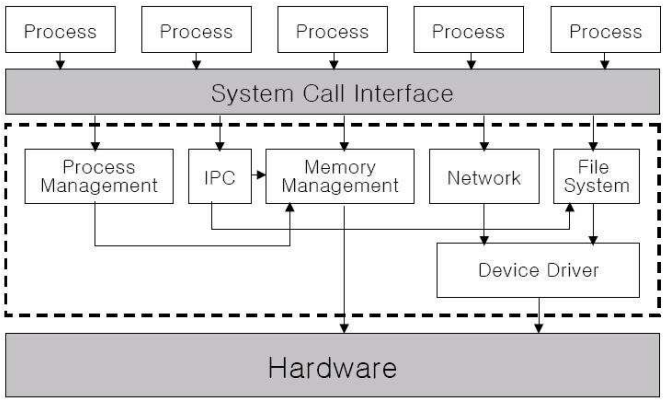
Slide 0

1부 개요
(3장. 운영체제 구조)

운영체제 구성

Slide 1

- 시스템 구성
  - 프로세스 관리
  - 주기억장치 관리
  - 파일관리
  - 입출력 관리
  - 보조기억장치 관리



운영체제 구성 (Cont'd)

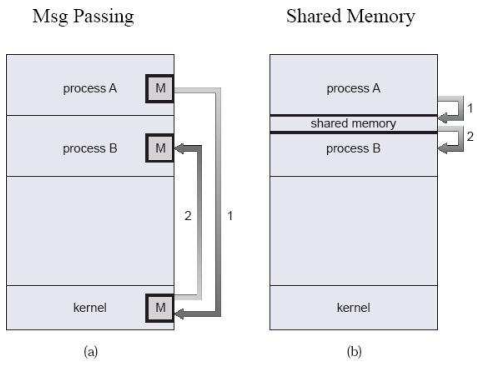
Slide 2

- 프로세스 관리
  - 설명
    - 운영체제는 사용자가 실행시키고자 하는 프로그램을 메모리에 적재하여 작업을 수행시키는 기능을 제공
    - 프로세스: 실행 중인 프로그램
    - 하나의 프로그램은 여러개의 프로세스를 생성할 수 있음
    - 프로세스는 하나의 작업 단위
    - 운영체제 프로세스와 사용자 프로세스로 구분
  - 프로세스 관련 운영체제의 기능
    - 사용자 프로세스와 시스템 프로세스의 생성과 제거
    - 프로세스의 중지와 재개 (스케쥴링)
    - 프로세스 동기화를 위한 기법 제공
    - 프로세스간 통신을 위한 기법 제공
    - 교착상태(deadlock) 방지를 위한 기법 제공

운영체제 구성 (Cont'd)

Slide 3

- 프로세스간 통신
  - IPC는 프로세스간 자원을
    - 공유 (resource sharing)
    - 동기화 (synchronize) 그리고
    - 데이터 교환 (exchanging data)을 위한 수단으로 사용됨
  - 종류
    - 공유 메모리(Shared memory)
    - 세마포어(Semaphore)
    - 시그널(Signal)
    - 파이프(Pipe)
    - 메시지 큐(Message queue)



운영체제 구성 (Cont'd)

Slide 4

- 주기억장치 관리
  - 설명
    - 사용자 프로그램이 실행되기 위해서는 저장 장치에서 읽어서 주기억 장치로 적재되어야 함
  - 주기억장치 관련 운영체제의 기능
    - 기억 공간의 사용 현황 관리
    - 적재할 프로세스 선택
    - 기억 공간의 할당과 회수
    - 다른 프로세스로부터의 보호

운영체제 구성 (Cont'd)

Slide 5

- 파일관리 (File Management)
  - 설명
    - 운영체제는 입력 및 출력의 대상으로 파일이라는 개념을 사용
    - 파일은 연관된 자료의 집합으로 정의되며, 프로그램 및 데이터와 같은 정보를 포함
  - 파일관련 운영체제의 기능
    - 파일의 생성과 삭제
    - 디렉토리 (폴더)의 생성과 삭제
    - 파일과 디렉토리를 조작하기 위한 기초 연산 제공
    - 보조기억장치와 파일 간에 매핑
    - 파일의 손실 방지 및 복구
    - 백업

운영체제 구성 (Cont'd)

