안녕하세요 저는 운영체제의 역할에 대해 발표할 소프트웨어학과 안태진입니다.

우선 목차입니다. 저희는 운영체제의 정의에 대해 설명하고 이 운영체제가 왜 있는건지 운영체제의 목적을 설명하겠습니다. 그리고 운영체제의 역할에 대해 설명할겠습니다. 저희는 운영체제의 역할에 대한 큰그림을 이렇게 사용자 인터페이스, 프로세스 관리, 메모리 관리, 저장장치 관리, 보안, 보호로 잡아봤습니다. 이 부분에서 특히 프로세스 관리 부분과 메모리 관리 부분에 공을 들여 조사를 했습니다.

운영체제는 두가지 관점, 바로 사용자 관점과 시스템 관점으로 볼 수 있습니다. 먼저 사용자 관점으로는 하드웨어와 응용 소프트웨어 중간에 위치하여 사용자들이 컴퓨터를 간편하게 이용 할 수 있도록 도와주는 시스템 소프트웨어입니다. 그리고 시스템 관점에서는 컴퓨터 시스템이 보유하고 있는 자원들을 효율적으로 관리하는 시스템 소프트웨어업니다.

이제 운영체제의 목적에 대해 설명해드리겠습니다. 운영체제는 응용 프로그램들이 메모리, CPU, 입출력 장치등의 자원을 사용할 수 있도록 만들어줍니다. 운영체제는 이 들을 추상화하여 파일 시스템 등의 서비스를 제공합니다. 예를 들어 CPU는 프로세스, 메모리는 가상메모리, 저장장치는 파일로 추상화합니다.

다음으로 운영체제의 역할에 대해 소개하겠습니다.

가장 먼저 운영체제는 사용자 인터페이스를 제공합니다. 사용자 인터페이스란 사용자가 컴퓨터와 편하게 의사소통 할 목적으로 만들어진 매개체입니다. 예를 들면 GUI와 CUI가 있습니다. GUI란 Graphic User Interface의 약자로 아이콘을 선택하여 명령을 실행하는 방식입니다. 대표적으로 저희가 많이 쓰는 윈도우가 있습니다. 반대로 CUI란 Character User Interface의 약자로 사용자가 직접 명령을 입력하여 실행하는 방식입니다. 이러한 방식은 MS-DOS나 리눅스가 사용합니다.

다음은 운영체제의 역할 중에 프로세스 관리 파트를 알아보겠습니다.

프로세스란 메인 메모리에 적재되어 있는 프로그램입니다. 프로세스는 밑의 다섯 가지 상태 중 한가지 상태로 존재합니다.

운영체제는 크게 밑의 4가지 기능을 하며 프로세스를 관리합니다.

운영체제는 프로세스간 통신을 도와줍니다. 프로세스간 통신이란 프로세스간 데이터 공유와 교환을 뜻합니다. IPC 방법에는 크게 4가지 종류가 있습니다. 첫번째로 PIPE입니다. 두개의 프로세스를 연결하는 파이프를 생성합니다. 그리고 데이터를 주고받는 방식입니다. 하나의 프로세스는 데이터를 쓰기만, 다른 프로세스는 읽기만 가능합니다. 그렇기 때문에 읽기와 쓰기 둘 다하려면 두개의 파이프를 생성해야합니다. 또 다른 IPC의 방법으로는 메시지 큐가 있습니다. 메시지 큐는 데이터에 번호를 붙여 커널 영역에 저장 후 여러 개의 프로세스가 데이터를 다루는 방식입니다. 간단하게 메시지를 커널영역에 저장한뒤, 프로세스가 그 영역에 접근하여 데이터를 공유하는 방식입니다.

다음은 공유 메모리입니다. 공유 메모리를 커널에 요청해 메모리 공간을 할당 받고 데이터를 그 공간에 공유하면 프로세스들이 데이터를 공유받는 방식입니다. 이 방법은 중개자가 없어 IPC중 가장 빠르게 작동합니다. 여기서

중개자란 프로세스들이 메시지를 주고 받는데 도움을 주는 것들입니다. 대표적으로 커널이 있습니다. 소켓은 컴퓨터끼리의 통신을 도와주는 방법입니다. 즉, 원격에서 프로세스간 데이터 공유 시 사용됩니다. 대표적으로 네트워크 시스템이 소켓 방법으로 이루어져있습니다.

