

운영체제(가) 1차 과제

일어일문학과

20181755 이건희

2023년 9월

1. Development Environment

장치 사양

장치 이름	ThinkBook-Plus
프로세서	Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz
설치된 RAM	16.0GB(15.8GB 사용 가능)
장치 ID	████████████████████████████████████████
제품 ID	████████████████████████████████████████
시스템 종류	64비트 운영 체제, x64 기반 프로세서
펜 및 터치	이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.

복사

이 PC의 이름 바꾸기

Windows 사양

에디션	Windows 10 Home
버전	22H2
설치 날짜	2023-03-31
OS 빌드	19045.3448
경험	Windows Feature Experience Pack 10

```
kh@ThinkBook-Plus: ~$ lsb_release -a
LSB Version:    core-11.1.0ubuntu4-noarch
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.3 LTS
Release:        22.04
Codename:       jammy
kh@ThinkBook-Plus: ~$
```

그림 1 개발환경 정보

본론에 들어가기 앞서, 본 보고서를 작성한 학생인 이건희(이하 필자)는 기초적인 리눅스 사용 경험이 있기에 초기 환경 설정 과정 및 기본적인 명령어 사용법 등은 생략함을 미리 알린다.

xv6 개발은 POSIX 환경이 필연적으로 요구된다. 윈도우에서 POSIX 환경을 에뮬레이트 하려면 MSYS2, MinGW, Cygwin 등의 프로그램으로 가능하나, 이들은 여전히 에뮬레이트된 지원이기에 완벽하지 못하다.¹⁾

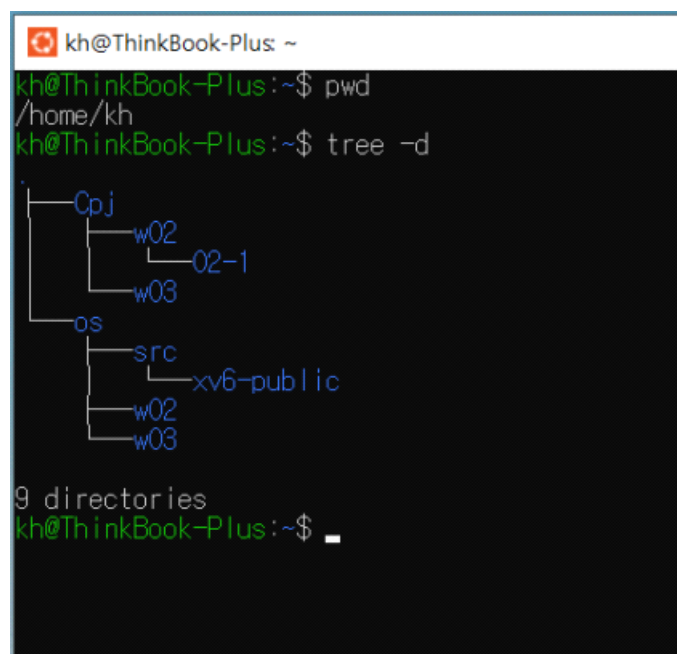
그림 1은 필자의 개발환경에 대한 간략한 정보를 담고 있다. 윈도우 환경에서 사용중이기에, WSL (Windows Subsystem for Linux) Version 2를 사용하여 개발환경을 구성했다. 배포판은 널리 사용되는 Ubuntu이고, 릴리즈는 22.04 LTS이다.

1) <https://www.msys2.org/wiki/MSYS2-introduction/#subsystems>

WSL2는 WSL1과 많은 부분에서 바뀌었는데, 이중 가장 큰 것은 실제 리눅스 커널이²⁾ Type 1(Bare Metal) Hypervisor인 MS Hyper-V³⁾에서 작동한다는 것이다.⁴⁾ 이로부터 오는 장점들은 아래와 같다.

- Type 2(Hosted) Hypervisor에서 실행한 것 보다 성능이 더 뛰어나다.⁵⁾
- VM이 알아서 관리 된다.
 - : WSL1과는 달리, WSL2는 백그라운드에서 시스템이 알아서 Linux VM을 관리하고 실행함
- 완전한 Linux syscall 호환성을 가진다.
 - : WSL1은 Linux의 syscall을 Windows API로 변환하는 구조라⁶⁾, 32비트 바이너리 실행이나 커널 의존적인 작업(드라이버 등)은 불가능했음

다시 말하자면 WSL2 이전의 WSL1이나 LXSS는 윈도우를 만든 제작사인 MS가 직접 만들어서 겹보기에 통합성이 좋게 느껴졌을 뿐, 근본적으로는 전 장의 POSIX 환경을 흉내 낸 프로그램들과 다를 바가 없는 것이다. WSL1을 사용해야 했으면 고민을 많이 했겠지만, WSL2는 망설일 이유가 없었다.



```
kh@ThinkBook-Plus: ~  
kh@ThinkBook-Plus:~$ pwd  
/home/kh  
kh@ThinkBook-Plus:~$ tree -d  
.  
├── Cpj  
│   ├── w02  
│   │   └── 02-1  
│   └── w03  
└── os  
    ├── src  
    │   └── xv6-public  
    ├── w02  
    └── w03  
  
9 directories  
kh@ThinkBook-Plus:~$
```

그림 2 디렉터리 구조

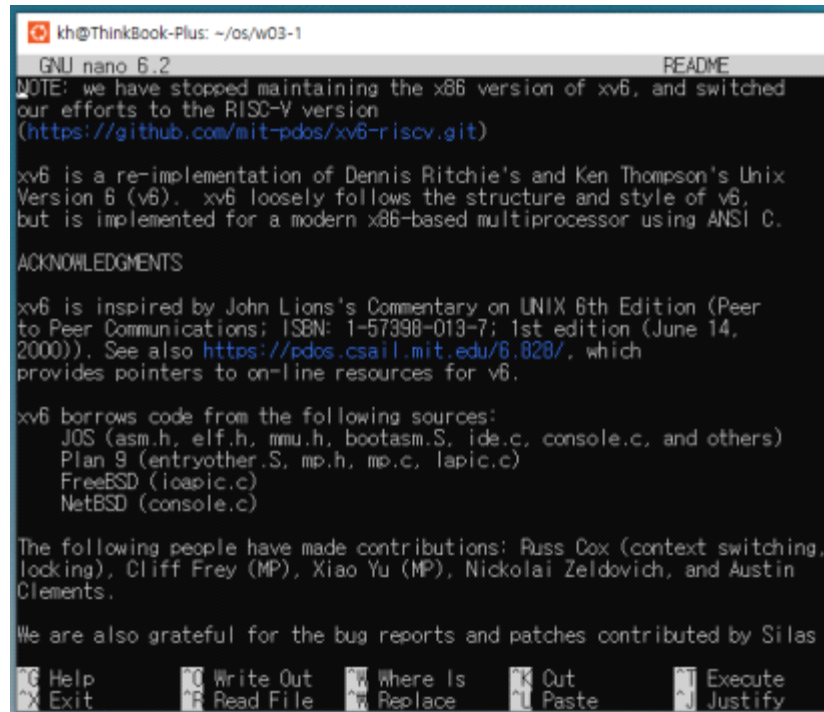
WSL의 홈 디렉터리 구조는 위와 같다. 한 주가 지날때마다 ~/os/w(주차번호)의 폴더에 XV6의 소스와 작업한 파일들이 저장될 예정이다.

수업 첫 시간에 코드 작성 및 수정시에 IDE는 사용하면 안된다는 교수님의 말씀이 있었다. 따라서 Ubuntu나 xv6에서 실행될 코드 작성 및 수정시에는 GNU nano를 사용할 것이다. Vim은 사용 자체가 익숙치가 않아 이번 학기 본 수업을 수강하면서 Vim에 익숙해지는 것을 목표중 하나로 삼아야겠다.

하지만 가독성 때문에 코드를 보기가 힘든 것은 사실인지라, 코드를 탐색할 때 만든 제한적으로 JetBrains의 CLion을 사용할 것 인데 이정도는 용인되었으면 좋겠다. (이마저도 불가능하다면 바로 말씀 부탁드립니다. 다음 보고서부터는 모든 작업을 Ubuntu상에서 텍스트에디터로만 진행 하겠습니다.)

