2018. 4. 17 목 – 2 회차 시험

과정 : TI, DSP, Xilinx Znq FPGA, MCU 기반의 프로그래밍 전문가 과정

Prof. 이상훈
gcccompil3r@gmail.com
Stu. 정상용
fstopdg@gmail.com

2. 369
A.

```
#include <stdio.h>
#include <unistd.h>
#include <signal.h>

void sighand(int signo)
{
	printf("\nclap!\n");
}

int main(void)
{
	int num, i;

	printf("Let's play the game, 369\n");

	for(i = 1; i < 100; i++)
	{
		printf("%d\n", i);
		if(i % 3)
		{
			scanf("%d", &num);

			if(i == num)
				printf("next\n");
			else
			{
				printf("lose\n");
				break;
			}
		}
		else if(i % 3 == 0)
		{
			signal(SIGINT, sighand);
			pause();

		}

		else
```

```
                {
                        printf("lose\n");
                        break;
                }

                alarm(2);
        }

        return 0;
}
```

3. OS 가 관리해야 하는 제일 중요한 5 가지
A. Task manager, Memory manager, Filesystem manager, Driver manager, Network manager

4. Unix 계열의 모든 OS 는 모든 것을 무엇으로 관리하는가
A. file

5. 장점에 대한 설명
→ 사용자 임의대로 재구성이 가능하다. : 커널을 수정할 능력이 되는 개발자인 경우 소스코드를 변경할 수 있다.
→ 열악한 환경에서도 HW 자원을 적절히 활용하여 동작한다. : 최소한의 HW 자원을 최적화시켜 동작한다.(추가로 고성능의 HW 를 이용해도 최고의 성능을 발휘한다. e.g 구글의 서버도 리눅스로 구성)
→ 커널의 크기가 작다. : 다른 운영체제에 비하여 작을 뿐…
→ 완벽한 멀티유저, 멀티태스킹 시스템
→ 뛰어난 안정성 : 이러한 이유로 네트워크나 서버를 리눅스로 관리(e.g 구글의 서버)
→ 빠른 업그레이드 : 업그레이드 버전이 빨리 나온다.(우리에게는 단점일수도...)
→ 강력한 네트워크 지원 : 뛰어난 안정성으로 인하여 가능
→ 풍부한 소프트웨어 : 리눅스관련된 다양한 소프트웨어가 존재

6. 32bit system 에서 User 와 Kernel 의 Space 구간을 적으시오
A. Krenel : 1GB, User : 3GB

7. Page Fault 발생시, 운영체제의 동작.
A. 운영체제의 경우, Page Fault 가 난 장소로 재접근하여 값을 넣어준 다음 다시 접근.

8. 리눅스 실행 파일 포맷
A. ELF(cf. 디버그 파일 포맷 : DWARF)

9. 프로세스와 스레드를 구별하는 방법
A. pid & tgid 값으로 구별
→ pid = tgid : Process OR thread leader
→ pid != tgid : thread

10. Kernel 입장에서 Process 혹은 Thread 를 만들면 무엇을 생성
A. task_struct

11. curret??
A. 현재 동작하는 task 를 가리키는 매크로
→ task_struct
→ thread_union
→ thread_info 와 관련

12. Memory Management 입장에서 Process 와 Thread 의 핵심적인 차이점
A. Process : parent 와 child 의 관계에 있어서 서로 독립적(메모리를 서로 공유하지 않는다. i.e child process 의 변수가 바뀌어도 parent 는 영향을 받지 않는다.)
Thread : parent 와 child 의 관계에 있어서 서로 종속적(메모리를 서로 공유한다.)

