Operating System Lab Week 5

Assignment - 5

Name : Shishu
Reg : 2020CA089

Question 1. You have to create a dummy program called “Nprocess” which takes input the number of process to create n process using fork. For example, if you want to create 8 process using fork program should be called as $ Nprocess 8.

Nprocess.c

```c
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"

int main(int argc, char *argv[])
{
      if (argc != 2)
      {
            printf(1, "Usage : Nprocess 8\n");
            exit();
      }
      int n = atoi(argv[1]) - 1;
      int i = 0;
      for (i = 0; i < n; i++)
      {
            if (fork() == 0)
            {
                  printf(1, "Child Pid is %d\n", getpid());
                  exit();
            }
      }
      for (i = 0; i < n; i++)
            wait();
```

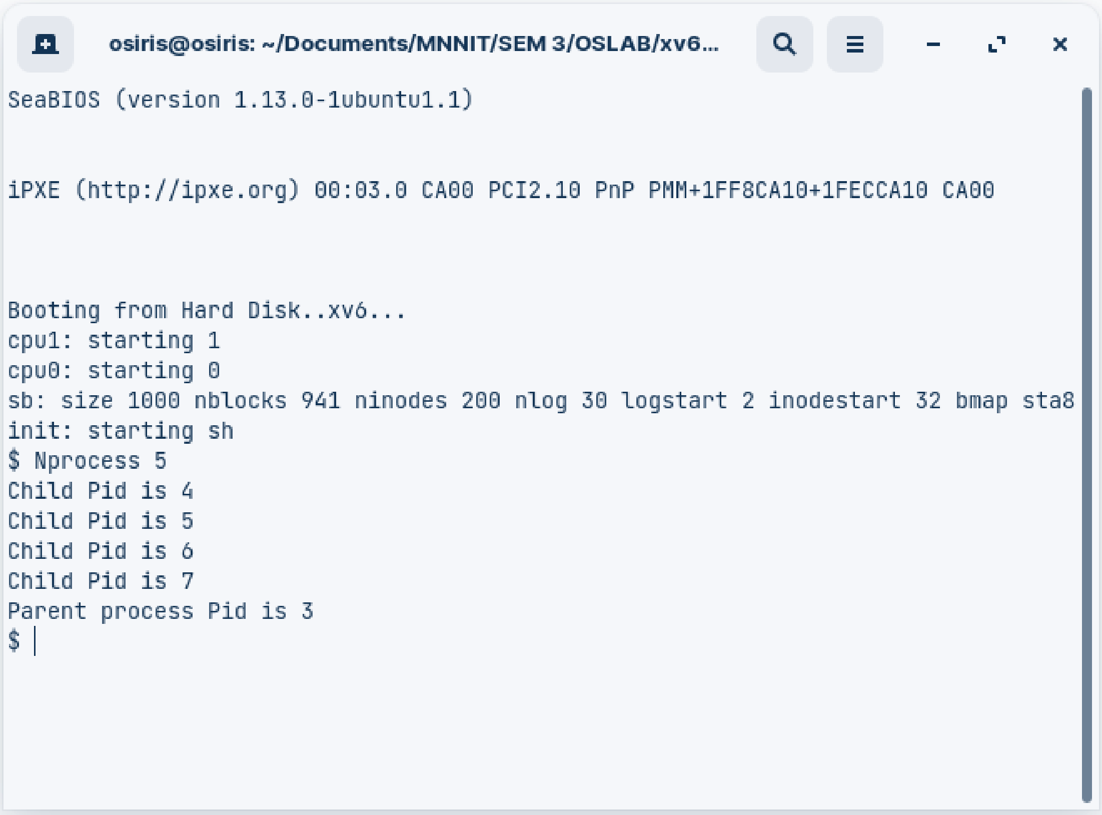
```
        printf(1, "Parent process Pid is %d\n", getpid());
        exit();
}
```

Added Nprocess to Makefile:

```
UPROGS=\
        _cat\
        _echo\
        _forktest\
        _grep\
        _init\
        _kill\
        _ln\
        _ls\
        _mkdir\
        _rm\
        _sh\
        _stressfs\
        _usertests\
        _wc\
        _zombie\
        _myls\
        _ps\
        _Nprocess\


EXTRA=\
        mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
        ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\
        printf.c umalloc.c myls.c ps.c Nprocess.c\
        README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
        .gdbinit.tmpl gdbutil\
```

