**实验二 目录树的遍历**

实验二主要以课本99－102页程序4-7为框架，在此基础上进行扩展。需要注意的问题主要有：

1. 正确理解程序4-7。程序4-7递归降序遍历目录层次结构，并按照文件类型进行计数。主要涉及到三个函数，ftw4(), dopath()和myfunc()。ftw4()函数以所带参数pathname为要遍历的起始目录，计算出该目录下各种不同类型的文件的个数和所占百分比，并显示出来。它调用了另外两个函数，一个是dopath()函数，这是一个递归函数，对指定的起始目录下的每个目录项，按深度优先进行遍历；而对所访问的节点，则调用myfunc()进行处理。main函数输出统计结果。三个函数的参数含义如下：
2. myftw(char\* pathname, Myfunc\* func);

pathname给出指要遍历开始的目录。

func是Myfunc类型的函数指针，定义访问的实际操作。

1. dopath(Myfunc\* func)；

参考第一个函数

1. int myfunc(const char \* pathname,const struct stat \*statptr,int type)；

pathname指向当前访问节点的路径名。

Statptr指向当前访问节点的i-节点的结构，该结构保存有许多该文件的 信息。

type给出当前访问节点的类型，在实验中可以自己定义它的涵义。

myfunc()的返回值通常是0，实际上在程序4-7中它的值总是0。但是在 dopath()函数中，myfunc()的返回值非0意味着终止遍历。

另外，程序4-7中用到课本2-3中的函数path\_alloc()，用于分配存放路径名的内存空间。

2．实验二要求根据用户输入的命令行选项的不同，来实现三种功能：

（1）argc为2时，命令格式为

myfind <pathname>

它除了实现程序4－7功能外，还要统计出，在常规文件中，文件长度不大于4096字节的常规文件，在所有允许访问的普通文件中所占的百分比。程序也不允许打印出任何路径名。这个功能实现比较简单，只要略加修改myfunc()和main()这两个函数就可以了。

（2）argc为4且argv[2] == “-comp”时，命令格式为

myfind <pathname> -comp <filename>

它的功能是，输出在<pathname>目录子树之下，所有与<filename>文件内容一致的文件的绝对路径名。不允许输出任何其它的路径名，包括不可访问的路径名。为提高程序效率，在比较文件是否相同时，可先比较两个文件的大小，如果大小不同，则内容肯定不同，这样就免去了读文件所浪费的时间；如果大小相同，则再通过读文件进行比较。此时应当注意输入缓冲区不必开的太大，你可以从实验二得到启发。由于要求输出符合要求的文件的绝对路径名，因此当参数pathname不是绝对路径时，要调用getcwd()等函数来取得文件的绝对路径名。

1. argc 大于等于4且argv[2] == “-name”时，命令格式为

myfind <pathname> -name <str>…

<str>…是一个以空格分隔的文件名序列(不带目录)。命令输出在<pathname>目录子树之下，所有与<str>…序列中文件名相同的文件的绝对路径名。不允许输出不可访问的或无关的路径名。实现方法可以通过循环，把当前遍历的文件名和<str>这个序列中的文件名进行比较，如果和序列中的一个文件名相同，就符合条件，此时输出符合条件的文件的绝对路径名。

3．在实现上述三种功能时，主要是通过实现三个不同的myfunc()来完成的。第一个功能只须对myfunc()略加修改即可。而第二个和第三个功能，则需要分别定义另外两个函数，不妨分别称做myfunc2()和myfunc3()，去完成指定的功能。至于dopath()函数则基本不必修改，因为它只是按深度优先完成对目录树的遍历，这是一种“标准”操作。而遍历时对节点（即文件或目录）的操作（所谓的“访问”）则是由具体参数func()决定的。不过，如果你觉得程序输出的绝对路径名因为含有“\\”而不够美观，则也可以对dopath()函数略做修改来纠正这个缺欠。