글로벌미디어학부 20182705 빙기범 비밀번호:kb9978

1. 개요

1) xv6 설치 및 컴파일

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

2_가) helloworld 쉘 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 쉘 프로그램

- 파일의 첫 번째 <n>행을 터미널에 출력하는 hcat.c 프로그램 작성
- 기존의 cat.c 파일을 복사
- 출력할 행 수를 저장할 전역번수 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

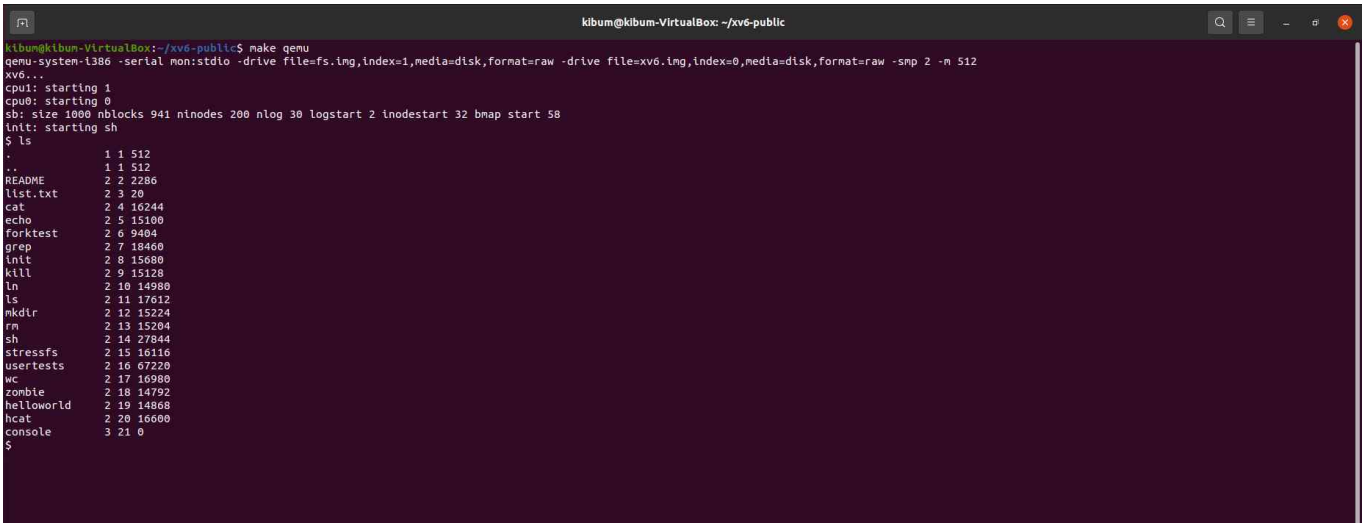
3) 부팅 시 Username: root, Password: 1234를 입력하여 로그인 구현

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

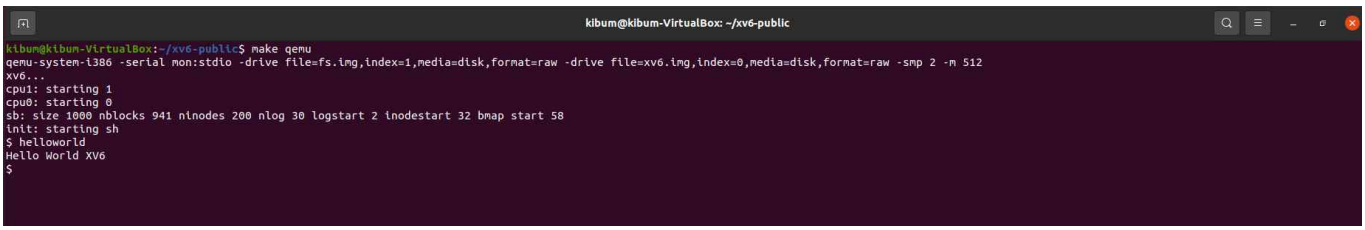
ssu_login.c
- Username과 Password를 입력으로 받음 (root, 1234)
- list.txt 파일을 읽어 입력 받은 Username, Password가 해당 파일에 존재하는지 확인
- Username과 Password과 존재한다면, ssu_login에서 shell 프로그램(sh)을 fork-exec으로 호출
* main에서 함수 check_idpw() 실행
* check_idpw()는 get_user_list()를 먼저 실행
* get_user_list()는 list.txt의 유저 정보를 읽고, 이를 2진 배열 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

