Advanced Operting System

Q 1) write a program to check file type (Regualr file,Directory file,Socket,FIFO,Charachter special,symbolic) using stat() system call

```
#include<stdio.h>
#include<sys/stat.h>
	void main(int argc, char *argv[]){
		struct stat sObj;
		for(int i=1;i<argc;i++){
			stat(argv[i],&sObj);
			if(S_ISREG(sObj.st_mode))
				printf("File: %s is Regular file\n",argv[i]);
			else if(S_ISDIR(sObj.st_mode))
				printf("File: %s is Directory file\n",argv[i]);
			else if(S_ISBLK(sObj.st_mode))
				printf("File: %s is Block file\n",argv[i]);
			else if(S_ISCHR(sObj.st_mode))
				printf("File: %s is Character Block file\n",argv[i]);
			else
				printf("Other file Type\n");
			}
}
```

OUTPUT

```
cc FileType.c
./a.out notes.txt
File: notes.txt is Regular file
```

Q 2) Write a program to find properties such as inode,file size,modification & acess time,file permission*

```
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<time.h>
int main(){
        struct stat sObj;
        struct stat *sObj1;
        struct timespec tObj;
        stat("myFile.txt",&sObj);
        stat("myFile.txt",sObj1);
        printf("Inode: %ld\n",sObj.st_ino);
        printf("Hard link: %ld\n",sObj.st_nlink);
        printf("file size: %ld\n",sObj.st_size);
        printf("Aceess Time: %ld\n",sObj.st_atime);
        printf("Modification Time: %ld\n",sObj.st_mtime);

printf( (sObj.st_mode & S_IRUSR) ? "r" : "-");
printf( (sObj.st_mode & S_IWUSR) ? "w" : "-");
printf( (sObj.st_mode & S_IXUSR) ? "x" : "-");

printf( (sObj.st_mode & S_IRGRP) ? "r" : "-");
printf( (sObj.st_mode & S_IWGRP) ? "w" : "-");
printf( (sObj.st_mode & S_IXGRP) ? "x" : "-");

printf( (sObj.st_mode & S_IROTH) ? "r" : "-");
printf( (sObj.st_mode & S_IWOTH) ? "w" : "-");
```

```
printf( (sObj.st_mode & S_IXOTH) ? "x" : "-");


//printf("Created on: %d-%d-%d %d:%d:%d", dt.tm_mday, dt.tm_mon, dt.tm_year 1900,
dt.tm_hour, dt.t

m_min, dt.tm_sec);

return 0;

}
```

OUTPUT

```
$cc StatFun.c

$ ./a.out

Inode: 1445773

Hard link: 1

file size: 0

Aceess Time: 1651752752

Modification Time: 1651517803
```

Q 3) Write a C program to find whether a given file is present in current directory or not.

```
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
int isFileExists_fopen(const char *path)
{
	// Check for file existence
	FILE *fd=fopen(path,"r");
	if (fd==NULL)
		return 0;
	return 1;
}

void main(int argc,char *argv[]){
	for(int i=1;i<argc;i++){
		int ret=isFileExists_fopen(argv[i]);
		if(ret==1)
			printf("File: %s is present\n",argv[i]);
		else
			printf("File: %s is Not present\n",argv[i]);
	}
}
```

OUTPUT

```
$cc FileExist_Fopen.c
$ ./a.out notes.txt
File: notes.txt is present
$ ./a.out NoData.txt
File: NoData.txt is Not present
```

Q 4) Write a C program to find file properties such as inode number, number of hard link, File permissions, File size, File access and modification time and so on of a given file using fstat() system call.

```
#include<stdio.h>
//#include<sys/types.h>
#include<fcntl.h>
#include<sys/stat.h>
void fstatFun(char *Fname){
        struct stat sObj;
        printf("File : %s\n",Fname);
        //int fd=open("MyFile.txt");
        int fd=open(Fname,'r');
        printf("Fd: %d\n",fd);
        fstat(fd,&sObj);
        printf("Inode: %ld\n",sObj.st_ino);
        printf("size: %ld\n",sObj.st_size);
        printf("Hard link: %ld\n",sObj.st_nlink);

        printf("Aceess Time: %ld\n",sObj.st_atime);
        printf("Modification Time: %ld\n",sObj.st_mtime);
        // printf("Created on: %d-%d-%d %d:%d:%d", dt.tm_mday,dt.tm_mon,dt.tm_year
1900, dt.tm_hour, dt.tm_min, dt.tm_sec);
        }
        int main(int argc,char **name){
                fstatFun(name[1]);
}
```