Running Nprocess in terminal:



```
SeaBIOS (version 1.13.0-1ubuntu1.1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
init: starting sh
$ Nprocess 5
Child Pid is 4
Child Pid is 5
Child Pid is 6
Child Pid is 7
Parent process Pid is 3
$ 
```

Question 2. First create 10 processes using Nprocess and list them using ps that we have implemented in the previous lab. You have to implement a version of ps that accepts arguments to modify its output. The options to be available are as follows:

Add name to syscall.h

```
#define SYS_cps     22
#define SYS_cpsn    23
#define SYS_cpsl    24
#define SYS_cpsd    25
#define SYS_cpss    26
#define SYS_cpsc    27
```

Add function signature to defs.h

```
// proc.c
int             cpuid(void);
void            exit(void);
int             fork(void);
int             growproc(int);
int             kill(int);
struct cpu*     mycpu(void);
struct proc*    myproc();
void            pinit(void);
void            procdump(void);
void            scheduler(void) __attribute__((noreturn));
void            sched(void);
void            setproc(struct proc*);
void            sleep(void*, struct spinlock*);
void            userinit(void);
int             wait(void);
void            wakeup(void*);
void            yield(void);
int             cps(void);
int             cpsn(void);
int             cpsl(void);
int             cpsd(char*);
int             cpss(char*);
int             cpsc(void);
```

Modification to user.h

```
// system calls
int fork(void);
int exit(void) __attribute__((noreturn));
int wait(void);
int pipe(int*);
int write(int, const void*, int);
int read(int, void*, int);
int close(int);
int kill(int);
int exec(char*, char**);
int open(const char*, int);
int mknod(const char*, short, short);
int unlink(const char*);
int fstat(int fd, struct stat*);
int link(const char*, const char*);
int mkdir(const char*);
int chdir(const char*);
int dup(int);
int getpid(void);
char* sbrk(int);
int sleep(int);
int uptime(void);
int cps(void);
int cpsn(void);
int cpsl(void);
int cpsd(char*);
int cpss(char*);
int cpsc(void);
int cpsm(void);
```

Add call to usys.S

```
SYSCALL(uptime)
SYSCALL(cps)
SYSCALL(cpsn)
SYSCALL(cpsl)
SYSCALL(cpsd)
SYSCALL(cpss)
SYSCALL(cpsc)
SYSCALL(cpsm)
```

Add call to syscall.c

```
extern int sys_chdir(void);
extern int sys_close(void);
```

```c
extern int sys_dup(void);
extern int sys_exec(void);
extern int sys_exit(void);
extern int sys_fork(void);
extern int sys_fstat(void);
extern int sys_getpid(void);
extern int sys_kill(void);
extern int sys_link(void);
extern int sys_mkdir(void);
extern int sys_mknod(void);
extern int sys_open(void);
extern int sys_pipe(void);
extern int sys_read(void);
extern int sys_sbrk(void);
extern int sys_sleep(void);
extern int sys_unlink(void);
extern int sys_wait(void);
extern int sys_write(void);
extern int sys_uptime(void);
extern int sys_cps(void);
extern int sys_cpsn(void);
extern int sys_cpsl(void);
extern int sys_cpsd(void);
extern int sys_cpss(void);
extern int sys_cpsc(void);
extern int sys_cpsm(void);

...............
```

```c
SYS_cps]      sys_cps,
[SYS_cpsn]    sys_cpsn,
[SYS_cpsl]    sys_cpsl,
[SYS_cpsd]    sys_cpsd,
[SYS_cpss]    sys_cpss,
[SYS_cpsc]    sys_cpsc,
[SYS_cpsm]    sys_cpsm,
```

ps.c

```
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"

int
main(int argc, char *argv[])
{
	if (argc == 1)
	{
		cps();
	}
	else if (argc == 2)
	{
		if (strcmp(argv[1], "-n") == 0)
		{
			cpsn();
		}
		else if (strcmp(argv[1], "-l") == 0)
		{
			cpsl();
		}
		else if (strcmp(argv[1], "-ch") == 0)
		{
			cpsc();
		}
		else if (strcmp(argv[1], "-m") == 0)
		{
			cpsm();
		}

	}
	else if (argc == 3)
```

```
        {
                if (strcmp(argv[1], "-d") == 0)
                {
                        cpsd(argv[2]);
                }
                else if (strcmp(argv[1], "-s") == 0)
                {
                        cpss(argv[2]);
                }
        }
        exit();
}
```

2.1 -n Only list the name of the processes.

