<Assignment4-1>

과제명	Assignment4-1 (User Authentication & Access Control)			
과목명	시스템프로그래밍(H020-3-0922-01)(화 5 목 6)			
성명	정승훈	연락처	핸드폰	010–8648-7561
학과	전자통신공학과		이메일	tiktaktok116@naver.com
학번	2015707003	지도교수	김태석 교수님	
개발기간	2020.06.04~2020.06.11			
개발 환경	OS	Linux (Ubuntu 18.04 LTS)		
	Language	C		
	Development Tools	Visual Studio Code, gcc compiler		

과제 요구사항 및 구현 내용

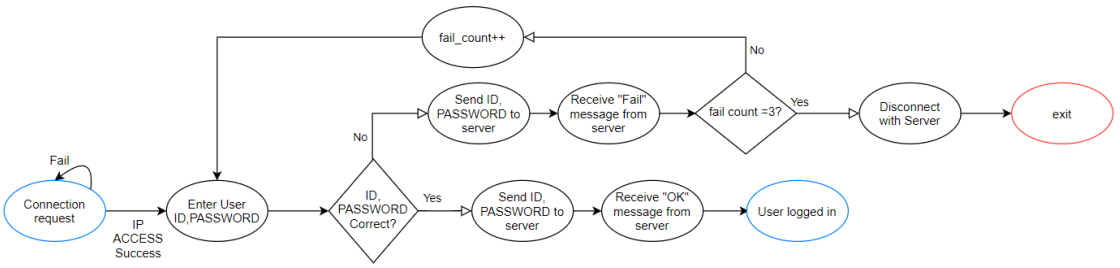
1. Client
   1) Client is executed with two parameter
      - IP address and port number of Server.
   2) After successful connection, displays the message "**It is connected to Server**"
   3) receives "REJECTION" or "ACCEPTED" from server
      - If it received "REJECTION" from server, (i.e. doesn't exist in the access.txt) than displays the message "**Connection refused**" and disconnects to server.
      - If it received "ACCEPTED" from server, than receives username and password.
   4) receives user name and password from standard input & passes user name and password to server
      - If it received "OK" from server, than displays the message "** User '...' logged in **".
      - If it received "FAIL" from server, than displays the message "** Log-in failed **".
      - If it received "DISCONNECTION" from server, (i.e. failures during 3 times) than displays the message "** Connection closed **" and disconnects to server
2. Server
   1) is executed with one parameter – Port number of server.
   2) After successful connection, displays the message "**Client is connected**", and displays client's IP, Port.
   3) checks client's IP to confirm possibility of connection using "access.txt" file.
      - If the client's IP doesn't exist in the access.txt, send "REJECTION", and display the message "**It is NOT authenticated client**".
      - If the client's IP exist in the access.txt, send "ACCEPTED".
   4) counts whenever client try to login, and displays the message "**User is trying to log-in (x/3)**".

I. Introduction

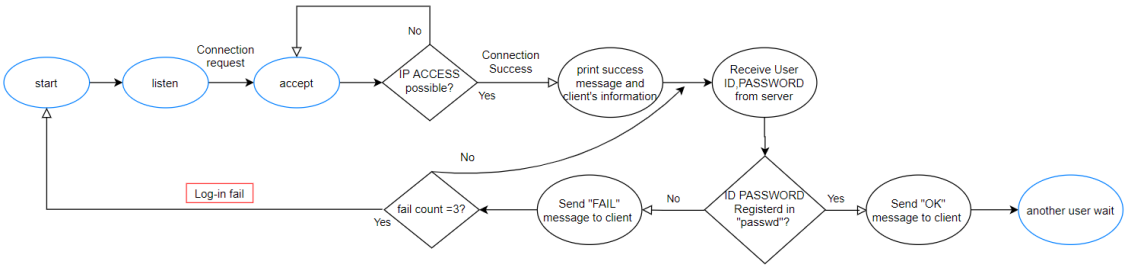
이번 Assignment 4-1에서는 User Authentication과 Access Control 을 구현하는 것이다. client에서 server로 해당 port와 IP 주소로 접근을 시도하면 Server에서 접근 가능한 IP인지 access.txt에 저장되어 있는 IP주소(wildcard 처리 포함)로 판단하고 접근 불가능할 때는 올바른 IP주소로 접속하라고 요구하고 접근 가능할 때 passwd에 저장되어 있는 ID와 PASSWORD를 바탕으로 로그인이 성공했을 때와 3번 실패했을 때 다시 접속하도록 구현하였다.