2) <https://learn.microsoft.com/en-us/windows/wsl/compare-versions#comparing-wsl-1-and-wsl-2>
3) <https://www.redhat.com/ko/topics/virtualization/what-is-a-hypervisor>
4) <https://learn.microsoft.com/ko-kr/windows/wsl/faq#wsl-2>
5) <https://aws.amazon.com/ko/compare/the-difference-between-type-1-and-type-2-hypervisors/>
6) <https://learn.microsoft.com/en-us/previous-versions/windows/desktop/cmdline/wsl-architectural-overview>

2. Closer look at xv6



```
kh@ThinkBook-Plus: ~/os/w03-1
GNU nano 6.2                                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
Version 6 (v6).  xv6 loosely follows the structure and style of v6,
but is implemented for a modern x86-based multiprocessor using ANSI C.

ACKNOWLEDGMENTS

xv6 is inspired by John Lions's Commentary on UNIX 6th Edition (Peer
to Peer Communications; ISBN: 1-57398-013-7; 1st edition (June 14,
2000)).  See also https://pdos.csail.mit.edu/6.828/, which
provides pointers to on-line resources for v6.

xv6 borrows code from the following sources:
  JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)
  Plan 9 (entryother.S, mp.h, mp.c, lapic.c)
  FreeBSD (ioapic.c)
  NetBSD (console.c)

The following people have made contributions: Russ Cox (context switching,
locking), Cliff Frey (MP), Xiao Yu (MP), Nikolai Zeldovich, and Austin
Clements.

We are also grateful for the bug reports and patches contributed by Silas

G Help  O Write Out  W Where Is  K Out  T Execute
X Exit  R Read File  A Replace  L Paste  J Justify
```

그림 3 XV6의 README

xv6를 처음 접해보는 학생들이 대부분일 것이고, 필자 역시나 이에 포함된다. 참고할만한 매뉴얼이 필요하여 README 파일을 열어보니 참고할만한 온라인 리소스들을 제공하는 xv6를 만든 MIT의 링크를 알려준다.

이 파일의 첫줄에는 x86 아키텍처 버전의 유지보수를 중단하고 RISC-V 아키텍처 버전에 집중한다고 적혀있기도 하다. 링크로 들어가서 다운로드 한 매뉴얼⁷⁾ 역시나 RISC-V 아키텍처라는 전제하에 작성되어있으나, 기본적인 사용법 정도를 익히는데는 무리가 없기에 이번 학기 본 과목의 수강에 있어서 많이 참고할 것 같다.

어디에서 인용했는지 레퍼런스를 달아야 하기에, 본 보고서가 포함된 압축파일에 ‘book-riscv-rev3’ 문서를 같이 첨부하겠다. 추후 과제의 보고서 역시 필요할 경우 해당 문서⁸⁾의 참고한 페이지를 각주로 달 예정이다.

7) <https://pdos.csail.mit.edu/6.828/2023/xv6/book-riscv-rev3.pdf>

8) Russ Cox, Frans Kaashoek, and Robert Morris, xv6: a simple, Unix-like teaching operating system.

2-1. XV6 빌드 및 qemu를 사용하여 실행

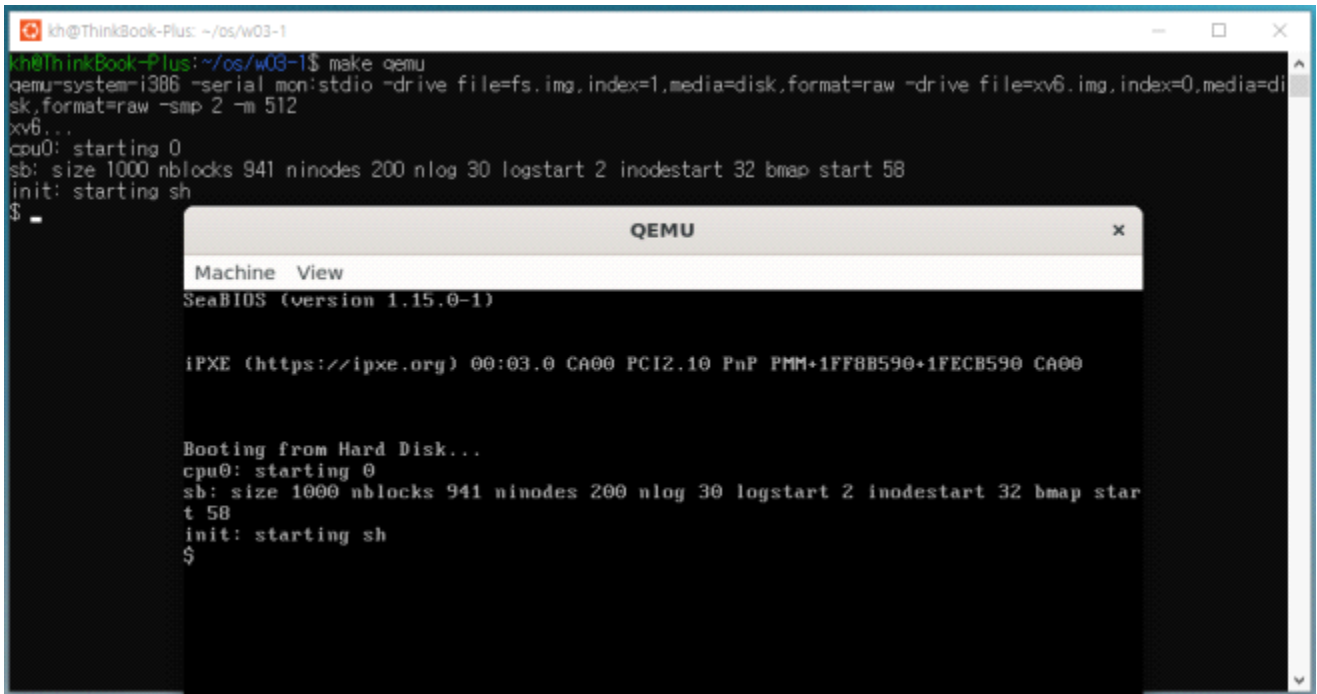


그림 4 빌드 및 실행

사실 xv6의 빌드 및 실행은 카이스트 OSLab의 가이드⁹⁾를 보고 2주차에 미리 혼자서 진행해보았다. 'make qemu-nox'로 빌드하면 터미널상에서 qemu가 바로 실행되어, 빌드 전후의 결과를 한 장의 스크린샷으로 담기 불편하기에 임의로 'make qemu'로 빌드하였다.

9) <https://oslab.kaist.ac.kr/xv6-install/>

2-2. xv6에 포함된 User Application 실행

```
kh@ThinkBook-Plus: ~/os/w03-1
SeaBIOS (version 1.15.0-1)

iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1FF8B4A0+1FECB4A0 CA00

Booting from Hard Disk..xv6...
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
cat       2 3 15440
echo      2 4 14320
forktest  2 5 8764
grep      2 6 18284
init      2 7 14940
kill      2 8 14408
ln        2 9 14304
ls        2 10 16872
mkdir     2 11 14428
rm        2 12 14408
sh        2 13 28464
stressfs  2 14 15340
usertests 2 15 62840
wc        2 16 15864
zombie    2 17 13988
console   3 18 0
$
```

그림 5 ls

make 이후 xv6가 부팅되었는데 뭐를 해야할지 막막해서, 그냥 아무 이유나 생각 없이 ls를 입력하니 무언가 반응이 있었다. ls가 출력하는 것에 ls가 있는 것을 보고, 저것들이 사용자가 실행할 수 있는 명령들이 아닐까? 하는 생각이 들었다. 이어서 눈에 익숙한 것들을 입력해보겠다.

```
kh@ThinkBook-Plus: ~/os/w03-1
$ echo hello
hello
$
```

그림 6 echo

```
kh@ThinkBook-Plus: ~/os/w03-1
$ mkdir testdir
$ ls
.          1 1 512
..         1 1 512
README    2 2 2286
cat       2 3 15440
echo      2 4 14320
forktest  2 5 8764
grep      2 6 18284
init      2 7 14940
kill      2 8 14408
ln        2 9 14304
ls        2 10 16872
mkdir     2 11 14428
rm        2 12 14408
sh        2 13 28464
stressfs  2 14 15340
usertests 2 15 62840
wc        2 16 15864
zombie    2 17 13988
console   3 18 0
testdir   1 19 32
$ rm testdir
$ ls
.          1 1 512
..         1 1 512
README    2 2 2286
cat       2 3 15440
echo      2 4 14320
forktest  2 5 8764
grep      2 6 18284
init      2 7 14940
kill      2 8 14408
ln        2 9 14304
ls        2 10 16872
mkdir     2 11 14428
rm        2 12 14408
sh        2 13 28464
stressfs  2 14 15340
usertests 2 15 62840
wc        2 16 15864
zombie    2 17 13988
console   3 18 0
$
```

그림 7 mkdir, rm

```
kh@ThinkBook-Plus: ~/os/w03-1
$ cat 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
Version 6 (v6). xv6 loosely follows the structure and style of v6,
but is implemented for a modern x86-based multiprocessor using ANSI C.

ACKNOWLEDGMENTS

xv6 is inspired by John Lions's Commentary on UNIX 6th Edition (Peer
to Peer Communications: ISBN: 1-57386-013-7; 1st edition (June 14,
2000)). See also https://pdos.csail.mit.edu/6.828/, which
provides pointers to on-line resources for v6.

xv6 borrows code from the following sources:
  JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)
  Plan 9 (entryother.S, mp.h, mp.c, lapic.c)
  FreeBSD (iopl.c)
  NetBSD (console.c)

The following people have made contributions: Russ Cox (context switching,
locking), Cliff Frey (NP), Xiao Yu (NP), Nikolai Zeldovich, and Austin
Clements.

We are also grateful for the bug reports and patches contributed by Silas
Boyd-Miklizer, Anton Burtsev, Cody Cutler, Mike CAT, Ted Chaired, eweiz800,
Nelson Elhage, Saar Ettinger, Alice Ferrazzi, Nathaniel Fillard, Peter
Froehlich, Yakir Golan, Shivan Handa, Bryan Henry, Jim Huang, Alexander
Kashuk, Anders Kaseorg, kahac95, Wolfgang Keller, Eddie Kohler, Austin
Liew, Ibar Marinescu, Yandong Mao, Natan Shabtay, Hitoshi Miike, Carmi
Merinovitch, Mark Morrissey, ntsn, Joel Nider, Greg Price, Ayan Shafast,
Eldar Sehayek, Yongming Shen, Can Tenny, tyfids, Rafael Ubal, Warren
Toomey, Stephen Tu, Pablo Ventura, Xi Wang, Keiichi Matanabe, Nicolas
Molovich, wxdoo, Grant Wu, Jindong Zhang, Icenowy Zheng, and Zou Chang Wei.

The code in the files that constitute xv6 is
Copyright 2006-2018 Frans Kaashoek, Robert Morris, and Russ Cox.

ERROR REPORTS

We don't process error reports (see note on top of this file).

BUILDING AND RUNNING XV6

To build xv6 on an x86 ELF machine (like Linux or FreeBSD), run
"make". On non-x86 or non-ELF machines (like OS X, even on x86), you
will need to install a cross-compiler gcc suite capable of producing
x86 ELF binaries (see https://pdos.csail.mit.edu/6.828/).
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC
simulator and run "make qemu".
```

