12201922 이규민

2. Homework

1) Login to the system. Show the current directory. Show what files are in your directory.

```
kyumin@DESKTOP-NUDFAPK ~
$ pwd
/cygdrive/c/Users/kyumin/AppData/Roaming/SPB_Data

kyumin@DESKTOP-NUDFAPK ~
$ ls
cdssetup  d1  d2  f1
```

Pwd 를 통해 현 위치 확인이 가능하고, ls 를 이용해 현 위치의 파일들을 볼 수 있다.

2) Go to “/etc” directory. "file *" will show the information for all files in the current directory. Combine "file *" and "grep" using the pipe symbol(|) to display file information only for text files.

$ file * | grep text

```
kyumin@DESKTOP-NUDFAPK /etc
$ file *
DIR_COLORS:         ASCII text
alternatives:       directory
bash.bash_logout:   ASCII text
bash.bashrc:        ASCII text
bash_completion.d:  directory
crypto-policies:    directory
defaults:           directory
fstab:              ASCII text
fstab.d:            sticky, directory
hosts:              symbolic link to /
```

```
kyumin@DESKTOP-NUDFAPK /etc
$ file * | grep text
DIR_COLORS:         ASCII text
bash.bash_logout:   ASCII text
bash.bashrc:        ASCII text
fstab:              ASCII text
man_db.conf:        ASCII text
nsswitch.conf:      ASCII text
profile:            ASCII text
shells:             ASCII text
vimrc:              ASCII text, with escape sequences
xattr.conf:         ASCII text
```

Etc 디렉토리 안에는 여러가지 파일이 있다. 여기서 file * | grep text 명령어를 사용한다면 text 파일들만 출력된다.

File *은 모든 파일의 타입을 출력하는 명령어고, |파이프를 넣은 후 grep text 를 입력하면 text 형식의 파일들만 찾아서 출력하라는 명령어다.

3) (If your Cygwin has no /etc/passwd, make one with “mkpasswd > /etc/passwd”.)
Find the location of the password file (“passwd”), the location of C header files such as "stdio.h", and the location of utility programs (or Linux commands) such as “ls”. Use "whereis" commad. What is the difference between /usr/bin/passwd and /etc/passwd? Use "ls -l /usr/bin/passwd /etc/passwd" and “file /usr/bin/passwd /etc/passwd” to explain the difference.

```
kyumin@DESKTOP-NUDFAPK /etc
$ mkpasswd > /etc/passwd
```

/etc 에 Passwd 파일이 없기 때문에 mkpasswd > /etc/passwd 명령어로 파일을 만들었다.

```
kyumin@DESKTOP-NUDFAPK /
$ whereis passwd
passwd: /usr/bin/passwd.exe /etc/passwd /usr/share/man/man1/passwd.1ossl.gz

kyumin@DESKTOP-NUDFAPK /
$ whereis stdio.h
stdio: /usr/include/stdio.h

kyumin@DESKTOP-NUDFAPK /
$ whereis ls
ls: /usr/bin/ls.exe /usr/share/man/man1/ls.1.gz
```

Whereis 명령어로 passwd, stdio.h, ls 의 실행파일 및 man 페이지 위치를 확인하였다.

```
kyumin@DESKTOP-NUDFAPK /
$ ls -l /usr/bin/passwd.exe
-rwxr-xr-x 1 kyumin Administrators 22547 Feb 27 21:02 /usr/bin/passwd.exe

kyumin@DESKTOP-NUDFAPK /
$ ls -l /etc/passwd
-rw-r--r-- 1 kyumin 없음  1121 Mar 13 09:13 /etc/passwd
```

Ls -l 명령어를 사용하여 /usr/bin/passwd 와 /etc/passwd 의 차이점을 확인했다. Passwd.exe 는 다른 그룹원이나 다른 사람들도 실행할 수 있는 권한이 있다. 하지만 passwd 는 오직 오너만 실행할 수 있고, 그 외의 사람들은 읽기 권한밖에 없다.

```
kyumin@DESKTOP-NUDFAPK /
$ file /usr/bin/passwd.exe
/usr/bin/passwd.exe: PE32+ executable (console) x86-64, for MS Windows, 11 secti
ons

kyumin@DESKTOP-NUDFAPK /
$ file /etc/passwd
/etc/passwd: CSV text
```

File 명령어로 /usr/bin/passwd 와 /etc/passwd 의 차이점을 확인했다. Passwd 는 텍스트 파일이지만 passwd.exe 는 컴파일된 실행파일이다.

