Operating Systems

Homework 4: Adding System Calls in xv6

Students: 208076919, 207477753

---

1. Introduction

In this assignment, we extended the xv6 operating system by implementing three new system calls (getNumProc, getMaxPid, getProcInfo) and a user-space utility (ps). These additions allow users to query and display information about active processes, including their state, memory usage, and context switch counts.

2. Summary of Changes

A. Header Files

- processInfo.h (New): Defined struct processInfo to transfer process data (state, ppid, sz, nfd, nrswitch) from kernel to user space.
- proc.h: Added the integer field 'nrswitch' to 'struct proc' to track context switches.
- user.h: Added prototypes for the new system calls and included processInfo.h.
- syscall.h: Defined new system call numbers: SYS_getNumProc (22), SYS_getMaxPid (23), and SYS_getProcInfo (24).

B. Kernel Implementation (proc.c)

- allocproc(): Initialized p->nrswitch = 0 when a new process is allocated.
- scheduler(): Added logic to increment p->nrswitch whenever a process is context-switched into the CPU (state changes to RUNNING).
- getNumProc(): Implemented logic to lock ptable, count active processes (state != UNUSED), and release the lock.
- getMaxPid(): Implemented logic to find the maximum PID among active processes.
- getProcInfo(pid, info): Implemented the core logic to populate the user structure:
  * Copies state, sz, nrswitch from struct proc.
  * Calculates nfd by counting non-null entries in p->ofile[].
  * Handling parent PID: Returns 0 if pid is 1 (init), otherwise returns p->parent->pid.

C. System Call Interface (sysproc.c & syscall.c)

- sysproc.c: Added wrapper functions (sys_getNumProc, etc.). Used argptr() to safely validate the user-space pointer passed to sys_getProcInfo.
- syscall.c: Registered the new system calls in the syscalls[] table.

D. User Programs

- ps.c (New): Implemented the 'ps' command. It calls getNumProc and getMaxPid for the header, then loops from 1 to MaxPid. For every valid process found via getProcInfo, it prints the details (PID, State, PPID, SZ, NFD, NRSWITCH) aligned with tabs.
- Makefile: Added '_ps' to the UPROGS list to include the program in the file system.

3. Verification

We verified the implementation using two methods:

1. Logic Test: A custom C program (test_hw4) confirmed that:
   - getNumProc increases after fork.
   - nfd increases correctly when files are opened.
   - sz increases by 4096 bytes after sbrk(4096).
   - nrswitch increases after forcing context switches (via sleep).
2. Visual Test: The 'ps' command output was manually checked. It correctly displays the header, sorts processes by PID, aligns columns, and translates state integers to strings.
   - Confirmed that 'init' (PID 1) correctly shows PPID 0.
   - Confirmed that sleeping processes correctly show 'sleeping' state.