4장의 1번은 멀티스레딩을 지원하는 프로세스 모델을 그림으로 그리고 ! 이거는 앞에서 봤던 여러개의 스택이 포함된 오른쪽예제를 그리믕로 그리면 되고 각 dythehmf을 설명하면 되는데 다른거는 어렵지 않은데 tcb와 pcb를 잘 구분하엿 ㅓㄹ명해줘야한다. 그 안에 있는 정보가 어떻게 갈려서 들어갔는지 tcb에 있는 정보와 pcb에 있는 정보를구분할 수 있어야한다.

멀티스레딩을 지원하는 시스템의 장점 하지 않는거랑 비교했을 때

5가지 상태에 대한 그림! 을 그릴 수 있다.

멀티스레드를 구현하는 3가지 방법은 커널레벨 스레드를 사용하는거랑 유저레벨 스레드를 사용하는거랑 컴바인드 어프로치를사용하는 방법 이렇게 있음 그림만 그리면 안됨! 이해한 내용을 다 말로 설명할 수 있어야한다. 정확히 세가지 스레드가 무슨 역할을 하는지 말로 설명할 수 있어야함

커널레벨 스레드의 장점과 단점은 멀티스레딩을 사용하는 시스템에서 커널레벨 스레드를 갖고 있나 유저레벨 스레드를 갖고 있나를 비교해씅ㄹ 때 장점과 단점을 보는거 !

6번 부터는 이제 설명해야되는 부분 !!!!! 지금부터 하는 내용들이 분량은 작지만 훨신 문제가 될만한것들이 많음

3장에서 프로세스를 기반으로 하는시스템이 무엇인지 어떤식으로 프로세스가관리가 되는지 설명! 이론적인 이야기 프로세스는이렇게 관리한다~ 이런거 실제로 제일 마지막의 유닉스 시스템을 예로 보면 이론적으로 배운내용과 다름 여기서도 마찬가지!지금까지 4장에서 어떻게 멀티스레드를 구현하는지 이야기를ㄷ ㅡㄹ었는데 실제로 시스템을 만들라하면 각각의 시스템의 특색에 따라 맞춰서 만들어감

3가지 시스템 설명 1. 윈도우 2. 솔라리스 3. 리눅스 세가지가 소개가 됐다는거는 다 기본설계 방식이 방향이 다르다는거임! 완전히 다름! 설명을 들으면서 뭐가 다른지를 알아야함

윈도우 프로세스는 오브젝트 오리엔티드 디자인 프로세스가 하나이상의 스레드를 포함한다. 멀티스레딩을 포함하는 시스템이니까 3. 프로세스 뿐만 아니라 스레드도 싱그로나이제이션을 할 수 있는 기능을 가짐 뭔가 하면 동기화툴을 가지고 있다는거 그이야기는 간단히 이야기하면 a랑 비 프로세스가 있으면 에이가 비보다 먼저 작업해야되는데 비한테 나 긑날때까지 시작하지마라고 얘기하는거 얘기를 하는 툴들 내가 먼저 작업을 끝내기 전까지 너는 작업하지마 하는 툴 이 툴들은 중요! 프로세스간에 스레드간에 동기화 툴을 가진다.

프로세스 오브젝트는 어떤 프로펄티가 있냐? 를 보면 프로세스는 자원을 관리하는 단위 아래쪽에 보면 익셉션 타임 아이오 카운터등 이 프로세스가 시피유를 얼만큼 차지했는지 아이오작업얼만큼했는지 등 자원을 얼만큼 사용했는지에 대해 보여주는 프로펄티들 자원을 관리하고 사용하고 이런거는 프로세스 단위로 관리됨 스레드 오브젝트는 스레드 컨텍스트가 중요 ! ! 프로그램을 실행하다가 중단되면 그때의 시피유의 레지스터 값들을 말한다. 실행하는게 스레드 단위로 실행되니까 컨ㅌ텍스트는 스레드 컨텍스트다

다이나믹 프라이어리티 베이스 프라이어리티 스레드 프로세서 어피니티 라는 세가지 정보가 주용해보임 우선순위가 있다는 이야기는 얘가 스케쥴링 한다는거 레디큐에 줄서는거 시피유가 이거를 뽑아가는데 우선순위에 따라서 우선순위가 높으면 먼저 뽑아가고 아니면 나중에 뽑아감

다이나믹 프라이어리티는 우선순위임 실행하다가 os 마음에 들게 실행하면 먼저 뽑고 아니면 우선순위를 낮춘다.

