# Forensic Study - 2
# OS + Windows Theory (1)

2017.10.09

KUICS

2014210009 주어진

# 목차

# What is OS?

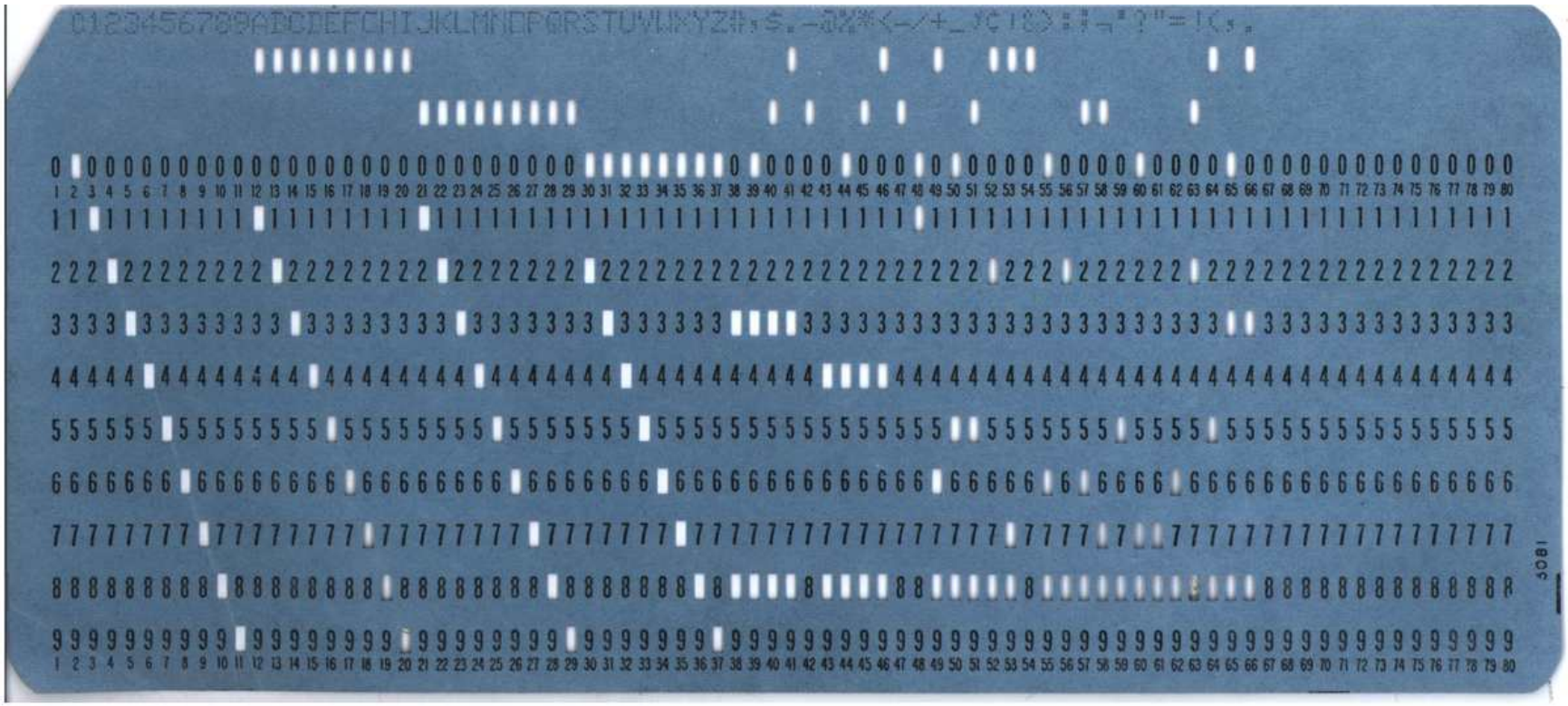
운영 체제(運營體制, 문화어: 조작체계) 또는 오퍼레이팅 시스템(영어: Operating System, OS)은 시스템 하드웨어를 관리할뿐 아니라 응용 소프트웨어를 실행하기 위하여 하드웨어 추상화 플랫폼과 공통 시스템 서비스를 제공하는 시스템 소프트웨어이다. 최근에는 가상화 기술의 발전에 힘입어 실제 하드웨어가 아닌 하이퍼바이저 위에서 실행되기도 한다.