다음은 운영체제의 프로세스 관리중 프로세스 생성 및 제거, 동기화에대해 알아보겠습니다. 먼저 프로세스 생성에는 두가지 종류가 있습니다. 첫번째로 사용자가 직접 프로그램을 실행하는 방식입니다. 예를 들어 리눅스 터미널에서 이런 명령어를 치거나, 윈도우 명령 프롬프트에서 이런 명령어를 치는식이죠. 두번째로는 프로그램 안에서 다른 프로그램을 실행하는 방법입니다. 예를 들어 업무용 소프트웨어에서 출력 프로그램을 실행하는 방식입니다. 이때 생성된 프로세스가 자식 프로세스, 그 프로세스를 생성한 프로세스가 부모 프로세스입니다.

프로세스의 제거는 프로세스에 할당된 자원을 모두 회수하고, 디스크로 돌려보내는 작업입니다. 이 제거는 자식 프로세스가 종료를 위해 종료 상태를 부모 프로세스에게 보내고, 부모 프로세스는 이 정보를 받으면 자식 프로세스를 제거하는 방식으로 이루어집니다. 만약 자식 프로세스의 종료 상태를 부모 프로세스가 얻어가지 않는 경우에는 좀비 프로세스가 생성됩니다. 좀비 프로세스는 실행이 종료되었지만, 제거되지 않은 프로세스입니다. 이 경우를 방지하기 위해 프로세스 동기화가 필요합니다 프로세스 동기화란 자식 프로세스가 완전히 종료되었는지 부모 프로세스가 특정 함수로 계속 확인하는 과정입니다.

그리고 또 다른 올바르지 못한 제거는 자식 프로세스보다 부모 프로세스가 먼저 종료된 경우입니다. 이 경우에는 고아 프로세스가 생성됩니다. 부모 프로세스가 자식 프로세스보다 먼저 종료되었을 때의 자식 프로세스를 고아 프로세스라고 하는데, 이 프로세스는 init프로세스의 자식 프로세스로 등록됩니다. init프로세스는 말그대로 모든 프로세스의 시초, 부모 프로세스입니다. 다른 모든 프로세스를 실행시키는 프로세스입니다. 운영체제가 부팅될 때 실행됩니다.

운영체제는 프로세스의 중지와 재수행을 시키는 역할도 합니다. 운영체제가 이 역할을 하는 목적은 전체적인 부하를 감소시키고, 시스템 자원을 관리하기 위함입니다. 중지의 경우에는 3가지 경우가 있습니다. 먼저 시스템 기능에 이상이 발생한 경우입니다. 이 경우 프로세스가 중단된후, 회복된뒤 프로세스가 재수행됩니다. 다음 경우는 의심되는 부분이 있는 프로세스가 실행되었을 경우입니다. 이 경우 사용자가 실행중인 프로세스가 중지되고, 이 프로세스를 점검후 재수행 여부를 결정합니다. 마지막으로는 시스템에 부하가 걸린 경우입니다. 이 때는 몇 개의 프로세스를 중단시킨 후 시스템 정상화시 프로세스를 재수행합니다.

마지막으로 운영체제는 프로세스 스케줄링을 합니다. 준비 상태에 있는 프로세스들은 CPU할당을 받기 위해 레디큐라는 공간에서 기다립니다. 이때 어떤 프로세스에게 CPU를 먼저 할당해줄지 결정하는 것을 스케줄링이라고 하고 이 것은 CPU 스케줄러에 의해 관리됩니다. 디스패처에의해 CPU에 할당 됩니다. 프로세스에게 CPU를 할당해주는 작업을 디스패치라고 하는데, 이 작업을 수행하는 모듈이 디스패처입니다.

스케줄링은 적용 시점에 따른 구분과 우선 순위 변동 여부에 따라 구분 할 수 있습니다. 먼저 적용 시점에 따라서는 비선점형 스케줄링과 선점형 스케줄링이 있습니다. 선점이란 뜻은 남보다 앞서서 차지함이라는 뜻을 가지고 있습니다. 따라서 비선점형 스케줄링은 선점할 수 없는 스케줄링입니다. 즉, 프로세스가 CPU를 할당 받으면 프로세스가 종료되거나 I/O 이벤트로 자발적으로 중지 될 때까지 실행이 보장됩니다. 선점형 스케줄링은 반대로 프로세스가 CPU를 할당 받아 실행중이어도 프로세스를 중지시키고 CPU를 강제 점유 가능합니다. 우선 순위 변동 여부에 따라서는 정적 스케줄링과 동적 스케줄링 방식이 있습니다. 정적 스케줄링은 프로세스에 부여된 우선 순위

가 바뀌지 않는 스케줄링 방법이고, 동적 스케줄링은 프로세스의 우선 순위가 바뀔 수 있는 스케줄링 방법입니다.

