

운영체제

Introduction

양희재 교수 (hjyang@ks.ac.kr) / 경성대학교 컴퓨터공학과

운영체제란?

- PC를 구입하면
 - Windows XP, Windows 7, Linux, MS-DOS
 - Mac OSX, iOS
- 운영체제: Operating System
- 운영체제가 없는 컴퓨터?
 - 컴퓨터: 프로세서와 메모리
 - 전원을 켜면 어떤 일이? 휘발성 메모리 - *야생마*
 - 프로그램을 실행하려면?
 - 여러 개의 프로그램을 동시에 실행시키려면?
 - 프린터에 인쇄 명령을 내리려면?
 - 하드 디스크에 저장하려면?

운영체제

- 컴퓨터 하드웨어를 잘 관리하여
 - 프로세서, 메모리, 디스크, 키보드, 마우스, 모니터, 네트워크, 스피커, 마이크, GPS, ...
- 성능을 높이고
 - Performance
- 사용자에게 편의성 제공
 - Convenience
- 컴퓨터 하드웨어를 관리하는 프로그램
 - *Control program for computer*

부팅 (Booting)

- 컴퓨터 구조

- 프로세서, 메모리 (ROM, RAM), 디스크
- POST (Power-On Self-Test)
- 부트로더 (Boot loader)

- 운영체제

- 관리(Management) 프로그램
- 프로세서, 메모리, 디스크, 입출력장치 드라이브
- 커널 (kernel) vs 명령 해석기 (shell, command interpreter)

운영체제의 위치

- 하드웨어 > 운영체제 > 애플리케이션
 - 2가지 그림
- 운영체제 vs 정부 (Government)
 - 자원 관리자 (resource manager)
 - 자원 할당자 (resource allocator)
 - 주어진 자원을 어떻게 가장 잘 활용할까? 국토, 인력, 예산
 - 정부가 직접 일하지는 않는다
 - 업무별 부서: 행정부, 교육부, 국방부, 건설교통부, 농림부, ...
 - 프로세스, 메모리, 입출력장치, 파일시스템, ...

역사

- 컴퓨터 역사: 1940년대 말~
 - 하드웨어 발전 vs 운영체제 기술 발전
 - Card reader > memory > processing > line printer
- Batch processing system (일괄처리) – resident monitor
- Multiprogramming system (다중프로그래밍)
 - 컴퓨터는 비싼 자원
 - 빠른 CPU, 느린 i/o → 메모리에 여러 개의 job
 - CPU scheduling, 메모리 관리, 보호
- Time-sharing system (시공유 시스템)
 - 강제 절환, interactive system (대화형)
 - 가상 메모리, 프로세스간 통신, 동기화

OS 기술 천이

- 컴퓨터 규모별 분류
 - Supercomputer > Mainframe > Mini > Micro
 - Supercomputer > Server > Workstation > PC > Handheld > Embedded
- 고성능컴퓨터의 OS 기술이 Handheld/Embedded 까지
 - Batch processing
 - Multiprogramming
 - *Timesharing*
- 고등 컴퓨터 구조 (Advanced Computer Architectures)
 - 고등 운영체제의 등장

고등 운영체제

- 다중 프로세서 시스템 (Multiprocessor system)
 - 병렬 시스템 (parallel system)
 - 강결합 시스템 (tightly-coupled system)
 - 3가지 장점: performance, cost, reliability
 - 다중 프로세서 운영체제 (Multiprocessor OS)
- 분산 시스템 (Distributed system)
 - 다중 컴퓨터 시스템 (multi-computer system)
 - 소결합 시스템 (loosely-coupled system)
 - 분산 운영체제 (Distributed OS)
- 실시간 시스템 (Real-time system)
 - 시간 제약: Deadline
 - 공장 자동화 (FA), 군사, 항공, 우주
 - 실시간 운영체제 (Real-time OS = RTOS)

인터럽트 기반 시스템

Interrupt-Based System

인터럽트

- 현대 운영체제는 인터럽트 기반 시스템!
- 부팅이 끝나면?
 - 운영체제는 메모리에 상주 (resident)
 - 사건 (event) 을 기다리며 대기: 키보드, 마우스, ...
- 하드웨어 인터럽트 (Hardware interrupt)
 - 인터럽트 결과 운영체제 내의 특정 코드 실행 (ISR)
 - *Interrupt Service Routine* 종료 후 다시 대기
- 소프트웨어 인터럽트 (Software interrupt)
 - 사용자 프로그램이 실행되면서 소프트웨어 인터럽트 (운영체제 서비스 이용 위해)
 - 인터럽트 결과 운영체제 내의 특정 코드 실행 (ISR)
 - ISR 종료 후 다시 사용자 프로그램으로

인터럽트 기반 운영체제

- 운영체제는 평소에는 대기 상태
 - 하드웨어 인터럽트에 의해 운영체제 코드 (ISR) 실행
 - 소프트웨어 인터럽트에 의해 "
 - 내부 인터럽트(Internal interrupt)에 의해 "
- ISR 종료되면
 - 원래의 대기상태 또는 사용자 프로그램으로 복귀
- 인터럽트 기반 운영체제
 - 그림 참조

이중모드

- 한 컴퓨터를 여러 사람이 동시에 사용하는 환경
 - 또는 한 사람이 여러 개의 프로그램을 동시에 사용
 - 한 사람의 고의/실수 프로그램이 전체 영향
 - STOP, HALT, RESET 등
- 사용자 프로그램은 STOP 등 치명적 명령 사용 불가하게!
 - 사용자 (user) 모드 vs 관리자 (supervisor) 모드
 - 이중 모드 (dual mode)
 - 관리자 모드 = 시스템 모드 = 모니터 모드 = 특권 모드
 - Supervisor, system, monitor, privileged mode
- 특권 명령 (privileged instructions)
 - STOP, HALT, RESET, SET_TIMER, SET_HW, ...

