Introduction

The main objective of this project, briefed out during the Software Design1E lecture on the 24th April 2002 by C. Coghill was to produce a prototype of a security keypad system that could be positioned at the entrance to a building so that employees could sign in or out using a PIN number. Display of people logging in and out, warning when someone gets their PIN wrong too many times and emergency alert options were required to be printed onto a screen, so that a security guard could monitor activites within the building.

The project was designed to give the students some experience using simple hardware from a C program on a UNIX system using dynamic data structures. The hardware used in this project is the Rippert Board, a fairly versatile board developed within the School of Engineering.

Project specifications :

- The C program written should run by typing something like ./keypad at the Linux command line.
- When the program starts, it should display a brief welcome message on the LCD for about 2 seconds.
- Program should then, on the LCD ask the employee for their ID and let them type in a 3 digit number to identify themselves.
- If ID number is valid, it should say "hello" and ask for the 4-digit PIN number, else display a warning message saying the ID was incorrect and ask again.
- If the ID and PIN are correct a welcome message should be displayed on the LCD telling them if they are signed in or out, how many people are signed in and display a small menu allowing them to sign out or change PIN.
- If no buttons are pressed for about 5 seconds it should return to the screen asking for the ID.
- If the PIN number entered by an employee is incorrect a warning message should be displayed for about 2 seconds and PIN should be asked for again. If employee enters an incorrect PIN for three times in a row the keypad should be disabled for 5 seconds after displaying a warning message and return to 'Enter Employee ID:' menu.
- If the employee decides to change PIN after signing in, a new 4 digit PIN number should be asked for twice. If they match, an appropriate message should be displayed letting them know the PIN was changed.
- If employee decides to sign out their status should be updated appropriately and displayed. Should resume from 'Enter Employee ID:' menu.
- The Security Guard Console (PC) should display an activity log from the keypad with date and time for each of the following:
    - every time someone enters an invalid ID
    - every time someone enters an invalid PIN more than three times in a row
    - a signed in/out message every time someone signs in or out
    - a 'changed PIN' message every time someone changes their PIN
    - a highlighted 'EMERGENCY' message if someone holds down the * (star) key on the keypad, at any time, for more 2 seconds

Development of the design

The initial attempt in this stage was to read from the given input file and write data into appropriate members in the structure. Checking for availability of records, allocating memory while creating the linked list and then creating the structure members as required was carried out. It was also important that the end of structure was marked, at the end of all the records in the input file. Details given out in the project hand out were followed step by step ensuring all requirements were met. In this stage the format of the input file was exactly as shown in the handout.

The second stage of the project was to activate the PIN change when user selected '2' in the menu after signing in and writing it to the structure so that the next time the user signs in the new PIN would be active. The 4 digit PIN number needed to be asked twice, compared, and if equal PIN was to be changed.

Timer was considered as the third stage of the project. This was achieved using standard functions in <time.h> library. Firstly, the time delay method (get_time function in the submitted code) was worked on so that messages could be displayed on the LCD for the required duration. Secondly, the alarm was to be activated when * (star) key was pressed down for around 2 seconds. To achieve this task, the provided getkey function was modified so that the loop terminates and returns the assigned value indicating alarm button was pressed.

The final stage of the project was to print relevant messages to the Security Console as shown in the project sheet. The date and time were obtained using strftime and finally printed out in a nice format. A summary was printed out to screen every 20 lines with the total number of people signed in, along with their first and last names.

These were the main specifications set out in the project hand out. Some extra features have been added to the code such as refreshing the screen if fewer digits have been entered for id or PIN, refreshing the display if user doesn't enter anything for longer than 5 seconds in any menu and printing out the final summary in a nicer format along with input file error handing functions. These special features have been explained in detail later in the report.