베이스 프라이어리티 라는거는 프로세스에도 있음 스레드에도 있고 근데 왜 프로세스가 베이스 프라이어리티를 가지고 있을까 이 값은 우선순위를 정하는데 각각의 스레드가 실행하는데 실행하면서 오에스에 잘보이면 우선순위가 올라가고 아니면 내려감 실행하면서 우선순위가 변함

프로세스의 종류에 따라 윈도우에서는 os가 별도의 프로세스로 실행됨 프로세스 기반으로 실행됨 시스템 안에 프로세스 중에 유저 프로세스도 있고 시스템 프로세스도 있음 os 프로세스 중에는 우선순위가 높아서 제일 먼저 작업해야되는 애가 있고 같은 시스템 프로세스지만 천ㄴ천히 작업해도 되는애가 있음 베이스 프라이어리티는 유저냐 시스템프로세스냐 그리거 어떤 작업을 하는 프로세스냐에 따라 기본값을 정해줌 베이스 프라이어리티에서 시작해서 그 안의 스레드가 어떻게 행동하냐에 따라서 우선순위 정해짐

스레드 프로세서 어피니티라는 개념도 있음 기본적으로 윈도우 오에스는 시피유 하나라고 가정 안함 멀티 프로세서 smp 처럼 여러 개의 프로세서 가진다고 생각 10개의 시피유가 있는데 어떤 스레드가 3번 시피유에서 실행을 했음 실행하다가 타임아웃되서 레디큐로 돌아옴 다시 실행하려고 하는데 3번도 놀고 있고 5번도 놀고 7번도 놀고 이러고 있는 경우 어떤 시피유에 갈지를 생각하면 당연히 3번 시피유에 가야한다. 실행속도가 달라짐!!! 캐쉬의 개념을 생각한다면 시피유 옆에 캐쉬라는 속도가 빠른 작은 메모리가 있는데 프로그램 실행하면 자주사용하는명령어나 데이터를 거기에 넣어놓음 그래서 3번 시피유에서 작업하다 중단하면 주요한 명령들과 데이터들이 다 들어있으니까 굳이 다른 곳에서 하게 되면 그 캐슁된 데이터들을 다 다시 가져와야한다. 데이터랑 명령들을 메모리에서 다 가져와야됨!

메모리에서 가져오는 속도가 시간이 걸리니까 처음에 시작할 때 속도가 느릴수밖에 없다 3번에서 계속 작업하면 추가적인 캐슁없이 아주 빠르게 작업할 수 있다. 스레드 프로세서 어피니티는 3번에서 했으면 나는 3번에서 계속 하고 싶다 이런 어디에서 작업하고 싶은지를 나타내는거 실행하는 단위니까 스레드 프로세서 어피니티가 된다.

윈도우스레드의 상태 여기에서는 6개의 상태를 가지고 있음 6개의 상태를 가지고 있는 그림에서는 new 라는 상태가 빠지고 여기에서 2가지 상태가 추가로 더 붙은 거라고 할 수 있다. 위쪽 회색에 3개읫 ㅏㅇ태가 있고 아래쪽 회색에 세개의 상태가 있다 위쪽은 실행가능한상태 아래쪽은 실행가능하지 않은 상태를 나타낸다.

Ready 상태는 새로 스레드 만들어지면 레디큐로 들어감 레디큐에 줄 서있음 시피유가 이중 하나를 뽑아서 시랳ㅇ해야하는데 그냥 딱 뽑아서 실행하는게 아니라 1번스레드를 실행하고 있으면 다음에 실행시킬애를 미리 뽑아서 스탠바이 상태로 해놓음 스탠바이에 있는애들은 다음에 실행할 애들 자기가 1순위면 스탠바이로 가서 스위칭 되고 러닝됨 레디에서 곧장 러니응로 가지 않고 스탠바이를 거쳐간다.

