开发过程

任务说明

用gcc makefile构建个工程，写一个ls工具(至少可以传3种参数)，需要了解编译到二进制文件全过程。并且用git管理开发过程并推送到远端仓库，开发过程要求markdown记录，并写使用说明（pdf）。建议开发工具（vim，git，markdown，makefile，gdb，others shell cmd）。

2022年9月21日

1. 认识Markdown，熟悉语法结构，开始使用Markdown记录开发过程。
2. 学习Linux系统编程：基本程序框架，文件IO及目录IO，尝试写一个最基本的ls命令。
3. 了解编译到二进制文件全过程：
    1. 预处理：.c源文件 -- > cpp预处理器 -- > .c文件
    2. 编译：.c文件 -- > gcc编译器 -- > .s文件
    3. 汇编：.s文件 -- > as 汇编器 -- > .o文件（又称目标文件）
    4. 链接：.o文件 -- > ld 链接器 -- > 二进制可执行目标程序

2022年9月22日

1. 总结归纳IO的用法（IO总结.md）。

2. 学习使用 gcc 命令行方式编译C程序：gcc -c #.c + gcc #.o -o "name" + 执行./"name"

3. 编写一个最基本的ls工具，相当于shell指令里面'ls'。

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>

int main(int argc,char * argv[])
{
    //步骤一：定义变量，+目录流指针
    DIR *dp;
    struct dirent *dir;

    if(argc!=2){  //报错提示：需要输入某目录才能打印
        printf("usage %s <name file>\n",argv[0]);
        return -1;
    }

    dp = opendir(argv[1]);
    if(dp == NULL){
        printf("opendir is error\n");
        return -2;
    }
```

```
    while(1){
        dir = readdir(dp);
        if(dir != NULL)
            printf("%s  ",dir->d_name);
        else{
            printf("\n");
            break;
        }
    }

    closedir(dp);
    return 0;
}
```

4. 编写一个单参数的ls工具：获取输入地址文件的信息，包括大小，设备号，模式。

```
//获取输入地址文件的信息，包括大小，设备号，模式
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>

int main(int argc, char *argv[])
{
    struct stat filestat;
    int message;

    //原型：int stat(const char *path, struct stat *buf);

    message = stat(argv[1],&filestat);
    if(message<0){
        printf("stat() failed\n");
        return -1;
    }
    else{
        printf( "the size of %s is %ld\n",argv[1],filestat.st_size );
        printf( "the dev of %s is %ld\n",argv[1],filestat.st_dev  );
        printf( "the mode of %s is %d\n",argv[1],filestat.st_mode );
    }

    return 0;
}
```

5. 将以上两个整合为一个ls工具，相当于简易版shell指令里面'ls -l'。

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
```

```c
#include <sys/types.h>
#include <dirent.h>
int main(int argc,char *argv[])
{
    //定义变量
    DIR *dp;
    struct dirent *dir;
    struct stat filestat;
    int message;

    //如果输入参数少于2个则提示错误
    if(argc!=2){
        printf("usage %s <name file>\n",argv[0]);
        return -1;
    }

    //获取目录流指针
    dp = opendir(argv[1]);
    if(dp == NULL){
        printf("opendir is error\n");
        return -2;
    }

    //打印文件的基本信息
    while(1){
        dir = readdir(dp);
        if(dir != NULL){

            message = stat(dir->d_name,&filestat);
            if(message<0){
                printf("stat() failed\n");
                 return -1;
            }
            else{ //输出ls -l简易版
                printf( "%-6ld",filestat.st_size ); //大小
                printf( "%-4lu",filestat.st_nlink  ); //链接数
                printf( "%-6d",filestat.st_uid );//所有者用户id
                printf( "%-6d",filestat.st_gid );//组别id
                printf( "%-12ld",filestat.st_mtime );//最后修改时间
                printf( "%s\n",dir->d_name);//文件名

            }
    }
        else{
            printf("\n");
            break;
        }
    }

    //关闭目录
    closedir(dp);
    return 0;
}
```

```
534   1   1000  1000  1663816873  date.md
4096  2   1000  1000  1663809083  pactice_OI1
4096  31  1000  1000  1663810340  ..
4096  4   1000  1000  1663814940  pactice_OI2
377   1   1000  1000  1663836790  timetest.c
692   1   1000  1000  1663816764  task1_1.c
667   1   1000  1000  1663828705  task1_2.c
4096  2   1000  1000  1663745278  .vscode
1494  1   1000  1000  1663837094  task1_3.c
4096  5   1000  1000  1663837098  .
9040  1   1000  1000  1663837098  a.out
```

2022年9月23日

1. 学习make工具。
2. 开始编写可以传入三个参数的ls工具（ls为二进制可执行文件），使用命令 "./ls ./ l" 可以打印出当前路径文件的一些信息，[l]为可选项。
3. 遇到问题：编写的工具只能打印当前目录的文件名和信息，其它路径无法打印。于是将工具设计改为：./ls + -a + -L + 排序
4. 编写了一个ls工具："./ls [-a] [-l]" 主要结构如下：↓

```
if(argv[1] == NULL){
        while(1){
            dir = readdir(dp);
            if(dir != NULL){
                message = stat(dir->d_name,&filestat);
                if(message<0){
                    printf("stat() failed because without argv[2] \n");
                    return -1;
                }
                else{
                    if(strncmp(dir->d_name, ".", 1) == 0) continue;
                    printf( "%s  ",dir->d_name);//非以.开头的文件名
                }
            }
            else{
                printf("\n");
                break;
            }
        }
    }
```

