응용보안

1. 운영체제 이해

경기대학교 AI컴퓨터공학부 이재흥
jhlee@kyonggi.ac.kr

KGU 경기대학교

CONTENTS
PRESENTATION

- 운영체제의 개념과 기능
- 윈도우의 이해
- 리눅스/유닉스의 이해

학습 목표

- 운영체제의 기능과 목적을 이해한다.
- 운영체제의 구성 요소와 각 기능을 이해한다.
- 운영체제의 종류와 특징을 이해한다.
- 운영체제별 프로세스를 이해하고 구별할 수 있다.

운영체제의 개념과 기능

운영체제의 개념과 기능

- 운영체제
  - 사용자가 컴퓨터 시스템을 손쉽게 사용하도록 하고, 시스템 자원(기억 장치, 프로세서, 입출력 장치, 정보, 네트워크 등)을 효율적으로 관리할 수 있도록 하는 프로그램 집합

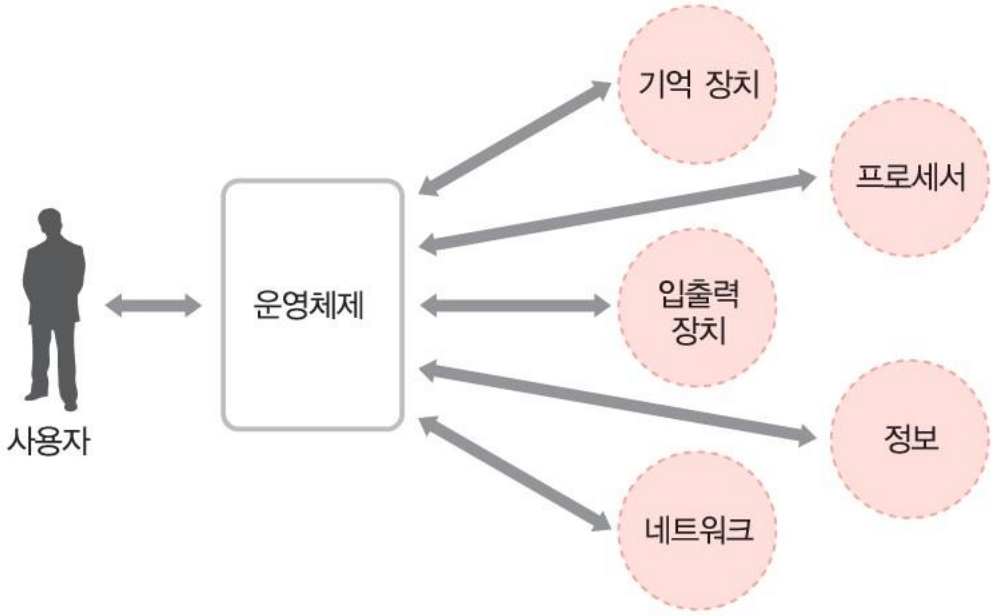


그림 1-1 운영체제 역할

운영체제의 개념과 기능

- 운영체제의 기능
  - 일반 PC는 단일 사용자 운영체제 구성 모델, 즉 사용자 명령 인터페이스를 중심으로 메모리 관리자, 프로세서 관리자, 장치 관리자, 파일 관리자 등 네 가지 서브 시스템 관리자로 기본 구성됨
  - 네트워크를 지원하는 운영체제에는 네트워크 관리자가 추가됨

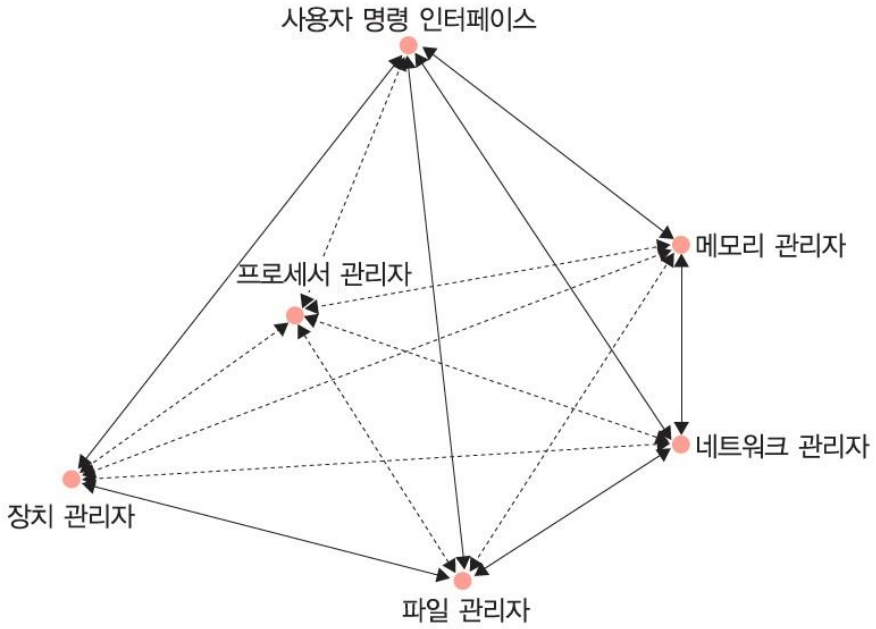


그림 1-2 단일 사용자 운영체제 구성 모델

운영체제의 개념과 기능

- 운영체제 구성요소
  - 사용자 명령 인터페이스
    - 사용자와 시스템의 대화 수단
    - CLI(Command Line Interface), GUI(Graphic User Interface)
  - 메모리 관리자
    - 프로그램이 메모리를 요청하면 적합성을 점검하고, 적합하다면 메모리를 할당
    - 할당된 메모리를 다른 프로그램이 접근하지 못하게 관리하고 보호
    - 프로그램을 종료할 때는 할당된 메모리를 회수
  - 프로세서 관리자
    - 명령어들을 체계적이고 효율적으로 실행할 수 있도록 작업 스케줄링하고 사용자의 작업 요청을 수용하거나 거부

