Aside Software exceptions in C++ and Java

The exception mechanisms provided by C++ and Java are higher-level, more structured versions of the C `setjmp` and `longjmp` functions. You can think of a `catch` clause inside a `try` statement as being akin to a `setjmp` function. Similarly, a `throw` statement is similar to a `longjmp` function.

```
linux> ./restart
starting
processing...
processing...
Ctrl+C
restarting
processing...
Ctrl+C
restarting
processing...
```

There a couple of interesting things about this program. First, To avoid a race, we must install the handler *after* we call `sigsetjmp`. If not, we would run the risk of the handler running before the initial call to `sigsetjmp` sets up the calling environment for `siglongjmp`. Second, you might have noticed that the `sigsetjmp` and `siglongjmp` functions are not on the list of async-signal-safe functions in Figure 8.33. The reason is that in general `siglongjmp` can jump into arbitrary code, so we must be careful to call only safe functions in any code reachable from a `siglongjmp`. In our example, we call the safe `sio_puts` and `sleep` functions. The unsafe `exit` function is unreachable.

8.7 Tools for Manipulating Processes

Linux systems provide a number of useful tools for monitoring and manipulating processes:

STRACE. Prints a trace of each system call invoked by a running program and its children. It is a fascinating tool for the curious student. Compile your program with `-static` to get a cleaner trace without a lot of output related to shared libraries.

PS. Lists processes (including zombies) currently in the system.

TOP. Prints information about the resource usage of current processes.

PMAP. Displays the memory map of a process.

`/proc`. A virtual filesystem that exports the contents of numerous kernel data structures in an ASCII text form that can be read by user programs. For example, type `cat /proc/loadavg` to see the current load average on your Linux system.

8.8 Summary

Exceptional control flow (ECF) occurs at all levels of a computer system and is a basic mechanism for providing concurrency in a computer system.

At the hardware level, exceptions are abrupt changes in the control flow that are triggered by events in the processor. The control flow passes to a software handler, which does some processing and then returns control to the interrupted control flow.

There are four different types of exceptions: interrupts, faults, aborts, and traps. Interrupts occur asynchronously (with respect to any instructions) when an external I/O device such as a timer chip or a disk controller sets the interrupt pin on the processor chip. Control returns to the instruction following the faulting instruction. Faults and aborts occur synchronously as the result of the execution of an instruction. Fault handlers restart the faulting instruction, while abort handlers never return control to the interrupted flow. Finally, traps are like function calls that are used to implement the system calls that provide applications with controlled entry points into the operating system code.

At the operating system level, the kernel uses ECF to provide the fundamental notion of a process. A process provides applications with two important abstractions: (1) logical control flows that give each program the illusion that it has exclusive use of the processor, and (2) private address spaces that provide the illusion that each program has exclusive use of the main memory.

At the interface between the operating system and applications, applications can create child processes, wait for their child processes to stop or terminate, run new programs, and catch signals from other processes. The semantics of signal handling is subtle and can vary from system to system. However, mechanisms exist on Posix-compliant systems that allow programs to clearly specify the expected signal-handling semantics.

Finally, at the application level, C programs can use nonlocal jumps to bypass the normal call/return stack discipline and branch directly from one function to another.

Bibliographic Notes

Kerrisk is the essential reference for all aspects of programming in the Linux environment [62]. The Intel ISA specification contains a detailed discussion of exceptions and interrupts on Intel processors [50]. Operating systems texts [102, 106, 113] contain additional information on exceptions, processes, and signals. The classic work by W. Richard Stevens [111] is a valuable and highly readable description of how to work with processes and signals from application programs. Bovet and Cesati [11] give a wonderfully clear description of the Linux kernel, including details of the process and signal implementations.