4) Go to your login directory ("cd" without arguments will move you to your login directory). Make ex1.c using vi. Compile with "gcc" and run.
vi ex1.c

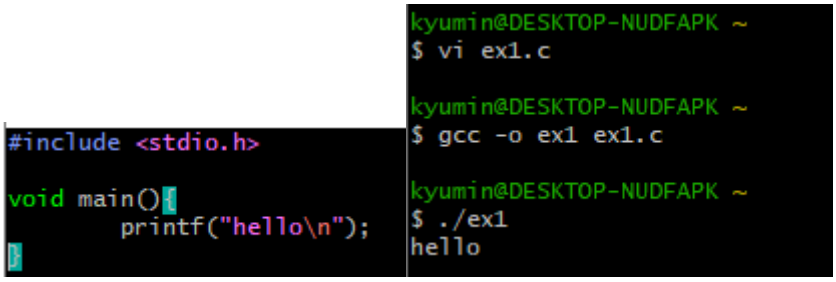
```
#include <stdio.h>

void main(){
```

```
    printf("hello₩n");
}
```

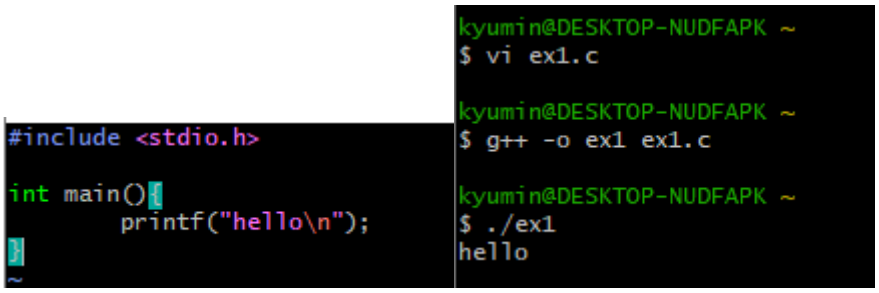
```
gcc -o ex1 ex1.c
./ex1
hello
```

To compile with g++, change void main() => int main()

```
#include <stdio.h>
int main(){
    printf("hello₩n);
}
g++ -o ex1 ex1.c
./ex1
hello
```



Vi ex1.c 명령어로 코맨드 모드 – 인서트 모드 진입 후 c 언어로 코드를 작성하였다. 이후 gcc -o ex1 ex1.c 명령어로 ex1.c 를 ex1 으로 컴파일했다. 이후 ./ex1 명령어로 컴파일 된 프로그램을 실행했고, 정상적으로 메시지가 출력된다.



Vi ex1.c 명령어로 코맨드 모드 – 인서트 모드 진입 후 c++언어로 코드를 작성하였다. 이후 g++ -o ex1 ex1.c 명령어로 ex1.c 를 ex1 으로 컴파일했다. 이후 ./ex1 명령어로 컴파일 된 프로그램을 실행했고, 정상적으로 메시지가 출력된다.

5) Display the contents of ex1.c using cat and xxd. With xxd, you can see the ascii code for each character in ex1.c. Find the ascii codes for the first line of the program: "#include <stdio.h>".

```
kyumin@DESKTOP-NUDFAPK ~
$ cat ex1.c
#include <stdio.h>

int main(){
        printf("hello\n");
}

kyumin@DESKTOP-NUDFAPK ~
$ xxd ex1.c
00000000: 2369 6e63 6c75 6465 203c 7374 6469 6f2e  #include <stdio.
00000010: 683e 0a0a 696e 7420 6d61 696e 2829 7b0a  h>..int main(){.
00000020: 0970 7269 6e74 6628 2268 656c 6c6f 5c6e  .printf("hello\n
00000030: 2229 3b0a 7d0a                           ");.}.
```

Cat ex1.c 는 이전에 c++ 언어로 작성했던 코드 내용을 볼 수 있다. Xxd ex1.c 는 작성했던 문자들을 16 진수로 확인이 가능하다. 예를 들면 위 사진에서 초록색으로 2369 가 확인이 된다. 23 과 69 가 16 진수이고, 그 숫자를 아스키 코드표를 통해서 확인해보면 문자로는 각각 “#”과 “i”임을 알 수 있다.

