CS 발표 - 9단원 운영체제 시작하기

Chapter 09 - 운영체제 시작하기

■ 학습 목표 ■

- 운영체제의 이해
- 개발자가 운영체제를 알아야 하는 이유를 알아보자
- 커널이 무엇인지 학습
- 시스템 호출과 이중 모드의 이해
- 운영체제가 제공하는 핵심 서비스의 종류 학습

◆ 운영체제를 알아야 하는 이유

- 대표적 데스크톱 운영체제 - 윈도우, mac OS, 리눅스
- 스마트폰 운영체제 - 안드로이드, iOS
- 컴퓨터 부품들은 전기만 공급하면 마법처럼 알아서 작동하는 것이 아니다.

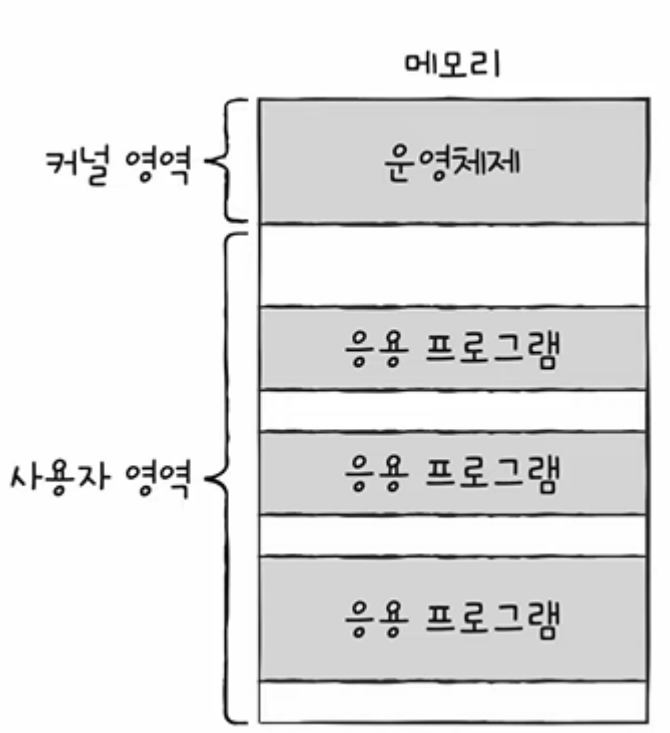
◎ 운영체제란

- 모든 프로그램은 하드웨어를 필요로 한다.
- 프로그램 실행에 마땅히 필요한 요소 ⇒ **시스템 자원 (자원)**
- CPU, 메모리, 보조기억장치, 입출력장치와 같은 컴퓨터 부품들 모두 자원이라 볼 수 있다.
- 모든 프로그램은 실행되기 위해 반드시 자원이 필요하다.

- 실행할 프로그램에 자원을 할당하고, 프로그램이 올바르게 실행되도록 돕는 특별한 프로그램
- ⇒ **운영체제**

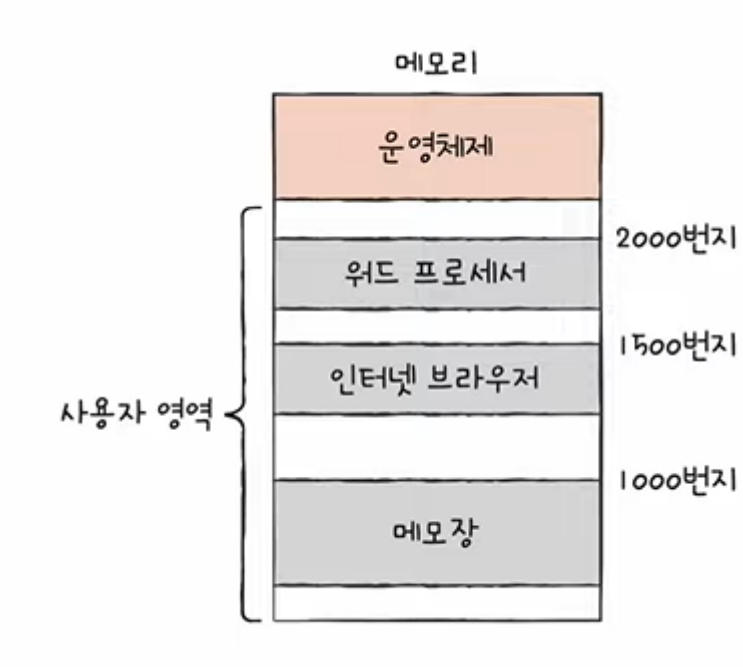
◎ 운영체제의 실행

- 운영체제 또한 여느 프로그램과 마찬가지로 메모리에 적재되어야 한다.
- 다만 운영체제는 매우 특별한 프로그램이기 때문에 항상 컴퓨터가 부팅될 때 메모리 내 **커널 영역**이라는 공간에 따로 적재되어 실행



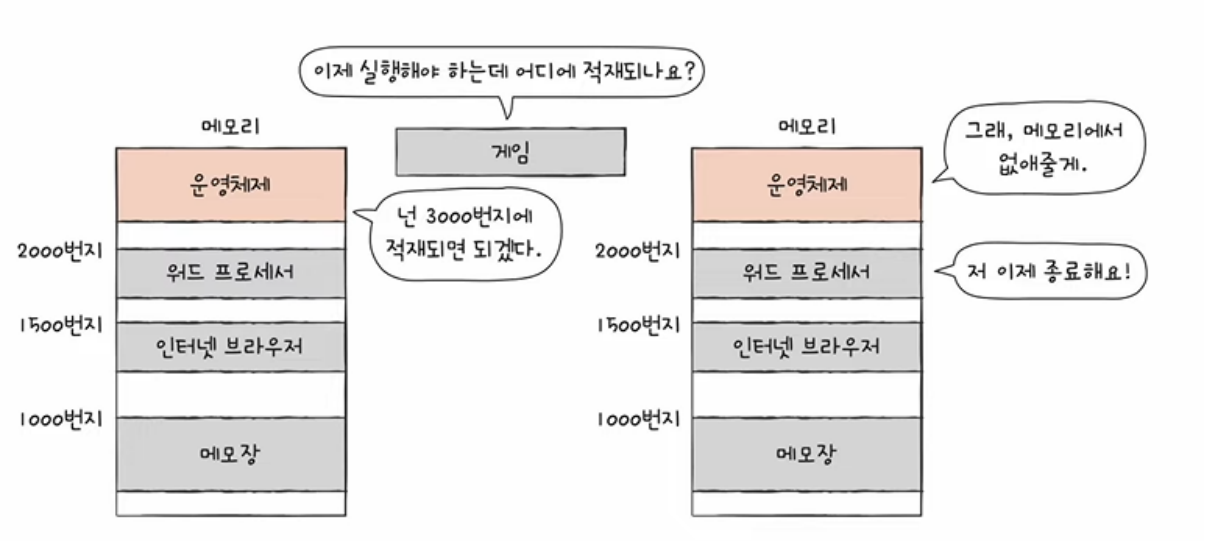
- **커널 영역 (kernel space)**
- **사용자 영역 (user space)** : 커널 영역을 제외한 나머지 영역, 사용자가 이용하는 응용 프로그램이 적재되는 영역
- 즉, 운영체제는 **커널 영역**에 적재되어 **사용자 영역**에 적재된 프로그램들에 자원을 할당하고 이들이 올바르게 실행되도록 돕는다.

