Chandler Bottomely

```
//      via swtch back to the scheduler.
void
scheduler(void)
{
  struct proc *i;
  struct proc *p;
  struct cpu *c = mycpu();
  c->proc = 0;

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

    // Loop over process table looking for process to run.
    acquire(&ptable.lock);
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
      if(p->state != RUNNABLE)
        continue;

      // Switch to chosen process.  It is the process's job
      // to release ptable.lock and then reacquire it
      // before jumping back to us.
      for(i = ptable.proc; i <&ptable.proc[NPROC]; i++){
        if(i->state == RUNNABLE &&i->priority < p->priority)
                p =i;
      }

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

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

      // Process is done running for now.
      // It should have changed its p->state before coming back.
      c->proc = 0;

      // priority decay
      for( i = ptable.proc; i < &ptable.proc[NPROC]; i++){
        if(i->state == RUNNABLE){
           if( i == p && i->priority < 31){
                i->priority = i->priority +1;
           }
           else if(i != p && i->priority >0){
                i->priority = i->priority -1;
           }
}}
//      cprintf("\n process ID = %d, Priority = %d", p->pid, p->priority);
        p= ptable.proc;

"proc.c" 725L, 15606C                                         524,8
```

In proc.c I changed the scheduler to take the higher priority process first. With 0 being the highest priority

```
#include "syscall.h"
#include "traps.h"

#define SYSCALL(name) \
  .globl name; \
  name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret

SYSCALL(fork)
SYSCALL(exit)
SYSCALL(wait)
SYSCALL(pipe)
SYSCALL(read)
SYSCALL(write)
SYSCALL(close)
SYSCALL(kill)
SYSCALL(exec)
SYSCALL(open)
SYSCALL(mknod)
SYSCALL(unlink)
SYSCALL(fstat)
SYSCALL(link)
SYSCALL(mkdir)
SYSCALL(chdir)
SYSCALL(dup)
SYSCALL(getpid)
SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
SYSCALL(wait1)
SYSCALL(exit1)
SYSCALL(waitpid)
SYSCALL(setpriority)
SYSCALL(getpriority)
~
~
```

In usys.S i added setpriority and getpriority

```
void
setpriority(int pid, int priority){
        struct proc *p;
                if(priority <0)
        priority =0;
        else if(priority > 31)
                priority =31;
        acquire(&ptable.lock);
        for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
                if(p->pid == pid)
                        p->priority = priority;
        }
        release(&ptable.lock);
        return;
}

int
getpriority(){
        struct proc *cur = myproc();
        return cur->priority;
}
```

Making setpriority and getpriority

```
extern int sys_wait(void);
extern int sys_wait1(void);
extern int sys_waitpid(void);
extern int sys_write(void);
extern int sys_uptime(void);
extern int sys_setpriority(void);
extern int sys_getpriority(void);

static int (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_exit]    sys_exit,
[SYS_wait]    sys_wait,
[SYS_pipe]    sys_pipe,
[SYS_read]    sys_read,
[SYS_kill]    sys_kill,
[SYS_exec]    sys_exec,
[SYS_fstat]   sys_fstat,
[SYS_chdir]   sys_chdir,
[SYS_dup]     sys_dup,
[SYS_getpid]  sys_getpid,
[SYS_sbrk]    sys_sbrk,
[SYS_sleep]   sys_sleep,
[SYS_uptime]  sys_uptime,
[SYS_open]    sys_open,
[SYS_write]   sys_write,
[SYS_mknod]   sys_mknod,
[SYS_unlink]  sys_unlink,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
[SYS_exit1] sys_exit1,
[SYS_wait1] sys_wait1,
[SYS_waitpid] sys_waitpid,
[SYS_setpriority] sys_setpriority,
[SYS_getpriority] sys_getpriority,
};
```

Added new system calls

```
/ System call numbers
define SYS_fork    1
define SYS_exit    2
define SYS_wait    3
define SYS_pipe    4
define SYS_read    5
define SYS_kill    6
define SYS_exec    7
define SYS_fstat   8
define SYS_chdir   9
define SYS_dup    10
define SYS_getpid 11
define SYS_sbrk   12
define SYS_sleep  13
define SYS_uptime 14
define SYS_open   15
define SYS_write  16
define SYS_mknod  17
define SYS_unlink 18
define SYS_link   19
define SYS_mkdir  20
define SYS_close  21
define SYS_exit1 22
define SYS_wait1 23
define SYS_waitpid 24
define SYS_setpriority 25
define SYS_getpriority 26
```

Adding the new system calls