(1) Created procinfo.h and added the following:

```
#ifndef _PROCINFO_H_
#define _PROCINFO_H_

#include "types.h"

struct proc_info {
  int pid;
  char name[16];
  char state[16];
  uint sz;
};

#endif
```

(2) Added in proc.c:

```
#include "procinfo.h"


int
get_proc_info_kernel(int pid, struct proc_info *info)
{
  struct proc *p;

  acquire(&ptable.lock);
  for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
    if(p->pid == pid){
      info->pid = p->pid;
      safestrcpy(info->name, p->name, sizeof(info->name));

      char *st = "UNKNOWN";
      switch(p->state){
      case UNUSED:   st = "UNUSED";   break;
      case EMBRYO:   st = "EMBRYO";   break;
      case SLEEPING: st = "SLEEPING"; break;
      case RUNNABLE: st = "RUNNABLE"; break;
      case RUNNING:  st = "RUNNING";  break;
      case ZOMBIE:   st = "ZOMBIE";   break;
      }
      safestrcpy(info->state, st, sizeof(info->state));

      info->sz = p->sz;

      release(&ptable.lock);
      return 0;
    }
  }
  release(&ptable.lock);
  return -1;
```

```
}
```

(3) Added in proc.h:

```
#include "procinfo.h"

int get_proc_info_kernel(int pid, struct proc_info *info);
```

(4) Added in sysproc.c:

```
#include "procinfo.h"

int
sys_pinfo(void)
{
  int pid;
  struct proc_info info;
  struct proc_info *uaddr;

  if(argint(0, &pid) < 0)
    return -1;
  if(argptr(1, (void*)&uaddr, sizeof(*uaddr)) < 0)
    return -1;

  if(get_proc_info_kernel(pid, &info) < 0)
    return -1;

  if(copyout(myproc()->pgdir, (uint)uaddr, (char*)&info, sizeof(info)) < 0)
    return -1;

  return 0;
}
```

(5) Added in syscall.h

```
#define SYS_pinfo  22
```

(6) Added in syscall.c

```
extern int sys_pinfo(void);

[SYS_pinfo]   sys_pinfo,
```

(7) Added in usys.S

```
SYSCALL(pinfo)
```

(8) Added in user.h:

```
#include "procinfo.h"

int pinfo(int pid, struct proc_info *info);
```

(9) Created pinfo.c and added the following:

```
#include "types.h"
#include "user.h"
#include "procinfo.h"

int
main(int argc, char *argv[])
{
  if(argc != 2){
    printf(1, "Usage: pinfo <pid>\n");
    exit();
  }

  int pid = atoi(argv[1]);
  struct proc_info info;

  if(pinfo(pid, &info) < 0){
    printf(1, "pinfo: PID %d not found\n", pid);
    exit();
  }

  printf(1, "PID: %d\n", info.pid);
  printf(1, "Name: %s\n", info.name);
  printf(1, "State: %s\n", info.state);
  printf(1, "Size: %d\n", info.sz);

  exit();
}
```

(10) Added in Makefile in UPROGS section:

```
  _pinfo\
  _testproc\
```

(11) Cerated testproc.c and added the following:

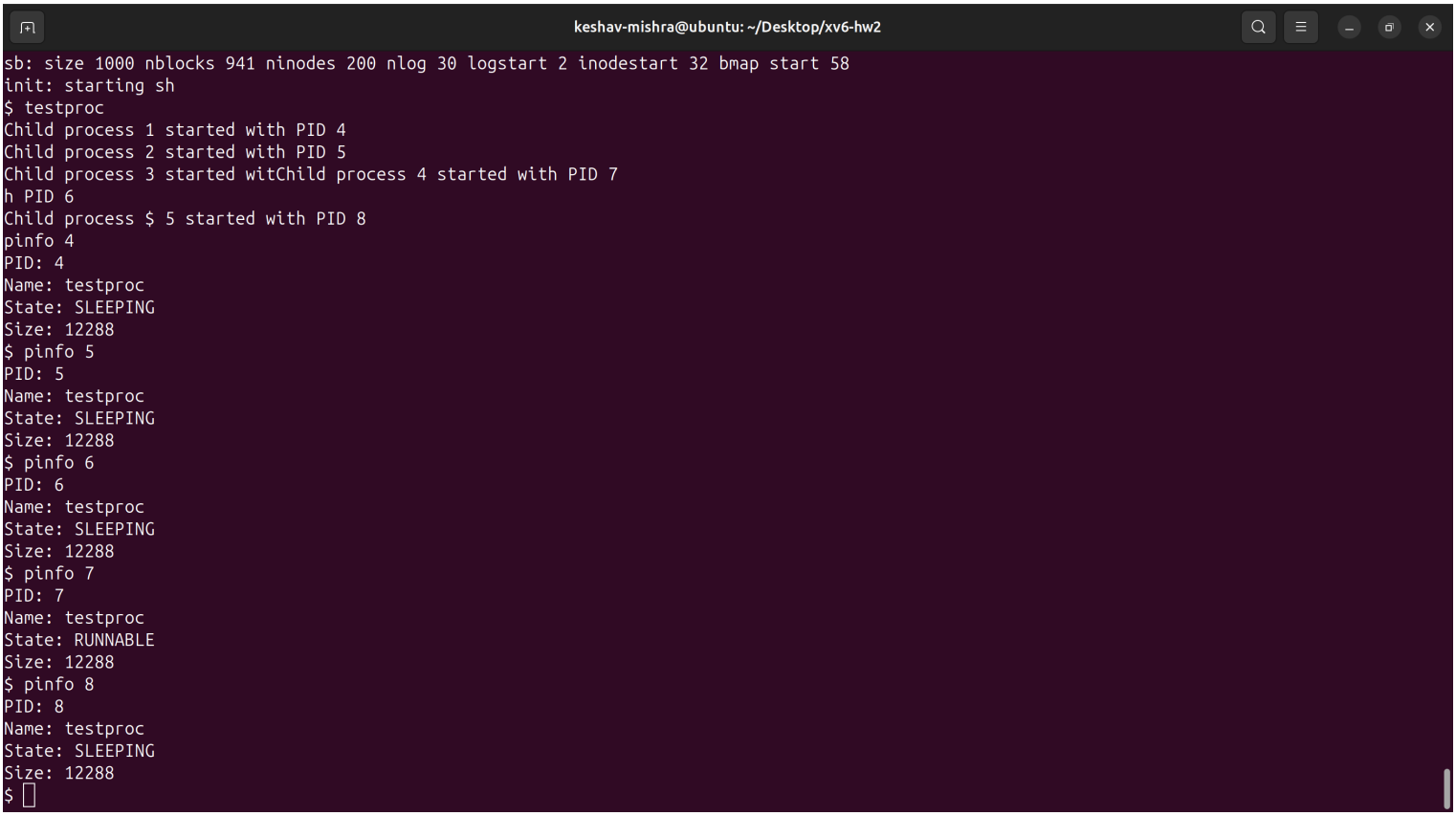
```
#include "types.h"
#include "user.h"

int
main(void)
```

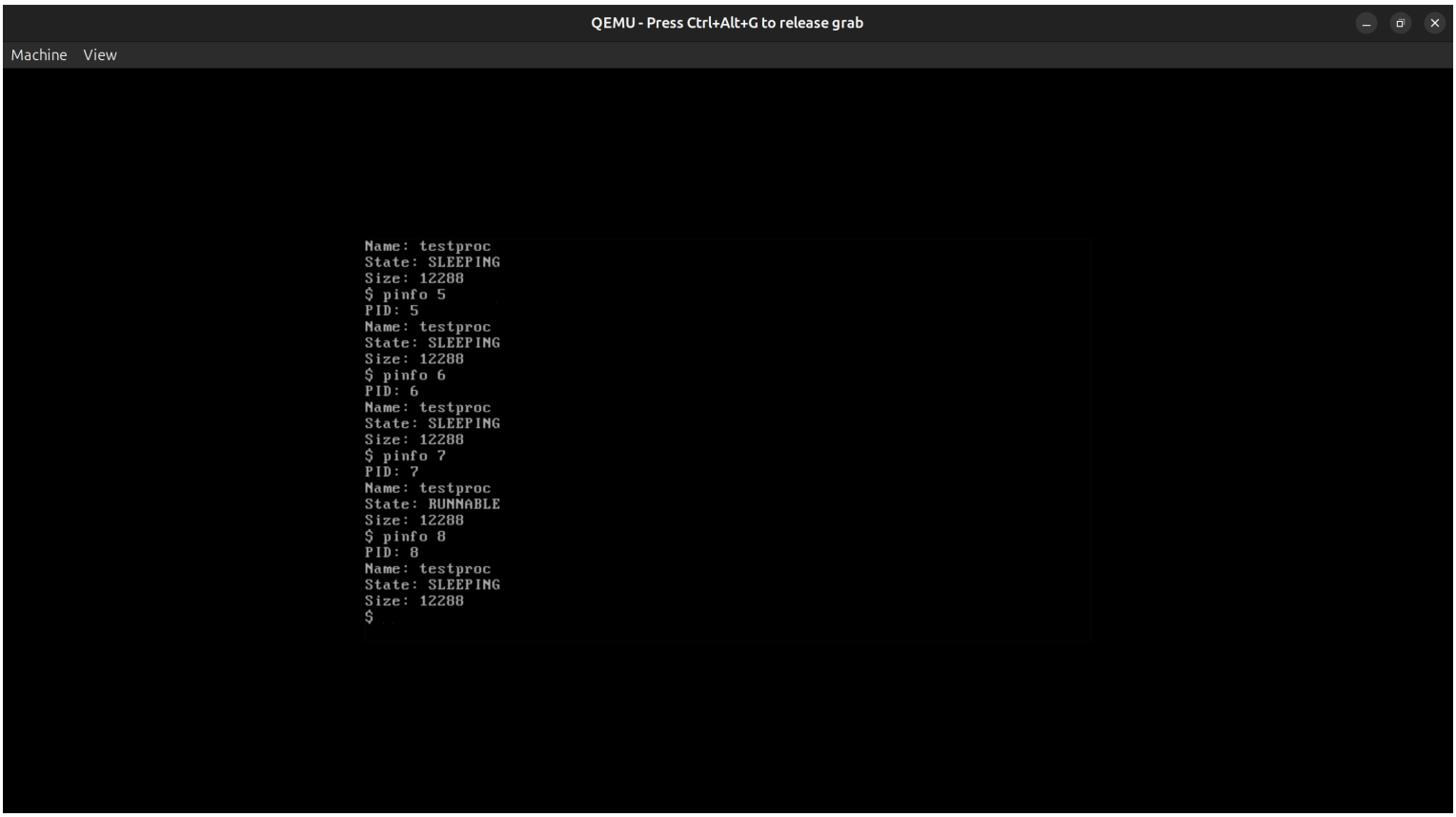
```
{
  int i;
  int num_children = 5;

  for(i = 0; i < num_children; i++) {
    int pid = fork();
    if(pid < 0) {
      printf(1, "Fork failed\n");
      exit();
    }
    if(pid == 0) {
      printf(1, "Child process %d started with PID %d\n", i+1, getpid());
      while(1) {
        sleep(100);
      }
    }
  }

  exit();
}
```



Screenshot of the output in terminal



Screenshot of the output in QEMU terminal