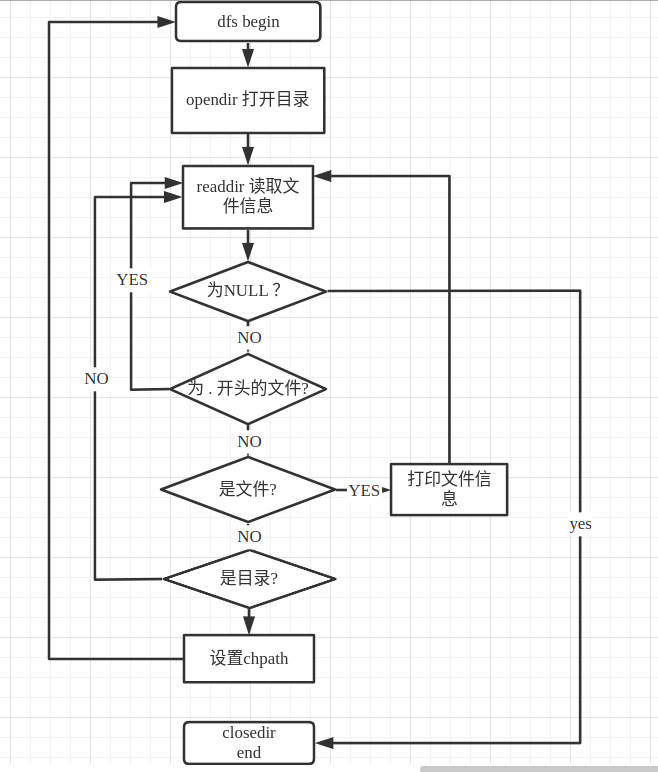
**实验报告要求**

1. 实验名称：文件系统扩展实验
2. 实验要求：对【程序5\_9】进行扩展，要求参数为目录名，且其下至少有三层目录，分别用深度遍历及广度遍历两种方法对此目录进行遍历，输出此目录下所有文件的大小及修改时间。
3. 实验目的：（在实验过程中应用的操作系统理论知识）
4. 熟悉和应用深度优先和广度优先遍历算法
5. 熟练掌握linux 系统下 关于文件系统的API
6. 掌握有关数据结构和指针的应用
7. 掌握使用gdb 调试的方法
8. 掌握实际运用中堆栈的内存分配
9. 实验内容：（为完成实验，如何进行程序设计）

DFS:

整体思路: 从所给的目录进入,按照reddir打开指针的顺序开始逐个遍历，遇到文件就按照规定格式输出信息。遇到目录就对该目录进行DFS，之后返回上一级继续遍历。当当前目录没有文件时返回

流程图:



重点部分: 重点是DFS 对 退出循环和 回溯时间点的把握。

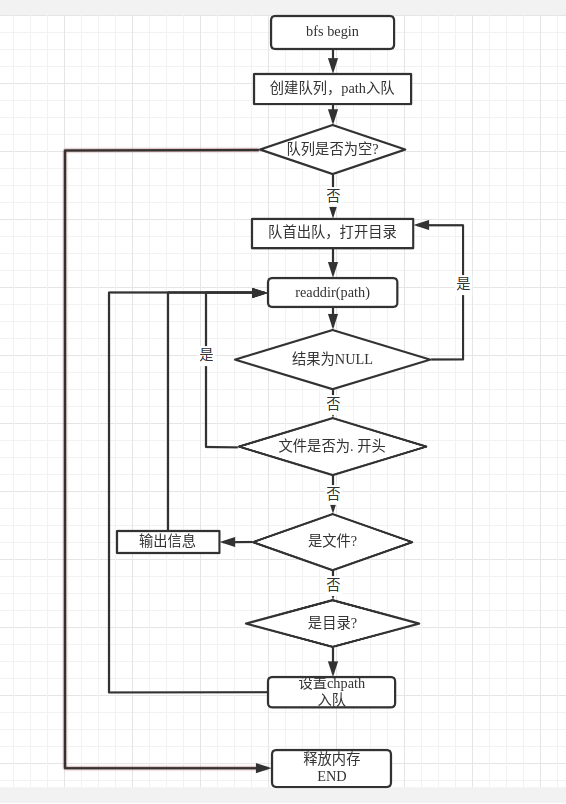
这里 退出循环的时间点是当前目录没有可读文件或者目录。

回溯时间点是 当读到的文件为目录时，进入下一层DFS

BFS:

整体思路： 利用队列，参考二叉树的层次遍历。先将根目录入队，循环出队列。进入目录后，当读取到文件时，输出信息；当读取到目录时，将目录入队，继续读取下一个文件；当当前目录的文件遍历完成时，退出此目录循环，从队列中弹出一个目录，再进入循环。当队列为空时，退出。

流程图：



重点部分: 1. 队列的建立，队列使用链表连接，存放字符串数据，要注意内存的分配和释放

1. 入队出队的时机；先将根目录入队，退出循环的条件为队列为空。入队的条件为当前文件为目录。出队的条件为一个目录遍历完毕。
2. 程序：（较之已有程序有修改的、新增的程序段加下划线）

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <dirent.h>

#include <time.h>

#include <string.h>

#include <ctype.h>

/\*辅助函数 输出文件的大小和最后修改时间\*/

static int get\_file\_size\_time(const char \*filename)

{

struct stat statbuf;

if(stat(filename,&statbuf)==-1) //取 filename 的状态

{

printf("Get stat on %s Error:%s\n",filename,strerror(errno));

return(-1);

}

if(S\_ISDIR(statbuf.st\_mode))

return(1); //判断是否是目录文件

if(S\_ISREG(statbuf.st\_mode)) //判断是否是普通文件

printf("\tsize:%ldbytes\tmodifiedate: %s\n",statbuf.st\_size,ctime(&statbuf.st\_mtime)); //输出文件的大小和最后修改时间

return(0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BFS 利用队列\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// 在strcpy 指针前要先 申请空间

struct node {

char\* data;

struct node\* next;

// struct node\* pre;

} \*head,\*tail,\*tmp; // 头节点，尾节点，临时节点

// 队列大小

int count;

// 创建队列

void create() {

head = NULL;

tail = NULL;

tmp = NULL;

}

//入队

void enque(char\* path)

{

// 如果是空队列

if (head == NULL) {

// 申请空间

head = (struct node\*) malloc(sizeof(struct node));

head->data = (char\*) malloc(strlen(path)+1);

strcpy(head->data,path);

head->next = NULL;

tail = head;

}

// 不是空队列

else {

tmp = (struct node\*) malloc(sizeof(struct node));

strcpy(tmp->data,path);

tail->next = tmp;

tail = tmp;

}

count++;

}

// 出队列

char\* peek()

{

char\* returnData ;

if (head == NULL) {

return NULL;

} else {

returnData = (char\*) malloc(strlen(head->data)+1);

strcpy(returnData,head->data);

if (head->next == NULL) {

free(head);

head=NULL;

tail=NULL;

} else {

struct node\* now = head;

head = head->next;

free(now);

}

}

count--;

return returnData;

}

// 返回队列大小

int size() {return count;}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DFS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void dfs(const char \*path)

{

DIR \*dir; // 定义目录指针

char chdpath[256]; // 子目录的全路径

if ( (dir=opendir(path)) == 0 ) return; // 打开目录

struct dirent \*stdinfo; // 用于存放从目录读取到的文件和目录信息

while (1)

{

if ((stdinfo=readdir(dir)) == 0) break; // 读取一记录 逐个读取，读到目录最后退出

if (strncmp(stdinfo->d\_name,".",1)==0) continue; // 以.开始的文件不读 没有这句会死循环

if (stdinfo->d\_type==8) // 如果是文件，显示出来

{

printf("%s/%s",path,stdinfo->d\_name);

char\* file = (char\*)malloc(256);

sprintf(file,"%s/%s",path,stdinfo->d\_name); // 重新设置路径

get\_file\_size\_time(file);

