Exercise2

Shulei Xu

1. When a user process is interrupted or causes a processor exception, the x86 hardware switches the stack pointer to a kernel stack, before saving the current process state. Explain why.

First it could guarantee reliability. The process’s user-level stack pointer sometimes is not a valid memory address when the program has bugs, but the kernel handler must continue to work properly.

Second it could guarantee security. On a multiprocessor like x86, other threads running in the same process can modify user memory during the system call. If the kernel handler stores its local variables on the user-level stack, the user program may be able to modify kernel’s return address, cause the kernel to jump to arbitrary code.

1. For the “Hello world” program, we mentioned that the kernel must copy the string from the user program to screen memory. Why must the screen’s buffer memory be protected? Explain what might happen if a malicious application could alter any pixel on the screen, not just those within its own window.

The screen’s buffer memory must be protected because the applications could be untrusted and buggy. If a malicious application could alter any pixel on the screen, some buggy application would ruin other applications displaying windows, or even the entire screen would be in chaos if some buggy application overflow the screen buffer.

1. For each of the three mechanisms that supports dual-mode operation — privileged instructions, memory protection, and timer interrupts — explain what might go wrong without that mechanism, assuming the system still had the other two.

If without privileged instructions, all instructions would be able to executed by any process. Large number of applications would be allowed to access privileged data and even control the entire computer, which would let the operating system crash.

If without memory protection, applications would be able to read and write other applications’ memory. So other applications’ or even the operating system’s data may be changed, and that application would potentially be able to take control over the system.

If without timer interrupts, the operating system would not able to take control of processor periodically. If some buggy application running on the processor with infinite loop, the operating system would not able to take processor back and let other applications run.

1. Define three types of user-mode to kernel-mode transfers.

Interrupts: A interrupts is an asynchronous signal to the processor that some external event has occurred that may require its attention.

Processor exceptions: A processor exception is a hardware event caused by user program behavior that causes a transfer of control to the kernel.

System calls: A system call is any procedure provided by the kernel that can be called from user level.

1. Define four types of kernel-mode to user-mode transfers.

New process: To start a new process, kernel copies the program into memory, sets the program counter to the first instruction of the process, sets the stack pointer to the base of user tack, and switches to user mode.

Resume after an interrupt, processor exception, or system call: After kernel finishes handling the request, it resumes execution of the interrupted process by restoring its program counter and registers, and changing mode back to user level

Switch to a different process. In some cases like a timer interrupt, the kernel switches to another process from current running one.

User-level upcall: Many operating systems provide user programs with the ability to receive asynchronous notification of events delivered by the kernel to a user-level process.

1. Most hardware architectures provide an instruction to return from an interrupt, such as iret. This instruction switches the mode of operation from kernel-mode to user-mode.
2. Explain where in the operating system this instruction would be used.

This instruction is used when the operating system wants to give control back to user-level program. For example, in the end of context switch, or interrupt handling is finished.

1. Explain what happens if an application program executes this instruction.

If an application program executes this instruction, it will cause a hardware exception because iret instruction is a privileged instruction, an application program cannot execute this.

15. Explain the steps that an operating system goes through when the CPU receives an interrupt.

1. Save current stack pointer, program counter and processor status word

2. Switch to kernel stack and put stack pointer, program counter and processor status word onto stack

3. Switch to kernel mode

4. Invoke interrupt handler, saves registers it might clobber.

5. Restore the interrupted process, (back to user mode)