러닝을 하고 있다가 세가지 가능한 길이 있는데 1. 프리앰티드 타이머 인터럽트 걸려서 타임아웃돼서 프리엠티드 되어서 레디 상태로감 2. 스레드가 마지막 문장까지 다 실행을 완료 그래서 리턴함 그러면 터미네이티드 상태가됨 3. 웨이팅이라는 상태 대기를 한다는뜻 우리가 아는 블락드 상태랑 비슷함 똑같지는 않음 왜냐면 블락과 서스펜드 두가지 적혀있음 화살표에! 블락은 입출입 할 때 실행 못할 때 그때 웨이팅 상태로감 서스펜드는 우리가 3장에서 배운 서스펜드를 생각했을 때 하드디스크로 쫓겨나는 상태 라고 생각했음 이유가 여러가지인데 1.공ㄱ간 2. Os가 이상하다 생각해서 3. 패런트 프로세스가 촤일드 프로세스 서스펜드 시키는 경우 이런 경우에 일단 일시 중지를 하는데 프로세스 안에 사실 스택 하나 밖에 없음 프로세스가 멈추면 그냥 멈춤 그래서 당연히 하드디스크로 보냄

지금은 상황이 다른게 프로세스 안에 스레드가 여러 개임 1번 스레드가 실행을 하는데 뭔가 이상한 작업을 한느거 같으면 오에스가 이걸 멈추게 함 하지만 2번 3번 스레드는 살아있음 다른 스레드가 또 다른 스레드를 멈추게 할 수 도 있음 1번쓰레드가 2번스레드를 멈추게 할 수 있음 하짐나 나머지가 다 살아있으면 하드디스크로 쫓아낼 수 없음 서스펜드는 여기에서 일시중지 상태라고 생각하는 것이 좋음 프로세스의 모든 스레드가 다 서스펜드가 되면 그때 하드디스크로 쫓겨남 스레드 한명의 상태로 봤을 때는 스레드가 일시중지된 상태라고 생각하면 된다.

그래서 스레드 입장에서 블락이 되어서 중지가 됐던 서스펜드가 되어서 중지가 됐던 둘다 웨이팅 상태로 표현함 웨이팅 상태에 있다가 나중에 블락이 해제 되고 딱 가니까 가는 길이 두가지가 있음 여기서 말하는 리소스는 한쪽은 리소스가 어베일러블 하다하고 한쪽은 리소스가 어베일러블 하지 않다고 말함 블락 또는 서스펜드여서 웨이팅 상태에 있었는데 이거는 스레드 상태 ! 프로세스가 메모리에서 하드디스크로 옮겨지면 이거는 프로세스 단위로 이루어짐 스레드읫 ㅏㅇ태가 아님 프로세스의 모든 스레드가 블락 아니면 서스펜드에 있으면 모두 웨이팅 상태에 있다는 거를 판단하고 하드디스크로 옮김 그런데 io 가 끝남 그러면 이때 os를 봤더니 해당하는 스레드를 포함하는 프로세스가 하드디스크에 가있는거 리소스가 어베일러블한거는 메모리에 있는거고 어베일러블 하지 않은거는 하드디스크로 쫓겨났다는거! 하드디스크로 쫓겨나 있는 상태면 트랜지션 상태로 보냄! 프로세스가 메모리로 올라오면 레디상태로 바꿔줌 웨이팅 하고 있는데 여전히 메모리에 있는 상태였으면 그냥 레디상태로 가면 됨 !