다음으로는 CPU 스케줄러의 알고리즘에 대해 알아보겠습니다. 먼저 우선 순위 스케줄링이 있습니다. 시간 제한, 메모리 요구량, 프로세스의 중요성 등을 판단해서 우선순위를 정하고, 이 우선 순위가 높은 프로세스에게 CPU를 먼저 할당하는 방법입니다. 현재 프로세스보다 우선순위가 높은 프로세스가 들어왔을 때 CPU를 강제 점유할 수도 있고, 그 프로세스가 종료될 때까지 기다릴 수 있기 때문에 선점형, 비선점형 스케줄링 둘 다 될 수 있습니다. 이 알고리즘은 한 프로세스보다 우선 순위가 높은 프로세스가 계속 들어오면 그 프로세스는 실행되지 않는다는 단점이 있습니다. FCFS 스케줄링은 선입선출 스케줄링 방법으로써 CPU 요청 순서대로 할당을 합니다. CPU 요청 순서대로 할당을 하고 할당이 끝나면 다른 프로세스가 할당되니 비선점 스케줄링입니다. 단점은 CPU의 평균대기 시간이 길다는 것입니다.

다음 스케줄링 방법은 SJF 스케줄링입니다. 최단 작업 우선 스케줄링으로 말 그대로 가장 짧은 CPU할당 시간을 가진 프로세스를 먼저 할당하는 방법입니다. 이 스케줄링 방법도, 우선 순위 알고리즘과 같은 이유로 선점, 비선점 스케줄링 둘 다 될 수 있습니다. 단, CPU의 할당 시간을 예측 할 수 없기 때문에 다음 할당 시간이 이전 할당 시간과 비슷할거라고 예측하여 진행합니다. 다음은 RR 스케줄링입니다. 이 방법은 일정한 시간을 정하여 모든 프로세스에게 일정한 시간의 CPU할당 시간을 주는 방식입니다. 예를 들어, 3ms초라고 시간을 정했으면, 모든 프로세스에게 3ms의 CPU할당 시간을 주는 겁니다. 이 방법은 일정한 시간이 지나면 다른 프로세스에게 CPU를 할당하기 때문에 선점 스케줄링입니다. 이 시간을 얼마나 적절하게 주느냐에 따라 알고리즘의 성능이 좌우됩니다. 다음은 SRT 스케줄링입니다. 최단 잔여시간 스케줄링으로 진행 중인 프로세스가 있어도 최단 잔여시간 프로세스에게 우선권을 부여하는 방법입니다. 진행 중인 프로세스가 있어도 다른 프로세스에게 우선권을 부여하는 방법이기 때문에 선점형 스케줄링입니다.

다음은 운영체제의 메모리 관리에 대해서 알아보겠습니다.

메모리 관리란 프로세스들을 위해 메모리를 할당, 제거, 보호하는 활동입니다. 이러한 메모리 관리 기법에는 3가지가 있습니다. 첫번째로 반입 전략은 메인 메모리에 적재할 다음 프로세스의 반입 시기를 결정하는 전략입니다. 즉, 언제 적재할 것인가?에대한 것이죠. 이 반입전략에는 두가지 요구 반입 전략, 예상 반입 전략이 있습니다. 요구 반입 전략은 CPU의 요구가 있을 때마다 메인 메모리에 적재하는 방식입니다. 반대로 예상 반입 전략은 요구될 가능성이 큰 데이터와 프로그램을 운영체제가 스스로 예상하여 미리 반입하는 방식입니다. 예상 반입 전략은 요구 반입 전략보다 시간은 덜 들지만, 공간 낭비 가능성이 있습니다.

두번째 전략으로는 배치 전략이 있습니다. 배치 전략은 반입한 프로세스를 메인 메모리의 어느 위치에 저장할지 결정하는 전략입니다. 이 배치 전략에도 대표적으로 3가지 방법이 있습니다. 첫째 최초 적합입니다. 최초 적합은 메인 메모리 공간을 순차적으로 탐색하다 제일 먼저 프로세스가 적재될 곳이 있다면 그 위치에 적재하는 방법입니다. 두번째 최적 적합은 메인 메모리 공간을 탐색하고, 제일 적절하게 들어갈 수 있는 공간에 프로세스를 적재하는 방법입니다. 마지막 최악 적합은 프로세스보다 큰 공간 중 가장 큰 공간에 프로그램을 할당하는 방법입니다. 마지막 전략으로는 교체 전략이 있습니다. 메인 메모리에 적재하다보면 메모리의 모든 영역이 사용중인 상태가 발생하게 됩니다. 이때 어느 영역을 삭제하고 교체할 것인지를 결정하는 전략이 교체 전략입니다.

