TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

[2 개월차 시험]

강사 - 이상훈
gcccompil3r@gmail.com
학생 – 이우석
colre99@naver.com

[100 문제]

1.파이프 통신을 구현하고 c type.c 라고 입력할 경우

현재 위치의 디렉토리에 type.c 파일을 생성하도록 프로그래밍하시오.

2.369 게임을 작성하시오.

2 초내에 값을 입력하게 하시오.

박수를 쳐야 하는 경우를 Ctrl + C 를 누르도록 한다.

2 초 내에 값을 입력하지 못할 경우 게임이 오버되게 한다.

Ctrl + C 를 누르면 "Clap!" 이라는 문자열이 출력되게 한다.

3.리눅스 커널은 운영체제(OS)다.

OS 가 관리해야 하는 제일 중요한 5 가지에 대해 기술하시오.

: CPU, MEMORY, DISK, TERMINAL, NETWORK

4.Unix 계열의 모든 OS 는 모든 것을 무엇으로 관리하는가 ?

: file

5.리눅스에는 여러 장점이 있다.(배점 0.2 점)

아래의 장점들 각각에 대해 기술하라.

* 사용자 임의대로 재구성이 가능하다. (오픈소스로서 사용자 임의변경 가능)
* 열약한 환경에서도 HW 자원을 적절히 활용하여 동작한다. (호환이 좋다=최적화)
* 커널의 크기가 작다. (사이즈 변경이 가능하다)
* 완벽한 멀티유저, 멀티태스킹 시스템 (호환성이 좋아서 가능)
* 뛰어난 안정성 (수많은 실력자들의 수정을 통해 만들어짐)
* 빠른 업그레이드 (오픈소스이다 보니 수정이 많을 수 있다)
* 강력한 네트워크 지원 (사용자들의 많은 업그레이드로 인해)
* 풍부한 소프트웨어 (GNU 덕분에 자료가 많다)

6.32 bit System 에서 User 와 Kernel 의 Space 구간을 적으시오.

7.Page Fault 가 발생했을때

운영체제가 어떻게 동작하는지 기술하시오

: 동작을 무시한다

8.리눅스 실행 파일 포맷이 무엇인지 적으시오.

:ELF 파일

9.프로세스와 스레드를 구별하는 방법에 대해 기술하시오

: 동작중인 프로그램이면 프로세스 이고, 동작중이지 않고 있는거라면 스레드

10.Kernel 입장에서 Process 혹은 Thread 를 만들면 무엇을 생성하는가 ?

: TASK

11.리눅스 커널 소스에 보면 current 라는 것이 보인다. 이것이 무엇을 의미하는 것인지 적으시오.

커널 소스 코드와 함께 기술하시오.

: 의미→ 커널 내부에 정의되어 있는 매크로로서 현재 태스크의 task_struct 구조체를 가리킬 수 있게 해주는 매크로.

ex)

```
SYSCALL_DEFINE0(getpid)
{
        return task_tgid_vmr(current);      → tastk_tgid_vmr()은 해당 task_struct 구조체의 tgid 필드를 리턴한다.
}
```

12.Memory Management 입장에서 Process 와 Thread 의 핵심적인 차이점은 무엇인가 ?

:Memory 에 올라가면 Process, 올라가지 않는다면 Thread.

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 ?

: 문맥교환(context switch)할때 태스크의 현재실행 위치에 대한 정보를 유지하며, 실행중이던 태스크가 대기 상태나 준비 상태로 전이할때 이 태스크가 어디까지 실행했는지 기억해 두는 공간.

14.리눅스 커널의 스케쥴링 정책중 Deadline 방식에 대해 기술하시오.

