운영체제 1장

2015253039 권진우

**실전 연습문제**

1.1 OS(운영체제)의 3가지 주요 목적은?

-컴퓨터 사용자가 컴퓨터 하드웨어에서 프로그램을 편리하고 효율적으로 실행할 수 있는 환경을 제공하기 위해서이다.

-문제를 해결하기 위해 필요에 따라 컴퓨터의 하드웨어 및 소프트웨어 자원을 할당을 제어하며 할당된 프로세스는 가능한 공평하고 효율적으로 할당해준다.

-컨트롤 프로그램으로써 2가지 주요 기능이 있다. 1) 사용자 프로그램의 커널 모드 권한 실행에 대한 방어(protection)와 컴퓨터의 부적절한 사용을 방지. 2) 입출력 장치들의 작동과 컨트롤을 관리한다.

1.2. 컴퓨터 하드웨어를 효율적으로 사용하기 위한 운영체제의 필요성이 강조되는데, 운영체제가 이 원칙을 버리고 자원을 '쓰레기'자원으로 만들어 버리는 것이 적절한가? 왜 이러한 시스템이 낭비만은 아닌가?

-Single-사용자 시스템은 사용자를 위한 시스템 사용을 극대화 하여야 한다. GUI는 CPU사이클을 낭비할 수 있지만, 이것이 시스템과 사용자간의 상호작용을 최적화한다. 예를 들어, 실행중인 프로그램을 CPU가 점유하고 있지만 사용자가 원하는 이벤트를 실행하기 위해서 실행중인 프로그램의 CPU사이클을 일부 버리고 사용자의 요구를 위한 작업을 수행하기도 한다.

1.4. 운영체제를 만들 때 다양한 정의를 생각하고, 운영체제가 웹 브라우저나 메일과 같은 프로그램들을 포함해야 하는지 여부를 고려한다. 운영체제가 해야 하는 것과 하지 말아야 하는 것을 확인해보고 의견을 뒷받침 해보아라.

-운영체제를 정의할 때 자주 사용하는 인기있는 프로그램을 포함시키는 것은 경우에 따라 다른것 같다고 생각합니다. 인기있는 프로그램일지라도 운영체제의 일부에 사용자가 자주 사용하는 프로그램을 넣는 것은(운영체제 내장형 프로그램) 커널 내에서 실행된다는 점에서 보안상 취약할 수 있으며 이러한 프로그램을 많이 넣었을 때는 운영체제가 용량이 무거워 질 수 있으며 이에 따라 전체적인 속도 저하가 일어날수도 있다고 생각하여서 정말 자주 사용하며 보안상 취약점이 덜 한 프로그램은 집어넣는 것에 찬성 하지만 모든 프로그램들을 집어넣는 것은 좋지 못할 수 있다고 생각합니다.

1.5. 커널모드와 사용자모드의 구분이 어떻게 protection(보호) 시스템의 초보적인 형태로서 가능한가?

-커널모드와 사용자 모드의 구별이 초보적인 형태로 보호를 해주는 이유는 특권 명령어와 같은 CPU의 사이클 제어와 같은 것은 사용자가 제어할 수 없는 OS가 제어하는 Kernel 모드에서만 명령어를 쓸 수 있도록 함으로써 사용자가 하드웨어에 직접 액세스하지 못하게 OS를 통하여 서비스를 받고 따라서 사용자의 명령어가 특권명령어를 사용할 수 없게 함(특권 명령어 사용시 프로그램 도중 종료)으로서 컴퓨터의 주요한 자원의 보호를 할 수 있습니다.

사용자 모드 : 특권 명령어 사용 불가, 커널 모드 : 특권 명령어 사용

1.6. 다음 중 특권을 부여해야 할 지침은?

a.타이머 값 설정.

b.시계 읽기

c.메모리 삭제

d.함정 명령어에 대한 지시

e.인터럽트 중단

f.장치의 상태표에 대한 수정

g.사용자로부터 커널 모드로 전환

h.입출력장치에 액세스

-a,c,e,f,h는 특권 명령어로 사용할 필요가 있다고 생각한다.

-a,c,e,f,h는 타이머값 설정, 메모리 삭제, 인터럽트 중지, 장치의 상태에 대한 수정, 입출력 장치 액세스이다.

1.10 캐시가 유용한 이유를 2가지 서술하시오. 캐시는 어떤 문제를 해결하고 그들은 어떠한 문제를 야기할 수 있는가? 캐시가 캐싱하는 장치(디스크)만큼 크게 만들 수 있는 경우, 왜 실제로는 그렇게 크게 만들고 디스크 같은 장치를 제거하지 않는가?

-캐시는 둘 이상의 저장장치가 데이터를 교환할 때 유용하며 두 저장장치 사이간 속도차이로 인한 병목현상을 줄여줄 수 있으며 속도의 버퍼를 제공한다고 할 수 있다. 고속 장치(Ex. CPU)가 캐시에서 필요로 하는 데이터를 발견하면, 더 느린 장치를 기다릴 필요가 없다. 캐시의 데이터는 구성요소의 데이터와 일치해야 하는 것이다. 만약 구성품이 변경된 데이터 값을 갖고 있고, 지난 데이터가 캐시에 있는 경우, 캐시도 변경된 데이터 값으로 변경이 되어야 한다.(변경된 데이터가 캐시에 업데이트되지 않는 아주 잠깐의 순간이 존재 가능) 이것은 특히 두 개 이상의 프로세스가 기준점에 접근하고 있는 다중 프로세서 시스템에서 문제가 되고 구성요소는 동일한 크기의 캐시에 의해 제거될 수 있지만, 다음과 같은 경우 해당된다. 구성품은 동등한 상태절약 능력을 가지고 있다. 즉, 부품이 전기를 제거할 때 데이터를 보관하는 경우, 캐시는 또한 데이터를 보관하여야 한다. 캐시가 클 경우 그 저장 용량에 따른 가격이 매우 비싸질 수 있다. 또한 캐시(cache)는 휘발성 메모리이다.