- 일반적으로 메모리에는 여러 개의 응용 프로그램이 사용자 영역에 적재되어 실행된다.
- **응용 프로그램 (application software)** : 사용자가 특정 목적을 위해 사용하는 일반적인 프로그램 (메모장, 워드프로세서, 게임, 인터넷 브라우저 등)



- 누가 2000번지, 1500번지, 1000번지처럼 메모리 주소가 겹치지 않게 프로그램들을 적재해줬을까?

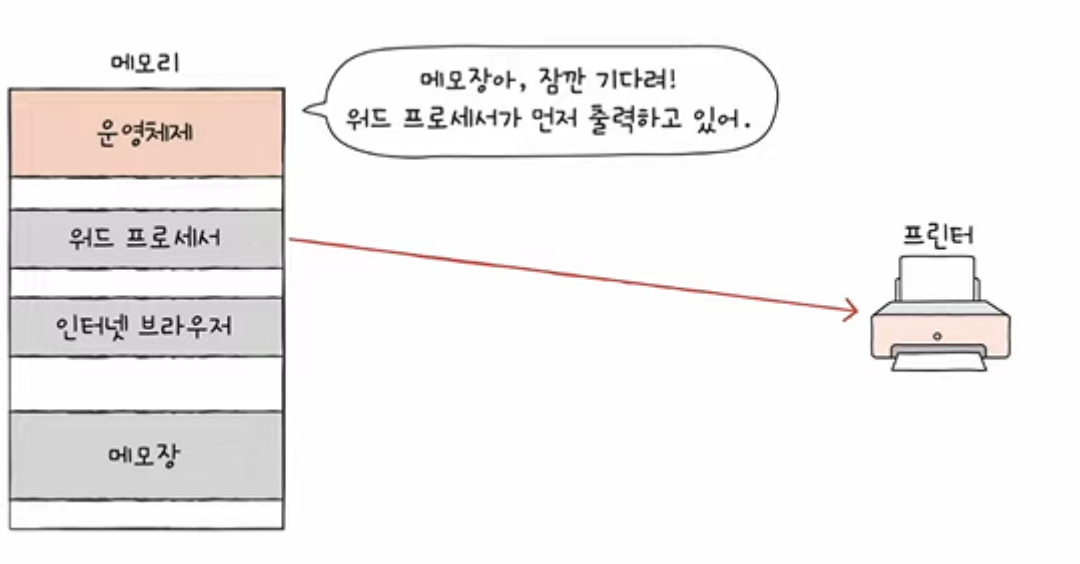
⇒ 운영체제



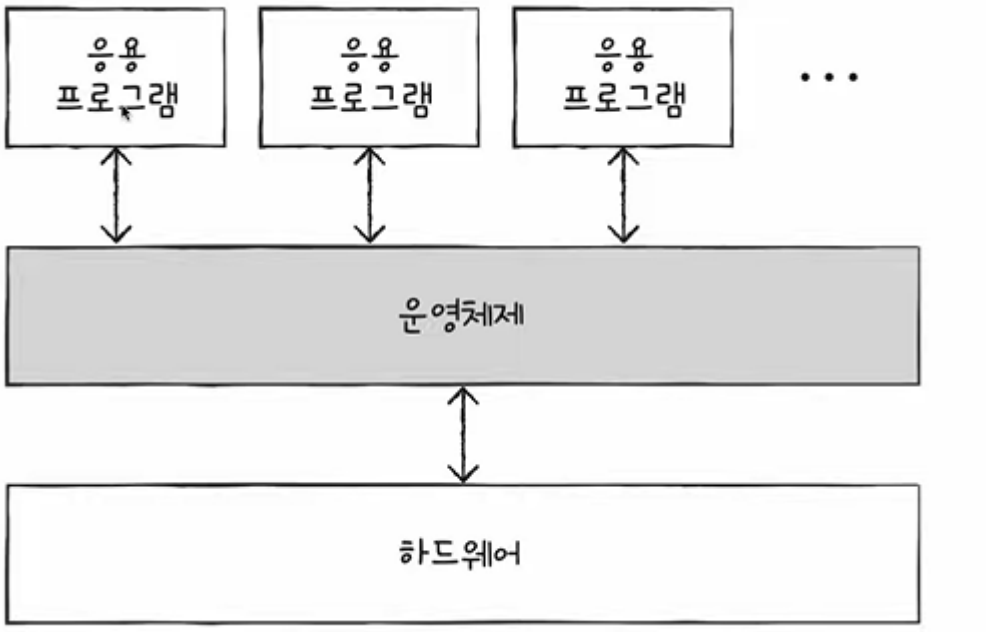
- 운영체제는 실행할 프로그램들을 메모리에 적재하고, 더 이상 실행되지 않는 프로그램을 메모리에서 삭제하며 지속적으로 메모리 자원을 관리한다.