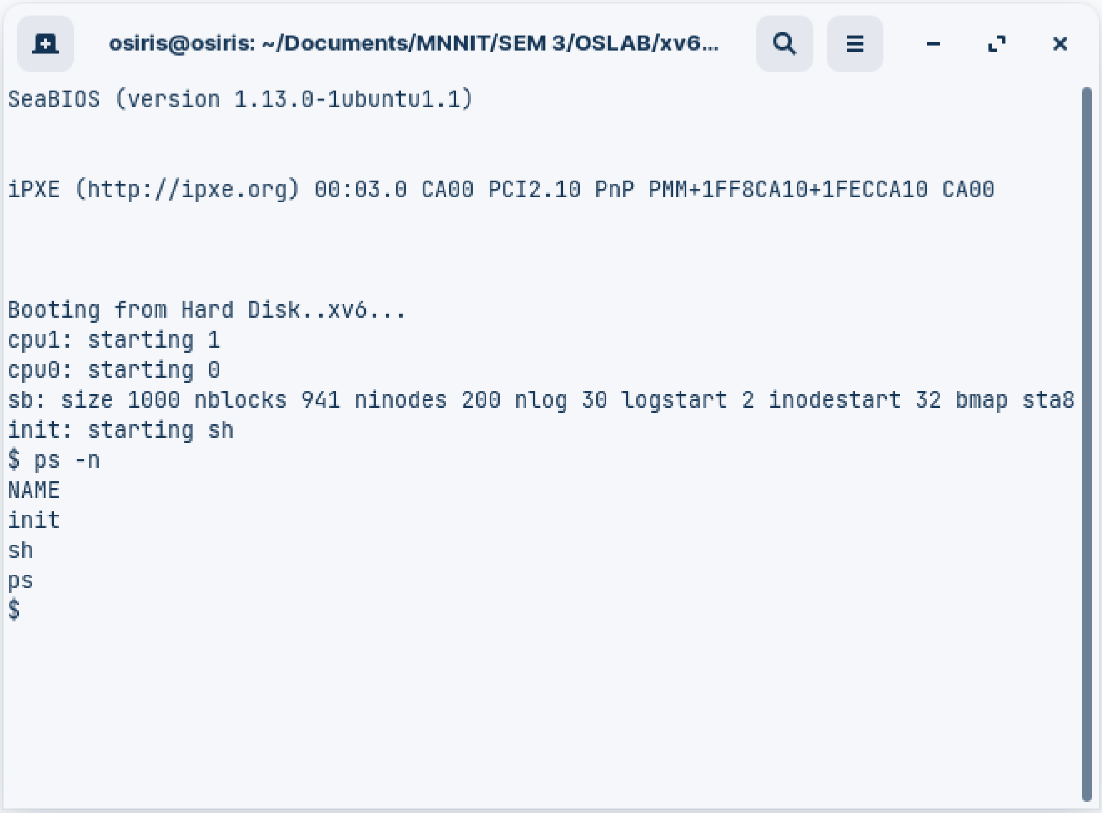
Add function call to sysproc.c

```
int
sys_cpsn(void)
{
return cpsn();
}
```

Modified code in proc.c

```
// current process name
int
```

```
cpsn()
{
     struct proc * p;
     // Enable interrupts on this processor.
     sti();
     // Loop over process table looking for process with pid.
     acquire(&ptable.lock);
     cprintf("NAME\n");
     for (p = ptable.proc; p<&ptable.proc[NPROC]; p++)
     {
          if (p->state == SLEEPING || p->state == RUNNING)
          {
               cprintf("%s\n", p->name);
          }
     }
     release(&ptable.lock);
     return 23;
}
```



```
SeaBIOS (version 1.13.0-1ubuntu1.1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
init: starting sh
$ ps -n
NAME
init
sh
ps
$
```

