1. 디바이스 드라이버 개발 시 커널 빌드를 하는 이유
   1. 디바이스 드라이버는 하드웨어에 접근하기 위해 커널 모드에서 실행되어야 함.
2. Virtual memory, Physical memory, MMU:
   1. 가상 메모리: 컴퓨터의 메모리보다 더 큰 프로그램을 돌리기 위한 방법. 페이징으로 프로그램을 쪼개서 필요할 때마다 메모리에 적재하고 빼냄.
   2. 물리 메모리: 실제로 컴퓨터에 설치된 메모리.
   3. MMU(Memory Management Unit): 가상 메모리 주소를 물리적 주소로 변환하는 하드웨어. 이를 통해 개발자는 프로그램을 메모리에 적재할 때 주소를 신경쓰지 않아도 됨(옛날엔 했다고 함)
3. Kernel Space, User Space
   1. 커널 영역: 커널이 실행되는 메모리 영역. 하드웨어 자원에 직접 접근할 수 있음. 유저 영역과 격리되어 있음.
   2. 유저 영역: 사용자 애플리케이션이 실행되는 메모리 영역. 커널의 보호를 받아서 커널 공간이나 디바이스에 접근하려면 시스템 콜이 필요함.
4. 디바이스 드라이버와 커널 모듈의 차이점
   1. 디바이스 드라이버는 하드웨어와 운영체제의 인터페이스를 제공, 리눅스 시스템에서 커널 모듈 형태로 구현됨
   2. 커널 모듈은 커널 기능 확장을 위해 동적으로 로드되는 코드. 드라이버 말고도 다양한 기능 구현 가능
   3. 모든 디바이스 드라이버는 커널 모듈로 구현될 수 있지만, 커널 모듈이 모두 드라이버는 아님.
5. Monolithic vs Microkernel
   1. 모놀리식: 모든 운영체제 서비스가 커널 모드에서 실행됨. 성능이 빠르지만 안정성이 떨어질 수 있음. 디바이스 드라이버를 추가/삭제하려면 커널 재빌드해야 함. 라즈베리파이는 모놀리식이지만 커널 모듈을 사용했기 때문에 빌드 다시 안해도 된 것
   2. 마이크로커널: 최소한의 커널 기능만 포함하고, 나머지 서비스는 사용자 공간에서 서버 형태로 존재. 안정성이 높지만 성능이 떨어짐.