입출력과 메모리 할당과 같은 하드웨어 기능의 경우 운영 체제는 응용 프로그램과 컴퓨터 하드웨어 사이의 중재 역할을 한다. 그러나 응용 프로그램 코드는 일반적으로 하드웨어에서 직접 실행된다. 운영 체제는 휴대 전화, 게임기에서부터 슈퍼컴퓨터, 웹 서버에 이르기까지 컴퓨터를 포함하는 거의 모든 장치에서 볼 수 있다. 운영 체제는 한 면으로는 소비자를, 다른 한 면으로는 프로그램 개발자를 함께 하나의 시장으로 데려다 놓을 수 있는 양면 플랫폼이다. 잘 알려진 현대의 PC 운영 체제에는 마이크로소프트 윈도우, 맥 OS X, 리눅스가 있다. 이 밖에 BSD, 유닉스 등의 PC용 운영 체제도 존재한다.

운영 체제는 실행되는 응용 프로그램들이 메모리와 CPU, 입출력 장치 등의 자원들을 사용할 수 있도록 만들어 주고, 이들을 추상화하여 파일 시스템 등의 서비스를 제공한다. 또한 멀티태스킹을 지원하는 경우, 여러 개의 응용 프로그램을 실행하고 있는 동안, 운영 체제는 이러한 모든 프로세스들을 스케줄링하여 마치 그들이 동시에 수행되는 것처럼 보이는 효과를 낸다.

출처 : https://ko.wikipedia.org/wiki/%EC%9A%B4%EC%98%81_%EC%B2%B4%EC%A0%9C

# What is OS?



# What is OS?

사람이 직접 스케줄링하기는 어렵다.

그러면? 컴퓨터에게 맡긴다.

- 일괄 처리 : 사람이 스케줄링
- Automatic Job Sequencing : 소프트웨어가 담당.

# Kernel Design / Ring

- Monolithic
- Micro
- Hypervisor

# Kernel Design / Ring

- Monolithic

장점 : Application과 Kernel Service가 같은 주소 공간에 위치, 빠름
단점 : 일부분을 수정할 시에 전체에 영향을 미침.

# Kernel Design / Ring

- Micro

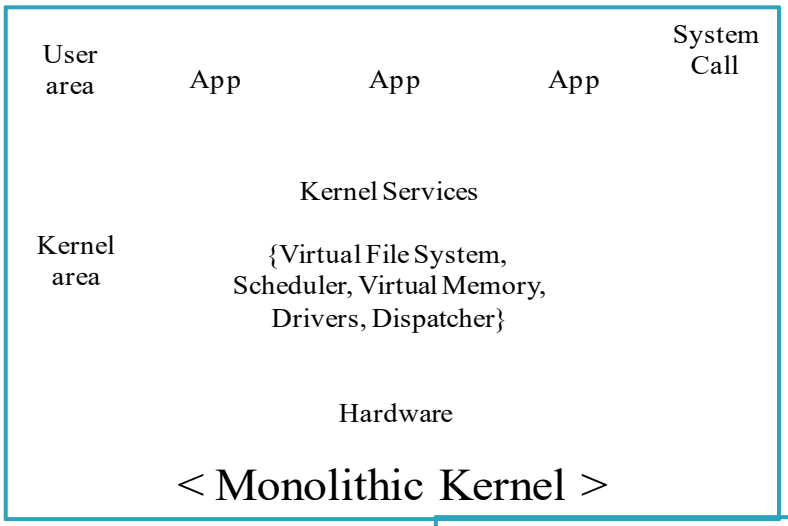
커널 서비스를 기능에 따라 모듈화. 각각 독립된 주소 공간. 모듈들을 서버라 부른다.
장점 : Monolithic보다 안정적이며 유지/보수 용이
단점 : 아무래도 독립된 서버들간의 Context Switching을 하다보니 성능이 떨어짐.

