운영체제 소개

2nd Week

Kim, Eui-Jik

Contents

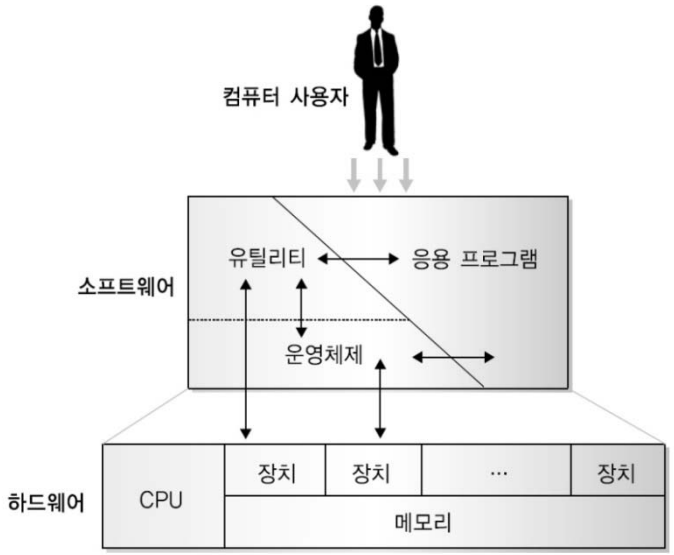
- 소개
- 운영체제란 무엇인가
- 운영체제의 유형
- 운영체제의 역사
- 응용 프로그램 기반
- 운영체제의 구성 요소와 목표
- 운영체제 아키텍처

소개

- 수십 년 동안 빠르게 진화해온 컴퓨터 분야
- 사용자 워크스테이션 BIPS (Billions of Instructions per second)의 처리 속도
- 일상 생활의 모든 영역에 컴퓨터 사용
  - 운영체제의 역할과 책임 변화
    - 문서, 게임, 음악, 비디오, 자산 관리
    - 노트북, PDA, 휴대폰, MP3

운영체제란 무엇인가

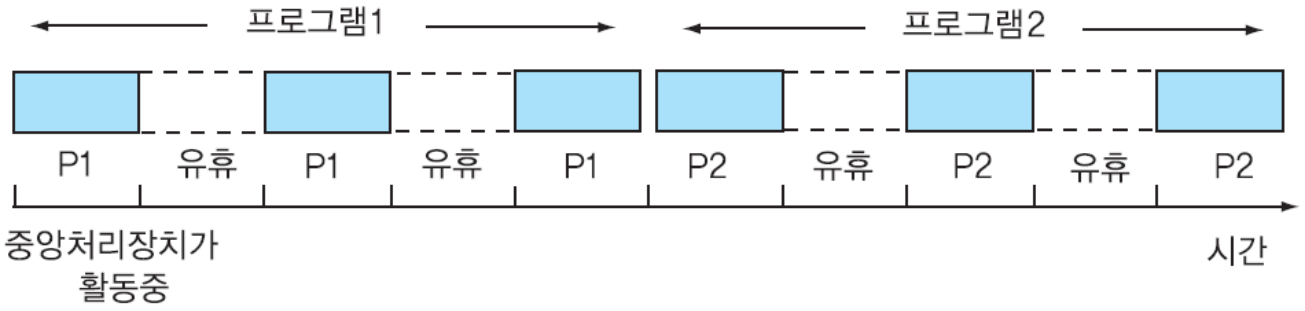
- [1960년대 개념] 운영체제는 '하드웨어를 제어하기 위한 소프트웨어'
- [현재 개념] 운영체제는 '응용 프로그램이 하드웨어와 상호 작용할 수 있게 해주는 소프트웨어'
  - 주어진 입력에 맞는 결과를 보장하도록 소프트웨어와 하드웨어 조작
  - 커널(Kernel): 운영체제의 핵심구성요소를 이루는 소프트웨어의 집합
- 운영체제는 자원 관리자
  - 하드웨어
    - 프로세서
    - 메모리
    - 입출력 장치
    - 통신 장치
  - 소프트웨어
    - 응용프로그램



