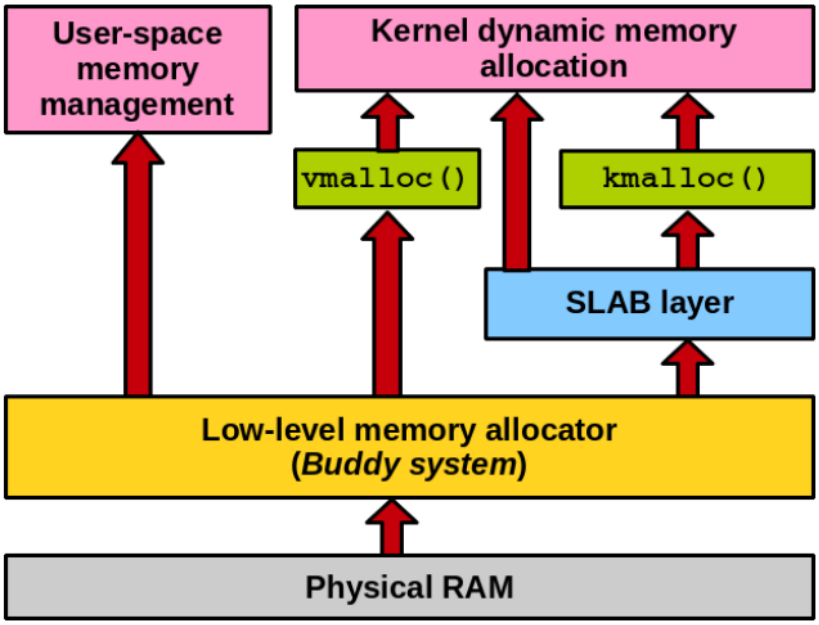
Embedded System Software [CSE4116]

실습 5 주차 : Kernel Memory Allocation & Time Management

Department of Computer Science and Engineering, Sogang University, Seoul, South Korea

Data-Intensive and Computing and System Laboratory

1. Kernel memory allocation



Kmalloc()	Vmalloc()
Physical memory 에서 연속적인 memory 공간을 할당	Kernel virtual address space 에서 연속적인 공간을 할당. Physical memory 에서 연속적인 공간 할당을 보장하지 않음. 큰 memory 공간을 할당할 때 이점이 있음.
Vmalloc()에 비해 빠름	Virtual memory 를 physical memory 에 mapping 하는 overhead 가 있기때문에 Kmalloc()보다 느림.
Interrupt context 에서 사용 가능	Interrupt context 에서 사용 불가

1.1. Function Form

- void *kmalloc(size_t size, gfp_t flags)

  Flags : GFP_KERNEL, GFP_ATOMIC, GFP_DMA, GFP_USER
- void kfree(const void *addr)

- void *vmalloc(unsigned long size)
- void vfree(const void *addr)

2. Time management

2.1. Timer Interrupt

- <include/asm/param.h>
- HZ : 1 초당 발생하는 타이머 인터럽트 수 (#define HZ 1000)

2.2. jiffies, jiffies_64, get_jiffies_64()

- jiffies : kernel 2.4 에서 초당 HZ 값만큼 증가하는 전역 변수
- jiffies_64 : kernel 2.6
- jiffies 값은 1/HZ 초 간격으로 1 씩 증가한다.

2.3. Delaying Executing

- (short delays) mdelay(), udelay(), ndelay()
- (long delays) jiffies, HZ 를 이용한 실행지연

2.4. Setting / Getting System Time

- void do_gettimeofday(struct timeval *tv)
- int do_settimeofday(struct timespec *tv)
- (unsigned long)mktime(year, month, day, hour, minute, second)

2.5. Kernel Timer

```
#include <linux/timer.h>
struct timer_list {
	…
	unsigned long expires;  // 만료 시간
	void (*function) (unsigned long); //만료시 호출 함수
	unsigned long data; //호출함수에 넣어지는 argument
}

void init_timer(struct timer_list *timer);
void add_timer(struct timer_list * timer);
int del_timer(struct timer_list * timer); //타이머 제거, interrupt context 내 사용가능
int del_timer_sync(struct timer_list * timer); //다른 프로세서에서 타이머 핸들러 실행 중일 경우, 핸들러 종료될 때까지 기다린 후 제거, interrupt context 내 사용불가
int mod_timer(struct timer_list * timer, unsigned long expires);
```

3. 실습

3.1. Remind

- 로그 레벨 변경

```
$ echo "7 6 1 7" > /proc/sys/kernel/printk
```

3.2. Kernel memory allocation

- 제공된 kernel_memory.tar 파일 압축해제 후 make 수행
  mem_ctrl 모듈은 write 시 user 로 부터 받은 데이터를 kmalloc 을 사용하여 저장하고 read 시 kfree 를 사용하여 할당했던 공간을 해제함. mem_ctrl.c 코드를 이해할 것.

```
ssize_t mem_ctrl_write(struct file *inode, const char *gdata, size_t length, loff_t
*off_what) {

   ptr = kmalloc(sizeof(char) * length, GFP_KERNEL);
   if (copy_from_user(ptr, gdata, length)) {
      return -EFAULT;
   }
   return 1;
}

ssize_t  mem_ctrl_read(struct  file  *inode,  char  *data,  size_t  length,  loff_t
*off_what) {
   if(copy_to_user(data, ptr, length)){
      return -EFAULT;
   }
   kfree(ptr);
   return 1;
}
```

