Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 전문가 과정

<리눅스 시스템 프로그래밍, 네트워크, 커널>

2018.04.17 - 2 개월차 시험

강사 - 이상훈
gcccompil3r@gmail.com

학생 - 안상재
sangjae2015@naver.com

<서술형 문제>

3. 리눅스 커널은 운영체제(OS)다.
OS 가 관리해야 하는 제일 중요한 5 가지에 대해 기술하시오.

A : 파일 시스템관리자, 메모리 관리자, 태스크 관리자, 네트워크 관리자, 디바이스 관리자

4. Unix 계열의 모든 OS 는 모든 것을 무엇으로 관리하는가 ?
A : 파일

5. 리눅스에는 여러 장점이 있다.(배점 0.2 점)
아래의 장점들 각각에 대해 기술하라.

* 사용자 임의대로 재구성이 가능하다.

→ 커널 소스코드를 수정해서 커스터마이징이 가능하다.

* 열약한 환경에서도 HW 자원을 적절히 활용하여 동작한다.

→ 리눅스는 최적화가 잘 되어있고, 모놀리식 구조를 기반으로 하고 디바이스 드라이버는 마이크로식으로 탈부착이 가능하다.

* 커널의 크기가 작다.

→ 불필요한 부분을 제외하고 필수적인 부분으로만 커널을 구성한다. 윈도우보다 훨씬 크기가 작다.

* 완벽한 멀티유저, 멀티태스킹 시스템

→ 여러 사용자들을 위해 권한을 설정해 놓았고, 여러 프로세스들을 스케줄링을 통해 멀티태스킹을 지원한다.

* 뛰어난 안정성

→ 안정성이 뛰어나므로 고성능의 네트워크 장비에 들어간다.

* 빠른 업그레이드

→ 전세계의 프로그래머들로부터 소스코드가 공유되고 수정이되므로 업그레이드가 빠르다

* 강력한 네트워크 지원

→ TCP/IP 에 강한 스택이 있다.

* 풍부한 소프트웨어

→ 스톨만의 GNU 재단에서 free 로 풀어놔서 소프트웨어가 풍부하다.

6. 32 bit System 에서 User 와 Kernel 의 Space 구간을 적으시오.

A : 프로세스 하나당 전체 4GB 의 가상메모리 레이아웃이 할당이 되고, 0~3GB 는 유저 영역, 3~4GB 는 커널영역에 할당이 된다. 유저영역의 구성은 스택,힙,데이터, 텍스트영역으로 구성된다. 유저 영역은 실제 물리 메모리에 4KB 씩 페이지 프레임으로 쪼개서 페이징이 되고, 커널영역은 8KB 의 실제 물리 메모리가 할당이 된다.

7. Page Fault 가 발생했을때 운영체제가 어떻게 동작하는지 기술하시오.

A : page fault 가 나면 page fault 핸들러가 동작되고 유저영역으로 부터 동작한 것인지, 커널영역으로부터 동작한 것인지 판별해서 유저영역으로 부터 동작한 것이면 Segmentation fault 가 나면서 빠져나간다. 만약 커널영역으로부터 동작한 것이면 page fault 핸들러가 해당 메모리에 page 를 할당하고 다시 해당 메모리에 접근한다.

8. 리눅스 실행 파일 포맷이 무엇인지 적으시오.

A : ELF(Executable and Linkable Format)

9. 프로세스와 스레드를 구별하는 방법에 대해 기술하시오.
A : 프로세스는 main 문 을 의미한다.
스레드는 프로세스안에서 구동이 되는 수행의 단위이다. 프로세스는 pid=tgid 이고 스레드는 pid!=tgid 이다.

10. Kernel 입장에서 Process 혹은 Thread 를 만들면 무엇을 생성하는가 ?
A : task_struct 구조체를 생성한다.

11. 리눅스 커널 소스에 보면 current 라는 것이 보인다. 이것이 무엇을 의미하는 것인지 적으시오. 커널 소스 코드와 함께 기술하시오.
A : current 는 thread_info 가리키는 태스크를 의미한다.

