1주차 - 운영체제 기초 (운영체제와 컴퓨터, 메모리, 프로세스의 상태)

날짜	@2025년 7월 2일
분류	운영체제

아이스브레이킹

1. 오늘은 무엇을 했나요?

ex. 어디 카페에 놀러갔어요, 어디에 가서 맛있는 거 먹었어요 등...

- 벡덩민 지각

2. 이번 주차 내용은 어땠나요?

스터디를 하면서, 앞으로 조정해야 될 내용이 있다면 이야기해봐요.

ex. 난이도 조정, 분량 조정, 방식 조정, 시간 조정 등 편하게 이야기해주세요.

- 질문 작성하는 마감일 정하기: 화요일 21:00
- 질문에 대한 답은 스터디 전까지 달아오기
- 미리 답을 달아서 꼬리 질문도 만들 수 있게 하기!
- 질문 최소량: 3개
- 말하는 연습할 질문을 정해서 각자 돌아가며 말해보기!

소은 질문

Q. OS가 무엇인지 설명해주실 수 있나요?

▼ A. 이소은

운영체제는 다른 프로그램이 유용한 작업을 할 수 있는 환경을 제공합니다. 컴퓨터 시스템은 하드웨어, 운영체제, 응용프로그램, 사용자로 구분되는데, 운영체제는 여기서 응용프로그램 간의 하드웨어 사용을 제어하고 조정하는 역할을 합니다.

▼ A1. 박주형

운영체제는 컴퓨터 하드웨어와 사용자 사이에서 중재자 역할을 하는 시스템 소프트웨어이다. 운영체제가 하는 일은 크게 4가지가 있는데 첫 번째는 'CPU 스케줄링과 프로세스 관리', 두 번째는 '메모리 관리', 세 번째는 '디스크 파일 관리', 네 번째는 'I/O 디바이스 관리'가 있다.

Q. 커널이 무엇인지 설명하실 수 있나요?

▼ A1. 이름

00입니다. 답변 안에 질문을 더 작성해도 됩니다.

▼ A1. 박주형

커널은 운영체제의 핵심 모듈로, 하드웨어 자원과 직접적으로 통신하면서, 응용 프로그램이 시스템 자원을 사용할 수 있도록 도와주는 역할을 한다. 커널은 시스템 호출을 처리하고 프로세스 및 스레드를 관리하며 메모리 및 파일 시스템을 관리한다. 커널의 종류로는 모놀리식 커널과 마이크로 커널이 있다.

▼ A2. 백동민

커널이란 운영체제가 동작하기 위한 프로그램 모음으로, 운영체제가 하드웨어를 컨트롤하기 위해 사용됩니다.

사용자 모드에서는 직접 커널을 건들 수 없고, 시스템 콜을 통해 커널 모드로 변경 후 접근 할 수 있습니다.

▼ Q. 메모리의 구조를 영역에 따라서 설명해줄 수 있나요?

