Operating Systems - Homework 2

Subata Naveen Khan - 18119

3rd October 2024

1. `ecall` is a RISC-V instruction that raises the hardware privilege level from user mode to supervisor mode. A user program in xv6 the `ecall` instruction to call a system call. `ecall` triggers the CPU to jump to the supervisor trap handler in `trap.c`. The trap handler checks the `a7` register for the system call number and handles it accordingly from the kernel.

2. The main reason this code cannot be implemented in xv6 is that xv6 is a bare bones operating system with limited functionality, so it does not support the operations required:

   - xv6 does have an `ls` command, but it doesn't have higher-level functionality such as `-al`.
   - xv6 does not have a built-in `tr` command.
   - The xv6 implementations of some system calls such as `fork()` and `wait()` are very limited in comparison to standard Unix systems.
   - Other functions like `perror()` and `execvp()` are also not available in xv6 and would require modifications to the code to implement them.

3.
   - Store the table in the kernel memory with no reference to it in user space.
   - Mark the pages storing the table data to not give user space the permission to read or write to them.
   - Kernel Address Space Layout Randomization (KASLR): Randomize the location of the table every time the system boots.
   - Prevent writing access to the table completely after initialization is complete.

4. a. `spinlock`: used to disable interrupts and implement mutual exclusion in multi-core environments.
   b. `proc *parent`: points to the parent process in order to maintain the process hierarchy.
   c. `trapframe`: used to store the CPU's registers that contain the process's state when a trap occurs, to be restored after the kernel deals with the trap.
   d. `context`: used to store the CPU's registers that contain the process's state when context switching to another process, to be restored when the kernel resumes execution.
   e. `file *ofile[]`: used to store pointers to all the open file descriptors that the process has opened
   f. `inode *cwd`: used to keep track of the process's location.

5.
- Implement KASLR.
- Ensure that system calls cannot be made directly from the user space.
- Check that arguments passed from the user space point to regions in the memory that the user space is allowed access to.

6. The `t` registers are temporary, so they are not preserved when a trap or context switch occurs. So the system call number cannot be placed there.

   The `s` registers are saved with the rest of the state of the process, so they contain relevant to the process, which the system call number is not.

   The `a` registers are used to pass information between functions. So it makes sense for the system call number, which is an argument used by the `syscall()` function, to be stored in them.

7. The CLINT handles local software and timer interrupts that are specific to the core. It is used for inter-process communication and time-sharing processes.

   The PLIC handles hardware interrupts from external devices such as a disk, keyboard, or network interface. It determines the priority of each interrupt and is responsible for routing them to the appropriate cores.

8. When a system call is made, `usys.S` places the relevant system call number (defined in `kernel/syscall.h`) into the `a7` register. Due to the hacking, the number stored in `a7` will not correlate with the values in `syscall.h`. Then `ecall` will be executed. In the kernel, the trap handler will check the cause of the trap and accordingly call the `syscall()` function in the `kernel/syscall.c` file. The value in `a7` will be stored in the `num` variable. In the `if` statement, the **`num < NELEM(syscalls)`** condition will return false, so the intended system call will not be executed and the `else` block will run instead.