Kevin Jiang

12/9/2020

Problem Set 7 Short Answers

A. If system call is set to -1, the instruction cmpl $nr_syscalls, %eax will determine that the system call number in %eax (which is now -1) is not within the bounds of known system call numbers, and as a result the instruction jae syscall_badsys will cause the program to jump to syscall_badsys. Here, -ENOSYS is moved into the slot into the kernel stack where user's eax register was saved. The system call then exists, and -ENOSYS will be popped back into %eax. The UNIX API will return -1 and set errno to %eax, which is -ENOSYS.
B. Whether or not there is additional rescheduling after exiting the system call, the program will always go through the restore_all label. At this label, the macro RESTORE_REGS pops all of the registers from kernel stack back to their original registers before returning to user mode. The register that changes is the return value register %eax. After this macro, iret instruction is executed, which restores five registers including eflags. Upon restoration of eflags, privilege level is set back to user level
C. i) For a pre-emptive Linux Kernel, PID 999 is prioritized. Upon return from the disk I/O interrupt handler, the TIF_NEED_RESCHED flag is noticed and the system call jumps to the work_resched label (from entry.S). Task with PID 999 gets the GPU while PID 123 is suspended. Sometime later, PID 123 will get the CPU again and tasks will resume
ii) For Linux Kernel with pre-emption off, the system call of PID 123 is handled first. After the system call finishes and before returning to user mode, at syscall_exit_work the TIF_NEED_RESCHED is noticed and the work_resched will operate. The CPU will be given to PID 999 before returning to PID 123 sometime later.