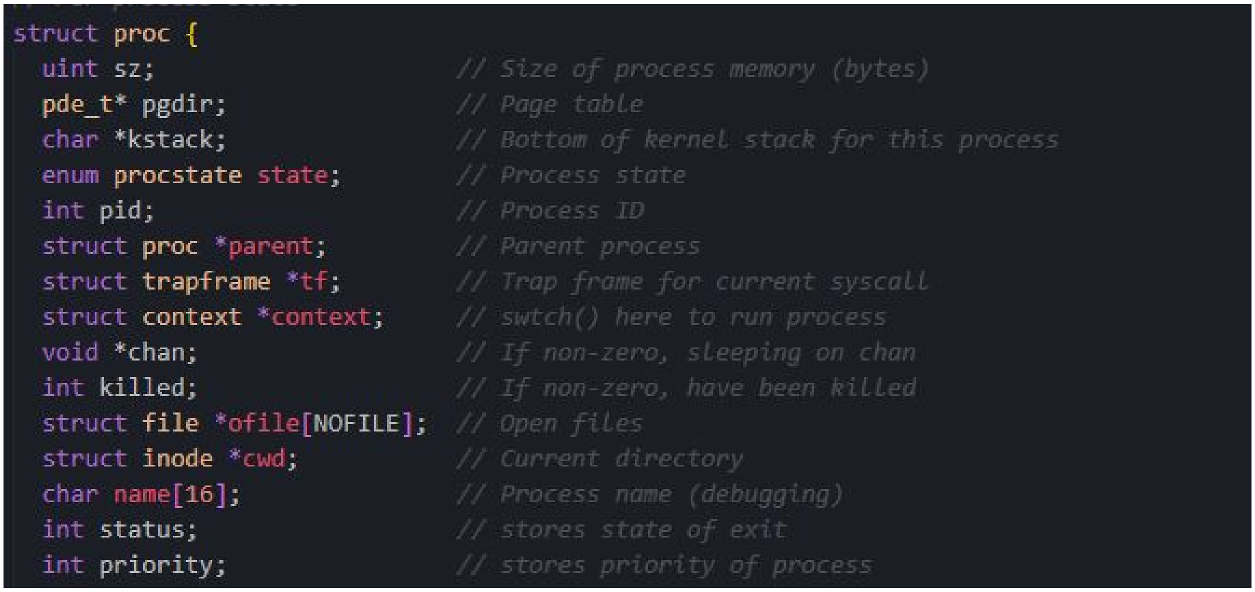
Phillip Nguyen 862050241
Fangsheng Chao 862016183

Lab 2 Report

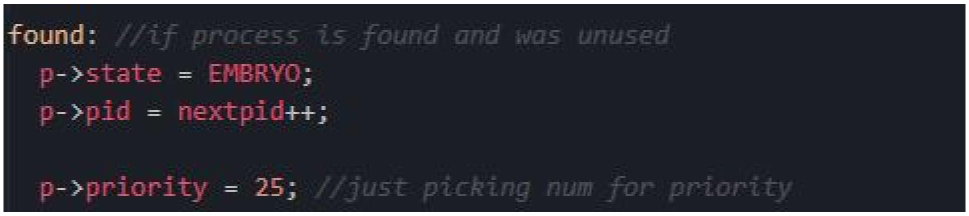
Adding Priority)