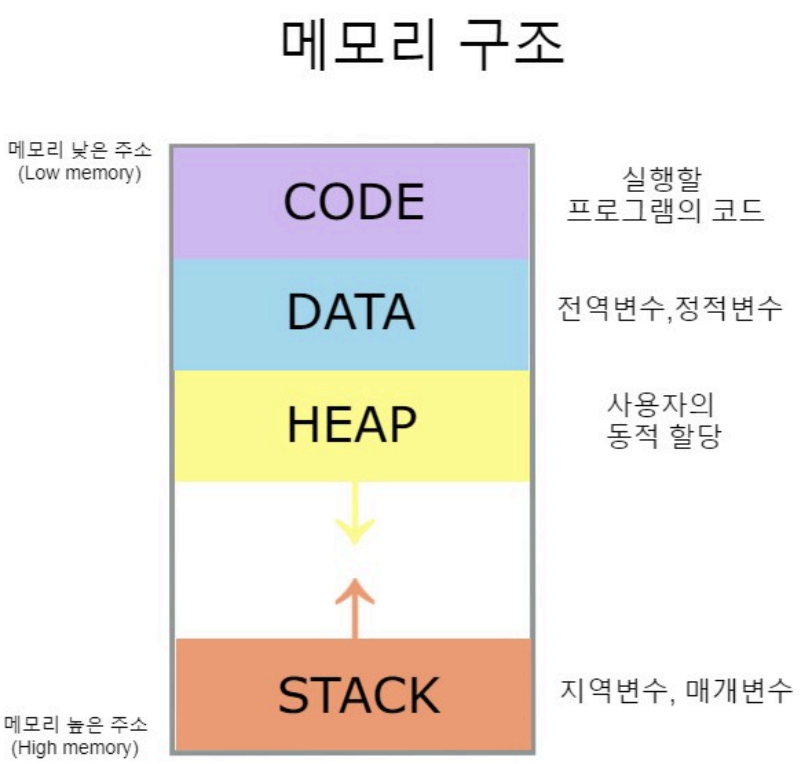
▼ A1. 박주형

하나의 프로세스는 메모리 공간을 4, 5개 영역으로 나누어 사용한다. 첫 번째는 실행할 기계어 명령이 저장된 공간인 '코드'이고, 두 번째는 전역 변수, static 변수를 저

장하는 '데이터', 세 번째는 초기화되지 않은 전역/정적 변수를 저장하는 'BSS', 네 번째는 힙(동적으로 할당되는 메모리 영역)과 스택(함수 호출, 지역 변수 저장)하는 부분이 있다.

▼ A2. 백동민

메모리 구조를 보통 4개의 영역으로 설명합니다.



1. 코드 : 코드는 리터럴과 상수등 읽기 전용으로 작성된 일종의 텍스트라고 볼 수 있습니다. 소스코드가 메모리에 할당 되면 제일 먼저 로드 됩니다.
2. 데이터 : 데이터는 소스코드가 실행 시 메모리에 로드가 됩니다. 그래서 코드 내 전역, static 변수가 저장되는 영역입니다.
3. 힙 : 런타임 중 개발자가 동적으로 할당한 메모리 영역입니다. 사용자가 allocation / free 할 수 있으며, 사용자가 메모리를 직접 관리한다고 볼 수 있습니다.
4. 스택 : 함수 호출 시 생성되는 지역 변수, 매개 변수, 리턴 주소가 저장됩니다. 보통의 함수 스택 팝을 통해 함수 해제가 동작하며, 런타임 시 메모리를 할당 받습니다. 또한 스택 영역에 저장된 포인터 변수가 힙 영역의 주소를 가르키는 관계입니다?

▼ A. 이소은

메모리 영역은 코드, 데이터, 힙, 그리고 스택 영역으로 나눌 수 있습니다. 그리고 메모리 공간 할당 방식에 따라서 크게 두가지로도 분류할 수 있습니다. 우선 코드와 데이터 영역은 정적 메모리 영역에 속합니다. 코드 영역은 실행할 프로그램 코드가 저장되는 영역입니다. 데이터 영역은 해당 프로그램의 전역변수 및 정적 변수가 저장되는 영역으로, 컴파일 시점에 크기가 결정되는 영역입니다. 다음으로 힙과 스택은 모두 동적 메모리 영역에 속합니다. 힙 영역은 사용자에 의해 동적으로 할당되는 공간으로, 런타임 시점에 크기가 결정됩니다. 특히 자바에서는 가비지컬렉터가 자동으로 해제합니다. 스택 영역은 함수 호출과 관련된 지역 변수와 매개 변수가 저장되는 영역이고, 함수 호출 시 할당되며 함수의 호출이 종료되면 소멸합니다.

Q. 메모리의 힙 영역과 스택 영역의 차이에 대해 설명해주세요.

▼ A1. 박주형

스택은 함수 호출 시 자동으로 할당되는 공간으로, 지역 변수와 매개 변수가 저장된다. 빠르지만 크기가 작고, 컴파일 타임에 크기가 결정된다.
힙은 동적 메모리 할당에 사용되는 공간으로, malloc, new 등을 통해 직접 할당 및 해제해야 한다. 유연하지만 상대적으로 느린것이 특징이다.

▼ A2. 백동민

앞선 메모리 계층을 설명 드리면서 설명했었는데, 서로 런타임 시간에 겹치게 메모리를 사용하지만.