다음은 메모리 할당 방법에 대해 알아보겠습니다. 메모리 할당은 연속 메모리 할당과 불연속 메모리 할당으로 나뉩니다. 연속 메모리 할당은 말 그대로 프로그램을 메모리에 쪼개지 않고 올려 놓고 관리하는 방법입니다. 반대

로 불연속 메모리 할당은 프로그램을 특정단위로 쪼개서 메모리에 할당하는 방법입니다.

연속 메모리 할당은 단일 프로그래밍과 다중 프로그래밍으로 나뉩니다. 단일 프로그래밍은 한 개의 프로그램만 메인메모리에 적재하여 실행하는 방법입니다. 하지만 이 방법은 프로그램의 크기가 작으면 사용자 영역이 낭비됩니다. 다중 프로그래밍은 단일 프로그래밍과 다르게 여러 개의 프로그램을 동시에 메인 메모리에 적재하여 실행하는 방식입니다. 이 다중 프로그래밍은 고정 분할과 동적 분할로 나뉩니다. 고정 분할은 메인 메모리의 사용자 영역을 여러 개의 고정 크기로 분할하는 방식입니다. 이 방식은 내부 단편화가 발생할 수 있습니다. 반대로 동적 분할 방식은 프로그램을 메인 메모리에 적재할 때 필요한 크기로 영역을 분할 합니다. 이 경우는 외부 단편화가 발생할 수 있습니다.

내부 단편화란 프로세스가 필요한 양보다 더 큰 메모리 공간이 할당되있는 경우입니다. 프로세스에서 사용하는 메모리 공간이 낭비됩니다. 외부 단편화란 메모리 할당, 해제 과정에서 작은 메모리들이 존재 하게 됩니다. 이런 경우에 총 메모리 공간은 충분하지만 실제로 할당할 수 없는 상황이 발생하게되는데 이때를 외부 단편화라고 합니다.

다음은 불연속 메모리 할당에 대해 알아보겠습니다. 불연속 메모리 할당의 고정 분할 방식인 페이징에 대해 알아보겠습니다. 가상 메모리를 일정한 크기로 나눈 것을 페이지, 물리 메모리를 일정한 크기로 나눈 것을 프레임이라고 합니다.. 즉, 페이징이란 가상메모리를 모두 같은 크기의 블록으로 분할하여 운용하는 방법입니다. 이때 하나의 프로세스는 하나의 페이지 정보를 저장하고 있는 페이지 테이블을 가집니다. 즉, 가상 메모리 상에 연속되어 있는 프로세스를 실제로는 물리 메모리상에서 불연속으로 나눠 운용하는 방식입니다. 이 페이징은 프로세스의 크기가 페이지 크기의 배수가 아닌경우 내부 단편화가 발생할 수 있습니다.

다음은 불연속 메모리 할당의 동적 분할 세그멘테이션에 대해 알아보겠습니다. 세그멘테이션은 임의로 일정한 크기로 가상메모리를 나누는게 아닌, 논리적 단위, 즉 의미상의 세그먼트로 나눠 메모리를 할당하는 방법입니다. 페이징 방법과 다르게 세그먼트들은 크기가 서로 달라 미리 분할해 두는 것이 불가능합니다. 세그먼테이션은 메모리에 적재될 때 빈 공간을 찾아 할당하는 사용자 관점의 가상 메모리 관리 기법입니다. 이 세그먼테이션은 프로세스들이 모두 같은 크기로 잘리지 않는 경우에 외부 단편화가 발생할 수 있습니다.

다음은 운영체제의 저장장치 관리에 대해 알아보겠습니다.

파일 시스템이란 컴퓨터에서 파일을 기록하고 사용하는 모든 작업의 조직적인 체계를 말합니다. 우리가 컴퓨터를 할 때 항상 만지는 파일이 모두 운영체제가 제공하는 것입니다. 이러한 파일 시스템의 기능으로는 파일에 대한 접근 제어 방법 제공, 파일의 생성 변경 삭제 관리, 파일의 무결성과 보안 유지 방안 제공, 데이터의 백업 및 복구 기능 제공, 데이터의 효율적인 저장과 관리가 있습니다.