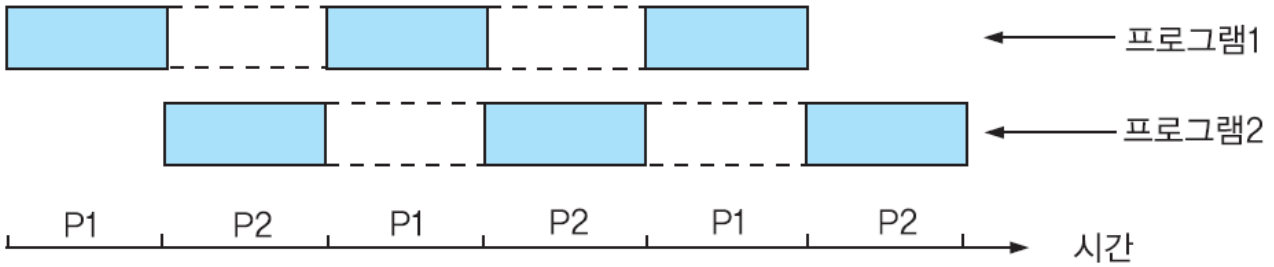
운영체제의 유형

- 일괄처리 시스템 (Batch Processing System)
  - 입력자료를 일정기간 또는 일정량을 모아서 한꺼번에 처리하는 시스템
- 다중 프로그래밍 시스템 (Multi-programming System)
  - 하나의 작업이 CPU를 사용하다 입출력 처리 등으로 CPU를 사용하지 않는 동안, 다른 작업에 CPU를 할당하여 CPU의 효율을 극대화
    - 중앙처리장치가 항상 수행되도록 하여 그 이용도를 높이기 위한 방안
    - 주기억장치 내에 여러 프로그램들이 존재하도록 함
- 시분할 시스템 (Time Sharing System)
  - CPU사용과 관련하여 일정 시간(Time Slice)을 각 작업에 할당하여 주어진 시간동안 컴퓨터와 대화식으로 프로그램을 실행하는 시스템

운영체제의 유형



(a) 순차성 실행



(b) 다중 프로그래밍에서의 실행

다중 프로그래밍 시스템

운영체제의 유형

- 실시간 시스템 (Real-time System)
  - 매우 엄격하게 정의되어 있는 시간제약 등과 같은 사건들의 제시된 상황을 분석
  - 사전에 정의된 제약 내에서 수행되어야 함

- 다중 처리 시스템 (Multi-processing System)
  - 복수의 CPU를 사용하여 다중 작업을 처리하는 시스템으로 작업 처리 속도가 빠르고 시스템의 안전성이 높으나 비용이 상승
  - 공유기억장치(common memory)를 통하여 하나로 연결된 다중처리기(multiprocessor)의 제어 및 공유를 위한 시스템

다중 처리 시스템

운영체제의 유형

- 임베디드 시스템 (Embedded System)
  - 마이크로프로세서 또는 마이크로컨트롤러를 내장하여 시스템 제작자가 의도한 몇 가지 혹은 특수한 기능만을 수행하도록 제작된 시스템
  - 임베디드 시스템과 그 한정된 자원들의 능력에 맞게 최적화 가능

초기 역사 : 1940, 1950년대

- 뚜렷한 변화를 보이며 발전한 운영체제
  - 1940년대
    - 초기 디지털 컴퓨터는 운영체제를 포함하지 않음
  - 1950년대
    - 한번에 한가지 작업만 수행
    - 단일 스트림 배치 처리 시스템 (single-stream batch-processing system)
      : 프로그램, 데이터를 순서대로 테이프나 디스크에 로드해서 일괄로 처리
    - 프로그래머가 시스템 자원을 직접 제어

1960년대

- 배치 처리 시스템
- 멀티 프로그래밍
  - 자원 공유
    - 운영 체제는 여러 작업을 동시에 처리하는 방향으로 발전
- 시분할 시스템
  - 대화식 컴퓨팅 지원, 프로그램과 데이터 공유
    - MIT의 CTSS(compatible Time-sharing System)
    - IBM의 TSS(time sharing System)
    - CTSS, CP/CMS(control program/conversational System)
    - 멀틱스(multics)

1970년대

- 멀티 모드 멀티프로그래밍 시스템
  - 배치 처리와 시분할, 실시간 응용 프로그램 모두 지원
  - 1970년 후반에는 개인 컴퓨터 혁명 시작 (by 마이크로프로세서)
- TCP/IP 통신 표준 활성화
  - 이더넷의 등장으로 근거리 통신망(LAN) 환경 발전
  - 보안 문제 등장
    - 암호 작성/해독 기술 주목
- 운영체제가 네트워크와 보안을 아우르는 수준으로 발전
  - 상업적 요구 충족할 만큼 개선