Special features of the design

- The most important feature is the very high stability and compatibility of the program. The program has been tested for all kinds of possible errors in the input file and it never crashed. The following possibilities are not detected as errors but taken as a normal entry and processed accordingly.
    1. Any amount of spaces at the beginning, between entries or at the end. (but total length should be within declared length of array record)
    2. Any number of blank lines anywhere in the input file.

    The following are detected as input errors:
    1. Any number of commas other than two
    2. A middle name
    3. More or less number of digits for employee id or PIN
    4. Any non-alpha numerical character in the file (i.e. Characters such as &, *, #, $)
    5. Employee name, id and/or PIN missing
    6. Input file missing.
    7. Only two (even less or more) commas present in one line. (i.e. First name, last name, id and PIN all missing but the commas present)

    It should also be noted that the program runs without failure even for a blank input file.

- The LCD refreshes automatically after 5 seconds if the user doesn't enter anything within that time in any given menu letting other users use the keypad in the event of someone leaving it after signing in.
- The LCD also refreshes if the user enters less than 3 digits for the employee id and/or less than 4 digits for the PIN and waits for 5 seconds without pressing any button.
- The # and * keys have been disabled so that user cannot enter them as id or PIN numbers, but can still use the * key if wanted to activate the emergency button. Any # and/or * characters are detected as errors in the input file too, so that there won't be a mismatch between entered digits and the actual record in the input file.
- The emergency button can be activated at any time in any given menu.
- Whenever an employee changes a PIN, the file is re-written with relevant changes ignoring errors if any. This helps the changed PIN to be active even the next time the program is run.
- Keypad is disabled for 5 seconds if employee gets the PIN wrong 3 times in a row providing better security.
- Emergency button can be activated even while keypad has been disabled.
- Emergency button can be activated at the end of each line right after the message has been displayed on the LCD.

- If there are any extra characters after one record, the last few unwanted characters are ignored and only the relevant members are created. This helps the program to run even if there is a full stop or any other accidentally pressed character in the input file after relevant record.
  Eg: Fred Bloggs, 001, 7878 likethis
- If employee decides to change PIN but enters anything less than 4 digits and waits for 5 seconds, it'll show a message "Enter 4 digit PIN number". Also it'll display appropriate messages whenever the PIN is changed or else when the user attempts to change but PINs do not agree. Thus the program is very user friendly.
- Once an employee signs in and if the screen is refreshed before the person signs out, the next time the employee enters the employee id, the PIN will be asked for, as means of higher security. This helps to verify the identity of the person.
- 'Signed in' messages are displayed to the security console and number of people logged in updated, only if the employee hadn't signed in before. Therefore after signing in for the first time employee has to sign out before a new 'signed in' message is printed to the screen and total number updated. This lets the Security Guard monitor the first time employee actually signed in and the total number of employees logged in.
- All variables (including fd) have been declared and used as local variables providing the option of using multiple keypads simultaneously.
- Id length and PIN length have been defined so that the program is capable of handling any number of digits for id and PIN number. This makes the program capable of handling large amounts of employee data provided that sufficient memory is available.

The basic algorithm used in this Project is shown below.

1. Get data from userlist.dat
2. Create linked list
3. Ask for user id
4. Check if id exists in userlist.dat. Ask for PIN if id is correct or else display a warning to LCD and a message to Security Guard Console. Repeat from step 4.
5. Ask for PIN number.
6. Check if PIN is correct. If PIN is incorrect goto step 6 and ask for PIN again. If employee gets PIN wrong 3 times in a row, display appropriate message to Security Guard Console and disable keypad for 5 seconds. Repeat from step 4.
7. If PIN is accepted, display appropriate message to Security Screen and a menu on the LCD where user can either sign out or change PIN along with the total number of people signed in.
8. If 'sign out' option is selected, show message on Security Screen and repeat from step 4.
9. If employee selects 'Change PIN' ask for new PIN twice and change PIN if they match. Show message on Security Console every time PIN is changed. Display a warning message on LCD otherwise. If user doesn't press any buttons for more than 5 seconds repeat from step 4.
10. Let employee activate alarm by pressing the * key at any given time. Show a highlighted "Emergency" message on Security Guard Console.