12. Memory Management 입장에서 Process 와 Thread 의 핵심적인 차이점은 무엇인가 ?
A : 프로세스끼리는 메모리를 공유하지 않고, 스레드끼리는 메모리를 공유한다.

13. Task 가 관리해야하는 3 가지 Context 가 있다. System Context, Memory Context, HW Context 가 있다. 이중 HW Context 는 무엇을 하기 위한 구조인가 ?
A : cpu 의 ALU 에 들어있는 HW 레지스터들에 대한 context-switching 이다.

14. 리눅스 커널의 스케쥴링 정책중 Deadline 방식에 대해 기술하시오.
A : deadline 이 가장 가까운 태스크를 스케줄링 대상으로 선정한다. DEADLINE 정책을 사용하는 태스크의 경우에 우선순위는 의미가 없다.

15. TASK_INTURRUPTIBLE 과 TASK_UNINTERRUPTIBLE 은 왜 필요한지 기술하시오.
A
- **TASK_INTURRUPTIBLE :** 태스크가 실행되는 도중에 비동기적인 사건이 발생했을때 인터럽트를 발생시켜서 처리하기 위해 **TASK_INTURRUPTIBLE** 상태가 필요하다.
- **TASK_UNINTERRUPTIBLE :** 태스크가 실시간적으로 수행해야할 연산(영상/음성, 레이다)이 너무 많아서 당장 인터럽트들을 받으면 안되는 상태의 경우에는 태스크를 **TASK_UNINTERRUPTIBLE** 의 상태로 만들어서 인터럽트를 막는다.

16. O(N)과 O(1) Algorithm 에 대해 기술하시오. 그리고 무엇이 어떤 경우에 더 좋은지 기술하시오.
A : O(N)은 시간에 따라 처리할 갯수가 늘어나는 것이고, **O(1)** 은 시간이 지나도 처리할 갯수가 고정되어 있는 것을 말한다. 처리할 데이터 양이 적을 땐 당장 시간이 적게 걸리는 **O(N)**이 좋고, 처리할 데이터 양이 많아질 때는 걸리는 시간이 불변하는 **O(1)**이 좋다.

17. 현재 4 개의 CPU(0, 1, 2, 3)가 있고 각각의 RQ 에는 1, 2 개의 프로세스가 위치한다.
이 경우 2 번 CPU 에 있는 부모가 fork()를 수행하여 Task 를 만들어냈다. 이 Task 는 어디에 위치하는 것이 좋을까 ?
그리고 그 이유를 적으시오.

A : 2 번 CPU 의 자식 프로세스는 RQ 에 위치한다. 그 이유는 RQ 에 있는 프로세스에 의해 생성이 된 프로세스이기 때문이다.

18. 15 번 문제에서 이번에는 0, 1, 3 에 매우 많은 프로세스가 존재한다. 이 경우 3 번에서 fork()를 수행하여 Task 를 만들었다. 이 Task 는 어디에 위치하는 것이 좋을까? 역시 이유를 적으시오.
A :

19. UMA 와 NUMA 에 대해 기술하고 Kernel 에서 이들을 어떠한 방식으로 관리하는지 기술하시오. 커널 내부의 소스 코드와 함께 기술하도록 하시오.
A : UMA 는 하나의 메모리를 클럭 속도가 동일한 여러 **CPU** 가 하나의 메모리 버스를 통해 공유하는 것이고**, NUMA** 는 여러 메모리를 클럭 속도가 다른 여러 **CPU** 가 여러 개의 메모리 버스를 통해 접근하는 시스템을 말한다**.**

20. Kernel 의 Scheduling Mechanism 에서 Static Priority 와 Dynamic Priority 번호가 어떻게 되는지 적으시오.
A : Static Priority 는 **0~99** 번 **, Dynamic Priority** 는 **100~139** 번이다**.**

21. ZONE_HIGHMEM 에 대해 아는대로 기술하시오.
A : 물리 메모리가 커널의 가상메모리 공간보다 크다면 커널의 가상주소공간과 물리메모리는 **1:1** 로 연결해주고**,** 물리메모리의 나머지 부분은 필요할 때 동적으로 연결하여 사용한다**.** 이 때 남은 물리메모리 공간이 **ZONE_HIGHMEM** 이다**.**

22. 물리 메모리의 최소 단위를 무엇이라고 하며 크기가 얼마인가 ? 그리고 이러한 개념을 SW 적으로 구현한 구조체의 이름은 무엇인가 ?
A : 물리 메모리의 최소 단위는 페이지 프레임이고 4KB 이다. SW 적으로 구현한 구조체 이름은 page 이다.

