Report

Name: Abir Al Mahdi Akhand
ID: 2221699642
Section: 7
Serial: 14
Course: CSE115

Please Turn Over

Project Parts Name:

I have done the following parts of the project as given below:

1. Authentication
2. Encrypting password
3. Checking the existance of username
4. Employee search option
5. Sending employee on leave feature
6. Displaying of the employee data
7. Counting the number of employee and leave files
8. Merging the whole code

1. Authentication:

I have declared a function 'authenticate()' which takes 4 arguments: char *username[100], char password[8], char *loginData[2][2], int k. Here, I have declared int variable isAuthenticated=0. Then using if statement and strcmp() function, I have checked whether the password matches with the matching index of the array or not. If yes, the value of isAuthenticated is changed to 1. Then this variable is returned.

Description/Explanation:

Now I will explain my code below:

1. The function 'authenticate()' which takes 4 arguments: char *username[100], char password[8], char *loginData[2][2], int k. Also, I have declared int variable isAuthenticated=0.

```
int authenticate(char *username[100],char
password[8],char*loginData[2][2],int k){
      int isAuthenticated=0;
}
```

2. Then using if statement and strcmp() function, I have checked whether the password matches with the matching index of the array or not. If yes, the value of isAuthenticated is changed to 1. Then this variable is returned.

```
  if (strcmp(password,loginData[k][1])==0) {
        isAuthenticated=1;
    } return isAuthenticated;
```

2. Encrypting Password:

I have declared a function 'encryptPassword ()' which takes 1 argument: char password[8]. Here, This function displays the characters of the password as asterisks (*) in the screen.

Description/Explanation:

Now I will explain my code below:

1. I have taken the password with getch() method after declaring int p=0.

```
int p=0;
password[p]=getch();
```

2. With each key stroke, instead of the actual letters, it will display asterisk (*) on the console. This is done using a do-while loop by incrementing the value of p as p++.

```
    do {
        password[p]=getch();
        if(password[p]!='\r' && password[p]!='\b')
        {
            printf("*");
        }
        p++;
    } while(password[p-1]!='\r');
```

3. Here, I have used an if statement where the character is not equal to '\r' (cursor) and not equal to '\b'. Then lastly, I have appended '\0' at the end of the string to mark its end

```
    do {
        password[p]=getch();
        if(password[p]!='\r' && password[p]!='\b') {
            printf("*");
        } p++;
    } while(password[p-1]!='\r');
    password[p-1]='\0';
```

3. Checking the Existence of Username:

This function checks whether the user inputted username is actually existing or not. If exists, it returns 1, else 0.

Description/Explanation:

Now I will explain my code below:

1. This function takes 2 arguments char*username[100] and char*loginData[2][2]. Ihave also declared int variable flag=0.

```
int checkUsername(char *username[100], char *loginData[2][2]){
    int flag=0;
}
```

2. Then, using a for loop, the function goes through every index of the array and matches the username with the content of the index in the array(using strcmp() function). If it matches, value of flag turns 1. Then 1 is returned, else 0 is returned.

```
    for (int k=0; k<2; k++)
    {
        if (strcmp(username,loginData[k][0])==0)
        {
            flag=1;
        }
    }
    return flag;
```

4. Employee Search Option:

This function returns nothing and takes no argument. It finds the matching ID number the user inputted, and displays the employee data in the console which user have searched.

Description/Explanation:

Now I will explain my code below:

1. Here, I have declared int id, char str[5], content[255], folderName[50]. Using strcpy(), I have set the value for folderName to "./Employees/". Then it takes employee ID as user input. Then it converts the id to string using sprintf() function.

```
int id;
char str[5], content[255], folderName[50];
strcpy(folderName,"./Employees/");
printf("Enter employee ID: ");
scanf("%d",&id);
sprintf(str, "%d", id);
```

2. Then, it adds ".txt" to the name to give it an extension. Then it is added to folderName to make it a file path. Then I have declared FILE pointer and opened the file with the name of folderName using fopen(). If fp is NULL, that means there was no such employees, else it will read the data of the employees using fgets(). Then it will print the data in the console.

```
strcat(str,".txt");
strcat(folderName,str);
FILE *fp=fopen(folderName,"r");
if (fp==NULL)
{
    printf("No such employees found");
}
else
{
    fgets(content,255,fp);

    puts(content);
    fclose(fp);
}
returnToMenu();
```

5. Sending Employee on Leave feature:

This function returns nothing and takes no argument. It finds the matching ID number the user inputted, and sends the employee on leave/vacation.