First thing we did in the lab was add a priority field in proc.h



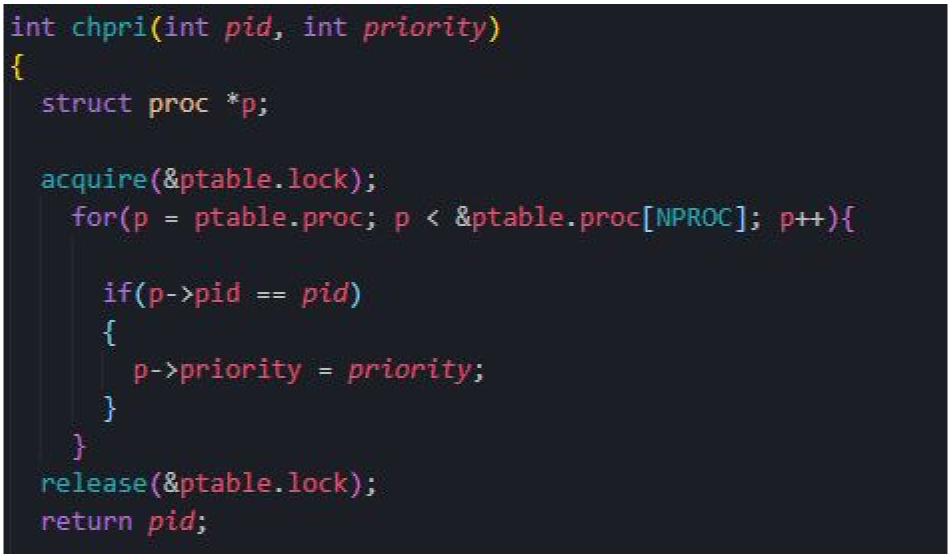
```
struct proc {
  uint sz;                     // Size of process memory (bytes)
  pde_t* pgdir;                // Page table
  char *kstack;                // Bottom of kernel stack for this process
  enum procstate state;        // Process state
  int pid;                     // Process ID
  struct proc *parent;         // Parent process
  struct trapframe *tf;        // Trap frame for current syscall
  struct context *context;     // swtch() here to run process
  void *chan;                  // If non-zero, sleeping on chan
  int killed;                  // If non-zero, have been killed
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
  int status;                  // stores state of exit
  int priority;                // stores priority of process
```

And then we set the default priority in allocproc to 25.

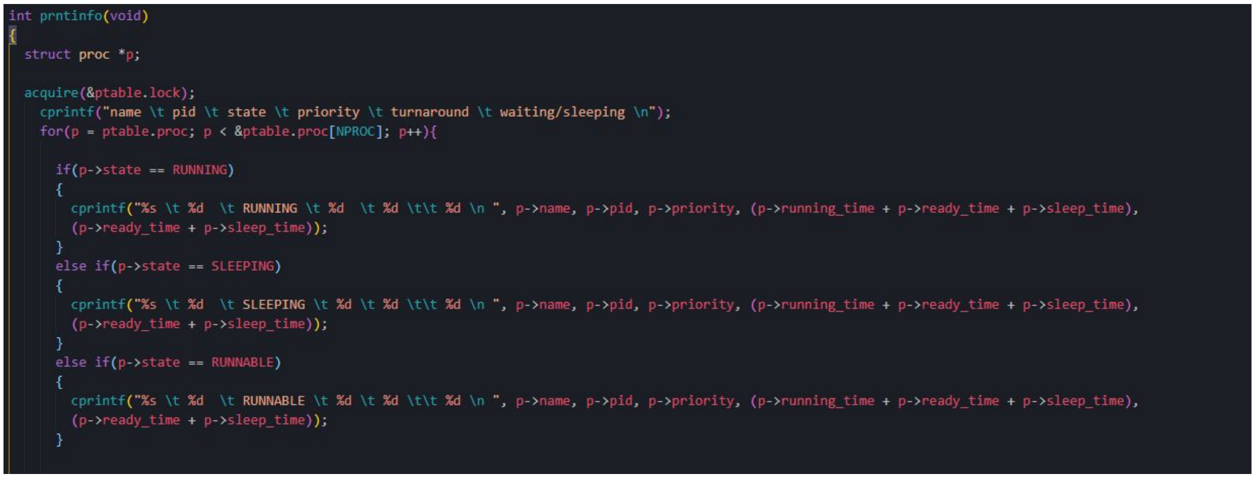


```
found: //if process is found and was unused
  p->state = EMBRYO;
  p->pid = nextpid++;

  p->priority = 25; //just picking num for priority
```

We then added function calls to change the priority and print out information relating to that process such as the priority and turnaround time for the bonus.



