

시작화면 UI 구현

```
1 #!/bin/bash
2 clear
3 chmod 600 Profile.txt
4
5 JOIN="\e[44;1m JOIN \e[0m"
6 SIGN_IN="\e[44;1m SIGN IN \e[0m"
7 EXIT="\e[44;1m EXIT \e[0m"
8 SIGN_OUT="\e[44;1m SIGN OUT \e[0m"
9 SELECTED=0
10
11
12 makeUI(){
13     echo -e "\n\n\n\n\n\n\n\n"
14     echo "
15     echo "
16     echo "
17     echo "
18     echo "
19     echo "
20     echo "
21     echo "
22     echo "
23     echo "
24     echo "
25     echo "
26     echo "
27     echo "
28     echo -e "                $JOIN                $SIGN_IN                "
29     echo ""
30     echo -e "                $EXIT                $SIGN_OUT                "
31     echo -e "\n\n"
32 }
```

Line 3 : 사용자의 정보를 입력받기 위해 Profile.txt를 만들었다. 사용자만 접근이 가능하게 권한을 주었다

Line 5~8 : 글자배경이 파란색인 변수들 설정. (파란색은 44)

Line 9 : 방향키를 입력했을 때 선택한 정보를 선택하기 위한 SELECTED 변수 초기값은 0

Line 12 ~ 32 : 기본 UI는 방향키를 입력할때마다 새로 호출을 해줘야하기에 함수로 만들었다.

```
34 input_key(){
35     read -s -n 1 INPUT
36     if [ "$INPUT" = "^[" ];then
37         read -s -n 1 a
38         if [ "$a" = "[" ];then
39             read -s -n 1 b
40             INPUT="$INPUT$a$b";
41         fi
42     fi
43     unset a;unset b;
44 }
```

원래의 구현은 "read -s -n 3 INPUT" 이었지만 만약 사용자가 3바이트가 아닌 키를 입력했을 경우 오류가 생겼다. 따라서 조건문을 걸어서 하나씩 체크해보는 방법밖에는 없었다. 강의자료에 나온 방식을 토대로 구현을 하였다. (옵션에서 -s 는 시크릿모드)

시작화면 키보드 입력 시

```
46 while true;
47 do
48     makeUI
49     input_key
50     if [ $SELECTED = "0" ];then
51         JOIN="\e[41;1m JOIN \e[0m";SELECTED=1
52     else
53         if [ $SELECTED = "1" ];then
54             JOIN="\e[44;1m JOIN \e[0m"
55             if [ "$INPUT" = "^[A" ] || [ "$INPUT" = "^[B" ];then
56                 EXIT="\e[41;1m EXIT \e[0m";SELECTED=3
57             elif [ "$INPUT" = "^[C" ] || [ "$INPUT" = "^[D" ];then
58                 SIGN_IN="\e[41;1m SIGN IN \e[0m";SELECTED=2
59             elif [ $INPUT = "" ];then
60                 clear
61                 chmod 755 JOIN.sh
62                 ./JOIN.sh
63                 SELECTED=0
64             else
65                 JOIN="\e[41;1m JOIN \e[0m"
66             fi
```

사용자가 EXIT에서 enter를 치는 경우까지 무한루프를 돌려야 한다.

while문이 실행될 때마다 UI와 input을 받는다.

SELECTED의 값이 0일 경우 JOIN의 배경을 빨간색으로 바꾸고 SELECTED를 1로 초기화하였다.

만약 그렇지 않을 경우 SELECTED에 따라 방향키를 입력했을 때의 변화를 구현하였다.

Enter 또는 상하좌우를 입력했을 경우를 제외하고는 배경색이 빨간색이 나오도록 구현을 하였고

Enter 또는 상하좌우를 입력했을 경우에는 배경색이 다시 파란색이 나오도록 구현을 하였다.

