

1. 개요

1) xv6 설치 및 컴파일

- xv6 다운로드
- QEMU 다운로드 및 설치
- xv6 컴파일 및 실행

2_가) helloworld 셸 프로그램

- “Hello World XV6”를 출력하는 helloworld.c 프로그램 작성
- Makefile의 UPLOGS에 _helloworld\ 추가, EXTRA에 helloworld.c 추가
- xv6 실행 후 helloworld 명령어 실행

2_나) hcat 셸 프로그램

- 파일의 첫 번째 <n>행을 터미널에 출력하는 hcat.c 프로그램 작성
- 기존의 cat.c 파일을 복사
- 출력할 행 수를 저장할 전역변수 추가, 명령어 호출시 행 수를 받아 integer형으로 변환 후 전역변수에 저장
- void cat(int fd) 함수 부분 수정
- * write를 1byte 씩 진행시키며 값을 확인하고, 값이 '\n'일 경우 count를 증가시켜 행 수를 기억함
- * 명령어 호출시 받은 행 수와 현재 행 수가 같아지면 write를 멈춤
- Makefile의 UPLOGS에 _hcat\ 추가, EXTRA에 hcat.c 추가
- xv6 실행 후 hcat 명령어 실행

3) 부팅 시 Username: root, Password: 1234를 입력하여 로그인 구현

int.c

- 부팅 시 가장 먼저 실행하는 프로그램
- ssu_login을 호출하여 로그인 프로세스를 하도록 수정

ssu_login.c

- Username과 Password를 입력으로 받음 (root, 1234)
- list.txt 파일을 읽어 입력 받은 Username, Password가 해당 파일에 존재하는지 확인
- Username과 Password가 존재한다면, ssu_login에서 shell 프로그램(sh)을 fork-exec으로 호출
- * main에서 함수 check_idpw() 실행
- * check_idpw()는 get_user_list()를 먼저 실행
- * get_user_list()는 list.txt의 유저 정보를 읽고, 이를 2진 배열 전역 변수에 형식에 맞춰 저장함
- * check_idpw()는 Username과 Password를 입력 받고, 반복문을 사용하여 유저 정보를 저장한 전역변수에 존재하는지 확인
- * 존재한다면 shell 프로그램(sh)을 fork-exec으로 호출, 존재하지 않으면 다시 입력 받음.
- list.txt 파일은 [Username][Password] 형식을 구성
- Makefile의 UPLOGS에 _ssu_login\ 추가, EXTRA에 ssu_login.c, list.txt 추가, fs.img에 list.txt 추가

2. 결과

1) xv6 설치 및 컴파일

```
kibum@kibum-VirtualBox: ~/xv6-public
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ls
.          1 1 512
..         1 1 512
README    2 2 2286
list.txt  2 3 20
cat        2 4 10244
echo       2 5 15100
forktest   2 6 9404
grep       2 7 18460
init       2 8 15080
kill       2 9 15120
ln         2 10 14980
ls         2 11 17612
mkdir      2 12 15224
rm         2 13 15204
sh         2 14 27044
stressfs   2 15 10116
usertests  2 16 67220
wc         2 17 16980
zombie     2 18 14792
helloworld 2 19 14868
hcat       2 20 16600
console    3 21 0
$
```

2_가) helloworld 쉘 프로그램

```
kibum@kibum-VirtualBox: ~/xv6-public
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ helloworld
Hello World xv6
$
```

2_나) hcat 쉘 프로그램

```
kibum@kibum-VirtualBox: ~/xv6-public
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ hcat 5 README
NOTE: we have stopped maintaining the x86 version of xv6, and switched
our efforts to the RISC-V version
(https://github.com/mit-pdos/xv6-riscv.git)

xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix
$
```

3) 부팅 시 Username: root, Password: 1234를 입력하여 로그인 구현

```
kibum@kibum-VirtualBox: ~/xv6-public
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting login
Username: root
Password: 1234
$
```

3. 소스코드

1) helloworld.c

```
1 #include "types.h"
2 #include "stat.h"
3 #include "user.h"
4
5 int main(int argc, char **argv)
6 {
7     printf(1, "Hello World XV6\n");
8     exit();
9 }
```