힙 영역은 사용자가 직접 메모리를 접근합니다. 따라서 사용자가 유연하게 관리 할 수 있지만, 메모리 누수의 위험이 있습니다.

스택 영역은 함수 호출에 의해 지역 변수와 매개 변수가 저장되는 영역입니다. 함스가 스택에서 pop시 자동으로 해제됩니다.

▼ A. 이소은

스택과 힙은 같은 공간을 공유하지만, 스택이 높은 주소부터 할당되고, 힙은 낮은 주소부터 할당됩니다. 힙은 런타임에 크기가 결정되고, 스택은 컴파일 타임에 크기가 결정된다는 차이가 있습니다. 그러다가 각 영역이 침범하게 되면 힙/스택 오버플로우가 발생합니다.

Q. 메모리의 힙 영역을 너무 크게 잡으면 어떤 일이 생길까요?

▼ A1. 박주형

메모리 부족으로 인해 프로그램이 비정상적으로 종료되거나, 시스템 전체 성능 저하가 발생할 수 있다. 특히 힙이 스택 영역과 충돌하면 치명적인 충돌 오류가 생길 수 있다.

▼ A2. 백동민

당연히 함수가 그만큼 스택 영역이 줄어들기 때문에, 함수 스택에 쌓여 메모리 부족한 경우, 당연히 개발자 의도에 맞지 않게 error를 띄울 수 있습니다. 그 뿐만 아니라, 스택 영역이 힙 영역을 침범할 수 있습니다.

▼ A. 이소은

Java를 기준으로, JVM 가비지컬렉션의 수행시간이 너무 오래 걸리기 때문입니다. ? GC 잘 몰라서 어떻게 써야할지 모르겠다 ~~~~~

Q. 컨텍스트 스위칭에 대해 설명하고, 왜 컨텍스트 스위칭이 필요한지 설명해주세요.

▼ A1. 박주형

컨텍스트 스위칭은 CPU가 현재 실행 중인 프로세스의 상태를 저장하고, 다른 프로세스틔 상태를 복원하는 작업이다. 예를 들어, A 프로세스를 멈추고 B 프로세스를 실행해야 할 때 CPU의 레지스터, PC, 스택 포인터 등을 저장 및 복원한다. 컨텍스트 스위칭이 필요한 이유는 멀티태스킹을 구현하고 시스템 자원을 효율적으로 분배하기 위함이다.

▼ A2. 백동민

컨택스트 스위칭이란 CPU는 한 프로세스를 실행할 수 밖에 없기 때문에, 기존에 있는 (ㄱ) 프로세스를 잠시 PCB에 저장하고, (ㄴ) 프로세스를 실행하고 다시 A 프로세스를 실행함을 말합니다.

CPU는 기본적으로 1개의 cpu가 1개의 프로세스를 처리해야하고, 선점의 문제 시간에 따른 분할로 인해 필요합니다.

당연하게도 시스템 콜으로 인해 동작 해야한다면, 기존 것을 멈추고 처리를 해야하기 때문입니다.

▼ A3. 이소은

context switching을 통해 이전 프로세스의 실행을 중단하고, 다른 프로세스의 실행을 시작하기 위해 CPU 상태를 저장하고 불러올 수 있습니다. 운영체제를 통해서 여러개의 프로세스를 동시에 실행하는 것처럼 보이게 만들어야 하는데 이를 멀티태스킹이라고 합

니다. 멀티태스킹을 통해서 CPU하나로 매우 빠르게 여러 프로세스 사이를 전환하면서 실행하게 되는데 이 과정이 context switching입니다.

▼ Q. PCB에 대해 더 자세히 설명해주세요.

▼ A. 소은

프로세스 제어 블록은 프로세스를 관리하기 위한 정보를 포함하는 커널 자료구조입니다. PCB는 프로세스 상태, 프로그램 카운터, CPU 레지스터 정보 등을 담고 있습니다. 프로그램이 프로세스로 변환되려면 PCB가 필요합니다. 저장장치에 저장된 정적 상태인 프로그램이 PCB를 통해서 메모리에 올라온 동적 상태인 프로세스가 됩니다.