OUTPUT

```
cc fstatFun.c
```

$ ./a.out MyFile.txt

File : MyFile.txt

Fd: 3

Inode: 1445777

size: 13

Hard link: 1

Aceess Time: 1651750729

Modification Time: 1651750722

$ ./a.out demo.txt

File : demo.txt

Fd: 3

Inode: 1445811

size: 5

Hard link: 1

Aceess Time: 1651752752

Modification Time: 1651751504

Q 5) Takes multiple files as Command Line Arguments and print their inode number.

```
#include<stdio.h>
//#include<sys/types.h>
#include<fcntl.h>
#include<sys/stat.h>
void fstatFun(char *Fname){
        struct stat sObj;
        printf("File : %s\n",Fname);
        int fd=open(Fname,'r');
        fstat(fd,&sObj);
        printf("Inode: %ld\n",sObj.st_ino);
        }
        int main(int argc,char **name){
                for(int i= 1; i<argc; i++)
                        fstatFun(name[i]);
}
```

OUTPUT

```
cc multipeFIle_Inode.c
$ ./a.out demo.txt MyFile.txt fstatFun.c

File : demo.txt
Inode: 1445811

File : MyFile.txt
Inode: 1445777

File : fstatFun.c
Inode: 1444298
```

Q 6) To demonstrate the use of atexit() function.

```
#include<stdio.h>
#include<stdlib.h>
        void functionA(){
        printf("In user defined function A\n");
        }
        void functionB(){
        printf("In user defined function B\n");
        }
        int main(){
        atexit(functionA);
        functionB();
        printf("In Main function\n");
}
```

OUTPUT

```
cc atExist.c
$ ./a.out
In user defined function B
In Main function
In user defined function A
```

Q 7) Open a file goes to sleep for 15 seconds before terminating.

```
#include<stdio.h>
#include<unistd.h>
int main()
{
        printf("Sleep for 15 second to exit.\n");
        sleep(15);
        return 0;
}
```

OUTPUT

```
./a.out
Sleep for 15 second to exit
```

Q 8) To create a file with hole in it.

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <fcntl.h>
int main(int argc, const char *argv[])
{
	char random_garbage[8192]; /* Don't even bother to initialize */
	int fd = -1;
	if (argc < 2) {
		fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
		return 1;
	}
	fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0666);
	if (fd < 0) {
		perror("Can't open file: ");
		return 2;
	}
	write(fd, random_garbage, 8192);
	lseek(fd, 5 * 4096, SEEK_CUR);
	write(fd, random_garbage, 8192);
	close(fd);
	return 0;
}
```

OUTPUT

```
cc holeFile.c
./a.out demo.txt
```

(create hole file with name demo.txt)

Q 9) //To handle the two-way communication between parent and child using pipe.#include <stdio.h>

```
#include<stdlib.h>
#include<unistd.h>
int main()
{
	pid_t pid;
	int r;
	char *ch=NULL;
	char *ch1=NULL;
	int readpipe[2];
	int writepipe[2];
	int a;
	int b;
	a=pipe(readpipe);
	b=pipe(writepipe);
		// check a and b
	pid=fork();
		// check pid
	if(pid==0)
	{ //CHILD PROCESS
		close(readpipe[1]);
		close(writepipe[1]);
		read(readpipe[0],ch,sizeof(ch));
		printf("\nREAD = %s",ch);
		close(readpipe[0]);
		ch1="YES";
		write(writepipe[1],ch1,sizeof(ch1));
		close(writepipe[1]);
```

```
        }
```

OUTPUT

```
./a.out demo.txt chown.c


ACK RECEIVED (null)
READ = (null)
```

Q 10) Write a C program to demonstrates the different behavior that can be seen with automatic, global, register, static and volatile variables (Use setjmp() and longjmp() system call).

```
#include<stdio.h>
#include<setjmp.h>
jmp_buf buf;

int g=0;

void fun(){

longjmp(buf,1);
}

void main(){

static int s=0;
auto int a=0; // auto is nothing but local variable. we cant declare auto var as a static variable. scope of auto variab
le is within a block.
register int r =0;
volatile int v = 0;

printf("Before setjump all assigned to 0----------\n");
printf("static %d\n",++s);
printf("auto %d\n",++a);
printf("global %d\n",++g);
printf("register %d\n",++r);
printf("volatile %d\n\n\n",++v);
```

```
if(setjmp(buf)){

printf("After setjump all increased by 1 ----------\n");
printf("static %d\n",++s);
printf("auto %d\n",++a);
printf("global %d\n",++g);
printf("register %d\n",++r);
printf("volatile %d\n",++v);

} else {
fun();
}
}
```

OUTPUT

```
cc setlongjmp.c
$ ./a.out
Before setjump all assigned to 0----------
static 1
auto 1
global 1
register 1
volatile 1

After setjump all increased by 1 ----------
static 2
```

auto 2

global 2

register 2

volatile 2