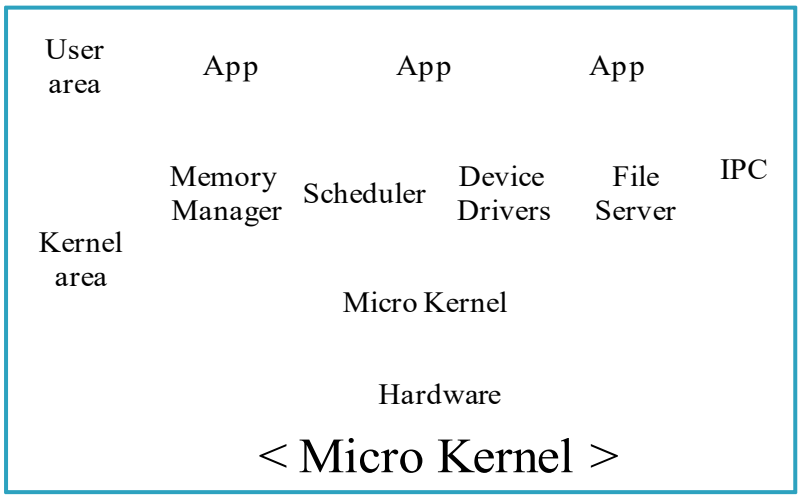
# Kernel Design / Ring

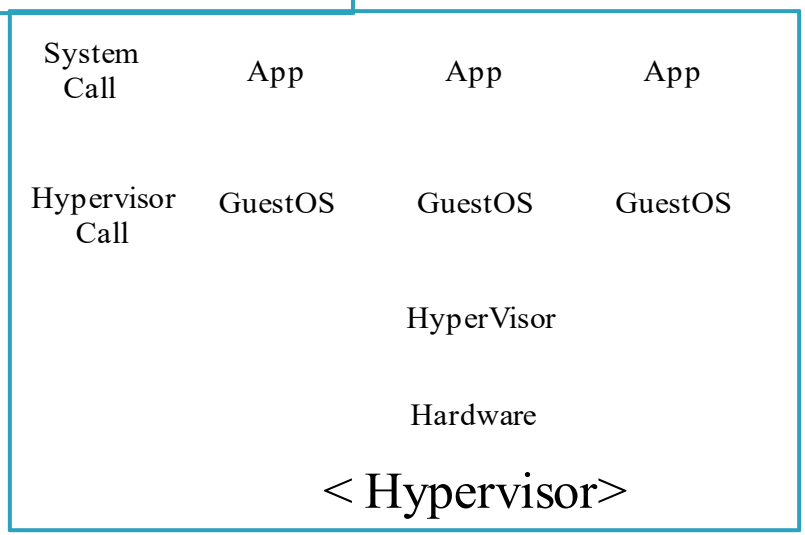
- Hypervisor

가상화된 컴퓨터 H/W 지원을 위한 계층. 각 Guest OS간의 CPU, 메모리 등 시스템 자원 분배하는 최소한의 역할만 수행.