▼ A2. 박주형

PCB는 운영체제가 프로세스의 상태를 저장하는 데이터 구조이다. 각 프로세스마다 하나씩 존재한다. PCB에는 프로세스 ID, 상태, 프로그램 카운터, CPU 레지스터 상태, 메모리 관리 정보 등이 포함된다.

▼ A3. 백동민

프로세스 컨트롤 블록의 줄임말로써, 운영체제 내에서 프로세스의 정보를 저장하는 자료구조입니다. 프로세스 상태 (idle... 등등), 프로그램 카운터, 레지스터 정보 등등을 가지고 있습니다.

PCB를 통해 context switching을 구현합니다.

🌱 주형 질문

Q. 유저 모드와 커널 모드는 무엇이며, 왜 나뉘어 있나요?

▼ A1. 박주형

사용자의 입장에서 보았을 때 두 모드는 권한을 기준으로 나눈 것이다. 먼저 유저 모드는 사용자 프로그램이 실행되는 제한된 모드이고, 커널 모드는 OS 핵심 기능에 접근할 수 있는 모드이다. 두 모드의 전환을 관리하는 것은 시스템 콜이라는 인터페이스이며, 시스템 콜이 modebit 값을 관리하며 두 모드의 전환을 관리한다. 이렇게 나누는 이유는 시스템 자원을 보호하고 보안성을 확보하기 위함이다.

▼ A. 이소은

우선 유저 모드와 커널 모드로 나눈 이유는, 운영체제가 사용자가 컴퓨터 시스템의 요소를 공유하기 때문에 잘못된 프로그램으로 인해 운영체제 자체가 잘못 실행될 수 없도록 보장해야 합니다. 즉, 운영체제와 시스템 자원을 보호하기 위해 나뉘어 있습니다. 유저모드는 쉽게 말해서 제한된 모드입니다. 악영향을 끼칠 수 있는 특권 명령을 수행하려고 하면, 운영체제는 이를 불법적인 명령으로 간주해서 실행하지 않습니다. 커널모드에서만 이러한 특권 명령을 사용할 수 있습니다.

▼ A3. 백동민

유저모드란, 사용자 프로그램이 실행 시 동작하는 모드이고, 커널에 접근 불가능합니다.

하드웨어 컨트롤에 접근하기 위해서 커널을 사용해야 하므로, 커널 모드로 들어가게 됩니다.

사용자가 함부로 커널을 접근하게 되면 시스템 자원을 잘못 사용할 가능성이 있고 큰 보안적인 위험이 있습니다. 따라서 그 두 개를 구분해서 OS가 수월하게 관리 할 수 있도록 나눠났습니다. (private와 public의 차이랄까요)

Q. 시스템 콜이란 무엇이며, 어떤 역할을 하나요?

▼ A1. 박주형

시스템 콜은 유저 프로그램이 커널 기능을 요청할 수 있게 해주는 인터페이스이다. 예를 들어, 파일 열기, 메모리 할당, I/O 요청 등의 작업은 직접 하드웨어에 접근하지 않고, 시스템 콜을 통해 커널에 위임한다. 시스템 콜은 트랩을 발생시켜 유저 모드에서 커널 모드로 진입하게 된다.

▼ A3. 백동민

커널은 이제 하드웨어에 직접 접근해서 동작 시키는 프로그램인데, 이는 응용프로그램이 직접 접근 불가능합니다. 따라서, OS 내부에서 원하는 커널을 사용해서 원하는 동작을 시키기 위해서는 커널 모드 내에서 시스템 콜을 통해 동작시킵니다. 프로세스가 하드웨어에 접근하는 인터페이스라고 말할 수 있습니다.

▼ A. 이소은

시스템 콜은 응용 프로그램이 운영체제의 커널 기능을 요청하는 방법입니다. 유저레벨의 프로그램은 유저모드의 함수들 만으로는 많은 기능을 구현하기 힘들기 때문에 커널의 도움이 필요하고, 커널모드로 전환해야 해당 작업을 수행할 권한이 생기기 때문에 시스템 콜이 필요합니다.