그림 8 cat README

p.5에서 Ubuntu상 nano로 열었던 README 파일을 xv6상에서 cat로 출력한 것 이다.

```
kh@ThinkBook-Plus: ~/os/w03-1
$ grep 'the' README
$ grep the README
NOTE: we have stopped maintaining the x86 version of xv6, and switched
our efforts to the RISC-V version
Version 6 (v6). xv6 loosely follows the structure and style of v6,
xv6 borrows code from the following sources:
  JOS (asm.h, elf.h, mmu.h, bootasm.S, ide.c, console.c, and others)
  Plan 9 (entryother.S, mp.h, mp.c, lapic.c)
We are also grateful for the bug reports and patches contributed by Silas
The code in the files that constitute xv6 is
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC
$
```

그림 9 grep

xv6상 grep은 일반적인 리눅스상의 grep과는 사용법이 다른 듯 하다.

```
kh@ThinkBook-Plus: ~/os/w03-1
$ wc README
50 329 2286 README
$
```

그림 10 wc

결론적으로 xv6 파일들이 있는 폴더는 Ubuntu에서 /bin(혹은 /usr/bin) 폴더와 비슷하게 실행파일을 담고있는것으로 생각된다. User Application 실행은 이정도로 끝내겠다.

2-3. Hello, World?

구현 이후로 본 장을 쓰러 되돌아보니 너무나 주먹구구식으로 해결한것같아 이렇게 해결한 것을 보고서랍시고 적어도 될지 조금 조심스럽다.

여러모로 관찰하다가 한가지 의문이 들었다. 이전장에서 xv6상에서 ls를 입력하니 w03-1의 모든 파일들이 표시되지 않고, 언더바(_)로 시작하는 파일들만 표시됐다. 이를 Ubuntu상에서 보니 아래와 같다.

```
kh@ThinkBook-Plus: ~/os/w03-1$ ls
BUGS          bootmain.d    file.d         ioapic.d      ls.sym        proc.o        string.c      user.h
LICENSE       bootmain.o    file.h         ioapic.o      main.c        rm.asm       string.d      usertests.asm
Makefile      buf.h         file.o         kalloc.c      main.d        rm.c         string.o      usertests.c
Notes         cat.asm       forktest.asm  kalloc.d      main.o        rm.d         swtch.S       usertests.d
README        cat.c         forktest.c    kalloc.o      memide.c      rm.o         swtch.o       usertests.o
TRICKS        cat.d         forktest.d    kbd.c         nkmlayout.h  rm.sym       syscall.c     usertests.sym
_cat          cat.o         forktest.o    kbd.d         nkml.asm     runoff        syscall.d     usys.S
echo          cat.sym       fs.c          kbd.h         nkml.c        runoff.list  syscall.h     usys.o
forktest      console.c     fs.d          kbd.o         nkml.d        runoff.spec  syscall.o     vectors.S
grep          console.d     fs.h          kill.c         nkml.o        runoff1       syscall.o     vectors.o
init          console.o     fs.ing        kernel.asm    nkml.sym      sh.asm       syscall.d     vectors.pl
kill          outh         fs.o          kernel.ld     nkfs          sh.c         sysfile.c    vm.c
ln            date.h       gdbutil       kernel.sym    nkfs.c        sh.d         sysproc.c    vm.d
ls            defs.h       grep.asm      kill.asm      nmuh          sh.o         sysproc.o    wc.asm
mkdir         dot-bochsrc  grep.c        kill.d        np.c          showl        toc.ftr      wc.c
rm            echo.asm     grep.d        kill.o        np.h          sign.pl      toc.hdr      wc.d
sh            echo.c       grep.o        kill1.sym     np.o          sleep1.p     trap.c       wc.o
stressfs      echo.d       grep.sym      ide.c         lapic.c       sleeplock.c  trap.d       wc.sym
usertests     echo.o       ide.d         ide.o         lapic.d       sleeplock.d  trap.o       xv6.ing
wc            elf.h        entry.S       init.asm      ln.asm        picirq.c     sleeplock.h  trapasm.S  xv6.ing
zombie        asm.h        entry.o       init.d        ln.d          pipe.c        spinlock.c   traps.h    zombie.asm
asm.c         bio.c        entryother.S  init.o        ln.o          pipe.d        spinlock.d   types.h    zombie.c
bio.d         bio.o        entryother.o  initcode     ln.sym        pr.pl        spinlock.h   uart.c     zombie.d
bootasm.S     bootasm.d    entryother.d  initcode.asm log.c         printf.c     spinlock.o   uart.d     zombie.o
bootasm.o     bootasm.o    entryother.o  initcode.d   log.d         printf.d     stat.h       uart.o     zombie.sym
bootblock     bootblock.asm exec.c         initcode.o   log.o         printf.o     stressfs.asm ulib.c
bootblock.o   bootblock.o  exec.d        initcode.o   ls.asm        printcs      stressfs.c   ulib.o
bootblockother.o font1.h      initcode.out  ls.c         proc.c        stressfs.d   umalloc.c
bootmain.c    file.c       ioapic.c      ls.o         proc.d        stressfs.o   umalloc.d
kh@ThinkBook-Plus: ~/os/w03-1$
```

그림 11 w03-1 에서 ls

git으로 clone한 xv6-public 원본 폴더에서 어떤 파일들이 있는지 관찰하였다.

```
kh@ThinkBook-Plus: ~/os/src/xv6-public$ ls
BUGS          cat.c          font1.h        ioapic.c      memide.c      printf.c      sign.pl       syscall.c   ulib.c
LICENSE       console.c      file.c         kalloc.c      nkmlayout.h  printcs       sleep1.p     syscall.h  umalloc.c
Makefile      outh          file.h         kbd.c         nkml.c        proc.c        sleeplock.c  sysfile.c  user.h
Notes         date.h        forktest.c     kbd.h         nkfs.c        proc.h        sleeplock.h  sysproc.c  usertests.c
README        defs.h        fs.c           kernel.ld     nmuh          rm.c          spinlock.c   toc.ftr    usys.S
TRICKS        dot-bochsrc   fs.h           kill.c         nkml.o        runoff        spinlock.h   toc.hdr    vectors.pl
asm.h         echo.c        gdbutil       lapic.c       np.c          runoff.list  spinlock.o   trap.c     vm.c
bio.c         elf.h         entry.S        init.c        ln.c          param.h      stressfs.c   traps.h    wc.c
bootasm.S     entry.o       ide.c         log.c         ln.d          picirq.c     stressfs.d   types.h    xv6.h
bootmain.c    entryother.S  initcode.S    main.c        ln.o          pipe.c       string.c     types.h    zombie.c
buf.h         exec.c        initcode.o     ls.c          pr.pl         showl        swtch.S      uart.c
kh@ThinkBook-Plus: ~/os/src/xv6-public$
```

그림 12 xv6-public 에서 ls

터미널 상에서 초록색으로 표시된 파일들은 실행파일이다. 언더바로 시작하는 파일은 make를 통해 생성되는 걸 알 수 있었다. 이는 다시 말해 필자가 구현하려는 helloworld의 실행파일 역시 xv6를 make 하면서 생성되어야 한다는 것을 자연스럽게 생각해낼 수 있다.

만들어야 할 프로그램이 helloworld.c라 하자. 이번 과제의 공지에 ‘반드시 xv6 운영체제에 구현되어있는 printf 함수를 사용할 것’이라는 것을 보고 크게 두 가지의 생각이 들었다.

첫 번째는 helloworld.c에 어떤 헤더가 포함되어야 하는지이다. printf가 포함된 stdio.h는 보통의 리눅스 환경에서 /usr/include에 있는데, xv6는 usr폴더가 아예 없기 때문이다. 수많은 헤더파일중 하나의 어딘가에 printf가 포함되어있는지 찾아서 이를 helloworld.c에서 불러와야한다.

```

kh@ThinkBook-Plus: ~/os/w03-1
GNU nano 6.2
/*
#include "unknown header"
...
*/

int main (void){
    printf("Hello, World!");
    return 0;
}
    
```

그림 13 helloworld.c 초안

대강 초안을 적어보자면 이정도이다. 이제 필요한 헤더파일을 찾아야한다.