II. Flow Chart

Client's Flow Chart



Server's Flow Chart



III. Source Code

1) Client

```
////////////////////////////////////////////////////////////////////////
// File Name      : cli.c                                            //
// Date           : 2020/06/05 ~ 2020/06/11                          //
// OS             : Ubuntu 18.04.4 LTS                               //
// Student Name   : Seung Hoon Jeong                                 //
// Student ID     : 2015707003                                       //
// ------------------------------------------------------------------ //
// Title : System Programming Assignment #4-1                        //
// Description : User Authentication & Access Control                //
////////////////////////////////////////////////////////////////////////

/* 필요한 header file 선언 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

```
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <netinet/in.h>

#define MAX_BUF 20
#define CONT_PORT 20001

////////////////////////////////////////////////////////////////////////////////////////////////////
// Function : void log_in(int sockfd)                                                            //
// ==================================================================================================
== //
// Input: ID,PASSWORD                                                                            //
// Output: 1. login success : login check                                                        //
//         2. login fail : if you fail 3 times, relogin please                                   //
// ==================================================================================================
== //
// Purpose: Log in to server after IP ACCESS OK                                                  //
////////////////////////////////////////////////////////////////////////////////////////////////////
void log_in(int sockfd){
    int n;
    char user[MAX_BUF], *passwd, buf[MAX_BUF];
    /* 코드 작성 (hint:  Check if the ip is acceptable ) */
    int count = 0;
    read(sockfd,buf,sizeof(buf));

    /* receives “REJECTION” from server */
    if (!strcmp(buf,"REJECTION")) {
        printf("** Connection refused **\n");
        close(sockfd);
        exit(0);
    }
    /* receives “ACCEPTED” from server */
    else if (!strcmp(buf,"ACCEPTED")) {
        printf("** It is connected to Server **\n");
        bzero(buf,sizeof(buf));
    }

    for(;;) {
```

```
  /* 코드 작성 (hint: pass username to server) */
  /////////////// Log-in to server ////////////////////////
    write(STDOUT_FILENO, "Input ID : ",sizeof("Input ID : ")); // Enter ID
    read(STDIN_FILENO, user, MAX_BUF);
    passwd = getpass("Input Password : "); // Enter Password (Password security operated)
    write(sockfd,user,sizeof(user));
    write(sockfd,passwd,sizeof(passwd));

    n = read(sockfd, buf, MAX_BUF);
    buf[n] = '\0';
    if(!strcmp(buf, "OK")) {
      n = read(sockfd, buf, MAX_BUF);
      buf[n] = '\0';
      if(!strcmp(buf, "OK")) {
        /* 코드 작성 (hint: login success) */
        printf("** User '[user_name]' logged in **\n");
        break;
      }
      else if(!strcmp(buf, "FAIL")) {
        /* 코드 작성 (hint: login fail) */
        count++; // add count if you login fail
        printf("** Log-in failed **\n");
        if(count == 3) break; // if you login fail 3 times client connection refused
        else {
          bzero(buf,sizeof(buf));
          continue;
        }
      }
      else{ // buf is "DISCONNECTION"
        /* 코드 작성 (hint: three times fail) */
        printf("** Connection refused **\n");
        close(sockfd);
        exit(0);
      }
    }
  /////////////// Log-in End to server ////////////////////////
  }
}

////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
// Function : int main(int argc, char **argv)                                    //
// ==========================================================================================
== //
// Input: ./cli [IP_ADDRESS] [PORT_NUMBER]                                       //
// Output: 1.success : IP access success and login success                       //
//         2.fail : IP access fail or login fail                                 //
// ==========================================================================================
== //
// Purpose: IP ACCESS and login to server                                        //
//////////////////////////////////////////////////////////////////////////////////////////////
int main(int argc, char *argv[])
{
    int sockfd, n, p_pid;
    struct sockaddr_in servaddr;

