```
#include <sys/types.h>                    /*标准的POSIX头文件*/
#include <sys/stat.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
struct stat sbuf;                         /*用于stat调用，看文件是否为sym连接*/

search(char *dir_name)
{                                         /*可执行表的递归查找*/
    DIR *dirp;                            /*指向打开目录流的指针*/
    struct dirent *dp;                    /*指向目录项的指针*/

    dirp = opendir(dir_name);             /*打开此目录*/
    if (dirp == NULL) return;             /*不能打开目录时返回*/
    while (TRUE) {
        dp = readdir(dirp);               /*读下一个目录项*/
        if (dp == NULL) {                 /*NULL表示已完成操作*/
        chdir ("..");                     /*回到父目录*/
        break;                            /*跳出循环*/
    }
    if (dp->d_name[0] == '.') continue;   /*跳过 "." 和 ".." 目录*/
    lstat(dp->d_name, &sbuf);             /*项是符号连接吗？ */
    if (S_ISLNK(sbuf.st_mode)) continue; /*跳过符号连接*/
    if (chdir(dp->d_name) == 0) {         /*如果chdir成功，则必定是目录*/
        search(".");                      /*如果是，进入目录查询*/
    } else {                                  /*无（文件），则感染*/
        if (access(dp->d_name,X_OK) == 0) /*如是可执行文件就感染*/
            infect(dp->d_name);
    }
    closedir(dirp);                       /*dir运行完毕，关闭程序并返回*/
}
```

图9-27 在UNIX系统上查找可执行文件的递归过程

病毒可以通过很多种方法不断“改善”。第一，可以在infect里插入产生随机数的测试程序然后悄然返回。如调用超过了128次病毒就会感染，这样就降低了病毒在大范围传播之前就被被检测出来的概率。生物病毒也具有这样的特性：那些能够迅速杀死受害者的病毒不如缓慢发作的病毒传播得快，慢发作给了病毒以更多的机会扩散。另外一个方法是保持较高的感染率（如25%），但是一次大量感染文件会降低磁盘性能，从而易于被发现。

第二，infect可以检查文件是否已被感染。两次感染相同的文件无疑是浪费时间。第三，可以采取方法保持文件的修改时间及文件大小不变，这样可以协助把病毒代码隐藏起来。对大于病毒的程序来说，感染后程序大小将保持不便；但对小于病毒大小的程序来说，感染后程序将变大。多数病毒都比大多数程序小，所以这不是一个严重的问题。

一般的病毒程序并不长（整个程序用C语言编写不超过1页，文本段编译后小于2KB），汇编语言编写的版本将更小。Ludwig（1998）曾经给出了一个感染目录里所有文件的MS-DOS病毒，用汇编语言编写并编译后仅有44个字节。

稍后的章节将研究反病毒程序，这种反病毒程序可以跟踪病毒并除去它们。而且，在图9-27里很有趣的情况是，病毒用来查找可执行文件的方法也可以被反病毒程序用来跟踪被感染的文件并最终清除病毒。感染机制与反感染机制是相辅相成的，所以为了更有效地打击病毒，我们必须详细理解病毒工作的原理。

从Virgil的观点来说，病毒的致命问题在于它太容易被发现了。毕竟当被感染的程序运行时，病毒就会感染更多的文件，但这时该程序就并不能正常运行，那么用户就会立即发现。所以，有相当多的病毒把自己附在正常程序里，在病毒发作时可以让原来的程序正常工作。这类病毒叫做寄生病毒（parasitic virus）。

寄生病毒可以附在可执行文件的前端、后端或者中间。如果附在前端，病毒首先要把程序复制到RAM中，把自己附加到程序前端，然后再从RAM里复制回来，整个过程如图9-28b所示。遗憾的是，这时的程序不会在新的虚拟地址里运行，所以病毒要么在程序被移动后重新为该程序分配地址，要么在完成自己的操作后缩回到虚拟地址0。