어로 이루어진다. 사용자 인터페이스는 역사적으로 컴퓨터 명령어를 한 줄씩 입력해 나가는 명령 줄 인터페이스와 일반적으로 창, 단추, 아이콘을 이루는 시각 환경이 존재하는 그래픽 사용자 인터페이스로 나뉜다.

그래픽 사용자 인터페이스

 이 부분의 본문은 그래픽 사용자 인터페이스입니다.

현대의 대부분의 컴퓨터 운영체제는 그래픽 사용자 인터페이스(GUI)를 지원한다. 마이크로소프트 윈도우와 맥 OS와 같은 일부 컴퓨터 시스템에서 GUI는 커널에 통합되어 있다.

기술적으로 그래픽 사용자 인터페이스는 운영체제의 서비스가 아니지만 운영체제 커널에 통합하면 GUI가 출력 명령을 수행하는 데 필요한 수많은 **문맥 교환**을 없앴으로써 GUI를 더 반응적으로 만들 수 있다. 다른 운영체제로는 그래픽 하부 시스템을 커널과 운영체제로부터 분리시키는 **모듈성**이 있다. 1980년대 유닉스에서 VMS 등은 이러한 방식으로 만들어진 운영체제였다. GNU/리눅스 및 맥 OS X 또

```
# netstat -tln | grep sshd
```

```
PING google.com [74.125.95.103] 64(84) bytes of data:
```

```
64 bytes from london-1209.lga20n.net [74.125.95.103]: icmp_seq=1 ttl=67 time=5.2 ms
```

```
rtt min/avg/max/mdev = 4.812/4.851/4.933/0.000 ms
```

```
^C
```

```
mmap ^ # / s
```

```
Desktop README
```

```
root ~ # cd /
```

```
root ~ # / s
```

```
bin root home testfound apt proc wlan srs var
```

```
boost ext lib media opt root
```

```
extra/libsource 2.6.6-1
```

```
In library extracted from Pkgin
```

```
extrapkgin 2.6.6-1
```

```
- Multi-protocol instant messaging client
```

```
extrapkgin+encryption 2.6.3-
```

```
+ A Pkgin plugin providing transparent RSA encryption using NSS
```

```
extrapurple-pkgin-gsm 2.6.3-1
```

```
Plugin path for Pkgin
```

```
extractedqemu+nss 2.6.3-1 (telephony)
```

```
A telephony-backend to use Libpurple IPkgin protocols.
```

```
community/pflocations 2.16-1
```

```
A set of GUI popup notifications for pkgin
```

```
community/pkgin-fusemounter 0.1.6-1
```

```
Adds a fileshare button to the conversation window
```

```
community/pkgin-libeventy 0.16-3
```

```
pkgin plugin that enables packages when someone logs in or messages you
```

```
community/pkgin-musictracker 0.4.21-2
```

```
Provides access to Pkgin which displays the music track currently playing
```

```
community/pkgin-net 2.2.6-1
```

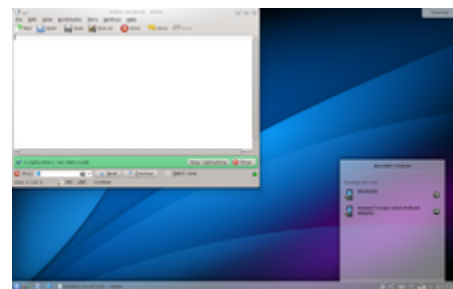
```
Off-the-record Messaging Plugin for Pkgin
```

```
root ~ # cat /etc/passwd | sed -e 's/:$//g' > /tmp/.passwd
```

명령 줄의 한 예. 각 명령어는 프롬프트 뒤에서 입력하면 되며 이에 대한 출력은 아래에 나타난다. 현재의 명령 프롬프트가 아래에 있다.


한 이러한 방식을 취한다. 윈도우 비스타와 같은 현대의 마이크로소프트 윈도우는 거의 사용자 공간에 위치한 그래픽 하부 시스템을 포함하고 있지만 윈도우 NT 4.0과 윈도우 서버 2003 버전 사이의 그래픽 구현 루틴은 거의 커널 공간에 존재한다. 윈도우 9x는 인터페이스와 커널 사이의 구별이 거의 없다.

