操作系统实验

文件系统扩展实验

姓名：	李毓琪
学号：	2220191195
专业班级：	网络工程 2019-2
选课序号：	4

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：

   (1) 掌握通过 stat 系统调用从程序中访问和显示文件元数据的方法。

   (2) 掌握通过 readdir 系统调用读取目录文件信息的方法。

   (3) 掌握 Open,Close 等对文件夹的基本操作

   (4) 掌握 Linux 的目录结构。

   (5) 掌握广度遍历以及深度遍历算法。

   (6) 掌握 C 语言字符串操作。
4. 实验内容：（为完成实验，如何进行程序设计）

深度遍历目录：可使用堆栈或者递归对目标目录进行遍历，遍历文件时通过get_file_size_time函数判断该文件类型，如果为普通文件时，通过 dirp 指针获取文件名以及文件的最后修改时间；当遍历到文件夹时，将其文件夹名放入堆栈或者暂存起来，等到当前目录遍历完，在从堆栈中取出文件夹名对其进行遍历或者再对获取到的文件夹进行递归操作进行遍历。

广度遍历目录：与深度遍历目录类似。使用队列来存储遍历到的文件名，当遍历到文件夹时，将其文件夹名放入队列中。当当前目录遍历结束之后，从队列中取出一个文件夹，在对其进行遍历，以此实现

广度遍历目录。在 Linux 操作系统中，可以使用系统自带的消息队列机制即 sys/msg.h 库，省去创建队列以及对其操作的函数编写。

5. 程序：（较之已有程序有修改的、新增的程序段加下划线）

深度遍历：

```
#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>

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("%-20s			size:%-5ldbytes\t\tmodifiedat%-
20s",filename,statbuf.st_size,ctime(&statbuf.st_mtime));
	return(0);
}
static void get_file_from_dir(const char *dirname){
	DIR *dirp;
	struct dirent *direntp;
	char dirs[100][1024];
	char path[1024];
	int i = 0;
	if((dirp=opendir(dirname))==NULL)
	{
		printf("Open Directory %s Error:%s\n",dirname,strerror(errno));
	}
	while((direntp = readdir(dirp)) != NULL) {
		memset(path, 0, 1024);
		strcpy(path,dirname);
		strcat(path,"/");
```

```
          strcat(path,direntp -> d_name);

          if(get_file_size_time(path)==1){

         if(strcmp(direntp -> d_name,".") == 0 ||

                         strcmp(direntp -> d_name, "..") == 0){

                          continue;

                     }

               strcpy(dirs[++i],path);

          }

     }

     for(;i > 0;){

          get_file_from_dir(dirs[i--]);

     }

     closedir(dirp);

}

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

     int stats;

     if(argc!=2)

     {

          printf("Usage:%s filename\n\a",argv[0]);

          exit(1);

     }

     if((stats=get_file_size_time(argv[1])) != 1)
```

```
        {
                exit(1);
        }
        get_file_from_dir(argv[1]);
}
```