운영체제의 개념과 기능

- 운영체제 구성요소
  - 장치 관리자
    - 프린터, 디스크 드라이버, 모뎀, 모니터 등 시스템 안의 모든 장치(Device)를 프로그램에 할당하거나 회수
  - 파일 관리자
    - 모든 파일을 대상으로 사용자별로 파일 접근(Access) 권한을 부여
    - 접근 권한에 따라 파일을 할당(Open)하고 해제(Close)
  - 네트워크 관리자
    - 네트워크에서 접근 가능한 CPU, 메모리, 프린터, 디스크 드라이버, 모뎀, 모니터 같은 자원을 관리

윈도우의 이해

윈도우의 역사

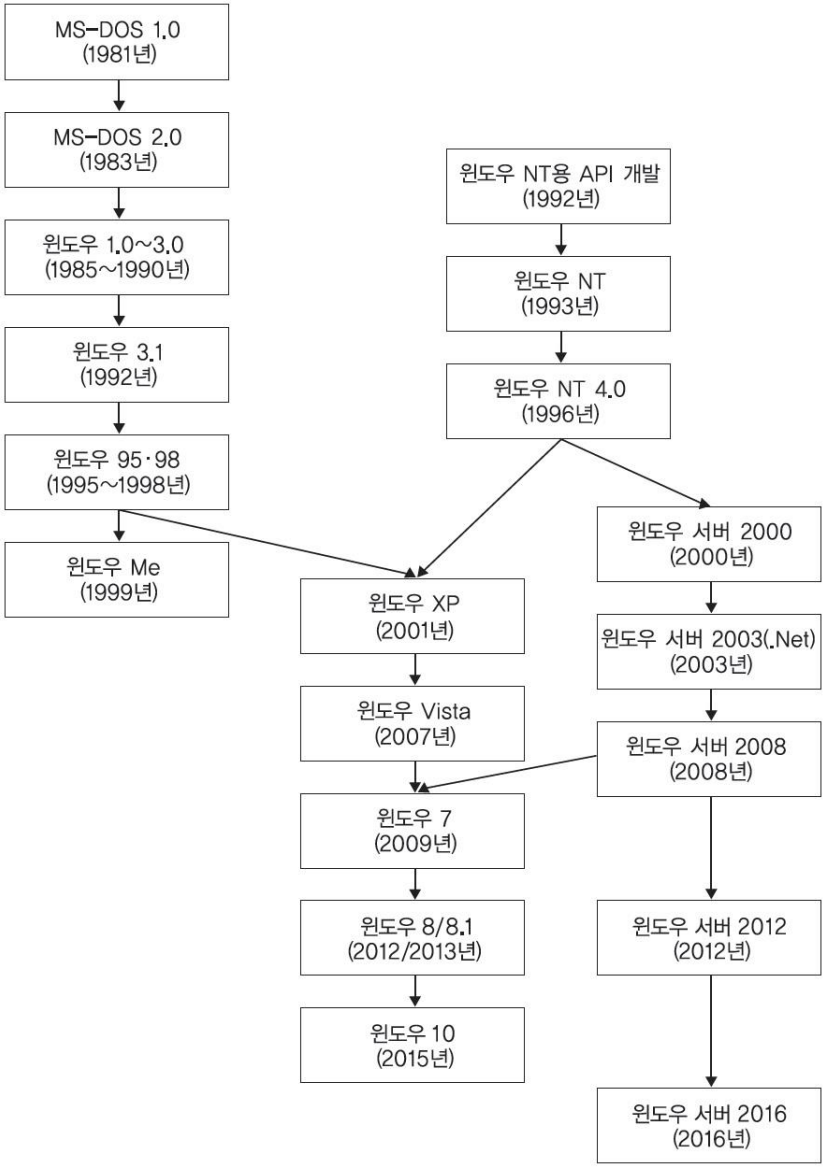


그림 1-3 **윈도우 발전 과정**

윈도우의 역사

- 참고 영상
  - 윈도우(Windows)의 역사 : 인류 역사에 기록될 소프트웨어
    - https://youtu.be/a2yItDga9_o
  - 윈도우 시작과 끝, Windows 역사 총정리 | POST IT
    - https://youtu.be/W7gGo16SD1E

윈도우의 구조

- 커널
  - 인터럽트(Interrupt) 처리, 프로세스 관리, 메모리 관리, 파일 시스템 관리, 프로그래밍 인터페이스 제공 등 운영체제 기본 기능을 제공하는 핵심
- 윈도우 구조