: 기존 리눅스의 실시간 태스크 스케쥴링 정책은 우선순위에 기반하여 스케쥴링 대상을 선정하는데 반해, DEADLINE 정책은 deadline 이 가까운(가장 급한) 태스크를 스케줄링 대상으로 선정한다. 덧붙여, 완료시간을 deadline, 작업량을 runtime, 초당 30 회라는 주기성을 period 라고 부른다. Deadline 방식은 간단히 말하자면 각각의 '해야하는일' 은 언제까지는 반드시 수행이 완료되어야 한다는 '완료시간'이 정해져 있고, 이 정책을 사용하는 태스크들의 runtime 합은 CPU 의 최대 처리량을 넘어서는 안된다. 예로는 어떤 태스크가 1 초당 30 번씩 '해야하는일'을 가지고 있다하면 그 해야하는 일은 적어도 (1 초/30) 보다는 적은 시간 내에 수행될 수 있는 '작업량'을 가지고 있다는 의미이다.

15.TASK_INTURRUPTIBLE 과 TASK_UNINTERRUPTIBLE 은 왜 필요한지 기술하시오.

:(TASK_RUNNING)상태에 있던 태스크가 특정한 사건을 기다려야 할 필요가 있으면 대기상태로 전이한다.

TASK_UNINTERRUPTIBLE 상태는 시그널에 반응하지 않는 점이 있지만, 둘다 모두 특정사건을 기다린다는 점이 있다.
결론만 말하자면, 위의 대기상태들은 사건을 기다리게 하는 태스크를 사건이 발생할 때 까지 대기상태로 전이시키고,
준비상태에 있던 다른 태스크를 실행시킴으로써 CPU 의 효율을 높인 것이기에 이러할 때 필요하다.

16.O(N)과 O(1) Algorithm 에 대해 기술하시오.
그리고 무엇이 어떤 경우에 더 좋은지 기술하시오.
:O(N)과 O(1) Algorithm 은 실시간 태스크중 우선순위가 가장높은 태스크를 찾는 효율적인 방법이다.
모든 태스크는 tasklist 라느 이중연결 리스트에 연결되어있어 이 리스트를 이용하면 시스템내의 모든 태스크를 접근
하는것이 가능하다. 따라서 태스크 중 가장높은 우선순위의 태스크를 골라내면 되는데 태스크의 개수가 늘어나면 그만큼
스케줄링에 걸리는 시간도 선형적으로 증가하므로 O(N)(시간복잡도)는 적은양을 찾을때에는 알맞지만, 점차적으로
늘어나는 태스크 또는 많은양이면 O(1)이 더욱 효율적이다. O(1)은 시간이 계속 지남에 따라 상관없이 처리할 개수를
일정하게 처리하여 선형적인 폭이 되지않는다. 즉, O(1)이 시간이 흐를수록 사용하기에 더욱 낫다. 스케줄링에 걸리는
시간이 증가되지 않기 때문이다.

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.

18.앞선 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.

19.UMA 와 NUMA 에 대해 기술하고 Kernel 에서 이들을 어떠한 방식으로 관리하는지 기술하시오.
커널 내부의 소스 코드와 함께 기술하도록 하시오.
: NUMA 같은 경우, 복수 개의 CPU 메모리 등의 자원을 공유하기 때문에 성능상 병목현상이 발생할 수 있는데, CPU 들을 몇개의 그룹으로 나누고 각각의 그룹에게 별도의 지역메모리를 주는 구조가 NUMA(Non-Uniform Memory Access).
UMA 는이에 반해 기존 시스템을 말한다 (Uniform Memory Access)

20.Kernel 의 Scheduling Mechanism 에서 Static Priority 와 Dynamic Priority 번호가 어떻게 되는지 적으시오.

21. ZONE_HIGHMEM 에 대해 아는대로 기술하시오.
: 물리메모리가 1GB 이상이라면 896MB 까지를 커널의 가상주소공간과 1:1 로 연결해주고, 나머지 부분은 필요할 때 동적으로 연결하여 사용하는 구조. 이때, 896MB 이상의 메모리 영역을 ZONE_HIGHMEM 이라고 지칭한다.

