

설계 과제 #1 : xv6 Tour

○ 과제 목표

- xv6 설치 및 컴파일
- xv6 셸 명령어 추가를 통한 “Hello Soongsil xv6” 출력

○ 기본 지식

- xv6
 - ✓ 미국 MIT에서 멀티프로세서 x86 및 RISC-V 시스템을 위해 개발한 교육용 운영체제
 - ✓ UNIX V6를 ANSI C 기반으로 구현
 - ✓ 리눅스나 BSD와 달리 xv6은 단순하지만 UNIX 운영체제의 중요 개념과 구성을 포함하고 있음
- Cross Compile 방법 학습
 - ✓ xv6에는 텍스트 편집기 또는 gcc 컴파일러가 없음. 따라서 본 과제에서는 자신의 리눅스 환경에서 xv6 프로그램 작성 및 컴파일 후, 생성된 실행파일을 xv6 상에서 수행함

○ 과제 내용

1. xv6 설치 및 컴파일

- ✓ xv6 다운로드

```
$ git clone https://github.com/mit-pdos/xv6-public
```

- ✓ QEMU 다운로드 및 설치

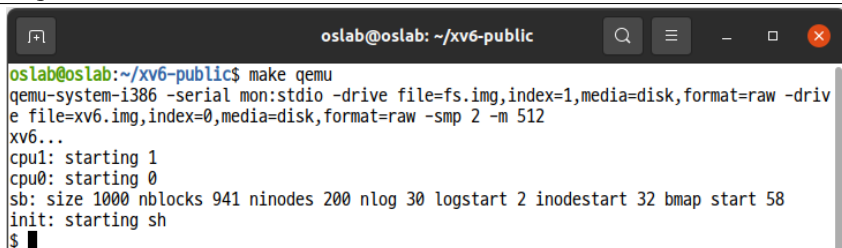
- xv6 운영체제는 자신의 컴퓨터에서 x86 하드웨어를 에뮬레이트 하는 QEMU x86 에뮬레이터에서 실행됨 (에뮬레이터 없이도 운용 가능하나, 커널 함수 등을 수정하기 위해 에뮬레이터 사용을 권장)

```
$ apt-get install qemu-kvm
```

- ✓ xv6 컴파일 및 실행

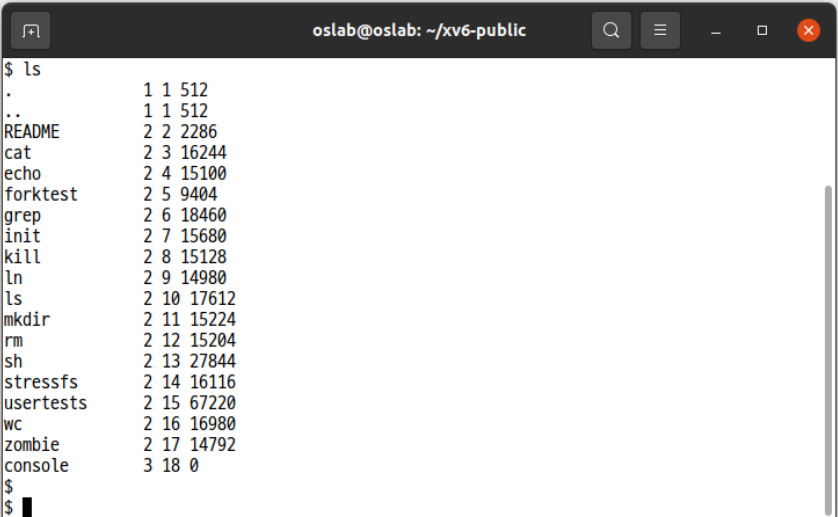
```
$ make  
$ make qemu
```

(예시 1). make qemu 실행 결과



```
oslab@oslab: ~/xv6-public  
oslab@oslab:~/xv6-public$ make qemu  
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive  
e 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  
$
```

(예시 2). ls 실행 결과



```
$ ls
.          1 1 512
..         1 1 512
README    2 2 2286
cat       2 3 16244
echo      2 4 15100
forktest  2 5 9404
grep      2 6 18460
init      2 7 15680
kill      2 8 15128
ln        2 9 14980
ls        2 10 17612
mkdir     2 11 15224
rm        2 12 15204
sh        2 13 27844
stressfs  2 14 16116
usertests 2 15 67220
wc        2 16 16980
zombie    2 17 14792
console   3 18 0
$
```