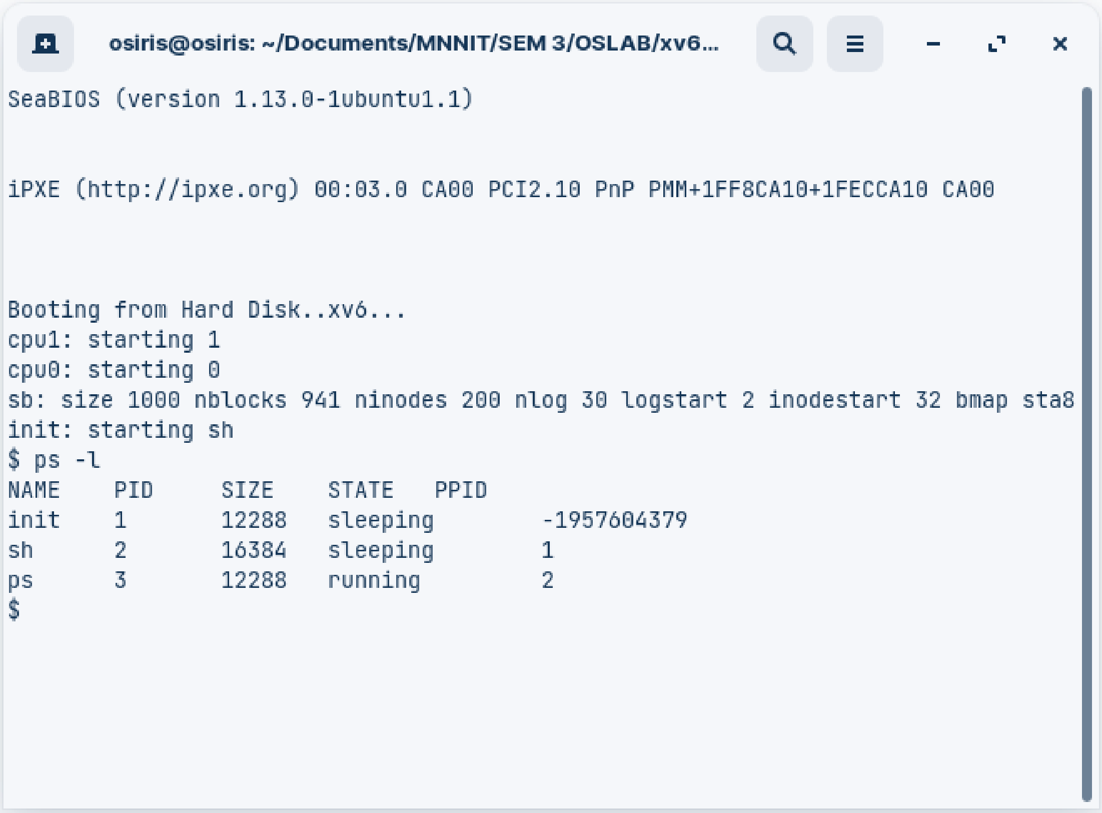
2.2 -l Print a long listing format output, i.e. all the process details are displayed in a tabular format.

sysproc.c

```c
int
sys_cpsl(void)
{
       return cpsl();
}
```

proc.c

```c
// current process list
int
cpsl()
{
	struct proc * p;
	// Enable interrupts on this processor.
	sti();
	// Loop over process table looking for process with pid.
	acquire(&ptable.lock);
	cprintf("NAME\tPID\tSIZE\tSTATE \tPPID\n");
	for (p = ptable.proc; p<&ptable.proc[NPROC]; p++)
	{
		if (p->state == SLEEPING)
		{
			cprintf("%s\t%d\t%d\tsleeping\t%d\n", p->name,
p->pid, p->sz,
				p->parent->pid);
		}
		else if (p->state == RUNNING)
		{
			cprintf("%s\t%d\t%d\trunning \t%d\n", p->name,
p->pid, p->sz,
				p->parent->pid);
		}
	}
	release(&ptable.lock);
	return 24;
}
```



```
SeaBIOS (version 1.13.0-1ubuntu1.1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
init: starting sh
$ ps -l
NAME    PID     SIZE    STATE   PPID
init    1       12288   sleeping        -1957604379
sh      2       16384   sleeping        1
ps      3       12288   running         2
$
```

2.4 -ch Print all sleeping processes by grouping them according to the channel on which they are sleeping.