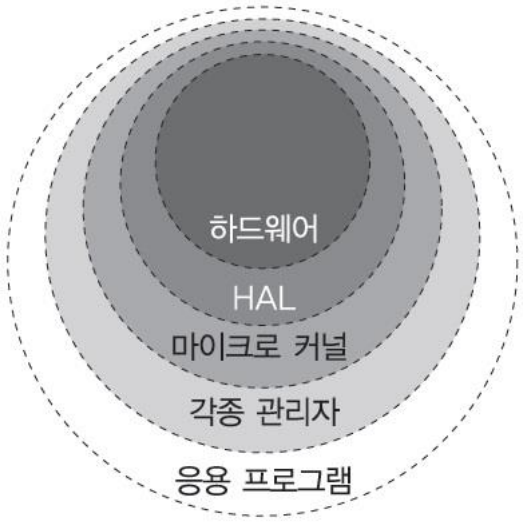


그림 1-4 윈도우 링 구조

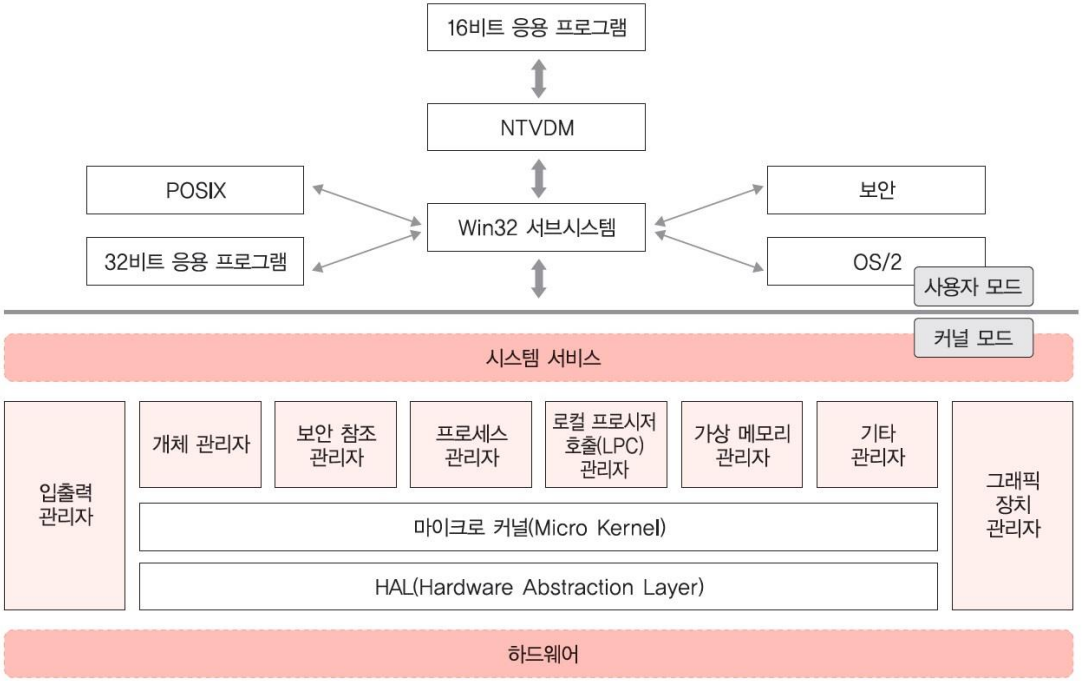


그림 1-5 윈도우 시스템 구조

윈도우 관리자의 역할

- 입출력 관리자(I/O Manager)
  - 장치 드라이버 사이에서 메시지를 전달함으로써 시스템의 입출력을 제어
  - 응용 프로그램이 하드웨어와 곧바로 통신할 수 있는 통로를 제공
- 개체 관리자(Object Manager)
  - 파일, 포트, 프로세스, 스레드 등의 개체 정보를 제공
- 보안 참조 관리자(Security Reference Monitor)
  - 각 데이터나 시스템 자원의 제어를 허가 하거나 거부함으로써 시스템의 강제 보안 설정을 책임짐
- 프로세스 관리자(Process Manager)
  - 스레드를 생성하고 요청에 따라 처리

윈도우 관리자의 역할

- 로컬 프로시저 호출 관리자(Local Procedure Call Manager)
  - 프로세스 간 통신 수행
- 가상 메모리 관리자(Virtual Memory Manager)
  - 응용 프로그램의 요청에 따라 RAM 메모리를 할당
  - 가상 메모리의 페이징(Paging)을 제어
- 그래픽 장치 관리자(Graphics Device Interface)
  - 화면에 선이나 곡선을 그리거나 폰트 등을 관리
- 기타 관리자
  - 캐시(Cache) 관리자
  - PNP(Plug and Play) 관리자
  - 전원 관리자

윈도우의 파일 시스템

- FAT
  - 윈도우 95
  - FAT 테이블의 기본 크기 16비트 + 클러스터(Cluster) 크기 32KB
    - 전체 가능 용량 2GB(이유는?)
- FAT32
  - 윈도우 95 OSR(OEM Service Release)2에서 처음 도입
  - FAT 테이블의 기본 크기를 32비트로 확장
  - 접근 제어를 설정할 수 없다는 문제점 존재
- NTFS(New Technology File System)
  - 개별 폴더와 파일에 사용 권한 설정 가능
  - 각 파일과 폴더에 해당 계정만 접근하여 읽을 수 있게 암호화 가능
  - 감사(Auditing) 기능 제공