22.물리 메모리의 최소 단위를 무엇이라고 하며 크기가 얼마인가 ?
그리고 이러한 개념을 SW 적으로 구현한 구조체의 이름은 무엇인가 ?
: 최소단위 - page frame
  크기 – 4KB

구조체의 이름 - zone 구조체

23.Linux Kernel 은 외부 단편화와 메모리 부하를 최소화하기 위해 Buddy 할당자를 사용한다.

Buddy 할당자의 Algorithm 을 자세히 기술하시오.

: Buddy 할당자는 zone 구조체에 존재하는 free_area[] 배열을 통해 구축된다. Buddy 할당자는 요청된 크기를 만족하는 최소의 order 에서 페이지 프레임을 할당해 준다. 만일 그 order 에 가용한 페이지 프레임이 존재하지 않으면 상위 order 에서 페이지 프레임을 할당받아 두 부분으로 나누어, 한 부분은 할당해주고 나머지 나머지 부분은 하위 order 에서 가용 페이지 프레임으로 관리한다. 이때 나누어진 두 부분을 친구(Buddy)라고 부르며 이 때문에 이 할당자를 버디 할당자 라고 부른다.

24.앞선 문제에 이어 내부 단편화를 최소화 하기 위해 Buddy 에서 Page 를 받아 Slab 할당자를 사용한다.

Slab 할당자는 어떤식으로 관리되는지 기술하시오.

: Slab 할당자는 어느 공간을 해제한다면 버디로 반납하는 것이 아니라 미리 할당 받아 관리하던 공간에서 다시 가지고 있게 하는데, 마치 일종의 캐시(cache)로 사용하는것. 이러한 cache 의 집합을 통해 메모리를 관리하는 정책을 Slab 할당자라고 부른다. 관리는 함수(kmem_cache_alloc(), kmem_cache_free())들을 통해 어느 캐시에서 공간을 할당받고, 어느 캐시로 공간을 해제시켜야 하는지를 지정해 주어야한다. 더이상 할당해줄 공간이 없다면 슬랩 할당자는 버디로 부터 페이지 프레임을 더 할당받아야 하는데 이때는 kmem_cache_grow()같은 함수를 호출하여 슬랩을 확장한다.

25.Kernel 은 Memory Management 를 수행하기 위해 VM(가상 메모리)를 관리한다. 가상 메모리의 구조인 Stack, Heap, Data, Text 는 어디에 기록되는가 ? (Task 구조체의 어떠한 구조체가 이를 표현하는지 기술하시오)

26. 25 번에서 Stack, Heap, Data, Text 등 서로 같은 속성을 가진 Page 를 모아서 관리하기 위한 구조체 무엇인가 ?
(역시 Task 구조체의 어떠한 구조체에서 어떠한 식으로 연결되는지 기술하시오)
:page_directory

27. 프로그램을 실행한다고 하면 fork(), execve()의 콤보로 이어지게 된다.
이때 실제 gcc *.c 로 컴파일한 a.out 을 ./a.out 을 통해 실행한다고 가정한다.
실행을 한다고 하면 a.out File 의 Text 영역에 해당하는 부분을 읽어야 한다.
실제 Kernel 은 무엇을 읽고 이 영역들을 적절한 값으로 채워주는가 ?

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.

29.VM(가상 메모리)와 PM(물리 메모리)를 관리하는데 있어 VM 을 PM 으로 변환시키는 Paging Mechanism 에 대해 Kernel 에 기반하여 서술하시오.

30.MMU(Memory Management Unit)의 주요 핵심 기능을 모두 적고 간략히 설명하시오.

:대부분의 CPU 는 가상주소를 사용하는 운영체제가 원활히 수행될 수 있도록 하기위해 가상 주소로부터 물리주소로의 변환을 담당하는 별도의 하드웨어를 가지는데, 이것이 MMU 이다. 주요 핵심 기능으로는, 각각의 독립된 태스크가 개별적인 가상 메모리 공간을 점유하게 하는것이고, 또한 태스크가 물리주소나 다른 프로그램에 대한 정보없이 독립적으로 동작할 수 있도록 한다. 그리고 가상적으로만 연속인 메모리를 할당 받아서 사용하는데 이때 사용하는 함수가 vmalloc() 과 vfree()이다.