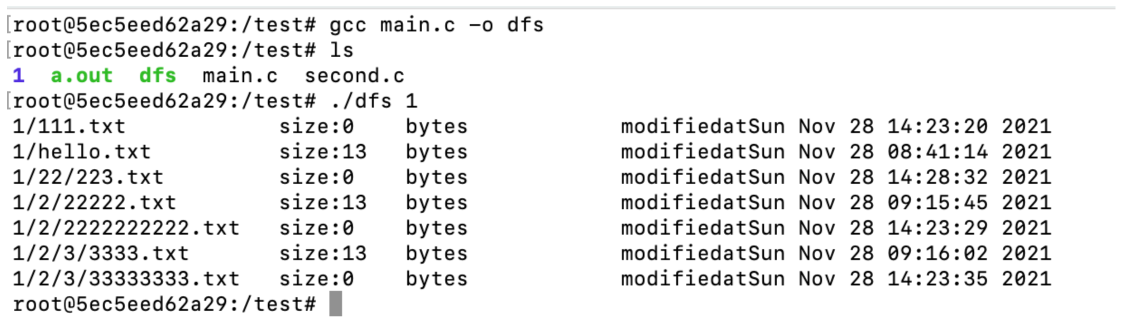
广度遍历：

```
#include <unistd.h>
#include <stdio.h>
#include <sys/msg.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>
int msgid;
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("%-20s                    size:%-5ldbytes\t\tmodifiedat%-
20s",filename,statbuf.st_size,ctime(&statbuf.st_mtime)); //输出文件的
大小和最后修改时间
        return(0);
}
static void get_file_from_dir(){
        DIR *dirp;
        struct dirent *direntp;
        char path[1024];
        char pathtmp[1024];
      while(msgrcv(msgid,&path,sizeof(path),0,IPC_NOWAIT) != -1){
                if((dirp=opendir(path))==NULL) //打开目录，将打开的目录信
息放至 dirp 中，若为空，则打开失败
                {
                        printf("Open                Directory                %s
Error:%s\n",path,strerror(errno));
            }
            while((direntp = readdir(dirp)) != NULL) {
                  strcpy(pathtmp,path);
                  strcat(pathtmp,"/");
                  strcat(pathtmp,direntp -> d_name);
                 if(get_file_size_time(pathtmp)==1){
                        if(strcmp(direntp -> d_name,".") == 0 ||
                        strcmp(direntp -> d_name, "..") == 0){
                        continue;
                  }
               msgsnd(msgid,pathtmp,sizeof(pathtmp),IPC_NOWAIT);
                  }
            }
            closedir(dirp);
      }
}

int main(int argc,char **argv) {
      int stats;
      if(argc!=2)
      {
        printf("Usage:%s filename\n\a",argv[0]);
        exit(1);
    }
    if((stats=get_file_size_time(argv[1])) != 1)
    {
        exit(1);
    }
    if((msgid = msgget(IPC_PRIVATE,0666))==-1){
        printf("mesget error!");
        exit(1);
    }
    msgsnd(msgid,argv[1],sizeof(argv[1]),IPC_NOWAIT);
    get_file_from_dir();
}
```

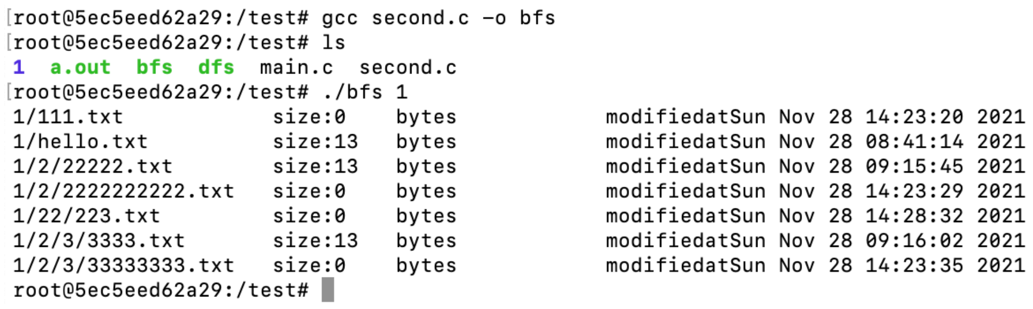
6. 运行结果：（进行反白处理后截图）

深度遍历目录：



```
root@5ec5eed62a29:/test# gcc main.c -o dfs
root@5ec5eed62a29:/test# ls
1  a.out  dfs  main.c  second.c
root@5ec5eed62a29:/test# ./dfs 1
1/111.txt           size:0    bytes           modifiedatSun Nov 28 14:23:20 2021
1/hello.txt         size:13   bytes           modifiedatSun Nov 28 08:41:14 2021
1/22/223.txt        size:0    bytes           modifiedatSun Nov 28 14:28:32 2021
1/2/22222.txt       size:13   bytes           modifiedatSun Nov 28 09:15:45 2021
1/2/2222222222.txt  size:0    bytes           modifiedatSun Nov 28 14:23:29 2021
1/2/3/3333.txt      size:13   bytes           modifiedatSun Nov 28 09:16:02 2021
1/2/3/33333333.txt  size:0    bytes           modifiedatSun Nov 28 14:23:35 2021
root@5ec5eed62a29:/test# 
```

