프로그래머의 블로그

커널(kernel)

프로그래밍/프로그래밍 공부 2016.02.17 11:10

위키백과

kernel
①낟알 ②핵심 ③커널
(과실의) 인(仁); (콩 등의) 꼬투리·겉껍질 속의 종자[열매]; (밀 등의) 낟알.

중심부, 중핵; 핵심; 요점, 요지

컴퓨터 과학에서 커널(kernel)은 운영 체제의 핵심 부분으로서, 운영 체제의 다른 부분 및 응용 프로그램 수행에 필요한 여러 가지 서비스를 제공한다. 핵심(核心)[1]이라고도 한다.

커널의 역할

커널은 운영 체제의 핵심 부분이므로, 커널의 역할 역시 운영 체제의 핵심 역할이라 할 수 있다.

보안

커널은 컴퓨터 하드웨어와 프로세스의 보안을 책임진다.

자원 관리

한정된 시스템 자원을 효율적으로 관리하여 프로그램의 실행을 원활하게 한다. 특히 프로세스에 처리기를 할당하는 것을 스케줄링이라 한다.

추상화

같은 종류의 부품에 대해 다양한 하드웨어를 설계할 수 있기 때문에 하드웨어에 직접 접근하는 것은 문제를 매우 복잡하게 만들 수 있다. 일반적으로 커널은 운영 체제의 복잡한 내부를 감추고 깔끔하고 일관성 있는 인터페이스를 하드웨어에 제공하기 위해 몇 가지 하드웨어 추상화(같은 종류의 장비에 대한 공통 명령어의 집합)들로 구현된다. 이 하드웨어 추상화는 프로그래머가 여러 장비에서 작동하는 프로그램을 개발하는 것을 돕는다. 하드웨어 추상화 계층(HAL)은 제조사의 장비 규격에 대한 특정한 명령어를 제공하는 소프트웨어 드라이버에 의지한다.

초기의 커널

초창기의 컴퓨터에서 운영 체제 커널은 필수적인 것이 아니었다. 초기의 프로그램은 하드웨어 추상화나 운영 체제의 지원을 받지 않고도 컴퓨터만으로 불러들인 다음 실행될 수 있었으며, 이것은 초창기 컴퓨터들의 일반적인 운영 방식이었다. 다른 프로그램을 실행하기 위해서는 컴퓨터의 전원을 껐다가 켬으로써 다시 입력자료를 읽어들여야 하는 방식이었다. 이러한 과정이 반복되면서 사람들은 로더와 디버거 같은 작은 프로그램들이 상주해 있는 것이,

다른 프로그램으로 교체하거나 새로운 프로그램을 개발하는 데 유리하다는 사실을 알게 되었다. 이와 같은 로더, 디버거들이 초기 운영 체제 커널의 기초가 되었다.

종류

단일형 커널(monolithic kernel) – 커널의 다양한 서비스 및 높은 수준의 하드웨어 추상화를 하나의 덩어리(주소 공간)로 묶은 것이다. 운영 체제 개발자 입장에서 유지 보수가 일반적으로 더 어려우나 성능이 좋다.

마이크로커널(microkernel) – 하드웨어 추상화에 대한 간결한 작은 집합을 제공하고 더 많은 기능은 서버라고 불리는 응용 소프트웨어를 통해 제공한다.

혼합형 커널(hybrid kernel) – 성능 향상을 위해 추가적인 코드를 커널 공간에 넣은 점을 제외하면 많은 부분은 순수 마이크로커널과 비슷하다. 수정 마이크로커널이라고도 한다.

나노커널(nanokernel) – 실질적으로 모든 서비스를 책임진다.

엑소커널(exokernel) – 낮은 수준의 하드웨어 접근을 위한 최소한의 추상화를 제공한다. 전형적으로 엑소커널 시스템에서는 커널이 아닌 라이브러리가 단일형 커널 수준의 추상을 제공한다.

… 커널 별 구조

http://www.terms.co.kr/kernel.htm

커널은 컴퓨터 운영체계의 가장 중요한 핵심으로서, 운영체계의 다른 모든 부분에 여러 가지 기본적인 서비스를 제공한다. 비슷한말로는 '뉴클리어스'라는 용어가 있다. 커널은 쉘과 대비될 수 있는데, 쉘은 운영체계의 가장 바깥부분에 위치하고 있으면서, 사용자 명령에 대한 처리를 담당한다. 커널과 쉘이라는 용어는 IBM 메인프레임을 제외하고, 유닉스와 기타 몇몇 운영체계에서 자주 사용된다.

일반적으로, 커널에는 종료된 입출력 연산 등, 커널의 서비스를 경쟁적으로 요구하는 모든 요청들을 처리하는 인터럽트 처리기와, 어떤 프로그램들이 어떤 순서로 커널의 처리시간을 공유할 것인지를 결정하는 스케줄러, 그리고 스케줄이 끝나면 실제로 각 프로세스들에게 컴퓨터의 사용권을 부여하는 수퍼바이저 등이 포함되어 있다. 커널은 또한, 메모리나 저장장치 내에서 운영체계의 주소공간을 관리하고, 이들을 모든 주변장치들과 커널의 서비스들을 사용하는 다른 사용자들에게 고루 나누어주는 메모리 관리자를 가지고 있다. 커널의 서비스는 운영체계의 다른 부분이나, 흔히 시스템 호출이라고 알려진 일련의 프로그램 인터페이스들을 통해 요청된다.

커널을 유지하기 위한 코드는 지속적으로 사용되기 때문에, 보통 커널은 자주 사용되지 않는 운영체계의 나머지 부분에 의해 덮어씌워져 훼손되지 않도록, 보호된 메모리 영역에 적재된다.

