OS Project 1

OS Team 7

Preprocess

- Change __NR_syscalls in arch/arm/include/asm/unistd.h
  - __NR_syscalls must be divided by four
- Define __NR_ptree in arch/arm/include/uapi/asm/unistd.h
- Add CALL(sys_ptree) in arch/arm/kernel/calls.S
- Declare struct prinfo in include/linux/prinfo.h
- Add Object file(ptree.o) in kernel/Makefile

int ptree(struct print *buf, int *nr)

- Input
  - struct prinfo * buf : Pointer where to copy process tree information
  - int * nr : size of buffer
- Output
  - struct print * buf : process tree information
  - int * nr : number of entries written buf
- Return value
  - success : total number of entries on success (can be different from *nr)
  - buf/nr is null or *nr <= 0 : -EINVAL
  - Can't write to buf/nr : -EFAULT
  - Not enough memory : -ENOSPC

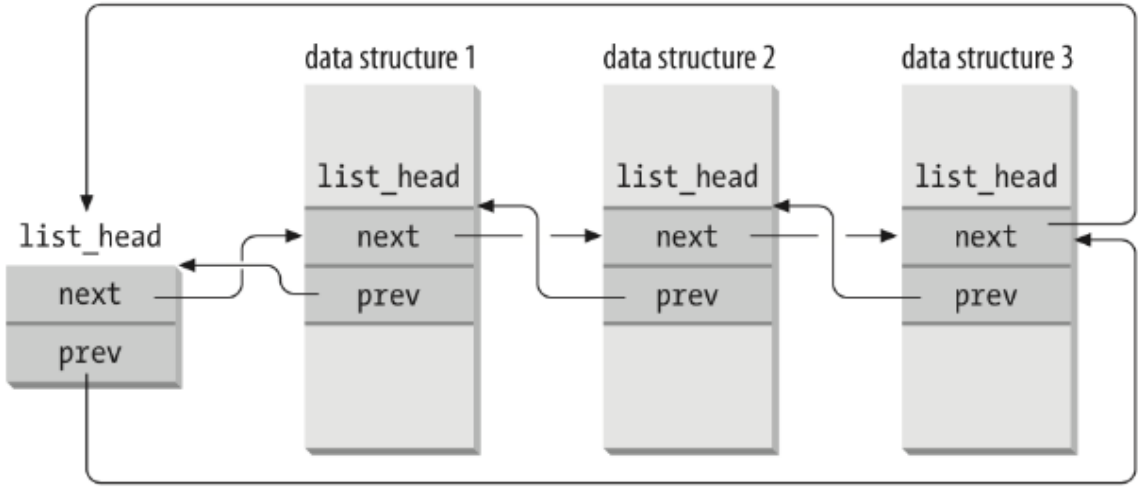
Process infomation

- Are stored with "struct task_struct" type

struct prinfo	struct task_struct
long state	volatile long state;
pid_t pid	pid_t pid
pid_t pid	struct task_struct __rcu *real_parent
pid_t first_child_pid	struct list_head children
pid_t next_sibling_pid	struct list_head sibling
long uid	task_uid(struct task_struct)
char comm[64];	char comm[TASK_COMM_LEN]

What is list_head?

```
struct list_head {
        struct list_head *next, *prev;
};
```




```
#define list_for_each(pos, head)
for(pos = (head)->next; pos != (head); pos = pos->next)

#define list_entry(ptr, type, member)
  container_of(ptr, type, member)

#define container_of(ptr, type, member) ({                  \
  const typeof( ((type *)0)->member ) *__mptr = (ptr);    \
  (type *)( (char *)__mptr - offsetof(type,member) );})
```

ptree_dfs

```
int ptree_dfs(struct task_struct *now) {
        if(now != &init_task && memory enough)
                fill process information

        list_for_each(p, &now->children) {
                t = list_entry(p, struct task_struct, sibling);
                all += ptree_dfs(t)
        }

        return all;
}
```

ptree.c (without checking error)