- 세 개의 응용 프로그램이 실행되려면 반드시 CPU가 필요
- 어느 한 프로그램이 CPU를 독점하면 다른 프로그램들은 올바르게 실행될 수 없다
- 따라서 운영체제는 최대한 공정하게 여러 프로그램에 CPU 자원을 할당

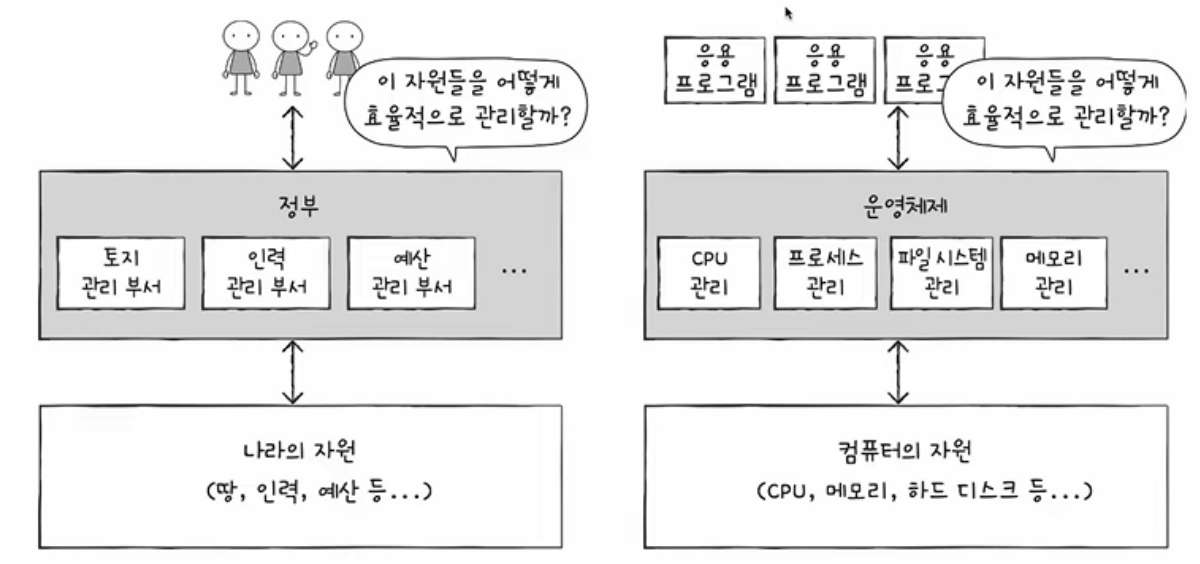
- 두 프로그램이 동시에 동일한 프린터를 이용하려는 상황



- 운영체제는 동시에 두 개의 프로그램이 프린터를 사용 못하도록 막고, 하나의 프로그램이 프린터를 이용하는 동안 다른 프로그램은 기다리게 만들어 프린터 자원을 관리



⇒ 이처럼 **운영체제**는 응용 프로그램과 하드웨어 사이에서 응용 프로그램에 필요한 자원을 할당하고, 응용 프로그램이 올바르게 실행되도록 관리하는 역할



- 이러한 운영체제의 역할은 정부의 역할과 비슷

◎ 운영체제를 알아야 하는 이유

- **운영체제**가 없다면?
- 아무리 간단한 프로그램이라도 하드웨어를 조작하는 코드를 개발자가 모두 직접 작성해야 함
- **운영체제** 덕분에 개발자는 하드웨어를 조작하는 코드를 직접 작성할 필요가 없음

1과 2를 더한 결과를 모니터에 출력하는 간단한 프로그램

> 프로그램을 메모리에 적재하는 코드

> CPU 로 하여금 1과 2를 더하게 하는 코드

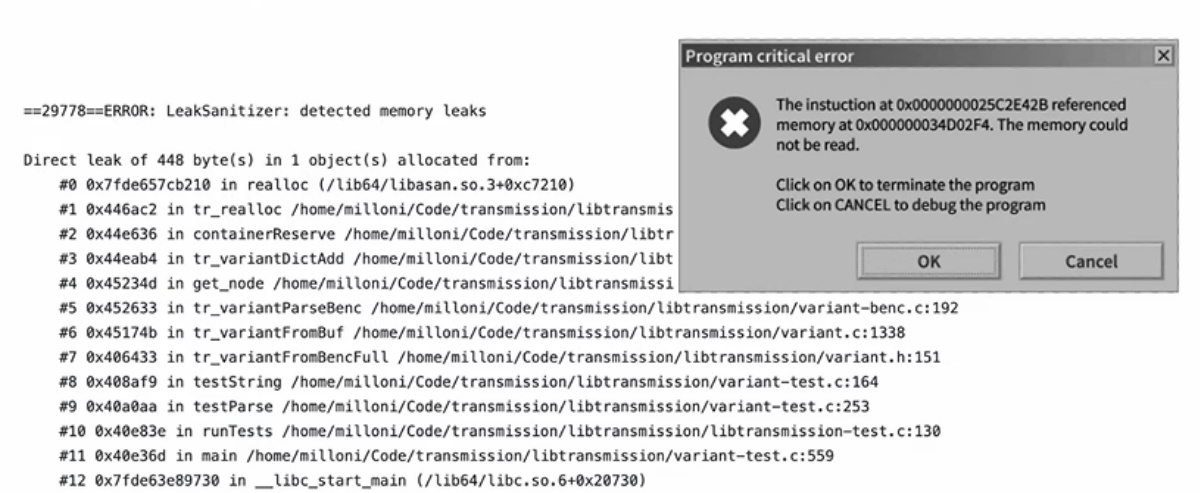
> 모니터에 계산 결과를 출력하는 코드

...

- 운영체제를 깊이 이해하면 운영체제가 우리에게 건네는 말을 제대로 이해할 수 있고, 운영체제에 제대로 명령할 수 있다.
- 결과적으로 하드웨어와 프로그램을 더 깊이 이해할 수 있다. (**문제 해결**의 실마리)