24. 21 번에 이어 내부 단편화를 최소화 하기 위해 Buddy 에서 Page 를 받아 Slab 할당자를 사용한다. Slab 할당자는 어떤식으로 관리되는지 기술하시오.
A : 미리 4KB 페이지 프레임을 할당 받은 뒤 64 Byte 크기로 분할해 둔다. 사용자가 메모리 공간을 요청하면 미리 할당받아 분할하여 관리하고 있던 이 공간에서 할당 해준다. 이렇게 캐시 집합처럼 메모리를 관리하는 것을 slab 할당자라고 한다.

28. User Space 에도 Stack 이 있고 Kernel Space 에도 Stack 이 존재한다. 좀 더 정확히는 각각에 모두 Stack, Heap, Data, Text 의 메모리 기본 구성이 존재한다. 그 이유에 대해 기술하시오.
A : 커널도 프로그램이기 때문에 커널 영역의 소스코드를 실행시키기 위한 가상 메모리 공간이 필요하기 때문이다.

30. MMU(Memory Management Unit)의 주요 핵심 기능을 모두 적고 간략히 설명하시오.
A : 가상 주소로부터 물리 주소로의 변환

31. 하드디스크의 최소 단위를 무엇이라 부르고 그 크기는 얼마인가 ?
A : 페이지**, 4KB**

32. Character 디바이스 드라이버를 작성할 때 반드시 Wrapping 해야 하는 부분이 어디인가 ? (Task 구조체에서부터 연결된 부분까지를 쭉 이어서 작성하라)
A : 사용자 태스크가 호출할 함수들과 특정 하드웨어를 위한 디바이스 드라이버 코어의 함수를 연결해주는 부분

33. 예로 유저 영역에서 open 시스템 콜을 호출 했다고 가정할 때 커널에서는 어떤 함수가 동작하게 되는가 ? 실제 소스 코드 차원에서 이를 찾아서 기술하도록 한다.
A :

34. task_struct 에서 super_block 이 하는 역할은 무엇인가 ?
A : 루트의 위치를 찾아준다**.**

35. VFS(Virtual File System)이 동작하는 Mechanism 에 대해 서술하시오.

A : 사용자 태스크에서 **a.txt** 라는 파일을 인자로 **open()** 시스템 콜을 하면 우선 **a.txt** 파일이 어떤 파일 시스템인지 판별한다**.** 그 후 해당 파일 시스템 내부에 구현되어 있는 고유한 **open** 함수를 호출한다**.** 해당 파일 시스템에서 **a.txt** 파일의 **inode** 를 찾아서 **inode** 의 정보를 **VFS** 가 넘겨준 구조체에 넣어서 리턴한다**.** 그러면 **VFS** 는 이 구조체를 사용자 태스크에게 넘겨준다**.**

36. Linux Kernel 에서 Interrupt 를 크게 2 가지로 분류한다. 그 2 가지에 대해 각각 기술하고 간략히 설명하시오.
A : 인터럽트는 외부 인터럽트와 내부 인터럽트로 분류된다**.** 외부 인터럽트는 **CPU** 외부 핀에 의해 전기적인 신호 또는 하드웨어에 의해 발생하는 비동기적인 **HW** 적인 인터럽트이다**.** 반면 내부 인터럽트는 **SW** 적인 인터럽트이고 하드웨어와 무관한 **CPU** 내부에서 발생하는 비동기적인 이벤트에 대한 처리이다**.**

37. 내부 인터럽트는 다시 크게 3 분류로 나눌 수 있다. 3 가지를 분류하시오.
A : fault, trap, abort

39. 예로 모니터 3 개를 쓰는 경우 양쪽에 모두 인터럽트를 공유해야 한다. Linux Kernel 에서는 어떠한 방법을 통해 이들을 공유하는가 ?
A : 장치파일 내에서 모니터에 해당하는 주번호를 할당하고, 각각의 모니터들에 대해 부번호를 할당해서 구분한다.

