TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/4/23
수업일수	43 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

1. sys_fork()

copy_process 내부 (3)

```
        retval = security_task_create(clone_flags);
        if (retval)
                goto fork_out;

        retval = -ENOMEM;
        p = dup_task_struct(current);
        if (!p)
                goto fork_out;
```

clone_flags = 17 을 인자로 security_task_create 함수 실행

security_task_create 내부

```
int security_task_create(unsigned long clone_flags)
{
        return call_int_hook(task_create, 0, clone_flags);
}
```

task_create 는 연결리스트 구조임. list_head 구조체로 구성됨.
(task_create, 0, clone_flags)를 인자로 가지고 함수 실행.

call_int_hook 내부 구조

```
#define call_int_hook(FUNC, IRC, ...) ({                        \
        int RC = IRC;                                           \
        do {                                                    \
                struct security_hook_list *P;                   \
                                                                \
                list_for_each_entry(P, &security_hook_heads.FUNC, list) { \
                        RC = P->hook.FUNC(__VA_ARGS__);         \
                        if (RC != 0)                            \
                                break;                          \
                }                                               \
        } while (0);                                            \
        RC;                                                     \
})
```

(FUNC, IRC, …)
… → 가변인자를 받음
RC = 0;

security_hook_list 구조

```
/*
 * Security module hook list structure.
 * For use with generic list macros for common operations.
 */
struct security_hook_list {
        struct list_head                list;
        struct list_head                *head;
        union security_list_options     hook;
};
```

list_head 구조체를 가지는 list 와 *head, 함수포인터 다발을 요소로 하는 security_list_options 이라는 구조체인 hook 을 요소로 가지는 구조체를 가지는 *P 를 선언함.

list_for_each_entry 구조

```
/**
 * list_for_each_entry  -       iterate over list of given type
 * @pos:        the type * to use as a loop cursor.
 * @head:       the head for your list.
 * @member:     the name of the list_head within the struct.
 */
#define list_for_each_entry(pos, head, member)                          \
        for (pos = list_first_entry(head, typeof(*pos), member);        \
             &pos->member != (head);                                    \
             pos = list_next_entry(pos, member))
```

list_for_each_entry(P, &security_hook_heads.FUNC, list) == llist_for_each_entry(pos, head, member)

list_first_entry 구조

```
/**
 * list_first_entry - get the first element from a list
 * @ptr:        the list head to take the element from.
 * @type:       the type of the struct this is embedded in.
 * @member:     the name of the list_head within the struct.
 *
 * Note, that list is expected to be not empty.
 */
#define list_first_entry(ptr, type, member) \
        list_entry((ptr)->next, type, member)
```

P 변수 연결리스트의 첫번째 요소에서 다음 요소를 가리키게 함.

list_entry 구조

```
/**
 * list_entry - get the struct for this entry
 * @ptr:        the &struct list_head pointer.
 * @type:       the type of the struct this is embedded in.
 * @member:     the name of the list_head within the struct.
 */
#define list_entry(ptr, type, member) \
        container_of(ptr, type, member)
```

container_of 구조

```
/**
 * container_of - cast a member of a structure out to the containing structure
 * @ptr:        the pointer to the member.
 * @type:       the type of the container struct this is embedded in.
 * @member:     the name of the member within the struct.
 *
 */
#define container_of(ptr, type, member) ({                      \
        const typeof( ((type *)0)->member ) *__mptr = (ptr);    \
        (type *)( (char *)__mptr - offsetof(type,member) );})
```

container_of 매크로는 구조체 멤버의 포인터를 이용하여 구조체의 시작 주소를 찾는 역할을 한다.
security_hook_list 의 타입으로*__mptr 을 선언하고 ptr 을 대입한다. *__mptr 에서 type 구조체의 member 의 offsetof 매크로를 이용하여 값을 빼면 시작주소를 얻을 수 있음

list_next_entry 구조

```
/**
 * list_next_entry - get the next element in list
 * @pos:        the type * to cursor
 * @member:     the name of the list_head within the struct.
 */
#define list_next_entry(pos, member) \
        list_entry((pos)->member.next, typeof(*(pos)), member)
```