◎ 운영체제와의 대화

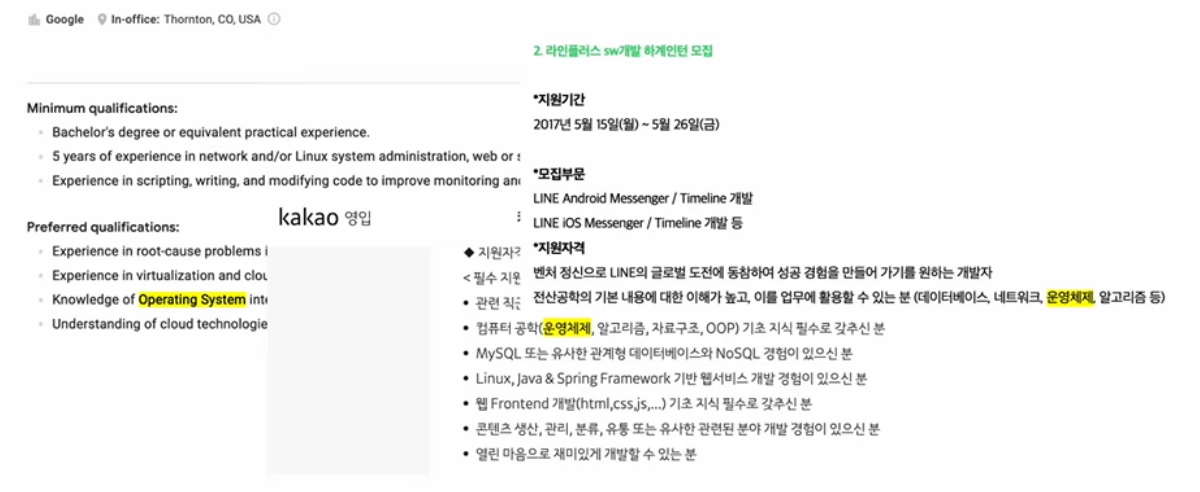
- 대표적인 운영체제와의 대화 → 오류 메시지



```
==29778==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 448 byte(s) in 1 object(s) allocated from:
    #0 0x7fde657cb210 in realloc (/lib64/libasan.so.3+0xc7210)
    #1 0x446ac2 in tr_realloc /home/milloni/Code/transmission/libtransmis
    #2 0x44e636 in containerReserve /home/milloni/Code/transmission/libtr
    #3 0x44eab4 in tr_variantDictAdd /home/milloni/Code/transmission/libt
    #4 0x45234d in get_node /home/milloni/Code/transmission/libtransmissi
    #5 0x452633 in tr_variantParseBenc /home/milloni/Code/transmission/libtransmission/variant-benc.c:192
    #6 0x45174b in tr_variantFromBuf /home/milloni/Code/transmission/libtransmission/variant.c:1338
    #7 0x406433 in tr_variantFromBencFull /home/milloni/Code/transmission/libtransmission/variant.h:151
    #8 0x408af9 in testString /home/milloni/Code/transmission/libtransmission/variant-test.c:164
    #9 0x40a0aa in testParse /home/milloni/Code/transmission/libtransmission/variant-test.c:253
    #10 0x40e83e in runTests /home/milloni/Code/transmission/libtransmission/libtransmission-test.c:130
    #11 0x40e36d in main /home/milloni/Code/transmission/libtransmission/variant-test.c:559
    #12 0x7fde63e89730 in __libc_start_main (/lib64/libc.so.6+0x20730)
```


- 작성한 소스 코드를 하드웨어가 제대로 실행하지 못하면 운영 체제는 우리에게 오류 메시지를 띄워준다.



- 다수의 기업에서 운영체제에 대한 이해를 필수적인 기초 역량으로 요구
- 채용 과정에서 기술 면접 등을 통해 검증

◆ 운영체제의 큰 그림

- 커널이란 무엇인지 이해
- 우리가 개발하고 실행하는 응용 프로그램이 어떻게 운영체제의 도움을 받으며 실행되는지 학습
- 운영체제가 응용 프로그램에 어떤 서비스를 제공하는지

◎ 배울 개념

- 커널
- 이중 모드
- 시스템 호출
- 운영체제의 서비스 종류

◎ 운영체제의 심장, 커널

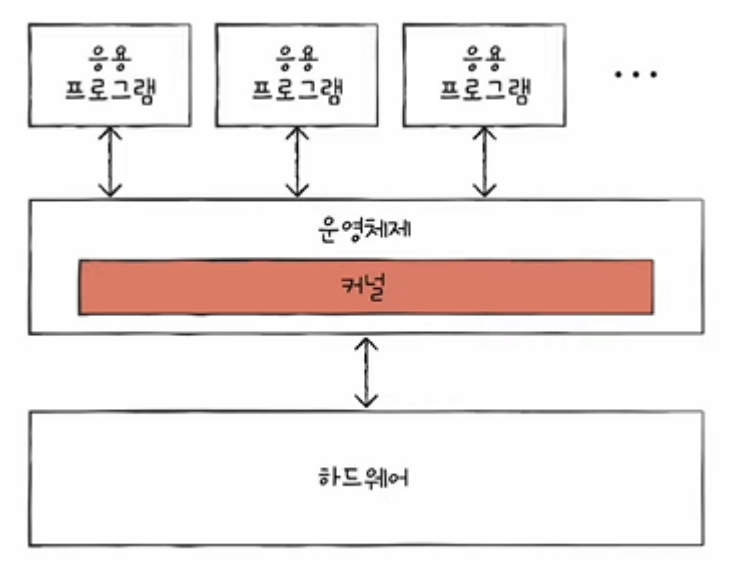
- 운영체제는 현존하는 프로그램 규모가 가장 큰 프로그램 중 하나
- 리눅스 2700만줄

Linux in 2020: 27.8 million lines of code in the kernel, 1.3 million in systemd

Language	Files	Lines
C	**928**	**492371**
C Header	636	58575

- 운영체제가 응용 프로그램에 제공하는 서비스 종류는 다양하지만, 그 중에서도 가장 핵심적인 서비스들이 있다.
- 자원에 접근하고 조작하는 기능, 프로그램이 올바르고 안전하게 실행되게 하는 기능 → 운영체제의 핵심 서비스에 속함

- 이러한 운영체제의 핵심 서비스를 담당하는 부분 → **커널**



- 커널은 마치 사람의 심장, 자동차의 엔진과 같음
- 어떤 커널을 사용하는 지에 따라 우리가 실행하고 개발하는 프로그램이 하드웨어를 이용하는 양상이 달라지고, 결과적으로 컴퓨터 전체의 성능도 달라질 수 있음

◎ 커널에 포함되지 않는 서비스

- 운영체제가 제공하는 서비스 중 커널에 포함되지 않는 서비스도 있다
- 대표적으로 **사용자 인터페이스 (UI : User Interface)**

⇒ 윈도우의 바탕화면과 같이 사용자가 컴퓨터와 상호작용할 수 있는 통로

◎ 사용자 인터페이스의 종류

- 2가지
- **그래픽 유저 인터페이스 (GUI : Graphical User Interface)** : 그래픽을 기반으로 컴퓨터와 상호작용할 수 있는 인터페이스 (윈도우-마우스 , 안드로이드-터치)