    /* 코드 작성 */
    /* argument count exception handling */
    if(argc != 3) {
        printf("Usage :  %s [IP_ADDRESS] [PORT_NUMBER]\n", argv[0]);
        exit(0);
    }

    /* open socket */
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket");
        exit(0);
    }

    memset(&servaddr, '\0', sizeof(servaddr)); // initialize server socket information struct to zero
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]); // short data(port number) to network byte order
    servaddr.sin_port = htons(atoi(argv[2]));

    /* connect to server */
    if(connect(sockfd,(struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect");
        exit(1);
    }

    log_in(sockfd); // User authentication after IP access control
```

```
8    close(sockfd);
11   return 0;
9 }
12
0
12
1
12
2
12
3
12
4
12
5
12
6
12
7
12
8
12
9
13
0
13
1
13
2
13
3
13
4
13
5
```

Colored by Color Scripter

2) Server

```c
////////////////////////////////////////////////////////////////////////////
// File Name       : srv.c                                                //
// Date            : 2020/06/05 ~ 2020/06/11                              //
// OS              : Ubuntu 18.04.4 LTS                                   //
// Student Name    : Seung Hoon Jeong                                     //
// Student ID      : 2015707003                                           //
// ---------------------------------------------------------------------- //
// Title : System Programming Assignment #4-1                             //
// Description : User Authentication & Access Control                     //
////////////////////////////////////////////////////////////////////////////

/* 필요한 header file 선언 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <pwd.h>
#include <arpa/inet.h>

#define MAX_BUF 20

////////////////////////////////////////////////////////////////////////////////////////////
// Function : int ip_match(char *first, char *second)                                     //
// ====================================================================================== //
// Input: x                                                                               //
// Output: 1. int 1: matching success                                                     //
//         2. int 0: matching fail                                                        //
// ====================================================================================== //
// Purpose: IP ACCESS CONTROL, to match wild card characters                              //
////////////////////////////////////////////////////////////////////////////////////////////
int ip_match(char *first, char *second)
{
    if (*first == '\0' && *second == '\0') return 1; // If we reach at the end of both strings, we are done
    /* Make sure that the characters after '*' are present in second string.
       Assume that the first string will not contain two consecutive '*' */
    if (*first == '*' && *(first+1) != '\0' && *second == '\0') return 0;
    if (*first == '?' || *first == *second) return ip_match(first+1, second+1); // If the first string contains '?', or current characters of both strings match
    /* If there is *, then there are two possibilities.
      1. We consider current character of second string.
      2. We ignore current character of second string. */
    if (*first == '*') return ip_match(first+1, second) || ip_match(first, second+1);
    return 0;
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////
// Function : int user_match(char *user, char *passwd)                                        //
// ========================================================================================== //
// Input: x                                                                                   //
// Output: 1. int 1: ID, PASSWORD CORRECT                                                     //
//         2. int 0: ID, PASSWORD ERROR                                                       //
// ========================================================================================== //
// Purpose: USER ID,PASSWORD matching                                                         //
////////////////////////////////////////////////////////////////////////////////////////////////
int user_match(char *user, char *passwd) {
    FILE *fp;
    struct passwd *pw;
    fp = fopen("passwd", "r");

    /* 코드 작성 (hint: 인증 성공 시 return 1, 인증 실패 시 return 0  */
    memset((void *)&pw,0x00,sizeof(pw));
    while((pw = fgetpwent(fp)) != NULL) {
        if(!strcmp(user,pw->pw_name) && !strcmp(passwd,pw->pw_passwd)) // if user ID, PASSWORD equal, we are done
            return 1;
    }
    fclose(fp);
    return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////////
// Function : int log_auth(int connfd)                                                        //
// ========================================================================================== //
// Input: x                                                                                   //
// Output: 1. int 1 : login success, authentication OK                                        //
//         2. int 0 login fail, check 3 times, if you fail 3 times, Disconnect                //
// ========================================================================================== //
// Purpose: Log-in Authentication                                                             //
////////////////////////////////////////////////////////////////////////////////////////////////
int log_auth(int connfd)
{
    char user[MAX_BUF], passwd[MAX_BUF];
    int n, count=1;
    while(1) {
        /* 코드 작성 (hint: username과 password를 client로부터 받는다) */
        /* Read User ID */
        if((n = read(connfd,user,MAX_BUF)) > 0) {
            if(user[strlen(user)-1] == '\n') user[strlen(user)-1] = '\0';
        }