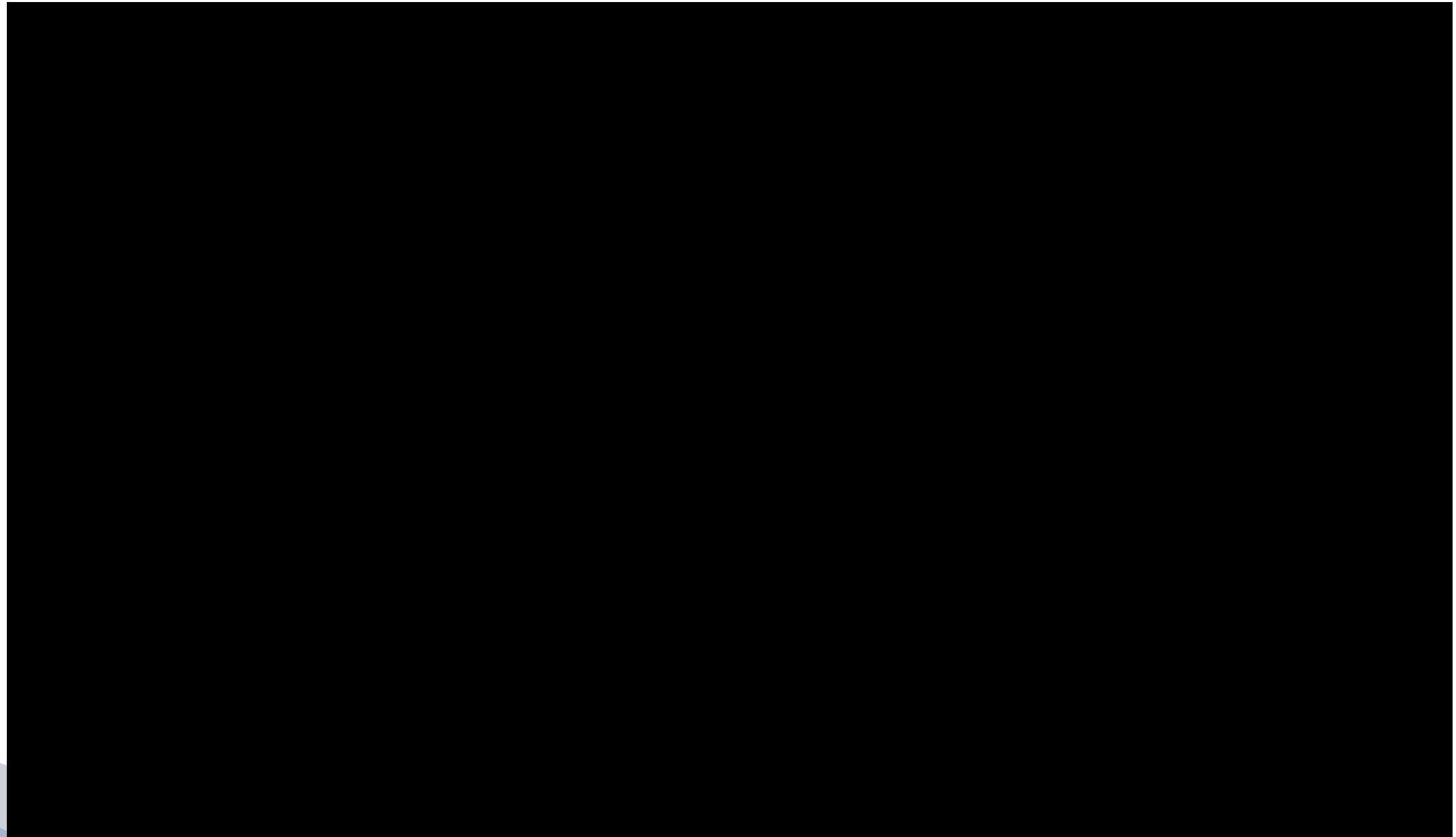
```
SYSCALL_DEFINE2(ptree, struct prinfo *, buf, int *, nr) {
        num_of_proc = ptree_dfs(&init_task);

        if(list_size > num_of_proc) list_size = num_of_proc;

        rb = list_size;
        kernel_buf += rb; user_buf += rb;
        while((rb = copy_to_user(user_buf - rb, kernel_buf - rb, rb)) > 0);

        return num_of_proc;
}
```

test_ptree.c

```
while(1) {
	data = (struct prinfo*)malloc(size * sizeof(struct prinfo));
	re = syscall(380, data, &size);
	if(re == size) break;
	size = re + 10;
	free(data);
}
```

Print process tree information

Test our ptree (Demo)



What we learned?

- How to use Git and Github properly
  - Branch usage
  - Push/Pull usage
- Structure of process in Linux system
  - task_struct : What information does it contains?
  - list_head : Easy-to-use linked list
- How to add system calls in Linux