그래픽 유저 인터페이스

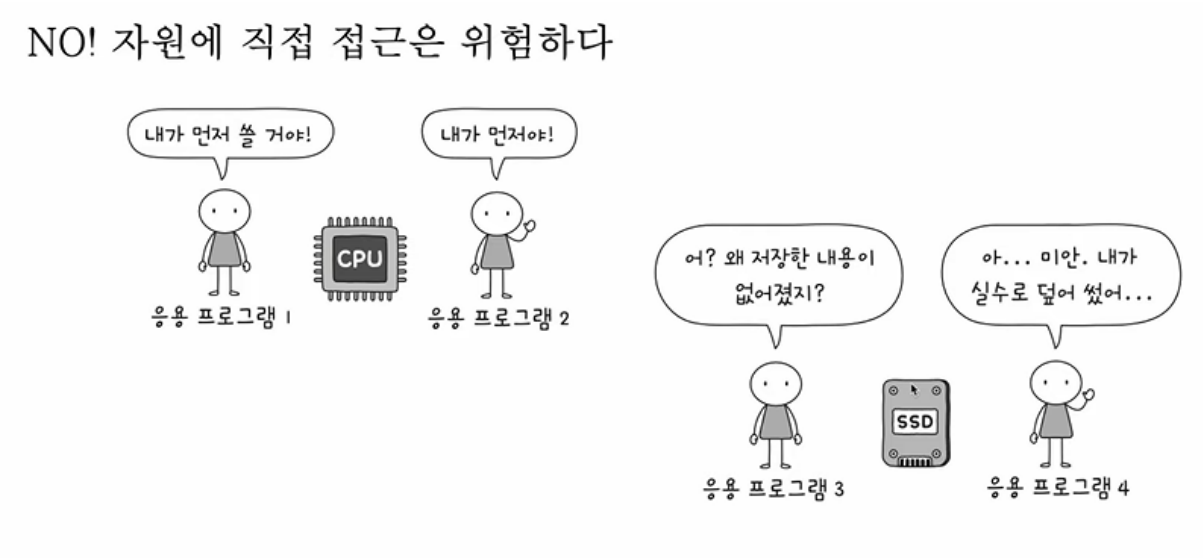
- **커맨드 라인 인터페이스 (CLI : Command Line Interface)** : 명령어를 기반으로 컴퓨터와 상호작용할 수 있는 인터페이스

커맨드 라인 인터페이스

```
minchul@minchul:~$ ll
total 92
drwxr-x--- 10 minchul minchul  4096 Mar 23 18:55 ./
drwxr-xr-x  3 root    root     4096 Feb 20 23:12 ../
-rw-------  1 minchul minchul  8902 Mar 22 00:42 .bash_history
-rw-r--r--  1 minchul minchul   220 Feb 20 23:08 .bash_logout
-rw-r--r--  1 minchul minchul  3796 Feb 21 02:38 .bashrc
drwx------ 10 minchul minchul  4096 Feb 20 23:47 .cache/
drwx------ 12 minchul minchul  4096 Mar  4 15:09 .config/
drwxr-xr-x  2 minchul minchul  4096 Feb 20 23:13 Desktop/
drwxr-xr-x  2 minchul minchul  4096 Feb 20 23:13 Documents/
drwxr-xr-x  2 minchul minchul  4096 Feb 20 23:13 Downloads/
drwx------  3 minchul minchul  4096 Feb 20 23:13 .local/
-rw-r--r--  1 minchul minchul   807 Feb 20 23:08 .profile
drwxr-xr-x  2 minchul minchul  4096 Feb 20 23:13 Public/
-rw-r--r--  1 minchul minchul     0 Feb 20 23:17 .sudo_as_admin_successful
-rw-------  1 minchul minchul 16765 Mar 20 01:30 .viminfo
-rw-rw-r--  1 minchul minchul    78 Feb 21 02:42 .vimrc
drwxrwxr-x  7 minchul minchul  4096 Mar 17 19:55 workspace/
-rw-------  1 minchul minchul    53 Mar 23 18:55 .Xauthority
minchul@minchul:~$
```

◎ 이중 모드와 시스템 호출

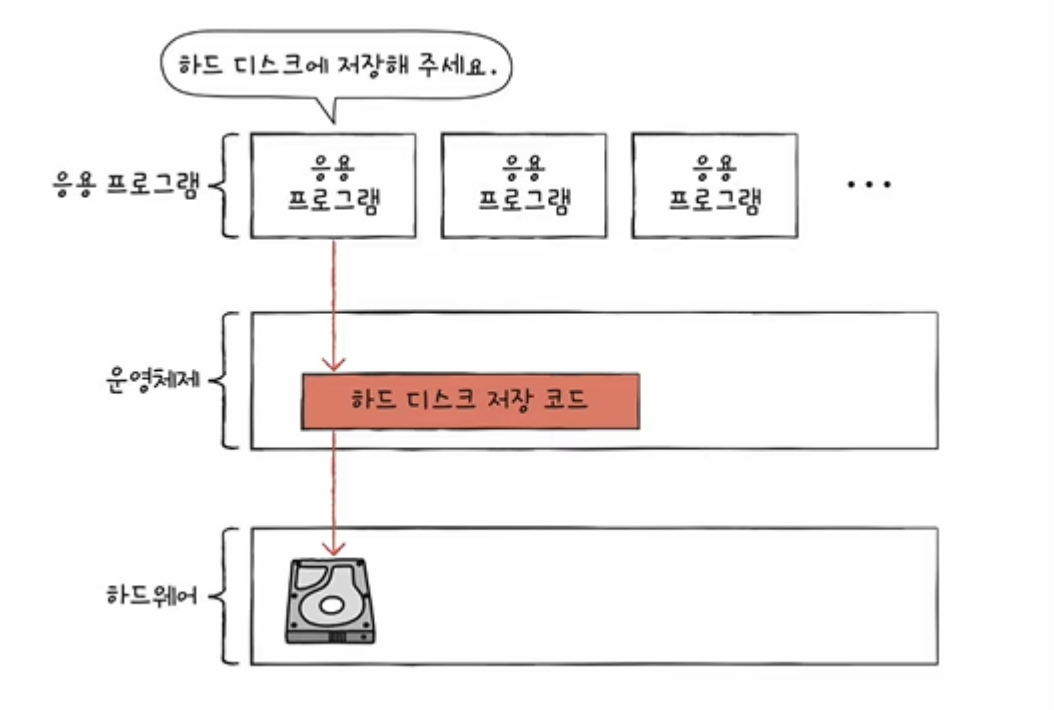
- 일반적인 응용 프로그램이 자원에 직접 접근하는 것은 위험하다!