수많은 컴퓨터 운영체제는 사용자가 원하는 인터페이스를 만들어 설치할 수 있게 하고 있다. X 윈도 시스템을 GNOME이나 KDE와 함께 쓰면 유닉스 및 유닉스 계열 시스템에서 이러한 설정을 할 수 있다. 수많은 윈도우 셸 치환을 통해 마이크로소프트 윈도우에서도 이러한 작업을 수행할 수 있는데, 윈도우 셸을 교체하는 방식을 쓰지만 윈도우로부터 셸 자체를 분리할 수는 없다.



그래픽 사용자 인터페이스의 한 예. 프로그램들은 화면 위에 그림의 모습을 띠고 있으며 파일, 폴더, 응용 프로그램들은 아이콘과 기호의 모습을 띤다. 마우스는 컴퓨터를 탐색하는 데 쓰인다.

실시간 운영체제

 이 부분의 본문은 실시간 운영체제입니다.

실시간 운영체제 (RTOS)는 정해진 기간 안에 수행이 끝나야 하는 응용 프로그램을 위하여 만들어진 멀티태스킹 운영체제이다. (실시간 연산) 이러한 응용 프로그램들에는 조그마한 임베디드 시스템, 자동차 엔진 제어 장치, 산업 로봇, 우주선, 산업 제어 장치, 일부 대형 컴퓨터 시스템 등이 있다.

초기의 대형 실시간 운영체제는 이를테면 아메리칸 항공과 IBM이 사브레 항공 예약 시스템을 위하여 개발한 트랜잭션 프로세싱 퍼실리티(TPF)가 있다.


일부 임베디드 시스템은 실시간 연산을 지원하지 않더라도 심비안 OS, 팜 OS, BSD, GNU/리눅스와 같은 운영체제를 이용한다.

취미 활동을 통한 운영체제 개발

운영체제 개발은 컴퓨팅에 취미를 둔 사람들이 관여하는 가장 복잡한 활동들 가운데 하나이다. 취미로 만드는 운영체제는 기존의 운영체제로부터 직접적으로 코드를 가져오지 않은 것으로 분류될 수 있으므로, 사용자와 활동 개발자들의 수는 적은 편이다.^[13]

취미 활동으로 개발된 운영체제의 예로는 ReactOS와 Syllable 등이 있다.

시장 점유율

 이 부분의 본문은 운영 체제 시장 점유율입니다.

2013년 운영체제별 전 세계 기기 선적량^[14]

운영체제	2012년 (단위: 100만 기기)	2013년 (단위: 100만 기기)
안드로이드	504	878
윈도우	346	328
iOS/맥 OS	214	267
블랙베리	35	24
기타	1,117	803
전체	2,216	2,300

출처: 가트너

리눅스 재단에 따르면 퍼블릭 클라우드 워크로드의 90%, 세계 스마트폰의 82%, 임베디드 기기의 62%, 슈퍼 컴퓨터 시장의 99%가 리눅스로 작동한다.^[15]

범위 및 논란

운영체제는 잘 정의된 인터페이스를 가지므로 운영체제와 응용 프로그램 간의 구분은 명확하지만, 어느 정도의 서비스를 운영체제 안에 포함시켜야 하는가에 대한 문제는 기술적인 문제일뿐만 아니라 사업적인 문제이기도 하다. 이 문제는 다음의 경우에 분명히 드러난다. 1998년 미국 법무부는 마이크로소프트에 대해 소송을 제기하였는데, 요점은 마이크로소프트가 운영체제에 너무 많은 기능을 포함시켜 응용 프로그램 제작업체들에게 피해를 주었다는 것이다.