        /* Read UserID's Password */
        if((n = read(connfd,passwd,MAX_BUF)) > 0) {
```

```
        if(passwd[strlen(passwd)-1] == '\n') passwd[strlen(passwd)-1] = '\0';
    }

    write(connfd, "OK", MAX_BUF); // send to "OK" message to client
    printf("** User is trying to log-in (%d/3) **\n",count);

    /* when ID,PASSWORD Authentication sucess */
    if((n = user_match(user, passwd)) == 1){
        /* 코드 작성 (hint: 인증 OK) */
        write(connfd, "OK", MAX_BUF);
        return 1;
    }

     /* when ID,PASSWORD Authentication fail */
    else if(n == 0){
        printf("** Log-in failed **\n");
        if(count >= 3) {
            /* 코드 작성 (hint: 3 times fail) */
            write(connfd,"DISCONNECTION", MAX_BUF); // send to "DISCONNECTION" message to client
            return 0;
        }
        write(connfd, "FAIL", MAX_BUF);
        count++; // add fail count
        continue;
    }
  }
  return 1;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
// Function : int main(int argc, char **argv)                                                     //
// ============================================================================================== //
// Input: ./cli [IP_ADDRESS] [PORT_NUMBER]                                                        //
// Output: 1.success : IP access success and login success                                        //
//         2.fail : IP access fail or login fail                                                  //
// ============================================================================================== //
// Purpose: IP ACCESS and login to server                                                         //
////////////////////////////////////////////////////////////////////////////////////////////////////
int main(int argc, char *argv[])
{
  int listenfd, connfd;
  struct sockaddr_in servaddr, cliaddr;
  FILE *fp_checkIP; // FILE stream to check client's IP

  /* 코드 작성 */
  int clilen;
```

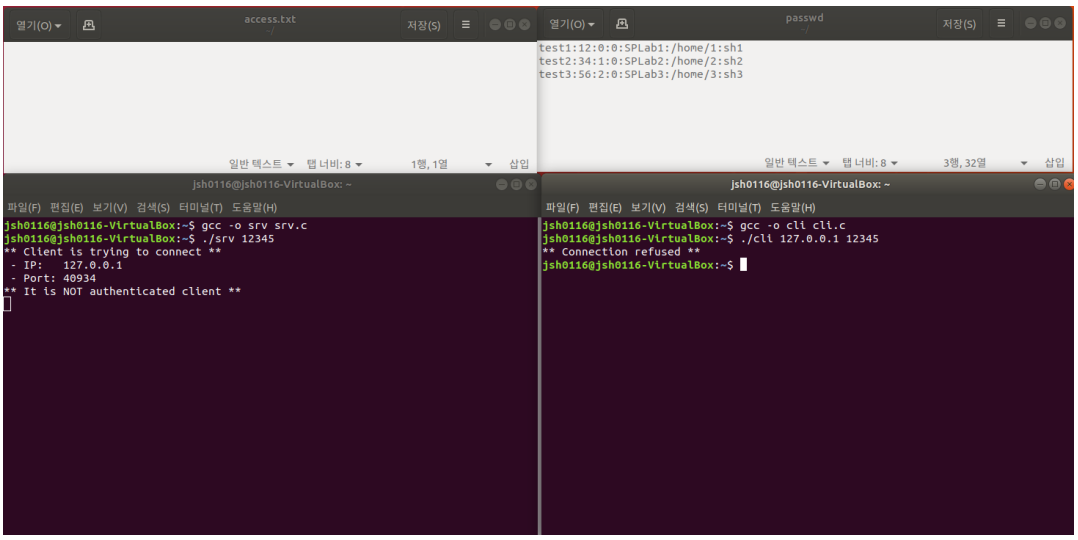
```
clilen = sizeof(cliaddr);
/* argument count exception handling */
if(argc !=2) {
    printf("Usage : %s [PORT_NUMBER]\n",argv[0]); // fixed argument format
    exit(0);
}
/* open socket */
if((listenfd = socket(AF_INET,SOCK_STREAM, 0)) < 0) {
    perror("socket");
    exit(0);
}