Slide 6

- 입출력 시스템 관리
  - 설명
    - 실행중인 프로세스가 입출력을 요구하는 경우
    - 운영체제는 프로세스를 대신해서 입출력 장치로부터 작업을 요청하고 결과를 프로세스에게 돌려줌
      →인터럽트(interrupt mechanism)
    - 보통 입출력 하위시스템 (I/O subsystem)을 사용하여 이 기능을 제공
  - 입출력 하위시스템
    - 버퍼링, 캐싱, 스풀링과 같은 기억장치 관리 구성요소
    - 일반적인 장치구동기 (device driver) 인터페이스
    - 특정 하드웨어를 위한 장치구동기

운영체제 구성 (Cont'd)

Slide 7

- 보조기억장치 관리
  - 설명
    - 주기억 장치(primary storage)는 휘발성이며 용량이 작기 때문에
    - 모든 데이터와 프로그램을 영구적으로 보관하기 어려움
    - 따라서 저장 장치를 이용하여 주기억 장치를 백업해야 함
  - 보조기억장치 관련 운영체제의 기능
    - 빈 공간 관리
    - 저장 공간 할당
    - 디스크 스케쥴링: 디스크와 관련된 요청의 실행 순서를 결정하는 문제

운영체제 구성 (Cont'd)

Slide 8

- 네트워킹
  - 설명
    - 네트워크 접근을 파일 접근과 같은 형태로 일반화 시킴
  - 네트워킹을 위한 운영체제의 기능
    - 네트워킹을 위한 각종 통신 프로토콜을 지원
    - 각종 네트워크 장치구동기를 위한 인터페이스 제공

운영체제 구성 (Cont'd)

Slide 9

- 보호 시스템 (Protection system)
  - 설명
    - 보호(Protection)는 프로세스 또는 사용자가 시스템 자원을 사용함에 있어서
    - 다른 프로세스 또는 사용자로부터 접근되는 것을 제어 할 수 있게 함
  - 운영체제의 기능
    - 프로세스간 보호 사용자간 보호 자원사용에 대한 보호를 제공
    - 주기억 장치 보호, 파일 보호등

운영체제 구성 (Cont'd)

Slide 10

- 명령 해석기 (command-interpreter)
  - 설명
    - 사용자의 입력을 해석해서 해당하는 작업을 수행하는 프로그램
    - 커널에 구현된 운영체제도 있음
    - MS-DOS 나 UNIX 에서는 커널에 있지 않고 특수한 목적의 프로그램으로 취급
    - 키보드나 마우스 입력을 받아 해석하여 처리함
  - 운영체제의 기능
    - 로그온시 명령 해석기를 구동시킴
    - 명령 해석기의 해석 결과에 따라 해당 작업을 수행

운영체제 서비스

Slide 11

- 프로그래머의 편리성을 위해 제공되는 서비스
  - 프로그램 실행
  - 입출력 수행
  - 파일시스템 조작
  - 통신: 공유메모리, 메시지 전달
  - 오류 검출
- 시스템의 효율적인 운영을 위해 제공되는 서비스
  - 자원할당
  - 자원 사용 정보
  - 보호 (protection)

시스템 호출

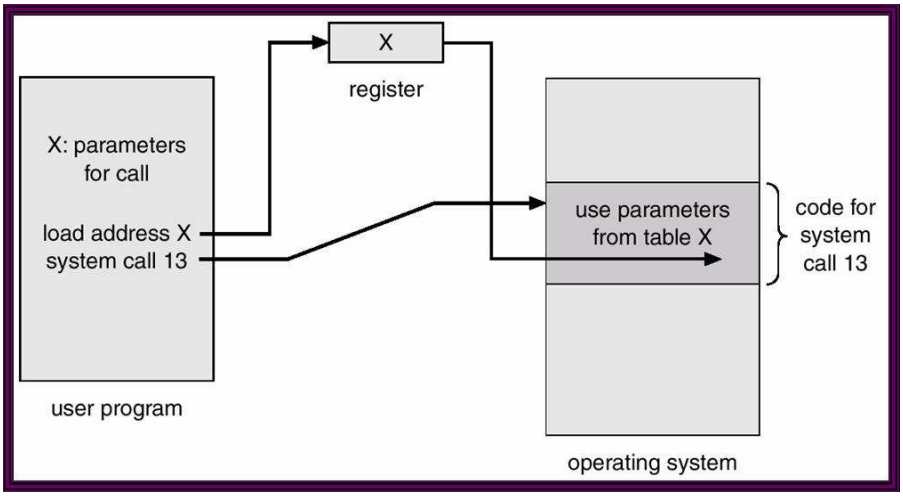
Slide 12

- 설명
  - 사용자 프로세스가 운영체제의 기능을 요청하는 것을 말함
  - 커널이 사용자 프로그램에게 제공하는 프로그래밍 인터페이스
  - 사용자 모드에서 커널 모드로 전환됨
  - 인텔 프로세서에서는 0x80 인터럽트를 이용 구현
- 시스템 호출시 파라미터 전달 방법
  - 레지스터를 통해 직접 전달 (파라미터가 적을 때만 가능)
  - 메모리 블럭이나 테이블에 저장한후 그 주소를 레지스터로 전달
  - 스택(stack)을 통해 전달