# Kernel Design / Ring



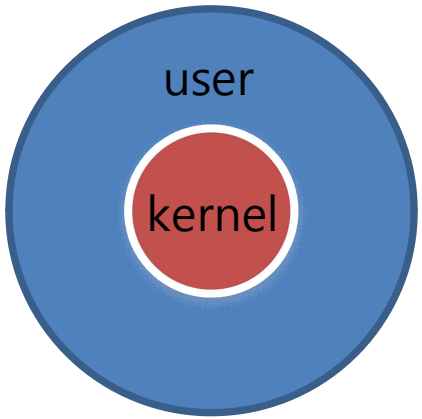



# Kernel Design / Ring

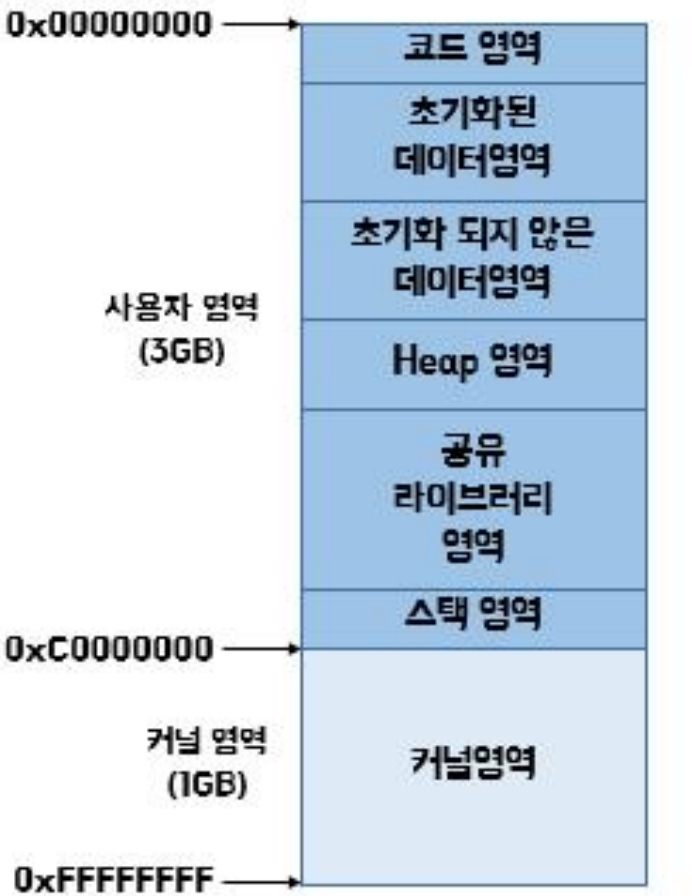
**Ring**

**시스템 보호를 위해 필요한 구조. 각각의 모드별로 권한이 설정.**
**H/W 지원이 필요하다.**

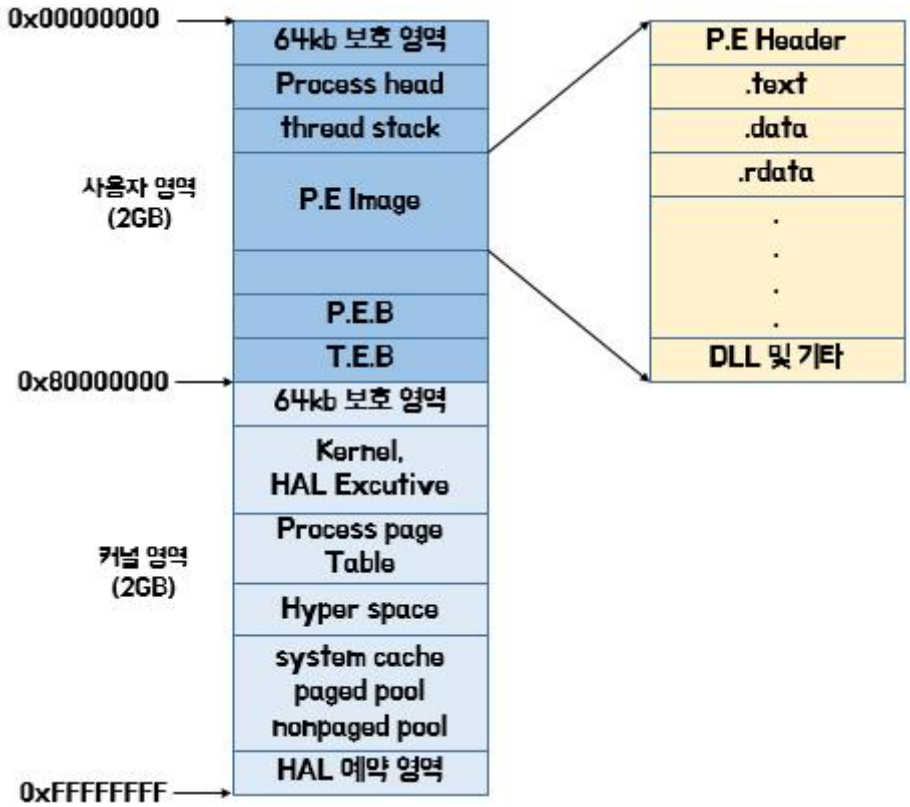
**Intel - Ring 0-3으로 4개의 모드**
**그 외 - 2개의 모드**



# Memory Structure



# Memory Structure



# Windows Memory Structure ( Advanced )

여기서부터는 이해 잘 못해도 됩니다.

이해하면 매우매우 좋고요. :)

커널은 너무 깊이 내려가야 해서 Pass!

