2019/20

Student Name: Fahmin Guliyev

Student ID: 21724195

Project Title: System Programming Homework

Supervisor: Konul Aliyeva, Nicolas Magaud

DEPARTMENT OF COMPUTER SCIENCE

1 Problem 1

Statement of the Exercise: Write a program which displays the type of the requested file (directory, ordinary file, . . .), as well as its permissions (read, write, execute, in the same way as the command ls does using option-l) and last modified date. The name of the file should be given as an argument in Linux shell.
As we see in the problem we are asked to define if the entry is directory or ordinary file? . For this case I used the foolowing code:

```
DIR *cand;
struct dirent *dir;
cand = opendir(argv[1]);
if(cand) //directory
else     // ordinary file
```

* Then we are asked to print the permissions

```
void displayPermissions(char * path){
    struct stat _stat; // ystem struct which is used for systeem calls
    if(stat(path,&_stat)==0){
        printf("File permission is:\t");
        printf( (S_ISDIR(_stat.st_mode)) ? "d" : "-"); //The current permission
            printf( (_stat.st_mode & S_IRUSR) ? "r" : "-");
            printf( (_stat.st_mode & S_IWUSR) ? "w" : "-");
            printf( (_stat.st_mode & S_IXUSR) ? "x" : "-");
            printf( (_stat.st_mode & S_IRGRP) ? "r" : "-");
            printf( (_stat.st_mode & S_IWGRP) ? "w" : "-");
            printf( (_stat.st_mode & S_IXGRP) ? "x" : "-");
            printf( (_stat.st_mode & S_IROTH) ? "r" : "-");
            printf( (_stat.st_mode & S_IWOTH) ? "w" : "-");
            printf( (_stat.st_mode & S_IXOTH) ? "x" : "-");
    }
    else fprintf( stderr, "File could not be read\n");

}
```

* Then we need to display the last modification date so

```
void displayLastModData(char* path){
    struct tm dateTime;
    time_t _time = time(NULL);
    localtime_r(&_time, &dateTime);
    long int offset=dateTime.tm_gmtoff;
    _stat.st_mtime+=offset;
    dateTime = *(gmtime(&_stat.st_mtime));
        printf("\tLast Modified Date:\t %02d/%02d/%d %d:%02d:%02d\n",  dateTime.tm_mday,
        dateTime.tm_mon+1, dateTime.tm_year + 1900,
       dateTime.tm_hour, dateTime.tm_min,dateTime.tm_sec);
}
```

Note: to perform 'ls -l' (long list) we will perform

```
for(int i = 1; (dir = readdir(cand)) != NULL; i++)
// This iteration will reach also the internal files of my entry(if it's directory)
```

For more details, please look at the screenshots and execute the codes

2 Problem 2

Write a program which takes as argument the name of a directory and then: 1. displays the system time, using seconds and micro-seconds (system call: gettimeofday) ; 2. runs (using one of the exec primitives) the ls command with the option -l on the directory provided as parameter; 3. display the time (as before) and the time the command ls took to run

To solve this exercise, I preferred to use the piping. Because we have a process and we need before and ending also duration(elapsed time)

Firstly we have to define following struct to perform timing actions:

```
struct timeval timeBeginning,timeEnding;
```

* For initial time i use:

```
printf("Beginning time:  %ld(seconds)\t%ld(mseconds)\n",timeBeginning.tv_sec,
    timeBeginning.tv_usec );
```

* For terminal time:

```
printf("\nEnding Time:  %ld(seconds)\t%ld(mseconds)\n",timeEnding.tv_sec,
        timeEnding.tv_usec );
```

* For elapsed time:

```
long int  taken = (timeEnding.tv_sec-timeBeginning.tv_sec)*1000000 +
    (timeEnding.tv_usec-timeBeginning.tv_usec);
```

For more details, please look at the screenshots and execute the codes

3 Problem 3

Write a program made of 2 processes: the parent reads data from the standard input and transmit them to its child through a pipe, the child then prints them on the standard output. The parent then waits for the termination of the child process.
void copy(int fdsrc, int fddst)
which copies the contains of a file whose descriptor is fdsrc to a file whose descriptor isfddst and then use this function in a simple (single-process) program to copy the standard input into the standard output. You may test your function with:
./a.out ¡ /bin/ls ¿ toto
cmp /bin/ls toto
If cmp does not display any error, then the copy went fine.

First of all we have to write the function :

```
void copy(int fdsrc,int fddst){
    char c;
    for(;read(fdsrc,&c,1)>0;){
        write(fddst,&c,1);
    }
}
```

Again to solve the problem I used the piping. The rest one is very very similar what we did in the class. However I attach here the source code

```
int tube[2];
    if (pipe(tube) != 0) {
        perror("Pipe Error");
        exit(1);
    }
    int pid = fork();
    if (pid == -1) {
        perror("Fork failed");
        exit(2);
    }
    else if (pid == 0) {
        close(tube[1]);
        copy(tube[0],1);
        exit(3);
    }
    else {
        close(tube[0]);
        copy(0,tube[1]);
        close(tube[1]);
    }
```

For more details, please look at the screenshots and execute the codes