40. System Call 호출시 Kernel 에서 실제 System Call 을 처리하기 위해 Indexing 을 수행하여 적절한 함수가 호출되도록 주소값을 저장해놓고 있다. 이 구조체의 이름을 적으시오.
A : sys_call_table

41. 38 에서 User Space 에서 System Call 번호를 전달한다. Intel Machine 에서는 이를 어디에 저장하는가 ? 또한 ARM Machine 에서는 이를 어디에 저장하는가 ?
A : 인텔에서는 idt_table 의 128 번 인덱스에 시스템 콜의 번호를 저장한다. ARM 에서는 vectors_start+0x1000 위치에 저장한다.

44. 커널 내부에서 메모리 할당이 필요한 이유는 무엇인가 ?
A : 커널의 프로그램이 구동되기 위한 메모리가 필요한 경우

45. 메모리를 불연속적으로 할당하는 기법은 무엇인가 ?
A : 가상화 기법

46. 메모리를 연속적으로 할당하는 기법은 무엇인가 ?
A : 버디**,** 슬랩 할당

47. Mutex 와 Semaphore 의 차이점을 기술하시오.
- 뮤텍스는 스레드들간에 공유가 배제되는 객체이다**.** 공유 자원이 수행 중에 오직 한 태스크에게만 소유되어야 할 필요가 있을 때 그 자원에 대한 뮤텍스 객체를 생성시킨다**.**
- 세마포어는 여러 개의 공유자원과 여러 개의 카운트 값을 가지는 뮤텍스라고 볼 수 있다**.** 공유자원에 대한 접속을 제어하기 위한 신호이다**.** 공유 자원에 접근할 수 있는 최대 허용치만큼 동시에 사용자 접근을 할 수 있게 한다**.**

48. module_init() 함수 호출은 언제 이루어지는가 ?
A : insmod 명령어를 수행할 때 **()**안의 함수를 동작시킴**.**

49. module_exit() 함수 호출은 언제 이루어지는가 ?
A : rmmod 명령어를 수행할 때 **()**안의 함수를 동작시킴**.**

50. thread_union 에 대해 기술하시오.
A : thread_union 의 구조체에는 **thread** 들의 정보가 있는 **thread_info** 구조체와 커널 영역의 메모리를 할당하는 **stack** 변수가 존재한다**.**

53. Kernel 자체에 kmalloc(), vmalloc(), __get_free_pages()를 통해 메모리를 할당할 수 있다. 또한 kfree(), vfree(), free_pages()를 통해 할당한 메모리를 해제할 수 있다. 이러한 Mechanism 이 필요한 이유가 무엇인지 자세히 기술하라.

A : 물리메모리 또는 가상 메모리 공간에 연속된 메모리 공간을 할당 받아서 캐시의 **locality** 를 증가시키기 위해서 필요함.

55. OoO(Out-of-Order)인 비순차 실행에 대해 기술하라.

A : CPU 는 원래 명령어들이 들어오는 순서대로 처리해주었는데**(**순차 실행**) CPU** 의 실행 속도를 빠르게 하기 위해 명령어의 실행 순서를 임의로 바꾸는 것을 말한다.

57. CISC Architecture 와 RISC Architecture 에 대한 차이점을 기술하라.

A : CISC 구조는 명령어의 갯수가 많고 구조가 복잡하다. 이로 인해 프로그래밍은 간단해지지만 전력소모가 많고 하드웨어 구조가 복잡하고 가격이 비싸다

RISC 구조는 명령어가 단순하고 숫자가 적다. 이로 인해 프로그래밍은 복잡해졌지만 전력소모가 적고 하드웨어 구조가 단순하다.

59. Pipeline 이 깨지는 경우에 대해 자세히 기술하시오.

A : goto 문이나 **branch** 문이 수행될 경우 현재 수행 중이던 파이프라인이 깨지게 된다. 파이프 라인이 깨지게 되면 이제까지 명령어를 수행하기 위해 소모되었던 **cpu** 클럭이 낭비가 되므로 효율성이 떨어진다.

60. CPU 들은 각각 저마다 이것을 가지고 있다. Compiler 개발자들은 이것을 고려해서 Compiler 를 만들어야 한다. 또한 HW 입장에서도 이것을 고려해서 설계를 해야 한다. 여기서 말하는 이것이란 무엇인가 ?

A : 기계어**(**어셈블리어**)**

61. Intel 의 Hyper Threading 기술에 대해 상세히 기술하시오.

A : fork() 시스템 콜을 **FPGA** 를 이용해 회로로 설계함으로써 **CPU** 를 마치 실제 갯수보다 **2** 배의 갯수처럼 보이게 만드는 기술이다.

