CODE

server.h

```
struct user_details {
  char user_name[30];
  int user_id;
  int group_id;
  char comments[50];
  char password[15];
  char directory[36];
  char shell[10];
};

struct shadow {
  char user_name[30];
  char encrypted_password [30];
  int elapsed_days;
  int minimum;
  int maximum;
  int w_days;
  int inactive;
  int expire;
};
void encrypt(char* str);
void shadow_details(const struct user_details *user, struct shadow
*user_shadow);
void write_in_passwd(FILE *file, const struct user_details user);
void write_in_shadow(FILE *file, const struct shadow
user_shadow);
int check_userID (FILE *fp, int user_id);
int check_username (FILE *fp, char* username);
int check_directory(const char *path);
void generate_path(char *path, char *filename);
```

client.c

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

// encrypts password
void encrypt(char* str) {
  char temp;
  temp = str[0];
  int len = strlen(str);
  for(int i=0;i<len-1;i++) {
    str[i] = str[i+1];
    if(i==len-2) {
      str[i+1] = temp;
  }
  }
}

void shadow_details(const struct user_details *user, struct shadow
*user_shadow) {
    strcpy (user_shadow->user_name, user->user_name);
    strcpy (user_shadow->encrypted_password, user->password);
    encrypt (user_shadow->encrypted_password);
    user_shadow->elapsed_days = 56;
    user_shadow->minimum = 9;
    user_shadow->maximum = 5000;
    user_shadow->w_days = 5;
    user_shadow->inactive = 5;
    user_shadow->expire = 1000;
}
```

```
// This function writes in passwd.txt
void write_in_passwd(FILE *file, const struct user_details user) {
  fseek (file, 0, SEEK_END);
  fprintf(file,
"%s:x:%d:%d:%s:%s:%s\n",user.user_name,user.user_id,user.grou
p_id,user.comments,user.directory,user.shell);
}

// This function writes in shadow.txt
void write_in_shadow(FILE *file, const struct shadow
user_shadow) {
  fseek (file, 0, SEEK_END);
  fprintf(file, "%s:%s:%d:%d:%d:%d:%d:%d\n",
user_shadow.user_name, user_shadow.encrypted_password,
user_shadow.elapsed_days, user_shadow.minimum,
user_shadow.maximum, user_shadow.w_days,
user_shadow.inactive, user_shadow.expire);
}

//to check userID
int check_userID (FILE *fp, int user_id) {
    char *temp;
    char Line [100];
    int count = 0;
    int uid = 0;
    while (1) {
        if (fgets (Line, 99, fp) == NULL){
            return 2;
            }
        temp = strtok (Line, ":");
        temp = strtok (NULL, ":");
        temp = strtok (NULL, ":");
        uid = atoi (temp);
        if (uid == user_id)
        {
            count = 1;
                printf("UserID already exists!\n");
            return 0;
        }
    }
    return count;
}

//to check username
int check_username (FILE *fp, char* username) {
    char *temp;
    char Line [100];
    int count = 0;
    char uname[30];

    while (1)
    {
        if (fgets (Line, 99, fp) == NULL)
            return 2;
        temp = strtok (Line, ":");
        strcpy (uname, temp);
        if (strcmp (uname, username) == 0)
        {
            count = 1;
            printf("Username already exists!\n");
            return 0;
        }
    }
    return count;
}



// checks the path of the direcory
int check_directory(const char *path) {
    struct stat stats;

    stat(path, &stats);

    // Check for dir existence
    if (S_ISDIR(stats.st_mode))
        return 1;

    else{
      char* dirname = "etc";

      mkdir(dirname,0777);
      return 1;
    };
}


// getenv gets the value of the environment variable
// generate_path returns the path of the file after using the
information from the environment variable PFILE

void generate_path(char *path, char *filename) {
  if (getenv("PFILE") == NULL) {
    return;
  }
  else {
    strcpy(path, getenv("PFILE"));
    strcat(path, filename);
  }
}
```

adduser.c

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

```
#include <string.h>
#include <sys/stat.h>
#include "server.h"

/*

using a to load file
“a+” – Searches file. If the file is opened successfully fopen( )
loads it into memory and sets up a pointer that points to the last
character in it. If the file doesn’t exist, a new file is created.
Returns NULL, if unable to open file.
The getenv function obtains the current value of the environment
variable, name.
The setenv function inserts or resets the environment variable
name in the current environment list. If the variable name does not
exist in the list, it is inserted with the given value. If the variable
does exist, the argument overwrite is tested; if overwrite is zero,
the variable is not reset, otherwise it is reset to the given value

*/

int main(int argc, char **argv) {
  if (argc < 9) {
    printf("Insufficient number of arguments\n");
    printf ("Usage: adduser <-u> <UID> <-g> <GID> <-c>
<Comments> <Name>\n");
    exit(1);
  }
  if (strcmp(argv[1], "adduser") != 0) {
    printf("Invalid command\n");
    exit(1);
  }
  setenv("PFILE", "./etc/", 1);
  check_directory(getenv("PFILE"));
  struct user_details user;
  struct shadow user_shadow;
  strcpy(user.password, "PES1UG19CS094");//password is
hardcoded as it is not taken as input in the project

  strcpy(user.directory, "/user/home");
  strcpy(user.shell,"/bin/bash");
  char passwd_path[1000];
  char shadow_path[1000];

  generate_path(passwd_path, "passwd.txt");
  generate_path(shadow_path, "shadow.txt");

  FILE *passwd = fopen(passwd_path, "a+");
  FILE *shadow = fopen(shadow_path, "a+");

  if (check_username(passwd, argv[8]) == 0) {
    exit(1);
  }
 fseek(passwd, 0, SEEK_SET);
 fseek(shadow, 0, SEEK_SET);
  if (check_userID(passwd, atoi(argv[3])) == 0) {
    exit(1);
  }
  user.user_id = atoi(argv[3]);
  user.group_id = atoi(argv[5]);
  strcpy(user.comments, argv[7]);
  strcpy(user.user_name, argv[8]);
  //initialising shadow
  shadow_details(&user, &user_shadow);

  write_in_passwd(passwd, user);
  write_in_shadow(shadow, user_shadow);

  fclose(passwd);
  fclose(shadow);
```

```
}
```

Makefile

```
all: adduser

adduser: adduser.o client.o
	gcc adduser.o client.o
adduser.o: adduser.c
	gcc -c adduser.c
client.o: client.c
	gcc -c client.c
```