<pre> kh@ThinkBook-Plus: ~/os/w03-1 GNU nano 6.2 cat.c #include "types.h" #include "stat.h" #include "user.h" char buf[512]; void cat(int fd) { int n; </pre>	<pre> kh@ThinkBook-Plus: ~/os/w03-1 GNU nano 6.2 echo.c #include "types.h" #include "stat.h" #include "user.h" int main(int argc, char *argv[]) { int i; for(i = 1; i < argc; i++) printf(1, "%s%s", argv[i], " "); </pre>	<pre> kh@ThinkBook-Plus: ~/os/w03-1 GNU nano 6.2 ls.c #include "types.h" #include "stat.h" #include "user.h" #include "fs.h" char* fmtname(char *path) { static char buf[DIRSIZ+1]; char *p; </pre>
<pre> 선택 kh@ThinkBook-Plus: ~/os/w03-1 GNU nano 6.2 mkdir.c #include "types.h" #include "stat.h" #include "user.h" int main(int argc, char *argv[]) { int i; if(argc < 2){ </pre>	<pre> kh@ThinkBook-Plus: ~/os/w03-1 GNU nano 6.2 rm.c #include "types.h" #include "stat.h" #include "user.h" int main(int argc, char *argv[]) { int i; if(argc < 2){ </pre>	<pre> kh@ThinkBook-Plus: ~/os/w03-1 GNU nano 6.2 wc.c #include "types.h" #include "stat.h" #include "user.h" char buf[512]; void wc(int fd, char *name) { int i, n; </pre>

그림 14 실행파일이 만들어지는 .c 파일들의 헤더

본 장의 처음에 주먹구구식으로 해결했다는 것이 이것이다. 정말 무식하게 하나하나 다 열어봤다. 그 결과 모든 파일이 “types.h”, “stat.h”, “user.h” 이 세가지 헤더파일을 포함하고 있었다. ls.c가 추가적으로 포함하고 있는 fs.h는 파일 관련한 작업을 하기 위한 File System 관련 헤더가 아닐까 추정해본다.

확신을 갖기위해 printf가 사용되는 .c파일을 전부 찾아보겠다. 첫장에서 말했던 것처럼 가독성을 위해 코드 탐색'만'은 IDE의 도움을 빌리겠다

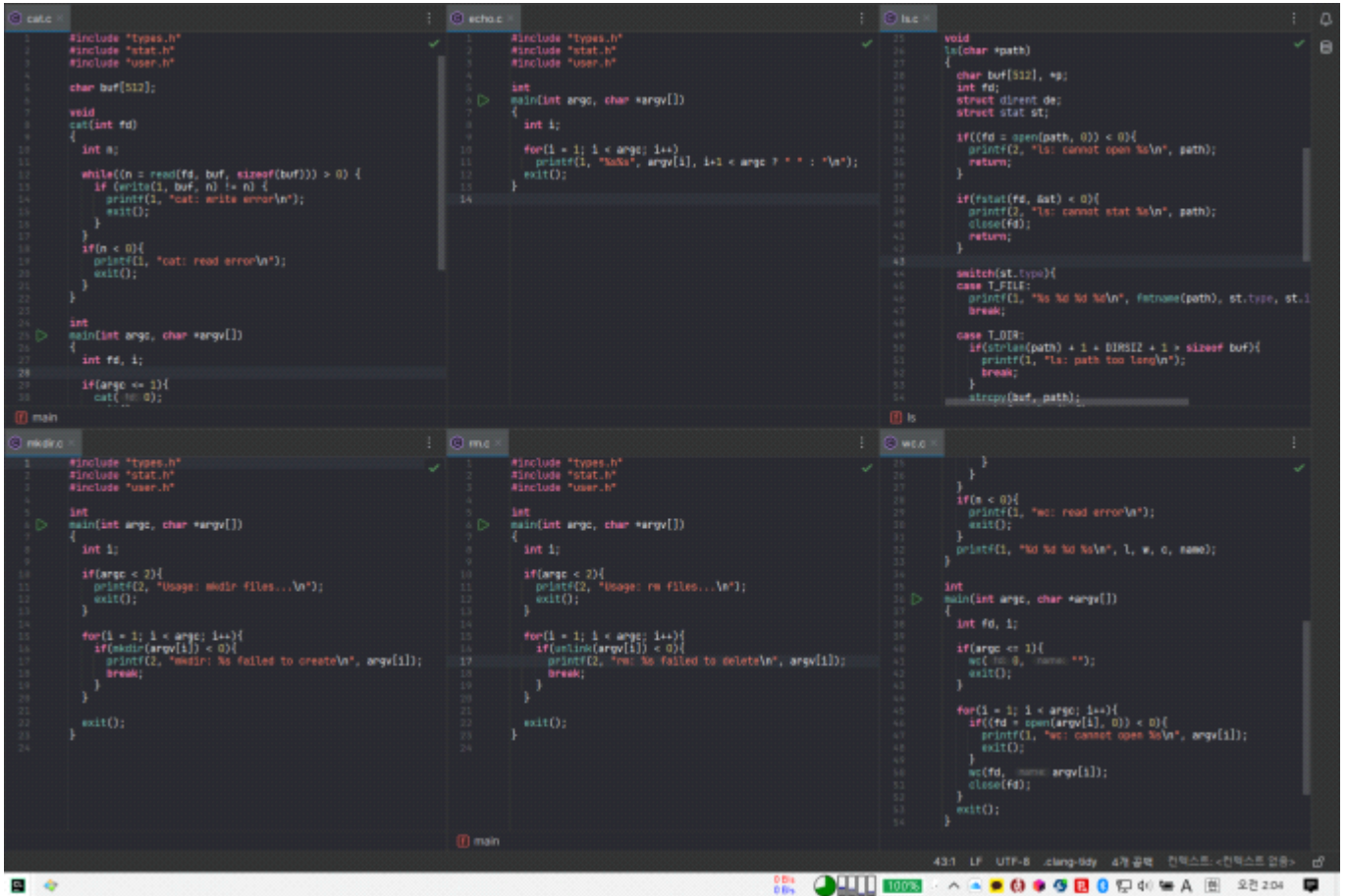


그림 15 실행파일이 만들어지는 .c 파일들

그림이 작아 잘 안보일수도 있다. 본 보고서와 같은 위치에 'images'폴더에 첨부해두었으니 필요시 확인 가능하다. 이전과 같은 실행파일이 만들어지는 .c 파일 6개이다.

이들은 공교롭게도 printf를 전부 포함하고 있다. 논리적으로 생각해봤을 때, printf를 사용하려는데 저 세 헤더파일 모두가 필요하지 않다 해도, 일단 전부 포함 시킨다면 사용은 가능한 것 아닌가? 추정에 확신이 더해진 다. 이에 더해서 printf의 사용법과 main함수의 종료가 필자가 알고있던것과는 달랐다. 이 역시 참고하면서 helloworld.c를 수정하였다.