위의 list_entry 와 동일

call_int_hook 를 다시 보면

```
#define call_int_hook(FUNC, IRC, … )
({
Int RC = IRC;
for(pos= list_first_entry(head, typeof(*pos), member) ; &pos->member != (head) ; pos = list_next_entry(pos, member) )
{
        RC = P->hook.FUNC(__VA_ARGS__);
        if(RC!=0)
                break;
}
RC;
})
```

P->list 가 &security_hook_heads.task_create 가 아닐 때 까지 pos 에 다음 엔트리를 넣으며 비교하여 반복한다.
for 문을 반복하며 RC 에 P->hook.task_create 를 대입하여 pos->member 의 주소가 head 와 같은 것을 찾을 때 까지 반복하고 만약 연결리스트의 끝까지 가서도 head 를 찾지 못한다면 RC = 0 이 될 것이고 do while 문을 나가게 된다.

copyprocess(3) 내부

```
        retval = security_task_create(clone_flags);
        if (retval)
                goto fork_out;

        retval = -ENOMEM;
        p = dup_task_struct(current);
        if (!p)
                goto fork_out;
```

copyprocess 로 돌아와 보면 retval 은 0 값이 들어오게 되어 if 문을 만족하지 않고 retval = - ENOMEM = -12 가 될 것이다.

task_struct *p 변수는 현재 태스크를 인자로 하는 dup_task_struct 함수의 리턴 값을 받을 것이다.

dup_task_struct 내부(1)

```
static struct task_struct *dup_task_struct(struct task_struct *orig)
{
        struct task_struct *tsk;
        struct thread_info *ti;
        int node = tsk_fork_get_node(orig);
        int err;
```

현재의 task_struct 를 인자로 가짐(부모 태스크) = orig
tsk 라는 task_struct, ti 라는 thread_info 라는 구조체 변수 선언.
정수 값을 가지는 node 는 tsk_fork_get_none(orig)를 실행.
현재 태스크의 fork 로 만든 태스크의 노드를 얻음

tsk_fork_get_none 내부

```
/* called from do_fork() to get node information for about to be created task
 */
int tsk_fork_get_node(struct task_struct *tsk)
{
#ifdef CONFIG_NUMA
        if (tsk == kthreadd_task)
                return tsk->pref_node_fork;
#endif
        return NUMA_NO_NODE;
}
```

orig == tsk

NUMA 이므로 if 문 실행
khread_task 는 tsk(orig)와 같은 구조체 task_struct 이므로 if 문 만족.

pref_node_fork 는 NUMA 구조일 때 task_strcut 내의 short 형 변수임. 어떤 값이 들었는지는 모르지만 이 값을 리턴.(이전 노드?)

dup_task_struct 내부(2)

```
        tsk = alloc_task_struct_node(node);
        if (!tsk)
                return NULL;
```

alloc_task_struct_node

```
static inline struct task_struct *alloc_task_struct_node(int node)
{
        return kmem_cache_alloc_node(task_struct_cachep, GFP_KERNEL, node);
}
```

2. 오답노트

31. 하드디스크의 최소 단위를 무엇이라 부르고 그 크기는 얼마인가 ?

Sector, 512 byte

43. 또한 Page Directory를 가르키는 Intel 전용 Register가 존재한다. 이 Register의 이름을 적고 ARM 에서 이 역할을 하는 레지스터의 이름을 적으시오.

Intel 의 경우엔 CR3

ARM 의 경우엔 CP15

44. 커널 내부에서 메모리 할당이 필요한 이유는 무엇인가 ?

os 또한 프로그램이기에 프로그램이 실행되기 위해서는 스택이 필요하여 메모리 할당이 필요. ARM 인 경우 8K 의 메모리 공간을 할당 받음.

45. 메모리를 불연속적으로 할당하는 기법은 무엇인가 ?

vmalloc()

46. 메모리를 연속적으로 할당하는 기법은 무엇인가 ?

kmalloc()

97. 리눅스 커널 arch 디렉토리에서 c6x 가 무엇인지 기술하시오.

TI DSP 에 해당하는 Architecture