시스템 호출 (Cont'd)

Slide 13

- 테이블을 통한 파라미터 전달



시스템 호출 (Cont'd)

Slide 14

- 유형
  - 입출력 조작: 입출력 디바이스에 대한 접근
  - 프로세스 제어: 프로세스가 자기 자신의 실행을 제어할 수 있도록
  - 프로세스간 통신: 다른 프로세스로 정보를 주고 받을 수 있도록 하기
  - 타이밍 서비스
  - 상태 정보의 제공

시스템 호출 (Cont'd)

Slide 15

- 일반적인 시스템 호출형태
  - 프로세스 제어 (Process control)
    - 종료(end), 취소(abort)
    - 적재(load), 수행(execute)
    - 생성(creation), 종료(temination)
    - 속성 (attribute)획득 및 설정
    - 대기시간
    - 사건대기, 시그널 (signal event)
    - 메모리 할당 해제
  - 파일 관리 (File management)
    - 생성 및 삭제
    - 열기 및 닫기
    - 읽기, 쓰기 및 재배치
    - 속성 획득 및 설정

시스템 호출 (Cont'd)

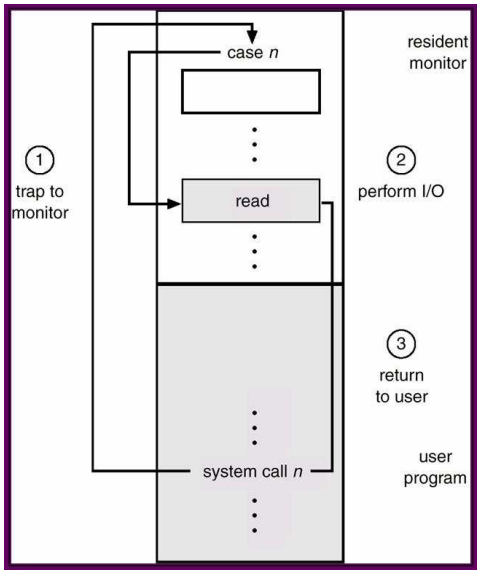
Slide 16

- 장치 관리 (Device management)(유닉스, 리눅스에서는 파일로 처리)
  - 장치 사용 요청 및 해제
  - 읽기, 쓰기, 재배치
  - 속성 획득 및 설정
  - 장치의 논리적 부착 및 제거
- 정보 유지보수
  - 시간과 날짜 요청
  - 시스템 정보 획득 및 설정
  - 프로세스, 파일, 장치속성 획득 및 설정
- 통신 (Communication)
  - 통신 연결 생성 및 제거
  - 데이터 전송 및 수신
  - 상태 정보 전달
  - 원격장치 부착 및 제거

시스템 호출 (Cont'd)

Slide 17

- 시스템 호출 개념도



시스템 호출 (Cont'd)