同时遇到警告问题：段错误 (核心已转储)，暂时不知如何解决。

2022年9月26日

1. 学习Makefile的语法及编写。
2. 完善程序，改进其中存在的问题。

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <dirent.h>
#include <string.h>

int main(int argc,char *argv[])
{
    //定义变量
    DIR *dp;
    struct dirent *dir;
    struct stat filestat;
    int message;

    //获取目录流指针
    dp = opendir(".");
    if(dp == NULL){
        //printf("opendir is error\n");
        return -2;
    }
    else{
        printf("opendir is right\n");
    }

//当只输入一个参数时，打印当前目录的文件名（不含隐藏文件），功能与shell中'ls'相同
    if(argc == 1){   //如果只输入了执行文件作为参数
        while(1){
            dir = readdir(dp);
            if(dir != NULL){
                message = stat(dir->d_name,&filestat);
                if(message<0){
                    printf("stat() failed because without argv[2] \n");
                    return -1;
                }
                else{
                    if(strncmp(dir->d_name, ".", 1) == 0) continue;
                    printf( "%s  ",dir->d_name);//非以.开头的文件名
                }
            }
            else{
                printf("\n");
                break;
            }
        }
    }

//当输入两个参数且第二个参数为'-a'时，打印当前目录的文件名（含隐藏文件），功能与shell
中'ls -a'相同
    if(argc == 2){//如果只有两个参数
        if (strcmp(argv[1], "-a" ) != 0){//如果第二个参数'-a'输入有误
            printf("please enter the correct format in the second parameter.\n");
```

```
        }
        else {//如果第二个参数'-a'输入正确
            while(1){
                dir = readdir(dp);
                if(dir != NULL){
                    message = stat(dir->d_name,&filestat);
                    if(message<0){
                        printf("stat() failed because without argv[2] \n");
                        return -1;
                    }
                    else{
                        printf( "%s  ",dir->d_name);//打印所有文件名
                    }
                }
                else{
                    printf("\n");
                    break;
                }
            }
        }
    }

//当输入三个参数，且第二个参数为'ls -a'，第三个参数为'-l'时，打印当前目录的文件信息，功能与shell中'ls -l'相同
    if( argc == 3){
        if (strcmp(argv[1], "-a" ) != 0){ //如果第二个参数'-a'输入有误
            printf("please enter the correct format in the second parameter.\n");
        }
        else if((strcmp(argv[1], "-a" ) == 0) && (strcmp(argv[2],"-l") != 0 )){
//如果第三个参数'-l'输入有误
            printf("please enter the correct format in the third parameter.\n");
        }
        else{//如果第二个参数'-a'输入正确，且第三个参数'-l'输入正确
            while(1){
                dir = readdir(dp);
                if(dir != NULL){
                    message = stat(dir->d_name,&filestat);
                    if(message<0){
                        printf("stat() failed because stat error\n");
                        return -1;
                    }
                    else{ //输出ls -l简易版
                        printf( "%-6ld",filestat.st_size ); //大小
                        printf( "%-4lu",filestat.st_nlink ); //链接数
                        printf( "%-6d",filestat.st_uid );//所有者用户id
                        printf( "%-6d",filestat.st_gid );//组别id
                        printf( "%-12ld",filestat.st_mtime );//最后修改时间
                        printf( "%s\n",dir->d_name);//文件名
                    }
                }
                else{
                    break;
                }
            }
```

```
        }
    }

    if( argc > 3){//如果多于三个参数
        printf("the parameters more than 3. \n");
    }

    closedir(dp); //关闭目录
    return 0;
}
```

3. 对于"段错误(核心已转储)"此类问题进行总结： ①该问题在linux系统编程中属于相当常见的问题，原因通常为：栈溢出和指针问题。 ②在此前的开发过程中，"段错误(核心已转储)"属于指针问题出错，对于:

   ```
   if(argv[1] == NULL)
   ```

   这里对于指针argv来说，"空指针在与指向任何对象或函数的指针作比较时保证不会相等。"则对于想要判断输入的参数是否为空，可以采用判断argc的值作为后续参数是否为空的依据，从而避免指向空指针。③对于"栈溢出"这类问题，就是越过定义的最大栈顶，也就是栈溢出会发生段错误的主要原因。

4. 尝试实现shell中 'ls -l' 的命令，将现有程序的'./ls -a -l'显示的信息更加全面。其中'ls -l'显示的信息有文件信息权限、目录数、所属用户、所属用户数、文件大小(以字节为单位)、mttime。

5. 在学习过程中，遇到了stat函数调用结构体的问题，获取文件状态的st_mode时。经过查询，得知：

   1. 15bit ~ 12bit 保存文件类型
   2. 11bit ~ 9bit 保存执行文件时设置的信息
   3. 8bit ~ 0bit 保存文件访问权限 使用st_mode时，返回一个十进制的数，需要对应转换才能返回对应的文件信息。使用时调整为输出二进制，方便查看对比。

6. 学习localtime转换st_time的时间单位。

7. 优化代码结构，小工具基本编写完成。

2022年9月27日

1. 编写使用说明pdf。
2. 了解git管理开发过程并推送到远端仓库。