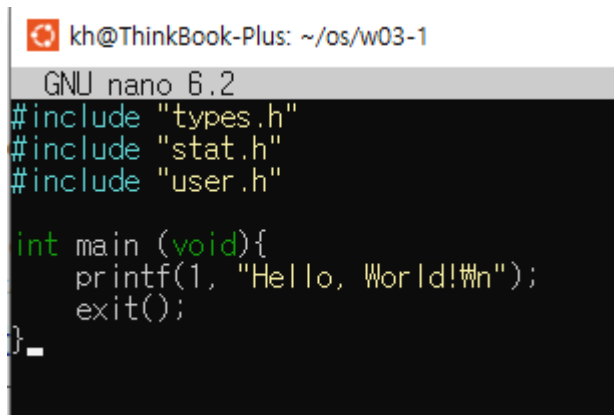
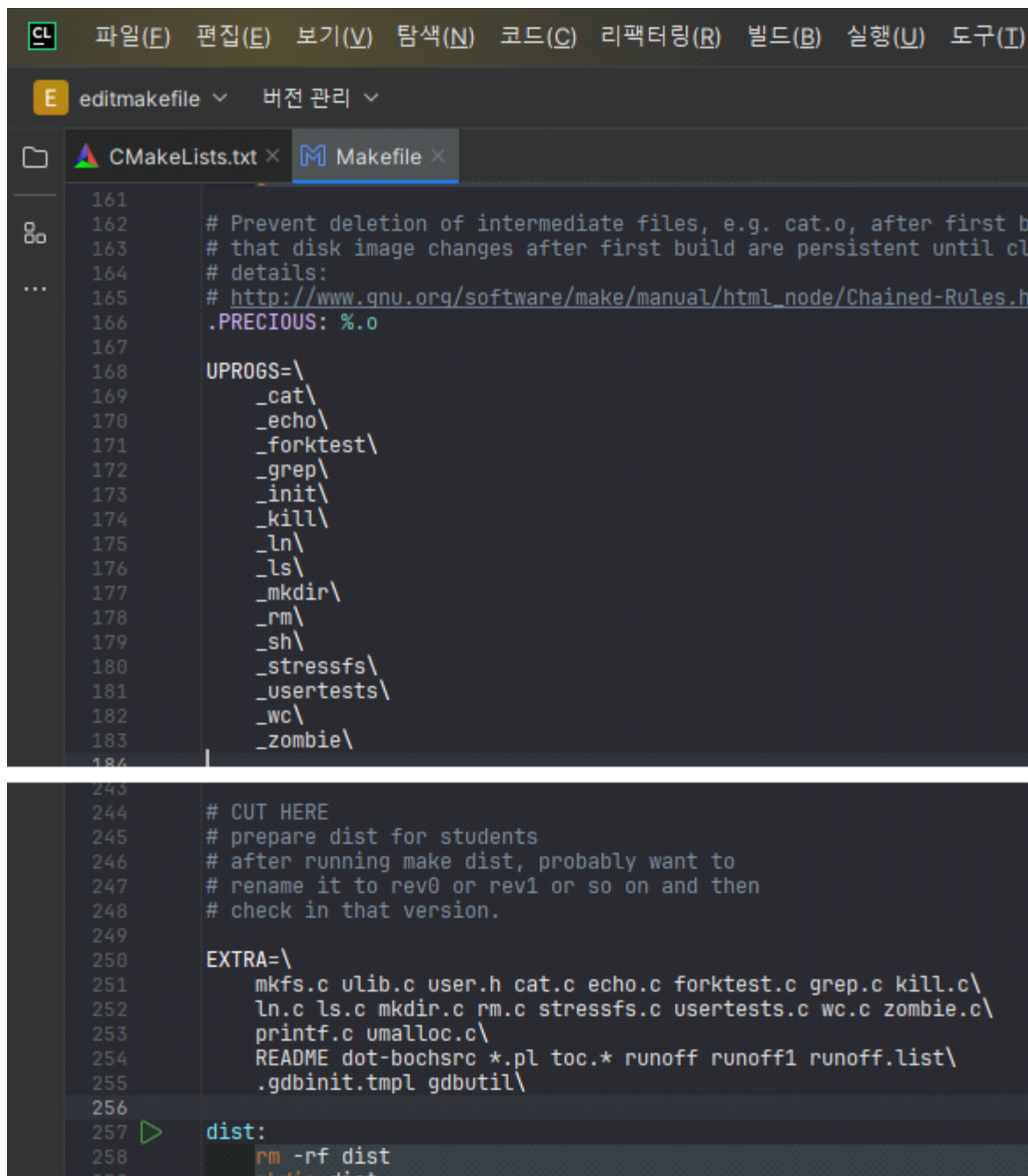


그림 16 helloworld.c 최종

두 번째는 helloworld.c를 어떻게 make 시에 포함시켜야 할지다. 백문이 불여일견이라고 일단 Makefile 파일을 열어보기로 했다.



```
161
162 # Prevent deletion of intermediate files, e.g. cat.o, after first b
163 # that disk image changes after first build are persistent until cl
164 # details:
165 # http://www.gnu.org/software/make/manual/html_node/Chained-Rules.h
166 .PRECIOUS: %.o
167
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
243
244 # CUT HERE
245 # prepare dist for students
246 # after running make dist, probably want to
247 # rename it to rev0 or rev1 or so on and then
248 # check in that version.
249
250 EXTRA=\
251     mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
252     ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c\
253     printf.c umalloc.c\
254     README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
255     .gdbinit.tmpl gdbutil\
256
257 dist:
258     rm -rf dist
259     make dist
```

그림 17 CLion으로 열어본 Makefile 파일의 일부

Makefile을 열어서 실행파일 이름이 들어가는 부분을 또 주먹구구식으로 전부 찾아보니 두 부분에서 나왔다. 이들을 전부 수정해주었다.

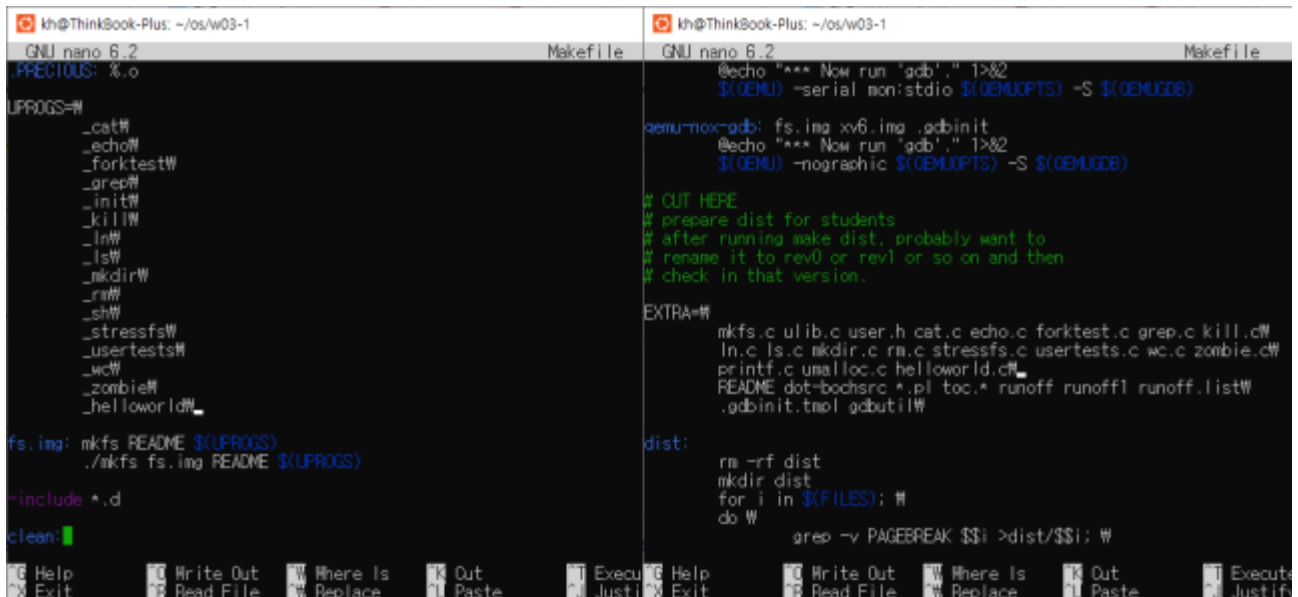


그림 18 Makefile 수정

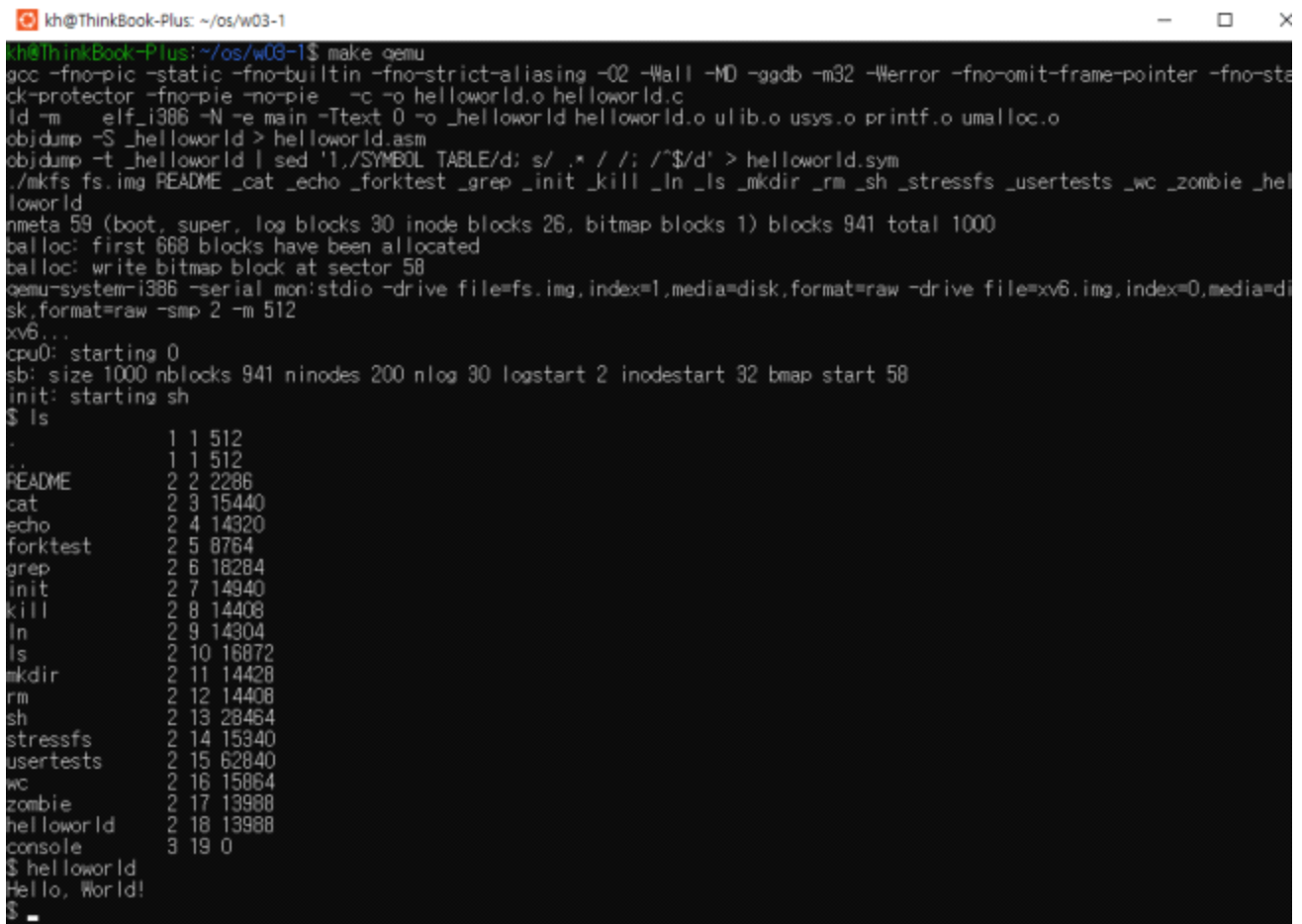


그림 19 Hello, World!