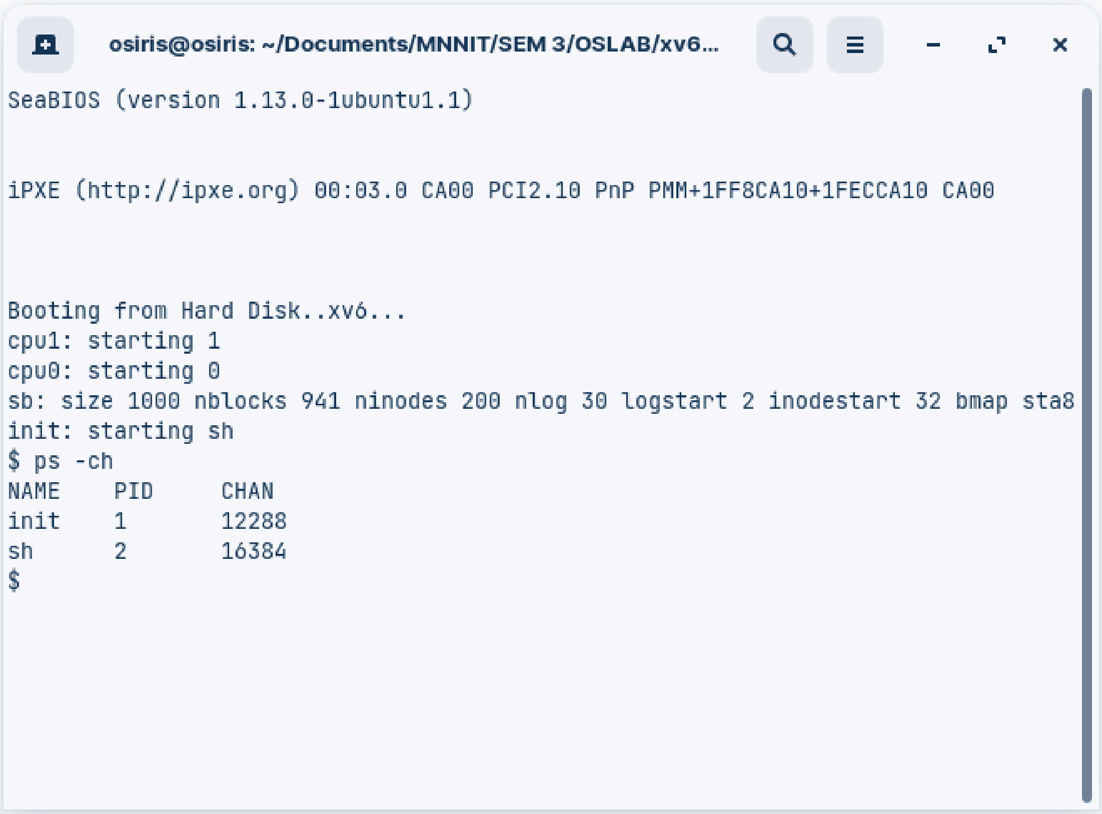
sysproc.c

```
int
sys_cpsc(void)
{
	return cpsc();
}
```

proc.c

```
// sleeping process with channel
int
```

```
cpsc()
{
	struct proc * p;
	// Enable interrupts on this processor.
	sti();
	// Loop over process table looking for process with pid.
	acquire(&ptable.lock);
	cprintf("NAME\tPID\tCHAN\n");
	for (p = ptable.proc; p<&ptable.proc[NPROC]; p++)
	{
		if (p->state == SLEEPING)
		{
			cprintf("%s\t%d\t%d\n", p->name, p->pid, *(int*)
p->chan);
		}
	}
	release(&ptable.lock);
	return 27;
}
```


osiris@osiris: ~/Documents/MNNIT/SEM 3/OSLAB/xv6...

```
SeaBIOS (version 1.13.0-1ubuntu1.1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
init: starting sh
$ ps -ch
NAME    PID     CHAN
init    1       12288
sh      2       16384
$
```

2.5 -d name Print the pid of the process with name name.