윈도우의 부팅 순서

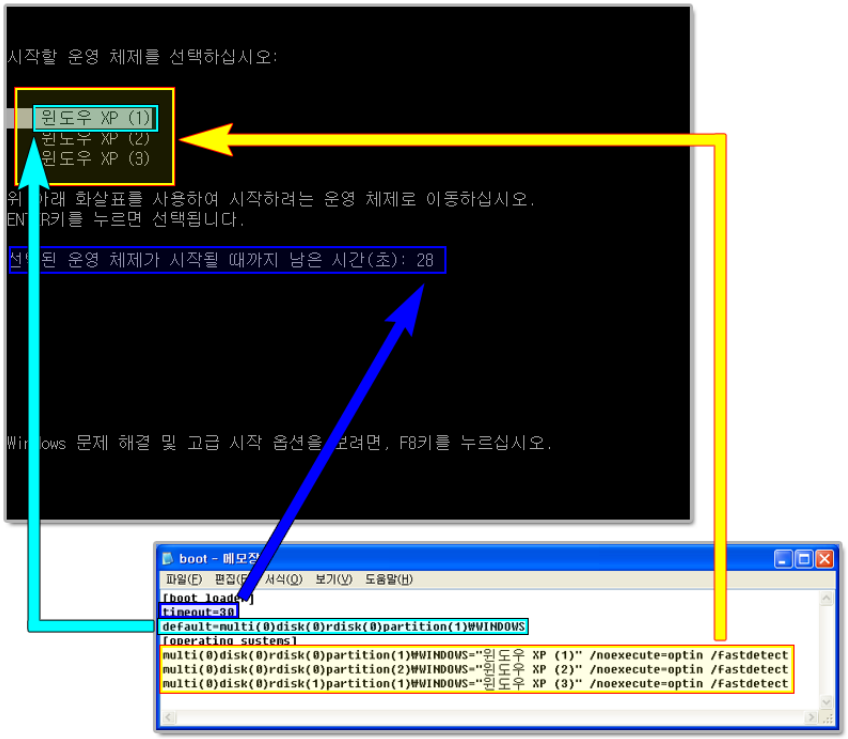
- 윈도우 XP, 윈도우 서버2000/2003의 부팅 순서
  - 1단계. POST(Power On Self Test) 실행
    - 하드웨어 스스로 시스템에 문제가 없는지 기본 사항을 검사하는 과정
    - BIOS(Basic Input/Output System)가 POST를 실행

윈도우의 부팅 순서

- 윈도우 XP, 윈도우 서버2000/2003의 부팅 순서
  - 2단계. 기본 부팅 관련 설정사항 로드
    - BIOS가 CMOS(Complementary Metal-Oxide Semiconductor)에 설정된 시스템 설정 사항 및 부팅과 관련된 여러 가지 정보를 읽어 시스템에 적용
  - 3단계. MBR(Master Boot Record, 마스터 부트 레코드) 로드
    - MBR은 저장 매체의 첫 번째 섹터(LBA 0)에 위치하는 512바이트 영역으로, 부팅 매체의 기본 파일 시스템 정보가 들어 있음
    - 운영체제를 부팅할 때 저장 매체의 첫 번째 섹터를 호출하면 해당 부트 코드를 수행
    - 부트 코드의 주 역할은 파티션 테이블에서 부팅 가능한 파티션을 찾아 해당 파티션의 부트 섹터를 호출하는 것
    - MBR과 관련한 메시지 중 'Missing Operating System'은 운영체제를 설치하지 않았거나 CMOS에서 부팅 매체를 잘못 설정했을 때 확인할 수 있음

윈도우의 부팅 순서

- 윈도우 XP, 윈도우 서버2000/2003의 부팅 순서
  - 4단계. NTLDR(NT LoaDeR) 실행
    - 하드 디스크의 부팅 파티션에 있는 프로그램으로 윈도우를 부팅할 수 있도록 간단한 파일 시스템을 실행하고, boot.ini 파일 내용을 읽어 가능한 부팅 옵션을 보여줌



윈도우의 부팅 순서

- 윈도우 XP, 윈도우 서버2000/2003의 부팅 순서
  - 5단계. NTDETECT.com 실행
    - 시스템에 설치한 다음 하드웨어를 검사
      - PC의 CPU 유형
      - 버스 유형
      - 비디오 보드 유형
      - 키보드와 마우스 종류
      - 컴퓨터에 장착되어 있는 직렬 포트와 병렬 포트
      - 플로피 드라이브

윈도우의 부팅 순서

- 윈도우 XP, 윈도우 서버2000/2003의 부팅 순서
  - 6단계. ntoskrnl.exe(NT OS Kernel) 실행 – HAL.DLL을 로드
    ① 커널 로드
      - 시스템 설정을 로드하고, 이것을 HKEY_LOCAL_MACHINE₩System₩CurrentControlset₩Services 에 저장
      - 이 정보를 확인하여 로드할 드라이브와 그 순서를 결정

    ② 커널 초기화
      - 드라이버에서 현재 제어 설정을 검사하고 작업을 시작

    ③ 서비스 로드
      - 세션 관리자 서브 시스템(smss.exe)과 Win32 서브시스템을 로드

윈도우의 부팅 순서

- 윈도우 XP, 윈도우 서버2000/2003의 부팅 순서
  - 6단계. ntoskrnl.exe(NT OS Kernel) 실행 – HAL.DLL을 로드
    ④ 서브 시스템 시작
    - 윈도우 서브시스템을 초기화
    - Win32 서브시스템은 로그인을 처리하고 Winlogon.exe를 시작
    - Ctrl + Alt + Delete를 누르면 로그인 창을 활성화하고, 계정과 패스워드를 입력 받아 로컬 보안 인증 서버(Local Security Authentication Server, lsass.exe)에 보냄
    - 계정과 패스워드를 전달받은 로컬 보안 인증 서버는 보안 계정 관리자(SAM, Security Accounts Manager)에 저장된 정보와 비교하여 서로 일치하면 Userinit.exe 프로세스가 HKEY_LOCAL_MACHINE₩SOFTWARE₩Microsoft₩Windows NT₩CurrentVersion₩Winlogon의 셸 값에서 참조되는 셸을 실행

