Itai Plotkin- 207050717, Gilad Rabinovitch- 209285469

HW4 OS XV6

To complete the homework assignment, we modified several of the original xv6 files. Below is an outline of the changes made along with a brief description of each modification:

- **Proc.h**
  We updated the proc structure definition by adding two new fields: nrswitch (to record the number of context switches) and nfd (to track the number of open file descriptors).
- **Proc.c**
  In this file, we enhanced the proc structure to include the fields nrswitch and nfd. The allocproc() function was altered to initialize these fields for any new process. Additionally, the scheduler() function was updated to increment nrswitch each time a process is scheduled.
- **Syscall.h**
  New constants were defined here to serve as indices in the syscall jump table for the new system calls.
- **Syscall.c**
  We introduced new entries into the syscall jump table for the system calls: getNumProc, getMaxPid, and getProcInfo.
- **Sysfile.h**
  Several file-related system calls were updated:
  - **nfd Tracking:** The sys_open and sys_close calls were modified to update the nfd field in the process structure, ensuring the count of open file descriptors remains accurate.
  - **fdalloc Function:** This was modified to increment the nfd counter when a new file descriptor is allocated.
  - **sys_close Function:** Adjusted to decrement the nfd counter when a file descriptor is closed.
- **Sysproc.c**
  We implemented the new system calls (getNumProc, getMaxPid, and getProcInfo) in this file. These calls make use of helper functions and the process tracking enhancements established in proc.c.
- **User.h**
  Prototypes for the new system calls were added here, enabling them to be accessed by user-space programs.
- **Usys.S**
  Assembly linkage was added for the three new system calls (getNumProc, getMaxPid, and getProcInfo). This setup allows user applications to request process information, akin to the UNIX ps command.
- **ProcessInfo.h**
  A new structure, processInfo, was defined in this file. It is used by the getProcInfo system call to pass process details from the kernel to user space.
- **Ps.c**
  We created the ps.c program to utilize the new system calls (getNumProc, getMaxPid, and getProcInfo) for gathering information on the processes running within the xv6 system. The program retrieves and displays various process attributes, including:
  - Process ID (pid)
  - Process state (such as embryo, running, runnable, sleeping, or zombie)
  - Parent process ID (ppid)

- Memory size (sz)
- The number of open file descriptors (nfd)
- The number of context switches (nrswitch)

Explanation of the Three New System Call Functions

- **getNumProc**
  This function returns the total count of active processes in the system. An active process is defined as any process that is not in the unused state, meaning it may be in a running, runnable, sleeping, or zombie state. The function iterates through the entire process table and counts only those processes that are active.
- **getMaxPid**
  This function scans the process table for the highest process ID (pid) among those that are active (i.e., not in the unused state). It returns the largest pid found.
- **getProcInfo**
  This function accepts a pid and a pointer to a structure. It populates the structure with specific details about the process, including its state, parent process ID (ppid), memory usage (in bytes), number of open file descriptors (nfd), and the number of times it has been context-switched (nrswitch).
- **Makefile**
  Finally, the Makefile was updated to include the new user program ps in the list of programs to be compiled into the xv6 filesystem, ensuring that the ps command is available within the xv6 environment.