# Windows Memory Structure ( Advanced )

## PEB? TEB?

```
typedef struct _PEB {
  BYTE                          Reserved1[2];
  BYTE                          BeingDebugged;
  BYTE                          Reserved2[1];
  PVOID                         Reserved3[2];
  PPEB_LDR_DATA                 Ldr;
  PRTL_USER_PROCESS_PARAMETERS  ProcessParameters;
  BYTE                          Reserved4[104];
  PVOID                         Reserved5[52];
  PPS_POST_PROCESS_INIT_ROUTINE PostProcessInitRoutine;
  BYTE                          Reserved6[128];
  PVOID                         Reserved7[1];
  ULONG                         SessionId;
} PEB, *PPEB;
```

```
typedef struct _TEB {
    PVOID Reserved1[12];
    PPEB ProcessEnvironmentBlock;
    PVOID Reserved2[399];
    BYTE Reserved3[1952];
    PVOID TlsSlots[64];
    BYTE Reserved4[8];
    PVOID Reserved5[26];
    PVOID ReservedForOle;  // Windows 2000 only
    PVOID Reserved6[4];
    PVOID TlsExpansionSlots;
} TEB, *PTEB;
```

# Windows Memory Structure ( Advanced )

**Segment Descriptor Table**

**FS 세그먼트 레지스터는 현재 thread의 TEB를 가리키는데 사용됨.**
**32비트 운영체제에서는 메모리가 4G인데 FS 레지스터는 16bit이다.**

**실제로 FS 레지스터가 직접 TEB 주소를 가리키는게 아니라 TEB 주소를 가지고 있는 Segment Descriptor Table의 Index 값을 가지고 있어서 가능.**

**Segment Descriptor Table은 커널 메모리 영역에 존재하고,**
**Global Descriptor Table Register (GDTR)에 그 주소가 저장.**

# Windows Memory Structure ( Advanced )

```
void main() {
        _TEB* tib = NtCurrentTeb();
#ifdef _AMD64_
        uint64_t x64sstib = __readgsqword(0x30);
        uint64_t x64id = __readgsqword(0x40);
        uint64_t x64peb = __readgsqword(0x60);
        printf("SubSystemTIB in x64 : %08I64X₩nProcess ID in x64 : %08I64d₩nPEB in x64 : %08I64X₩n",
                x64sstib,
                x64id,
                x64peb);
#else
        uint32_t x86sstib, x86peb, x86id;
        __asm {
                mov eax, FS:[0x18]
                mov x86sstib, eax
                mov eax, FS:[0x20]
                mov x86id, eax
                mov eax, FS:[0x30]
                mov x86peb, eax
        }
        printf("SubSystemTIB in x86 : %08X₩nProcess ID in x86 : %08d₩nPEB in x86 : %08X₩n",
                x86sstib,
                x86id,
                x86peb);
#endif
        system("pause");
}
```

# Windows Memory Structure ( Advanced )

( 64bit 기준 )

ntdll!_PEB+0x002 = BeingDebugged ( IsDebuggerPresent이 참조하는 값 )

ntdll!_PEB+0x018 = LDR ( 프로세스에 로드 된 모듈에 대한 정보를 가지고 있으며, 디버깅 시에 Heap 영역중 쓰이지 않는 부분을 0xABABABAB로 채움. )