Q. 운영체제에서 인터럽트는 무엇이고, 왜 사용하나요?

▼ A1. 박주형

인터럽트는 CPU가 작업을 수행하던 중, 외부 또는 내부의 이벤트에 의해 실행 흐름이 변경되는 것을 말한다. 대표적으로 I/O 장치에 의한 하드웨어 인터럽트와 0으로 숫자를 나누는 등의 소프트웨어 인터럽트가 존재한다. 이 소프트웨어 인터럽트를 '트랩'이라고 부르기도 한다.

▼ A2. 백동민

인터럽트란, CPU에 자극을 주어 새로운 이벤트가 일어 났음을 알리는 알람이라고 볼 수 있습니다. 인터럽트에 의해 스케쥴러의 큐에 들어가게 되고 그에 따라 context switching이 발생할 수 있습니다. 특별하게 소프트웨어 단의 인터럽트는 '트랩'이라고 부릅니다.

▼ A. 이소은

인터럽트란, 현재 실행중인 작업을 멈추고 운영체제가 급한 작업을 처리하도록 CPU에게 신호를 보내는 것입니다. 인터럽트는 크게 내부 인터럽트와 외부 인터럽트 두 종류가 있습니다. 외부 인터럽트는 주로 입출력 장치에 의해 발생되고, 내부 인터럽트는 하드웨어 고장 또는 명령어 오류로 발생합니다.

Q. 캐시 메모리와 지역성(Locality)에 대해 설명해주세요.

▼ A1. 박주형

캐시는 속도 차이를 줄이기 위해 사용되는 임시 메모리이다. 대표적으로 CPU와 메모리는 속도 차이가 너무 커서 병목현상이 발생하는데 두 계층 사이의 속도 차이를 줄이기 위해 별도의 캐싱 계층이 존재한다. 계층이 아닌 별도로 지정도 가능한데 이는 지역성(Locality)를 기반으로 한다. 지역성은 해당 데이터 코드가 최근에 얼마나 사용되었는지를 나타내는 개념이다.

▼ A2. 백동민

캐시 메모리란, 임시적으로 저장하는 메모리로 컴퓨터 성능을 위해 사용하며, 다시 사용하거나, 자주 사용하는 데이터를 저장합니다. 지역성은 시간적, 공간적 지역성으로 나눌 수 있고 핵심은 얼마나 빠르게, 얼마나 가까운 위치에서 접근 가능하는 지를 말합니다. 얻는 advantage로 빠른 접근이 가능하므로 성능이 상승합니다.

▼ A. 이소은

지역성이란, 기억 장치에 접근하는 패턴이 메모리의 특정 영역에 집중되는 성질을 의미합니다. 지역성에는 세 가지가 있는데, 공간의 지역성, 시간의 지역성, 순차적 지역성이 있습니다. 공간의 지역성이란, 현재 위치에서 가까운 데이터에 접근할 확률이 먼 거리에 있는 데이터에 접근할 확률보다 높음을 의미하는 것입니다. 시간의 지역성이란, 현재를 기준으로 가장 가까운 시간에 접근한 데이터가 더 먼 시간에 접근한 데이터보다 사용될 확률이 높음을 의미합니다. 순차적 지역성은 작업이 순서대로 진행되는 것을 의미합니다. 지역성의 예시로 캐시가 있습니다.

캐시는 지역성 이론을 사용하는 대표적인 장치로, 자주 사용되는 데이터나 명령어를 메인 메모리보다 가까운 곳에 저장해서 접근 속도를 향상시켜줄 수 있는 장치입니다. 시간 지역성을 통해 최근에 접근한 데이터나 명령어를 캐시에 저장해서 다시 사용할 때 빠르게 접근할 수 있습니다. 공간 지역성을 통해 인접한 데이터까지 미리 블록 단위로 캐시에 로딩해서 다음 접근을 빠르게 해줄 수 있습니다. 순차적 지역성을 통해 순차적으로 실행되는 코드를 예측하여 다음 명령어를 미리 로딩할 수 있습니다.