커널을 바이오스와 혼동하면 안된다.

https://namu.wiki/w/%EC%BB%A4%EB%84%90

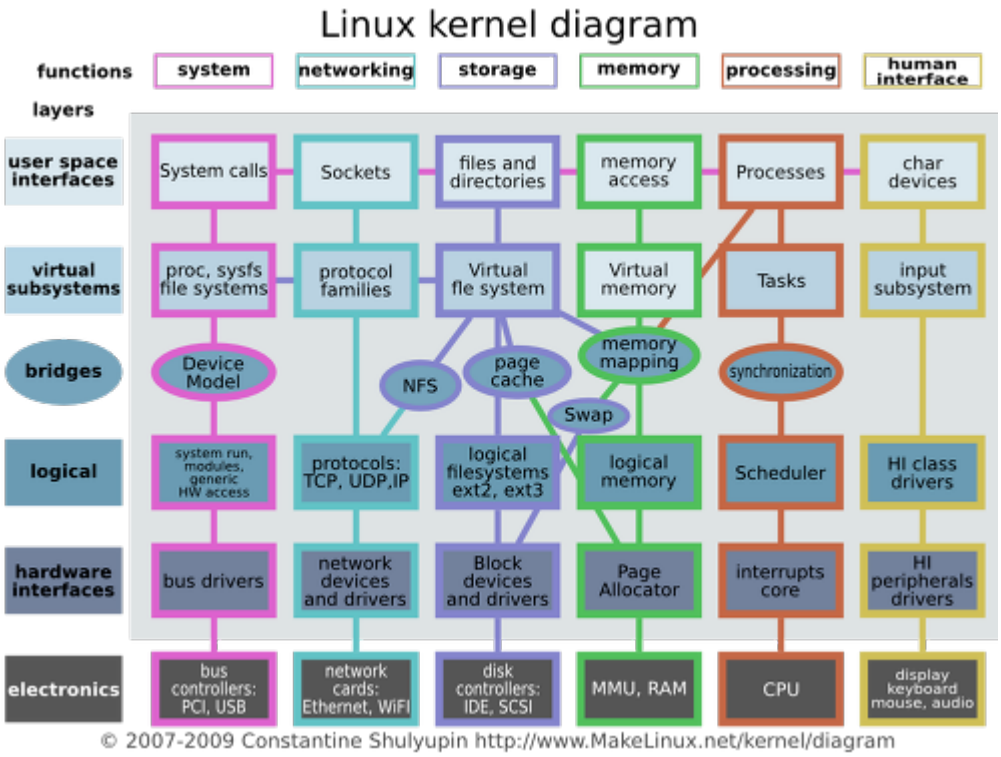
OS의 심장이자, OS를 규정짓는 매우 중요한 부분. 하드웨어의 자원을 자원이 필요한 프로세스에 나눠주고, 덩달아 프로세스 제어(태스크 매니저), 메모리 제어, 프로그램이 운영체제에 요구하는 시스템 콜 등을 수행하는 부분으로 운영체제 맨 하부에서 돌아간다. 쉽게 말해, OS를 하나의 기업체로 비유한

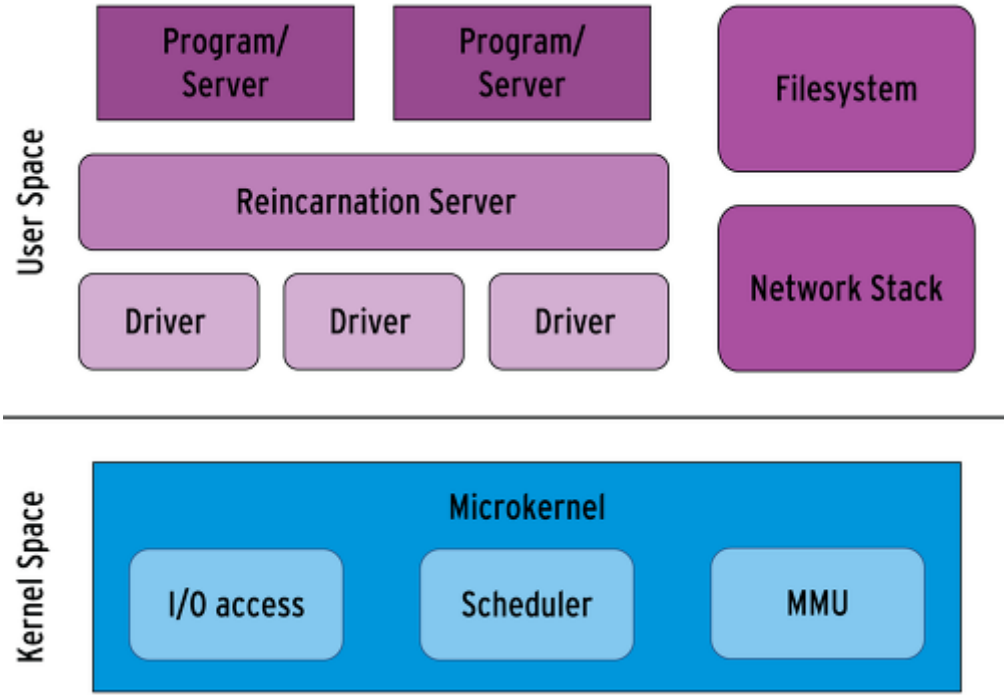
다면 커널은 인사담당 부서인 셈이다. 현재 많이 사용되고 있는 OS는 커널 위에 여러 가지 레이어를 올린 것. 이렇기 때문에 커널이 날아가게 되면 운영체제를 못 쓰게 된다. 이 커널도 한번씩 오작동 하여 정지할 때가 있는데, 이를 가리켜 커널 패닉이라고 부른다. 물론, 현재는 억지로 볼 수는 있지만, 일반적인 상황에선 꽤나 보기 힘들다.

