1장 과제

1731017 김민경.

1. 운영체제를 자신의 말로 간단히 정의해보라.

운영체제는 사용자에게 GUI 등의 환경을 제공해주고, 컴퓨터 자원들을 독점적으로 관리하는 시스템 소프트웨어이다. 따라서 사용자와 하드웨어 간 인터페이스의 역할을 한다. 세부적으로는 사용자는 시스템 호출과 같은 인터페이스를 통해 운영체제 영역에 접근할 수 있고, 하드웨어는 인터럽트와 같은 인터페이스를 통해 운영체제에 접근할 수 있다.

2. 운영체제가 자원을 관리하여 성취하고자 하는 목적은 무엇인가?

운영체제는 자원을 독점적으로 관리하는 소프트웨어이다.
이와 같은 배타적이고 독점적인 자원 관리는 여러 방면에서 목적이 있다.
먼저 효율성 측면에서 보자. CPU, 메모리, 입출력 장치 등 여러 자원을 유형에 따라 분류하여 관리를 하기 때문에 효율성이 향상된다. 다음은 운영체제가 관리하는 자원 관리의 종류이다. 프로세스 자원 관리에서는 작업시간을 할당하거나 우선순위를 부여하는 등 효율적으로 작업이 실행되도록 한다. 메모리 관리에서는 메모리 공간을 할당하고 회수하는 등 전체 메모리 기억 공간을 효율적으로 사용한다. 파일관리는 파일의 생성, 삭제, 변경, 유지 등을 별도로 관리한다. 입출력 장치 역시 별도의 스케줄링으로 여러가지 입출력 장치를 관리하고 제어한다.

그 다음은 편의성 측면이다. 사용자로 하여금 컴퓨터에 대한 내부 지식을 다 알지 못해도 하드웨어에 접근할 수 있게 한다.

마지막으로 운영체제와 사용자 영역간 분리는 보안적 측면에서 우수하다.
사용자가 직접 자원에 접근할수 없기 때문이다.

3. 운영체제의 기능을 5가지만 말해보라.

운영체제는 컴퓨터 사용의 편리성을 높이고 자원관리의 효율성을 높이기 위해 여러 기능들을 가지고 있다. 첫째, 프로세스/프로세서 관리 둘째, 메모리 관리, 셋째, 파일 관리 넷째, 장치 관리 다섯째. 보안.

4. 만일 운영체제가 없는 시스템에서 애플리케이션이 하드웨어 자원을 마음대로 활용하게 되면 어떤 문제가 발생할 수 있는지 간단히 설명하라.

만일 사용자 영역과 운영체제가 가지는 커널 영역이 분리되지 않고 애플리케이션이 하드웨어를 자원을 접근하려고 하면 자원이 손상될 수 있다.
예를 들어, 동시에 여러 사용자나 프로그램이 파일을 만들려고 할 때, 각 사용자들은 각 하드 디스크의 빈 곳을 찾아 저장을 해야하는데 이를 구체적으로 아는데 한계가 있어서 같은 디스크 영역에 파일을 만들게 될 경우 파일 데이터가 훼손될 수 있다. 따라서 자원을 배타적, 독점적으로 관리할 주체의 필요성이 생긴다. 사용과 영역과 커널 영역을 분리하고 커널 영역에서 직접 입출력을 제어하고 처리를 하면 자원을 더 효율적으로 관리할 수 있으며 사용자가 하드 디스크의 내부 구조를 알지 못해도 자원할당을 받을 수 있다. 또한, 엄격한 영역 구분을 통해 응용 프로그램의 에러 격리성이 보장되며. 사용자가 임의적으로 내부 자원 공간에 접근 및 수정 할 수 없기 때문에 보안적으로도 뛰어나다.

5. 다중 프로그래밍이 도입됨에 따라 컴퓨터 기술은 새롭게 극복해야 할 많은 문제들을 직면하게 되었다. 어떠한 것들이 있는지 간단히 소개하라.