A more detailed algorithm with relevant data flow is summarised in the following page:

Overview of functions used:

1. main ()
   This is the function that is called first when the code is compiled and executed. It checks for all necessary hardware devices and proceeds if available or else terminates after displaying the relevant error message. It calls function create(), where the linked list is created and then calls function compare_in, where employee data is asked for. Finally it frees all memory that was used up and exits.

2. EMPLOYEE* create (void)
   The linked list is created through this function. It checks for possible errors in the input file and doesn't create a new structure if any errors were detected in a particular record and moves onto the next record. Memory is also allocated through this function if the record wasn't erroneous. It was important that a pointer was used to keep track of the last useful record since memory is allocated before the next record is scanned, checked and the structure created. Finally the last useful record's 'next' member is made NULL and a pointer to the beginning of the linked list is returned.

3. void compare_in (EMPLOYEE* str, int fd)
   Employee id is asked for, after displaying a welcome message. If the id was found in the input file the employee's pin is asked for and displays if the person is in or else if the pin entered was incorrect. If entered pin was incorrect the first time, it'll ask for the pin two more times and disable the keypad for 5 seconds if all three tries failed. It also checks if the entered employee id was incorrect and if so gives an appropriate error message. This function lets the user choose if he or she wants to sign out or change pin after signing in and also displays the total number of employee signed in. If 'change pin' option was selected it lets the user change pin after entering the new pin twice. If the two pins entered are not the same it displays an error message – 'pins disagree'.

   This function also allows the user to activate the emergency button. The user will be notified whenever the alarm button is pressed and held for 2 seconds. It should be noted that the screen will automatically go back to the previous menu if user doesn't press any buttons for 5 seconds. However, the total number of people will be incremented and security guard notified, only if the user hadn't signed in once before. For example if the user signs in, leaves the keypad for a while (keypad would get refreshed during this time) and signs in again before signing out the total number of people would be the same as before.

   Relevant messages are also displayed to the Security Console every time a user enters an invalid id, signs in, enters incorrect pin three times, signs out, activates the alarm and changes pin.

4. int pin_related (int j, EMPLOYEE* ptr, EMPLOYEE *start,int fd)
   This function handles all pin related operations. It lets the user sign out or change pin after signing in or else activate the alarm. If 'change pin' option was selected it will ask the user to enter new pin twice and if they match the pin will be changed. The file will also be written to every time an employee changes the pin, so that the new pin will be active even if the program was restarted. If user enters less than 4 digits for the new pin, it'll display a "Enter 4 digit PIN number: " message thus letting the user know the length of pin should be four. Relevant messages are also printed to the Security Guard Console.

   The user is expected to enter only 1 or 2 (or else the * key if wanted to activate the alarm) in this menu. However if any other key is pressed a warning message will be displayed saying 'Invalid entry'.

4. int check_ids (char* string1, char string2[])
   Compares the two strings, string1 and string2 and returns 1 if they match and 0 otherwise.

5. void get_time (int i, EMPLOYEE *str, int fd)

This is a delay method. Uses standard "difftime(stop, start)" statement from <time.h> library. It also checks if alarm button was pressed so that user can activate alarm even while messages are displayed on the LCD.

6. void print_msg (int time, int hello, char* msg_ptr1, char* msg_ptr2, int fd, EMPLOYEE *start)
   This function is used to display messages to the LCD in the suggested format. 'msg_ptr1' is used to write the first line and 'msg_ptr2' the second. It also calls the get_time function if a message is to be displayed for a specific duration.

7. int check_buttons (int i, int fd, EMPLOYEE *ptr, EMPLOYEE *start)
   Used to check which button has been pressed in the menu after an employee signs in. It also returns an assigned value if user pressed no button for longer than 5 seconds. If alarm button was pressed a message will be displayed on screen and Security notified.