윈도우의 부팅 순서

- 윈도우 비스타 이후 버전 부팅 순서
  - 1~3단계. POST 실행 및 기본 부팅 관련 설정사항 로드, MBR 로드
    - 앞에서와 같음
  - 4단계. 윈도우 부트 서브시스템(Window Boot Manager) 실행
    - MBR에서 NTLDR을 실행하지 않고 윈도우 부트 서브시스템을 실행
    - 윈도우 부트 서브시스템은 bootmgr.exe를 실행하고 부트 설정 데이터(BCD, Boot Configuration Data)를 읽어 실행 가능한 운영체제의 목록을 보여줌
    - 부트 설정 데이터는 bcdedit.exe를 이용하여 편집이 가능함
  - 5단계. 윈도우 OS 로더(Winload.exe) 실행
    - 윈도우 OS 로더인 Winload는 NTDETECT처럼 각종 장치 드라이브를 로드하고 ntoskrnl.exe을 실행함

리눅스/유닉스의 이해

리눅스/유닉스의 역사

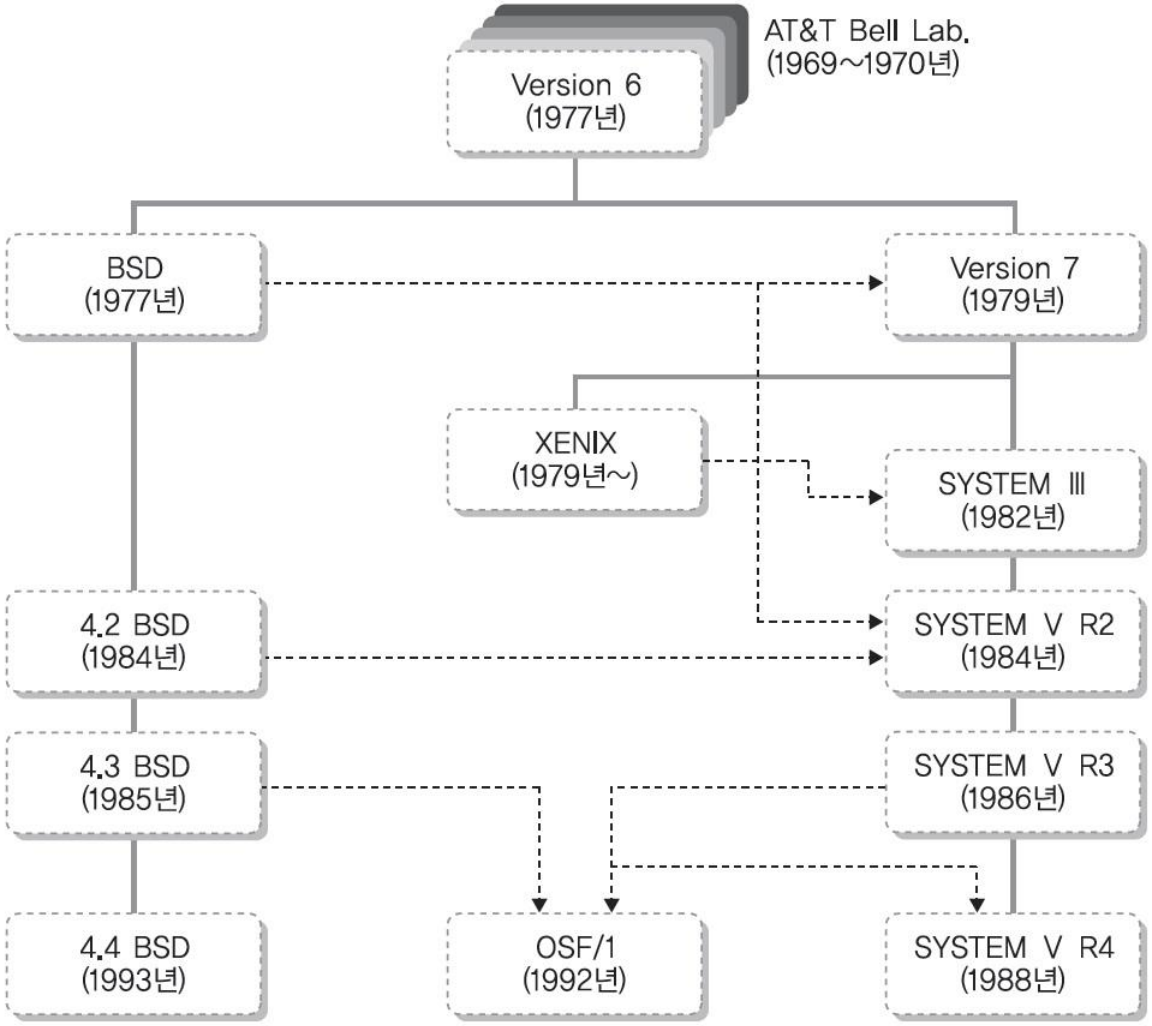


그림 1-6 유닉스 시스템 발전 과정

리눅스/유닉스의 역사

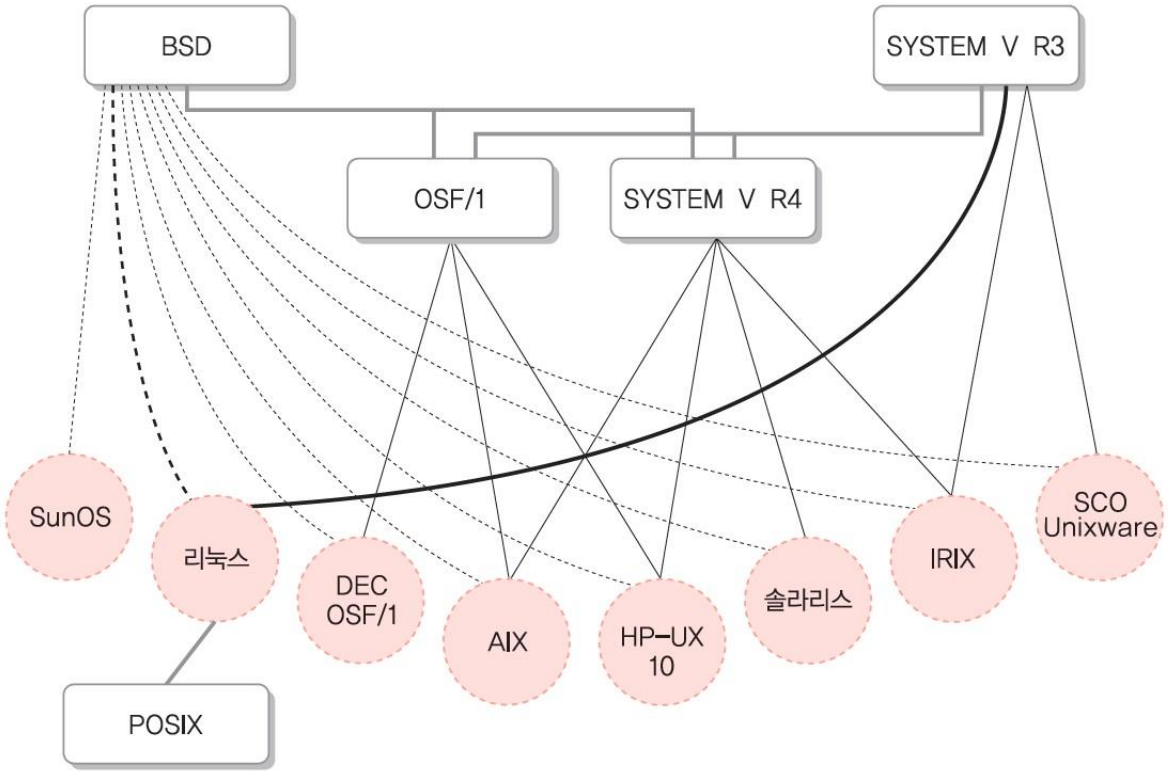


그림 1-7 유닉스 시스템 분화 과정

리눅스/유닉스의 역사

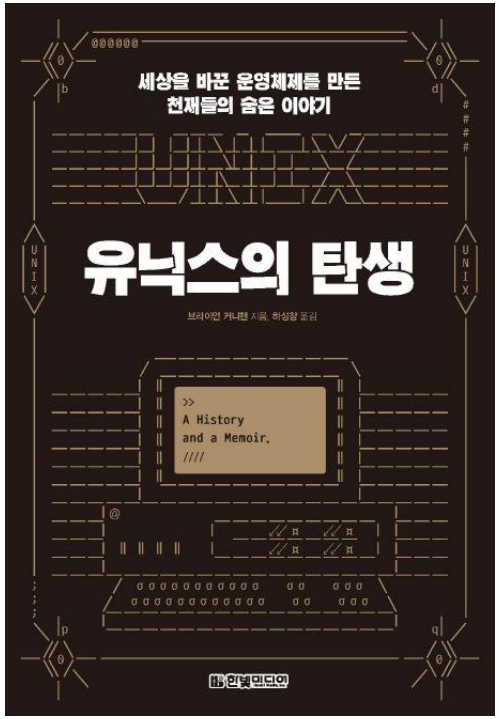
- 리눅스/유닉스의 역사
  - 유닉스 시초는 멀틱스(Multics)라는 시분할 운영체제로 1963년부터 3년 동안 MIT, GE(General Electric), 벨 연구소가 공동으로 개발