31.하드디스크의 최소 단위를 무엇이라 부르고 그 크기는 얼마인가 ?

: 최소단위: 섹터, 크기: 512byte

32.Character 디바이스 드라이버를 작성할 때 반드시 Wrapping 해야 하는 부분이 어디인가 ?

(Task 구조체에서 부터 연결된 부분까지를 쭉 이어서 작성하라)

: file_operations

33.예로 유저 영역에서 open 시스템 콜을 호출 했다고 가정할 때 커널에서는 어떤 함수가 동작하게 되는가 ?

실제 소스 코드 차원에서 이를 찾아서 기술하도록 한다.

34.task_struct 에서 super_block 이 하는 역할은 무엇인가 ?

: '/'를 사용함으로써 자신이 관리하는 공간의 맨 앞부분에 적어두면 나중에 언제라도 이 디스크를 사용하려 하는 시점에 디스크의 맨 앞부분만 읽으면 '/' 디렉터리의 위치를 알수있고, 따라서 '/' 디렉토리 하위에 존재하는 다른 모든 파일을 찾을수 있게 하는 역할.

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

: 사용자 태스크는 자신이 접근하는 파일이 어는 파일 시스템에 저장되어 있는지 굳이 알 필요가 없잉 일관된 POSIX 표준 인터페이스만을 사용하는것이 가능해진다. 사용자 태스크 입장에서 본다면 VFS 는 가상적이긴 하지만 실제 접근을 하게되는 파일시스템 인것.
VFS 는 다양한 파일시스템과 데이터를 주고 받아야하는데, 리눅스는 VFS 내에 5 개의 객체(object)를 도입하였으며, 사용자 태스크에게 제공할 일관된 인터페이스를 정의함. 1. 수퍼블록 2. 아이노드 객체 3. 파일객체 4. 디엔트리 객체 5. 패스객체.
태스크가 ext2 파일시스템을 사용하려(마운트)한다고 가정하면, 사용자의 마운트 요청을 받은 VFS 는 ext2 의 마운트 함수를 호출하면서 인자로 빈 수퍼블록 객체를 하나 넘긴다.그러면 ext2 는 자체적으로 구현한 파일시스템 내부함수를 이용하여, 파티션 앞부분에 기록해 두었던 수퍼블록 구조체를 읽은 뒤 이를 바탕으로 VFS 가 넘긴 수퍼블록 객체의 내용을 채워서 리턴 해준다.

36.Linux Kernel 에서 Interrupt 를 크게 2 가지로 분류한다.
그 2 가지에 대해 각각 기술하고 간략히 설명하시오.

: 1. 외부 인터럽트: 현재 수행중인 태스크와 관련없는 주변장치에서 발생된 비동기적인 하드웨어적인 사건.
2. 내부 인터럽트: 현재 수행중인 태스크와 관련있는 동기적으로 발생하는 사건.

37.내부 인터럽트는 다시 크게 3 분류로 나눌 수 있다. 3 가지를 분류하시오

: fault, trap, abort

38.앞선 문제에서 분류한 녀석들의 특징에 대해 기술하시오.

: 1. fault: 리눅스 커널은 fault 를 일으킨 명령어 주소를 eip 에 넣어 두었다가 해당 핸들러가 종료되고 나며
eip 에 저장되어 있는 주소부터 다시 수행을 시작한다.

2. trap: 리눅스 커널은 trap 을 일으킨 명령어의 다음주소를 eip 에 넣어 두었다가 그 다음부터 다시 수행한다.

3. abort: 심각한 에러인 경우이므로 eip 값을 저장해야할 필요가 없으며, 현재 태스크를 강제 종료 시키면 된다.

39.예로 모니터 3 개를 쓰는 경우 양쪽에 모두 인터럽트를 공유해야 한다.

Linux Kernel 에서는 어떠한 방법을 통해 이들을 공유하는가 ?

