😀

챕터 09. 운영체제 시작하기

```
// 챕터 목표
운영체제의 이해
개발자가 운영체제를 알아야 하는 이유
커널이란 무엇인가
시스템 호출과 이중모드
운영체제가 제공하는 핵심 서비스의 종류
```

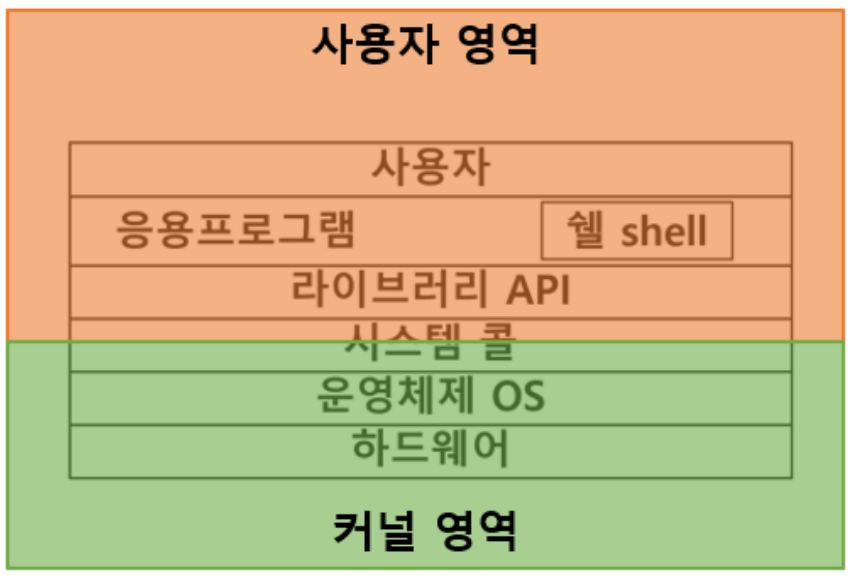
09-1. 운영체제를 알아야 하는 이유

운영체제란?

- 컴퓨터 프로그램의 실행에 마땅히 필요한 요소들을 가리켜 **시스템 자원** 혹은 **자원**이라고 한다.
- 모든 프로그램이 실행되기 위해서는 반드시 자원이 필요.

→ 실행할 프로그램에 필요한 자원을 할당하고,
프로그램이 올바르게 실행하도록 돕는 특별한 프로그램이 바로 운영체제.

커널 영역과 사용자 영역



- 운영체제도 프로그램이기 때문에, 메모리에 적재되어야 한다.
  - 단, 특별한 프로그램이기에, 항상 부팅될 때 **메모리 내의 커널 영역**에 따로 적재되어 실행됨.
  - 이때, 커널 영역을 제외한, 사용자가 이용하는 나머지 영역은 **사용자 영역**이라고 함.

→ 운영체제는 메모리의 커널 영역에 적재되는 프로그램.

운영체제의 역할

> **응용 프로그램**은, 사용자가 특정 목적을 위해 사용하는 일반적인 프로그램.
>
> - ex) 워드 프로세서, 인터넷 브라우저, 메모장, 게임 등

1. **메모리 자원 관리**

→ **운영체제**는 응용 프로그램을 실행하기 위해 메모리에 적재하고,
더 이상 실행되지 않으면 메모리에서 삭제하는 등의 **메모리 자원 관리 역할**을 수행.

2. **CPU 자원 할당**
   - 응용 프로그램이 실행되기 위해선 반드시 CPU가 필요하다. 어떤 프로그램부터 CPU를 사용하게 할지, 얼마나 오래 사용하게 할지 등을 결정해야 함.

   → **운영체제**는 응용 프로그램에 대해 최대한 공정하게 **CPU 자원을 할당**.

3. **기타 자원 관리**
   - 두 개의 프로그램이 동시에 특정 하드웨어를 이용하려 할 때, 동시에 사용하지 못하도록 막고, 하나의 프로그램이 사용하고 있으면 다른 프로그램은 대기하도록 해야 함.