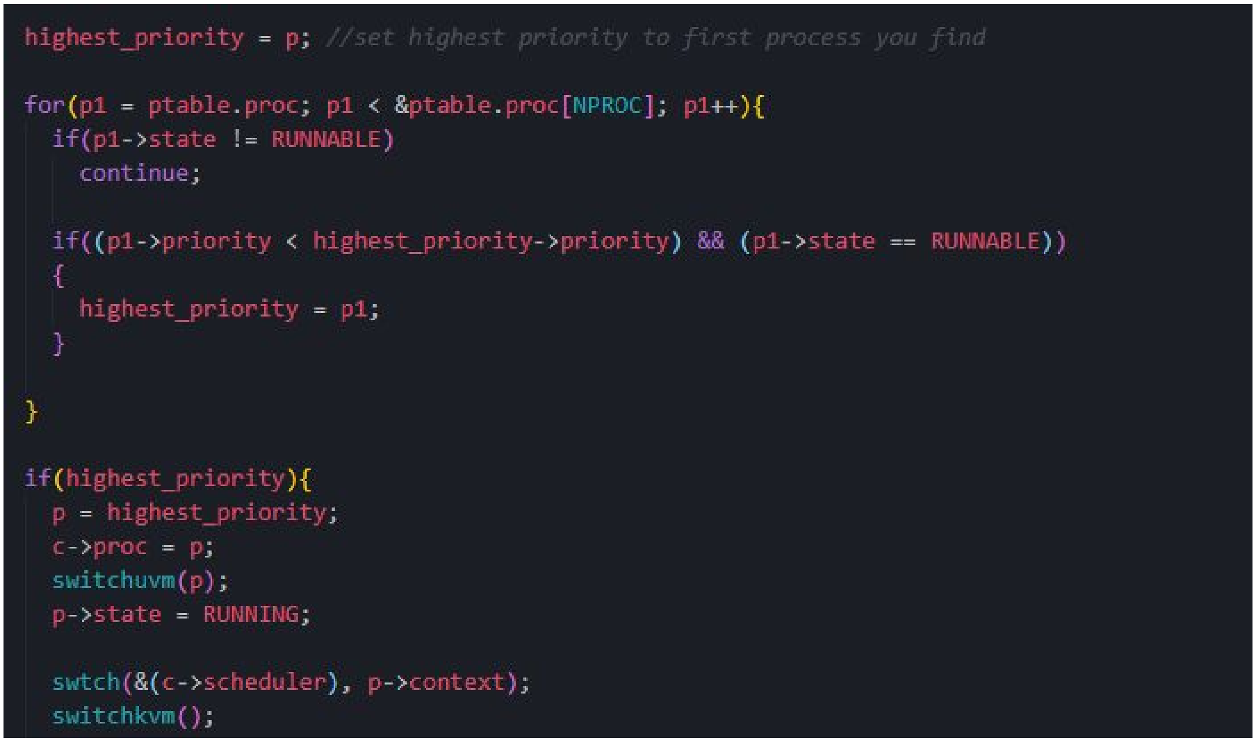
```
int chpri(int pid, int priority)
{
  struct proc *p;

  acquire(&ptable.lock);
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){

      if(p->pid == pid)
      {
        p->priority = priority;
      }
    }
  release(&ptable.lock);
  return pid;
```



```
int prntinfo(void)
{
  struct proc *p;

  acquire(&ptable.lock);
    cprintf("name \t pid \t state \t priority \t turnaround \t waiting/sleeping \n");
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){

      if(p->state == RUNNING)
      {
        cprintf("%s \t %d  \t RUNNING \t %d  \t %d \t\t %d \n ", p->name, p->pid, p->priority, (p->running_time + p->ready_time + p->sleep_time),
        (p->ready_time + p->sleep_time));
      }
      else if(p->state == SLEEPING)
      {
        cprintf("%s \t %d  \t SLEEPING \t %d \t %d \t\t %d \n ", p->name, p->pid, p->priority, (p->running_time + p->ready_time + p->sleep_time),
        (p->ready_time + p->sleep_time));
      }
      else if(p->state == RUNNABLE)
      {
        cprintf("%s \t %d  \t RUNNABLE \t %d \t %d \t\t %d \n ", p->name, p->pid, p->priority, (p->running_time + p->ready_time + p->sleep_time),
        (p->ready_time + p->sleep_time));
      }
```

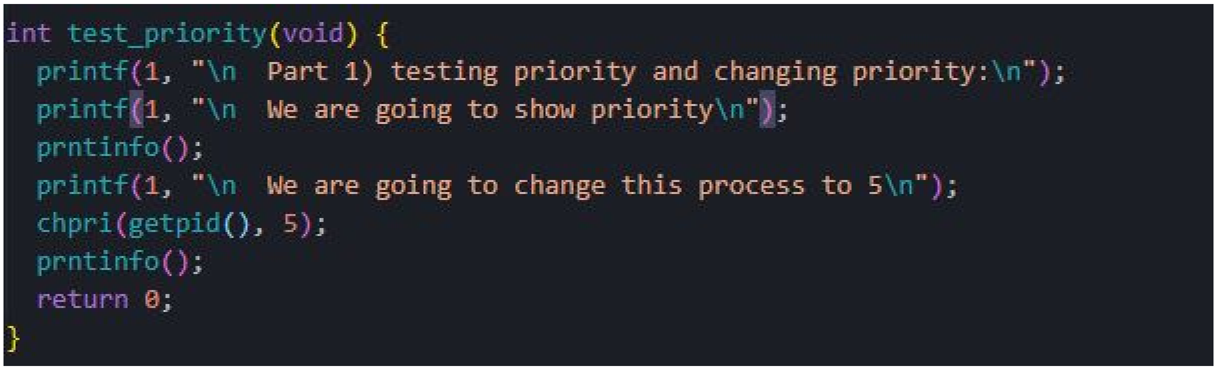
Scheduler Modification)

We then modified the scheduler to select the highest priority process that is runnable first. We first searched by the highest priority number and set the highest priority to that process and told the scheduler to switch to that process.



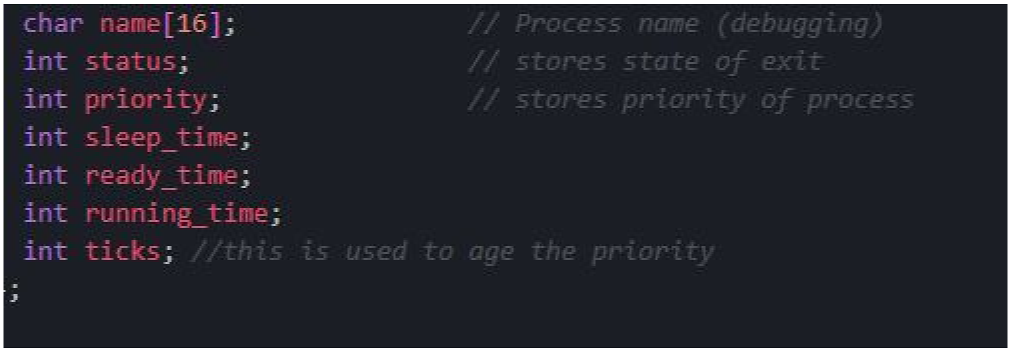
```
highest_priority = p; //set highest priority to first process you find

for(p1 = ptable.proc; p1 < &ptable.proc[NPROC]; p1++){
  if(p1->state != RUNNABLE)
    continue;

  if((p1->priority < highest_priority->priority) && (p1->state == RUNNABLE))
  {
    highest_priority = p1;
  }

}

if(highest_priority){
  p = highest_priority;
  c->proc = p;
  switchuvm(p);
  p->state = RUNNING;

  swtch(&(c->scheduler), p->context);
  switchkvm();
```

We then wrote a simple test program to make sure that the priority changes correctly.



```
int test_priority(void) {
  printf(1, "\n  Part 1) testing priority and changing priority:\n");
  printf(1, "\n  We are going to show priority\n");
  prntinfo();
  printf(1, "\n  We are going to change this process to 5\n");
  chpri(getpid(), 5);
  prntinfo();
  return 0;
}
```

We then added fields to measure the statistics of each process.

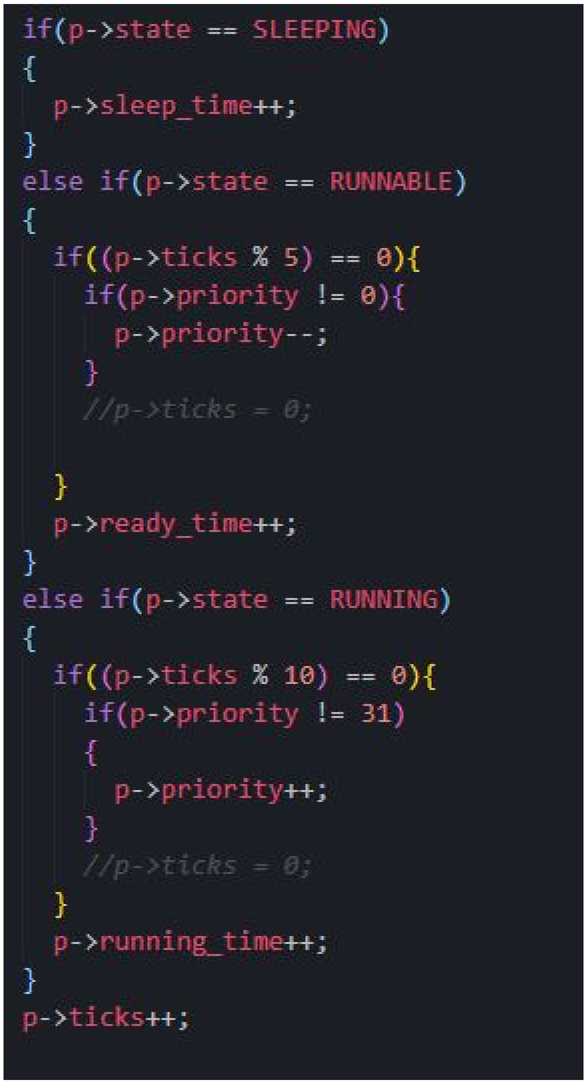


```
 char name[16];               // Process name (debugging)
 int status;                  // stores state of exit
 int priority;                // stores priority of process
 int sleep_time;
 int ready_time;
 int running_time;
 int ticks; //this is used to age the priority
;
```

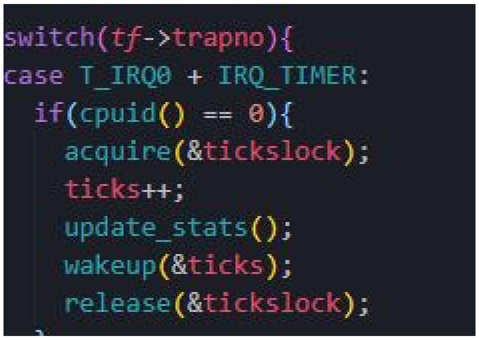
Bonus section for aging priority and stats for process)

We added a function that would count the amount of time a process was in a certain state inside proc.c and then called that function every tick inside trap.c.