만약 Enter를 입력했을 경우 만들어 놓은 JOIN.sh의 파일의 실행권한을 755로 주고 실행한다.

```
98         elif [ $INPUT = "" ];then
99             clear
100             chmod 755 SIGN_OUT.sh
101             ./SIGN_OUT.sh
102         else
103             SIGN_OUT="\e[41;1m SIGN OUT \e[0m"
104         fi
105     fi
106 fi
107 clear
108 done
```

while문이 끝나기 전에는 항상 clear로 화면을 지운 후 UI를 출력할 수 있게 설정하였다.

이외에 다른 코드는 JOIN과 같으므로 보고서에는 생략하였습니다.

SIGN IN UI 구현

```
1  #!/bin/bash
2
3  MyId="ID"
4  MyPw="PW"
5  ID="\e[44;1m          $MyId          \e[0m"
6  Duplicate_check="\e[44;1m Duplicate check \e[0m"
7  PW="\e[44;1m          $MyPw          \e[0m"
8  SIGN_IN="\e[44;1m SIGN IN \e[0m"
9  EXIT="\e[44;1m EXIT \e[0m"
10 SELECTED=0
11
12
13 makeUI(){
14     echo -e "\n\n\n\n\n\n\n\n\n\n"
15     echo " "
16     echo " "
17     echo " "
18     echo " "
19     echo " "
20     echo " "
21     echo " "
22     echo -e " "          $ID    $Duplicate_check
23     echo " "
24     echo -e " "          $PW
25     echo " "
26     echo -e " "          $SIGN_IN    $EXIT
27     echo -e "\n\n\n\n\n"
28 }
```

변수들은 "시작화면 UI"에서와 다르게 MyId와 MyPw를 추가하였다.

UI는 "시작화면UI"에서 만든 것과 동일하게 구현을 하였다.

```
42 while true;
43 do
44     makeUI
45     input_key
46     if [ $SELECTED = "0" ];then
47         ID="\e[41;1m          $MyId          \e[0m";SELECTED=1
48     else
49         if [ $SELECTED = "1" ];then
50             ID="\e[44;1m          $MyId          \e[0m"
51             if [ "$INPUT" = "^[A" ];then
52                 SIGN_IN="\e[41;1m SIGN IN \e[0m";SELECTED=4
53             elif [ "$INPUT" = "^[B" ];then
54                 PW="\e[41;1m          $MyPw          \e[0m";SELECTED=3
55             elif [ "$INPUT" = "^[C" ] || [ "$INPUT" = "^[D" ];then
56                 Duplicate_check="\e[41;1m Duplicate check \e[0m";SELECTED=2
57             elif [ $INPUT = "" ];then
58                 read -p "ID를 입력하시오 : " MyId
59                 ID="\e[44;1m          $MyId          \e[0m";SELECTED=1
60             else
61                 ID="\e[41;1m          $MyId          \e[0m"
62             fi
63         fi
64     fi
65 done
```

시작화면UI에서 구현한 것과 동일하게 구현을 하였고 차이점은 enter를 눌렀을 경우이다.

아이디를 입력을 받고 SELECTED는 자기 자신을 선택하게 구현을 하였다.

이때 배경색은 파란색이 나오도록 구현을 하였다.

SIGN IN에서Duplicate check의 구현과 Profile.txt의 형식.

```
63     elif [ $SELECTED = "2" ];then
64         Duplicate_check="\e[44;1m Duplicate check \e[0m"
65         if [ "$INPUT" = "^[A" ] || [ "$INPUT" = "^[B" ];then
66             EXIT="\e[41;1m EXIT \e[0m";SELECTED=5
67         elif [ "$INPUT" = "^[C" ] || [ "$INPUT" = "^[D" ];then
68             ID="\e[41;1m $MyId \e[0m";SELECTED=1
69         elif [ "$INPUT" = "" ];then
70             profile=`cat ./Profile.txt | grep -w "$MyId"`
71             id=$(echo $profile | cut -d ' ' -f 1)
72             cutId=1
73             while [ "$id" != "" ]
74             do
75                 if [ "$id" = "$MyId" ];then
76                     break
77                 fi
78                 cutId=`expr $cutId + 4`
79                 id=$(echo $profile | cut -d ' ' -f $cutId)
80             done
81             if [ "$id" = "$MyId" ];then
82                 Duplicate_check="\e[44;1m 같은 ID 존재 \e[0m"
83                 clear
84                 makeUI
85                 read -s a
86                 Duplicate_check="\e[41;1m Duplicate check \e[0m"
87             else
88                 Duplicate_check="\e[44;1m 회원 가입 가능 \e[0m"
89                 clear
90                 makeUI
91                 read -s a
92                 Duplicate_check="\e[41;1m Duplicate check \e[0m"
93             fi
94         else
95             Duplicate_check="\e[41;1m Duplicate check \e[0m"
96         fi
```