- 따라서 **운영체제**는 사용자가 실행하는 응용 프로그램들이 하드웨어 자원에 직접 접근하는 것을 방지하여 자원을 보호
- 오직 자신을 통해서만 접근하도록 함 (일종의 문지기 역할)

- 응용 프로그램이 자원에 접근하기 위해서는 운영체제에 도움을 요청해야 함
- 운영체제의 코드를 실행

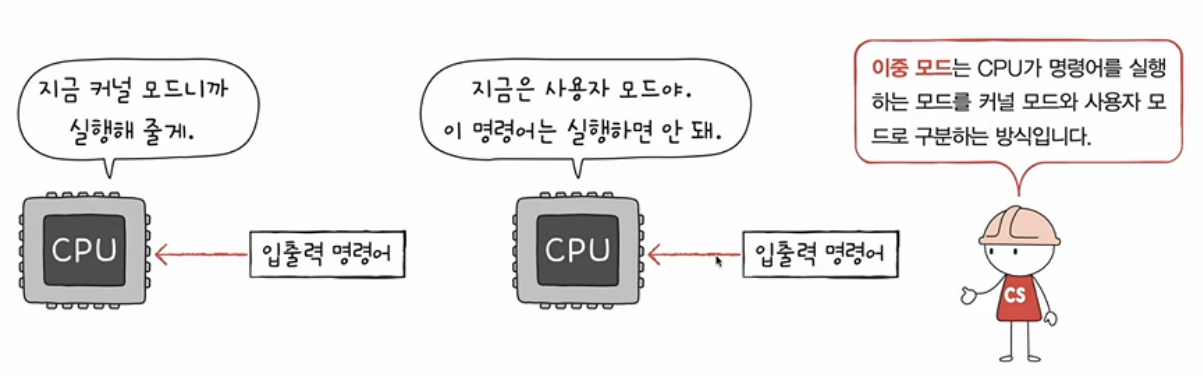
- 응용 프로그램이 실행 과정에서 하드 디스크에 접근하여 데이터 저장 시



- 운영체제에 도움을 요청
- 운영체제는 커널 영역 내의 하드 디스크에 데이터를 저장하는 코드를 실행하여 응용 프로그램의 작업을 대신 수행

◎ 이중 모드

- 운영체제의 문지기 역할은 이중 모드로써 구현
- **이중 모드** : CPU가 명령어를 실행하는 모드를 크게 **사용자 모드**와 **커널 모드**로 구분하는 방식



◎ **사용자 모드**

- 운영체제 서비스를 제공 받을 수 없는 실행 모드
- 커널 영역의 코드를 실행할 수 없는 실행 모드
- 일반적인 응용프로그램은 기본적으로 사용자 모드로 실행되기 때문에 자원에 접근 불가

◎ **커널 모드**

- 운영체제의 서비스를 제공 받을 수 있는 실행 모드
- 자원 접근을 비롯한 모든 명령어 실행 가능

플래그 종류	의미	사용 예시
부호 플래그	연산한 결과의 부호를 나타낸다.	부호 플래그가 1일 경우 계산 결과는 음수, 0일 경우 계산 결과는 양수를 의미한다.
제로 플래그	연산 결과가 0인지 여부를 나타낸다.	제로 플래그가 1일 경우 연산 결과는 0, 0일 경우 연산 결과는 0이 아님을 의미한다.
캐리 플래그	연산 결과 올림수나 빌림수가 발생했는지를 나타낸다.	캐리 플래그가 1일 경우 올림수나 빌림수가 발생했음을 의미하고, 0일 경우 발생하지 않았음을 의미한다.
오버플로우 플래그	오버플로우가 발생했는지를 나타낸다.	오버플로우 플래그가 1일 경우 오버플로우가 발생했음을 의미하고, 0일 경우 발생하지 않았음을 의미한다.
인터럽트 플래그	인터럽트가 가능한지를 나타낸다. 인터럽트는 04-3절에서 설명한다.	인터럽트 플래그가 1일 경우 인터럽트가 가능함을 의미하고, 0일 경우 인터럽트가 불가능함을 의미한다.
슈퍼바이저 플래그	커널 모드로 실행 중인지, 사용자 모드로 실행 중인지를 나타낸다. 커널 모드와 사용자 모드는 09장에서 설명한다.	슈퍼바이저 플래그가 1일 경우 커널 모드로 실행 중임을 의미하고, 0일 경우 사용자 모드로 실행 중임을 의미한다.

- CPU가 사용자 모드로 실행 중인지, 커널 모드로 실행 중인지는 플래그 레지스터 속 **슈퍼바이저 플래그**를 보면 알 수 있음

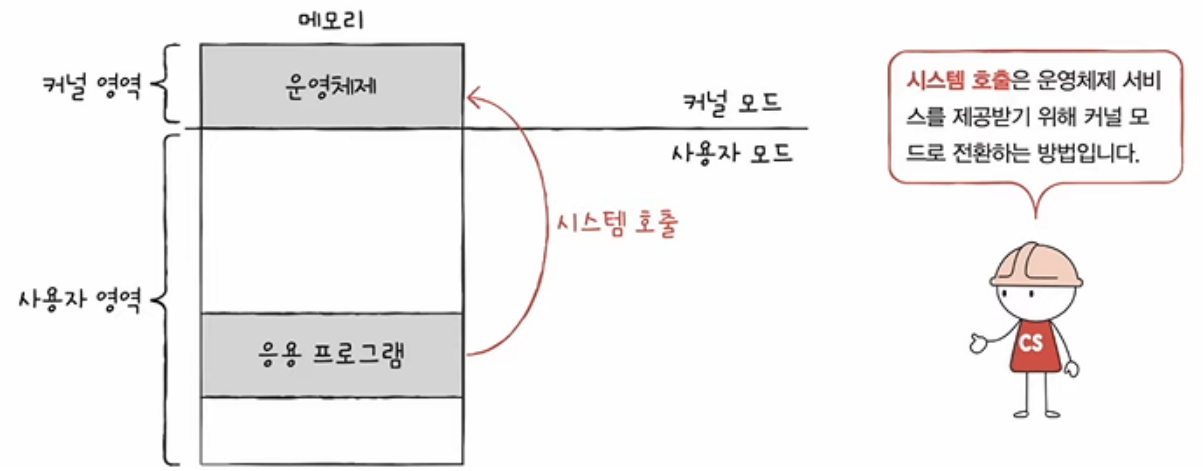
▼ **추가 개념? 가상머신**

- 이중 모드는 커널 모드와 사용자 모드, 두가지 모드를 지원하는 실행 모드

- 가상 머신을 통한 가상화를 지원하는 현대 CPU는 두 가지 모드 이상 지원
- **가상 머신** : 소프트웨어적으로 만들어낸 가상 컴퓨터, 설치하면 새로운 운영 체제와 응용 프로그램을 설치하고 실행 가능
- 윈도우 운영체제에 **가상 머신**을 설치하면 가상 머신 상에 리눅스 운영체제와 그 기반의 응용 프로그램 설치 가능
- 가상화를 지원하는 CPU - **커널 모드**, **사용자 모드**, 가상 머신 위한 모드인 **하이퍼바이저 모드**
- 가상 머신 상에서 작동하는 응용 프로그램들은 **하이퍼바이저 모드**로써 가상 머신에 설치된 운영체제로부터 운영체제 서비스를 받음