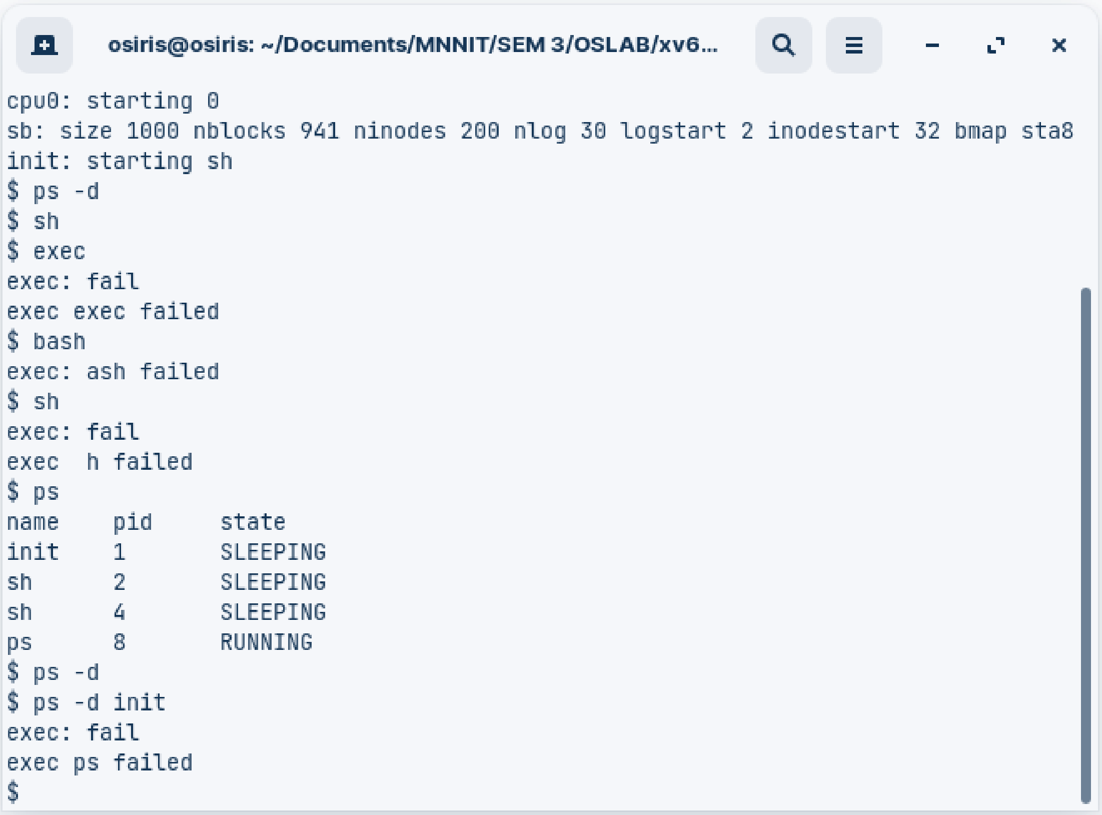
<u>**sysproc.c**</u>

```c
int
sys_cpsd(char* n)
{
       argstr(0, &n);
       return cpsd(n);
}
```

<u>**proc.c**</u>

```c
// process id with given name
int
cpsd(char *n)
{
       struct proc * p;
```

```
        // Enable interrupts on this processor.
        sti();
        // Loop over process table looking for process with pid.
        acquire(&ptable.lock);
        for (p = ptable.proc; p<&ptable.proc[NPROC]; p++)
        {
              if (memcmp(p->name, n, strlen(n)) == 0)
              {
                    cprintf("PID : %d\n", p->pid);
              }
        }
        release(&ptable.lock);
        return 25;
}
```



osiris@osiris: ~/Documents/MNNIT/SEM 3/OSLAB/xv6...

```
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
init: starting sh
$ ps -d
$ sh
$ exec
exec: fail
exec exec failed
$ bash
exec: ash failed
$ sh
exec: fail
exec  h failed
$ ps
name    pid     state
init    1       SLEEPING
sh      2       SLEEPING
sh      4       SLEEPING
ps      8       RUNNING
$ ps -d
$ ps -d init
exec: fail
exec ps failed
$
```