Slide 18

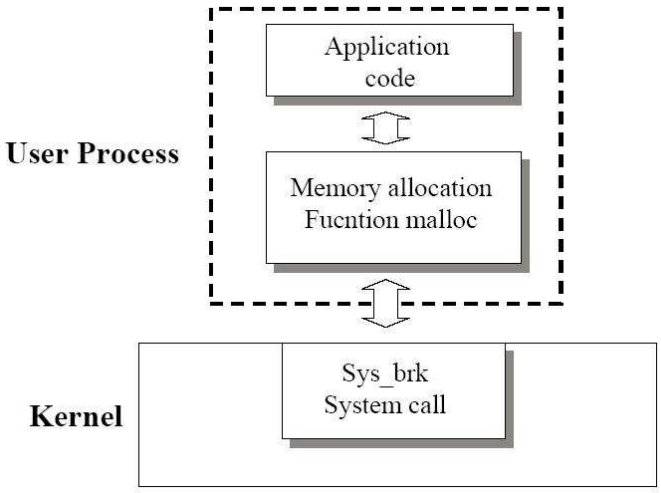
- 시스템 호출 함수 목록-리눅스 커널 예

번호	함수명	소스코드 위치
1	sys_exit	kernel/exit.c
2	sys_fork	arch/i386/kernel/process.c
3	sys_read	fs/read_write.c
4	sys_write	fs/read_write.c
5	sys_open	fs/open.c
6	sys_close	fs/open.c
7	sys_waitpid	kernel/exit.c
8	sys_creat	fs/open.c
9	sys_link	fs/namei.c
10	sys_unlink	fs/namei.c
11	sys_execve	arch/i386/kernel/process.c
12	sys_chdir	fs/open.c
...	...	...
184	sys_capget	kernel/capability.c
185	sys_capset	kernel/capability.c
186	sys_sigaltstack	arch/i386/kernel/signal.c
187	sys_sendfile	mm/filemap.c
190	sys_vfork	arch/i386/kernel/process.c

시스템 호출 (Cont'd)

Slide 19

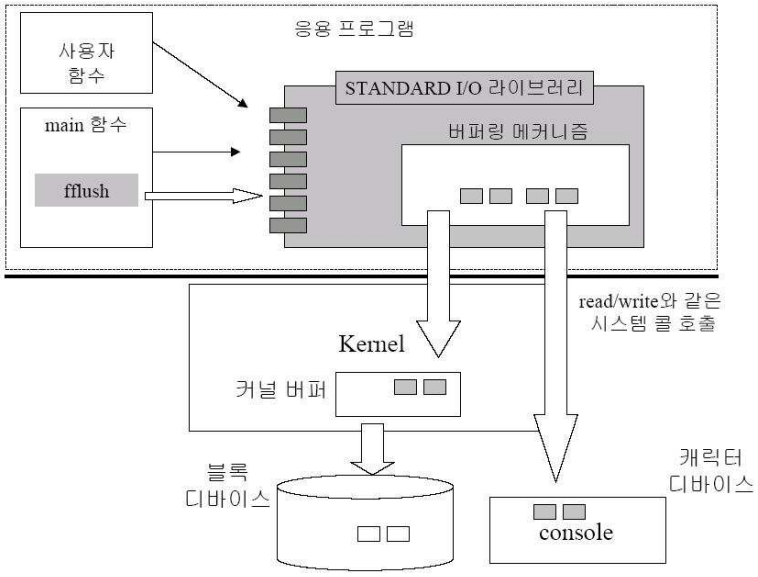
- 메모리 할당 시스템 호출 예



시스템 호출 (Cont'd)

Slide 20

- fflush 시스템 호출 예



시스템 프로그램

Slide 21

- 설명
  - 운영체제 상위에 있는 프로그램 (그림 1.1을 참조)
  - 프로그램 개발과 실행을 위해 보다 편리한 환경을 제공
- 시스템 프로그램의 종류
  - 파일관리: 파일 생성, 삭제, 복사, 이름변경, 인쇄, 덤프등 예) 윈도우의 탐색기
  - 상태정보: 날짜, 시간, 디스크공간등을 알아내는 프로그램
  - 파일편집: 파일의 내용을 생성, 수정 예) 윈도우 wordpad, Unix 의 vi
  - 프로그래밍 언어 지원: 컴파일러, 어셈블러, 인터프리터
  - 프로그램 적재와 실행: 로더 및 디버거
  - 통신: 파일전송, 원격 로그인, 메세지 전송, 전자우편
  - 가장 중요한 시스템 프로그램 ➥명령어 해석기

가상기계 (virtual machine)