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.

41.38 에서 User Space 에서 System Call 번호를 전달한다.

Intel Machine 에서는 이를 어디에 저장하는가 ?

또한 ARM Machine 에서는 이를 어디에 저장하는가 ?

42. Paging Mechanism 에서 핵심이 되는 Page Directory 는 mm_struct 의 어떤 변수가 가지고 있는가 ?

: pgd

43. 또한 Page Directory 를 가르키는 Intel 전용 Register 가 존재한다.

이 Register 의 이름을 적고 ARM 에서 이 역할을 하는 레지스터의 이름을 적으시오.

: INTEL: cr3 register

ARM: ttb base register

44.커널 내부에서 메모리 할당이 필요한 이유는 무엇인가 ?

: 커널또한 메모리가 필요하기 때문에 (메모리에 올라가야 동작)

45.메모리를 불연속적으로 할당하는 기법은 무엇인가 ?

:list

46.메모리를 연속적으로 할당하는 기법은 무엇인가 ?

:array

47.Mutex 와 Semaphore 의 차이점을 기술하시오.

: Mutex 와 Semaphore 의 차이점은 공유한다 안한다의 방법의 차이

Mutex 는 공유를 하지않고, Semaphore 는 공유하는 것이다.

48.module_init() 함수 호출은 언제 이루어지는가 ?

: 모듈이 시작될때 호출된다.

49.module_exit() 함수 호출은 언제 이루어지는가 ?

: 모듈이 종료될때 호출된다.

50.thread_union 에 대해 기술하시오.

: 태스크 당 할당되는 커널스택 이라 불린다. thread_info 구조체를 포함하고 있다.

커널과 사용자 수준간의 변화 시에 현재까지의 작업상황을 어딘가에 저장해 놓는데, 커널로 진입되는 시점에

커널 스택안에 현재 레지스터의 값들을 구조체를 이용하여 일목요연하게 저장함으로써 이뤄진다.이러한 목적으로 사용된다.

51.Device Driver 는 Major Number 와 Minor Number 를 통해 Device 를 관리한다.

실제 Device 의 Major Number 와 Minor Number 를 저장하는 변수는 어떤 구조체의 어떤 변수인가 ?

(역시 Task 구조체에서부터 쭉 찾아오길 바람)

: inode 구조체의 i_rdev 변수

52.예로 간단한 Character Device Driver 를 작성했다고 가정해본다.

그리고 insmod 를 통해 Driver 를 Kernel 내에 삽입했으며 mknod 를 이용하여 /dev/장치파일을 생성하였다.
그리고 이에 적절한 User 프로그램을 동작시켰다. 이 Driver 가 실제 Kernel 에서 어떻게 관리되고 사용되는지
내부 Mechanism 을 기술하시오.

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.

54.Character Device Driver 를 아래와 같이 동작하게 만드시오.
read(fd, buf, 10)을 동작시킬 경우 1 ~ 10 까지의 덧셈을 반환하도록 한다.
write(fd, buf, 5)를 동작시킬 경우 1 ~ 5 곱셈을 반환하도록 한다.
close(fd)를 수행하면 Kernel 내에서 "Finalize Device Driver"가 출력되게 하라!

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

56.Compiler 의 Instruction Scheduling 에 대해 기술하라.

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

58.Compiler 의 Instruction Scheduling 은 Run-Time 이 아닌 Compile-Time 에 결정된다. 고로 이를 Static Instruction Scheduling 이라 할 수 있다. Intel 계열의 Machine 에서는 Compiler 의 힘을 빌리지 않고도 어느정도의 Instruction Scheduling 을 HW 의 힘만으로 수행할 수 있다. 이러한 것을 무엇이라 부르는가 ?

59.Pipeline 이 깨지는 경우에 대해 자세히 기술하시오

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 ?

61.Intel 의 Hyper Threading 기술에 대해 상세히 기술하시오.