2. 셸 프로그램 작성

가. helloworld 셸 프로그램

- “Hello World xv6”를 출력하는 helloworld.c 프로그램 작성
- Makefile을 수정하여 helloworld.c 파일도 make 시 컴파일 되도록 변경
- xv6 실행 후 helloworld 명령어 실행

(예시 3). helloworld.c 파일 작성 예시

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

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

(예시 4). Makefile 수정 필요

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

(예시 5). Makefile 수정 필요

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

(예시 6). helloworld 실행 결과



```
oslab@oslab: ~/xv6-public
oslab@oslab:~/xv6-public$ make qemu
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
$
```

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
- void cat(int fd) 함수 부분 수정
- 위 helloworld 예제와 같이 Makefile을 수정하여 hcat.c 또한 컴파일 될 수 있도록 변경
- xv6 실행 후 hcat 명령어 실행

(예시 7). hcat.c cat() 함수 아래 예시를 바탕으로 수정

```
void
cat(int fd)
{
    int n;

    while((n = read(fd, buf, )) > 0) {

        if (write(1, buf, n) != n) {
            printf(1, "cat: write error\n");
            exit();
        }

    }
    if(n < 0){
        printf(1, "cat: read error\n");
        exit();
    }
}
```

(예시 8). hcat.c main 함수 아래 예시를 바탕으로 구현

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

    if(argc <= 1) {
        cat(0);
        exit();
    }

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

(예시 9). hcat 실행 결과

```
oslab@oslab: ~/xv6-public
oslab@oslab:~/xv6-public$ make qemu
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

- ✓ init.c
 - 부팅 시 가장 먼저 실행하는 프로그램
 - ssu_login을 호출하여 로그인 프로세스를 실행하도록 수정

(예시 10). init.c 수정 예시 (ssu_login 구현 필요)

```
printf(1, "init: starting login\n");
pid = fork();
if(pid < 0){
    printf(1, "init: fork failed\n");
    exit();
}
if(pid == 0) {
    exec("ssu_login", argv);
    //exec("sh", argv);
    printf(1, "init: exec login failed\n");
    exit();
}
```

- ✓ 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
 - list.txt 파일은 [Username] [Password] 형식으로 구성
 - Makefile 수정을 통해 list.txt 파일을 xv6에 추가

(예시 11). list.txt 구성 예시 (root 1234 / user 3456 은 Username Password로 각 한 쌍)

```
$ cat list.txt
root 1234
user 3456
```

(예시 12). ssu_login.c 아래 예시 바탕으로 구현

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

```

#include "fcntl.h"

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

void get_user_list() {
    int fd;

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

    for (i = 0; i < 10; i++) {
        // list.txt에서 Username, Password 정보를 userID, pwdID에 저장
    }
}

int check_idpw() {
    // 입력받은 Username, Password와 list.txt 비교
}

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

```

(예시 12). 부팅 시 로그인 실행 결과

```

oslab@oslab:~/xv6-public$ make qemu
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 star
t 58
init: starting login
Username: root
Password: 1234
$

```

○ 과제 제출 마감

- 2022년 09월 12일 (월) 23시 59분 59초까지 구글클래스룸으로 제출
- 보고서 (hwp, doc, docx 등으로 작성 - 총 3개의 프로그램이 수행된 결과 (캡처 등이 포함된) 와 소스코드 (helloworld.c, hcat.c, ssu_login.c, init.c, Makefile)를 제출해야 함. 또한 별도로 학생이 구현한 프로그램 있으면 함께 제출. (단, xv6 전체 소스코드는 제출 불필요.)

- 1일 지연 제출마다 30% 감점. 4일 지연 제출 시 0점 처리 (이하 모든 설계 과제 동일하게 적용)

○ 필수 사항 (설치 및 설명 등)

- 1, 2, 3

○ 배점 기준

- 1 : 30점

- 2 - 가 : 20점

- 2 - 나 : 20점

- 3 : 30점