Profile.txt + (~/.2018603020) - VIM 96x41				
1	.ID	.PW	.승수	.패수
2	123	123	0	0

Line 70 : `cat 파일 | grep -w "단어"` 형식으로 단어가 들어간 모든 라인에 profile은 입력을 받는다.

grep -w 은 단어를 기준으로 잘라서 더욱 정교한 값을 가질 수 있다.

Line 71 : id는 profile 변수에 대한 값을 추출 받아야 하므로 echo를 통해 첫번째 단어를 받았다

ID만 계속확인하기 위하여 cutId 변수를 추가하였다.

Id가 공백이 아닐때까지 while문을 통해 실행하였다.

Profile.txt의 형식이 한 라인에 4개의 단어가 있다 따라서 다음 Id는 4칸 떨어진 곳에 있으므로 cutId의 값을 4씩 추가해서 ID의 존재여부를 확인하였다.

만약 ID가 존재한다면 Duplicate_check 변수를 바꾸고 UI를 출력하였다.

이때 아무키나 입력을 하면 다시 원래 형식으로 작동할 수 있게 설정하였다.

SIGN IN 에서의 SIGN IN 구현

```
123         while [ "$sid" != "" ]
124         do
125             if [ "$sid" = "$MyId" ];then
126                 break
127             fi
128             cutId=`expr $cutId + 4`
129             id=$(echo $profile | cut -d ' ' -f $cutId)
130         done
131         if [ "$sid" = "$MyId" ];then
132             break
133         else
134             echo -e "$MyId\t\t$MyPw\t\t0\t\t0" >> Profile.txt
135             break
136         fi
```

다른 부분들은 Duplicated check와 같으므로 생략을 하였다.

만약 아이디가 같다면 "시작화면 UI"로 돌아가도록 구현을 하였다.

그렇지 않다면 Profile.txt에 "아이디 패스워드 0 0" 을 추가하였다.

초기 승패는 0 이기에 0을 2개 넣어주었다.

cat [파일] | grep -w "123" 에 추가적인 내용

Profile.txt가 다음과 같다고 하자

1	.ID	.PW	.승수	.패수
2	123	123	0	0
3	1234	123	0	0
4	12345	123	0	0

```
njswindells@ubuntu:~/2018603020$ cat Profile.txt | grep -w "123"
123      123      0      0
1234     123      0      0
12345    123      0      0
```

같은 아이디는 존재할 수 없지만 같은 비밀번호는 존재할 수 있다. 그렇기에

첫번째 ID에서 4칸씩 옮겨가며 알맞은 id가 있는지 확인하는 과정이 꼭 필요하다 따라서

정확성을 위해 ID를 하나씩 일일이 체크할 수 밖에 없기에 저러한 방식으로 구현을 하였다.

SIGN OUT UI 구현

```
4 MyId="ID"
5 MyPw="PW"
6 ID="\e[44;1m          $MyId          \e[0m"
7 PW="\e[44;1m          $MyPw          \e[0m"
8 SIGN_OUT="\e[44;1m SIGN OUT \e[0m"
9 EXIT="\e[44;1m EXIT \e[0m"
10 SELECTED=0
11
12
13 makeUI(){
14     echo -e "\n\n\n\n\n\n\n\n\n"
15     echo "
16     echo "
17     echo "
18     echo "
19     echo "
20     echo "
21     echo "
22     echo -e "                                $ID"
23     echo ""
24     echo -e "                                $PW"
25     echo ""
26     echo -e "                                $SIGN_OUT $EXIT"
27     echo -e "\n\n\n\n"
28 }
```

"SIGN IN UI"와 동일한 변수를 설정하였다. UI도 마찬가지로 설정을 하였다.