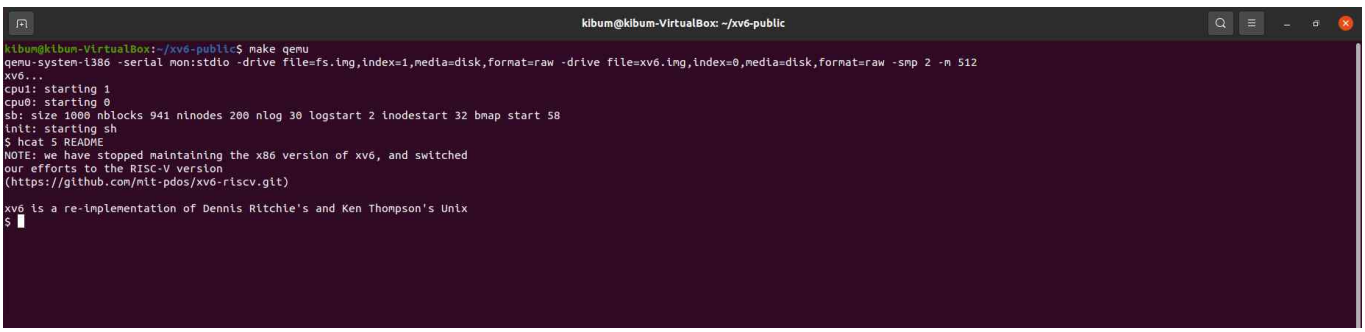
1) xv6 설치 및 컴파일



2_가) helloworld 쉘 프로그램



2_나) hcat 쉘 프로그램



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

3. 소스코드

1) helloworld.c

```c
#include "types.h"
#include "stat.h"
#include "user.h"

int main(int argc, char **argv)
{
     printf(1, "Hello World XV6\n");
     exit();
}
```

2) hcat.c

```c
#include "types.h"
#include "stat.h"
#include "user.h"

char buf[512];
int row;

void
cat(int fd)
{
  int n;
  int row_count = 0;
  int byte_count = 0;

  while((n = read(fd, buf, sizeof(buf))) > 0) {
      while(row_count != row){
        write(1, buf+byte_count, 1);
        if(buf[byte_count] == '\n'){
            row_count++;
        }
        byte_count++;
      }
      exit();
  }
  if(n < 0){
    printf(1, "hcat: read error\n");
    exit();
  }
}

int
main(int argc, char *argv[])
{
  int fd, i;

  if(argc <= 2){
    cat(0);
    exit();
  }

  row = atoi(argv[1]);
  if(row < 0){
    printf(1, "hcat: the number of rows");
    exit();
  }

  for(i = 2; i < argc; i++){
    if((fd = open(argv[i], 0)) < 0){
      printf(1, "hcat: cannot open %s\n", argv[i]);
      exit();
    }
    cat(fd);
    close(fd);
  }
  exit();
}
```

3) ssu_login.c

```c
#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"

char userID[16][32];
char pwdID[16][32];

char cur_user[32];
char cur_pwd[32];

void get_user_list(){

    int fd;
    fd = open("list.txt", O_RDONLY);

    int n;
    char buf[2048];

    n = read(fd, buf, sizeof(buf));

    int byte_count = 0;
    int count = 0;

    for(int i=0; i<10; i++){

        if(byte_count<n){
            while(buf[byte_count] != ' '){
                userID[i][count] = buf[byte_count];
                byte_count++;
                count++;
            }
            count = 0;
            byte_count++;

            while(buf[byte_count] != '\n'){
                pwdID[i][count] = buf[byte_count];
                byte_count++;
                count++;
            }
            count = 0;
            byte_count++;
        }
    }
    close(fd);
    return;
}

int check_idpw(){

    get_user_list();

    while(1){

        printf(1, "Username: ");
        gets(cur_user, sizeof(cur_user));
        cur_user[strlen(cur_user) - 1] = '\0';

        printf(1, "Password: ");
        gets(cur_pwd, sizeof(cur_pwd));
        cur_pwd[strlen(cur_pwd) - 1] = '\0';

        for(int i=0; i<16; i++){
            if(strcmp(cur_user, userID[i]) == 0 && strcmp(cur_pwd, pwdID[i]) == 0){
                char *argv[] = {"sh", 0};
                exec("sh", argv);
                printf(1, "ssu_login: exec sh failed\n");
                exit();
            }
        }
    }

    return 0;
}

int main(int argc, char *argv[])
{
    check_idpw();
    return 0;
}
```

4) init.c

```c
// init: The initial user-level program

#include "types.h"
#include "stat.h"
#include "user.h"
#include "fcntl.h"

char *argv[] = { "sh", 0 };

int
main(void)
{
  int pid, wpid;

  if(open("console", O_RDWR) < 0){
    mknod("console", 1, 1);
    open("console", O_RDWR);
  }
  dup(0);  // stdout
  dup(0);  // stderr

  for(;;){
    printf(1, "init: starting login\n");
    //printf(1, "init: starting sh\n");
    pid = fork();
    if(pid < 0){
      printf(1, "init: fork failed\n");
      exit();
    }
    if(pid == 0){
      exec("ssu_login", argv);
      printf(1, "init:exec login failed\n");
      //exec("sh", argv);
      //printf(1, "init: exec sh failed\n");
      exit();
    }
    while((wpid=wait()) >= 0 && wpid != pid)
      printf(1, "zombie!\n");
  }
}
```

5) Makefile

```
UPROGS=\
    _cat\
    _echo\
    _forktest\
    _grep\
    _init\
    _kill\
    _ln\
    _ls\
    _mkdir\
    _rm\
    _sh\
    _stressfs\
    _usertests\
    _wc\
    _zombie\
    _helloworld\
    _hcat\
    _ssu_login\

fs.img: mkfs README list.txt $(UPROGS)
    ./mkfs fs.img README list.txt $(UPROGS)
```

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