이중모드

- 이중 모드 (dual mode)
 - 레지스터에 모드를 나타내는 플래그(flag)
 - 운영체제 서비스 실행될 때는 관리자 모드
 - 사용자 프로그램 실행될 때는 사용자 모드
 - 하드웨어/소프트웨어 인터럽트 발생하면 관리자 모드
 - 운영체제 서비스가 끝나면 다시 사용자 모드
- 일반적 프로그램의 실행
 - 프로그램 적재 (on memory)
 - *User mode > (키보드, 마우스) > system mode (ISR) > user mode > (모니터, 디스크, 프린터) > system mode > user mode*
 - 그림 참조

하드웨어 보호

- 입출력장치 보호
 - Input/output device protection
- 메모리 보호
 - Memory protection
- CPU 보호
 - CPU protection

(1) 입출력장치 보호

- 사용자의 잘못된 입출력 명령
 - 다른 사용자의 입출력, 정보 등에 방해
 - 예: 프린트 혼선, 리셋 등
 - 예: 다른 사람의 파일 읽고 쓰기 (하드디스크)
- 해결법
 - *입출력 명령을 특권명령으로: IN, OUT*
 - 입출력을 하려면 운영체제에게 요청하고 (system mode 전환),
 - 운영체제가 입출력 대행; 마친 후 다시 user mode 복귀
 - 올바른 요청이 아니면 운영체제가 거부
- 사용자가 입출력 명령을 직접 내린 경우?
 - *Privileged instruction violation*

(2) 메모리 보호

- 다른 사용자 메모리 또는 운영체제 영역 메모리 접근
 - 우연히 또는 고의로
 - 다른 사용자 정보/프로그램에 대한 해킹
 - 운영체제 해킹
- 해결법
 - *MMU 를 두어 다른 메모리 영역 침범 감시하도록* (Memory Management Unit)
 - MMU 설정은 특권명령: 운영체제만 바꿀 수 있다
- 다른 사용자 또는 운영체제 영역 메모리 접근 시도?
 - *Segment violation*

(3) CPU 보호

- 한 사용자가 실수 또는 고의로 CPU 시간 독점
 - 예: while (n = 1) ...
 - 다른 사용자의 프로그램 실행 불가
- 해결법
 - *Timer 를 두어 일정 시간 경과 시 타이머 인터럽트*
 - 인터럽트 > 운영체제 > 다른 프로그램으로 강제 전환

운영체제 서비스

- 프로세스 관리
- 주기억장치 관리
- 파일 관리
- 보조기억장치 관리
- 입출력 장치 관리
- 네트워킹
- 보호
- 기타 ...

1 프로세스 관리

- Process management
- 프로세스 (process)
 - 메모리에서 실행 중인 프로그램 (program in execution)
- 주요기능
 - 프로세스의 생성, 소멸 (creation, deletion)
 - 프로세스 활동 일시 중지, 활동 재개 (suspend, resume)
 - 프로세스간 통신 (interprocess communication: IPC)
 - 프로세스간 동기화 (synchronization)
 - 교착상태 처리 (deadlock handling)

2 주기억장치 관리

- Main memory management
- 주요기능
 - 프로세스에게 메모리 공간 할당 (allocation)
 - 메모리의 어느 부분이 어느 프로세스에게 할당되었는가 추적 및 감시
 - 프로세스 종료 시 메모리 회수 (deallocation)
 - 메모리의 효과적 사용
 - 가상 메모리: 물리적 실제 메모리보다 큰 용량 갖도록

3 파일 관리

- File management
- Track/sector 로 구성된 디스크를 파일이라는 논리적 관점으로 보게
- 주요기능
 - 파일의 생성과 삭제 (file creation & deletion)
 - 디렉토리(directory)의 생성과 삭제 (또는 폴더 folder)
 - 기본동작지원: open, close, read, write, create, delete
 - Track/sector – file 간의 매핑(mapping)
 - 백업(backup)

4 보조기억장치 관리

- Secondary storage management
 - 하드 디스크, 플래시 메모리 등
- 주요기능
 - 빈 공간 관리 (free space management)
 - 저장공간 할당 (storage allocation)
 - 디스크 스케줄링 (disk scheduling)

5 입출력 장치 관리

- I/O device management
- 주요기능
 - 장치 드라이버 (Device drivers)
 - 입출력 장치의 성능향상: buffering, caching, spooling

시스템 콜

- System calls

- 운영체제 서비스를 받기 위한 호출

- 주요 시스템 콜

- **Process**: end, abort, load, execute, create, terminate, get/set attributes, wait event, signal event
- **Memory**: allocate, free
- **File**: create, delete, open, close, read, write, get/set attributes
- **Device**: request, release, read, write, get/set attributes, attach/detache devices
- **Information**: get/set time, get/set system data
- **Communication**: socket, send, receive

예제: MS-DOS

- INT 21H
- 관련 자료
 - <http://spike.scu.edu.au/~barry/interrupts.html>
- 예제: 파일 만들기 (Create file)
 - AH = 3CH, CX = file attributes, DS:DX = file name

예제: Linux

- INT 80H
- 관련 자료
 - http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html
- 예제: 파일 만들기 (Create file)
 - EAX = 8, ECX = file attributes, EBX = file name
- 시스템 콜 라이브러리 (library)
 - <http://www.digilife.be/quickreferences/qrc/linux%20system%20syscall%20quick%20reference.pdf>