62.그동안 많은 것을 배웠을 것이다. 최종적으로 Kernel Map 을 그려보도록 한다. (Networking 부분은 생략해도 좋다) 예로는 다음을 생각해보도록 한다. 여러분이 좋아하는 게임을 더블 클릭하여 실행한다고 할 때 그 과정 자체를 Linux Kernel 에 입각하여 기술하도록 하시오. (그림과 설명을 같이 넣어서 해석하도록 한다) 소스 코드도 함께 추가하여 설명해야 한다.

63.파일의 종류를 확인하는 프로그램을 작성하시도록 하시오.

64.서버와 클라이언트가 1 초 마다 Hi, Hello 를 주고 받게 만드시오.

65.Shared Memory 를 통해 임의의 파일을 읽고 그 내용을 공유하도록 프로그래밍하시오

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

67.Multi-Tasking 의 원리에 대해 서술하시오.
(Run Queue, Wait Queue, CPU 에 기초하여 서술하시오)

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

69.System Call Mechanism 에 대해 기술하시오.

70.Process 와 VM 과의 관계에 대해 기술하시오.

71.인자로 파일을 입력 받아 해당 파일의 앞 부분 5 줄을 출력하고 추가적으로 뒷 부분의 5 줄을 출력하는 프로그램을 작성하시오.

72.디렉토리 내에 들어 있는 모든 File 들을 출력하는 Program 을 작성하시오.

73.Linux 에서 fork()를 수행하면 Process 를 생성한다.

이때 부모 프로세스를 gdb 에서 디버깅하고자하면 어떤 명령어를 입력해야 하는가 ?

: gcc -g

74.C.O.W Architecture 에 대해 기술하시오.

: copy on write 로 fork() 할때 야기되는 주소 공간 복사 비용을 줄일때 사용한다.

75.Blocking 연산과 Non-Blocking 연산의 차이점에 대해 기술하시오.

:

76.자식이 정상 종료되었는지 비정상 종료되었는지 파악하는 프로그램을 작성하시오.

77.데몬 프로세스를 작성하시오.

잠시 동안 데몬이 아니고 영구히 데몬이 되게 하시오.

78.SIGINT 는 무시하고 SIGQUIT 을 맞으면 죽는 프로그램을 작성하시오.

79.goto 는 굉장히 유용한 C 언어 문법이다.

그러나 어떤 경우에는 goto 를 쓰기가 힘든 경우가 존재한다.

이 경우가 언제인지 기술하고 해당하는 경우 문제를 어떤식으로 해결 해야 하는지 프로그래밍 해보시오.

80.리눅스에서 말하는 File Descriptor(fd)란 무엇인가 ?

: 커널이 태스크마다 관리해 주어야 할 정보.

81.stat(argv[2], &buf)일때 stat System Call 을 통해 채운 buf.st_mode 의 값에 대해 기술하시오.

82.프로세스들은 최소 메모리를 관리하기 위한 mm, 파일 디스크립터인 fd_array, 그리고 signal 을 포함하고 있는데 그 이유에 대해 기술하시오.

83.디렉토리를 만드는 명령어는 mkdir 명령어다.

man -s2 mkdir 을 활용하여 mkdir System Call 을 볼 수 있다.

이를 참고하여 디렉토리를 만드는 프로그램을 작성해보자!

84.이번에는 랜덤한 이름(길이도 랜덤)을 가지도록 디렉토리를 3 개 만들어보자!

(너무 길면 힘드니까 적당한 크기로 잡도록함)

85.랜덤한 이름을 가지도록 디렉토리 3 개를 만들고 각각의 디렉토리에 5 ~ 10 개 사이의 랜덤한 이름(길이도 랜덤)을 가지도록 파일을 만들어보자! (너무 길면 힘드니까 적당한 크기로 잡도록함)

86.앞선 문제까지 진행된 상태에서 모든 디렉토리를 순회하며
3 개의 디렉토리와 그 안의 모든 파일들의 이름 중 a, b, c 가 1 개라도 들어있다면 이들을 출력하라!
출력할 때 디렉토리인지 파일인지 여부를 판별하도록 하시오.