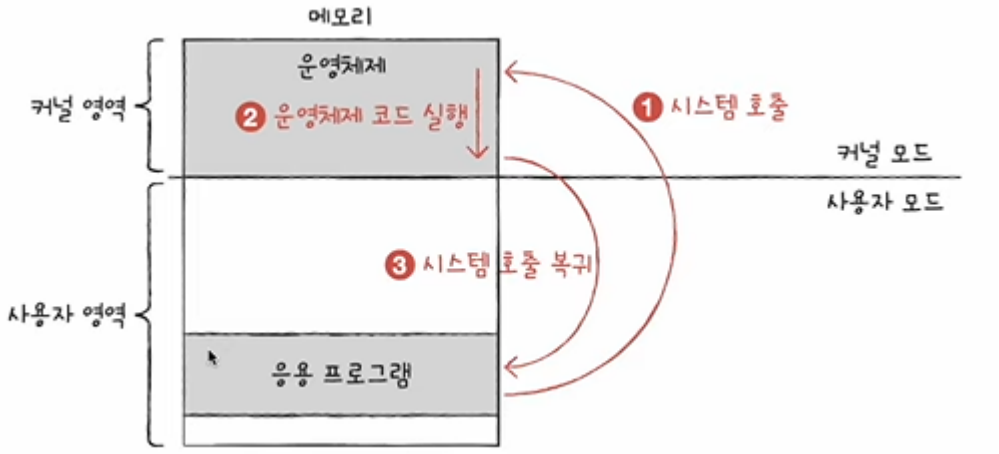
◎ 시스템 호출

- 사용자 모드로 실행되는 프로그램이 자원에 접근하는 운영체제 서비스를 제공 받으려면 운영체제에 **요청**을 보내 커널모드로 전환되어야 함 → 이 요청이 **시스템 호출**
- 커널 모드로 전환하여 실행하기 위해 호출
- 일종의 **소프트웨어 인터럽트**



- 인터럽트는 입출력장치에 의해 발생하기도 하지만,
- 인터럽트를 발생시키는 특정 명령어에 의해 발생하기도 함 → **소프트웨어 인터럽트**
- 04장

- 시스템 호출이 처리되는 방식은 하드웨어 인터럽트 처리 방식과 유사
- 시스템 호출 발생시키는 명령어 발생 → 지금까지의 작업을 백업 → 커널 영역 내의 시스템 호출을 수행하는 코드 (인터럽트 서비스 루틴)를 실행 → 기존 실행하던 응용 프로그램으로 복귀하여 실행을 계속해 나감



- 일반적으로 응용 프로그램은 실행 과정에서 운영체제 서비스들을 매우 빈번하게 이용

- 그 과정에서 빈번하게 시스템 호출을 발생 시키고, 사용자 모드와 커널 모드를 오가며 실행

- 시스템 호출은 운영체제마다 정해져 있음

종류	시스템 호출	설명
프로세스 관리	fork()	새 자식 프로세스 생성
	execve()	프로세스 실행(메모리 공간을 새로운 프로그램의 내용으로 덮어씌움)
	exit()	프로세스 종료
	waitpid()	자식 프로세스가 종료할 때까지 대기
파일 관리	open()	파일 열기
	clase()	파일 닫기
	read()	파일 읽기
	write()	파일 쓰기
	stat()	파일 정보 획득
디렉터리 관리	chdir()	작업 디렉터리 변경
	mkdir()	디렉터리 생성
	rmdir()	비어 있는 디렉터리 삭제
파일 시스템 관리	mount()	파일 시스템 마운트
	umount()	파일 시스템 마운트 해제

- 리눅스의 시스템 콜

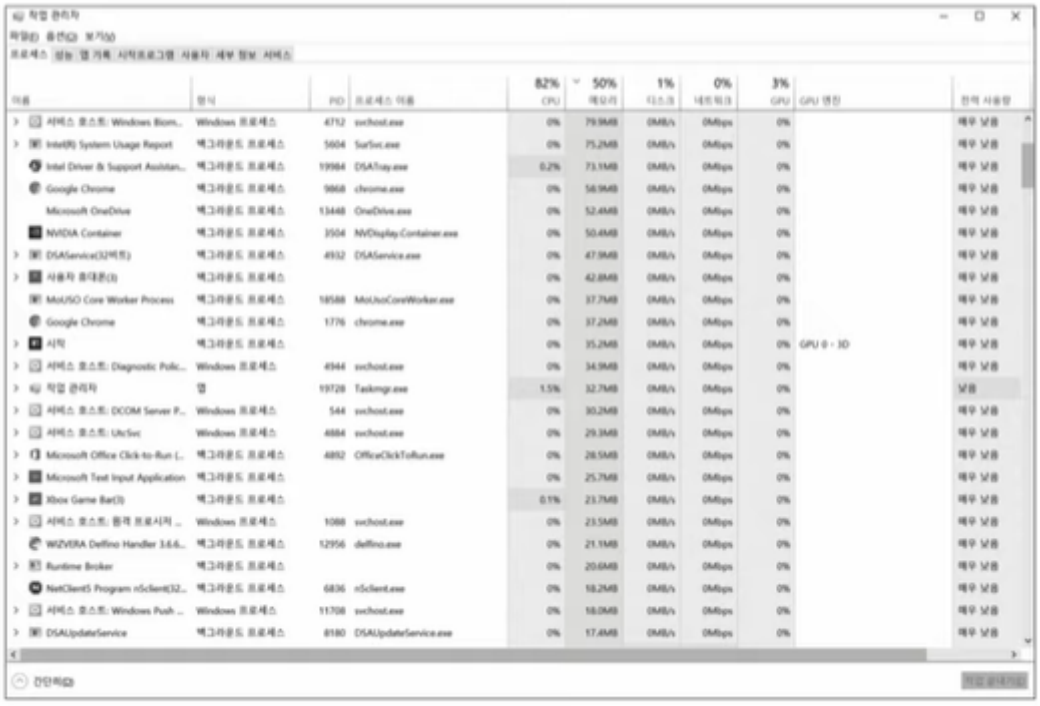
https://github.com/kangtegong/self-learning-cs/tree/main/system_calls