int option = 1;
/* prevent bind error after server terminated */
if(setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option)) < 0) {
    perror("setsockopt");
    exit(0);
}

memset(&servaddr, '\0', sizeof(servaddr)); // initialize server socket information struct to zero
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(atoi(argv[1]));

/* bind socket */
if(bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
    perror("bind");
    exit(0);
}

/* listen socket */
if(listen(listenfd,5) < 0) {
    perror("listen");
    exit(0);
}
for(;;) {
    connfd = accept(listenfd, (struct sockaddr *) &cliaddr, &clilen) ;
    /* 코드 작성 (hint: Client의 IP가 접근 가능한지 확인) */
    printf("** Client is trying to connect **\n");
    printf(" - IP:   %s\n",inet_ntoa(cliaddr.sin_addr));
    printf(" - Port: %d\n",ntohs(cliaddr.sin_port));
    fp_checkIP = fopen("access.txt", "r"); // check what IP is access posible?
    char pattern[MAX_BUF];
    fgets(pattern,sizeof(pattern),fp_checkIP);
    if(pattern[strlen(pattern)-1] == '\n') pattern[strlen(pattern)-1] = '\0'; // remove newline
```

```
    char str_ip[INET_ADDRSTRLEN] = {0};
    inet_ntop(AF_INET,&(cliaddr.sin_addr),str_ip,INET_ADDRSTRLEN); // get it back after storing this IP address in sockaddr

    /* when ip pattern matched */
    if(ip_match(pattern,str_ip) == 1) {
        printf("** Client is connected **\n");
        write(connfd,"ACCEPTED",sizeof("ACCEPTED")); // send to "ACCEPTED" message to client
        bzero(pattern,sizeof(pattern));
    }

    /* when ip pattern not matched */
    else if(ip_match(pattern,str_ip) == 0) {
        printf("** It is NOT authenticated client **\n");
        write(connfd,"REJECTION",sizeof("REJECTION")); // send to "REJECTION" message to client
        bzero(pattern,sizeof(pattern));
        continue;
    }

    if(log_auth(connfd) == 0) { // if 3 times fail (ok : 1, fail : 0)
        printf("** Fail to log-in **\n");
        close(connfd);
        continue;
    }