Q. 가상 메모리(Virtual Memory)는 무엇이고, 왜 필요한가요?

▼ A1. 박주형

가상 메모리는 프로세스마다 실제 메모리처럼 보이는 추상적 메모리 공간을 나타내는 기술이다. 실제 메모리보다 더 큰 주소 공간을 사용하는 것처럼 동작할 수 있다. 가상 메모리를 통해 프로그램은 독립적인 주소 공간을 사용 가능하고, 메모리 보호 및 분할 실행이 가능하다. 가상 주소와 실제 메모리 주소 간의 변환은 MMU와 페이지 테이블을 통해 이루어진다.

▼ A2. 백동민

가상 메모리는 페이징과 세그멘테이션 기법을 활용해서 프로그램이 필요한 메모리 공간보다 더 충분하게 제공하는 방법입니다.

물리적인 메모리 공간 확보가 힘들어도, 동시에 프로그램이 동작하고, 각자의 독립적인 메모리 공간을 갖게 만듭니다.

▼ A. 이소은

가상 메모리는 물리적 메모리 용량 자체가 부족해지는 문제를 해결하기 위해 사용합니다. 가상 메모리를 통해 컴퓨터는 물리적 메모리보다 더 많음 메모리를 사용할 수 있도록 가상 공간을 제공받을 수 있고, 프로그램이 실행될 때 필요한 메모리보다 더 큰 메모리를 할당할 수 있어 물리적 메모리의 제한을 극복할 수 있습니다.

또한 가상메모리는 프로세스를 격리하고 메모리를 보호할 수 있습니다.

Q. 페이지 폴트(Page Fault)와 스와핑(Swapping)이란 무엇인가요?

▼ A1. 박주형

페이지 폴트는 가상 메모리 주소에 대응되는 페이지가 실제 메모리에 없는 경우에 발생하는 현상이다. 트랩을 통해 OS가 개입하게 되며, OS는 HDD에서 필요한 데이터를 불러오고, 스와핑을 통해 메모리 프레임에 교체를 한다. 자주 발생하면 시스템 성능이 급격히 저하되며, 심하면 스레싱이 발생할 수 있다.

▼ A2. 백동민

가상 메모리가 페이징을 통해 페이지 파일을 접근을 했는데 접근을 못하면 page fault 입니다.

스와핑은 프로세스나 여러 페이지를 디스크 영역로 밀어내고 메모리를 확보하는 기법입니다.

▼ A. 이소은

페이지 폴트는 프로세스가 접근하려는 페이지가 현재 메인 메모리에 없어서 디스크에서 가져와야 하는 것입니다.

디스크에서 메모리를 swap-in, swap-out하는 것을 swapping이라고 합니다. swapping은 일반적으로 메모리가 초과 사용되어 가용 공간을 확보해야 할 때 쓰입니다.

🌱 동민 질문

Q. 메모리 계층에서 레지스터와 하드 디스크의 차이점을 속도와 용량의 관점에서 비교해주세요.

▼ A1. 박주형

레지스터는 CPU 내부에 위치한 가장 빠른 메모리이며 용량이 매우 작다. 반면에 하드디스크는 가장 느린 저장 장치이지만, 용량이 매우 크다. 레지스터는 휘발성이며, 하드디스크는 비활성 메모리이다.

▼ A. 이소은

레지스터는 CPU 내부에 위치하여 매우 빠르고 용량이 작고, 비쌉니다.. 하드디스크는 보조저장장치로, 매우 느리고 용량은 크고, 저렴한 편입니다. 따라서 컴퓨터의 메모리는 속도와 비용을 모두 고려해 계층적으로 구성되어있습니다.

▼ A2. 백동민

레지스터는 CPU 내에서 빠른 연산을 동작하는 메모리 이며, 속도는 제일 빠르고 용량은 제일 작습니다. 휘발성입니다.

하드 디스크는 처리 속도가 제일 느리고 용량을 제일 큽니다. 비휘발성입니다.

Q. 캐시 매핑은 몇가지 분류로 나눌 수 있고, 간단하게 설명해주세요.

▼ A1. 박주형

3가지가 있다. 첫 번째는 직접 매핑으로 하나의 메모리 블록이 하나의 캐시 위치에만 매핑되는 것을 말한다. 두 번째는 연관 매핑으로, 순서를 일치시키지 않고 관련 있는 캐시와 메모리를 매핑하는 것을 말한다. 마지막 집합 연관 매핑은 캐시를 여러 집합으로 나누고, 각 집합에대해 연관 매핑을 적용하는 것을 말한다.

