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>;

```
int
sys_procState(void)
{
  return procState();

}
```

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.

```
int
procState(void)
{
   struct proc *p;
const char *procstate[]={"UNUSED", "EMBRYO", "SLEEPING", "RUNNABLE", "RUNNING",
"ZOMBIE" };
   acquire(&ptable.lock);
   cprintf("name    state       ID    Memory \n");
  for(p = ptable.proc; p < &ptable.proc[NPROC-1]; p++)
     {
   if(p->state == UNUSED)
         continue;

   cprintf("%s  |  %s,  |  %d |  %d
Kbytes\n",p->name,procstate[p->state],p->pid,p->sz);
     }
  release(&ptable.lock);
  exit();
```

```
    return 1;
}
```