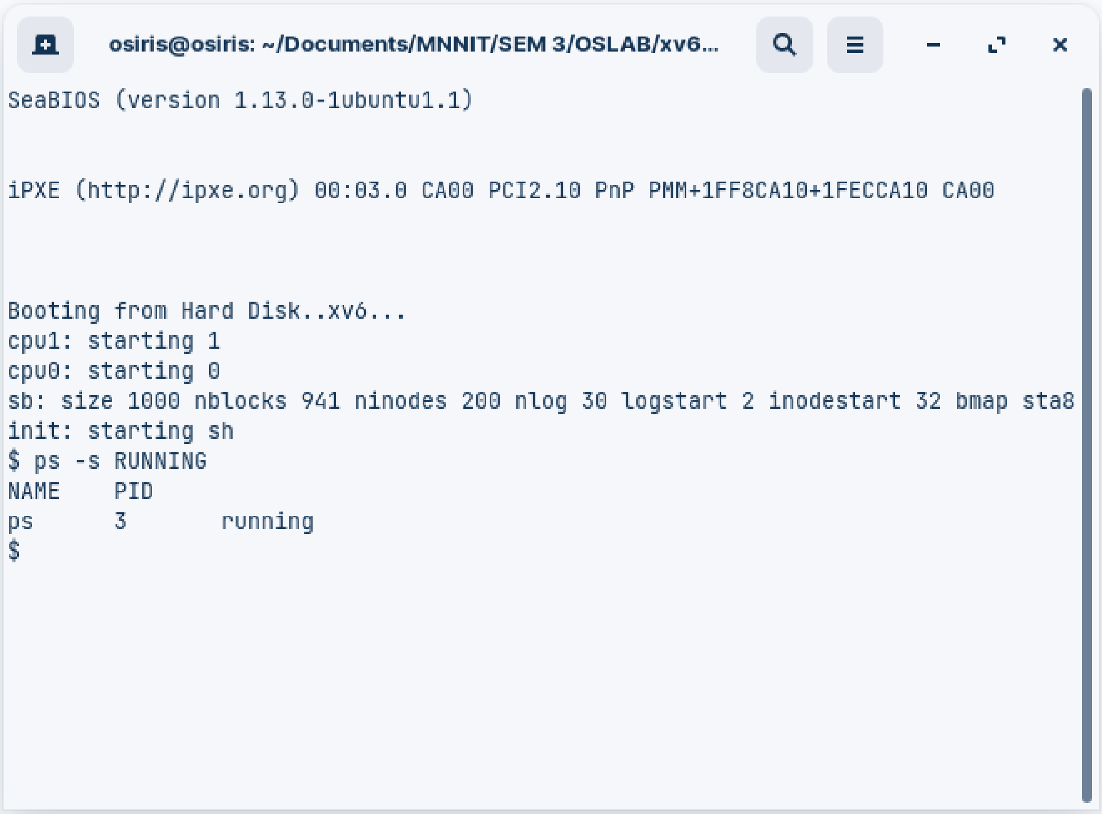
2.6 -s state Print all the processes with their state equal to state.

__sysproc.c__

```
int
sys_cpss(char* n)
{
      argstr(0, &n);
      return cpss(n);
}
```

__proc.c__

```
// process with given state
int
cpss(char *n)
{
      struct proc * p;
      // Enable interrupts on this processor.
      sti();
      // Loop over process table looking for process with pid.
      acquire(&ptable.lock);
      cprintf("NAME\tPID\n");
      for (p = ptable.proc; p<&ptable.proc[NPROC]; p++)
      {
            if (memcmp(n, "RUNNING", 7) == 0 && p->state == RUNNING)
            {
                  cprintf("%s\t%d\trunning\n", p->name, p->pid);
            }
            else if (memcmp(n, "SLEEPING", 8) == 0 && p->state ==
SLEEPING)
            {
                  cprintf("%s\t%d\tsleeping\n", p->name, p->pid);
            }
      }
      release(&ptable.lock);
      return 26;
}
```



```
SeaBIOS (version 1.13.0-1ubuntu1.1)


iPXE (http://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8CA10+1FECCA10 CA00



Booting from Hard Disk..xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
init: starting sh
$ ps -s RUNNING
NAME    PID
ps      3       running
$
```

2.7 -m Print processes in decreasing order of the amount of memory that they are occupying.