8. void recreate_struc (EMPLOYEE* ptr)
   Function used to write to the file with relevant changes whenever an employee changes pin.

9. char store_idpin (int fd, int diff_time, EMPLOYEE *str)
   This function returns the char value of the key pressed. It also checks if pressed key was to activate alarm or else if no button was pressed for 5 seconds. Returns 'N' if no button was pressed and 'A' if alarm was activated after printing an appropriate message.

10. void clear (int fd)
    Used to clear the LCD display.

11. void alarms (int fd, EMPLOYEE *start),
    int alarms_return (int fd, EMPLOYEE *start)
    Both these functions are used to check if alarm was activated. The return type of the first is void and that of the second is int.

    The first function is called when it's not important to check is alarm was activated or not. For example after the first welcome message has been displayed (Keypad version 3.0.).

    The second function is called whenever it's important to check if alarm was activated or not before displaying the next menu. For example if alarm was activated in the menu "You're In ( ). 1.Out 2.Change PIN" the next menu that should be displayed is "Please Enter Employee ID:". If no alarm was activated it should proceed onto the menu as requested by employee.

12. void print_time (char *ptr1, char *ptr2, char *ptr3, char *ptr4, int fd, EMPLOYEE *start)
    It is required to print appropriate messages to the Security Guard Console every time an employee signs in, changes pin, enters incorrect id, enters incorrect pin 3 times and signs out. It is also a requirement that a brief summary of the number of employees signed in be printed along with their names approximately every 20 lines. These details should be printed out with date and time.

    A static variable has been used in this function to keep track of the number of lines. The variables declared static so that it doesn't get initialized every time the function is called.

    'strftime' has been used to get the system time. strcat is used to concatenate strings to the required format as shown in the specification sheet. The four char pointers, declared in the function definition are used to print out relevant messages in the required format.

Recommendations:

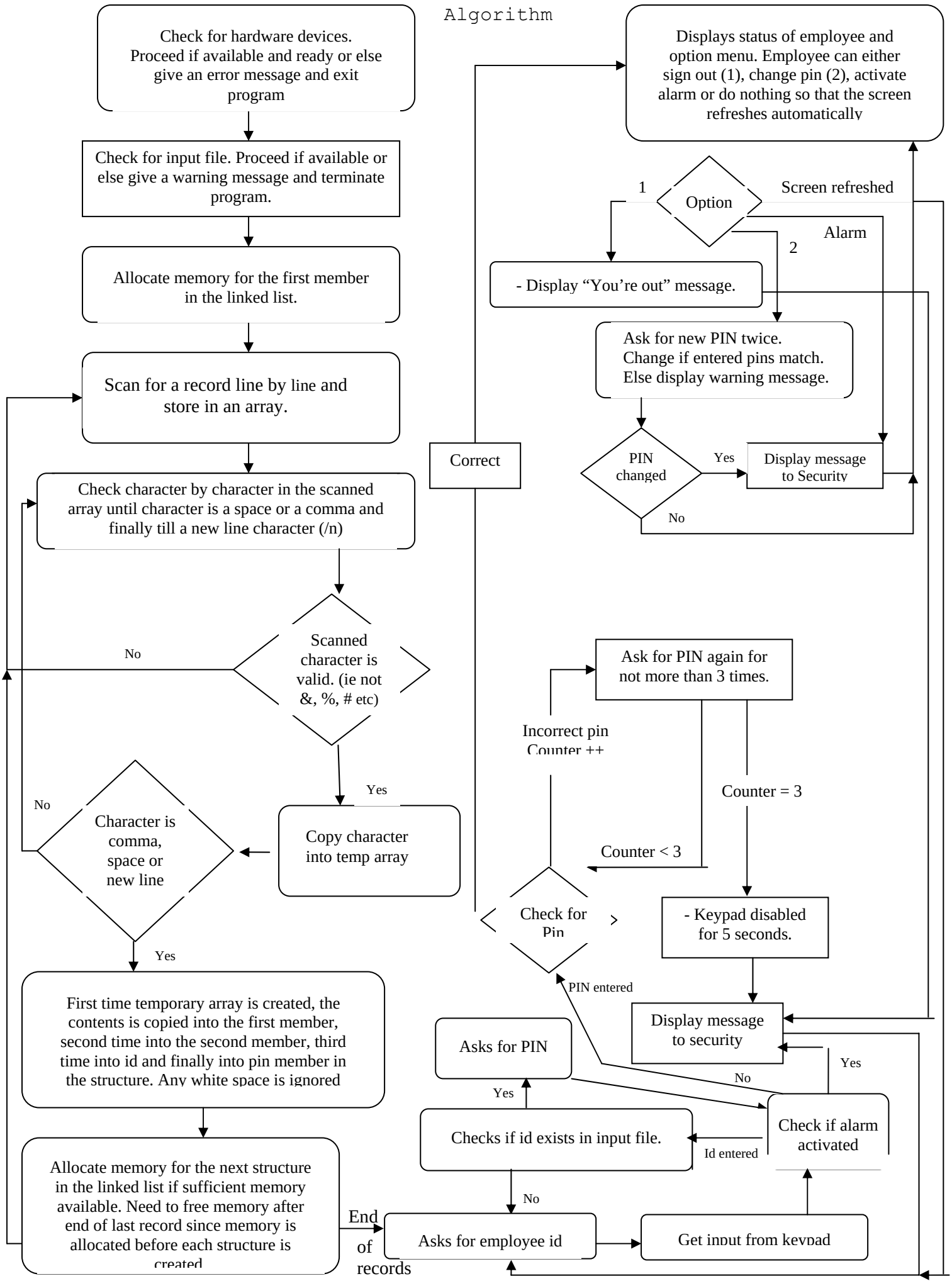
Depending on the usage of the keypad the following could be implemented as means of higher practical importance and security.

