Experiment 2: File and Directory Experiment

信息与软件工程学院
School of information and software engineering

Experiment 2: File I/O Management

Aim:

Mastering how to manage the file I/O defined in the POSIX and ANSI C.

how to open, close, create file;

how to read and write file;

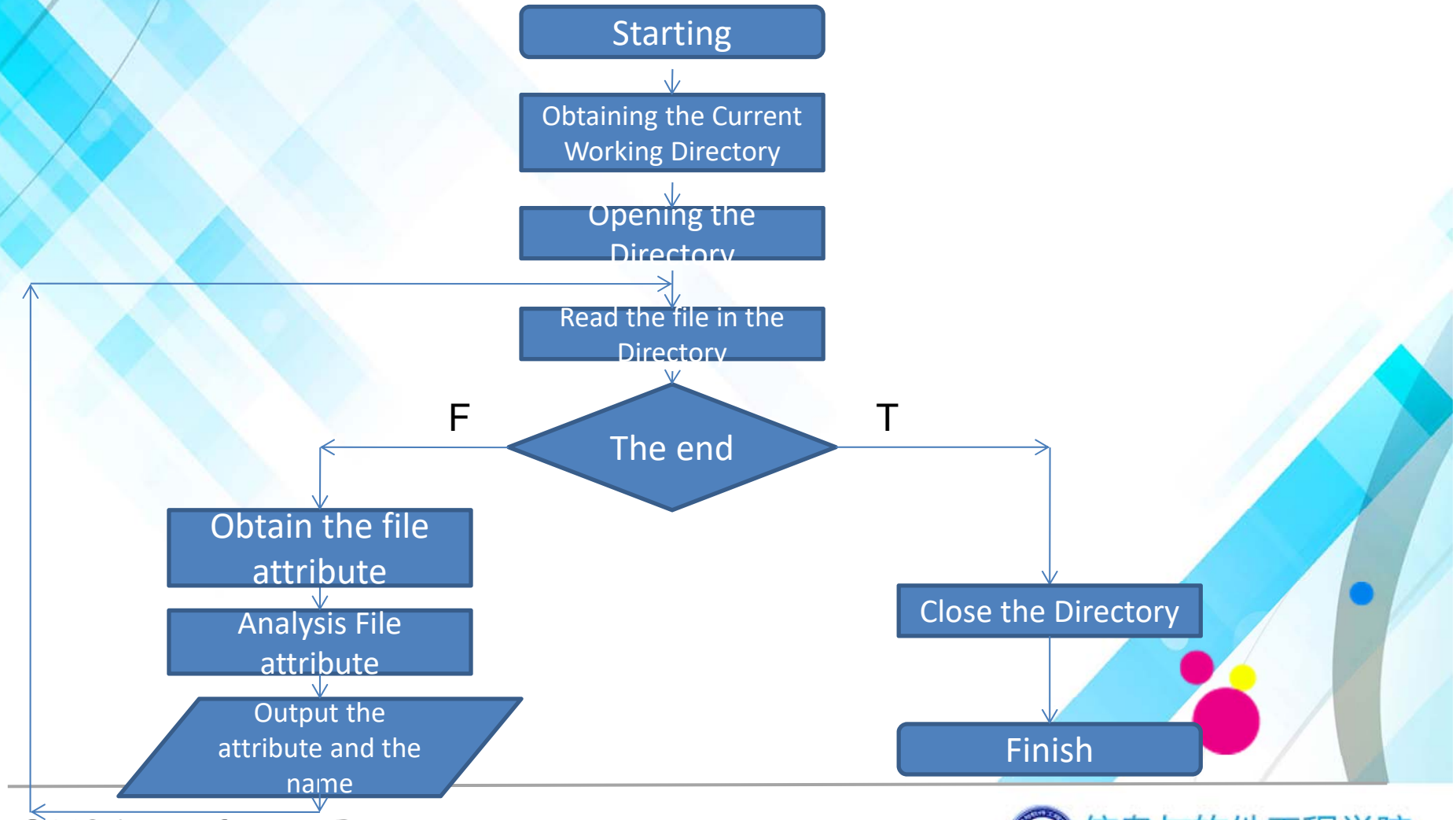
how to locate the file.

Experiment Content for Experiment 2

Content:

writing the program to realize the "ls -l" by employing the POSIX API and the macros for determining the file type.

