MyXV6

1.System call -- getprocsinfo
Based on the xv6 System, I implemented a system call named getprocsinfo.
The function prototype is:
int getprocsinfo(struct procinfo)*
*s*truct procinfo is a structure with two data members, an integer pid and a string pname. The syscall should write the data into the passed argument for each existing process at the time of the call. The syscall should return the number of existing processes, or -1 if there is some error.

I added a procinfo.h to put the struct procinfo, included this head file in proc.c, sysproc.c, testgetprocsinfo.c, declared the procinfo struct in proc.h, user.h.
In the usys.S file, add "SYSCALL(getprocsinfo)" to link the system call.
In the user.h file, I declared a function:
int getprocsinfo(struct procinfo);*
In the syscall.h, I defined a system call number:
#define SYS_getprocsinfo 22
In the syscall.c file, I combine the SYS_getprocsinfo with system call function sys_getprocsinfo:
extern int sys_getprocsinfo(void)
[SYS_getprocsinfo] sys_getprocsinfo
In the sysproc.c file, I implement the system call function sys_getprocsinfo(void) to check the argument and call for the low level system call function getprocsinfo():

```
int
sys_getprocsinfo(void)
{
        struct procinfo *info;
        if(argptr(0, (void*)&info, NPROC*sizeof(*info)) < 0)
        return -1;
        return getprocsinfo(info);
}
```

In the proc.c file, I impelemented the low level getprocsinfo function:

```
int
getprocsinfo(struct procinfo* info)
{
        struct proc *p;
        struct procinfo *in;
        in = info;
        int count = 0;
        acquire(&ptable.lock);
        for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
        {
        if (p->state == EMBRYO || p->state == RUNNABLE || p->state == RUNNING || p->state == SLEEPING)
                {
                        in->pid = p->pid;
                        safestrcpy(in->pname, p->name,16);
                        count++;
                        in++;
```

```
                }
            }
            release(&ptable.lock);
            return count;
        }
```

I implemented the test case in testgetprocsinfo.c file.
Finally, modify the Makefile to add the testgetprocsinfo.c into it.

Considering the method I impelemnted my system call is only add my own method into xv6 kernal, there is nothing I have done to modify the oriangl code, so it could have minimum impact on kernal overhead.

To run my xv6 system in qemu, the command is:
qemu-system-i386 -serial mon:stdio -hdb fs.img xv6.img -smp 2 -m 512

To run my test file, the command is:
testgetprocsinfo