Homework 3: Adding system calls in xv6 - Documentation

System calls:

We added 3 new system calls:

1. getNumProc() system call:
   The system call iterates over the process table of the system and increases a counter every time it finds an active process (active means state != UNUSED).
2. getMaxPid() system call:
   The system call iterates over the process table until it finds the highest PID within the active processes (active process – same definition).
3. getProcInfo(pid, &processInfo) system call:
   the system call receives a required PID and checks if it exists. If it doesn't – return -1.
   If the process exists, the function fills the fields of the struct processInfo with the information about the process.
   The fields: state, parent PID and size are simply accessed from the process table and copied.
   For the field nfd (number of open file descriptors in the process), the function iterates over the open files table of the process and counts how many open files the process has.
   For the field nrswitch (number of context switches performed into the process, an additional field is added to the process struct defined in file proc.h, and additional logic is added to the functions: static struct proc* allocproc(void), void scheduler(void), defined in proc.c file.
   Upon process allocation, the context switch counter is initialized to zero.
   In the scheduler, every time a certain process is chosen to be the next running process, the context switch counter is incremented.

The addition of the system calls is done in several files across the system. We found the relevant locations by searching for other system calls' appearance in the file system, as shown in recitation.

In all the system calls added, every access to the process table is protected by the system's process table lock.

This is done as following:

```
//function logic without process table
  acquire(&ptable.lock);
//use process table data
  release(&ptable.lock);
//function logic without process table
```

Modules:

1. ***Process Info**:*
   New struct added.

```
struct processInfo
 {
    int state;        //process state
    int ppid;         //parent PID
    uint sz;          //size of process memory, in bytes
    int nfd;          //number of open file descriptors in the process
    int nrswitch;     //number of context switches in
};
```

2. ***Proc:***
   The struct was already defined and used in the system. The field count_switches was added.

```
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 count_switches;               // Number of time the scheduler performed context switch
into the process
};
```

User programs:

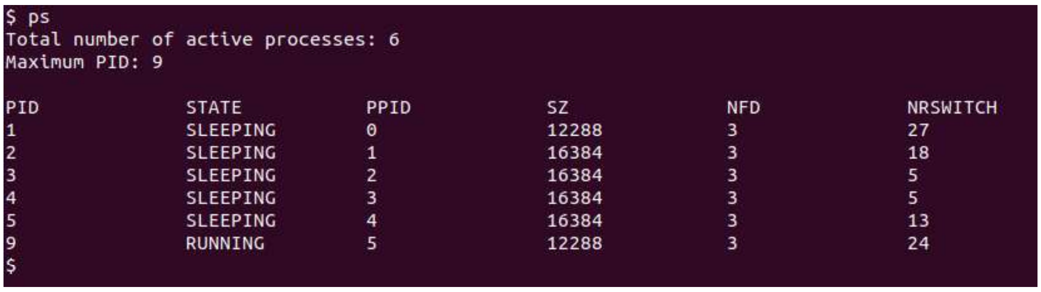
We added a user program ps.c that adds a new shell command – ps.

The program uses the new system calls.

The program outputs a table of all active processes (active – same definition as above) and their properties: PID, state, parent PID, size, number of open file descriptors, number of context switches.

The program is defined in user space, therefore it doesn't have direct access to the process table. To iterate over all the processes, the program first calls getMaxPid system call, and then iterates over all possible PIDs from 0 to maxpid and uses getProcInfo system call on each possible PID.

An example for the command output:



```
$ ps
Total number of active processes: 6
Maximum PID: 9

PID             STATE           PPID            SZ              NFD             NRSWITCH
1               SLEEPING        0               12288           3               27
2               SLEEPING        1               16384           3               18
3               SLEEPING        2               16384           3               5
4               SLEEPING        3               16384           3               5
5               SLEEPING        4               16384           3               13
9               RUNNING         5               12288           3               24
$
```