13. Task 가 관리해야하는 3 가지 Context 중 HW context 는?
A. HW context : Context – Switching 을 할 경우 prev task 의 연산내용을 저장해놔야 한다. 그 저장관련된 context 를 말한다.(cpu_context_save 와 관련)
→ task_sturct
→ thrad_union
→ thread_info
→ cpu_context_save

14. Deadline 스케줄링 방식에 대하여 기술
A. 태스크를 크게 2 가지로 분류가능하다. 첫 번째는 실시간 태스크, 두 번째는 동적우선순위 태스크이다.
실시간 태스크를 관리하는 스케줄러로는 FIFO, RR, DEADLINE 이 존재한다. 이 중 하나가 Deadline 이다.
Deadline 스케줄링은 우선순위에 따라 task 를 선택하는 것이 아닌, deadline 에 가까운 task 먼저 수행시키는 방식이다. 예를 들어 영상처리와 같이 주기성을 가지고 동작을 해주는 작업들이 있다.

15. TASK_INTERRUPTIBLE 과 TASK_UNINTERRUPTIBLE 의 필요성
A. TASK_RUNNING(running) 현재 동작하고 있는 TASK 가 어떤 이벤트를 대기해야 할 경우 TASK_INTERRUPTIBLE 또는 TASK_UNINTERRUPTIBLE 상태가 된다. 이벤트를 대기하는 경우 필요한 상태이다. (cf. 차이점 : interrupt 를 받을 수 있는지 없는 지에 대한 유무)

16. O(N)과 O(1) Algorithm 비교
A. O(N)은 시간에 따라 선형적인 함수
O(1)은 시간에 따라 변화하지 않는 상수를 가진다.
충분히 적은 시간에 있어서는 O(N)이 훨씬 효율적이지만, 시간이 충분히 증가한 경우 O(1)이 더 효율적이다.

17.
A. 2 번 CPU 의 RQ 에 할당되야 효율적이다.
Parent task 가 사용하는 캐시를 같이 사용할 수 있게 되어 가장 효율적이라고 볼 수있다.

18.
A. 역시 2 번 CPU 의 RQ 에 할당되어야 효율적이다.
비록 parent task 가 3 번 CPU 의 RQ 에 있지만, 부하가 심하게 걸려있으므로 부하가 낮은 CPU 의 RQ 로 가는 것이 효율적이다.

19. UMA 와 NUMA 에 대한 기술과 Kernel 에서 이들을 어떠한 방식으로 관리하는가.
A. UMA : 단일 node 를 가진 형태이다. 즉, cpu 에 접근하는 뱅크의 속도가 전부 동일하다.
→ contig_page_data 에서 관리(구조체는 pg_data_t)
NUMA : 복수의 node 를 가진 형태이다. 즉, cpu 에 접근하는 뱅크의 속도가 동일 하지 않다.
→ pg_data_t 에서 관리

20. Kernel 스케줄링 메커니즘에서 Static Priority 와 Dynamic Priority 번호가 어떻게 되는지 적으시오
A. Static Priority : 0~99
Dynamic Priority : 100 ~ 139

21. ZONE_HIGHMEM 에 대하여 서술하시오
A. 커널의 입장에서 구조체 pglist_data 안에는 뱅크를 관리하는 node 가 존재한다. 이 node 는 다시 zone 으로 나누어지고 zone 의 종류에는 ZONE_DMA, ZONE_NORMAL, ZONE_HIGHMEM 등이 존재한다. 여기서 ZONE_HIGHMEM 은 32bit 에만 존재하고, 896MB 이상의 가상메모리일 경우 간접참조를 통하여 물리메모리와 연결시킨다.

22. 물리메모리의 최소단위의 이름과 크기, 이 개념을 SW 적으로 구현한 구조체이름
A. Page frame, 4KB, Page

23. Buddy allocator 에 대하여 서술
A. 페이지 프레임의 단위는 4KB 이다. 이 보다 훨씬 큰 데이터가 들어왔을 경우는 Buddy allocator 가 페이지프레임 여러개를 합쳐서 관리하게 된다. (4KB, 8KB, 16KB…) free_area.map 과 free_list 를 통하여 실제 물리메모리를 할당하게 된다. free_area 를 통하여 4KB, 8KB, 16KB, 32KB 단위로 데이터가 들어갈 수 있는지 판단하여 free_list 에 작성하여 데이터가 들어오는 크기에 따라 실제 물리메모리에 할당해준다.