기존의 배치 운영체제와 배치 처리 시스템은 한번에 하나의 프로그램을 메모리에 로딩하여 실행시키기 때문에 CPU의 노는 시간 (Idle time)이 많았다. 따라서 컴퓨터 시스템의 처리율을 높이고자 여러 프로그램을 메모리에 올려놓고 프로그램을 실행하다가 I/O가 발생하면 메모리에 로딩된 다른 프로그램을 실행시키는 방식인 다중프로그래밍 방식이 도입되었다. 여러 프로세스들을 동시에 관리해야 함에 따라 몇몇 문제들이 생겼다.
첫째, 여러 프로세스들을 동시에 메모리에 올려 놓아야 하기 때문에 배치 시스템에서 사용했던것 보다 큰 메모리가 필요하다.

둘째,

둘째, 각 프로그램을 메모리의 어느 위치에 놓을 것이며 프로그램에 할당할 메모리의 크기에 대한 기준, 몇 개의 프로그램을 로딩하는 것이 합리적인지 등의 메모리 관리가 필요하다.

셋째, 한 개의 프로그램만 메모리에 로딩하는 것이 아니다 보니 다른 프로그램에게 할당된 메모리 공간을 침범하는 경우가 있을 수 있으므로 메모리를 보호하는 방법이 필요하다.

넷째, 메모리에 로딩된 어떤 프로그램을 먼저 실행시킬지 결정하는 CPU 스케줄링과 실행 중인 프로그램의 상태 (context)를 저장하는 컨텍스트 스위칭이 중요해졌다.

다섯째, I/O 장치가 요청받은 입출력이 끝났다면 이를 알리기 위한 방법이 필요한데, 따라서 인터럽트에 대한 개념이 도입되었다.

여섯째, 여러 프로그램이 동시에 실행되면서 공평한 자원을 사용할 수 있게 되므로 자원의 상태가 바뀌어지는 등의 문제가 생긴다. 따라서 동기화에 대한 문제가 생겼다.

일곱째, 교착상태가 발생한다. 만약 프로그램 1이 자원 A를 소유한 상태에서 프로그램 그가 소유한 자원 B를 사용하고자 요청한다. 동시에 프로그램 그는 자원 B를 소유한 상태에서 자원 A의 사용을 요청하는 경우, 두 프로그램은 서로 상대가 소유한 자원을 요청하면서 무한정 대기 하게 된다.

6. 시분할 운영체제에서 시분할의 뜻은 무엇인가?

시분할이란 여러 프로세스들이 CPU를 사용하는 시간을 나누어서 사용하도록 하는 의미이다.
시분할 운영체제는 메모리에 로딩된 모든 프로그램을 1초 혹은 100ms 등 공평한 시간 간격으로 나누어 돌아가면서 CPU를 할당하여 idle time (노는시간)을 줄이는 운영체제이다. time slice는 매우 짧기 때문에 프로그램은 일을 처리하는데 크게 방해받지 않고 많은 프로그램들을 동시에 실행시킬 수 있다. 이와 같은 처리는 사용자에게 빠른 응답을 줄 수 있다는 데 장점이 있다. 다중 처리 방식에서는 프로세스를 최대한 많이 메모리에 올려 작업의 효율을 높이고자 하였지만 시분할 처리 방식에서는 응답시간을 최소화하는 데 목적이 있다.

7. 내장 프로그래밍 방식의 출현은 컴퓨터의 발전에서 어떤 변화를 가지고 왔는가?

내장 프로그래밍 방식의 출현이 가지는 가장 큰 의미는 컴퓨터를 하드웨어와 소프트웨어로 분리하여 보기 시작했다는 점이다. 이전 고정 프로그램 방식의 컴퓨터에서는 CPU와 메모리의 개념이 분명치 않았다. 하나로 뒤섞여 있었고 하드웨어와 소프트웨어의 개념이 없었다. 하지만 내장 프로그램 컴퓨터는 CPU와 메모리, 입력장치, 출력 장치라는 하드웨어의 구조를 분명히 하고, 프로그램은 입력 장치를 통해 메모리에 로딩하여 실행시킨다. 실행할 프로그램을 메모리에 담고 CPU가 프로그램을 실행시키는 방식으로 동작한다.

이와 같은 방식은 프로그램을 제어하고 수정하는데 편리성을 주게 되었다. 기존 고정 프로그램 방식에서 제어 순서를 변경하려면 장비를 다시 배선해야 하므로 비효율적이었다. 하지만 내장 프로그래밍 방식의 도입으로 프로그램 수정이 편해져 비교적 간단하게 제어할 수 있게 되었다.