같이 보기

- 운영체제의 역사
- 운영체제 목록
- 운영 체제 최적화
- 모바일 운영 체제
- 범용 운영 체제
- 전용 운영 체제
- 실시간 운영 체제 (RTOS)
- 임베디드 시스템
- 객체 지향 운영 체제

각주

1. Stallings, p. 6.
2. Dhotre, p. 1
3. Operating system market share (<http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=10>)

4. Lorch, Jacob R., and Alan Jay Smith. "Reducing processor power consumption by improving processor time management in a single-user operating system." Proceedings of the 2nd annual international conference on Mobile computing and networking. ACM, 1996.
5. Mishra, B.; Singh, N.; Singh, R. (2014). 〈Master-slave group based model for co-ordinator selection, an improvement of bully algorithm〉 . 《International Conference on Parallel, Distributed and Grid Computing (PDGC)》 . 457–460쪽. doi:10.1109/PDGC.2014.7030789 (<https://dx.doi.org/10.1109%2FPDGC.2014.7030789>). ISBN 978-1-4799-7682-9.
6. Gagne, Silberschatz Galvin (2012). 《Operating Systems Concepts》 . New York: Wiley. 716 쪽. ISBN 978-1118063330.
7. Hansen, pp. 4-7
8. Ritchie, Dennis. "Unix Manual, first edition" (<https://web.archive.org/web/20080518013206/http://cm.bell-labs.com/cm/cs/who/dmr/1stEdman.html>). Lucent Technologies. 2008년 5월 18일에 원본 문서 (<http://cm.bell-labs.com/cm/cs/who/dmr/1stEdman.html>)에서 보존된 문서. 2012년 11월 22일에 확인함.
9. W3Counter - Global Web Stats (<http://www.w3counter.com/globalstats.php>)
10. Operating system market share (<http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8>)
11. <http://gs.statcounter.com/#os-ww-monthly-200910-200910-bar>
12. 어떠한 프로그램 파일들이 준비 상태나 대기 상태로 실행되기 전에 적재되는 것
13. "My OS is less hobby than yours" (http://www.osnews.com/story/22638/My_OS_Is_Less_Hobby_than_Yours). 《Osnews》 . 2009년 12월 21일. 2009년 12월 21일에 확인함.
14. Lance Whitney (2014년 1월 7일). "Android device shipments to top 1 billion this year -- Gartner" (http://news.cnet.com/8301-1035_3-57616768-94/android-device-shipments-to-top-1-billion-this-year-gartner/?part=rss&subj=cnet&tag=제목&utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+cnet%2FNnTv+%28CNET+River+RSS%29).
15. 컴퓨팅세계 평정한 '리눅스'...커뮤니티도 살아있네 2017.10.30
http://m.zdnet.co.kr/news_view.asp?article_id=20171030170645

참고 문헌

- Auslander, Marc A.; Larkin, David C.; Scherr, Allan L. (1981). "The evolution of the MVS Operating System" (<http://www.research.ibm.com/journal/rd/255/auslander.pdf>) (PDF). IBM J. Research & Development.
- Deitel, Harvey M.; Deitel, Paul; Choffnes, David. 《Operating Systems》 . Pearson/Prentice Hall. ISBN 978-0-13-092641-8.
- Bic, Lubomur F.; Shaw, Alan C. (2003). 《Operating Systems》 . Pearson: Prentice Hall.
- Silberschatz, Avi; Galvin, Peter; Gagne, Greg (2008). 《Operating Systems Concepts》 . John Wiley & Sons. ISBN 0-470-12872-0.
- O'Brien, J. A., & Marakas, G. M.(2011). *Management Information Systems*. 10e. McGraw-Hill Irwin.
- Leva, Alberto; Maggio, Martina; Papadopoulos, Alessandro Vittorio; Terraneo, Federico (2013). 《Control-based Operating System Design》 . IET. ISBN 978-1-84919-609-3.

- Arpaci-Dusseau, Remzi; Arpaci-Dusseau, Andrea (2015). 《Operating Systems: Three Easy Pieces》 (<http://pages.cs.wisc.edu/~remzi/OSTEP/>).

외부 링크

- (영어) 운영체제 (https://curlie.org/Computers/Software/Operating_Systems) - Curlie
 - 운영체제의 동작 원리 (<http://computer.howstuffworks.com/operating-system.htm>)
 - 세계 최초의 컴퓨터 운영체제 (<http://millosch.wordpress.com/2007/09/07/the-worlds-first-computer-operating-system-implemented-at-general-motors-research-labs-in-warren-michigan-in-1955/>)
 - 운영체제에 관한 TUNES 리뷰 (<https://web.archive.org/web/20030810110648/http://tunes.org/Review/OSes.html>)
-

원본 주소 "<https://ko.wikipedia.org/w/index.php?title=운영체제&oldid=33838682>"