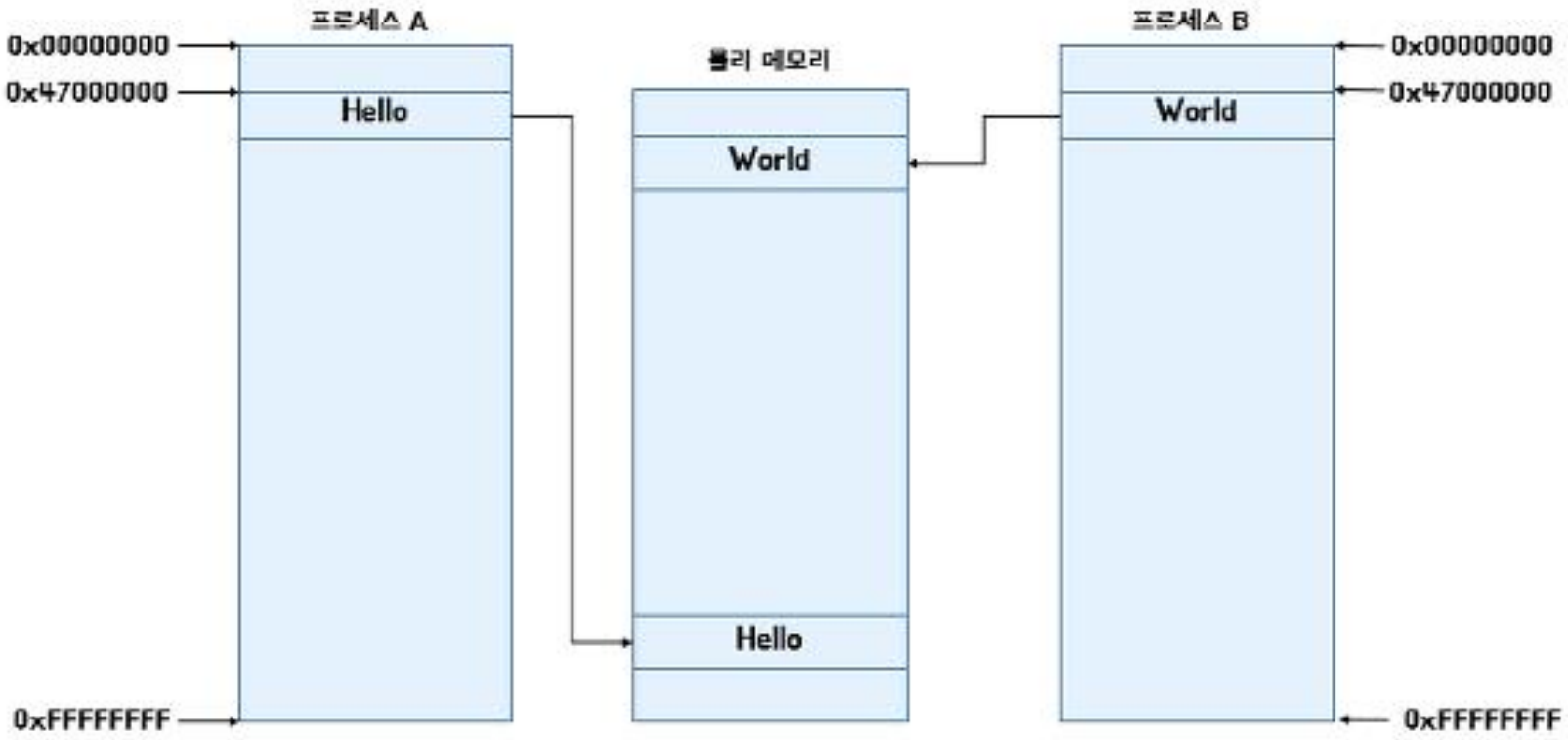
ntdll!_PEB_LDR_DATA+0x030 = InInitializationOrderModuleList

Address of InInitializationOrderModuleList - 0x20 = LDR_MOUDLE(LDR_DATA_TABLE_ENTRY) 주소 / 접근하면 ntdll.dll이 첫번째 dll이다.

LDR list는 Double linked list므로 InLoadOrderLinks를 계속 들어가다 보면 원하는 dll의 주소를 얻을 수 있음.

이를 이용해 ASLR 무시가능.

# Memory Structure



# Memory Structure

**페이징 기법(paging)은 컴퓨터가 메인 메모리에서 사용하기 위해 2차 기억 장치로부터 데이터를 저장하고 검색하는 메모리 관리 기법이다.**

**즉, 가상기억장치를 모두 같은 크기의 블록으로 편성하여 운용하는 기법이다. 이때의 일정한 크기를 가진 블록을 페이지(page)라고 한다. 주소공간을 페이지 단위로 나누고 실제기억공간은 페이지 크기와 같은 프레임으로 나누어 사용한다.**

**페이지 테이블(Page Table)은 프로세스의 페이지 정보를 저장하고 있으며, 하나의 프로세스는 하나의 페이지 테이블을 가진다.**

**테이블은 다음과 같이 색인과 내용으로 구성되어 있다.**

**- 색인 : 페이지 번호.**
**- 내용 : 해당 페이지에 할당된 물리 메모리(프레임)의 시작 주소. 이 시작 주소와 페이지 주소를 결합하여 물리 메모리 주소를 알 수 있다.**

출처 : https://ko.wikipedia.org/wiki/%ED%8E%98%EC%9D%B4%EC%A7%95

# Memory Structure

메모리 덤프를 떴을 때,

커널랜드는? - 온전히 보존.
유저랜드는? - No. Why?

