实验报告

姓名：王苑铮 学号：2015K8009922002

1.实验题目： socket应用编程实验

2.实验内容：

- **master分发任务**
  - Master通过读取workers.conf配置文件，获取每个worker的IP地址，然后分别建立连接
  - Master获取war_and_peace.txt文件长度，将统计任务等分到所有的worker
  - Master给每个worker发送消息，包括如下内容：
    - 消息总长度（4个字节）
    - 文件所在位置（因为master和worker在同一主机同一目录，所以给出相对位置即可）
    - 需要进行字符统计的起始位置（4个字节）和终止位置（4个字节）
- **Worker计算并返回结果**
  - 每个worker收到消息后，进行解析，根据指定统计区间对文件进行统计
  - Worker统计结束后，将每个字符出现的次数以4字节整数形式（网络字节序）返回给Master，因此传输消息长度为104字节
  - Master收到所有worker的消息后，进行聚合并输出到屏幕

3.实验过程

master:

(1)创建socket

```
    //Create socket
    int sock[2];

    for(int i=0; i<WORKERS_NUM ; ++i){
        sock[i] = socket(AF_INET,SOCK_STREAM,0);
        if(sock[i] == -1){
            printf("could not create socket %d\n",i);
        }
        printf("socket %d created %d\n",i,sock[i]);
    }
```

(2)连接worker

```
    //get workers ip addr
    char workers_ip[WORKERS_NUM][20];
    struct sockaddr_in workers[2];
    FILE* conf = fopen("./workers.conf","r");

    for(int i=0;i<WORKERS_NUM;++i){
        fscanf(conf,"%s",workers_ip[i]);
        printf("%s",workers_ip[i]);

        workers[i].sin_addr.s_addr = inet_addr(workers_ip[i]);
        workers[i].sin_family = AF_INET;
        workers[i].sin_port = htons(8888);

        //connect to worker
        if(connect(sock[i],(struct sockaddr*)&workers[i],sizeof(workers[i]))<0){
            printf("connect failed. Error\n");
            return 1;
        }
        printf("Connected\n");
    }
```

(3)统计文件行数

```
    //count lines
    FILE* txt=fopen(argv[1],"r");
    int line=0;
    char buf[2000];
    for(line=0 ; fgets(buf,sizeof(buf),txt)!=NULL ; ++line,memset(buf,0,sizeof(buf)) )
        ;
```

(4)发送、接收、统计

```
typedef struct send_package{
    char locate[200];
    int start;
    int end;
}send_package;

    //send and receive
    int stat[26]={0};
    int worker_stat[2][26]={0};
    int offset = line/2;
    int to=0;
    int send_len=0,recv_len=0;
    send_package s_p;

    memset(s_p.locate,0,sizeof(s_p.locate));
    strcat(s_p.locate,"./");
    strcat(s_p.locate,argv[1]);

    for(to=0, s_p.start=0 ; to<WORKERS_NUM ; ++to, s_p.start+=offset){
        s_p.end = s_p.start + offset;
        //send
        if( (send_len=send(sock[to],(char*)&s_p,sizeof(s_p),0)) <0 ){
            printf("send failed\n");
            return -1;
        }
        //receive
        if( (recv_len=recv(sock[to],(char*)worker_stat[to],sizeof(worker_stat[to]),0))
            printf("recv failed\n");
            return -1;
        }
        //stat
        for(int i=0;i<26;++i){
            stat[i] += worker_stat[to][i];
        }
    }
```

(5) 关闭socket

```
    close(sock[0]);
    close(sock[1]);
```

worker:

(1)统计：

```
void stat_char(char* message,int stat[]){
    int key;
    low(message); //upper to lower
    for(int i=0; i<strlen(message) ;++i){
        key = (message[i]>='a' && message[i]<='z')? message[i]-'a' : -1;
        if(key != -1)
            ++stat[key];
    }
    return;
}
```