2) hcat.c

```
1 #include "types.h"
2 #include "stat.h"
3 #include "user.h"
4
5 char buf[512];
6 int row;
7
8 void
9 cat(int fd)
10 {
11     int n;
12     int row_count = 0;
13     int byte_count = 0;
14
15     while((n = read(fd, buf, sizeof(buf))) > 0) {
16         while(row_count != row){
17             write(1, buf+byte_count, 1);
18             if(buf[byte_count] == '\n'){
19                 row_count++;
20             }
21             byte_count++;
22         }
23         exit();
24     }
25     if(n < 0){
26         printf(1, "hcat: read error\n");
27         exit();
28     }
29 }
30
```

```
31 int
32 main(int argc, char *argv[])
33 {
34     int fd, i;
35
36     if(argc <= 2){
37         cat(0);
38         exit();
39     }
40
41     row = atoi(argv[1]);
42     if(row < 0){
43         printf(1, "hcat: the number of rows");
44         exit();
45     }
46
47     for(i = 2; i < argc; i++){
48         if((fd = open(argv[i], 0)) < 0){
49             printf(1, "hcat: cannot open %s\n", argv[i]);
50             exit();
51         }
52         cat(fd);
53         close(fd);
54     }
55     exit();
56 }
```

3) ssu_login.c

```

1 #include "types.h"
2 #include "stat.h"
3 #include "user.h"
4 #include "fcntl.h"
5
6 char userID[16][32];
7 char pwdID[16][32];
8
9 char cur_user[32];
10 char cur_pwd[32];
11
12 void get_user_list(){
13
14     int fd;
15     fd = open("list.txt", O_RDONLY);
16
17     int n;
18     char buf[2048];
19
20     n = read(fd, buf, sizeof(buf));
21
22     int byte_count = 0;
23     int count = 0;
24
25     for(int i=0; i<16; i++){
26
27         if(byte_count<n){
28             while(buf[byte_count] != ' '){
29                 userID[i][count] = buf[byte_count];
30                 byte_count++;
31                 count++;
32             }
33             count = 0;
34             byte_count++;
35
36             while(buf[byte_count] != '\n'){
37                 pwdID[i][count] = buf[byte_count];
38                 byte_count++;
39                 count++;
40             }
41             count = 0;
42             byte_count++;
43         }
44     }
45     close(fd);
46     return;
47 }
48
49 int check_idpw(){
50
51     get_user_list();
52
53     while(1){
54
55         printf(1, "Username: ");
56         gets(cur_user, sizeof(cur_user));
57         cur_user[strlen(cur_user) - 1] = '\0';
58
59         printf(1, "Password: ");
60         gets(cur_pwd, sizeof(cur_pwd));
61         cur_pwd[strlen(cur_pwd) - 1] = '\0';
62
63         for(int i=0; i<16; i++){
64             if(strcmp(cur_user, userID[i]) == 0 && strcmp(cur_pwd, pwdID[i]) == 0){
65                 char *argv[] = {"sh", 0};
66                 exec("sh", argv);
67                 printf(1, "ssu_login: exec sh failed\n");
68                 exit();
69             }
70         }
71     }
72
73     return 0;
74 }
75
76 int main(int argc, char *argv[])
77 {
78     check_idpw();
79     return 0;
80 }

```

4) init.c

```
1 // init: The initial user-level program
2
3 #include "types.h"
4 #include "stat.h"
5 #include "user.h"
6 #include "fcntl.h"
7
8 char *argv[] = { "sh", 0 };
9
10 int
11 main(void)
12 {
13     int pid, wpid;
14
15     if(open("console", O_RDWR) < 0){
16         mknod("console", 1, 1);
17         open("console", O_RDWR);
18     }
19     dup(0); // stdout
20     dup(0); // stderr
21
22     for(;;){
23         printf(1, "init: starting login\n");
24         //printf(1, "init: starting sh\n");
25         pid = fork();
26         if(pid < 0){
27             printf(1, "init: fork failed\n");
28             exit();
29         }
30         if(pid == 0){
31             exec("ssu_login", argv);
32             printf(1, "init:exec login failed\n");
33             //exec("sh", argv);
34             //printf(1, "init: exec sh failed\n");
35             exit();
36         }
37         while((wpid=wait()) >= 0 && wpid != pid)
38             printf(1, "zombie!\n");
39     }
40 }
```

5) Makefile

```
168 UPROGS=\
169     _cat\
170     _echo\
171     _forktest\
172     _grep\
173     _init\
174     _kill\
175     _ln\
176     _ls\
177     _mkdir\
178     _rm\
179     _sh\
180     _stressfs\
181     _usertests\
182     _wc\
183     _zombie\
184     _helloworld\
185     _hcat\
186     _ssu_login\
187
188 fs.img: mkfs README list.txt $(UPROGS)
189     ./mkfs fs.img README list.txt $(UPROGS)
```

```
253 EXTRA=\
254     mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
255     ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\
256     printf.c umalloc.c helloworld.c hcat.c ssu_login.c\
257     README list.txt dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
258     .gdbinit.tmpl gdbutil\
```