24. Slab allocator
A. Buddy allocator 와 반대로 너무 작은 크기의 데이터가 물리메모리에 할당될 경우, 4KB(페이지 프레임의 단위)는 비효율적이게 된다. 이를 막기 위하여 4KB 를 나누어서 작은 크기의 데이터를 할당하는데 사용한다.

25. VM 의 Stack, Heap, Date, Text
A. → task_struct
→ mm_struct (메모리와 관련된 구조체)
→ vm_area_struct (가상메모리와 관련된 구조체)

26. Page 를 모아서 관리하기 위한 구조체
A. zone
→ task_sturct 에서 :cs find 0 pglist_data
→pglist_data
→ zone
27. a.out 실행시 Text 영역에서 무엇을 읽고 이 영역을 적절한 값으로 채워주는가?
A. 기계

28. User 어 space 와 Kernel space 의 메모리 기본 구성에 대한 이유서술
A. User space 의 경우, 프로그램의 지역변수, 동적할당, 전역변수, 기계어를 저장하기 위하여 필요하다.
Kernel space 의 경우를 생각해보자. Kernel 도 사실 따지고 보면 결국은 프로그램이다. 프로그램을 동작시키려면 stack 이 필요하다. 다시말해 Kernel 도 프로그램이므로 이 정보를 저장하기 위한 Stack, Heap, Data, Text 가 필요한 것이다.

29. Paging 에 관하여 서술
A. VM 에 존재하는 변수들과 정보들을 pgd 를 통하여 세 번의 table 을 넘어서 물리메모리에 적어주게 된다. 이 동작을 Paging 이라고 한다.

30. MMU 란
A. CPU 에 존재하는 하드웨어의 한 부분으로 가상메모리를 물리메모리로 바꾸어주는 역할을 한다.

31. 하드디스크의 최소단위
A. Disk block

32.

33.

34. super_block 의 역할
A. super_block 은 최상위 directory 이다. filesystem 으로 원하는 파일을 찾을 때 super_block 을 통하여 최상위디렉토리부터 찾아갈 수 있도록 한다.

35. VFS 동작 메커니즘을 서술하시오
A. VFS 는 가상파일시스템으로 사용자입장에서의 file system 이다.
사용자가 어떤 파일을 호출하였을 때, 가상파일시스템에서 수퍼블록을 찾아 그 파일이 어떤 파일시스템을 가지고 있는지 파악을 한다. 그 다음 정보를 담을 구조체 생성한다. 파일시스템은 inode 를 찾아 파일의 내용을 읽어와 아까 생성한 구조체에 넣은 뒤 사용자에게 넘겨준다.

36. 인터럽트에 대하여 서술하시오.
A. 인터럽트는 크게 외부인터럽트와 내부인터럽트로 나누어진다.
외부인터럽트는 CPU 를 기준으로 하여 pin 을 통하여 들어오는 인터럽트이다.
내부인터럽트는 CPU 내부에서 일어나는 인터럽트이다.

37. 내부인터럽트에 대하여 설명하시오.
내부인터럽트는 다시 한 번 fault, trap, abort 로 나누어 질 수 있다.

38. fault, trap, abort 에 대하여 서술하시오
fault : page fault 나 segmentation fault 와 같이 해당하는 장소에 접근권한이 없을 경우 생기는 인터럽트이다. fault 가 발생했을 경우에는 명령어 주소를 레지스터(eip)에 저장하였다가 핸들러가 종료되면 다시 레지스터에 저장된 주소를 실행한다.
Trap : system call 과 같이 유일한 SW Intrerrupt 이다. fault 와 차이점으로는 레지스터에 다음에 실행할 주소를 저장한다.
Abort : 태스크를 강제 종료(비정상종료로 인식)시킨다고 보면된다.