   → **운영체제**는 프로그램과 자원이 **올바르게 실행되도록 관리**.

→ **운영체제**는 응용 프로그램에 자원을 효율적으로 배분하고, 실행할 프로그램들이 지켜야 할 규칙을 만들어 **컴퓨터 시스템 전체를 관리하는 역할**을 수행.

운영체제를 알아야 하는 이유?

만약, 운영체제가 없다면?

- 간단한 프로그램이라도 운영체제가 없다면 하드웨어를 조작하는 코드를 개발자가 모두 직접 작성해야 함.
- **운영체제**가 있기에, 개발자는 **하드웨어를 조작하는 코드**를 직접 작성할 필요가 없음.

운영체제를 알아야 하는 이유

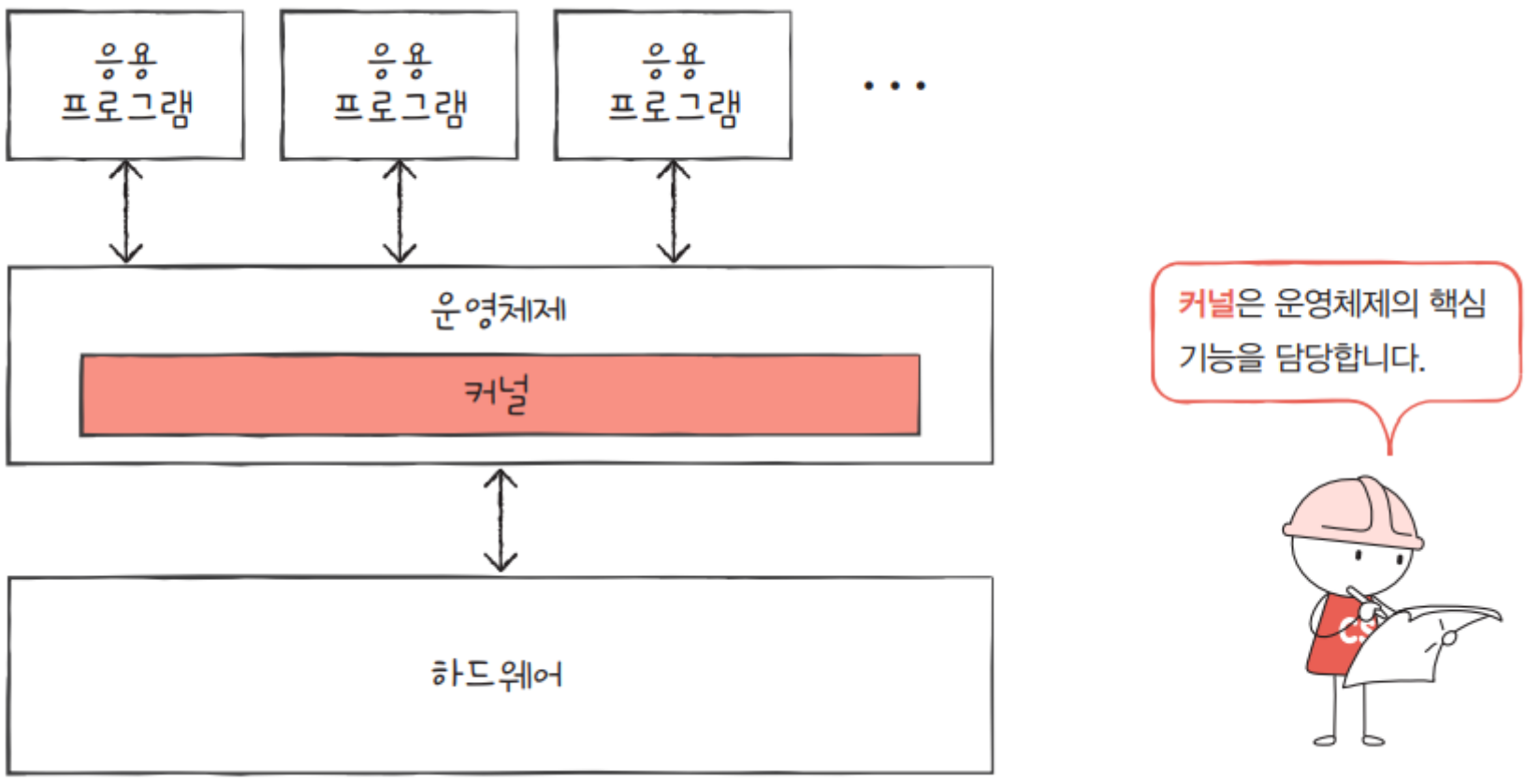
- 운영체제를 깊게 이해한다면,
  하드웨어, 프로그램의 동작 원리와 과정을 이해할 수 있어서, 문제 해결의 실마리를 찾을 수 있음.
- 대표적인 예시
  1. **에러 메시지를 통한 문제 원인 추적**
     - 대다수의 오류 메시지의 근원은 운영체제.
     - 우리가 작성한 소스 코드를 하드웨어가 제대로 실행하지 못할 때, 에러 메시지를 띄움.
     - ex) 메모리 누수

```
==2074307==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 16 byte(s) in 1 object(s) allocated from:
    #0 0x7fd478782bc8 in malloc (/lib/x86_64-linux-gnu/libasan.so.5+0x10dbc8)
    #1 0x55af57a8f3e6 in push ../../src/balance/balance.c:15
    #2 0x55af57a8fb13 in areBracketsBalanced ../../src/balance/balance.c:64
    #3 0x55af57a90210 in main ../../src/balance/balance.c:110
    #4 0x7fd477b3d0b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)
```

09-2. 운영체제의 큰 그림

운영체제의 심장, 커널



→ **커널**은 자원 접근 및 조작 / 프로그램이 올바르고 안전하게 실행되게 하는 기능과 같은
운영체제의 핵심 서비스를 담당.

- 운영체제가 설치된 모든 기기에는 **커널**이 있음.
    - 어떤 커널을 사용하는지에 따라서,
      프로그램이 **하드웨어를 이용하는 양상**이 달라지고
      결과적으로 **컴퓨터 성능**도 달라질 수 있음.

사용자 인터페이스

- **사용자 인터페이스**는 윈도우의 바탕화면과 같이 **사용자가 컴퓨터와 상호 작용하는 통로**.
- 운영체제가 제공하는 서비스 중, **커널에 포함되지 않는** 서비스 중 대표적인 것이,
  사용자 인터페이스 (UI)
- **종류**
    - 그래픽 유저 인터페이스 (GUI) : 그래픽 기반
    - 커맨드 라인 인터페이스 (CLI) : 명령어 기반

이중 모드와 시스템 호출

- 운영체제는 사용자가 실행하는 응용프로그램이 하드웨어 자원에 직접 접근하는 것을 방지하여 자원을 보호.
- 응용프로그램은 하드웨어 자원(하드디스크, 마우스, 모니터 등)에 접근하려면,
  반드시 운영체제를 거쳐야 함.

→ 이러한 기능은 **이중 모드**로써 구현됨.

이중 모드?

- CPU가 실행하는 모드는 크게 **사용자 모드**와 **커널 모드**로 구분하는 방식.

→ **CPU**는 **사용자 모드 또는 커널 모드**로 명령어를 실행할 수 있음.

사용자 모드?

- 운영체제 서비스를 제공 받을 수 없는 실행 모드.
  즉, **커널 영역의 코드를 실행할 수 없는 모드.**

→ **일반적인 응용 프로그램**은 **기본적으로 사용자 모드**로 실행되기 때문에,
하드웨어 자원에 접근하는 명령어 사용 불가

커널 모드?

- 운영체제 서비스를 제공받을 수 있는 실행 모드.
  즉, **커널 영역의 코드를 실행할 수 있는 모드.**

→ **CPU가 커널 모드로 명령어를 실행**하면 자원에 접근하는 명령어를 비롯한 모든 명령어 실행 가능하기 때문에, **하드웨어 자원에 접근하는 명령어 사용 가능**.