# Windows Process / User & Kernel Mode Intro

**Windows에는 중요한 프로세스들이 있다.**

**Explorer.exe : 작업 표시줄 / 바탕 화면 표시 사용자 셸.**

**svchost.exe : Windows에서 동적 링크 파일(DLL)로 작성된 백그라운드 서비스를 실행해주고 관리해주는 호스트 프로그램이다. Windows Service Process의 집합.**

**Ctfmon.exe : 입력 도구 모음**

**Spoolsv.exe : 프린터 / 팩스 지원**

**lsass.exe : 로컬 보안 인증 서버, 강제종료시 BSOD**

**csrss.exe : 콘솔 창, 스레드 생성과 삭제 및 16비트 가상 MS-DOS 환경 담당. 강제종료시 BSOD**

**smss.exe : Session Manager SubSystem, 프로세스 시작 / 시스템 변수 설정등의 작업 수행. 강제종료시 BSOD**

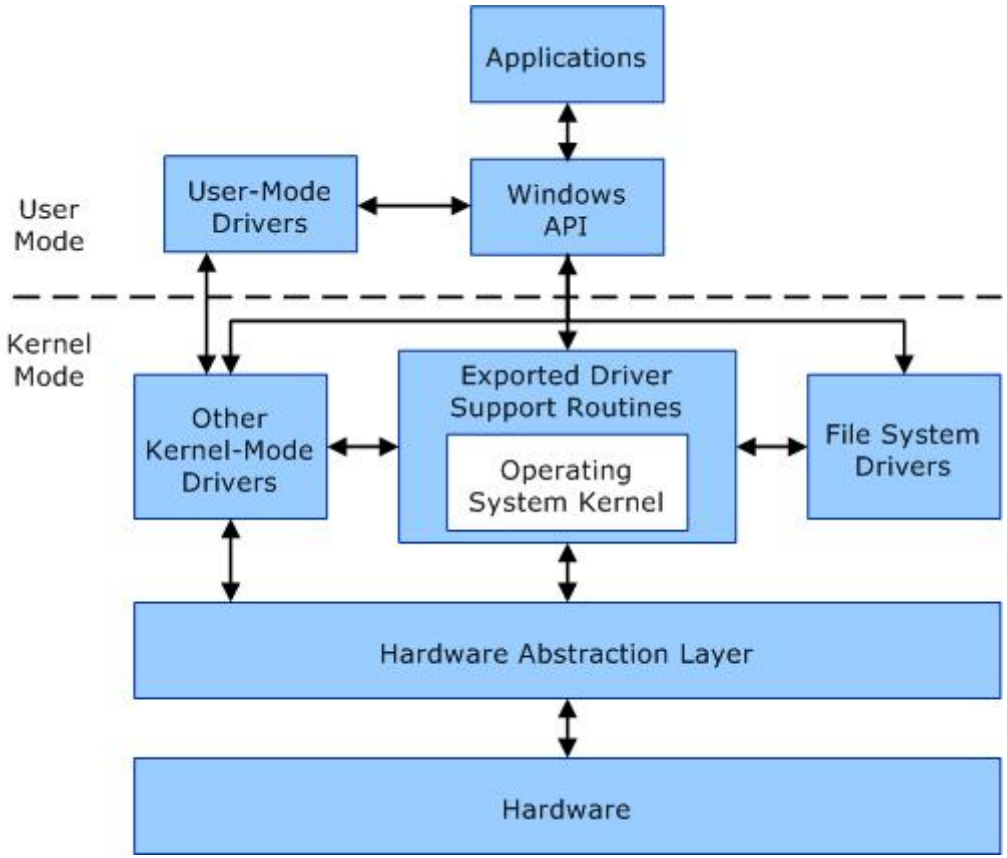
# Windows Process / User & Kernel Mode Intro

**Windows에는 중요한 프로세스들이 있다.**

**System : 시스템 커널 모드 스레드의 집합**

**System Idle Process : 다른 스레드를 처리하지 않을 때의 프로세서 시간**

# Windows Process / User & Kernel Mode Intro



# 감사합니다

## 과제 : 없음!

참조 :
http://babu1447.tistory.com/6
https://en.wikipedia.org/wiki/Protection_ring
https://msdn.microsoft.com/ko-kr/library/windows/desktop/aa813706(v=vs.85).aspx
https://msdn.microsoft.com/ko-kr/library/windows/desktop/ms686708(v=vs.85).aspx
https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode