HW4. XV6 System Call 구현하기

학번: 20150986
이름: 박준영
제출일: 2021.4.5

과제 개요

본 과제에서는 프로세스와 관련된 세 가지 시스템콜인 getnice, setnice, ps를 구현하고 test 코드를 통해 올바른 동작을 확인한다.

소스코드

(수정한 코드를 캡처하고 간단히 설명함)

1) Setnice

프로세스의 존재를 확인하기위해서 exist 변수를 만들어서 확인하며 ptable안에 우리가 찾는 pid가 존재할 경우 p의 priority를 nice로 설정하고 exist는 1로 바꿉니다.
이때 exsit가 0일 경우 -1을 리턴합니다.

```
int setnice(int pid, int nice)
{
    if( nice < 0 || nice > 10)
        return -1;

    int exist = 0;
    struct proc *p;
    acquire(&ptable.lock);
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
            if(p->pid == pid){
                    p->priority = nice;
                    exist = 1;
                    break;
            }
    }
    release(&ptable.lock);
    if(exist == 0) return -1;
    else return 0;
}
```

2) getnice

ptable안에서 우리가 찾고자 하는 pid를 찾은 경우 nice_value를 priority 값으로 넣고 exist를 1로 바꾸어줍니다. 이때 exist가 0 인경우 -1을 리턴합니다.

```
int getnice(int pid)
{
    struct proc *p;
    acquire(&ptable.lock);
    int nice_value = 0;
    int exist = 0;
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
            if(p->pid == 1){
                    p->priority = nice_value = 5;
            }
            if(p->pid == pid){
                    nice_value = p->priority;
                    exist = 1;
                    break;
            }
    }
    release(&ptable.lock);
    if(exist == 0){
            return -1;
    }else{
        return nice_value;
    }

}
```

3) ps

proc.h에서 struct proc에 runtime 변수를 선언해주었습니다.

```
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 priority;                // Process priority
  int runtime;
```

```
void ps(void)
{

   struct proc *p;
   acquire(&ptable.lock);
   cprintf("name\tpid \t state \t priority \t runtime \t tick %d\n",ticks);
   for( p = ptable.proc; p < &ptable.proc[NPROC]; p++)
   {
       if(p->pid==0)
           continue;
       if(p->state==UNUSED)
           cprintf("%s\t%d \t %s \t %d \t %d\n",p->name,p->pid,"UNUSED",p->priority, p->runtime);
       else if(p->state==EMBRYO)
           cprintf("%s\t%d \t %s \t %d \t %d\n",p->name,p->pid,"EMBRYO",p->priority, p->runtime);
       else if(p->state==SLEEPING)
           cprintf("%s\t%d \t %s \t %d \t %d\n",p->name,p->pid,"SLEEPING",p->priority, p->runtime);
       else if(p->state==RUNNABLE)
           cprintf("%s\t%d \t %s \t %d \t %d\n",p->name,p->pid,"RUNNABLE",p->priority, p->runtime);
       else if(p->state==RUNNING)
           cprintf("%s\t%d \t %s \t %d \t %d\n",p->name,p->pid,"RUNNING",p->priority, p->runtime);
       else if(p->state==ZOMBIE)
           cprintf("%s\t%d \t %s \t %d \t %d\n",p->name,p->pid,"ZOMBIE",p->priority, p->runtime);

   }
   release(&ptable.lock);
   return;

}
```

결과

```
$ ps
name    pid     state   priority        runtime         tick 169
init    1       SLEEPING        0       0
sh      2       SLEEPING        0       0
ps      3       RUNNING         0       0
ps
$ test_nice
=== TEST START ===
case 1. get nice value of init process: OK
case 2. get nice value of non-existing process: OK
case 3. set nice value of current process: OK
case 4. set nice value of non-existing process: OK
case 5. set wrong nice value of current process: OK
case 6. get nice value of forked process: WRONG
=== TEST   END ===
$
```