Chapter 1

1.1 For this exercise, we use the following two arguments for the `ls`(1) command: `-i` prints the i-node number of the file or directory (we say more about i-nodes in Section 4.14), and `-d` prints information about a directory instead of information on all the files in the directory.

Execute the following:

```
$ ls -ldi /etc/. /etc/..          -i says print i-node number
 162561 drwxr-xr-x  66 root    4096 Feb  5 03:59 /etc/./
      2 drwxr-xr-x  19 root    4096 Jan 15 07:25 /etc/../
$ ls -ldi /. /..                  both . and .. have i-node number 2
      2 drwxr-xr-x  19 root    4096 Jan 15 07:25 /./
      2 drwxr-xr-x  19 root    4096 Jan 15 07:25 /../
```

1.2 The UNIX System is a multiprogramming, or multitasking, system. Other processes were running at the time this program was run.

1.3 Since the *msg* argument to `perror` is a pointer, `perror` could modify the string that *msg* points to. The qualifier `const`, however, says that `perror` does not modify what the pointer points to. On the other hand, the error number argument to `strerror` is an integer, and since C passes all arguments by value, the `strerror` function couldn't modify this value even if it wanted to. (If the handling of function arguments in C is not clear, you should review Section 5.2 of Kernighan and Ritchie [1988].)

1.4 During the year 2038. We can solve the problem by making the `time_t` data type a 64-bit integer. If it is currently a 32-bit integer, applications will have to be recompiled to work properly. But the problem is worse. Some file systems and backup media store times in 32-bit integers. These would need to be updated as well, but we still need to be able to read the old format.

1.5 Approximately 248 days.

Chapter 1 Supplemental Exercises and Answers

1.6 **EXERCISE:** Most UNIX system programs are relatively small and designed to do one thing well. List at least three advantages to this approach.

SOLUTION:

1. Smaller programs are easier to maintain.
2. Smaller programs are easier to test.
3. It is easier to have confidence that a smaller program does what it is supposed to do than a larger program.

4. Small programs can be combined in interesting ways to solve new problems. For example, to find all of the misspelled words in a document, you might type

```
spell filename | sort | uniq
```

This is superior to building a spelling checker into every document editor.

1.7 **EXERCISE:** Usually the CPU time of a program doesn't exceed the wall clock time (the running time) of the program. Explain the circumstances under which this can be false.

SOLUTION: On a multiprocessor, if an application is multithreaded, more than one processor can accumulate CPU time at the same time, so it is possible for the CPU time to exceed the running time of the program.

Chapter 2

2.1 The following technique is used by FreeBSD. The primitive data types that can appear in multiple headers are defined in the header `<machine/_types.h>`. For example:

```
#ifndef _MACHINE__TYPES_H_
#define _MACHINE__TYPES_H_

typedef int             __int32_t;
typedef unsigned int    __uint32_t;
    ⋮

typedef __uint32_t      __size_t;
    ⋮

#endif  /* _MACHINE__TYPES_H_ */
```

In each of the headers that can define the `size_t` primitive system data type, we have the sequence

```
#ifndef  _SIZE_T_DECLARED
typedef __size_t       size_t;
#define _SIZE_T_DECLARED
#endif
```

This way, the `typedef` for `size_t` is executed only once.

2.2 This is an activity for the reader, intended to increase familiarity with the system header files.

2.3 If OPEN_MAX is indeterminate or ridiculously large (i.e., equal to LONG_MAX), we can use getrlimit to get the per-process maximum for open file descriptors. Since the per-process limit can be modified, we can't cache the value obtained from the previous call (it might have changed). See Figure 1.

```
#include "apue.h"
#include <limits.h>
#include <sys/resource.h>

#define OPEN_MAX_GUESS  256

long
open_max(void)
{
    long openmax;
    struct rlimit rl;

    if ((openmax = sysconf(_SC_OPEN_MAX)) < 0 ||
      openmax == LONG_MAX) {
        if (getrlimit(RLIMIT_NOFILE, &rl) < 0)
            err_sys("can't get file limit");
        if (rl.rlim_max == RLIM_INFINITY)
            openmax = OPEN_MAX_GUESS;
        else
            openmax = rl.rlim_max;
    }
    return(openmax);
}
```

Figure 1 Alternative method for identifying the largest possible file descriptor

Chapter 2 Supplemental Exercises and Answers

2.4 **EXERCISE:** Use sysconf to determine the maximum number of standard I/O streams a process can open at a time. Then try to open this number of streams. How many can you open? Does it match what sysconf reports?

SOLUTION: The results differ depending on which platform you use. Solaris 10 reports STREAM_MAX to be 256, but we can open only 253 standard I/O streams. This is because our program already has 3 open streams when it starts: standard input, standard output, and standard error. On Mac OS X 10.6.8, the behavior is similar to Solaris. FreeBSD 8.0 reports STREAM_MAX to be 20,000, but we can only open 15,098 standard I/O streams. In this case, we probably are running into the system-wide limit for open files. Linux 3.2.0 reports STREAM_MAX to be 16, but we can open 1021 streams. This means the limit is really 1024.