    printf("** Success to log-in **\n");
    close(connfd);
    }
}
```

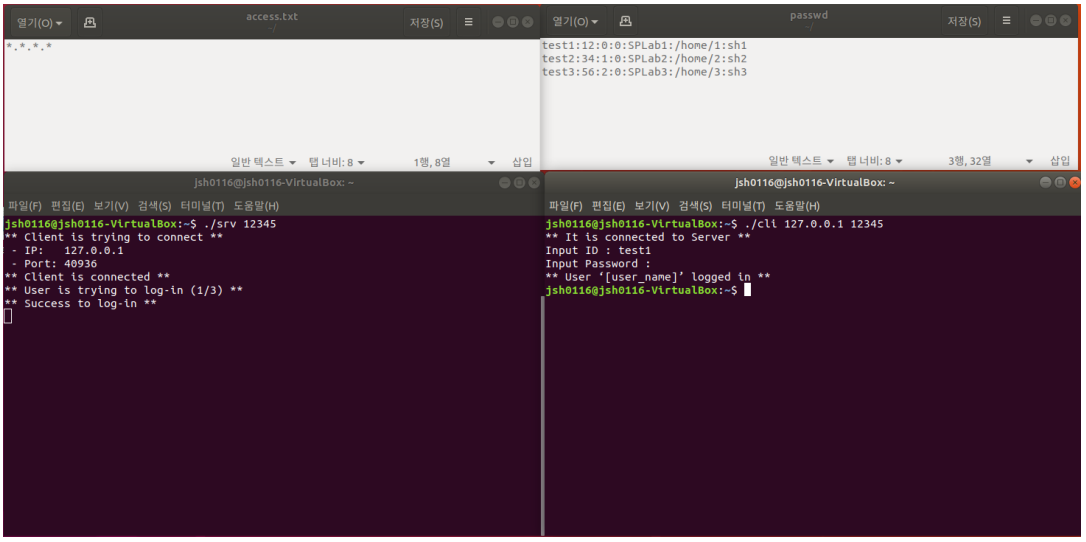
Colored by Color Scripter

IV. Result

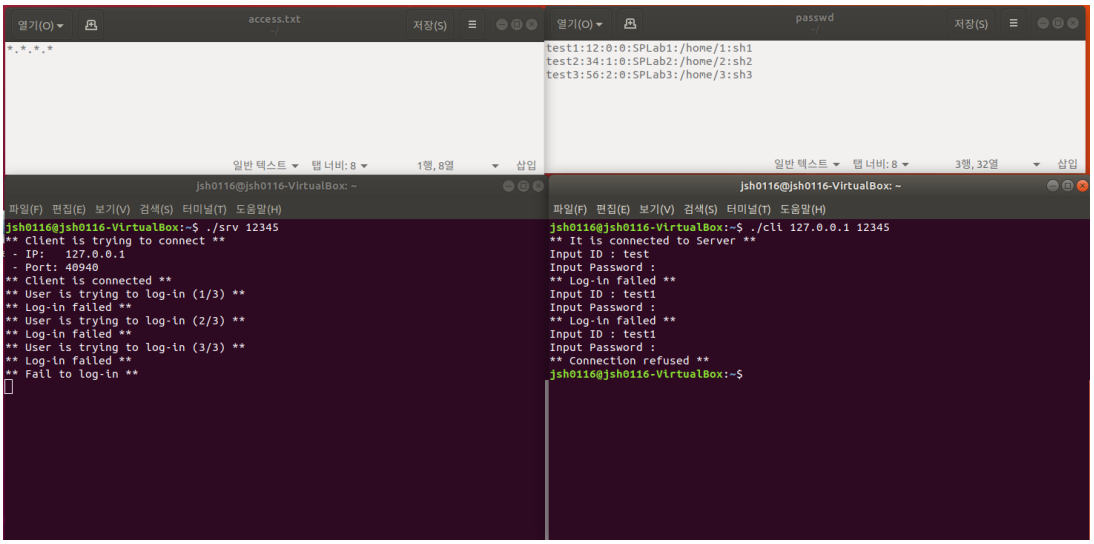
1) 접속이 불가한 IP를 가진 Client가 접속할 경우



2) 성공적으로 로그인을 마친 경우



3) 로그인을 세 번 시도했지만 모두 실패한 경우


access.txt
..*.*
passwd
test1:12:0:0:SPLab1:/home/1:sh1
test2:34:1:0:SPLab2:/home/2:sh2
test3:56:2:0:SPLab3:/home/3:sh3
jsh0116@jsh0116-VirtualBox: ~
jsh0116@jsh0116-VirtualBox:~$ ./srv 12345
** Client is trying to connect **
- IP: 127.0.0.1
- Port: 40940
** Client is connected **
** User is trying to log-in (1/3) **
** Log-in failed **
** User is trying to log-in (2/3) **
** Log-in failed **
** User is trying to log-in (3/3) **
** Log-in failed **
** Fail to log-in **
jsh0116@jsh0116-VirtualBox: ~
jsh0116@jsh0116-VirtualBox:~$ ./cli 127.0.0.1 12345
** It is connected to Server **
Input ID : test
Input Password :
** Log-in failed **
Input ID : test1
Input Password :
** Log-in failed **
Input ID : test1
Input Password :
** Connection refused **
jsh0116@jsh0116-VirtualBox:~$