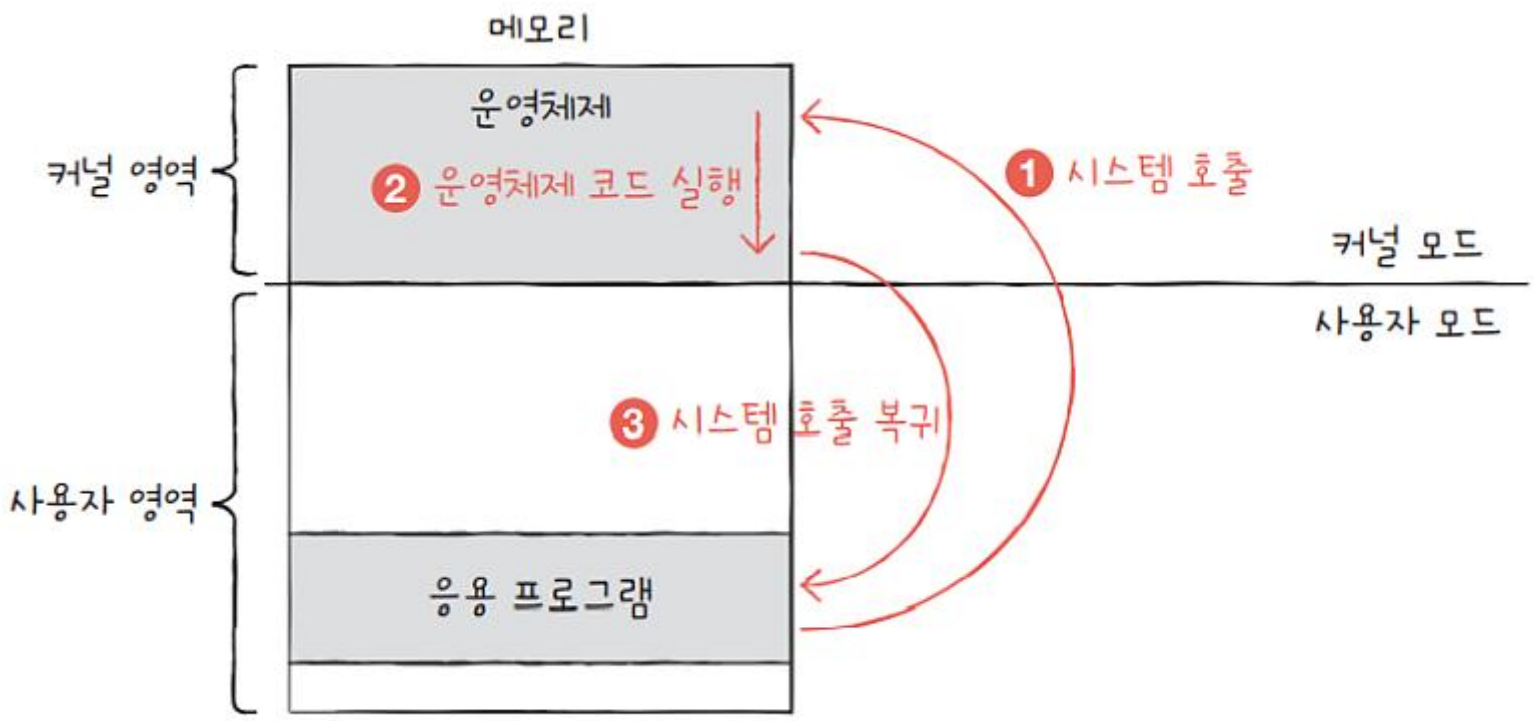
시스템 호출 (시스템 콜)?

- 사용자 모드로 실행되는 프로그램이 자원에 접근하는 운영체제 서비스를 제공 받으려면,
  운영체제에 요청을 보내 커널 모드로 전환되어야 함.

→ 이때의 요청을 **시스템 호출(시스템 콜)** 이라고 함.

시스템 호출 처리 순서.