  - 멀틱스는 미국 국방성(ARPA)의 지원으로 추진된 프로젝트로 GE가 하드웨어를 만들었고 PL/1 (Programming Language 1) 언어로 개발
  - 1970년대에 벨 연구소가 멀틱스에서 손을 떼자 벨 연구소에서 근무하던 데니스 리치(Dennis Ritchie)는 파견지에서 돌아와 켄 톰슨(Ken Thomson)과 함께 PDP- 7에 운영체제를 만들기 시작하며 멀틱스의 여러 개념을 구현하는데 힘쓰기 시작
  - 이때 PDP-7에 구현한 운영체제에는 파일 시스템 구성의 개념, 사용자가 명령을 내려 바로 실행하는 명령어 인터프리터(Command Interpreter)의 개념, 각 명령이 새로운 프로세스를 형성해서 실행하도록 하는 개념이 모두 포함되어 있었음

리눅스/유닉스의 역사

- 리눅스/유닉스의 역사
  - 1973년 C 언어로 다시 쓴 유닉스는 C 언어 범용성 바람을 타고 여러 하드웨어에 이식되었으며, 사람들의 관심을 받기 시작함
  - 1970년대 말 버클리대학교는 AT&T 연구소에서 유닉스 소스 코드를 400달러에 구입하였고, 버클리대학교 학생인 빌 조이(Bill Joy)와 척 핼리(Chuck Haley)는 유닉스 소스 코드를 조금씩 수정하여 발전시켜 BSD(Berkeley Software Distribution)라는 이름으로 당시 50달러 가격으로 판매
  - 리눅스는 1991년에 핀란드 학생 리누스 토르발스가 개인 프로젝트로 개발한 오픈 소스 운영체제 커널