어쨌든 커널로 운영체제의 정체성을 결정하기에 매우 중요하다고 볼 수 있다. 페도라, 우분투등이 다 리눅스로 묶이는 것도 이들이 리눅스 커널을 사용하고 있기 때문.

아직 와닿지 않는다면, NT 커널 기반에서 KERNEL32.dll, Hal.dll, ntdll.dll, ADVAPI32.dll, KERNELBASE.dll (윈7) 등을 날려보면 된다. 이제 부팅이 되지 않는다. 정말로 망했어요 그리고 재부팅하고 비명을 지르면 된다. 자신이 리눅스를 쓴다면 실행 중인 init 프로세스를 날려보자 근데 요즘은 init 에다가 kill 날려도 안 죽는다

1.2. 커널의 종류





원본 출처
http://blog.naver.com/quwoo/120029090135

리눅스 커널이란?
majoR/Sp/Os | 2006/05/25 14:46

리눅스를 만지면서 커널 커널 하지만 의외로 커널이 뭐야? 라고 물으면 모르는 사람들이 많다. 간단히 정리해보자

커널은 시스템에 존재하는 자원을 효율적으로 관리하는 자원 관리자로, 그 기능을 간단히 살펴보면

프로세서 관리 - 처리 속도를 향상시키기 위해 여러 프로세서를 병렬로 연결하여 사용한다. 시스템에서 동작하는 프로세스도 커널에서는 관리해야할 자원이고, 운영체제의 처리 요구에 맞춰 동작할 수 있도록 각 프로세스에 필요한 프로세서를 효율적으로 할당하고 수행하도록 관리한다.

프로세스 관리 - 운영체제에서는 최소한 하나 이상의 프로세스가 동작한다. 프로세스는 다른말로 태스크라고도 하며, 주어진 일을 수행하는 기본 단위다. 커널은 스케줄러를 이용하여 여러 프로세스가 동작할 수 있도록 각 프로세스를 생성하고 제거하며, 외부환경과 프로세스를 연결하고 관리한다.

메모리 관리 - 각각의 프로세스가 독립적인 공간에서 수행할 수 있도록 가상 주소 공간을 제공한다. 가상메모리를 바탕으로 물리적인 한계를 극복할수 있는 기능을 제공한다.

이외에도 **파일 시스템 관리, 디바이스 제어, 네트워크 관리** 정도가 있다.

리눅스 커널은 운영체제에서 가장 중요한 부분이다. 프로세서와 시스템 메모리에 상주하면서 디바이스나 메모리 같은 하드웨어 자원을 관리하고, 프로세스의 스케줄을 관리하여 다중 프로세스를 구현하고, 시스템에 연결된 입출력을 처리하는 운영체제의 핵심 역할을 수행한다.

그렇다면 **리눅스 커널의 특징**은 무엇일까

모놀리식 커널 - 리눅스 커널은 대부분 유닉스 커널과 같은 모놀리식(monolithic)이다. 모놀리식이기 때문에 논리적으로 구분되는 여러 구성요소들이 상호간에 연결되어 동작한다.

비선점형(커널 2.4)과 선점형(커널2.6) - 비선점형 커널은 프로세스의 동작 상태가 사용자 모드에서 커널 모드로 진입하면 외부에서 해당 프로세스를 중지시키지 못한다. 반대로 프로세스가 커널 모드로 동작하더라도 스케줄링 정책이나 다른 외부적인 접근을 통해 프로세스를 강제로 중지시킬 수 있는 경우를 선점형이라고 한다. 2.6커널은 선점형과 비선점형을 선택할 수 있다.

가상 메모리 시스템(VM) - 리눅스 커널은 다양한 플랫폼에서 동작하는 운영체제이므로 i386에서 동작되던 메모리 관리 시스템을 표준화 하여 다양한 MMU 디바이스에 적용할 수 있도록 구성되어있다.

No MMU 지원 - 리눅스 커널은 주로 MMU를 이용한 메모리 관리를 수행하지만 임베디드 시스템에서 사용하는 프로세서의 경우는 MMU디바이스가 없는 경우도 있다. 커널 2.6에서는 MMU가 없는 시스템도 지원한다.

가상 파일 시스템(VFS) - 리눅스에서는 ext2를 비롯 다양한 파일 시스템을 사용할수 있고, 윈도우에서 동작하는 NTFS파일 시스템과 FAT32도 처리할 수 있다.

모듈을 이용한 커널 확장 - 운영 체제가 동작하는 중에도 커널 코드를 추가하거나 삭제할수 있다.

커널 스레드 - 커널 2.4이전에는 매우 제한된 커널 스레드를 지원했지만, 2.6에서부터는 NPTL(Native POSIX Threading Libray)과 NGPT(Next Generation POSIX Threading Package)를 지원한다.

멀티스레드 지원, 멀티 프로세서 지원, GPL 라이센스

http://software-engineer.gatsbylee.com/%EB%A6%AC%EB%88%85%EC%8A%A4-linux-%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80-%EA%B8%B0%EC%B4%88%EA%B3%B5%EB%B6%80/

커널 (Kernel) 이란 무엇인가?