- 일반적으로 응용 프로그램은 실행 과정에서 운영체제 서비스들을 매우 빈번하게 이용하고,
  그 과정에서 빈번하게 시스템 호출을 발생시키며, 사용자 모드와 커널 모드를 오가며 실행.



1. 시스템 호출을 발생시키는 명령어를 실행하고, CPU는 지금까지의 작업을 백업.
2. 커널 영역 내 시스템 호출을 수행하는 코드를 실행.
3. 기존에 실행하던 응용프로그램으로 복귀하여 실행을 계속함.

운영체제의 핵심 서비스

- 프로세스 관리
- 자원 접근 및 할당
- 파일 시스템 관리

프로세스 관리

- 프로세스란? ⇒ 실행 중인 프로그램
- 일반적으로 하나의 CPU는 한 번에 하나의 프로세스만 실행 가능.
- CPU는 한 프로세스를 실행하다가 다른 프로세스로 실행을 전환하고,
  이후 또 다른 프로세스로 전환 후 실행을 반복하며 동작함.

→ **운영체제는 이러한 프로세스가 효율적으로 실행되도록 관리.**

- 여러 프로세스가 동시에 실행되는 환경에서 '**프로세스 동기화**' 수행.
- 프로세스가 실행되지 못하는 현상인 '**교착 상태**'를 해결.
- (후에 12장, 13장에서 자세히 다룸)

자원 접근 및 할당

- 컴퓨터의 네 가지 핵심 부품 (자원)
  - CPU, 메모리, 보조기억장치, 입출력 장치

→ **운영체제는 프로세스들이 사용할 자원에 접근하고 조작하면서, 프로세스에 필요한 자원을 할당.**

- 프로세스들이 공정하게 CPU를 할당받기 위해서 어떤 프로세스부터, 얼마나 오래 CPU를 이용하게 할 것인지 결정하는 '**CPU 스케줄링' 수행.**
- 프로세스에게 **메모리를 할당하는 방식을 결정**하고, '**메모리 부족 현상'**을 해결.
- '**인터럽트 서비스 루틴'**을 제공.
  - 하드웨어 인터럽트란? ⇒ 하드웨어의 전기 신호 등으로 인해, CPU가 하던 일을 멈추고 다른 일을 처리하는 것.
- (후에 11장, 14장에서 자세히 다룸)

파일 시스템 관리

→ 운영체제는 시스템의 파일의 생성, 수정, 삭제 및 디렉터리 관리 등을 수행.

- (후에 15장에서 자세히 다룸)

부록

- 가상 머신을 통한 가상화를 지원하는 현대 CPU는 두 가지 모드 이상을 지원.

가상 머신

- 소프트웨어 적으로 만들어낸 가상 컴퓨터
- 이러한 가상 머신을 실행 시키는 프로그램 또한 응용 프로그램 ⇒ 사용자 모드로 작동
  → 어? 커널 모드로 전환되어야 하는데?
- 그래서, 가상화를 지원하는 CPU는
  커널 모드, 사용자 모드 외에도 가상머신을 위한 '**하이퍼 바이저 모드**'를 따로 둠.

→ 이러한 하이퍼 바이저 모드를 통해,
가상 머신에 설치된 운영체제로부터 운영체제 서비스를 받을 수 있음.

대표적인 시스템 호출의 종류

프로세스 관리

- fork() - 새 자식 프로세스 생성
- execve() - 프로세스 실행 (메모리 공간을 새로운 프로그램의 내용으로 덮어쓰기)
- exit() - 프로세스 종료
- waitpid() - 자식 프로세스가 종료할 때까지 대기

파일 관리

- open() - 파일 열기
- close() - 파일 닫기
- read() - 파일 읽기
- write() - 파일 쓰기
- stat() - 파일 정보 획득

디렉터리 관리

- chdir() - 작업 디렉터리 변경
- mkdir() - 디렉터리 생성
- rmdir() - 비어있는 디렉터리 삭제

파일 시스템 관리

- mount() - 파일 시스템 마운트
- umount() - 파일 시스템 마운트 해제.

‘리눅스 시스템 호출의 종류’ 참고 → https://github.com/kangtegong/self-learning-cs