We also tried aging the priority if it is in a certain state for 10 ticks

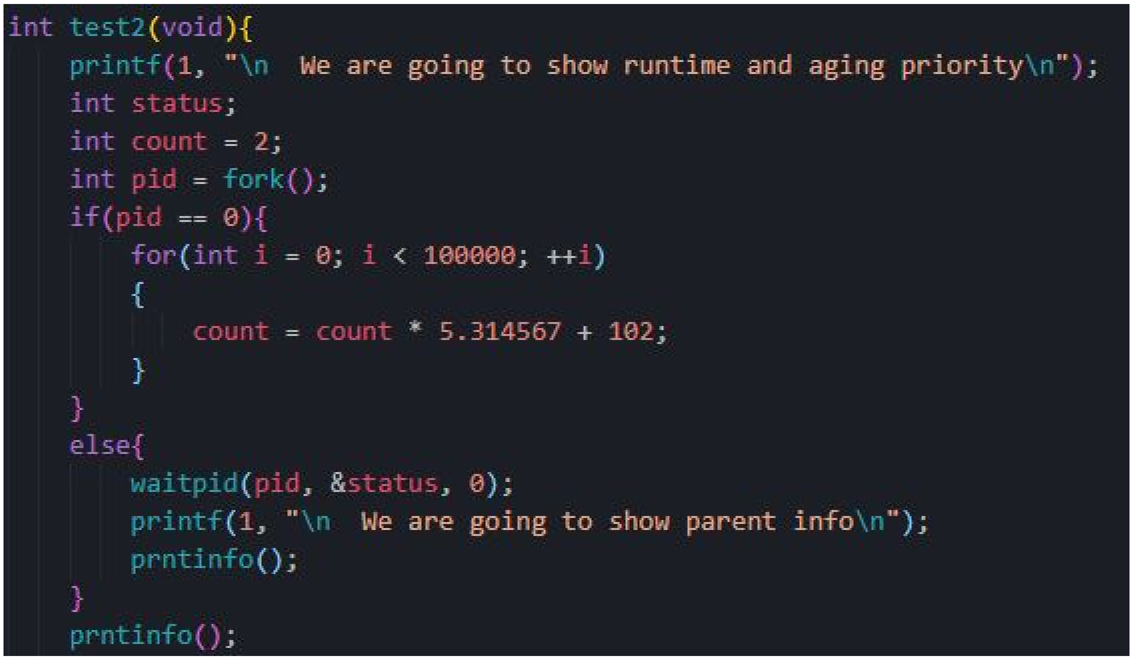


```
if(p->state == SLEEPING)
{
  p->sleep_time++;
}
else if(p->state == RUNNABLE)
{
  if((p->ticks % 5) == 0){
    if(p->priority != 0){
      p->priority--;
    }
    //p->ticks = 0;

  }
  p->ready_time++;
}
else if(p->state == RUNNING)
{
  if((p->ticks % 10) == 0){
    if(p->priority != 31)
    {
      p->priority++;
    }
    //p->ticks = 0;
  }
  p->running_time++;
}
p->ticks++;
```



```
switch(tf->trapno){
case T_IRQ0 + IRQ_TIMER:
  if(cpuid() == 0){
    acquire(&tickslock);
    ticks++;
    update_stats();
    wakeup(&ticks);
    release(&tickslock);
```

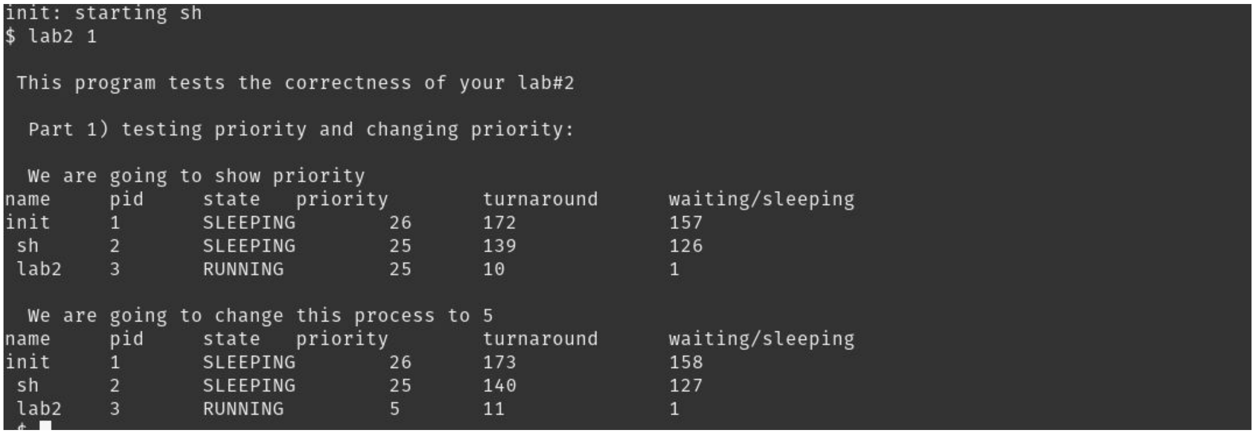
We then wrote a test for these bonus sections of the lab