필자도 왜 되는지는 모르겠으나 아무튼 오류하나 없이 작동은 잘 된다. 무슨 ‘두뇌자극 퀴즈100’같은 책을 푸는 것도 아닌데 정말 느낌 가는데로 끼워맞췄고, 이게 설명의 전부이다. 해결은 되었지만 참 스스로에게 부끄럽기 짝이 없다.

이렇게 되는데로 막 진행하면 구현은 되나 배워가는데 없으므로, 얼렁뚱땅 구현한 만큼 각각 어떤 역할을 하는지 사후적인 분석 역시나 제대로 해야하는데, 이 보고서는 09월 25일(월) 수업을 5시간 남겨두고 마무리중이다.

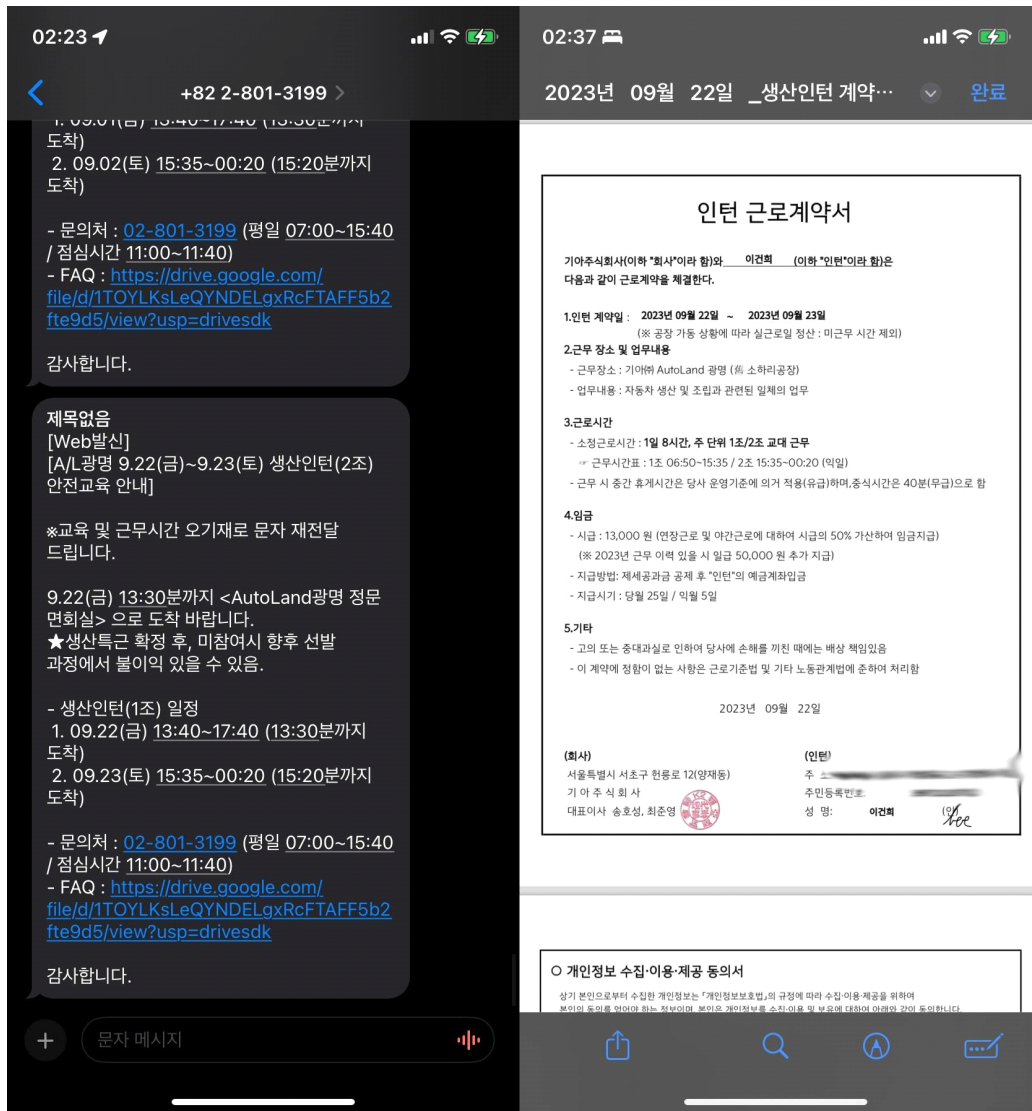
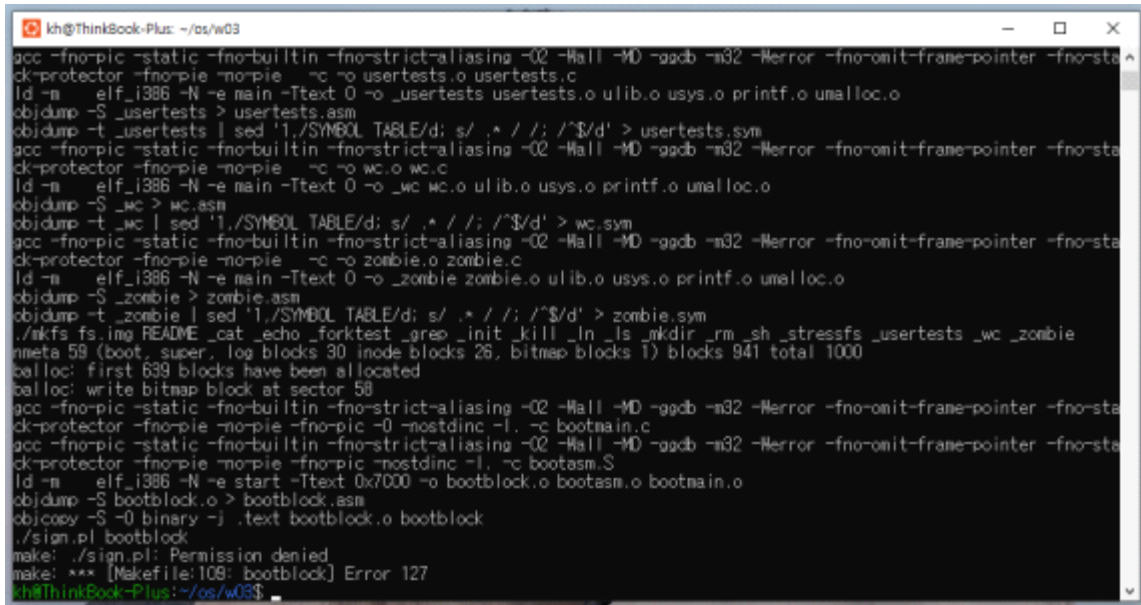


그림 20 좋지못한 금전 사정

좋지못한 금전 사정으로 금요일 공장인 필자는 9월 22(금)~23(토)에 기아자동차 소하리 공장의 야간조에서 자동차에 안전벨트를 장착하다 왔다. 스스로의 공부를 위해서라도 기능을 제대로 알고가는데 맞고, 다음번 보고서의 마지막장에 포함할 것이니 조금 기다려 주셨으면 좋겠다. 정말 약속드린다.