- Data in the input file can be checked before processing to ensure that the first and last names consist of only alphabetical characters and the id and pin number consist only of numerical digits.
- The user can be asked to enter the required input file so that the program would be capable of handling any file without getting restricted only to userlist.dat.

Conclusion:
The submitted code meets all the specifications set out in the Software Design1E Project 2 hand out. It also has some added special features (as mentioned in page 2 in this report) thus making it highly practical and very stable to use.

A copy of the submitted code is attached with this report. (please see Appendix A)

I will be very happy to provide any additional information you may need to evaluate the functionality of this code.


Algorithm
Check for hardware devices. Proceed if available and ready or else give an error message and exit program
Check for input file. Proceed if available or else give a warning message and terminate program.
Allocate memory for the first member in the linked list.
Scan for a record line by line and store in an array.
Check character by character in the scanned array until character is a space or a comma and finally till a new line character (/n)
Scanned character is valid. (ie not &, %, # etc)
No
Yes
Character is comma, space or new line
No
Yes
Copy character into temp array
First time temporary array is created, the contents is copied into the first member, second time into the second member, third time into id and finally into pin member in the structure. Any white space is ignored
Allocate memory for the next structure in the linked list if sufficient memory available. Need to free memory after end of last record since memory is allocated before each structure is created
End of records
Asks for employee id
Get input from keypad
Checks if id exists in input file.
Id entered
No
Yes
Asks for PIN
PIN entered
Check for Pin
Incorrect pin Counter ++
Counter < 3
Ask for PIN again for not more than 3 times.
Counter = 3
- Keypad disabled for 5 seconds.
Display message to security
Yes
No
Check if alarm activated
Correct
Displays status of employee and option menu. Employee can either sign out (1), change pin (2), activate alarm or do nothing so that the screen refreshes automatically
1
Option
Screen refreshed
Alarm
2
- Display “You’re out” message.
Ask for new PIN twice. Change if entered pins match. Else display warning message.
PIN changed
Yes
Display message to Security
No