1980년대

- 개인용 컴퓨터와 워크스테이션의 시대
  - 소형 컴퓨터 중앙 처리 장치인 마이크로프로세서 기술 발전
    - 워크스테이션(고성능 데스크톱) 등장
  - 소프트웨어의 발전
    - 스프레드시트, 워드 프로세서, 데이터베이스, 그래픽 패키지
  - 그래픽 사용자 인터페이스(graphical user interface) 등장

- 네트워크 기술의 발전
  - 컴퓨터간 정보 전송 경제적이고 현실적 수준의 전환
  - 클라이언트/서버 컴퓨팅 모델의 널리 보급
    - 분산 컴퓨팅 환경 제공

인터넷과 월드 와이드 웹의 역사

- ARPA (Advanced Research Project Agency)
  - ARPAnet 구현
    - 오늘날 인터넷의 시조
    - 빠르고 쉬운 통신 능력
    - 중앙 집중적인 통제 없이 작동
      : 네트워크의 일부가 고장나도 남은 부분은 다른 경로로 전송
    - TCP/IP 프로토콜
      : 오류 없는 전송 보장
    - 상업용 목적으로 활용
      : 대역폭 증가
      : 하드웨어와 통신비용 감소
  - World Wide Web (WWW)
    - 사용자는 모든 주제에 대해 멀티미디어 기반 문서 조회
    - 1989년 CERN에서 하이퍼링크 기반 문서 공유 방법 개발 시작

1990년대

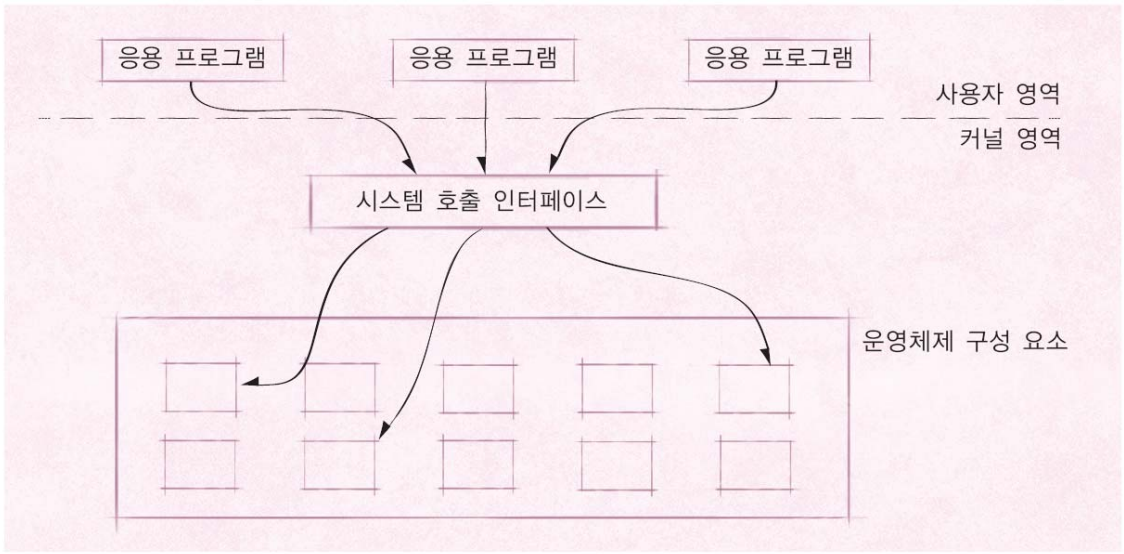
- 1990년 대 특성
  - 하드웨어 성능의 기하급수적 발전
    - 프로세싱 파워와 저장 공간의 증가
  - 월드 와이드 웹의 탄생으로 분산 컴퓨팅의 증가
    - 개인 컴퓨터간에도 분산 컴퓨팅이 일반적인 일
    - 자원의 사용 확대 및 효율 향상
    - But, 네트워크 속도가 컴퓨터의 처리 속도에 미치지 못한 한계
  - Microsoft의 성장
    - Windows 운영체제 개발
    - 사용자 친화적 인터페이스 제공
    - 데스크톱 운영체제 잠식
    - 기업용 운영체제 Window NT 출시
  - 객체 기술
    - 객체 지향 프로그램의 등장은 컴포넌트의 재사용 가능하게 하여 개발 시간 단축
    - 객체 지향 운영체제의 등장으로 기존 운영체제보다 유지보수와 확장이 용이
  - 오픈 소스 운동
    - 오픈 소스 소프트웨어를 지향하는 운동
      : 리눅스, 아파치 등