3. Trouble Shooting

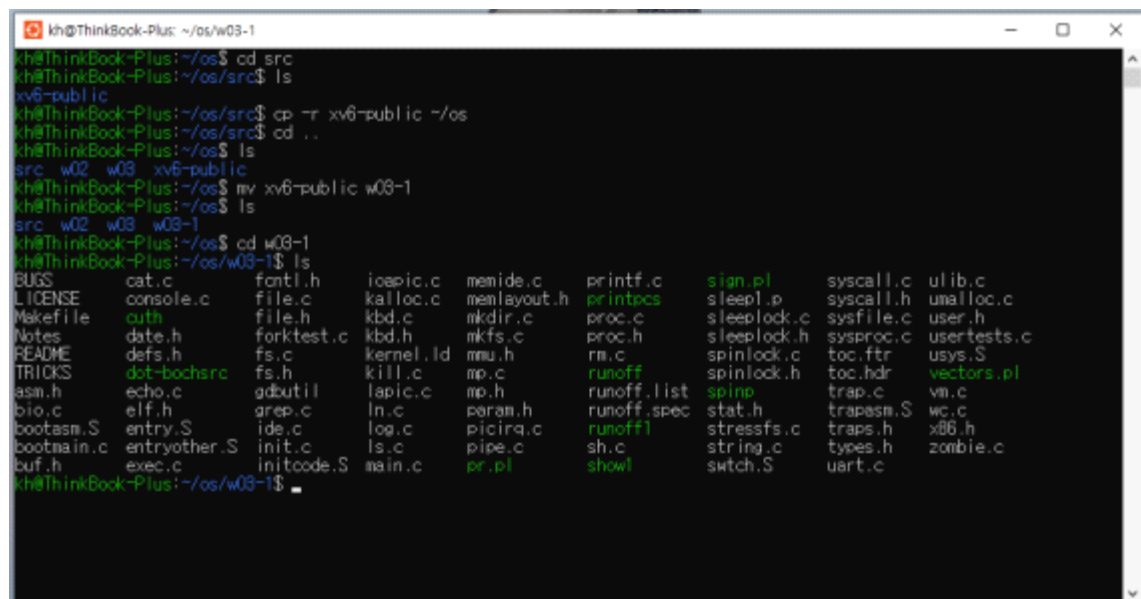
3-1. make error



```
kh@ThinkBook-Plus: ~/os/w03
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -c -o usertests.o usertests.c
ld -m elf_i386 -N -e main -Ttext 0 -o _usertests usertests.o ulib.o usys.o printf.o unalloc.o
objdump -S _usertests > usertests.asm
objdump -t _usertests | sed '1,/SYMBOL TABLE/d; s/ .* //; /$/d' > usertests.sym
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -c -o wc.o wc.c
ld -m elf_i386 -N -e main -Ttext 0 -o _wc wc.o ulib.o usys.o printf.o unalloc.o
objdump -S _wc > wc.asm
objdump -t _wc | sed '1,/SYMBOL TABLE/d; s/ .* //; /$/d' > wc.sym
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -c -o zombie.o zombie.c
ld -m elf_i386 -N -e main -Ttext 0 -o _zombie zombie.o ulib.o usys.o printf.o unalloc.o
objdump -S _zombie > zombie.asm
objdump -t _zombie | sed '1,/SYMBOL TABLE/d; s/ .* //; /$/d' > zombie.sym
./mkfs fs.img README _cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm _sh _stressfs _usertests _wc _zombie
mmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 941 total 1000
balloc: first 639 blocks have been allocated
balloc: write bitmap block at sector 58
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -fno-pic -O -nostdinc -I. -c bootmain.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -fno-pic -nostdinc -I. -c bootasm.S
ld -m elf_i386 -N -e start -Ttext 0x7000 -o bootblock.o bootasm.o bootmain.o
objcopy -S -O binary -j .text bootblock.o bootblock
./sign.pl bootblock
make: ./sign.pl: Permission denied
make: *** [Makefile:108: bootblock] Error 127
kh@ThinkBook-Plus: ~/os/w03$
```

그림 21 make시 파일 권한 오류 발생

make시 ./sign.pl에서 Permission denied 오류가 발생하였다. 파일 권한의 문제이다. chmod로도 해결이 가능하겠지만, 문제점을 찾기 위해 원본 파일을 복사하겠다.



```
kh@ThinkBook-Plus: ~/os/w03-1
kh@ThinkBook-Plus: ~/os$ cd src
kh@ThinkBook-Plus: ~/os/src$ ls
xv6-public
kh@ThinkBook-Plus: ~/os/src$ cp -r xv6-public -/os
kh@ThinkBook-Plus: ~/os/src$ cd ..
kh@ThinkBook-Plus: ~/os$ ls
src w02 w03 xv6-public
kh@ThinkBook-Plus: ~/os$ mv xv6-public w03-1
kh@ThinkBook-Plus: ~/os$ ls
src w02 w03 w03-1
kh@ThinkBook-Plus: ~/os$ cd w03-1
kh@ThinkBook-Plus: ~/os/w03-1$ ls
BUGS      cat.c      font1.h    ioapic.c   memide.c   printf.c   sign.pl     syscall.c  ulib.c
LICENSE   console.c  file.c     kalloc.c   memlayout.h  printcs    sleep1.p   syscall.h  unalloc.c
Makefile   auth       file.h     kbd.c      mmu.c      proc.c     sleeplock.c sysfile.c  user.h
Notes      date.h     forktest.c kbd.h      mkfs.c     proc.h     sleeplock.h sysproc.c  usertests.c
README     defs.h     fs.c       kernel.ld  mmu.h      rm.c       spinlock.c  toc.ftr    usys.S
TRICKS     dot-bochsrc fs.h       kill.c     mp.c       runoff     spinlock.h  toc.hdr    vectors.pl
asm.h      echo.c     gdbutil    lapic.c    mp.h       runoff.list spine       trap.c     vm.c
bio.c      elf.h      grep.c     ln.c       paran.h     runoff.spec stat.h      trapasm.S  wc.c
bootasm.S  entry.S    ide.c      log.c      picirq.c   runoff1    stressfs.c  traps.h    x86.h
bootmain.c entryother.S init.c     ls.c       pipe.c     sh.c       string.c   types.h    zombie.c
buf.h      exec.c     initcode.S main.c     pr.pl      showl      swtch.S    uart.c
```

그림 22 cp 및 mv

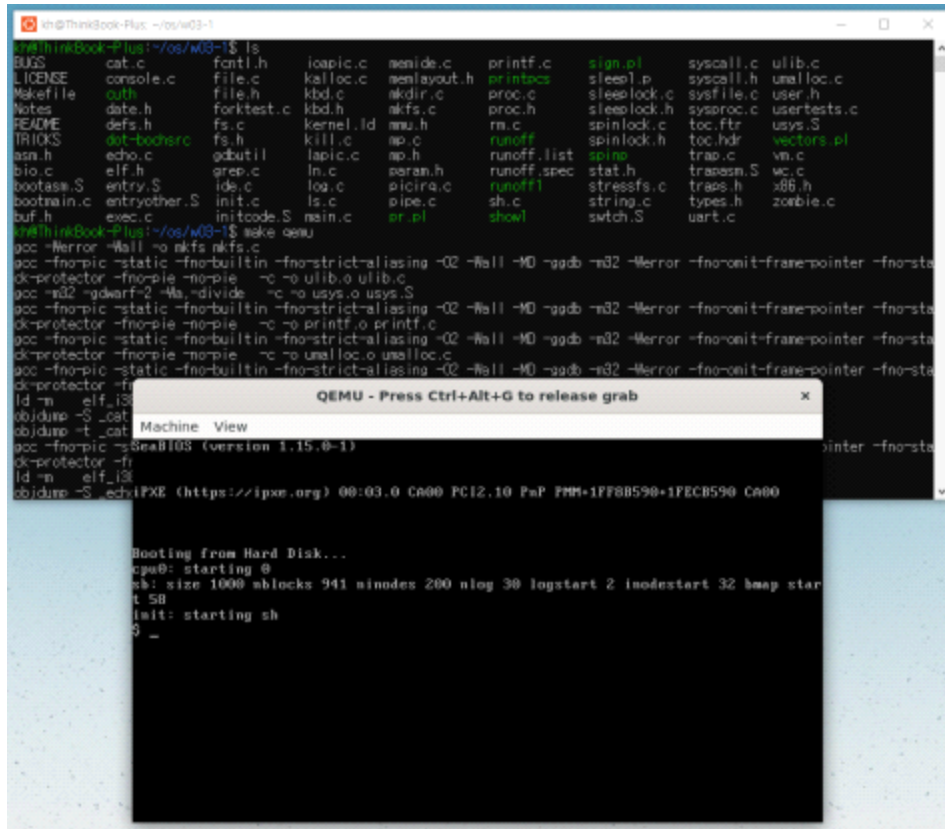


그림 23 에러없이 make 완료

Ubuntu상에서 복사하니 오류없이 make 된다. 오류의 원인을 찾기위해 조건을 바꾸면서 테스트 하다보니, 필자의 게으름으로 인해 윈도우 탐색기에서 복사를 한 것이 그 이유로 밝혀졌다.

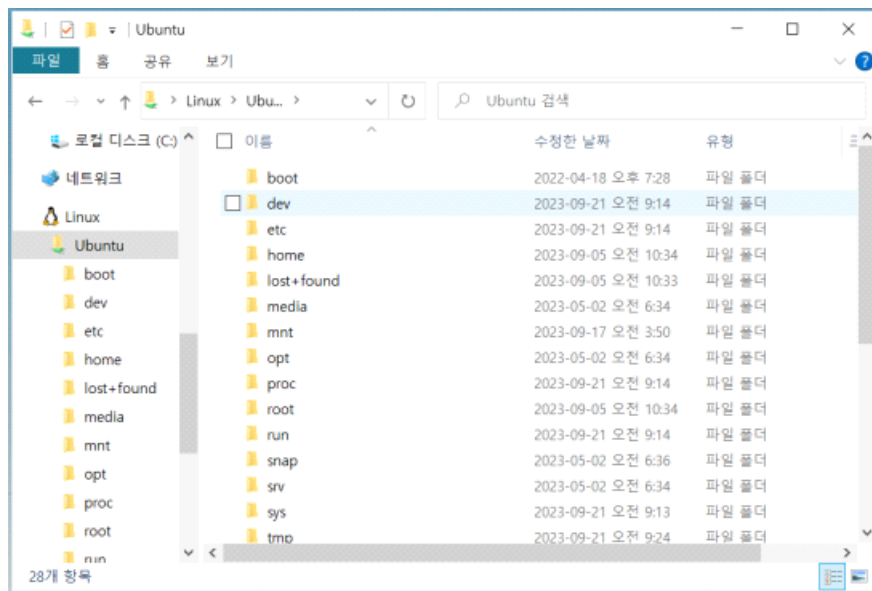


그림 24 윈도우 탐색기상에서 localhost로 마운트 된 Ubuntu의 vhd

필자의 환경에서 WSL의 vhd파일은
 ~\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu_79rhkp1fndgsc\LocalState
 에 저장이 되고, MS에서 Linux와의 통합성을 강조내지 과시하기 위해 탐색기의 사이드바에 자동으로 마운트되
 게끔 했을거라는 개인적인 추측이 있다.

kh@ThinkBook-Plus: ~/os/w03

-rw-r--r--	1	kh	kh	68	Sep 21 23:45	printf.d
-rw-r--r--	1	kh	kh	7104	Sep 21 23:45	printf.o
-rw-r--r--	1	kh	kh	367	Sep 5 11:22	printocs
-rw-r--r--	1	kh	kh	11717	Sep 5 11:22	proc.c
-rw-r--r--	1	kh	kh	110	Sep 22 03:22	proc.d
-rw-r--r--	1	kh	kh	2270	Sep 5 11:22	proc.h
-rw-r--r--	1	kh	kh	24316	Sep 22 03:22	proc.o
-rw-r--r--	1	kh	kh	39733	Sep 21 23:45	rm.asm
-rw-r--r--	1	kh	kh	322	Sep 5 11:22	rm.c
-rw-r--r--	1	kh	kh	60	Sep 21 23:45	rm.d
-rw-r--r--	1	kh	kh	3424	Sep 21 23:45	rm.o
-rw-r--r--	1	kh	kh	688	Sep 21 23:45	rm.sym
-rw-r--r--	1	kh	kh	5042	Sep 5 11:22	runoff
-rw-r--r--	1	kh	kh	628	Sep 5 11:22	runoff.list
-rw-r--r--	1	kh	kh	2575	Sep 5 11:22	runoff.spec
-rw-r--r--	1	kh	kh	2318	Sep 5 11:22	runoff1
-rw-r--r--	1	kh	kh	103240	Sep 21 23:45	sh.asm
-rw-r--r--	1	kh	kh	8240	Sep 5 11:22	sh.c
-rw-r--r--	1	kh	kh	61	Sep 21 23:45	sh.d
-rw-r--r--	1	kh	kh	23244	Sep 21 23:45	sh.o
-rw-r--r--	1	kh	kh	1058	Sep 21 23:45	sh.sym
-rw-r--r--	1	kh	kh	135	Sep 5 11:22	show1
-rw-r--r--	1	kh	kh	363	Sep 5 11:22	sign.pl
-rw-r--r--	1	kh	kh	1990	Sep 5 11:22	sleep1.p
-rw-r--r--	1	kh	kh	812	Sep 5 11:22	sleeplock.c
-rw-r--r--	1	kh	kh	132	Sep 22 03:22	sleeplock.d
-rw-r--r--	1	kh	kh	265	Sep 5 11:22	sleeplock.h
-rw-r--r--	1	kh	kh	7324	Sep 22 03:22	sleeplock.o
-rw-r--r--	1	kh	kh	2791	Sep 5 11:22	spinlock.c
-rw-r--r--	1	kh	kh	118	Sep 22 03:22	spinlock.d

kh@ThinkBook-Plus: ~/os/w03-1

-rw-r--r--	1	kh	kh	68	Sep 22 00:00	printf.d
-rw-r--r--	1	kh	kh	7108	Sep 22 00:00	printf.o
-rwxr-xr-x	1	kh	kh	367	Sep 21 23:58	printocs
-rw-r--r--	1	kh	kh	11717	Sep 21 23:58	proc.c
-rw-r--r--	1	kh	kh	110	Sep 22 00:00	proc.d
-rw-r--r--	1	kh	kh	2270	Sep 21 23:58	proc.h
-rw-r--r--	1	kh	kh	24320	Sep 22 00:00	proc.o
-rw-r--r--	1	kh	kh	39733	Sep 22 00:00	rm.asm
-rw-r--r--	1	kh	kh	322	Sep 21 23:58	rm.c
-rw-r--r--	1	kh	kh	60	Sep 22 00:00	rm.d
-rw-r--r--	1	kh	kh	3428	Sep 22 00:00	rm.o
-rw-r--r--	1	kh	kh	688	Sep 22 00:00	rm.sym
-rwxr-xr-x	1	kh	kh	5042	Sep 21 23:58	runoff
-rw-r--r--	1	kh	kh	628	Sep 21 23:58	runoff.list
-rw-r--r--	1	kh	kh	2575	Sep 21 23:58	runoff.spec
-rwxr-xr-x	1	kh	kh	2318	Sep 21 23:58	runoff1
-rw-r--r--	1	kh	kh	103240	Sep 22 00:00	sh.asm
-rw-r--r--	1	kh	kh	8240	Sep 21 23:58	sh.c
-rw-r--r--	1	kh	kh	61	Sep 22 00:00	sh.d
-rw-r--r--	1	kh	kh	23248	Sep 22 00:00	sh.o
-rw-r--r--	1	kh	kh	1058	Sep 22 00:00	sh.sym
-rwxr-xr-x	1	kh	kh	135	Sep 21 23:58	show1
-rwxr-xr-x	1	kh	kh	363	Sep 21 23:58	sign.pl
-rw-r--r--	1	kh	kh	1990	Sep 21 23:58	sleep1.p
-rw-r--r--	1	kh	kh	812	Sep 21 23:58	sleeplock.c
-rw-r--r--	1	kh	kh	132	Sep 22 00:00	sleeplock.d
-rw-r--r--	1	kh	kh	265	Sep 21 23:58	sleeplock.h
-rw-r--r--	1	kh	kh	7328	Sep 22 00:00	sleeplock.o
-rw-r--r--	1	kh	kh	2791	Sep 21 23:58	spinlock.c
-rw-r--r--	1	kh	kh	118	Sep 22 00:00	spinlock.d

그림 25 ls -al로 permission 조회 (좌 오류 발생하는 w03 / 우 오류 발생하지 않는 w03-1)

검증을 위해 permission을 조회해보았다. 역시나 예상한대로 오류가 발생한 파일은 실행권한(x)이 제대로 부여되지 않았고, 그 외의 파일들도 실행권한이 없는 것을 발견했다.

혹시나 WSL 환경에서 개발을 한다면 이를 주의해야 할 것이다.

3-2. QEMU 사용법 관련

copyright: see <https://qemu.siam.org/terms-privacy>

Table of Contents

"R.T.F.M." -- Anonymous

	Page
PREFACE	V
INTRODUCTION	
1. Overview	1.1
2. Software Design.	1.3
3. General Numerical Properties	1.7
CHAPTER 1. GENERAL MATRICES	
1. Overview	1.1
2. Usage.	1.2
3. Examples	1.6
4. Algorithms	1.10
5. Programming Details.	1.16
6. Performance.	1.19
7. Notes and References	1.34

그림 26 “R.T.F.M.” -- 익명

‘RTFM’이라는 유명한 속언이 있다. LINPACK이라는 소프트웨어의 메뉴얼¹⁰⁾에서 기원했다는 설이 유력하다. 어떤 소프트웨어를 사용함에 있어서 구글링해서 나오는 누군지도 모르는 일개 유저가 블로그에 써놓은 소프트웨어 경험담이 정확하겠는가, 아니면 해당 프로그램을 만든이가 만든 의도대로 이렇게 사용하라 직접 적어놓은 매뉴얼이 정확하겠는가? 필자는 매뉴얼이 아니면 믿지 않는다. 필자가 누구를 가르칠만한 사람도 아니고 가르칠 능력도 없으나, 제발 매뉴얼을 잘 읽자는 것은 수십번 강조해도 지나치지 않다.

qemu로 xv6를 처음 실행시켰을때는 다시 Ubuntu로 나오지도 못했었다. --help를 하니 커맨드의 파라미터에 대한 설명 뿐이어서 qemu의 공식 문서를 찾아봤다. 가장 궁금했던 에뮬레이터를 종료하는 키 조합은 Ctrl-a x라 한다.

본 보고서에 매뉴얼 상 나오는 키조합 전부를 적는건 무의미한 일인 것 같고, 이에 대한 링크만 걸어두겠다.

- <https://www.qemu.org/docs/master/> - QEMU man page
- <https://www.qemu.org/docs/master/system/mux-chardev.html> - backend 키 조합
- <https://www.qemu.org/docs/master/system/keys.html> - frontend 키 조합

10) <https://epubs.siam.org/doi/pdf/10.1137/1.9781611971811.fm>, 3.