A

TRACING SYSTEM CALLS

The *strace* command allows us to trace the system calls made by a program. This is useful for debugging, or simply to find out what a program is doing. In its simplest form, we use *strace* as follows:

```
$ strace command arg...
```

This runs *command*, with the given command-line arguments, producing a trace of the system calls it makes. By default, *strace* writes its output to *stderr*, but we can change this using the *−o filename* option.

Examples of the type of output produced by *strace* include the following (taken from the output of the command *strace date*):

```
execve("/bin/date", ["date"], [/* 114 vars */]) = 0
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=111059, ...}) = 0
mmap2(NULL, 111059, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7f38000
close(3)                                = 0
open("/lib/libc.so.6", O_RDONLY)        = 3
fstat64(3, {st_mode=S_IFREG|0755, st_size=1491141, ...}) = 0
close(3)                                = 0
write(1, "Mon Jan 17 12:14:24 CET 2011\n", 29) = 29
exit_group(0)                           = ?
```

Each system call is displayed in the form of a function call, with both input and output arguments shown in parentheses. As can be seen from the above examples, arguments are printed in symbolic form:

- Bit masks are represented using the corresponding symbolic constants.
- Strings are printed in text form (up to a limit of 32 characters, but the *−s strsize* option can be used to change this limit).
- Structure fields are individually displayed (by default, only an abbreviated subset of large structures is displayed, but the *−v* option can be used to display the whole structure).

After the closing parenthesis of the traced call, *strace* prints an equal sign (=), followed by the return value of the system call. If the system call failed, the symbolic *errno* value is also displayed. Thus, we see `ENOENT` displayed for the failure of the *access()* call above.

Even for a simple program, the output produced by *strace* is made voluminous by the system calls executed by the C run-time startup code and the loading of shared libraries. For a complex program, the *strace* output can be extremely long. For these reasons, it is sometimes useful to selectively filter the output of *strace*. One way to do this is to use *grep*, like so:

```
$ strace date 2>&1 | grep open
```

Another method is to use the *−e* option to select the events to be traced. For example, we can use the following command to trace *open()* and *close()* system calls:

```
$ strace -e trace=open,close date
```

When using either of the above techniques, we need to be aware that, in a few cases, the true name of a system call differs from the name of its *glibc* wrapper. For example, though we refer to all of the *wait()*-type functions as system calls in Chapter 26, most of them (*wait()*, *waitpid()*, and *wait3()*) are wrappers that invoke the kernel's *wait4()* system call service routine. This latter name is displayed by *strace*, and we must specify that name in the *−e trace=* option. Similarly, all of the *exec* library functions (Section 27.2) invoke the *execve()* system call. Often, we can make a good guess about such transformations by looking at the *strace* output (or looking at the output produced by *strace −c*, described below), but, failing that, we may need to check the *glibc* source code to see what transformations may be occurring inside wrapper functions.

The *strace(1)* manual page documents a host of further options to *strace*, including the following:

- The *−p pid* option is used to trace an existing process, by specifying its process ID. Unprivileged users are restricted to tracing only processes that they own and that are not executing set-user-ID or set-group-ID programs (Section 9.3).
- The *−c* option causes *strace* to print a summary of all system calls made by the program. For each system call, the summary information includes the total number of calls, the number of calls that failed, and the total time spent executing the calls.

- The *-f* option causes children of this process also to be traced. If we are sending trace output to a file (*-o filename*), then the alternative *-ff* option causes each process to write its trace output to a file named *filename.PID*.

The *strace* command is Linux-specific, but most UNIX implementations provide their own equivalents (e.g., *truss* on Solaris and *ktrace* on the BSDs).

> The *ltrace* command performs an analogous task to *strace*, but for library functions. See the *ltrace(1)* manual page for details.