8.4.6　利用 fork 和 execve 运行程序

像 Unix shell 和 Web 服务器这样的程序大量使用了 fork 和 execve 函数。shell 是一个交互型的应用级程序，它代表用户运行其他程序。最早的 shell 是 sh 程序，后面出现了一些变种，比如 csh、tcsh、ksh 和 bash。shell 执行一系列的读/求值(read/evaluate)步骤，然后终止。读步骤读取来自用户的一个命令行。求值步骤解析命令行，并代表用户运行程序。

图 8-23 展示了一个简单 shell 的 main 例程。shell 打印一个命令行提示符，等待用户在 stdin 上输入命令行，然后对这个命令行求值。

code/ecf/shellex.c

```
1   #include "csapp.h"
2   #define MAXARGS   128
3
4   /* Function prototypes */
5   void eval(char *cmdline);
6   int parseline(char *buf, char **argv);
7   int builtin_command(char **argv);
8
9   int main()
10  {
11      char cmdline[MAXLINE]; /* Command line */
12
13      while (1) {
14          /* Read */
15          printf("> ");
16          Fgets(cmdline, MAXLINE, stdin);
17          if (feof(stdin))
18              exit(0);
19
20          /* Evaluate */
21          eval(cmdline);
22      }
23  }
```

code/ecf/shellex.c

图 8-23　一个简单的 shell 程序的 main 例程

图 8-24 展示了对命令行求值的代码。它的首要任务是调用 parseline 函数(见图 8-25)，这个函数解析了以空格分隔的命令行参数，并构造最终会传递给 execve 的 argv 向量。第一个参数被假设为要么是一个内置的 shell 命令名，马上就会解释这个命令，要么是一个可执行目标文件，会在一个新的子进程的上下文中加载并运行这个文件。

如果最后一个参数是一个“&”字符，那么 parseline 返回 1，表示应该在后台执行该程序(shell 不会等待它完成)。否则，它返回 0，表示应该在前台执行这个程序(shell 会等待它完成)。

在解析了命令行之后，eval 函数调用 builtin_command 函数，该函数检查第一个命令行参数是否是一个内置的 shell 命令。如果是，它就立即解释这个命令，并返回值 1。否则返回 0。简单的 shell 只有一个内置命令——quit 命令，该命令会终止 shell。实际使用