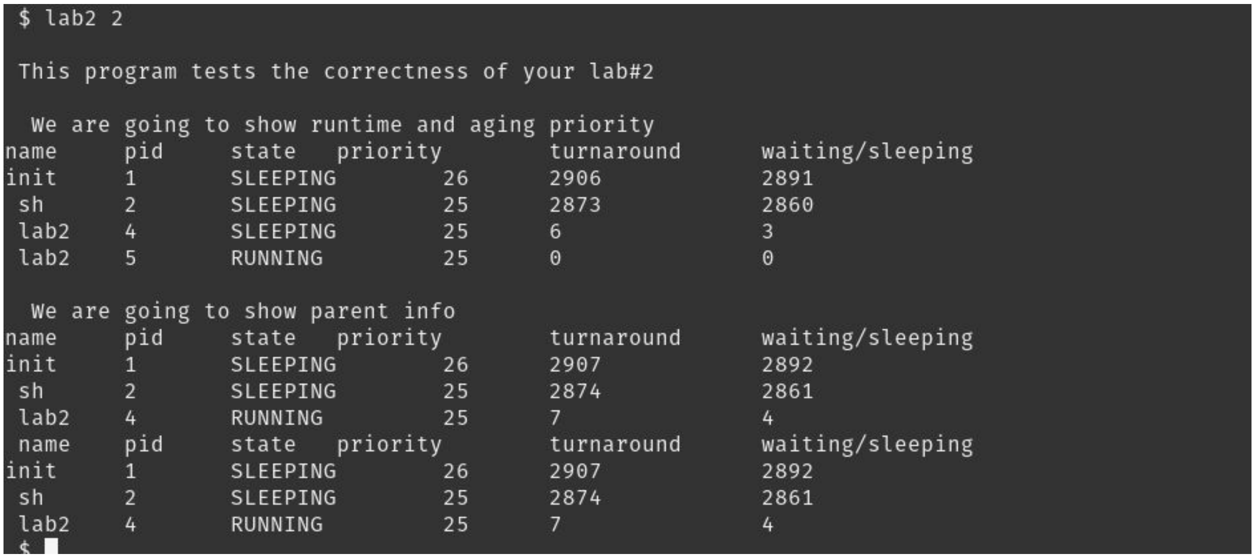
```
int test2(void){
    printf(1, "\n  We are going to show runtime and aging priority\n");
    int status;
    int count = 2;
    int pid = fork();
    if(pid == 0){
        for(int i = 0; i < 100000; ++i)
        {
            count = count * 5.314567 + 102;
        }
    }
    else{
        waitpid(pid, &status, 0);
        printf(1, "\n  We are going to show parent info\n");
        prntinfo();
    }
    prntinfo();
```

Example Outputs)