}

if (stdinfo->d\_type==4) // 如果是目录，再调用一次dfs

{

sprintf(chdpath,"%s/%s",path,stdinfo->d\_name); // 重新设置路径

dfs(chdpath);

}

}

closedir(dir); // 关闭目录指针

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*BFS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void bfs(char\* path) {

DIR \*dir; // 定义目录指针

char chdpath[256]; // 子目录的全路径

struct dirent \*stdinfo; // 用于存放从目录读取到的文件和目录信息

create();// 创建队列

char\* now=NULL;

enque(path); // 根目录入队

now = (char\*)malloc(200); // 给个额定大小

while(size()) {

strcpy(now,peek());//出队放到 now

// printf("现在的now:%s\n",now);

// 显示错误

errno = 0;

if ( (dir=opendir(now)) == 0 ) // 打开目录

{

printf("错误信息：%s %s\n",now,strerror(errno));

return;

}

// printf("现在的PATH:%s\n",now);

while(1) {

if ((stdinfo=readdir(dir)) == 0) break; // 读取一记录 逐个读取，读到目录最后退出

if (strncmp(stdinfo->d\_name,".",1)==0) continue; // 以.开始的文件不读 没有这句会死循环

if (stdinfo->d\_type==8) // 如果是文件，显示出来

{

printf("%s/%s",now,stdinfo->d\_name);

char\* file = (char\*)malloc(256);

sprintf(file,"%s/%s",now,stdinfo->d\_name); // 重新设置路径

get\_file\_size\_time(file);

}

if (stdinfo->d\_type==4) // 如果是目录，入队

{

sprintf(chdpath,"%s/%s",now,stdinfo->d\_name); // 重新设置路径

enque(chdpath);

}

}

closedir(dir);

}

free(now);

}

int main(int argc,char \*\*argv)

{

DIR \*dirp;

struct dirent \*direntp;

int stats;

char path[1024];

if(argc!=3) //判断输入是否为两个参数

{

printf("Usage:%s filename way(bfs or dfs)\n",argv[0]);

exit(1);

}

// DFS

if (!strcmp(argv[2],"dfs"))

dfs(argv[1]);

// BFS

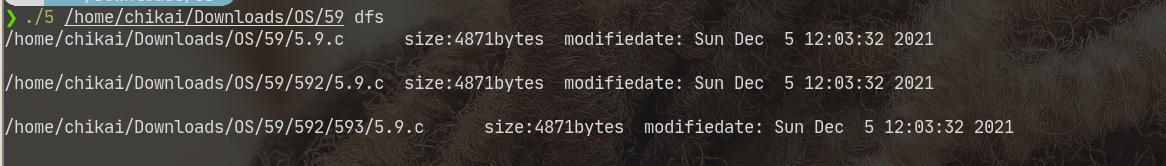
else

bfs(argv[1]);

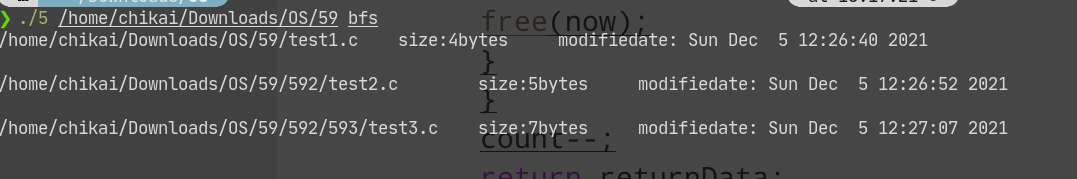
return 0;

}

1. 运行结果：（进行反白处理后截图）

DFS:

BFS

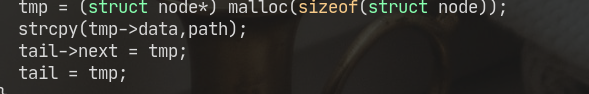


1. 实验总结：编程、调试过程中遇到的问题及解决办法。

BFS:

1. 使用队列数据结构遇到的问题:

（1)段错误问题：在建立链表队列时，使用指针连接前后节点，同时节点保存的数据为 字符串。 入队操作: 当将一个字符串入队时进行的操作:



出现段错误。

原因是使用strcpy 时没有为char\* 类型的tmp->data 分配内存。

只是段错误的一个例子，在完成实验中，遇到多次段错误，但是原因都是没有分配内存。

结论: 在使用指针和char\* 类型操作时，要格外注意内存的分配，每当要给指针所指向的地址赋值时，都要提前分配足够的内存。

(2)内存泄漏问题:

运行程序结束后，内存占用上升，原因是在malloc申请堆空间后,没有及时free().

(3)堆栈内存分配问题



开始没有malloc 申请内存，直接用数组存放拿出的字符串，结果得到段错误。

原因: 在栈空间申请的内存，在函数结束后会释放，返回就会得到段错误。所以要改为malloc 申请堆空间，堆空间不会立即释放，而是会在程序结束后释放。

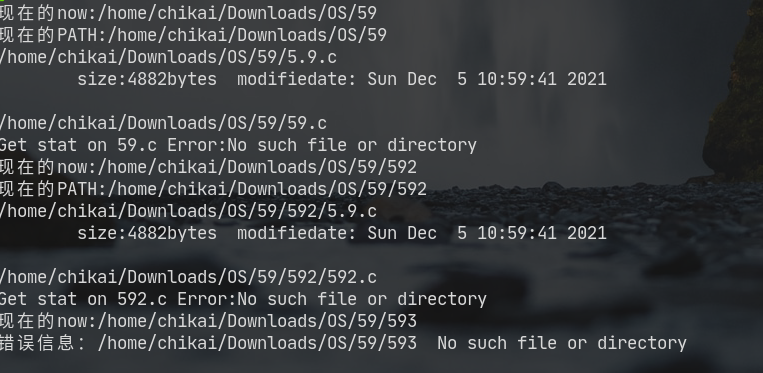
1. 算法逻辑的问题:

入队出队问题:

bfs关键的是入队和出队的时机,要先将一个节点入队，才能进入循环，然后每当一个节点目录完毕要及时关闭。

1. 未解决的问题和debug 策略

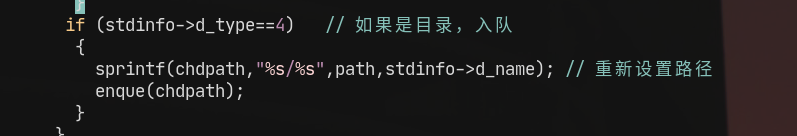
使用bfs时，不能遍历最后一层目录。



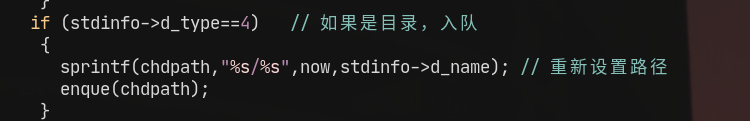
刚开始不知道在哪里出问题，通过printf打断点的方式查出来在 opendir() 处退出，又通过strerror得到具体的错误问题是 No such file or directory。但是这个目录是存在的，于是换一个目录验证，发现只能识别两层目录，不能打开最后一层。使用gdb单步调试，没有异常。

解决: 发现最后进入队列的目录没有修改

修改前:



修改后:



DFS：

算法逻辑问题

1. 关于 . 文件的问题

linux 中 以. 开头的 文件通常为隐藏文件和特殊文件，比如 . 和 .. 文件，一个代表本目录，一个代表上一层目录。当没有对. 文件进行判断时，会重复进入一个目录，形成死循环。

解决方法:



1. 重新设置路径问题

在进行下一层循环时，要重新设置路径，这里采用 将path （本层目录）和stdinfo->d\_name(下层目录) 结合的方法



1. 退出时机问题

当本层目录没有可读取的文件时，退出循环。

readdir函数读取下一个文件的信息，如果没有就返回0。加一条判断:

