私有协议文件管理服务器项目需求(百度网盘)

一期功能

功能概述：
编写服务器端，服务器端启动，然后启动客户端，通过客户端可以输入以下命令进行服务器上的文件查看：

1. cd 进入对应目录
2. ls 列出相应目录文件
3. puts 将本地文件上传至服务器
4. gets 文件名 下载服务器文件到本地
5. remove 删除服务器上文件
6. pwd 显示目前所在路径
7. 其他命令可以自己添加（mkdir 等等）
8. 无效命令不响应

项目启动方法
服务端
./ftpserver ../conf/server.conf
客户端
./client ip port

二期功能

1、密码验证

对于linux的登陆验证我想大家可能都知道是将输入的密码进行相应编码加密后与/etc/shadow文件中对应的密码部分进行比较，如果相同则验证通过，如果不同则表明密码错误，但是问题是我们要如何将用户输入的密码加密然后进行比较。

为了提高安全性，Linux引入了salt，所谓salt即为一个随机数，引入的时候为一个12bit的数值，当用户设置密码时，会随机生成一个salt与用户的密码一起加密，得到一个加密的字符串(salt以明文形式包含在该字符中），存储到密码文件中。crypt函数可以将用户输入的密码和salt一起适应某种算法进行加密，该加密后的字符串就是我们需要的与密码文件中密码对比的加密字符串。
crypt为支持不同的方式将salt格式化为

```
$id$salt$encode
```

其中id代表不同的算法

```
      1   | MD5
      2a  | Blowfish (not in mainline glibc; added in some
          | Linux distributions)
      5   | SHA-256 (since glibc 2.7)
      6   | SHA-512 (since glibc 2.7)    当前我们Linux采用的加密算法
```

这样我们就可以利用crypt函数将用户输入的字符加密，然后与密码文件中的密码进行对比了，有一个函数`getspnam()`可以根据用户名从/etc/shadow中得到密码，函数返回的数据结构为

```
    struct spwd {
```

```
        char *sp_namp;     /* Login name */
        char *sp_pwdp;     /* Encrypted password */
        long  sp_lstchg;   /* Date of last change (measured
                              in days since 1970-01-01 00:00:00 +0000 (UTC)) */
        long  sp_min;      /* Min # of days between changes */
        long  sp_max;      /* Max # of days between changes */
        long  sp_warn;     /* # of days before password expires
                              to warn user to change it */
        long  sp_inact;    /* # of days after password expires
                              until account is disabled */
        long  sp_expire;   /* Date when account expires (measured
                              in days since 1970-01-01 00:00:00 +0000 (UTC)) */
        unsigned long sp_flag;  /* Reserved */
    };
```

我们现在创建一个test用户，并将密码设置为1234来做个测试看一下
这是创建之后从/etc/shadow中取出来的密码部分

```
$6$qOrvsN41$gGlv8z7P1sAKuyaTAY03AvCn9/Z6Ygc4DJ9UweORVzNAI6TQsTfsdihPnIh3lSZV2CO2
Hjw.9bvJNdep3k.ER.
```

可以看到这里的salt为

```
$6$qOrvsN41
```

最后写个程序做下验证

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <shadow.h>
#include <errno.h>

void help(void)
{
    printf("用户密码验证程序\n第一个参数为用户名!\n");
    exit(-1);
}

void error_quit(char *msg)
{
    perror(msg);
    exit(-2);
}

void get_salt(char *salt,char *passwd)
{
    int i,j;

    //取出salt,i记录密码字符下标,j记录$出现次数
    for(i=0,j=0;passwd[i] && j != 3;++i)
    {
        if(passwd[i] == '$')
            ++j;
    }
```

```
    strncpy(salt,passwd,i-1);
}

int main(int argc,char **argv)
{
    struct spwd *sp;
    char *passwd;
    char salt[512]={0};

    if(argc != 2)
        help();

    //输入用户名密码
    passwd=getpass("请输入密码:");
    //得到用户名以及密码,命令行参数的第一个参数为用户名
    if((sp=getspnam(argv[1])) == NULL)
        error_quit("获取用户名和密码");
    //得到salt,用得到的密码作参数
    get_salt(salt,sp->sp_pwdp);