다음은 입출력 시스템입니다. 컴퓨터에 연결된 다양한 입출력 하드웨어 장치들과 소통하는 시스템을 말합니다. 이것 역시 운영체제가 관리합니다. 입출력 시스템의 관리 기능은 첫번째로 버퍼링, 캐싱, 스풀링의 기능을 제공합니다. 버퍼링이란 컴프시간에도 배운 버퍼를 사용하는 것입니다. 운영체제가 버퍼에 데이터를 임시적으로 모았다가 사용하는 것입니다. 캐싱은 성능을 위해 자주 사용할 것 같은 데이터의 일부분을 빠른 저장장치인 캐시에 저장하는 것입니다. 스풀링은 프린터 같은 한번에 하나의 작업을 실행하는 장치를 위해 다른 작업들을 스풀에 저장하는 것을 뜻합니다. 그리고 운영체제는 입출력 장치들에 대한 장치 드라이버 인터페이스와, 특정 하드웨어 장치

에 대한 드라이버도 운영체제가 제공합니다. 드라이버란 특정 하드웨어나 장치를 제어하는 프로그램을 일컫습니다.

다음으로 운영체제는 대용량 저장장치도 관리합니다. 대표적으로 보조 저장장치가 있습니다. 보조 저장장치에는 메인 메모리를 초과하는 데이터를 저장하고, 오랜 시간 보관되어야 하는 데이터가 저장됩니다. 주로 디스크 장치 즉, HDD, SSD가 사용됩니다. 운영체제는 대용량 저장장치의 자유 공간을 관리하는데 자유 공간은 간단히 빈 공간을 뜻합니다. 그리고 저장 공간을 할당합니다. 마지막으로 흩어져 있는 데이터들을 효율적으로 엑세스 하기위한 디스크 스케줄링을 합니다.

다음은 운영체제의 보안 기능에 대해 알아보겠습니다.

운영체제는 적절한 보호 시스템 뿐만 아니라, 시스템이 동작하는 외부 환경에 대해서도 고려합니다. 운영체제의 보안은 외부 보안, 내부 보안, 사용자 인터페이스 보안의 세가지로 구별됩니다. 외부 보안은 외부의 침입자나 천재지변으로부터 컴퓨터 시스템을 보호하는 방법입니다. 내부 보안은 불법 침입자로부터 데이터를 보호하기위한 접근 제어 코드를 내장하는 방식으로 이루어집니다. 접근 제어 코드란 누군가가 무언가를 사용하는 것을 허가하거나 거부하는 기능을 제공하는 코드입니다. 그리고 사용자 인터페이스 보안은 사용자의 신원을 운영체제가 먼저 확인 후 시스템의 데이터 자료들을 접근할 수 있게하는 방법입니다.

다음은 운영체제의 보호 기능에 대해 알아보겠습니다.

우선 운영체제는 프로세서, 기억장치, 파일, 기타 자원에 대한 적절한 접근 권한을 부여하면서 시스템 보호를 수행합니다. 프로세스는 하나의 보호 영역을 갖고, 이 영역에서만 동작합니다. 접근 권한이란 어떤 프로세스가 객체에 대한 조작을 수행할 수 있는 능력을 말합니다. 접근 제어의 목적은 컴퓨터의 자원들에 허가받능 사람만 접근하기 위함입니다.

운영체제는 접근 행렬을 통해 시스템 보호를 수행합니다. 접근 행렬이란 자원에 대한 접근 권한을 행렬로 표시한 것입니다. 행에는 주체들 즉, 사용자와 프로세스가 들어가고, 열에는 객체들 즉, 파일, 메모리, 프린터 등의 자원이 들어갑니다. 사용자는 자신의 객체애 대한 다른 사용자의 접근 권한을 부여할 수도 있고, 취소할 수도 있습니다.

운영체제는 접근 행렬을 여러가지 형태로 관리를 합니다. 그중 가장 단순한 형태인 전역 테이블이 있습니다. 이 전역 테이블은 영역, 객체, 권한 집합들의 집합으로 구성되어 있는 테이블입니다.

다음은 접근 제어 리스트입니다. 객체를 기준으로 주체들을 나타낸 리스트라고 보시면 될 것 같습니다. 파일에 접근 할 수 있는 사용자 리스트를 나타내고, 각 영역은 사용자명, 사용자 그룹명, 권한의 쌍으로 이루어져있습니다.

권한 리스트는 주체를 기준으로 객체들을 나타낸 리스트입니다. 접근 제어 행렬에 있는 각 행을 해당 영역과 결합한 것입니다. 각 영역에 대한 권한 리스트는 그 자원에 허용된 조작 리스트로 구성되어있습니다.