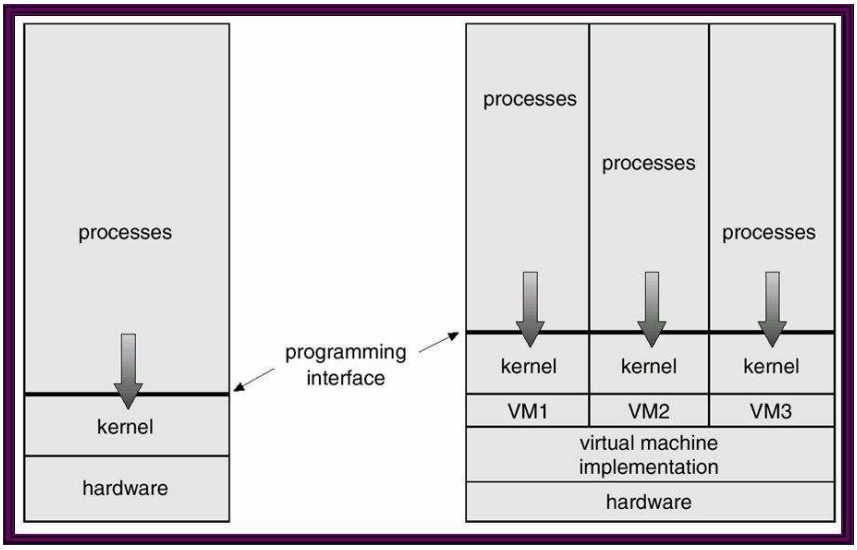
Slide 22

- 설명
  - 하드웨어 위에 하나의 새로운 가상기계층을 만듦
  - 가상 기계층 위에 여러개의 독립된 커널이 올라갈 수 있음
  - 예)
    - 가정
      - 선 워크스테이션 (Unix 시스템) 을 사용
      - 선 시스템에서 MS-DOS용 프로그램을 수행하기를 원할 경우
    - 해결
      - ➥가상 인텔 기계를 (프로그램)을 설치
      - 가상 인텔 기계상에 MS-DOS 가 설치되고
      - MS-DOS 위에서 MS-DOS 용 프로그램이 수행됨
      - 윈도우도 올릴 수 있음
  - 방법
    - 인텔용 명령어는 이와 동일한 작업을 하는 몇개의 명령으로 번역되어 실행됨

가상기계 (virtual machine) (Cont'd)

Slide 23

- 가상 기계 모델



가상기계 (virtual machine) (Cont'd)

Slide 24

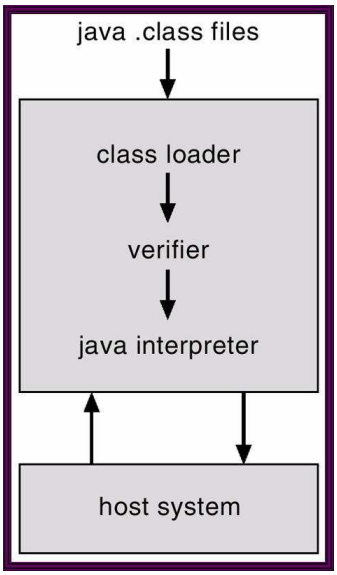
- Java
  - Sun Microsystems 사에서 개발한 언어
  - Java source code →(컴파일러) →Java bytecode →JVM 상에서 수행됨
  - JVM (Java Virtual Machine)
    - Java bytecode 를 수행하는 가상 기계
    - JVM 만 설치되어 있으면
      - 하드웨어 (인텔계열, Sparc 계열등)
      - 운영체제 (윈도우, 맥OS, Unix, Linux등)

      ➥ 보통 위의 둘을 합쳐서 플랫폼이라고 함
    - 에 상관없이 실행됨
  - JavaOS
    - JVM 이 일반 운영체제에서 수행될 경우 overhead 가 있음
    - 하드웨어에 JavaOS 를 설치후 JVM 이 설치되면 이런 overhead 를 피할 수 있음

가상기계 (virtual machine) (Cont'd)

Slide 25

- JVM 의 구성



시스템 설계 및 구현

Slide 26

- 설계 목표
  - 사용자 측면
    - 사용의 편리성, 이해의 용이성, 신뢰성, 안정성, 신속성
  - 제작자 측면
    - 설계, 구현, 유지, 보수가 쉬어야함
    - 적응성, 신뢰성, 효율성
- 구현
  - 초기에는 대부분 어셈블리어로 구현하였으나
  - 요즘에는 대부분 C 나 C++로 구현
  - 고급언어로 구현시 장단점
    - 장점: 구현 및 디버깅의 용이성, 코드 이해의 용이성, 높은 이식성
    - 단점: 속도저하, 저장공간 증가
  - 현재는 주로
    - 대부분을 고급언어로 구현 (좋은 데이터 구조가 성능에 더 중요)
    - 속도에 크게 영향을 주는 곳만 어셈블리어로 구현

시스템 설계 및 구현 (Cont'd)

Slide 27

- 설치 및 구동
  - 컴퓨터에 운영체제를 설치하고 구동시킴
  - 부팅
    - 운영체제의 커널을 주메모리에 적재하는 과정을 말함
    - 커널을 주메모리에 적재하는 프로그램
      - 부트스트랩 프로그램 혹은
      - 부트스트랩 로더 (bootstrap loader) 혹은
      - 부트 로더등으로 불림

      ➥ 윈도우에서는 BIOS 가 이 역할을 함