    //进行密码验证
    if(strcmp(sp->sp_pwdp,crypt(passwd,salt)) == 0)
        printf("验证通过!\n");
    else
        printf("验证失败!\n");

    return 0;
}
```

注意：gcc编译该程序时需要加入 -lcrypt

注：进行登录密码验证后，客户端只能看到自己的文件，不能看到其他用户的文件

`crypt` 加密函数

2.日志记录

1、日志记录客户端请求信息，及客户端连接时间
2、日志记录客户端操作记录，及操作时间

3.断点续传

1、进行`gets hello.avi`时候，判断本地是否存在hello.avi，如果存在，stat获取hello.avi的大小，传递给服务器的是gets hello.avi

4.mmap将大文件映射入内存，进行网络传递

1、当你发现文件大于100M，就使用`mmap`该文件，然后再传输

三期功能：

1、数据库连接

通过数据库存储用户名，salt值(采用随机字符串生成)，密码(密文形式存储)，通过数据库存储日志
如何生成随机字符串，参考下面代码

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define STR_LEN 10//定义随机输出的字符串长度。
char *GenerateStr()
{
    char str[STR_LEN + 1] = {0};
    int i,flag;

    srand(time(NULL));//通过时间函数设置随机数种子，使得每次运行结果随机。
    for(i = 0; i < STR_LEN; i ++)
    {
        flag = rand()%3;
        switch(flag)
        {
        case 0:
            str[i] = rand()%26 + 'a';
            break;
        case 1:
            str[i] = rand()%26 + 'A';
            break;
        case 2:
            str[i] = rand()%10 + '0';
            break;
        }
    }
    printf("%s\n", str);//输出生成的随机数。

    return str;
}
```

//下面的生成方法用到了malloc内存之后没有释放，存在内存泄漏的风险，不建议使用下面的方法，想想如何改进

```c
char* genRandomString(int length)
{
    int flag, i;
    char* string;
    srand((unsigned) time(NULL ));
    if ((string = (char*) malloc(length)) == NULL )
    {
        printf("Malloc failed!flag:14\n");
        return NULL ;
    }

    for (i = 0; i < length+1; i++)
    {
        flag = rand() % 3;
        switch (flag)
        {
            case 0:
                string[i] = 'A' + rand() % 26;
                break;
            case 1:
```

```
                string[i] = 'a' + rand() % 26;
                break;
            case 2:
                string[i] = '0' + rand() % 10;
                break;
            default:
                string[i] = 'x';
                break;
        }
    }
    string[length] = '\0';
    return string;
}
void main()
{
    GenerateStr();
    printf("%s\n",genRandomString(10));
}
```

2、密码验证

密码验证的方式是服务器先传输salt值给客户端，客户端crypt加密后，传输密码密文给服务器，服务器匹配后，判断是否成功

3、虚拟文件表

由数据库记录每一个用户的路径情况，每一个用户的文件及目录采用数据库的表进行存储，每个文件的文件内容以其MD5码存在磁盘上，并以md5码进行命名，所有用户的文件都存在一个目录里，各自用户只能看到自己的文件，不能看到其他人的文件，通过C接口得到每一个文件的MD5的源文件

四期功能：

1、实现长短命令分离

实现gets与puts命令与其他命令分离，可以考虑瞬间响应命令由主线程负责，上传下载命令由子线程负责，同时客户端在启动一个新线程时，通过TOKEN实现服务器的认证，下面是TOKEN生成及使用需要注意的链接
https://blog.csdn.net/zhu_xiao_yuan/article/details/77017196

2、超时断开连接

针对连接上的客户端，超过30秒未发送任何请求，关闭其描述符
推荐使用环形队列的方式实现

五期功能：多点下载

多点下载，多点下载是指客户端通过多个sfd连接不同的服务器，将一份大文件拆分为几段，不同段从不同的服务器进行下载
分为三级难度：

1. 固定从某几台服务器下载，而且文件多点下载一定一次性下载完毕
2. 从某台IP服务器上得到具有数据源的服务器地址，然后再从对应的地址进行下载，而且文件一次性下载完毕
3. 如果多点下载过程中，客户端断开，那么下次重新下载时，需要再次进行多点下载，所以客户端本地需要使用文件，或者数据库存储多点下载，每个位置下载到什么地方

想研究实际FTP协议，详见
http://blog.csdn.net/yxyhack/article/details/1826256
P2P协议
http://www.52im.net/thread-50-1-1.html

面试时询问和ftp有什么区别，个性化功能回答

1. 多级权限管理
2. 某一级主管可以分享给自己的下属小组查看文件
3. 文件分为可下载和在线阅读，读后即焚等