CLASS ASSESSMENT 3

Date-20-08-25
Roll no-12340740

STEP1(proc.h)-

```
// Per-process state
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
  struct inode
  Generate Copilot summary
  // If non-zero, sleeping on chan
  // If non-zero, have been killed
  E];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
  int sched_count;
  int run_ticks;
};
```

Last two lines

STEP2 proc.c

```
//PAGEBREAK: 32
static struct proc*
allocproc(void)
{
  struct proc *p;

  char *sp;

  acquire(&ptable.lock);

  for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
    if(p->state == UNUSED)
      goto found;

  release(&ptable.lock);
  return 0;

found:
  p->state = EMBRYO;
  p->pid = nextpid++;

  release(&ptable.lock);

  // Allocate kernel stack.
  if((p->kstack = kalloc()) == 0){
    p->state = UNUSED;
    return 0;
  }
  sp = p->kstack + KSTACKSIZE;

  // Leave room for trap frame.
  sp -= sizeof *p->tf;
  p->tf = (struct trapframe*)sp;

  // Set up new context to start executing at forkret,
  // which returns to trapret.
  sp -= 4;
  *(uint*)sp = (uint)trapret;

  sp -= sizeof *p->context;
  p->context = (struct context*)sp;
  memset(p->context, 0, sizeof *p->context);
  p->context->eip = (uint)forkret;
  p->sched_count=0;
  p->run_ticks=0;


  return p;
}
```

Last two lines

STEP 3 proc.c

```
void
scheduler(void)
{
  struct proc *p;
  struct cpu *c = mycpu();
  c->proc = 0;

  for(;;){
    // Enable interrupts on this processor.
    sti();

    // Loop over proc                    to run.
    acquire(&ptable.l
    for(p = ptable.pr                   ++){
      if(p->state != RUNNABLE)
        continue;
      p->sched_count++;
```

(enum procstate)RUNNABLE = 3

Generate Copilot summary

Last line

STEP 4 trap.c

```
switch(tf->trapno){
case T_IRQ0 + IRQ_TIMER:
  if(cpuid() == 0){
    acquire(&tickslock);
    ticks++;
    wakeup(&ticks);
    release(&tickslock);
  }
  lapiceoi();
  if(myproc() && myproc()->state ==RUNNING){
    myproc()->run_ticks++;
  }
  break;
```

STEP 5 syscall.h and syscall.c

1)syscall.h

```
#define SYS_getstats 22
```

2)syscall.c

```
[SYS_getstats] sys_getstats,
};
```

```
extern int sys_getstats(void);
```

STEP 6) sysproc.c

```
int
sys_getstats(void){
int *user_stats_ptr;
if(argptr(0,(void*)&user_stats_ptr,sizeof(int))<0)
  return -1;
struct proc *p = myproc();
int kernel_stats[2];
kernel_stats[0]=p->sched_count;
kernel_stats[1]=p->run_ticks;
if(copyout(p->pgdir,(uint)user_stats_ptr,(char*)kernel_stats,sizeof(kernel_stats))<0)
  return-1;
return 0;
```

STEP 7)user.h

```
struct procstats{
    int count;
    int ticks;
};
```

```
int getstats(int *stats_array);

// ulib.c
```

STEP 8) usys.S

```
SYSCALL(getstats)
```

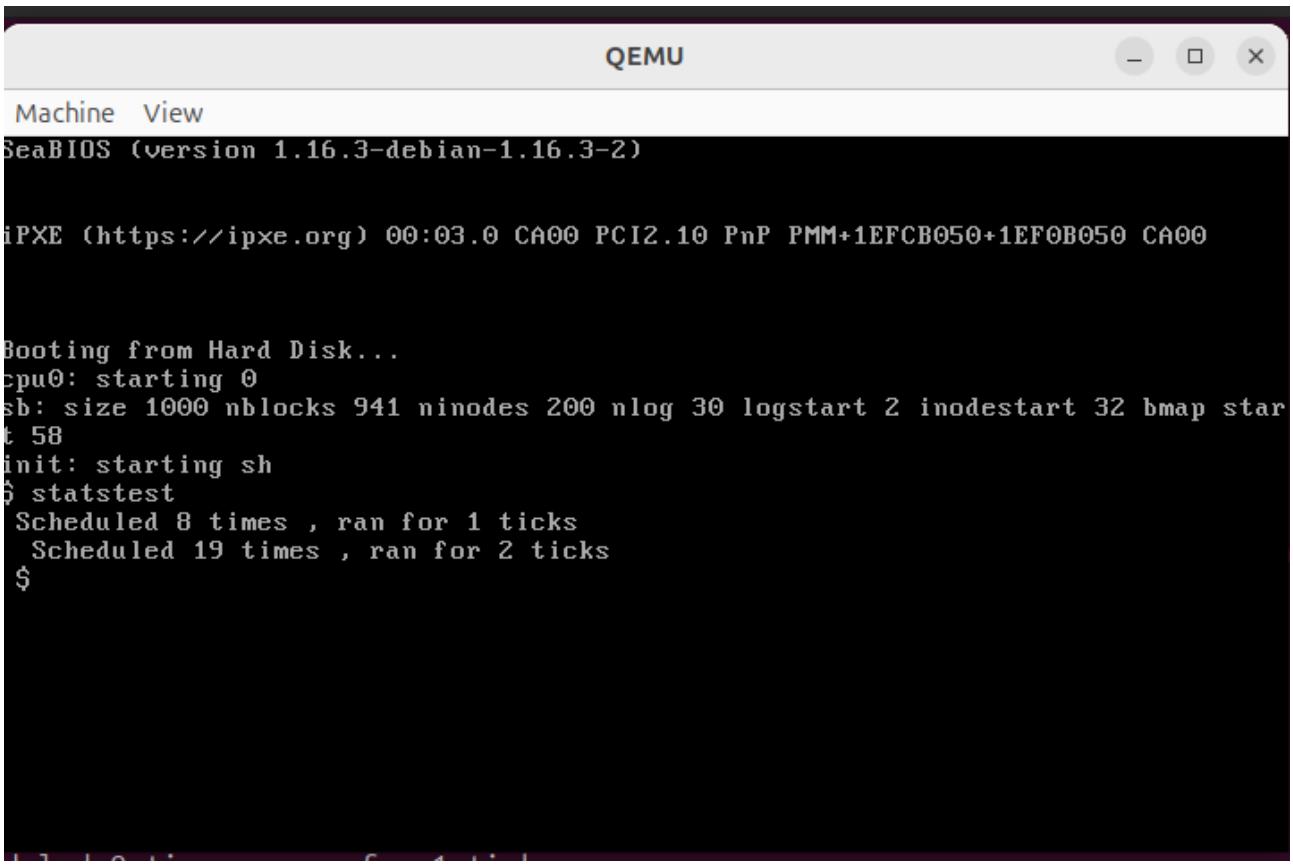
STEP 9)statstest.c

```c
#include "types.h"
#include "user.h"
#include "stat.h"
int main(void){
    int stats[2];
    int i;
    for(i=0;i<2;i++){
        if(getstats(stats)==0){
            printf (1 , " Scheduled %d times , ran for %d ticks \n " , stats [0] , stats [1]) ;
        }
        else{
            printf(2, "Error retrieving stats\n");
        }
        sleep(10);
    }
    exit();
}
```

STEP 10)UPDATING MAKE FILE

```
UPROGS=\
    _cat\
    _echo\
    _forktest\
    _grep\
    _init\
    _kill\
    _ln\
    _ls\
    _mkdir\
    _rm\
    _sh\
    _stressfs\
    _usertests\
    _wc\
    _zombie\
    _mytest\
    _statstest\
```

STEP 11)final output in xv6 os


QEMU
Machine View
SeaBIOS (version 1.16.3-debian-1.16.3-2)
iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1EFCB050+1EF0B050 CA00
Booting from Hard Disk...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
$ statstest
Scheduled 8 times , ran for 1 ticks
Scheduled 19 times , ran for 2 ticks
$