▼ A. 이소은

아니 박주형 왤케 빨라;;;;

캐시 매핑은 세 가지로 분류할 수 있습니다. 우선 직접 매핑은, 메모리 블록이 캐시의 한 위치에만 저장되는 것입니다. 연관 매핑은 어느 캐시 라인에도 자유롭게 저장되는 것을 말합니다. 집합 연관 매핑은, 메모리 블록이 특정 set에 속하고 그 set내 여러개의 캐시 라인 중 하나에 저장이 가능한 것을 말합니다.

▼ A3. 백동민

1. 직접 매핑 : 1 메모리블록 - 1 캐시 라인 하나 하나씩
2. 연관 매핑 : 주소 상 관련 있는 캐시와 메모리끼리 매핑하기 (어떤 메모리든 관련만 있음 ㅇㅋ)
3. 집합 연관 매핑 : 주소 집합 단위로 쪼개고 그 집합 부분으로 매핑

Q. 멀티 프로세싱과 멀티 쓰레딩의 차이점은 무엇인가요.

▼ A1. 박주형

멀티 프로세싱은 여러 CPU가 여러 프로세스를 병렬로 처리하는 것이고, 멀티 스레딩은 하나의 프로세스 내에서 여러 스레드가 자원을 공유하며 동시에 실행되는 것을 말한다.

▼ A. 이소은

멀티 프로세싱과 멀티 스레딩은 모두 애플리케이션에 대한 처리방식인데, 멀티 프로세싱은 하나의 응용 프로그램에 대해 동시에 여러개의 프로세스를 실행할 수 있도록 하는 기술을 말합니다. 멀티 스레딩은 하나의 프로세스 안에 여러개의 스레드가 있는 것을 말합니다. 근데 차이점을 콕 집어서 말해야하는디...

윈도우나 리눅스도 멀티 프로세싱을 지원하지만 기본적으로 멀티 스레딩을 하고 있습니다. 왜냐하면 멀티스레딩의 장점이 있기 때문입니다. 먼저 스레드는 프로세스보다 가벼워서 프로세스보다 생성과 제거가 빠릅니다. 멀티 스레드는 하나의 프로세스 내에서 여러개의 스레드를 생성하기 때문에 힙 영역과 같은 공유 메모리에 대해 스레드 간 자원 공유를 하는 데 용이합니다. 또한 상대적으로 프로세스 컨텍스트 스위칭의 오버헤드보다 오버헤드가 적어집니다.

근데 멀티스렝딩의 단점도 있는데 ㄱ그걸 말하라고 하면 어뜨캄

▼ A3. 백동민

멀티프로세싱은 코어에서 각 작업을 독립된 프로세스로 실행해 메모리를 완전히 분리하므로 안전하지만, 프로세스 생성,문맥 전환 오버헤드가 커서 무겁습니다.

멀티스레딩은 하나의 프로세스 내부에서 쓰레드로 쪼개고 그 쓰레드들이 메모리를 공유해 문맥 전환이 가볍고 데이터 전달이 빠르지만, 데드락 등 동기화 문제가 발생하기 쉽습니다.

Q. CPU 스케쥴링에서, 기아 상태를 방지하기 위해서 사용하는 방법은 무엇이 있나요?

▼ A1. 박주형

대표적으로 Aging이 있다. 오랫동안 대기한 프로세스의 우선순위를 점점 높여 결국 실행되도록 보장하는 것으로, 이를 통해 특정 프로세스가 기아 상태가 되는 것을 방지한다.

▼ A3. 백동민

1. Aging : 계속해서 오래 대기한 프로세스에 가산점 부여. (N수생 우선 선발제도)
2. 라운드 로빈 방식 : 동등하게 모든 프로세스가 나눠 가진다.

3. 랜덤 : 무작위로 돌린다.

▼ A. 이소은

에이징을 통해서 양보할 수 있는 상한선을 지정하면 기아 상태를 방지할 수 있습니다.

참고링크

https://hoons-dev.tistory.com/95
