1. When user program execute procState();
2. file <usys.S> is executed, this file define the global symbol “procState”, transform the “procState” to SYS\_procState, move it to %eax, and then make an interrupt of type T\_SYSCALL
3. function trap() in file <trap.c> is executed, trap() judges the interrupt type as “T\_SYSCALL” , then call function syscall() in file <syscall.c>
4. In file <syscall.c> define the function pointer array of system calls, call the sys\_procState() according to the “SYS\_procState” in %eax
5. sys\_procState() is in file <sysproc.c>, sys\_procState() will call procState() in file<proc.c>;
6. int
7. sys\_procState(void)
8. {
9. return procState();
11. }
12. procState() will read the ptable, which is the processes list of the xv6, cprint the process name, state, pid, size data from data struct proc of each process in ptable.
13. int
14. procState(void)
15. {
16. struct proc \*p;
17. const char \*procstate[]={"UNUSED", "EMBRYO", "SLEEPING", "RUNNABLE", "RUNNING", "ZOMBIE" };
18. acquire(&ptable.lock);
19. cprintf("name state ID Memory \n");
20. for(p = ptable.proc; p < &ptable.proc[NPROC-1]; p++)
21. {
22. if(p->state == UNUSED)
23. continue;
24. cprintf("%s | %s, | %d | %d Kbytes\n",p->name,procstate[p->state],p->pid,p->sz);
25. }
26. release(&ptable.lock);
27. exit();
28. return 1;
29. }