66. 자신이 사용하는 리눅스 커널의 버전을 확인해보고 https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/$(자신의 버전).tar.gz 를 다운받아보고 이 압축된 파일을 압축 해제하고 task_struct 를 찾아보도록 한다. 일련의 과정을 기술하면 된다.

A :

1) 터미널 창에 **"vi -t task_struct"** 명령어를 입력함으로 써 **task_struct** 가 어느 파일에 있는지 찾아본다.

2) task_struct 구조체가 정의되어 있는 파일의 번호를 입력하고 들어간다.

3) task_struct 구조체의 정의부분을 확인한다.

```
struct task_struct {
        volatile long state;    /* -1 unrunnable, 0 runnable, >0 stopped */
        void *stack;
        atomic_t usage;
        unsigned int flags;     /* per process flags, defined below */
        unsigned int ptrace;
```

67. Multi-Tasking 의 원리에 대해 서술하시오. (Run Queue, Wait Queue, CPU 에 기초하여 서술하시오)
A : 여러 태스크들이 **cpu** 자원을 번갈아 가며 점유하는 것을 말한다**. cpu** 를 점유하는 태스크들은 **Run Queue** 에 들어가서 돌아가며 **cpu** 를 점유하고**, cpu** 를 점유하지 못하는 태스크들은 **Wait Queue** 에 들어가 있다**.**

68. 현재 삽입된 디바이스 드라이버의 리스트를 보는 명령어는 무엇인가 ?
A : lsmod

69. System Call Mechanism 에 대해 기술하시오.
A : 시스템 콜은 유일한 **sw** 적인 인터럽트 이며**, idt_table** 의 **128(0x80)** 에 해당한다**. system_call** 이 호출되면 **idt_table** 의 **128** 번을 인자로 시스템콜을 발생 시킨다**.** 그 다음에 현재 태스크의 진행되고 있는 시점에서 문맥 저장 **(context save)**을 하고 **sys_call_table** 의 해당하는 시스템 콜 핸들러를 수행한다**.** 시스템 콜 핸들러를 수행하고 난 뒤 다시 문맥을 복원**(context restore)**해 주고 원래 태스크가 하던 일을 진행한다**.**

70. Process 와 VM 과의 관계에 대해 기술하시오.
A : 하드디스크에 있는 프로그램이 실행이 되기 위해서는 주 메모리에 올라와야 한다**.** 그러나 프로그램 용량이 주메모리 용량보다 큰 경우**,** 당장 실행에 필요한 부분만 저장하고 나머지는 하드 디스크에 저장하면서 동작하도록 하는 개념을 가상메모리라고 한다**.** 가상메모리에는 커널영역**,** 스택**,** 힙**,** 데이터**,** 텍스트 영역이 존재하며 각각의 영역을 세그먼트 또는 **region** 이라고 한다**.** 프로세스 **1** 개당 가상메모리 레이아웃이 **1** 개씩 할당이 되고 **32bit cpu** 에서는 **4GB** 가 할당이 된다**.**

74. C.O.W Architecture 에 대해 기술하시오.
A : fork()를 하게 되면 자식 프로세스가 생성이 되고**,** 부모 프로세스와 다른 독립적인 메모리 레이아웃을 할당받는다**.** 그러나 이 때**,** 부모 프로세스의 메모리 공간을 바로 복사해 주는 것이 아니라 자식 프로세스에서 **write** 작업을 수행할 때 복사를 해주어서 **overhead** 를 줄이게 된다**.** 그렇게 함으로써 예를들어 **fork()** 후 **exec()** 이 수행될 때에는 메모리를 복사할 필요가 없으므로 효율적이게 된다**.**

75. Blocking 연산과 Non-Blocking 연산의 차이점에 대해 기술하시오.
A : Blocking 방식은 함수를 호출하고 반환이 될 때까지 해당 위치에서 계속 대기하는 것이고**, Non-Blocking** 방식은 함수를 호출하고 다른 일을 수행하다가**,** 호출한 함수가 일을 끝내고 **Callback** 을 전달해주는 것이다**.**