- test_mem.c 파일을 크로스 컴파일
  test_mem 은 2^20 크기의 input data 를 write 하며 write 전후 시스템 메모리 크기를 측정하여 차이를 출력함. test_mem.c 코드를 이해할 것.

```
int main(int argc, char **argv)
{
…
   request_size = SIZE;
   memory_size = get_system_memory();
   write(fd, &input, request_size);
   printf("Request Size : %lu, Diff System Memory: %llu \n", SIZE, memory_size -
get_system_memory());
   off_t loc = lseek(fd, 0, SEEK_SET);
   read(fd, &output, request_size);
…
}
```

get_system_memory 함수는 시스템의 메모리 크기를 반환한다.

```
unsigned long long int get_system_memory()
{
    long pages = sysconf(_SC_AVPHYS_PAGES);
    long page_size = sysconf(_SC_PAGESIZE);
    return pages * page_size;
}
```

- 컴파일된 모듈 mem_ctrl.ko 파일과 teat_mem.c 바이너리 파일()을 보드로 전송
- 모듈 insmod 후 test 바이너리 파일 실행 시 다음과 같이 출력됨.

```
root@achroimx:/data/local/tmp # insmod mem_ctrl.ko
[ 1751.873485] dev_file: /dev/mem_ctrl , major: 246
[ 1751.878120] init module
root@achroimx:/data/local/tmp # ./a.out
Request Size : 1048576, Diff System Memory: 1048576
```

3.3.Kernel timer

- 제공된 kernel_timer.tar 파일 압축해제 후 make 수행
- 컴파일된 모듈(kernel_timer.ko)과 test 프로그램(kernel_timer_test)를 보드로 전송
- insmod 와 mknod 이후 test 프로그램 실행. 다음과 같이 출력됨.

```
$ insmod kernel_time.ko
$ mknode /dev/kernel_timer c 245 0
$ ./kernel_timer_test 10
```

```
root@achroimx:/data/local/tmp # ./kernel_timer_test 10
[679643.753647] kernel_timer_open
get 10[679643.757132] write

[679643.759604] data  : 10
[679643.762364] kernel_timer_release
root@achroimx:/data/local/tmp # [679644.760152] kernel_timer_blink 10
[679645.760156] kernel_timer_blink 11
[679646.760159] kernel_timer_blink 12
[679647.760152] kernel_timer_blink 13
[679648.760147] kernel_timer_blink 14
[679649.760157] kernel_timer_blink 15
```

약 1초 간격

4. 실습 과제

- mem_ctrl 모듈과 test 프로그램을 수정하여 2^20 크기보다 작은 크기와 큰 크기를 각각 kmalloc()으로 요청했을때 시스템 메모리 차이를 통해 실제로 어느정도 크기의 메모리가 할당되었는지 확인한다.
- 모듈을 다음과 같이 수정한다. 코드를 잘못 작성하면 kernel panic 이 발생할 수 있으므로 주의할 것.
  - char *ptr 를 이중 포인터 변수로 바꾼다.
  - 2 개 크기를 갖는 pointer 배열을 사용하기 위해, mem_ctrl_open()시 2 개의 포인터(char *) 크기를 kmalloc 한다. 배열 인덱스 관리를 위한 변수 idx 를 초기화 한다.
  - mem_ctrl_write()와 mem_ctrl_read()에서 인덱스 변수 idx 를 관리한다.
  - mem_ctrl_release()에서 이중포인터 ptr 을 완전히 kfree()한다.
- Test program 을 다음과 같이 수정한다.
  - 2^20 보다 자신의 학번 4 자리 만큼 작은 크기를 write 했을 때 시스템 메모리 크기의 차이를 출력하고 read 한다.
  - 이후 2^20 보다 자신의 학번 4 자리 만큼 큰 크기를 write 했을 때 시스템 메모리 크기의 차이를 출력하고 read 한다.
- 2 의 거듭제곱으로 메모리가 할당되는 것을 확인할 수 있다. 출력 예시는 다음과 같다.

```
root@achroimx:/data/local/tmp # ./a.out
Request Size : 1048576 - 1234, Diff System Memory: 1048576
Request Size : 1048576 + 1234, Diff System Memory: 2097152
root@achroimx:/data/local/tmp #
```

- **제출물**
  - Test program 수행 시 출력 되는 내용을 캡처한 이미지 파일
  - 수정된 mem_ctrl.c, test_mem.c 파일
  - 위 파일들을 tar 로 압축하여 학번_이름.tar 로 제출 (ex: 120221234_홍길동.tar)
    (이름은 영어로 해도 상관 없음)