2000년 이후

- 미들웨어의 역할이 중요해짐
  - 미들웨어: 네트워크를 통해 독립적인 응용 프로그램을 서로 연결하는 소프트웨어
    - 웹 어플리케이션에서 보편적으로 사용 ex) 웹서버 ↔ 데이터베이스
- 웹 서비스
  - 분산 컴퓨팅으로 전환 촉진
- 멀티프로세스와 네트워크 아키텍처
  - 새로운 하드웨어와 소프트웨어 설계 기술 개발 기회 제공
- 고도 병렬성 (massive parallelism)
  - 프로세서를 다수 보유해 여러 독립적인 계산으로 병렬로 수행
- 운영체제 인터페이스 표준화
  - 다양한 프로그램 지원하고 사용 용이

응용 프로그램 기반

- 응용 프로그램 기반
  - 응용 프로그램 개발환경을 구성하는 운영체제와 하드웨어의 조합
  - 운영체제는 응용소프트웨어 개발자들의 메모리관리, 입출력, 통신회선 관리 등의 부담을 해소
    - API 제공으로 하드웨어 조작을 간단히 해결
  - 응용 프로그램 개발자는 특정 루틴만을 호출



[그림 1-1] 응용 프로그램과 운영체제의 상호 작용

운영체제의 구성 요소와 목표

- 운영체제의 핵심 구성 요소
  - 프로세스 스케줄러 (Process scheduler)
    - 프로세서에서 프로세스를 실행할 시점과 기간을 결정
  - 메모리 관리자 (Memory manager)
    - 프로세스에 메모리를 할당할 시점과 방식 결정
  - 입출력 관리자 (I/O manager)
    - 하드웨어 장치들과 연동해 입출력 요청을 처리
  - 프로세스 간 통신 관리자 (IPC(Inter-Process Communication) manager)
    - 프로세스들이 서로 통신 가능하게 함
  - 파일 시스템 관리자 (File system manager)
    - 저장장치의 데이터를 조직화(file), 데이터에 접근할 수 있는 인터페이스 제공

운영체제의 구성 요소와 목표

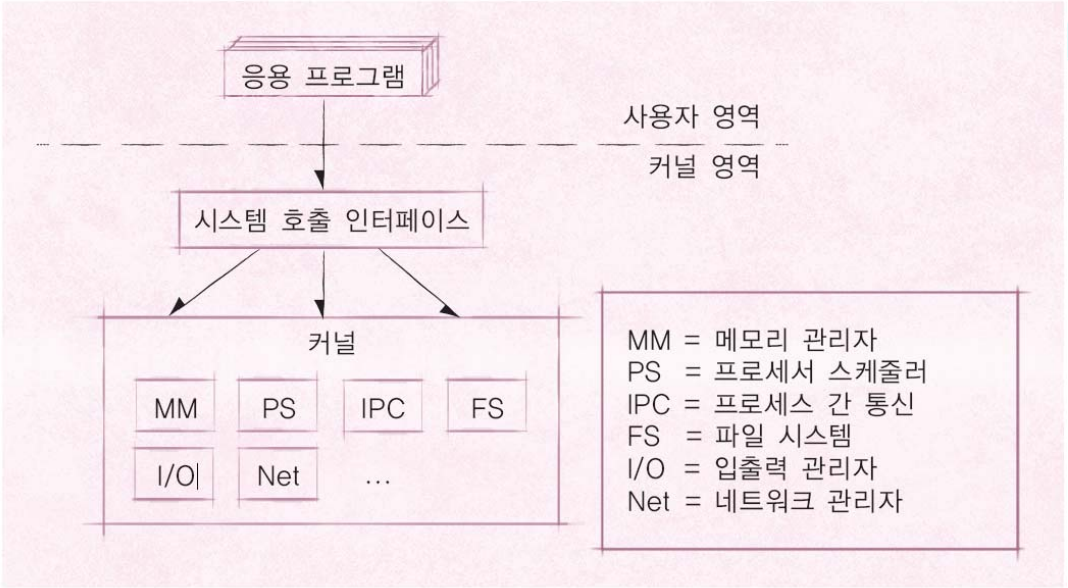
- 운영체제의 목표
  - 효율성 (efficiency)
    - 처리량이 높고, 처리시간이 짧음
  - 견고함 (robustness)
    - 장애 내구성, 신뢰성
  - 규모 확장성 (scalability)
    - 자원을 추가하면 해당 자원을 사용할 수 있음
    - 멀티 프로세서 시스템에서 중요
    - 프로세서 수에 성능이 비례하지 않음
  - 확장성 (extensibility)
    - 새로운 기술에 잘 적응
  - 이식성 (portability)
    - 다양한 하드웨어 구성에서 동작가능
  - 보안 (security)
  - 상호 작용성 (interactivity)
    - 사용자 또는 시스템에서 일어나는 이벤트에 빠르게 응답
  - 사용성 (usability)
    - 사용하기 쉬운 사용자 인터페이스 제공