SIGN OUT 에서의 SIGN OUT 구현.

```
85         elif [ "$INPUT" = "" ];then
86             profile=`cat ./Profile.txt | grep -w "$MyId"`
87             id=$(echo $profile | cut -d ' ' -f 1)
88             pw=$(echo $profile | cut -d ' ' -f 2)
89             cutId=1;cutPw=2
90             while [ "$id" != "" ]
91             do
92                 if [ "$id" = "$MyId" ];then
93                     break
94                 fi
95                 cutId=`expr $cutId + 4`;cutPw=`expr $cutPw + 4`
96                 id=$(echo $profile | cut -d ' ' -f $cutId)
97                 pw=$(echo $profile | cut -d ' ' -f $cutPw)
98             done
99             profile="$id $pw"
100             if [ "$id" = "$MyId" ] && [ "$pw" = "$MyPw" ];then
101                 sed -i "$profile/d" Profile.txt
102                 clear;break
103             else
104                 clear;break
105             fi
```

SIGN OUT에서는 ID만 동일한지 확인하면 안되고 PW 또한 동일한지 확인하는 과정이 필요하다.

따라서 cutPw 변수를 추가하여 PW 또한 같은지 확인을 하였다.

이때 profile에 변수를 삭제해야 하는 ID와 PW만으로 설정을 해주었다.


Line 101 : sed는 vi처럼 원본에 실시간 편집이 아니라 명령어형태로 파일을 편집할 수 있다.

-i 옵션은 그 파일 자체를 편집한다는 의미이다.

/\$profile/d 의 d는 삭제한다는 의미로 txt파일에서 profile이 들어간 라인을 삭제한다.

JOIN.sh 의 UI 구현

```
13 UI1P(){
14     echo -e "\n\n\n\n\n\n\n\n"
15     echo "
16     echo "
17     echo "
18     echo "
19     echo "
20     echo "
21     echo "
22     echo -e "
23     echo "
24     echo -e "
25     echo -e "\n"
26     echo -e "
27     echo -e "\n\n\n\n"
28 }
29 UI2P(){
30     echo -e "\n\n\n\n\n\n\n\n"
31     echo "
32     echo "
33     echo "
34     echo "
35     echo "
36     echo "
37     echo "
38     echo -e "
39     echo "
40     echo -e "
41     echo -e "\n"
42     echo -e "
43     echo -e "\n\n\n\n"
44 }
45 UISUCCESS(){
46     echo -e "\n\n\n\n\n\n\n\n"
47     echo "
48     echo "
49     echo "
50     echo "
51     echo "
52     echo -e "\n\n\n\n\n\n\n\n"
53
54 }
```



마찬가지로 여러번 호출을 해야하기에 함수로 구현을 하였다.

하나의 쉘파일에서 구현을 하였기에 1p에 대한 함수와 2p에 대한 함수 2개를 구성하여 설정을 하였다.

이때 같은 코드들이 사용이 많이 되지만 따로 함수를 만들어서 재사용하는건 불필요하다

생각해 1p와 2p에 대한 함수를 2개 만들었다.

JOIN 1P 의 enter

```
214         elif [ $INPUT = "" ];then
215             profile=`cat ./Profile.txt | grep -w "$MyId"`
216             id=$(echo $profile | cut -d ' ' -f 1)
217             pw=$(echo $profile | cut -d ' ' -f 2)
218             cutId=1;cutPw=2
219             while [ "$id" != "" ]
220             do
221                 if [ "$id" = "$MyId" ];then
222                     break
223                 fi
224                 cutId=`expr $cutId + 4`;cutPw=`expr $cutPw + 4`
225                 id=$(echo $profile | cut -d ' ' -f $cutId)
226                 pw=$(echo $profile | cut -d ' ' -f $cutPw)
227             done
228             if [ "$id" = "$MyId" ] && [ "$pw" = "$MyPw" ];then
229                 export LOGIN_1P=$id
230                 clear
231                 LOGIN2P
232                 break
233             else
234                 clear;break
235             fi
236         else
237             LOGIN="\e[41;1m  LOGIN  \e[0m"
238         fi
```

당연히 ID와 PW 둘 다 확인을 해주어 정확하게 비교를 하였다.