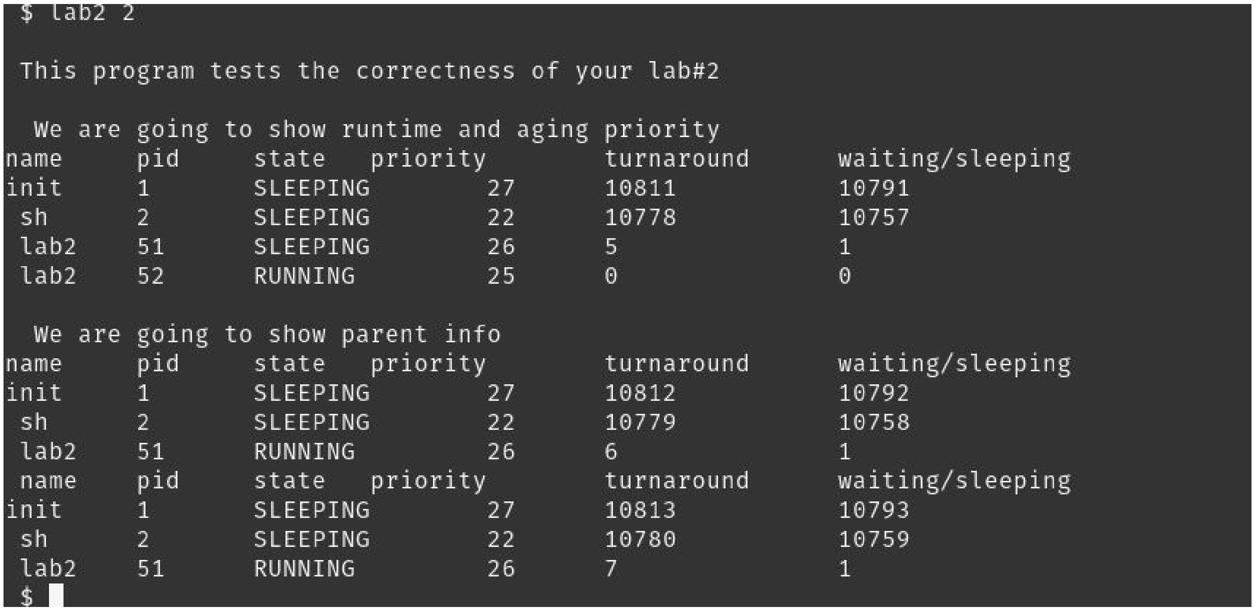
And here are some example outputs from running different test commands



```
init: starting sh
$ lab2 1

 This program tests the correctness of your lab#2

  Part 1) testing priority and changing priority:

  We are going to show priority
name     pid     state   priority        turnaround      waiting/sleeping
init     1       SLEEPING        26      172             157
 sh      2       SLEEPING        25      139             126
 lab2    3       RUNNING         25      10              1

  We are going to change this process to 5
name     pid     state   priority        turnaround      waiting/sleeping
init     1       SLEEPING        26      173             158
 sh      2       SLEEPING        25      140             127
 lab2    3       RUNNING         5       11              1
```



```
 $ lab2 2

 This program tests the correctness of your lab#2

  We are going to show runtime and aging priority
name     pid     state   priority        turnaround      waiting/sleeping
init     1       SLEEPING        26      2906            2891
 sh      2       SLEEPING        25      2873            2860
 lab2    4       SLEEPING        25      6               3
 lab2    5       RUNNING         25      0               0

  We are going to show parent info
name     pid     state   priority        turnaround      waiting/sleeping
init     1       SLEEPING        26      2907            2892
 sh      2       SLEEPING        25      2874            2861
 lab2    4       RUNNING         25      7               4
 name    pid     state   priority        turnaround      waiting/sleeping
init     1       SLEEPING        26      2907            2892
 sh      2       SLEEPING        25      2874            2861
 lab2    4       RUNNING         25      7               4
 $
```



```
 $ lab2 2

 This program tests the correctness of your lab#2

  We are going to show runtime and aging priority
name     pid     state   priority        turnaround      waiting/sleeping
init     1       SLEEPING        27      10811           10791
 sh      2       SLEEPING        22      10778           10757
 lab2    51      SLEEPING        26      5               1
 lab2    52      RUNNING         25      0               0

  We are going to show parent info
name     pid     state   priority        turnaround      waiting/sleeping
init     1       SLEEPING        27      10812           10792
 sh      2       SLEEPING        22      10779           10758
 lab2    51      RUNNING         26      6               1
 name    pid     state   priority        turnaround      waiting/sleeping
init     1       SLEEPING        27      10813           10793
 sh      2       SLEEPING        22      10780           10759
 lab2    51      RUNNING         26      7               1
 $
```

This last picture shows that information displayed when we use a new added system call that prints information.

You can also see how the priority of different processes change because they were running for a certain amount of time and the priority ages.