87.클라우드 기술의 핵심인 OS 가상화 기술에 대한 질문이다.
OS 가상화에서 핵심에 해당하는 3 가지를 기술하시오.

88.반 인원이 모두 참여할 수 있는 채팅 프로그램을 구현하시오.

89.앞선 문제의 답에 도배를 방지 기능을 추가하시오.

90.앞서 만든 프로그램 조차도 공격할 수 있는 프로그램을 작성하시오.

91.네트워크 상에서 구조체를 전달할 수 있게 프로그래밍 하시오.

92.앞선 문제를 응용하여 Queue 와 Network 프로그래밍을 연동하시오.

93.Critical Section 이 무엇인지 기술하시오.
: 임계영역. 공유공간으로 둘 이상의 스레드가 동시에 접근해서는 안되는 공간.

94.유저에서 fork() 를 수행할때 벌어지는 일들 전부를 실제 소스 코드 차원에서 해석하도록 하시오.

95.리눅스 커널의 arch 디렉토리에 대해서 설명하시오.

: 리눅스 커널 기능중 하드웨어 종속적인 부분들이 구현된 디렉토리이다. 그래서 이름도 architecture 를 의미하는 arch 이다.
이 디렉토리는 CPU 의 타입에 따라 하위 디렉토리로 다시 구분된다. 대표적으로 인텔의 i386, ARM(Advanced RISC Machine)의 ARM 계열 등등. i386 을 기준으로 arch/x86 디렉토리 중 arch/x86/boot 디렉토리에는 시스템의 부팅 시 사용하는 부트스크랩 코드가 구현되어 있다.

96.앞선 문제에서 arm 디렉토리 내부에 대해 설명하도록 하시오

97.리눅스 커널 arch 디렉토리에서 c6x 가 무엇인지 기술하시오.

: TI 의 DSP

98.Intel 아키텍처에서 실제 HW 인터럽트를 어떤 함수를 가지고 처리하게 되는지 코드와 함께 설명하시오.

99.ARM 에서 System Call 을 사용할 때 사용하는 레지스터를 적으시오.

:

100.벌써 2 개월째에 접어들었다. 그동안 굉장히 많은 것들을 배웠을 것이다. 상당수가 새벽 3 ~ 5 시에 자서 2 ~ 4 시간 자면서 다녔다. 또한 수업 이후 저녁 시간에 남아서 9 시 ~ 10 시까지 공부를 한 사람들도 있다. 하루 하루에 대한 자기 자신의 반성과 그 날 해야할 일을 미루지는 않았는지 성찰할 필요가 있다. 그 날 해야할 일들이 쌓이고 쌓여서 결국에는 수습하지 못할정도로 많은 양이 쌓였을 수도 있다.

사람이란 것이 서 있으면 앉고 싶고 앉으면 눕고 싶고 누우면 자고 싶고 자면 일어나기 싫은 법이다. 내가 정말 죽을둥 살둥 이것을

이해하기 위해 열심히 했는지 고찰해보자! 2 개월간 자기 자신에 대한 반성과 성찰을 수행해보도록 한다.

(맨 처음 과정에 들어왔을 때의 모습과 현재의 모습에 어떠한 발전이 있었는지 등등을 자세히 기술하도록 한다)

또한 앞으로는 어떠한 자세로 임할 것인지 작성하시오.

: 처음과 현재의 모습에 어떠한 발전이 있었는지에 대해 기술하자니 제자신이 너무 놀았다 정도밖의 수준으로 보입니다.