The data flow of ls -l



Head Files

```
ao#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/stat.h>
#include <utime.h>
#include <time.h>
#include <pwd.h>
#include <grp.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <errno.h>
```

File Type

int print_type(mode_t st_mode)

We can determine the file type with the macros.

S_ISREG() regular file

S_ISDIR() directory file

S_ISCHR() character special file

S_ISBLK() block special file

S_ISFIFO() pipe or FIFO

S_ISLNK() symbolic link

S_ISSOCK() socket

File Permission

int print_perm(mode_t st_mode)

Mode_t st_mode;

unsigned integer

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					U	G	T	R	W	X	R	W	X	R	W	X

File type | Special file attribute | Owner permission | Group permission | Other user permission

```
1 $ls  -l
2 drwxr-xr-x  3  user  group  102  Mar11  22:56  Filename
```

File type, owner permission, group permission, other user permission, the number of hard link or the number of subdirectory, owner name, group name, size, Modification time, filename

File Permission

```
int print_perm(mode_t st_mode)
{
int i;
unsigned int mask = 0x7;
static char *perm[] = {"---","--x","-w-","-wx","r--","r-x","rw-","rwx"};
for(i=3;i>0;i--)
{
    printf("%3s",perm[(st_mode >> (i-1)*N_BITS) & mask]);
}
printf(" ");
return 0;
}
```

Variables Defined in the Main

```
char buf[500];
DIR *currentdir = NULL;
struct dirent *currentdp = NULL;
struct stat currentstat;
struct passwd *p_passwd;
struct group *p_group;
char *p_time;
int i;
```

Process in the Main

- Check the input Argument. If the number of argument don't equal to 2, output "ERROR: Invalid Argument\n"
- void *memset(void *s, int ch, size_t n); to initialize the *buf*.
- sprintf(buf,"%s",argv[1]); //put the second argument into the buf.
- Open the directory, which is input as the argument.

  currentdir = opendir(buf)
- Read the opened directory.

  currentdp = readdir(currentdir)

Process in the Main

```
struct dirent
{
long d_ino; /* inode number 索引节点号 */
off_t d_off; /* offset to this dirent 在目录文件中的偏移 */
unsigned short d_reclen; /* length of this d_name 文件名长 */
unsigned char d_type; /* the type of d_name 文件类型 */
char d_name [NAME_MAX+1]; /* file name (null-terminated) 文件
    名，最长256字符 */
}
```

Process in the Main

memset(buf,0,500);

sprintf(buf,"%s",argv[1]);

sprintf(buf,"%s/%s",buf,currentdp->d_name);

printf("the file is %s\n",buf);

Then, adopt the lstat function to obtain file attributes.

int lstat(const char *restrict *pathname*, struct stat *restrict *buf*);

lstat(buf,&currentstat)

Next, adopt the following function to obtain the specific attribute.

p_time = ctime(&currentstat.st_mtime);

p_passwd = getpwuid(currentstat.st_uid);

p_group = getgrgid(currentstat.st_gid);

Process in the Main

```
print_type(currentstat.st_mode);
print_perm(currentstat.st_mode);
printf("%5d ",currentstat.st_nlink);
if(p_passwd != NULL)
    printf("%s ",p_passwd->pw_name);
else
    printf("%d ",currentstat.st_uid);
if(p_group != NULL)
    printf("%s ",p_group->gr_name);
else
    printf("%d ",currentstat.st_gid);
printf("%7d ",(int)currentstat.st_size);
for(i=0; p_time[i] !=0 && p_time[i]!='\n'; i++)
    putchar(p_time[i]);
    printf(" ");
printf("%s\n", currentdp->d_name);
```

ls -l实现关键代码

```
59  if((currentdir = opendir(buf)) == NULL)
60  {
61    printf("open directory fail\n");
62    return 0;
63  }
64  while((currentdp = readdir(currentdir))!= NULL)
65  {
66    if(currentdp->d_name[0] != '.')
67     {
68        if(lstat(currentdp->d_name,&currentstat) == -1)
69        {
70            printf("get stat error\n");
71            continue;
72        }
73        print_type(currentstat.st_mode);
74        print_perm(currentstat.st_mode);
75        print_link(currentstat.st_nlink);
76        print_usrname(currentstat.st_uid);
77        print_grname(currentstat.st_gid);
78        print_time(currentstat.st_mtime);
79        print_filename(currentdp);
80     }
81  }
82  closedir(currentdir);
83  return 0;
```