Documentation Homework 4 | Adding System Calls in xv6

Nicole Frumkin (211615372) & Yaron Silberstein (318321759)

Overview

We added three new system calls to xv6: getNumProc, getMaxPid, and getProcInfo. These help retrieve information about processes in the system and allow us to create a program similar to the ps command in Linux.

System Calls Implemented

1. **getNumProc():**
   What it does:
   Counts and returns the total number of active processes in the system.
   How it works:
   - Goes through the process table (ptable) and counts processes that are not in the UNUSED state.
   - Locks the process table while accessing it to prevent issues with multiple processes accessing it at the same time.

2. **getMaxPid():**
   What it does:
   Finds and returns the highest PID (process ID) among all active processes.
   How it works:
   - Checks each process in the table and keeps track of the largest PID it finds.
   - Like getNumProc, it uses a lock for safety.

3. **getProcInfo(pid, &processInfo):**
   What it does:
   Gets detailed information about a specific process given its PID.

   It updates the following fields in the processInfo struct:

   - Process state (e.g., running, sleeping, etc.)
   - Parent PID (or 0 for the init process)
   - Process memory size (in bytes)
   - Number of open file descriptors
   - Number of context switches the process has gone through

   How it works:
   - Takes the process PID and a pointer to a processInfo structure where the details will be stored.
   - Returns 0 if successful or -1 if no process with the given PID exists.

Changes Made

1. **Created processInfo.h:**
   - Defined the processInfo structure for passing process details between the kernel and user space.

```
struct processInfo {
    int state;        // Process state
    int ppid;         // Parent PID
    int sz;           // Size of process memory, in bytes
    int nfd;          // Number of open file descriptors
    int nrswitch;     // Number of context switches
};
```

   - We included processInfo.h in user.h and proc.c.

2. **Updated System Call Definitions:**
   - Updated syscall.h with new definitions:

```
#define SYS_getNumProc 22
#define SYS_getMaxPid 23
#define SYS_getProcInfo 24
```

   - Registered the system calls in syscall.c so the kernel knows how to handle them.

```
extern int sys_getnumproc(void);
extern int sys_getmaxpid(void);
extern int sys_getprocinfo(void);
[SYS_getNumProc] sys_getnumproc,
[SYS_getMaxPid] sys_getmaxpid,
[SYS_getProcInfo] sys_getprocinfo
```

3. **Added System Call Code:**
   - Wrote the getNumProc, getMaxPid, and getProcInfo functions in proc.c.
   - Ensured they lock the process table while accessing it.

4. **Connected to User Space:**
   - Declared the functions in user.h so user programs can call them:

```
int getNumProc(void);
int getMaxPid(void);
int getProcInfo(int pid, struct processInfo* proc_info);
```

   - Added syscall instructions in usys.S to link the kernel and user-space calls.

```
SYSCALL(getNumProc)
SYSCALL(getMaxPid)
SYSCALL(getProcInfo)
```

5. **proc.c Modifications:**
   - Added code to increment p->nrswitch++ in the scheduler function whenever the process is switched.
   - Calculated nfd by couting how many ofiles there are.
6. **Makefile Modifications:**
   - Added _ps to the UPROGS list in the Makefile to ensure the ps program is included in the build process.