다른분들은 위의 말씀처럼 밤 늦게까지 공부하며 잠도 잘 못이루며 하루하루 열심히 하였다는데 정작 저 본인은 그렇게 죽을둥 살둥 하며

하루하루의 목표치를 못이룬거 같습니다. 처음과 비교하자면 처음 수업시간때 보다 코딩을 이해하는 편이 나아지긴 했으나, 여전히

본인 스스로 프로그래밍 해 나아가기에 역부족인걸 늘 느낍니다. 예전엔 하루하루 수업을 이해하려고 책만 보고 자료만 찾아보는 정도에도

버거워하여 그정도로만 만족하였는데 앞으로는 찾아보는거 그 이상으로 분발해야겠다고 생각하고 있습니다. 사실 몇번 빠지기도 하고

지각도 하면서 수업에 대한 이해가 몇번 놓치니깐 따라가기가 점점 멀어지긴 했는데, 요즘은 다른 생각 안하고 초심의 마음으로 그날그날의 수업을 어느정도 이해하고 기록해두고 1 개월차 수업들을 차근차근 혼자 다져가는 중입니다. 1 개월차 시험을 보았을때 느낀게 확실히

본인 스스로가 직접 코딩을 작성해보며 이것저것 시도해 보는것이 그게 제일 실력을 다지기에 좋은거 같다고 생각하여 현재 그렇게 진행

중입니다. 하지만, 제가 다른분들에 비해 많이 뒤쳐짐 있음을 알고 있기에 현재의 방법으로만 유지하면 안되는걸 인지하고 앞으로는

어떻게 지금 보다 스스로를 발전시키고 잘 따라갈수 있을까 고민하며 시행해볼 계획입니다. 또한 저 자신은 잠을 줄여가며, 또는 개인시간에 공부를 더 많이 할애하거나 그렇게 하지 않았다는 점이 가장 반성할 점이 아닌가 싶습니다. 몸이 게을러지니 마음도 게으르게 되는것만

같아, 꾸준한 공부를 위해 제 자신의 건강도 신경쓰며 더욱 공부에 매진할 수 있도록 방법을 강구하고 있습니다. 이번 2 개월차 시험이

끝나면 당연히 3 개월차 수업이 진행될텐데, 수학도 현재 다 까먹고 미적분까지 배우질 않아서 그 또한 걱정이고 전자회로도 완전 머릿속에 없어서 걱정이지만, 저 자신은 스스로에게 포기하지 않고 끝까지 차근차근 버티면 뭐라도 남을거고 그또한 실력이 쌓일거라고 믿습니다.

물론, 다른분들에 비해 실력이 월등히 떨어질거고 매 수업도 지금처럼 버거울 순 있지만, 그래도 제가 버틸수 있고 행복한건 제가 하고싶은 꿈에 다가갈수 있도록 이 과정을 듣고있다는 점입니다. 이 과정을 쓸모없게, 또는 습득없이 이렇게 표현되지않도록 앞으로 현재 진행중인 방법으로 기초를 계속 다지면서 차근차근 따라잡으려고 할것입니다. 제가 1 개월차 과정을 지나고 시험을 보며 저의 실수는 이 과정에 죽을둥 말둥 하며 매달리는 마음이 부족한 것과 공부에 시간할애가 분명 다른분들보다 적었다는 점이었습니다. 2 개월차 시험 또한 낮은 점수이며 이해가 적겠지만, 남들보단 한템포 느리게 쫓아가도 차근차근 제가 습득하며 가는것이 현재는 저의 목표입니다.

잘하는 일은 아니겠지만, 매 수업만 따라가려 애쓰다 전의 배운 과정을 제것으로 만들지 않으면 머리속에 남질 않는거 같아서 일단은 하루하루 전의 과정을 습득하며 진행하고, 앞으로의 과정인 공업수업을 대비하여 매주 수학을 공부하고 있어야 하는 것이 좀더 낫겠다고 생각하고 있습니다. 1 개월차 반성때와 2 개월차때의 반성의 차이점은 저의 현재수준을 알고 행동하는 것. 그것인거 같습니다. 앞으로는 계속 저의 수업참여도와 꾸준한 공부를 더욱 다듬고 늘려가며 실력향상을 할 계획입니다. 이상으로, 단톡방에서 강사님께서 용기를 주시는 격려를 항상 되새기며 포기하지 않고 더욱 열심히 하겠습니다!!!