윈도우스 시스템이 나왔을 때 얘네가 자랑한거 중에 하나가 멀티스레딩이랑 마이크로 커널임 모든게 object 기반임 왼쪽은 유닉스 커널과 비슷한 모양 얘네를 레이어드 커널이라고 함 층층히 레이어드 된 형태로 연결해서 설계함 그러다 보니까 이 안에서 함수들이 서로를 호출하고 포인터를 가지고 데이터 구조를 사용하니까 다른 레이어드에 있는 데이터를 포인터로 사용하게 됨 하나를 수정하려고 하면 관련된 함수랑 데이터구조들을 다 수정해야함 그래서 업그레이드 하는게 큰 문제 오른쪽은 막대기 하나하나가 독립적임 모든 기능들을 하나의 오브젝트로 만들어서 다른 오브젝트와 연결고리 최소화 통ㅇ신은 전체 커널 중에 꼭 필요한 최소한의 것들만 마이크로 커널로 만들어서 거기에 넣고 메모리에 올려놓고 나머지 기능들은 그때그때 불러서 씀 프로세스 기반으로 함 프로세스 기반 프로세스 기반으로 오에스가 실행된다 모든 기능들이 마이크로 커널과 연결된 부분만 수정하면 되니까 업그레이드가 쉽다.

솔라리스 시스템은 멀티스레딩을 지원하는 대표적인 시스템이라고 보면 됨 4장의 대부분의 예들이 거의 그대로 적용된다 생각 큰 특징 하나 가짐 윈도우랑 다른 !!! 구분짓는 가장 큰 차이점은 윈도우 시스템에서는 커널레벨 스레드만 씀! 솔라리스 시스템은 컴바인드된 어프로치 사용 그래서 유저레벨 스레드도 사용가능하고 커널레벨 스레드도 사용 가능 스레드 라이브러리에서 유저레벨에서의 스레드 관리를 해주고 또 커널에서는 커널레벨 스레드 관리를 따로 함 솔라리스와 윈도우의 가장 큰 차이점! 기본은 프로세스가 있음 그 안에 여러 개의 쓰레드드릉ㄹ 가지고 있는데 유저레벨의 스레드도 있고 커널레벨 스레드도 있고 라이트 웨이트 스레드도 있음 세가지를 섞어 씀!!!

간단히 생각해서 유제레벨에서는 유저레벨스레드라는 이름으로 관리가됨 ! 유저레벨 스레드 라이브러리에서 10개의 스레드를 만들었으면 10개의 스레드가 20개를 만들으면 20개의 스레드가 유저레벨 스레드 라이브러리에서 관리됨 실제로는 이 시스템에서 한 프로세스에서 5개 이상의 커널 스레드를 지원 못한다하면 유저레벨은 스레드는 10개지만 실제로 사용할 수 있는거는 5개 밖에 없음 매핑을 시켜야됨 10개중에 실행가능한애는 5개니까 1~5 는 커널레벨 스레드를 만들고 나머지는 쉬자 서스펜드 시키자 함 실제로 5개만 커널레벨스레드를 만들고 싶으면 커널에 요청하는 명령이 있음 커널에 스레드를 만들어달라는 요청은 딱 5번만 읽어지는데 5개의 라이트웨이트 프로세스가 생김!

그러면 유저레벨 스레드 라이브러리에서 10개의 유저레벨 스레드가 있꼬 여기중 5개만 라이트웨이트 프로세스에 매핑 되어있음 커널레벨로 가보면 이 프로세스는 5개의 라이트웨이트 프로세스를 가지고 있음 그림으로 보면 솔라리스가 유닉스에서 시작된 시스템인데 유닉스 시스템의 최신 버전이라고 보면 됨 오른쪽에 보이는거처럼 프로세스 안에 여러 개의 라이트웨이트 프로세스들이 쭉 있음 필요한 만큼 만들어져있음 여기에서 보면 라이트 웨이트 프로세스가 우리가 본 스레드라고 볼 수 있음 커널 레벨의 스레드를 라이트 웨이트 프로세스라고 부름 이 시스템에서는!!! 그럼 커널레벨 스레드는 뭐냐 커널스레드는 뭐냐 이거는 매핑 되지 않은 커널 스레드 ! 그러니까 이 시스템에서는 우리가 컴바인드된 어프로치를ㄹ 사용하는 이유는 유저레벨 스레드 라이브러리를 굳이 갖는 이유는 시스템 전체에서 사용할 수 있는 스레드 숫자가 제한이 되어있고 한 프로세스당 만들어 사용할 수 있는 스레드 숫자가 제한 되어있어서 컴바인드된 어프로치를 사용!

45분 20초부터 들으셈