80. 리눅스에서 말하는 File Descriptor(fd)란 무엇인가 ?
A : 시스템으로 부터 할당받은 파일을 대표하는 정수값**,** 리눅스는 모든 것을 파일로 보기 때문에 프로세스에서 파일들에 접근할때 파일 디스크립터라는 개념을 씀**.**

81. stat(argv[2], &buf)일때 stat System Call 을 통해 채운 buf.st_mode 의 값에 대해 기술하시오.
A : buf.st_mode 의 최상위 4bit 는 파일의 종류를 나타내고,
- 11 번째 비트는 setuid(소유자 권한), 10 번째 비트는 setgid(그룹 권한), 9 번째 비트는 sticky bit(모든 유저가 공유)
- 8~6 번재 비트는 root 사용자 권한의 r,w,x 권한
- 5~3 번째 비트는 그룹 권한의 r,w,x 권한
- **2~0** 번째 비트는 제 **3** 자에 대한 **r,w,x** 권한

82. 프로세스들은 최소 메모리를 관리하기 위한 mm, 파일 디스크립터인 fd_array, 그리고 signal 을 포함하고 있는데 그 이유에 대해 기술하시오.
A : mm_struct 는 프로세스의 가상메모리 영역의 크기와 영역의 종류등을 관리하기 위해 사용되는 구조체이고**,** **fd_array** 는 **fd(**파일 디스크립터**)**를 인덱스로 사용해서 파일 객체를 가리키는 배열이다**. signal** 은 시그널 핸들러를 관리하기 위한 구조체 이다**.** 각각의 구조체 및 배열은 프로세스의 자원들을 효율적으로 관리하기 위해 커널영역에 정의되어 있는 변수이다**.**

87. 클라우드 기술의 핵심인 OS 가상화 기술에 대한 질문이다. OS 가상화에서 핵심에 해당하는 3 가지를 기술하시오.
A : CPU 가상화**,** 메모리 가상화**, I/O** 가상화 기술

93. Critical Section 이 무엇인지 기술하시오.
A : 여러 태스크들이 공유하는 한정된 메모리 공간을 말한다.

94. 유저에서 fork() 를 수행할때 벌어지는 일들 전부를 실제 소스 코드 차원에서 해석하도록 하시오.
A : 유저에서 **fork()**를 수행하면 프로세스가 하나 더 만들어지고 가상메모리 레이아웃을 부모 프로세스로부터 그대로 복사해온다. 자식 프로세스는 **fork()**를 한 명령어부터 수행이 되고 부모 프로세스와 멀티 태스킹으로 수행하게 된다.

95. 리눅스 커널의 arch 디렉토리에 대해서 설명하시오.
A : arch 는 **architecture** 의 약자로 여러 회사에서 만든 **cpu** 에 대한 정보를 담은 디렉토리를 말한다.

96. 95 번 문제에서 arm 디렉토리 내부에 대해 설명하도록 하시오.
A : arch 디렉토리 안의 **arm** 디렉토리 내부에는 **arm** 계열의 **cpu** 에서 아키텍처**, CPU** 내부 **HW** 에 관련된 정보가 들어있다.

97. 리눅스 커널 arch 디렉토리에서 c6x 가 무엇인지 기술하시오.
A : TI 사의 **DSP** 프로세서

98. Intel 아키텍처에서 실제 HW 인터럽트를 어떤 함수를 가지고 처리하게 되는지 코드와 함께 설명하시오.
A : irq_desc 구조체에 정의되어 있는 함수 포인터 다발을 이용해 인터럽트를 처리한다.

99. ARM 에서 System Call 을 사용할 때 사용하는 레지스터를 적으시오.
A : ax 레지스터

100. 반성문 서술
A : 2 개월 간 나름 열심히 한다고 매일매일 **5** 시간 미만으로 자면서 공부했지만 **,** 그래도 후회가 남는다. 수업 시간에 좀 더 집중하지 못했던 것 같고**,** 주말에는 약간 해이해져서 생각만큼 많이 하지는 못했던 것 같다. 좀 더 집중력을 가지고 초심으로 돌아가 남은 기간 하루하루를 후회없이 공부하면서 보내야 겠다.
또한 각 과목별로 노트 정리를 좀 더 신경써서 그날 그날 노트 정리한 것을 토대로 수업 내용을 복습하고 넘어가야 겠다.