◎ 운영체제의 핵심 서비스

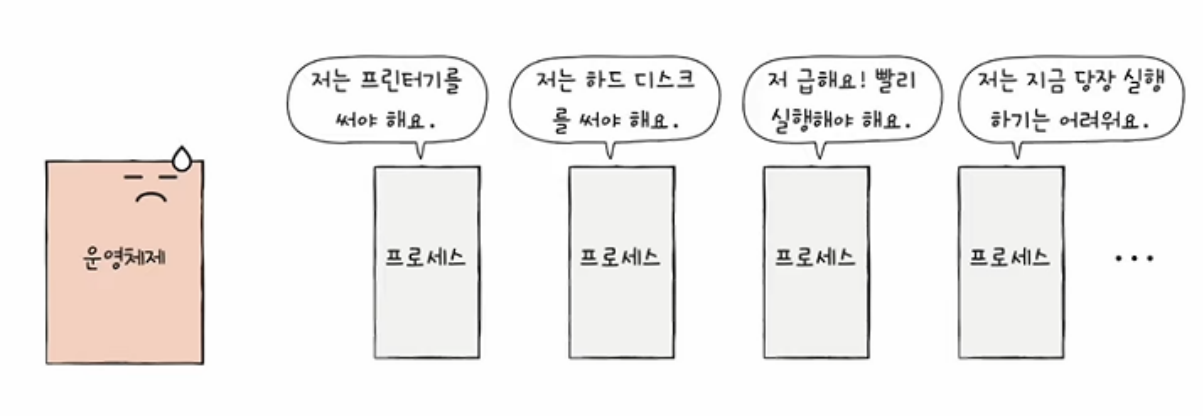
- 프로세스 관리
- 자원 접근 및 할당
- 파일 시스템 관리

◎ 운영체제의 핵심 서비스 1 - 프로세스 관리

- 프로세스 : 실행 중인 프로그램



- 컴퓨터를 사용하는 동안 메모리 안에서는 새로운 프로세스들이 마구 생성되고, 사용되지 않는 프로세스는 메모리에서 삭제
- 운영체제의 어떤 기법을 통해서 모든 프로세스가 굳이 메모리에 올라 와있지 않을 수 있다. ⇒ 추후에 배울 **페이징**, **스와핑** (14장)
- 일반적으로 하나의 CPU는 한 번에 하나의 프로세스만 실행
- 따라서 CPU는 이 프로세스를 조금씩 번갈아 가며 실행 (전환)



- 각 프로세스는 상태도, 사용하고자 하는 자원도 다양
- 이처럼 동시 다발적으로 생성/실행/삭제 되는 다양한 프로세스를 일목요연하게 관리

- 추가로, 여러 프로세스가 동시에 실행되는 환경에서는 '프로세스 동기화'가 필수적
- 프로세스가 꼼짝도 못하고 더 이상 실행되지 못하는 상황인 '교착 상태' 해결해야 함

◎ 운영체제의 핵심 서비스 2 - 자원 접근 및 할당

- 운영체제가 컴퓨터의 네 가지 핵심 부품(CPU, 메모리, 보조기억장치, 입출력장치)를 어떻게 관리하고, 결과적으로 어떤 기능을 제공하는지

◎ CPU

- 메모리에는 여러 프로세스가 적재
- 하나의 CPU는 한 번에 하나의 프로세스만 실행
- 다른 프로세스는 기다려야 함
- 프로세스들에 공정하게 CPU를 할당
- **CPU 스케줄링** : 어떤 프로세스를 먼저, 얼마나 오래 실행할까? ⇒ 11장

▼ CPU 스케줄링?

- 운영체제가 프로세스들에게 공정하고 합리적으로 CPU 자원을 배분하는 것
- 운영체제는 프로세스들에 '줄을 서서 기다릴 것'을 요구 ⇒ 이 줄을 **스케줄링 큐**를 통해 구현
- **선점형 스케줄링** : 강제로 빼앗기

- **비선점형 스케줄링** : 하나의 프로세스가 자원 사용 독점, 기다리기

◎ **메모리**

- 같은 프로세스라도 실행할 때마다 적재되는 주소가 달라질 수 있다.
- 운영체제는 새로운 프로세스가 적재될 때마다 어느 주소에 적재해야 할지 결정
- 운영체제가 프로세스에게 어떻게 메모리를 할당하는지, 그리고 메모리가 부족할 경우 어떻게 극복하는지 ⇒ 14장 (페이징, 스와핑)

▼ **페이징? 스와핑?**

- 대기상태인 프로세스나, 오랫동안 사용되지 않은 프로세스들을 임시로 보조기억장치 일부 영역으로 쫓아내고, 그 메모리 상의 빈 공간에 또 다른 프로세스를 적재하여 실행하는 방식 ⇒ **스와핑**
- 프로세스의 논리 주소 공간을 **페이지**라는 일정한 단위로 자르고, 메모리 물리 주소 공간을 **프레임**이라는 페이지와 동일한 크기의 일정한 단위로 자른 뒤 페이지를 프레임에 할당하는 가상 메모리 관리 기법 ⇒ **페이징**

◎ **입출력 장치**

- 인터럽트를 처리하는 프로그램 (인터럽트 서비스 루틴)을 제공함으로써 입출력 작업을 수행
- 입출력 장치가 CPU에 하드웨어 인터럽트 요청 신호 보냄 → CPU 작업 백업 → 커널 영역에 있는 인터럽트 서비스 루틴 실행

◎ 운영체제의 핵심 서비스 3 - 파일 시스템 관리

- 파일 시스템도 운영체제가 지원하는 핵심 서비스
- 파일 열기, 생성, 삭제

- 관련된 정보를 파일이라는 단위로 저장 장치에 보관
- 파일들을 묶어 폴더(디렉터리) 단위로 저장 장치에 보관

- 운영체제가 보조기억장치 속 데이터를 어떻게 파일과 디렉터리로 관리하는지? ⇒ 15장

◆ 정리

- **커널** : 운영체제의 핵심 서비스를 제공하는 부분
- **이중모드** : CPU가 명령어를 실행하는 모드를 커널 모드와 사용자 모드로 구분하는 방식
- 사용자 프로세스가 커널의 서비스를 제공받기 위해서는 사용자 모드에서 커널 모드로 전환
- 이는 **시스템 호출**을 통해 이루어진다.
- **시스템 호출** : 커널 모드로써 운영체제의 서비스를 제공 받을 수 있는 방법
- **대표적인 커널(운영체제)의 서비스** : 프로세스 관리, 자원 접근 및 할당, 파일 시스템 관리