- http://en.wikipedia.org/wiki/Kernel_%28computing%29
- http://bash.cyberciti.biz/guide/Kernel

쉽게 말하면, 소프트웨어와 하드웨어간의 커뮤니케이션을 관리하는 프로그램이다.
구체적으로 말하면, 커널이란 운영체제 (Operating System)에서 가장 중요한 구성요소로서 입출력을 관리하고 소프트웨어로부터의 요청 (System Call) 을 컴퓨터에 있는 하드웨어 (CPU, 메모리, 저장장치, 모니터 등등)가 처리 할 수 있도록 요청(System Call)을 변환하는 역할을 한다.
(User는 Shell을 이용하여 Kernel을 통해 하드웨어를 사용 할 수 있다.)
http://www.devpia.com/MAEUL/Contents/Detail.aspx?BoardID=69&MAEULNO=28&No=11050

커널은 실제적으로 CPU를 제어는 S/W 로 생각 하시면 되겠네요.
그다음 OS 서비스(스케줄링,IPC,동기화)및 드라이버가 올라가죠 실제적으로 OS 포팅할때에
IPC 및 메모리 ,동기화관련 서비스는 빼고 포팅이 되는
Embeded OS 도 있습니다.
자세한건 책한건 사시던가 OS 포팅 한번 해보시면 감이 오실듯 합니다.

추천 하는 OS 는 uCOS2 입니다 시중에 책(Micro C/OS-II)도 있죠.

저는 임베디드쪽을 공부했고 지금은 단말기 디바이스 개발일을 하고 있습니다

질문하신 내용으로 봐서는 커널의 기능 및 역활을 잘 이해를 못하시는 부분이 있으신거 같네요..

운영체제만 전공으로 해서 공부해도 스케줄링 기법등.. 많은부분이 있지만..

윈도우 아키텍처에서 OS 부팅 모드가 크게 3가지 레벨이이 있는거로 알고 있습니다..

Ring 0 모드 (일반적으로 커널모드라고 함) Ring 3 모드 (일반적으로 유저모드라고 함)

(Ring 모드 개념은 리눅스에도 비슷해요 부팅 옵션에서 3이면 유저, 0이면 커널 모드

참고로 제가 많이 사용했던 ubuntu linux 의경우 0-5 Level 이 있었고요 3,4,5 레벨이 유저 레벨,

각 레벨의 차이는 콘솔 모드로 부팅할것인지 GUI로 할것인지 차이 정도 였던거 같네요..;)

그외 레벨들도 존재 했던 거 같은데 이론을 놓은지가 좀돼서 가물가물 하네요..

여튼 중요한건 윈도우라는 OS를 예로 설명 한다고 할때..

OS 에서도 엄연하게 유저 영역과 커널 영역으로 구분이 되어 있습니다..

이건 윈도우를 설계한 개발자들이 만든 아키텍처가 그렇기 때문이죠..

위사항을 확인 하기 위해서 Ring 레벨 대해서 검색해보시면 보다 자세한 설명을 보실수 있고요..

실제 윈도우에서 파일로 존재하는 라이브러리를 확인 해보시려면 (xp 기준)

C:\WINDOWS\system32\kernal32.dll

C:\WINDOWS\system32\user32.dll

위 두가지 dll 파일이 존재 합니다 파일명에서 보시면 아시겠지만 ..

위에서 말씀드린 커널 모드 부분은 kernal32.dll 파일에 구현되어 있고, 유저 모드는 user32.dll 파일에서

구현 되어 있습니다..

커널 모드와 유저 모드는 윗분들 께서 간략하게 설명 해놓으셨는데요..

가장 간단하게 설명 해드리면 커널 모드는 유저모드를 포함한 윈도우 OS 의 모든 부분을 제어 할수가 있습니다

반면 유저모드의 경우 일정부분만 제어가 가능하며 커널모드로 작업이 필요 한부분은 커널모드에 작업을

요청하는 방식으로 이루어지며 유저모드에서 직접적으로 커널모드의 크리티컬한 작업을 할수 없습니다 (권한이 없음)

위 처럼 커널모드가 OS의 실질적인 모든 작업을 관리 하기 때문에 보안쪽 회사에서는 커널모드 쪽 작업을 베이스로

하게 되어집니다..

그리고 위에서 말씀 드린 dll 을 이용하여 Win32 API 를 통하여 사용 되고 있습니다

Window 쪽으로 일하시려면 API 는 기본이죠 .. 특히 보안쪽 가시려면 Native API 까지 알아야..

하는거로 알고 있습니다 실제 Native API 들이 해킹, 보안 의 주축이 되는 함수들이 많이 들어 있기 때문에..

모든 보안제품들, 공격 루틴들이 Native API 들을 통해서 많은 해킹기법으로 이용되는거로 알고 있어요..

커널의 역활설명을 간략히 드렸는데요..

책보다 사이트가 좀더 공부하시는데 도움 되실꺼 같네요

http://kldp.org/

리눅스에 관심이 있으시면 잘 아실테지만 .. 우리나라 리눅스 포럼중 가장 큰 규모이며 업데이트가 잘되고 있습니다

고수분들도 많으시고요.. ^^ 윈도우쪽은 저도 그닥 잘모르겠네요..

(참고로.. 전 임베디드쪽을 공부하면서 윈도우도 시스템 공부를 했지만 리눅스쪽을 좀더 많이 공부를 해서요..^^;;)

윈도우쪽은 API 는 윗분들 의견에서 나온 winapi 사이트에서 기본 베이스 공부하시기 좋고요

Native API 는 블로그나 검색으로 찾으셔야 할꺼에요.. 영어 압박이 좀 있을꺼에요..^^;

그리고 많은 OS 종류들이 존재 합니다.. 윈도우, 리눅스, RTOS 등..

이 중 사람들이 리눅스에 관심을 많이 가지며 작업하는거중 하나가 open 소스 기반이기 때문에..

리눅스의 라이센스 GNU 정책 때문에 open 소스 기반으로 개발 되어서 커널이 공개 되고 있습니다.

리눅스 장점중 하나죠.. 윈도우 같은경우 철저히 소스 공개를 하지 않기 때문에 지금 알려진 대부분의

내용들은 많은 해커들이 해킹을 통해서 알아낸것 입니다. 물론 Microsoft MVP 자격을 얻으면 windows 소스를

열람할수 있는 기회가 주어지는 것도 있더군요.. ;; (요즘은 해킹을통해서 윈도우가 많이 공개 된거로 알고 있어요..)

쓸데 없는 얘기가 길어졌는데요.. 커널 내부를 공부하시기에는 윈도우 보다는 리눅스가 접근하기 용이 합니다..

하지만 중요한건 윈도우와 리눅스는 아키텍처가 틀린 부분이 많기 때문에.. 리눅스를 공부했다고 윈도우를 다 아는게

아니죠.. 그점을 유의 하셔야 하고요.. 윈도우 같은경우 처음부터 접급해서 공부하기 어려운 부분이 많습니다..

그래서 전 리눅스 기반에서 공부를 많이 했고요.. 핵심적인 개념들은 비슷한게 많아요..;;

RTOS 중에서도 보시면 아시겠지만 리눅스 기반으로 개발 된것들이 많이 있습니다..

어느 부분을 공부하시던지 공통적인 부분은 나오길 마련이죠..^^
윈도우쪽에 보시면 rootkit 이란 사이트가 있고요..

www.rootkit.com // 윈도우 쪽애서 고급 기법들이 많이 공개 되고 있습니다.

최근에 루트킷 이란 책으로 발간된 국내 서적도 있습니다..

http://book.naver.com/bookdb/book_detail.php?bid=4298506 링크 참조

저도 WDK 쪽이나 rootkit 쪽은 관심이 많아서.. ^^

책은 구입해뒀는데 .. 시간상.. 이런 저런 핑계로 못보고 있네요..

평은 좋은거로 알고 있는데요 .. 기초가 없으면 처음 보기는 다소 어려운 내용 이지만..

기본적인 내용이 많아서 보시면 도움 될꺼 같네요..

해킹 기법적인 내용이 많지만.. 커널을 공부하는데 있어서 해킹 기법 만한게 없죠.. ㅎ

그럼 수고하세요.. ㅎ

커널이라고 하면 보통 운영체제에 많이 쓰이는 개념인데요.

다른말로 뉴클레어스, 코어라고도 합니

OS의 핵심부분을 의미하고요, 가장빈번하게 프로세스를 제어하는 부분이라고 할 수 있습니다.

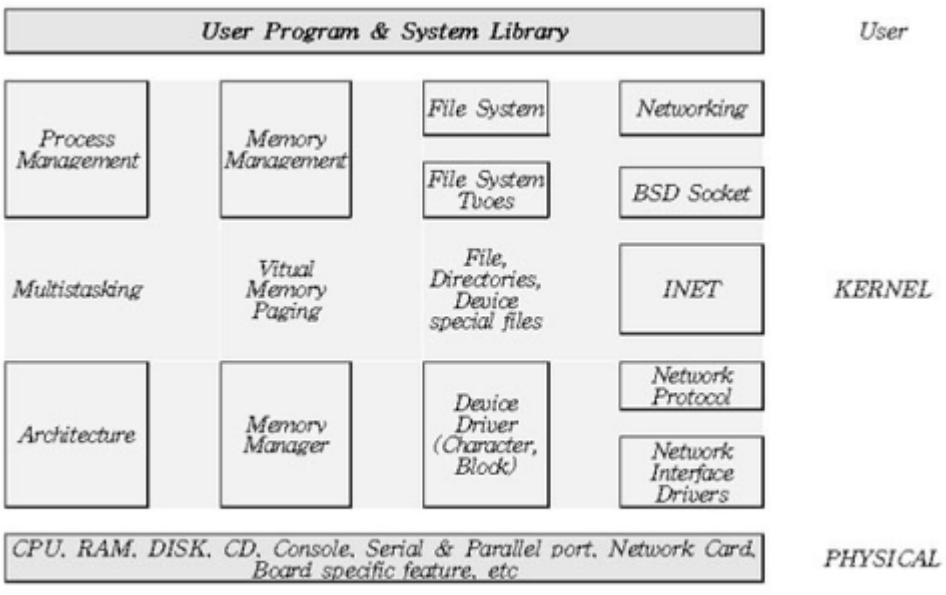
대부분의 인터럽트가 처리되기 때문에 오퍼레이팅시스템에서 중요한 부분이고요,

때문에 주로 메인 메모리에 상주됩니다.

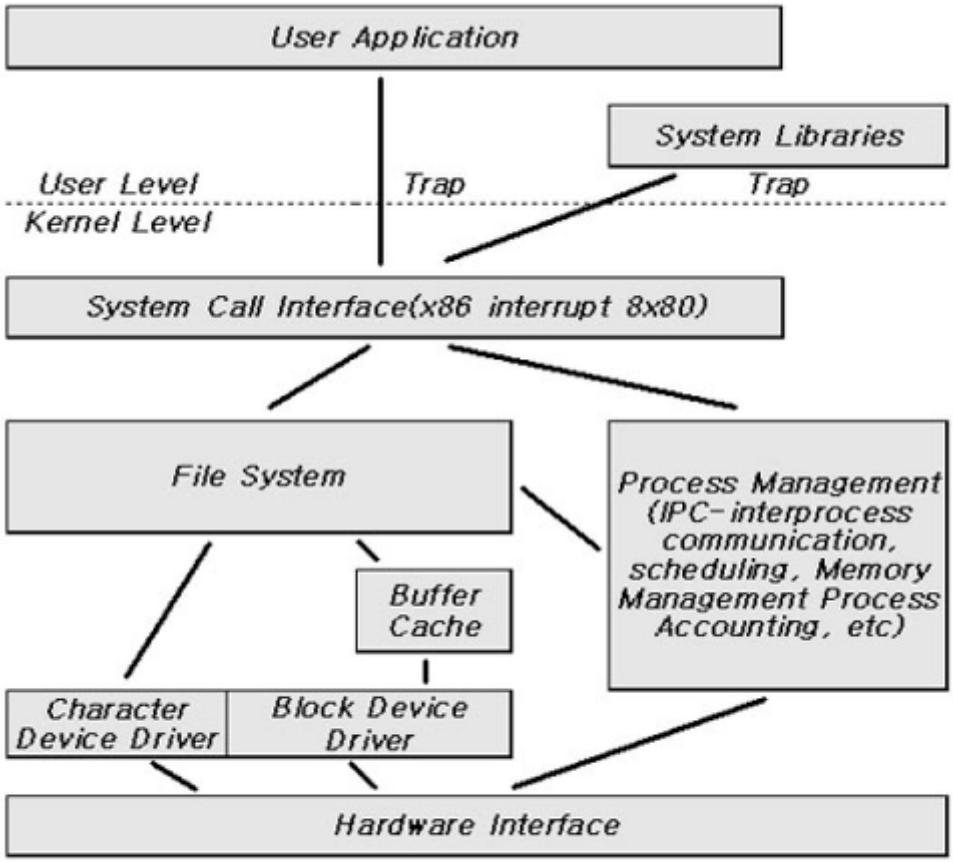
제가 전공이 거의 OS라서요.

오에스 책보시면 자세히 알 수 있어요.

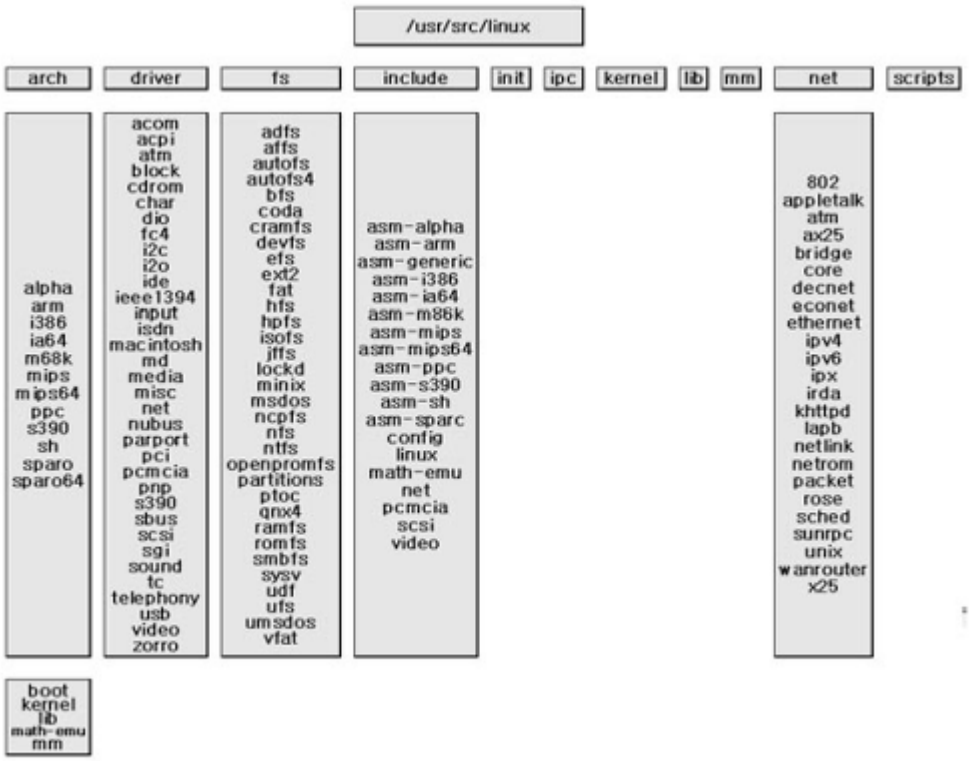
뭐 책들은 많습니다. 일단 기본적인 운영체제론 만 보셔도 확실할듯..



Linux Kernel 의 구조



커널 구성 요소간의 연관 관계



Linux Kenel 의 source tree 구성

커널(Kernel)

커널은 운영체제의 핵심을 이루는 요소로서 컴퓨터내의 자원을 사용자 프로그램(User Application)이 사용할 수 있도록 관리하는 프로그램이다.
커널은 프로세스, 파일 시스템, 메모리, 네트워크의 관리를 맡는다. 사용자 프로그램은 이러한 기능들을 정해진 규칙에 따라서 커널에 요구하게 되며, 커널은 이러한 요구들을
만족시켜 주도록 구성되어 있다. 사용자 프로그램은 시스템 라이브러리(System Library)의 도움을 받거나 아니면 직접적으로 소프트웨어 인터럽트(Software Interrupt)를 이용해서 트랩(Trap)을 걸어 커널에
접근하게 되며, 이러한 모든 접근은 시스템 콜 인터페이스(System Call Interface)를 통하여 이루어진다.

커널은 대부분이 C 코드(Code)로 작성되어 있으며, 프로세스의 구조에 의존적인(Processor Architecture Dependent) 부분들과 속도를 요하는 부분만 기계어 코드
(Assembly Code)로 작성되어 있다. 따라서 커널의 구조에 의존적인(Architecture Dependent) 부분을 고침으로 해서 새로운 프로세스로의 포팅(Porting)이 가능해진다.
[참고] Add-on Linux Kernel Programming [2001. 11 이귀영 著]
이하 내용은 `Running Linux 4rd` 에서 발췌하였습니다. 06. 5. 15 - subzero

커널은 운영체제의 핵심에 해당한다. 커널은 사용자 프로그램과 하드웨어 장치간의 인터페이스를 제어하고, 다중 작업을 지원하려고 프로세스 스케쥴링을 하며, 기타 시스템의 다양한 측면을 관리하는 소스 코드이다. 커널은 시스템에서 동작하는 어떤 개별 프로세스가 아니라, 항상 메모리에 존재하면서 모든 프로세스가 사용할 수 있는 루틴(routines)들의 집합이라고 할 수 있다. 커널 루틴은 여러 가지 방법으로 호출된다. 커널을 사용하는 직접적인 방법은 시스템 호출(system call)을 사용하는 것으로 커널이 호출
한 프로세스를 위해 특별한 코드를 실행한다. 예를 들어, read 시스템 호출은 파일 기술자(file descriptor)로부터 자료를 읽는다. 프로그래머에게는 C 함수처럼 보이지만 read 의 실제 코드는 커널 안에 있다.

커널 코드는 다른 상황에서도 실행된다. 예를 들어, 하드웨어 장치가 인터럽트(interrupt)를 발생시키면 커널 내부의 인터럽트 처리기를 찾는다. 어떤 프로세스가 어떤 행동을 취하고 결과를 기다려야 할 때는 커널이 개입하여 그 프로세스를 잠들게 하고 다른 프로세스가 활동할 수 있게 스케쥴링한다. 마찬가지로 커널은 한 프로세스에서 다른 프로세스로 이동할 때 클럭 인터럽트를 사용하여 프로세스간의 제어권을 신속하게 옮긴다. 다중 작업은 대개 이런 방식으로 이루어진다.

리눅스 커널은 단일(monolithic) 커널 방식으로, 모든 핵심 함수와 장치 드라이버가 커널의 일부다. 일부 운영체제는 장치 드라이버와 다른 코드가 필요할 때 적재되고 실행되는 마이크로 커널(micro kernel) 아키텍처를 사용하고 있다. 따라서 이 코드는 항상 메모리에 있을 필요가 없다. 두 방식에는 각각 장·단점이 있다.

대부분의 유닉스 구현에서는 일반적으로 단일 커널 아키텍쳐가 사용되며, 이는 시스템 V, BSD 와 같은 전통적인 커널에서 채택한 방식이다. 하지만 리눅스는(사용자의 명령에 따라 메모리에 적재하거나 빼
낼 수 있는) 적재 가능한 장치 드라이버도 지원하고 있다.

인텔 플랫폼에서 동작하는 리눅스 커널은 인텔 x86 프로세서(80386 부터 현재의 펜티엄 4 가지)의 특별한 보호모드 특성을 사용하도록 개발되었다. 특히 리눅스는 보호모드, 기술자 기반의 메모리 관리 정책과 기타 수많은 인텔 프로세서의 향상된 기능을 활용한다. x86 보호 모드 프로그래밍에 익숙한 사람은 이 프로세서가 유닉스 같은 다중 작업 시스템용(사실은 멀틱스의 영향을 받은 것이다)으로 설계된 것을 알고 있을 것이다. 리눅스는 이 기능을 십분 활용한다.

대부분의 현대적인 운영체제와 마찬가지로 리눅스는 다중 프로세서(Multi Processing) 운영체제다. 메인보드에 하나 이상의 CPU가 있는 시스템을 지원한다. 이 기능을 통해 동시에(또는 '병렬로') 다른 프로그램을 서로 다른 CPU에서 실행할 수 있다. 리눅스는 또한 하나의 프로그램 안에서 메모리의 데이터를 공유하고 있는 '제어 스레드'를 다중으로 생성하는 프로그래밍 기법인 스레드(thread)도 지원한다. 리눅스는 여러 가지 커널-레벨 그리고 사용자-레벨 스레드 패키지를 지원한다. 리눅스의 커널 스레드는 다중 CPU에서 동작하므로, 하드웨어 병렬 처리 능력을 제대로 활용한다고 할 수 있다. 리눅스 커널 스레드 패키지는 POSIC 1003.1c 표준과 호환된다.

리눅스 커널은 필요에 따라 페이징하고, 적재하는 실행 파일을 지원한다. 즉 프로그램 세그먼트 중 실제로 쓰이는 부분만 메모리로 읽어들인다. 그리고 한 프로그램이 동시에 여러번 실행 중일 때에는 프로그램의 복사본 하나만 메모리에 적재한다. 실행 프로그램은 동적으로 링크된 공유 라이브러리를 사용한다. 즉 실행 프로그램들은 디스크에 있는 라이브러리 파일 내의 공용 라이브러리 코드를 공유할 수 있다. 이렇게 하면 실행 파일이 차지하는 디스크 공간을 줄일 수 있다. 또한 라이브러리 코드 복사본 하나가 메모리에 올라가므로, 전체 메모리 사용량을 줄일 수 있다. 물론 공유 라이브러리에 의존하지 않고 '완전한' 실행파일을 가지고 싶은 사람들을 위해 정적 링크 라이브러리도 있다. 리눅스 공유 라이브러리는 실행 중 동적으로 링크되므로, 프로그래머는 실행 파일을 변경하지 않고도 라이브러리 모듈을 자신이 만든 루틴으로 변경할 수 있다.

시스템 메모리를 최대한 활용하기 위해, 리눅스는 디스크 페이징이라는 기능으로 가상 메모리(virtual memory)를 구현한다. 즉 일정량의 스왑 영역(swap space)을 디스크에 할당할 수 있다. 애플리케이션이 머신에 실제 설치되어 있는 물리적인 메모리보다 많은 메모리를 필요로 하는 경우, 리눅스는 사용하지 않는 페이지를 디스크에 스왑시킨다(페이지page란 운영체제가 메모리를 할당하는 단위다. 대부분의 아키텍처에서는 4KB다). 그리고 스왑아웃 시킨 페이지에 다시 접근할 때는 디스크에서 페이지를 읽어와 메인 메모리에 다시 넣는다. 이 기능을 통해 큰 애플리케이션을 실행할 수 있고, 동시에 더 많은 사용자를 지원할 수 있다. 물론 스왑으로 물리적인 RAM을 대신할 수는 없다.

이는 메모리보다 디스크에서 페이지를 읽는 것이 더 느리기 때문이다. 리눅스 커널은 메모리보다 디스크에 자주 접근하는 것을 피하기 위해, 최근 접근한 일부파일을 메모리에 보관한다. 커널은 디스크 접근 캐시를 위해 시스템에 남은 모든 메모리를 사용하므로, 시스템 부하가 적을 때는 여러 파일에 접근해도 메모리에서 재빨리 처리한다. 만약 사용자 애플리케이션이 더 많은 물리적인 메모리를 요구하면, 디스크 캐시의 크기를 줄인다. 물리 메모리를 쓰지 않고 방치하는 경우는 절대 없다.
또한 디버깅을 쉽게 하기 위해 잘못된 메모리 주소에 접근하는 등의 잘못된 연산을 수행한 프로그램의 코어 덤프(core dump)를 발생시킨다. 프로그래머는 프로그램을 실행시
킨 디렉토리에 저장된 core라는 코어 덤프를 사용하여 비정상 종료의 원인을 판단한다.

커널의 버전 번호
major.minor.patchlevel
리눅스 커널에는 항상 두 개의 '최신' 커널이 존재한다. 하나는 '안정(stable) 버전'이고 다른 하나는 '개발(development) 버전'이다. 안정 버전은 가장 최신의 실험적인 기능을
해킹하기 보다는 안정적이며 잘 작동하는 시스템을 원하는 대부분의 리눅스 사용자를 위한 것이다. 반면 개발 버전은 인터넷을 통해 개발자들이 새로운 기능을 추가하고, 테스
트하기 때문에 빠르게 변화한다. 안전 버전의 변화는 주로 버그 수정이나 보안 패치인 반면, 개발 버전의 변화는 완전히 다른 커널 하부시스템에서부터 좀 더 나은 성능을 위해
장치 드라이버를 개선한 것까지 모든 것이 그 대상이다.
안전 버전이 짝수 마이너(minor) 버전 번호(예를 들어, 2.4)를 가지는 반면, 개발 버전은 홀수 마이너 버전 번호(예를 들어, 2.5)를 갖는다. 따라서 현재 안정 버전이 2.6이라
면 개발 버전은 2.7이 된다.
각 커널 버전은 2.4.19 또는 2.5.12와 같이 셋째 '패치 레벨(patch level)' 버전 번호를 사용한다. 패치 레벨은 각 커널 버전의 갱신 횟수를 나타내며 높은

번호일수록 최신판이다.

2 구독하기

'프로그래밍 > 프로그래밍 공부' 카테고리의 다른 글

다중 스레딩(multi threading) (0)	2016.02.17
유저 레벨 스레드(user level thread) / 커널 레벨 스레드(kernel level thread) (0)	2016.02.17
커널(kernel) (0)	2016.02.17
재 진입성(re-entrant) (0)	2016.02.16
CPU (0)	2016.02.15
스레드(Thread)의 기본적인 특성 (0)	2016.02.15

Posted by GENESIS8
트랙백 0개, 댓글 0개가 달렸습니다

댓글을 달아 주세요

: 이름

: 패스워드

http:// : 홈페이지

비밀글

댓글 달기

이전 1 … 18 19 20 21 22 23 24 25 26 … 148 다음

GENESIS8

검색

- 관리자
- 글쓰기

카테고리

- 블로그 (148)
  - 프로그래밍 (126)
    - 프로그래밍 공부 (37)
    - C, C++ (12)
    - C++++ (C#) (5)
    - (비공개) AWS (0)
    - JAVA (0)
    - ASP (3)
    - Html (0)
    - PHP (0)
    - Windows (1)
    - Network (2)
    - Unity (1)
    - windows API (6)
    - DirectX 3D (0)
    - CMD (0)
    - 소켓 프로그래밍 (11)
    - 서버 프로그래밍 (3)
    - 웹 프로그래밍 (9)
    - DB (2)
    - MySQL (7)
    - 잡다한 궁금증 (1)
    - 자료구조, 알고리즘 (3)
    - 디자인 패턴 (1)
    - UML (0)
    - 프로그래밍 용어 사전 (20)
    - 에러 정리 (0)
  - 제작소 (12)
    - 프로젝트 오일러 (12)
    - 개인 제작 (0)
  - 유용한 정보들 (4)
    - 프로그래밍 관련 (4)

- 기타 정보 (1)
- ETC (2)
  - 잡담 (2)
- 포트폴리오 (3)
- 취미 (0)

TAG_LIST

최근에 올라온 글

- 이벤트 스케줄러(Ev... (1)
- 윈도우 서비스.
- ASP란 뭘까...
- IIS 및 ASP 사용 환...

최근에 달린 댓글

- 글을 되게 잘 정리... BBingStory 2018
- 감사히 담아가겠습... 맛 밀 2018
- 훌륭한 예제네요.... 요원009 2018
- 좋은글 감사합니다!!. 엉강 2018

최근에 받은 트랙백

글 보관함

- 2016/06 (4)
- 2016/03 (4)
- 2016/02 (25)
- 2016/01 (23)

달력

« 2019/10 »

일	월	화	수	목	금	토
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

링크

- Total : 391,932
- Today : 440
- Yesterday : 682