연습문제

1.19 인터럽트의 목적은 무엇이며, 트랩과 인터럽트의 차이점은 무엇인가? 트랩은 사용자 프로그램에 의해 의도적으로 발생할 수 있는가? 그렇다면 그 목적은 무엇인가?

-인터럽트의 목적은 CPU(중앙처리장치)가 연산을 함에 있어서 사이클마다 인터럽트를 확인하는데 이 목적은 Software 인터럽트, Hardware 인터럽트, exception 인터럽트가 있고 보통 인터럽트 신호를 CPU에게 보내서 기존 수행 중이던 작업을 일시적으로 스택에 저장하여 중지시키고 I/O장치의 데이터를 처리하거나 예외 처리 등 인터럽트를 처리한 후 완료가 되면 다시 원래의 수행 중이던 작업으로 되돌아와서 실행 되는 것이 인터럽트이며 인터럽트를 통해서 CPU가 한가지 작업 뿐만이 아닌 사용자가 원하는 여러가지의 작업을 수행하고 즉시 반응할 수 있도록 해준다. 트랩은 실행중인 [프로그램](http://www.terms.co.kr/program.htm) 내에 테스트를 위해 특별한 조건을 걸어 놓은 것을 말한다. 예를 들어 "인터럽트를 트랩 한다"는 것은 특정한 [인터럽트](http://www.terms.co.kr/interrupt.htm)가 발생할 때까지 기다렸다가, 그에 해당하는 [루틴](http://www.terms.co.kr/routine.htm)을 실행시키는 것을 말한다. 에러 트랩은 에러 조건을 시험하고, 복원 루틴을 제공하기 위한 것이다. [디버깅](http://www.terms.co.kr/debugging.htm) 트랩은 특정 [명령어](http://www.terms.co.kr/instruction.htm)의 실행을 기다렸다가 그 프로그램을 중지시키고, 바로 그 순간의 시스템 상태를 분석하는 것이다. 트랩은 사용자 프로그램에 의해서 의도적으로 실행 될 수 있으며, 사용자가 CPU에게도 인터럽트 신호를 보낼 수 있도록 해준다.

1.20 CPU의 부하가 증가하는 것을 피하기 위하여 직접 메모리 액세스(DMA) 방식이 고속 입출력 장치에 사용된다.

A. 전송을 조율하기 위하여 CPU는 어떻게 장치와 결합 작동하는가?

- CPU와 기억장치, I/O장치들과의 속도 차이로 인한 병목현상을 방지하기 위해서 CPU는 다른 기억장치들과의 연결 통로 사이에 버퍼를 사용하며 각 장치들 마다 컨트롤러를 두어서 예를 들어 입출력장치의 컨트롤러가 입출력에 필요한 데이터를 모두 처리하면 입출력장치 컨트롤러가 데이터를 갖고 있다가 CPU에게 인터럽트를 보내고 처리 완료되어 갖고 있던 데이터를 한번에 전송한다.

-EX) CPU - Device Controller - I/O Device

B. CPU는 메모리 연산이 종료되었음을 어떻게 알 수 있는가?

-앞서 말했듯이 각 기억장치나 외부장치들의 컨트롤러가 장치 드라이버를 통해 제어되고 장치 컨트롤러는 장치가 처리한 데이터를 갖고 있다가 CPU에 인터럽트 신호를 보내면 CPU가 신호를 신호를 받고 메모리 연산이 종료되었음을 알게 된다. 이후 ISR을 통해서 인터럽트를 처리한다.

C. DMA가 데이터를 전송하는 동안 CPU는 다른 프로그램을 실행할 수 있다. 이 프로세스는 사용자 프로그램의 실행을 방해하는가? 그렇다면 어떤 형태의 방해가 일어나는가?

-DMA는 장치의 컨트롤러가 데이터를 모두 처리한 후에 CPU에 인터럽트 신호를 보내는데 데이터의 전송 단위가 크지 않기 때문에 polling방식 보다는 CPU에 부하가 덜하지만 인터럽트 한번에 전송하는 데이터의 크기가 크지 않기 때문에 인터럽트 신호가 마찬가지로 많이 발생하게 되어서CPU가 많은 부하를 받게 된다. 이를 방지하기 위해 DMA 방식은 DMA컨트롤러가 따로 존재하여 CPU가 처리 해주어야 할 인터럽트 신호를 대신하여 CPU가 좀 더 적은 인터럽트를 받게 만들어준다. CPU가 할 일을 DMA가 일부 대신하여 줌(Device 인터럽트 신호를 대신 받고 많은 데이터를 모아서 원하는 데이터의 처리가 모두 완료되었을 때)으로써 CPU가 한번의 인터럽트 신호로 처리 완료된 데이터를 가져갈 수 있다. 사용자 프로그램 실행을 방해 여부는 아마도 CPU가 인터럽트 신호 한번에 많은 양의 데이터를 가져가기 때문에 사용자 프로그램을 실행 시 순간적으로 인터럽트 신호를 처리하기 위해 프로그램이 잠시 느려질 수 있다.