sysproc.c

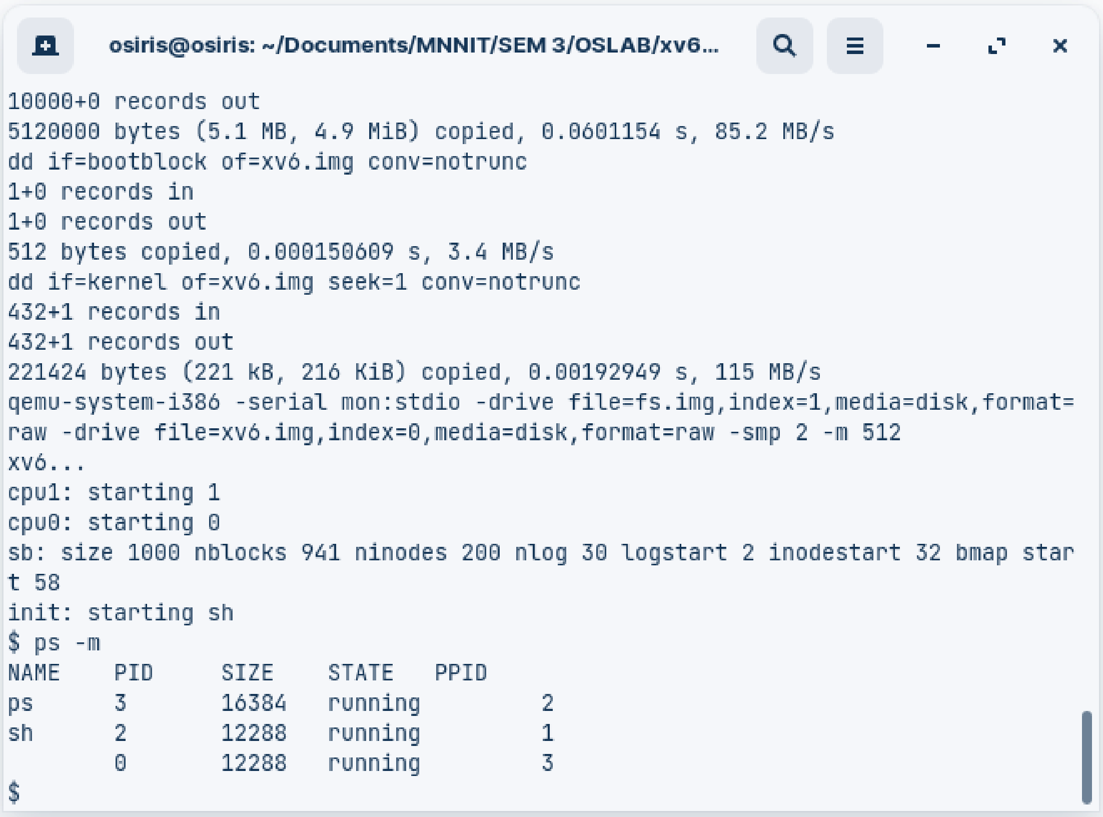
```
int
sys_cpsm(void)
{
       return cpsm();
}
```

proc.c

```
// current process list
int
cpsm()
```

```
{
      struct proc * p;
      struct SizeMap {
            int ppid;
            int size;
      }*s;

      // Enable interrupts on this processor.
      sti();
      // Loop over process table looking for process with pid.
      acquire(&ptable.lock);
     s = (struct SizeMap*)kalloc();
     int i=0;
      cprintf("NAME\tPID\tSIZE\tSTATE \tPPID\n");
      for (p = ptable.proc; p<&ptable.proc[NPROC]; p++)
      {
            s[i].ppid = p->pid;
         s[i].size = p->sz;
         i++;
      }
     // sort the SizeMap
     for(i=0; i<NPROC; i++)
     {
         for(int j=0; j<NPROC-i-1; j++)
         {
             if(s[j].size < s[j+1].size)
             {
                 struct SizeMap temp = s[j];
                 s[j] = s[j+1];
                 s[j+1] = temp;
             }
         }
     }
     for(i=0; i<NPROC; i++)
     {
         if(s[i].size != 0 && s[i].ppid != 0)
         {
             cprintf("%s\t%d\t%d\trunning \t%d\n",
ptable.proc[s[i].ppid].name,
                        ptable.proc[s[i].ppid].pid, s[i].size,
s[i].ppid);
          }
     }
      release(&ptable.lock);
      return 28;
}
```



osiris@osiris: ~/Documents/MNNIT/SEM 3/OSLAB/xv6...

```
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0601154 s, 85.2 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.000150609 s, 3.4 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
432+1 records in
432+1 records out
221424 bytes (221 kB, 216 KiB) copied, 0.00192949 s, 115 MB/s
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=
raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
$ ps -m
NAME    PID     SIZE    STATE   PPID
ps      3       16384   running         2
sh      2       12288   running         1
        0       12288   running         3
$
```