운영체제 아키텍처

- 현대 운영체제는 복잡
  - 다양한 서비스 제공
  - 다양한 하드웨어와 소프트웨어 지원
  - 운영체제의 구성 요소의 실행 권한을 지정하여 복잡성 해결
    - 모놀리식 커널
    - 마이크로커널

운영체제 아키텍처

- 모놀리식 (monolithic) 아키텍처
  - 운영체제의 모든 구성 요소를 커널에 포함
  - 기능 호출만으로 다른 구성 요소와 직접 통신 가능
  - 컴퓨터 시스템에 제한 없이 접근
  - 높은 성능
  - 오류나 악성 코드에 취약



[그림 1-3] 모놀리식 운영체제 커널의 아키텍처

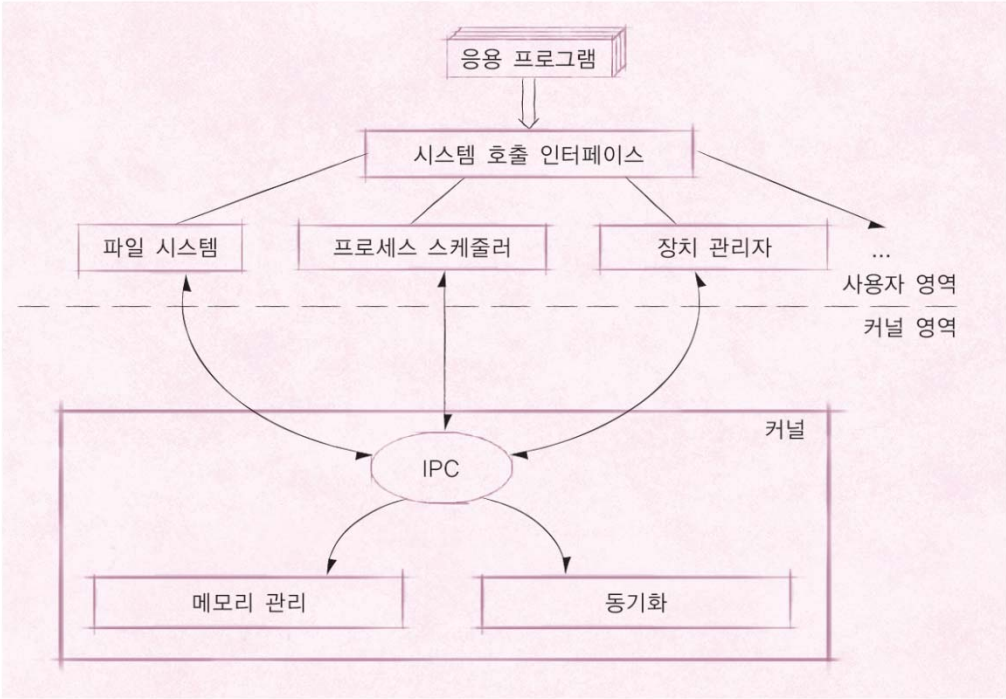
운영체제 아키텍처

- 계층적 (layered) 아키텍처
  - 유사한 기능을 수행하는 요소들을 그룹으로 묶어 계층으로 구분
  - 각 계층은 바로 상위 또는 하위 계층과 상호 작용
  - 하위계층은 구체적인 구현 숨기고 인터페이스만 제공
  - 모듈화
    - 운영체제에 구조와 일관성 부여
    - 소프트웨어 검증과 디버깅 및 수정 과정 간편

[그림 1-4] THE 운영체제의 계층

운영체제 아키텍처

- 마이크로커널 (microkernel) 아키텍처
  - 소수의 서비스만 제공
    - 커널 규모 감소, 규모 확장성 향상
  - 구성 요소를 낮은 수준의 권한으로 커널 외부에서 실행
  - 확장성, 이식성, 규모 확장성 향상
  - 모듈 간의 통신이 많아 성능 감소 우려



[그림 1-5] 마이크로커널 운영체제 아키텍처