(2)create,bind,listen,accept

```
    // Create socket
    if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Could not create socket\n");
                return -1;
    }
    printf("Socket created\n");


    // Prepare the sockaddr_in structure
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(8888);


    // Bind
    if (bind(s,(struct sockaddr *)&server, sizeof(server)) < 0) {
        perror("bind failed. Error\n");
        return -1;
    }
    printf("bind done\n");


    // Listen
    listen(s, 3);


    // Accept and incoming connection
    printf("Waiting for incoming connections...\n");


    // accept connection from an incoming client
    int c = sizeof(struct sockaddr_in);
    if ((cs = accept(s, (struct sockaddr *)&client, (socklen_t *)&c)) < 0) {
        perror("accept failed\n");
        return 1;
    }
    printf("Connection accepted\n");
```

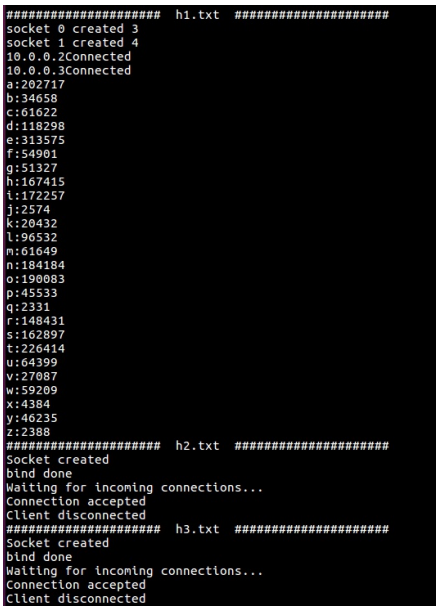
(3)收包，统计，返还结果

```
    // Receive a message from client
    int msg_len = 0;
    send_package sp;

    while ((msg_len = recv(cs, (char*)&sp, sizeof(sp), 0)) > 0) {
        char* locate = sp.locate;
        FILE* f=fopen(locate,"r");
        int stat[26]={0};
        char buf[2000]={0};
        int i;

        //ignore the line before start
        for(i=0 ; i<sp.start ; ++i){
            fgets(buf,sizeof(buf),f);
        }
        //stat
        for(int i=sp.start ; i<sp.end ; ++i){
            fgets(buf,sizeof(buf),f);
            stat_char(buf,stat);
        }

        write(cs, (char*)stat, sizeof(stat));
        memset(stat,0,sizeof(stat));
    }
```

python自动化脚本:

```
#auto make
os.system('mn -c')

#arg options
parser = argparse.ArgumentParser()
parser.add_argument('-notmake',action='store_true',help='if -notmake, we do not makefi
args = parser.parse_args()
if args.notmake==True:
    print('not make')
else:
    os.system('make clean')
    if os.system('make') != 0: #if success,return 0
        exit()
    print('Compile Success!!!')

print('preparing for mininet...')
topo = MyTopo()
net = Mininet(topo = topo)

print('net start')
net.start()
h1, h2, h3 = net.get('h1', 'h2', 'h3')

print('h2 running')
h2.cmd('./worker > h2.txt &')
time.sleep(3)
print('h3 running')
h3.cmd('./worker > h3.txt &')
time.sleep(3)
print('h1 running')
h1.cmd('./master war_and_peace.txt > h1.txt &')
time.sleep(10)
#CLI(net)
#net.stop()
print('net stop\n')

print('######################  h1.txt  ######################')
os.system('cat h1.txt')
print('######################  h2.txt  ######################')
os.system('cat h2.txt')
print('######################  h3.txt  ######################')
os.system('cat h3.txt')

print('test finish')
```

4.实验结果



5.结果分析

与标准的结果进行比对，运行结果正确

6.bug以及原因：

现象： 最开始的版本，每个字符的统计结果都明显偏少

排查： 在xterm中用gdb打开master和worker跟踪

结果： 最初始的设计中，我是让master给每个worker发半本war_and_peace去统计。数据量比较大。但是worker无法收到那么大的包。最终worker大约只收到master发出的数据的五分之一，因而结果偏少

解决： 只把war_and_peace的路径，以及每个worker开始与结束的行号发出去，数据包小了，就能正确收发了