广度遍历目录：



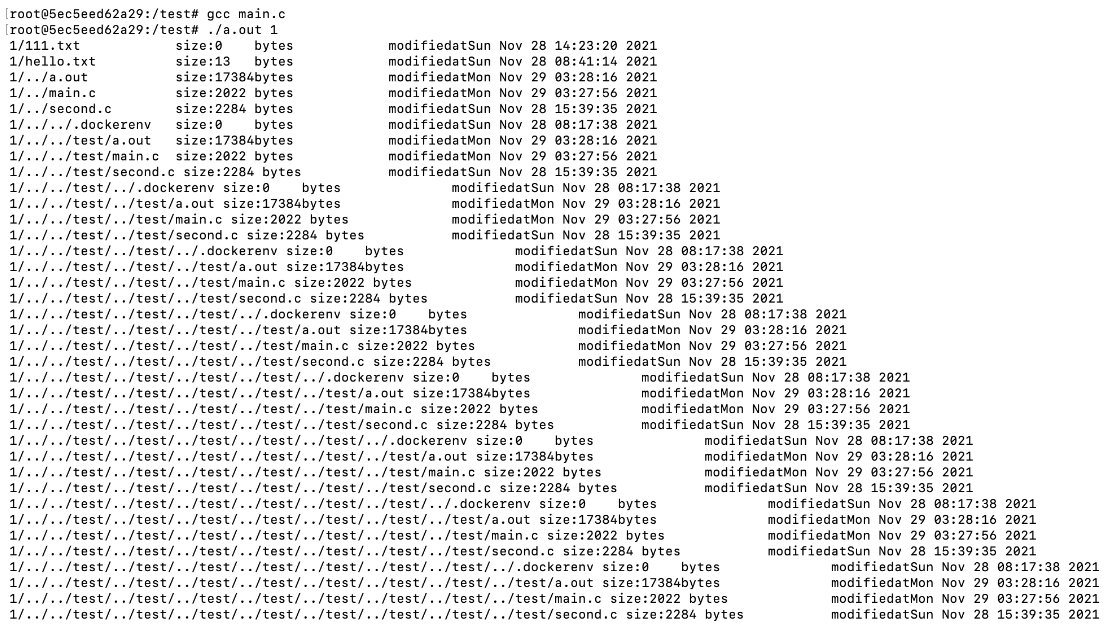
```
[root@5ec5eed62a29:/test# gcc second.c -o bfs
[root@5ec5eed62a29:/test# ls
1  a.out  bfs  dfs  main.c  second.c
[root@5ec5eed62a29:/test# ./bfs 1
1/111.txt            size:0    bytes           modifiedatSun Nov 28 14:23:20 2021
1/hello.txt          size:13   bytes           modifiedatSun Nov 28 08:41:14 2021
1/2/22222.txt        size:13   bytes           modifiedatSun Nov 28 09:15:45 2021
1/2/2222222222.txt   size:0    bytes           modifiedatSun Nov 28 14:23:29 2021
1/22/223.txt         size:0    bytes           modifiedatSun Nov 28 14:28:32 2021
1/2/3/3333.txt       size:13   bytes           modifiedatSun Nov 28 09:16:02 2021
1/2/3/33333333.txt   size:0    bytes           modifiedatSun Nov 28 14:23:35 2021
root@5ec5eed62a29:/test# 
```

