

# CSL301

## Assignment 2

### Amay Dixit - 12340220

#### Create procinfo.h Header File

**File:** procinfo.h

```
#ifndef _PROCINFO_H_
#define _PROCINFO_H_

#include "types.h"

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

#endif
```

---

#### Modify proc.h

**File:** proc.h

**Added at the end of the file:**

```
#include "procinfo.h"
int get_proc_info(int pid, struct proc_info *info);
```

---

#### Implement get\_proc\_info Function

**File:** proc.c

**Added function:**

```

Int
get_proc_info(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));

            const 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; // not found
}

```

## Add System Call Number

**File: syscall.h**

**Added line:**

```
#define SYS_get_proc_info 26
```

## Implement System Call Handler

**File: sysproc.c**

### **Added function:**

```
Int
sys_get_proc_info(void)
{
    int pid;
    struct proc_info *uinfo;

    if(argint(0, &pid) < 0)
        return -1;
    if(argptr(1, (char **) &uinfo, sizeof(struct proc_info)) < 0)
        return -1;
    return get_proc_info(pid, uinfo);
}
```

---

## **Wire System Call in syscall.c**

### **File: syscall.c**

#### **Added extern declaration:**

```
extern int sys_get_proc_info(void);
```

---

#### **Added to syscalls array:**

```
[SYS_get_proc_info]     sys_get_proc_info,
```

---

## **Update User Library**

### **File: user.h**

#### **Added function declaration:**

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

---

## **Add Assembly Stub**

### **File: usys.S**

#### **Added line:**

```
SYSCALL(get_proc_info)
```

---

## Create User Program pinfo.c

### File: pinfo.c

```
#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;

    int ret = get_proc_info(pid, &info);
    if(ret < 0){
        printf(1, "pinfo: process %d not found or error\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();
}
```

---

## Step 10: Create Test Program testproc.c

### File: testproc.c

```
#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); // never exits
    }
}
exit();
}
```

---

## Testing Results

### Test Output Screenshot:

```
12340220$ testproc
Child process 1 started with PID 19
Child process 2 started with PID 20
Child process 3 started with PID 21
Child process 4 started with PID 22
Child process 5 started with PID 23
```

```
12340220$ pinfo 20
PID: 20
Name: testproc
State: RUNNABLE
Size: 12288
12340220$ pinfo 22
PID: 22
Name: testproc
State: RUNNABLE
Size: 12288
12340220$ pinfo 23
PID: 23
Name: testproc
State: RUNNABLE
Size: 12288
12340220$ pinfo 25
pinfo: process 25 not found or error
12340220$
```