39. 모니터를 3 개 사용하는 경우
A. 인터럽트가 발생한 경우 IDT(Interrupt Descriptor Table)을 통하여 모니터가 해당하는 인덱스를 확인한다. 해당 인덱스를 확인한 뒤 do_IRQ() 함수를 통하여 irq_desc 에서 주번호(모니터)와 부번호(모니터갯수)를 이용하여 관리한다.

40.
A. IDT

41.
A. intel : IDT 에서 인덱스 128 에 저장

42.
A. pgd

43.
A.

44.
A. 커널도 하나의 프로그램이다. 그러므로 커널도 스택이 필요하므로 메모리 할당이 필요하다.

45.
A.

46.
A.

47. Mutex 와 Semaphore 의 차이점
A. 둘 다 critical section 을 방지하기 위하여 lock 을 걸어주는 방법이다. 세마포어는 큰 개념의 뮤텍스라고 볼 수 있다. 뮤텍스의 경우 하나에만 lock 을 걸수 있지만, 세마포어는 하나이상 걸수 있다.

48. module_init() 함수 호출은 언제 이루어지나요.
A. 명령어 insmod 동작시

49. module_exit() 함수 호출은 언제 이루어지나요.
A. 명령어 rmmod 동작시

50. thread_union 에 대하여 서술하시오.
A. 태스크당 할당되는 커널스택을 thread_union 이라고 한다.
→ thread_union
→ stack
→ thread_info
이 존재한다.
→ thread_info
→ *task //task_struct 를 가리키는 포인터
→ flags // flags == 1 인 경우, 리스케줄
→ cpu_context_save // context-switching 을 할 경우, 이전 task 의 내용을 저장

51. Major number 와 Minor number 를 관리하는 구조체를 찾으시오.

61. Hyper Threading
A. 하드웨어적인 기술로 실제 CPU 갯수의 2 배의 효과를 가진다.

63. 파일의 종류를 확인하는 프로그램.

```
#include <stdio.h>
#include <sys/stat.h>
#include <pwd.h>
#include <grp.h>

int main(int argc, char **argv)
{
   struct stat buf;
   struct passwd *pw;
   struct group *gr;
   char ch;
   char perm[11] = "----------";
   char rwx[4] = "rwx";
   int i;
   stat(argv[1], &buf);
```

```
    if(S_ISDIR(buf.st_mode))
      perm[0] = 'd';
    if(S_ISREG(buf.st_mode))
      perm[0] = '-';
    if(S_ISFIFO(buf.st_mode))
      perm[0] = 'p';
    if(S_ISLNK(buf.st_mode))
      perm[0] = 'l';
    if(S_ISSOCK(buf.st_mode))
      perm[0] = 's';
    if(S_ISCHR(buf.st_mode))
      perm[0] = 'c';
    if(S_ISBLK(buf.st_mode))
      perm[0] = 'b';
    for(i=0; i < 9; i++)
       if((buf.st_mode >> (8-i)) & 1)
          perm[i+1] =rwx[i%3];
    printf("%s ", perm);
    printf("%ld ",buf.st_nlink);
    pw = getpwuid(buf.st_uid);
    printf("%s ", pw -> pw_name);
    gr = getgrgid(buf.st_gid);
    printf("%s\n", gr -> gr_name);
    return 0;
}
```

66. 자신의 리눅스 커널의 번전을 확인, 다운로드 후 확인

A. terminal 에서 uname -r 명령어를 통하여 자신의 리눅스 커널의 버전을 확인

다운로드 받는 과정

→ 다운로드 받을 디렉토리 생성

```
mkdir kernel
```

→ 생성한 디렉토리에서 다운로드 실행

```
cd kernel
wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.4.tar.gz
```

→ 압축해제

```
tar zxvf linux-4.4.tar.gz
```

→ ctags 와 cscope 설치(현재 다운받은 linux-4.4 디렉토리 내부에서)

```
sudo apt-get install ctags cscope
```