7. 实验总结：编程、调试过程中遇到的问题及解决办法。（结合自己的实际情况如实说明，不说空话、套话！）

(1).在 MacOS 环境下编译时出现 Segment default :11

解决办法：MacOS 环境对内存分配较为严格，进行非对其访问时会出现 Segment fault 11 报错，Ubuntu 环境下未出现此问题。

(2).进行遍历时出现死循环的现象：



```
[root@5ec5eed62a29:/test# gcc main.c
[root@5ec5eed62a29:/test# ./a.out 1
1/111.txt            size:0    bytes           modifiedatSun Nov 28 14:23:20 2021
1/hello.txt          size:13   bytes           modifiedatSun Nov 28 08:41:14 2021
1/../a.out           size:17384bytes           modifiedatMon Nov 29 03:28:16 2021
1/../main.c          size:2022 bytes           modifiedatMon Nov 29 03:27:56 2021
1/../second.c        size:2284 bytes           modifiedatSun Nov 28 15:39:35 2021
1/../../.dockerenv   size:0    bytes           modifiedatSun Nov 28 08:17:38 2021
1/../../test/a.out   size:17384bytes           modifiedatMon Nov 29 03:28:16 2021
1/../../test/main.c  size:2022 bytes           modifiedatMon Nov 29 03:27:56 2021
1/../../test/second.c size:2284 bytes          modifiedatSun Nov 28 15:39:35 2021
1/../../test/../.dockerenv size:0    bytes             modifiedatSun Nov 28 08:17:38 2021
1/../../test/../test/a.out size:17384bytes             modifiedatMon Nov 29 03:28:16 2021
1/../../test/../test/main.c size:2022 bytes            modifiedatMon Nov 29 03:27:56 2021
1/../../test/../test/second.c size:2284 bytes          modifiedatSun Nov 28 15:39:35 2021
1/../../test/../test/../.dockerenv size:0    bytes             modifiedatSun Nov 28 08:17:38 2021
1/../../test/../test/../test/a.out size:17384bytes             modifiedatMon Nov 29 03:28:16 2021
1/../../test/../test/../test/main.c size:2022 bytes            modifiedatMon Nov 29 03:27:56 2021
1/../../test/../test/../test/second.c size:2284 bytes          modifiedatSun Nov 28 15:39:35 2021
1/../../test/../test/../test/../.dockerenv size:0    bytes             modifiedatSun Nov 28 08:17:38 2021
1/../../test/../test/../test/../test/a.out size:17384bytes             modifiedatMon Nov 29 03:28:16 2021
1/../../test/../test/../test/../test/main.c size:2022 bytes            modifiedatMon Nov 29 03:27:56 2021
1/../../test/../test/../test/../test/second.c size:2284 bytes          modifiedatSun Nov 28 15:39:35 2021
1/../../test/../test/../test/../test/../.dockerenv size:0    bytes             modifiedatSun Nov 28 08:17:38 2021
1/../../test/../test/../test/../test/../test/a.out size:17384bytes             modifiedatMon Nov 29 03:28:16 2021
1/../../test/../test/../test/../test/../test/main.c size:2022 bytes            modifiedatMon Nov 29 03:27:56 2021
1/../../test/../test/../test/../test/../test/second.c size:2284 bytes          modifiedatSun Nov 28 15:39:35 2021
1/../../test/../test/../test/../test/../test/../.dockerenv size:0    bytes             modifiedatSun Nov 28 08:17:38 2021
1/../../test/../test/../test/../test/../test/../test/a.out size:17384bytes             modifiedatMon Nov 29 03:28:16 2021
1/../../test/../test/../test/../test/../test/../test/main.c size:2022 bytes            modifiedatMon Nov 29 03:27:56 2021
1/../../test/../test/../test/../test/../test/../test/second.c size:2284 bytes          modifiedatSun Nov 28 15:39:35 2021
1/../../test/../test/../test/../test/../test/../test/../.dockerenv size:0    bytes             modifiedatSun Nov 28 08:17:38 2021
1/../../test/../test/../test/../test/../test/../test/../test/a.out size:17384bytes             modifiedatMon Nov 29 03:28:16 2021
1/../../test/../test/../test/../test/../test/../test/../test/main.c size:2022 bytes            modifiedatMon Nov 29 03:27:56 2021
1/../../test/../test/../test/../test/../test/../test/../test/second.c size:2284 bytes          modifiedatSun Nov 28 15:39:35 2021
```

解决办法：由于在遍历文件夹的时候同时遍历文件夹里的 “.” 目录以及 “ .. ” 目录，” . “为当前目录，” .. ”为上一级目录。当遍历到这两个文件夹时会将当前目录或者上一级目录放入堆栈、队列

或者进行递归操作。这样会造成死循环的形成。在放入堆栈队列或者进行递归操作时，对文件夹名进行判断，当文件夹名为 ”.” 或为 “..” 时，则不对其进行操作，直接跳过进入下一次循环即可解决。