리눅스/유닉스의 역사

- 참고 서적
  - 유닉스의 탄생
    - 브라이언 커니핸 지음, 하성창 옮김 (한빛미디어)
    - 소개 동영상
      - https://youtu.be/fLOEHLRW-ZU



리눅스/유닉스의 역사

- 참고 동영상
  - The Rise of Unix. The Seeds of its Fall.
    - https://youtu.be/HADp3emVABg
  - The History of UNIX
    - https://youtu.be/AEsdyAeumVQ
  - AT&T Archives: The UNIX Operating System
    - https://youtu.be/tc4ROCJYbm0

리눅스/유닉스의 구조

- 유닉스 링 구조
  - 하드웨어, 커널, 셸, 응용 프로그램으로 구성

그림 1-8 유닉스 링 구조

리눅스/유닉스의 구조

- 유닉스 커널 구조
  - 윈도우보다 훨씬 단순, 크게 파일 서브 시스템, 장치 드라이버, 프로세스 제어로 나뉘며, 커널의 파일 크기 또한 윈도우의 1/3 정도

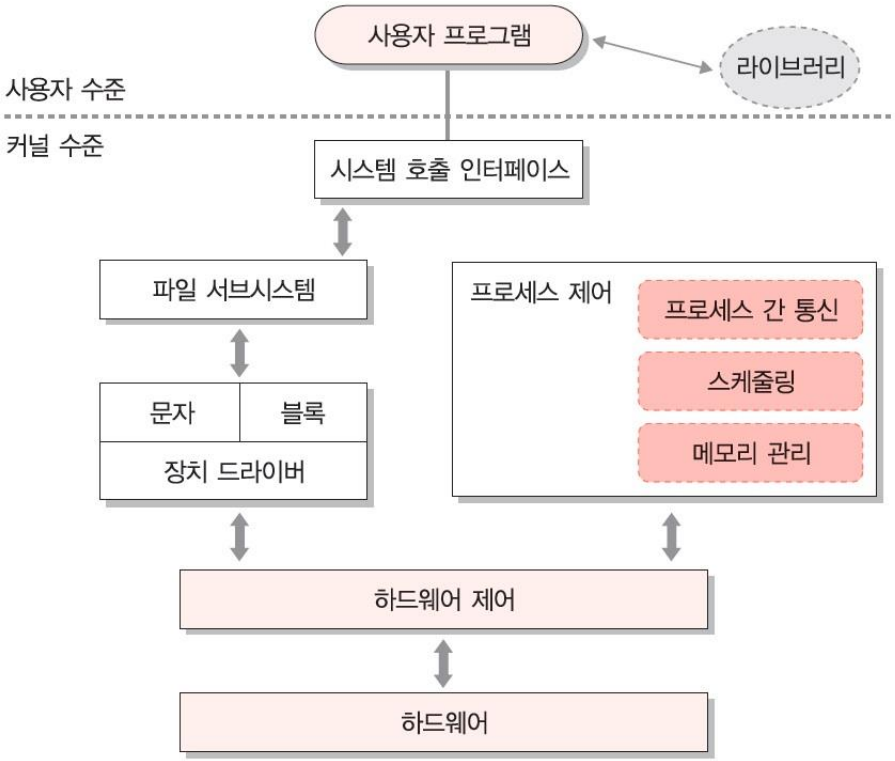


그림 1-9 유닉스 시스템 구조

리눅스/유닉스의 구조

- 유닉스 시스템 구조
  - 프로세스 제어
    - 전체 프로세스 간 통신, 스케줄링, 메모리 관리를 구현
  - 장치 드라이버
    - 윈도우에서처럼 하드웨어와 소프트웨어를 연결해주는 인터페이스를 제공
  - 파일 서브 시스템
    - 하드 디스크와 같은 저장 공간에 유닉스의 파일을 저장하고 읽는 역할

리눅스/유닉스의 구조

- 셸
  - 응용 프로그램에서 명령을 받아 커널에 전송하는 역할
  - 사용자의 키보드 입력 인식 해당 프로그램을 수행

- 셸(Shell)의 종류
  - 본 셸(Bourne Shell), 콘 셸(Korn Shell), C 셸(C Shell)

리눅스/유닉스의 구조

- 셸이 제공하는 주요 기능
  - 자체의 내장 명령어 제공
  - 입력/출력/오류의 방향 변경(redirection)
  - 와일드카드(wildcard)
  - 파이프라인
  - 조건부/무조건부 명령 열 작성
  - 서브 셸 생성
  - 백그라운드 처리(Background processing)
  - 셸 스크립트(프로그램) 작성

리눅스/유닉스의 구조

- 리눅스/유닉스의 파일 시스템
  - 일반 파일 : 일반적으로 생각하는 데이터 파일이나 실행 파일
  - 디렉터리 : 유닉스에서는 디렉터리도 파일에 해당, 디렉터리가 담고 있는 여러 파일과 하위 디렉터리 정보가 담겨 있음
  - 특수 파일 : 프린터나 터미널, 테이프 드라이버 같은 물리적인 장치를 특수 파일을 통해 접근, 특수 파일은 /dev(device)에 있음
  - 파이프 파일 : | 문자를 말하며, 명령 두 개를 연결할 때 사용, 임시 파일이 생성 되었다가 명령 수행을 마치면 사라지는 것으로, 이 파일을 파이프 파일이라고 함