→ ..vimrc 와 mkcscope.sh 파일생성후 코드 작성

→ tags 설정(linux-4.4 디렉토리 내부에서)

```
ctags -R
```

→ cscope 설정

```
chmod 755 ~/mkcscope.sh
sudo ~/mkcscope.sh /usr/local/bin
```

→ 완료 : /home/kernel/linux-4.4 에서 task_struct 오픈

```
vi -t task_struct
```

67. 멀티태스킹에 대한 설명

A. CPU 1 개는 1 의 태스크만 동작시킬 수 있다. 그리하여 런큐에 있는 태스크 중 1 개만 동작을 시키고, 나머지는 WAIT 큐에서 태스크들이 기다리고 있는 다고 보면 된다. 스케줄링에 따라서 현재 동작중인 태스크가 종료가 되거나 wait 큐로 갈 경우, 문맥교환이 일어나게 된다.

68. 현재 삽인된 디바이스 드라이버의 리스트를 보는 명령어
A. lsmod

94. User 영역에서 fork()를 수행할 경우, 일어나는 동작을 서술버
A. User 에서 fork()를 수행 → library 에서 fork()를 수행 → system call 로 인하여 clone()을 호출
system call 로 인하여 kernel 내부로 진입하여 sys_clone()호출 → do_fork()수행

95. arch 디렉토리에 대한 설명
A. 리눅스 커널내부에는 task 와 관련된 directory 가 2 개 존재한다. Kernel 과 arch 이다. Kernel directroy 의 경우에는 CPU 의 종류에 관계없이 task 를 관리하는 내용이 들어있다 . arch 의 경우에는 CPU 종류에(e.g arm, arm64, powerpc, sparc, i386, x86...) 따라 task 를 관리하는 방법이 바뀌는 내용이 들어있다.(e.g Context-switching)

96. arm 디렉토리 내부에 대해 설명
A. 리눅스커널 디렉토리로 진입 : /home/kernel/linux-4.4(본인기준)
arch 에서 arm 으로 으로 진입 : /arch/arm
mach – xxx : arm 을 사용하는 회사별로 나열(xxx : 회사이름)
plat – yyy : arm 을 사용하는 회사 중 규모가 큰 회사만 나열(yyy : 회사이름)
e.g
mach – davinci : 영상관련 분야 3 대장 중 하나
mach – exynos : 안드로이드 핸드폰에 들어감
mach – omap1 : TI
mach – omap2 : TI
mach – s3c24xx : 삼성
mach – zynq : Xilinx
plat – samsung : 삼성의 위엄(?)

97. 리눅스커널 arch 디렉토리에서 c6x 는 무엇인가
A. 이것은 바로 우리가 사용할 TI DSP

98.

99.

100.
A. 선생님의 말씀대로 2 개월이라는 시간이 벌써 흘렀다. 그 동안 C, 자료구조, 시스템 프로그래밍, 네트워크, 커널이라는 5 가지 과목에 대하여 배웠었다. 항상 한 과목, 한 과목이 지날때마다 주말에 정리를 해야지, 다음 과목 배울때 조금씩 정리해나가야지의 생각을 계속 가지고 있었다.  27 년간 가지고 있던 미루는 습관을 버리지 못하고 또 다시 반복하고 있었다. 결론적으로 지금까지 배운 5 과목 중 어느 하나도 완벽히 숙지를 못하고 있는 결과가 나오게 되었다. 실제 프로젝트기간에 들어가서 이론내용을 몰라서 다시 공부하고 있고 싶지는 않다. 이제부터 배우는 내용은 그 기간에 어떻게든 끝내도록 노력해야겠다. 더 이상 공부할 양을 쌓아둘 경우 돌이킬 수 없는 상황이 올 것 만같다. 아니, 올 수 밖에 없을 것이다. 그 날 배운 내용은 그 날 끝내겠다는 마음가짐으로 하루하루를 지내야겠다.