Module XXIV

Exception Handling

General Idea

- A computer includes extra hardware mechanisms that monitor the processor and I/O operations, and inform the processor when they detect errors or problems
- A variety of terms have been used to describe problems that occur
  - Early computers used *check* or *machine check*
  - Some vendors use *trap* or *fault* (especially when a referenced page is not resident)
  - The term *exception* refers broadly to many types of unusual situations as well as problems

Seven Key Types Of Exceptions

- Device interrupts
- Arithmetic exceptions
- Illegal memory references
- I/O errors (e.g., bus errors)
- Protection faults
- Invalid instructions
- Hardware failures

Handling Exceptions

- Typical hardware only *reports* exceptions
- The hardware informs the operating system when an exception occurs, and the operating system chooses how to handle the exception
- Possible actions include
  - Ignore the exception
  - Reset the hardware that raised the exception (e.g., clear the bus)
  - Identify and signal the process that caused the exception
  - Terminate the process that caused the exception
  - Reboot the system (especially embedded systems)
  - Panic and shut down the system

Implementation Of Panic In Xinu

```
/* panic.c - panic */

#include <xinu.h>

/*------------------------------------------------------------------------
 * panic  -  Display a message and stop all processing
 *------------------------------------------------------------------------
 */
void    panic (
         char   *msg                    /* Message to display           */
        )
{
        disable();                      /* Disable interrupts           */
        kprintf("\n\n\rpanic: %s\n\n", msg);
        while(TRUE) {;}                 /* Busy loop forever            */
}
```

- Note: on some hardware, Xinu can power off the system

The Complexity Of Exception Handling

- Suppose process *P* is running when an exception occurs
- Should the exception be attributed to the process?
- Answer: if process *P* made a system call, the exception should be attributed with the operating system rather than the process, but making that distinction requires knowing exactly what a process is doing

The Complexity Of Exception Handling (continued)

- As another example of complexity, suppose process *P* somehow writes into the memory of process *Q*, causing *Q* to raise an exception
- How can the exception be attributed to *P*?

Questions?