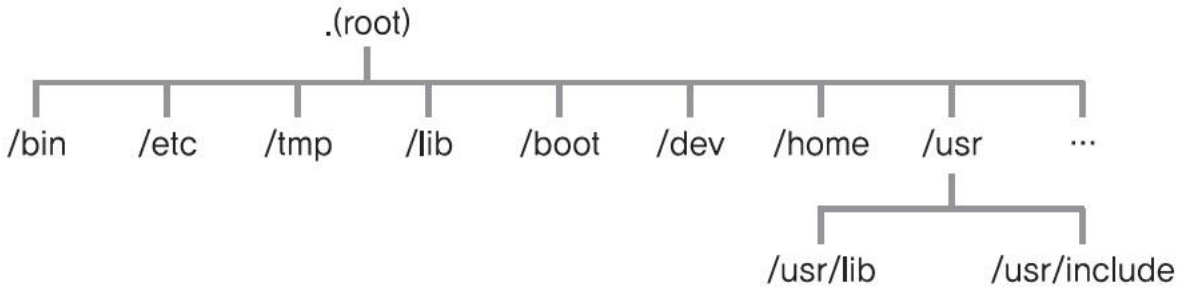


그림 1-10 유닉스 디렉터리 구조

리눅스/유닉스의 구조

표 1-1 유닉스 시스템 디렉터리별 역할

디렉터리	설명
/bin	기본적으로 실행 가능한 파일을 담고 있다. 예: echo, mv, copy, pwd, who 등
/etc	시스템의 환경설정 및 주요 설정 파일을 담고 있다. 예: passwd, hosts, xined.conf 등
/tmp	프로그램 실행 및 설치할 때 생성하는 임시 파일을 담고 있다. 이 디렉터리에 파일을 저장하면 재부팅할 때 임의로 삭제될 수 있다.
/lib	기본 프로그램의 모듈을 담고 있다.
/boot	커널용 프로그램 파일을 담고 있으며, 부팅할 때 읽어서 수행한다.
/dev	프린터나 터미널 같은 물리적인 장치를 다루는 특수 파일을 담고 있다.
/home	각 사용자의 작업 디렉터리를 담고 있다. 각 계정으로 로그인할 때 이 디렉터리 아래에 있는 자신의 작업 디렉터리가 시작 디렉터리가 된다.
/usr	사용자가 직접 쓰는 파일을 담고 있다. 다른 디렉터리에 있는 파일이 똑같이 위치할 때가 많은데, 이는 링크되어 있는 것이다.
/usr/lib	C 언어나 포트란 라이브러리를 담고 있다.
/usr/include	C 언어에서 사용하는 헤더 파일을 담고 있다.

리눅스/유닉스의 부팅 순서

- 일반적인 유닉스 부팅 순서
  - 1단계. POST(Power On Self Test) 실행
  - 2단계. 기본 부팅 관련 설정 사항 로드
  - 3단계. MBR(Master Boot Record, 마스터 부트 레코드) 로드
  - 4단계. 부트 로더(Boot Loader) 실행
    - 리눅스는 부트 로더로 LILO(LInux LOader)와 GRUB(GRand Unified Bootloader, GNU 프로젝트의 부트 로더)를 사용
    - 설정 사항 저장 위치
      - LILO : /etc/lilo.conf
      - GRUB : /etc/grub.conf(/boot/grub/grub.conf)

리눅스/유닉스의 부팅 순서

- 일반적인 유닉스 부팅 순서
  - 5단계. 실행 레벨에 따른 서비스 실행
    - 부트 로더는 스와퍼(Swapper)라는 pid 0번 프로세스를 실행하고, 스와퍼는 다시 pid 1번 init(/sbin/init) 프로세스를 실행
    - init 프로세스는 다시 /etc/inittab 파일을 읽음
    - inittab 파일은 부팅할 기본 모드를 선택하여 그에 따른 환경을 제공하는 분기점이라고 할 수 있음

```
# /etc/init.d/rc takes care of runlevel handling
#
# runlevel 0  is  System halt   (Do not use this for initdefault!)
# runlevel 1  is  Single user mode
# runlevel 2  is  Local multiuser without remote network (e.g. NFS)
# runlevel 3  is  Full multiuser with network
# runlevel 4  is  Not used
# runlevel 5  is  Full multiuser with network and xdm
# runlevel 6  is  System reboot (Do not use this for initdefault!)
#
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
#l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

리눅스/유닉스의 부팅 순서

- 일반적인 유닉스 부팅 순서
  - 5단계. 실행 레벨에 따른 서비스 실행
    - inittab 파일 : 프로그램 종류에 따라 다른 수준의 실행 레벨(Run Level) 부여
      - 실행 레벨 0 : 시스템을 종료할 때 사용
      - 실행 레벨 1 : 단일 사용자 모드(Single User Mode)로, 기본적으로 관리자 권한의 셸을 얻으나 대부분 데몬을 실행하지 않으므로 기능이 제약
      - 실행 레벨 2 : NFS(Network File System)를 지원하지 않는 다중 사용자 모드
      - 실행 레벨 3 : 일반 셸 기반의 인터페이스를 가진 다중 사용자 모드
      - 실행 레벨 4 : 기본적으로 사용되지 않지만 사용자가 임의로 정의하여 사용할 수 있음
      - 실행 레벨 5 : 기본은 실행 레벨 3과 같으나 GUI 환경을 지원
      - 실행 레벨 6 : 재부팅

THANK YOU!