이때 아이디와 패스워드가 일치한다면 export를 이용해 1P의 ID를 환경변수로 설정을 하였다.

그 후 2P의 UI가 나오게 설정을 하였고 2P의 UI가 끝나면 1P도 끝나도록 설정을 하였다.

1P의 enter를 눌러 2P가 실행됐을 경우.

```
68 LOGIN2P(){
69     MyId="ID"
70     MyPw="PW"
71     ID="\e[44;1m          $MyId          \e[0m"
72     PW="\e[44;1m          $MyPw          \e[0m"
73     LOGIN="\e[44;1m  LOGIN  \e[0m"
74     EXIT="\e[44;1m  EXIT   \e[0m"
75     SELECTED=0
76     while true;
77     do
```

변수들을 처음과 같이 초기화해준다. 그렇게 하지 않으면 1P에서 사용했던 ID와 PW가

남아서 UI가 깔끔하게 설정이 안되기에 초기화를 해주었다.

2P의 LOGIN 에서 enter를 눌렀을 때 구현

```
119         elif [ $INPUT = "" ];then
120             profile=`cat ./Profile.txt | grep -w "$MyId"`
121             id=$(echo $profile | cut -d ' ' -f 1)
122             pw=$(echo $profile | cut -d ' ' -f 2)
123             cutId=1;cutPw=2
124             while [ "$id" != "" ]
125             do
126                 if [ "$id" = "$MyId" ];then
127                     break
128                 fi
129                 cutId=`expr $cutId + 4`;cutPw=`expr $cutPw + 4`
130                 id=$(echo $profile | cut -d ' ' -f $cutId)
131                 pw=$(echo $profile | cut -d ' ' -f $cutPw)
132             done
133             if [ "$id" = "$MyId" ] && [ "$pw" = "$MyPw" ];then
134                 if [ "$id" != "$LOGIN_1P" ];then
135                     export LOGIN_2P=$id
136                     clear
137                     MyId="ID";MyPw="PW"
138                     UISUCCESS
139                     read -s a
140                     break
141                 else
142                     clear; break
143                 fi
144             fi
145         fi
146     fi
147 fi
```

아이디와 패스워드의 구분은 위와 동일하다.

만약 아이디와 패스워드가 같고 1P의 아이디와 다르다면

export로 2P의 아이디를 환경변수로 설정을 하고 UISUCCESS를 띄운다.

이때 아무키나 입력을 하면 시작 화면으로 돌아오도록 설정을 하였다.

만약 3개의 조건 중 하나라도 충족을 하지 않는 경우 시작화면의 UI로 돌아온다.

```
258 main(){
259     LOGIN1P
260 }
261 main
```

마지막 줄엔 큰 의미는 없지만 main을 따로 만들어서 실행하도록 구현을 하였다.

느낀점



id나 pw를 길게 입력했을 때 파란배경의 범위를 조정하기가 까다로웠다

바이트단위로 쪼개서 하는 방법을 잘 알지 못하여 우선 저렇게 구현을 하였다.

이 부분이 가장 아쉽다.

Profile.txt를 이용하여 사용자 로그를 남긴 이유.

만약 export를 사용해 환경변수로 남기게 된다면 셸을 로그아웃 하거나 다른 사용자가 사용했을 때 id가 나오지 않는다 그런 사항은 id의 의미가 없다고 생각을 하였다.

그렇기에 따로 txt파일을 만들어 id,pw,승,패 를 남기게 되었다.

물론 root 계정일 경우 /etc/bash.bashrc 에 변수를 넣어 전체 환경에서 환경변수를 사용할 수 있지만 일반 사용자계정에서 사용하므로 txt 파일에 저장하는 것이 더 좋다고 생각을 하였다.