Description/Explanation:

Now I will explain my code below:

1. I have called checkDirectory() function first to check whether the required directories exist. The I have declared char empId[10], folderNamee[10], folderName2[10],ch, empId2[10], contentt[255], FILE*fp=NULL, FILE*fp2=NULL variables.

```
    checkDirectory();
    char empId[10], folderNamee[10], folderName2[10],ch, empId2[10],
contentt[255];
    FILE*fp=NULL;
    FILE*fp2=NULL;
    strcpy(folderName2,"./Leave/");
```

2. Using strcpy() function, I have set the value of folderName2 to "./Leave/". Then I have taken input of empID and added ".txt". Then I have set the value of folderName to "./Employees/". Then I have opened the source file and target file. Using fgets() I have read the contents of source and copied it on target file using fprintf().

```
    scanf("%s",&empId);
    fflush(stdin);
    strcat(empId,".txt");
    strcpy(empId2,empId);
    strcpy(folderNamee,"./Employees/");
    strcat(folderNamee,empId2);
    fp=fopen(folderNamee, "r");
    strcat(folderName2,empId2);
    fp2=fopen(folderName2, "w");
    fgets(contentt,255,fp);
    fprintf(fp2,"%s",contentt);
    printf("Employee sent on Leave!");
    fclose(fp);
    fclose(fp2);
```

6. Displaying of employee Data:

I have created a function called displayEmployees() where all the employee data stored in different files are to be displayed using loop and file reading methods.

Description/Explanation:

Now I will explain my code below:

1. I have declared a file pointer fcpy, character variables folderName and fileName2 and content. I have set the value of folderName to "./Employees/".

```
FILE *fcpy=NULL;
char folderName[50];
char fileName2[10], content[300];
strcpy(folderName,"./Employees/");
```

2. Using for loop, I have set the value of int i to 0 and it will increment upto total number of employees (found using countFiles() function discussed later). With each loop, the employee ID is derived by 100+i and converted it to string using sprintf() function, which is also the name of the file where the individual employee data is stored. I have used strcat() function to add ".txt" to the filename. Then again using strcat() function, I have joined the name of the file with the name of the folder. Finally, I have read the contents of the folder using fcpy() which opens the file in reading mode. Then I have printed the content of the file in the console. In this way, the contents of all the files are displayed in the console using the for loop. Lastly, I have again set the value of folderName to "./Employees/" and fileName2 to empty. Then I have called the returnToMenu() function (described by other member).

```
for (int i=0; i<countFiles(); i++) {

     sprintf(fileName2, "%d", 100+i);
     strcat(fileName2,".txt");
     strcat(folderName,fileName2);
     fcpy=fopen(folderName, "r");
     fgets(content, 255, fcpy);
     printf("%s",content);
     fclose(fcpy);
     strcpy(folderName,"./Employees/");
     strcpy(fileName2,"");
 }
 returnToMenu();
```

7. Counting the number of employee files:

I have created two function called displayEmployees() and displayLeaveEmployees() where all the employee data stored in different files are to be displayed using loop and file reading methods.

Description/Explanation:

Now I will explain my code below:

1. I have declared integer variable numberOfFiles=0, directory pointer d, and a instance of the structure dirent *dir (dirent structure is fetched from <dirent.h> library). Then I have opened the directory named "Employees" using opendir() function.

```
int numberOfFiles=0;
DIR *d;
struct dirent *dir;
d = opendir("./Employees"); //OR "./Leaves/"
```

2. I have used conditional statement if directory exists, then a while loop will run. Its condition is set to readdir() function (that is, until all the files in the directory are iterated through). With each loop, the value of numberOfFiles will increment by 1. After that, I have closed the directory with closedir() function. Then the function will return the number of files minus 2 (2 is the extra unnecessary outputs).

```
if (d)
{
    while ((dir = readdir(d))!= NULL)
    {
        numberOfFiles++;
    }
    closedir(d);
}
return numberOfFiles-2;
```

5. Merging the whole code:

I have collected all the functions from other members of the group and merged all the functions to a single code format.

```
#include ...
struct EmpData{ … }
void mainMenu(){ … }
void checkDirectory(){ … }
int countFiles(){ … }
void displayEmployees(){ … }
int deleteEmployee(){… }
int edit(){ … }
int addEmployee(){ … }
int returnToMenu(){ … }
int main(){ … }
```

Conclusion:

Our project, Employee Management System is developed in the lights of the modern era software that almost all the corporate companies require. It focuses on Adding, Editing, Deleting and Displaying the employee data under a single platform. Using this prototype, similar software can be more effectively made using other technologies such as ElectronJS, Django or other frameworks. In light of the recent advancement of technology, such a software is highly demanded in the modern-day software development industry.

Please Turn Over