6) Display the contents of ex1 (the executable file). You cannot use "cat" to see ex1. Why?

```
kyumin@DESKTOP-NUDFAPK ~
$ cat ex1
MZ@       !L!This program cannot be run in DOS mode.
$PEd e&
        @  `P
                    @.text``.data` @.rdata0@@.buildid5@@@.pdataP@@.xd
atad`@@.bss p.idata`@@.rsrc@@.reloc
0@B/45@                                "@B/40$@B/19kl(@B/314
Kff. Ð                UH H  3H lH 4 H  ]Ð        %  1 H
p          %p          %Bp          H  (1  U  o              Ð              Ð
h   H   H 5j   H   H ,^H S8H H H  H 5f   H   H   (H  H   H   4H ^H   H
```

cat ex1 을하면 이상한 문자들이 출력된다. 그 이유는 ex1 파일은 ex1.c 가 기계어로 변환된 파일이기 때문이다.

6-0) You can look at the machine code corresponding to the C code in ex1.c with "objdump".
$ objdump -D -M intel ex1 > x
$ vi x
Search for <main>.

```
ex1:     file format pei-x86-64


Disassembly of section .text:

0000000100401000 <WinMainCRTStartup>:
   100401000:   48 83 ec 28             sub    rsp,0x28
   100401004:   48 8d 0d 75 00 00 00    lea    rcx,[rip+0x75]        # 100401080
 <main>
   10040100b:   e8 d0 00 00 00          call   1004010e0 <cygwin_crt0>
   100401010:   45 31 c0                xor    r8d,r8d
   100401013:   31 d2                   xor    edx,edx
   100401015:   31 c9                   xor    ecx,ecx
   100401017:   e8 e4 00 00 00          call   100401100 <cygwin_premain0>
   10040101c:   45 31 c0                xor    r8d,r8d
   10040101f:   31 d2                   xor    edx,edx
   100401021:   31 c9                   xor    ecx,ecx
   100401023:   e8 e8 00 00 00          call   100401110 <cygwin_premain1>
   100401028:   45 31 c0                xor    r8d,r8dR [rdi+0x4e],al
   10040102b:   31 d2                   xor    edx,edx
   10040102d:   31 c9                   xor    ecx,ecxTR [rax],esp
   10040102f:   e8 ec 00 00 00          call   100401120 <cygwin_premain2>
   100401034:   45 31 c0                xor    r8d,r8d
   100401037:   31 d2                   xor    edx,edxR [rax],al
   100401039:   31 c9                   xor    ecx,ecx
   10040103b:   48 83 c4 28             add    rsp,0x28
   10040103f:   e9 ec 00 00 00          jmp    100401130 <cygwin_premain3>
   100401044:   90                      nop.RXB
   100401045:   90                      nop.XB
   100401046:   90                      nop.XB cmp spl,BYTE PTR [r8]
   100401047:   90                      nop    BYTE PTR [rdi+0x4e],al
   100401048:   90                      noph   rbp
   100401049:   90                      nop    DWORD PTR [rax],esp
   10040104a:   90                      nop    DWORD PTR [rcx],esi
   10040104b:   90                      nopxor al,0x2e
   10040104c:   90                      nop    BYTE PTR [rax],al
   10040104d:   90                      nop
   10040104e:   90                      nop
   10040104f:   90                      nop
   100403130:   47                      rex.RXB
0000000100401050 <__gcc_register_frame>:rex.XB
   100401050:   4c 8d 05 a9 0f 00 00    lea    r8,[rip+0xfa9]        # 100402000
 <__data_start__>8 47 4e                sub    BYTE PTR [rdi+0x4e],al
   100401057:   31 d2                   xor    edx,edx
   100401059:   48 8d 0d 10 00 00 00    lea    rcx,[rip+0x10]        # 100401070
 <__gcc_deregister_frame>               xor    DWORD PTR [rcx],esi
   100401060:   e9 4b 00 00 00          jmp    1004010b0 <__cxa_atexit>
   100401065:   66 66 2e 0f 1f 84 00    data16 cs nop WORD PTR [rax+rax*1+0x0]
   10040106c:   00 00 00 00
search hit BOTTOM, continuing at TOP                          9,82-86       Top
```

```
0000000100401080 <main>:
   100401080:   55                      push   rbp
   100401081:   48 89 e5                mov    rbp,rsp
   100401084:   48 83 ec 20             sub    rsp,0x20
   100401088:   e8 33 00 00 00          call   1004010c0 <__main>
   10040108d:   48 8d 05 6c 1f 00 00    lea    rax,[rip+0x1f6c]        # 1004030
00 <.rdata>
   100401094:   48 89 c1                mov    rcx,rax
   100401097:   e8 34 00 00 00          call   1004010d0 <puts>
   10040109c:   b8 00 00 00 00          mov    eax,0x0
   1004010a1:   48 83 c4 20             add    rsp,0x20
   1004010a5:   5d                      pop    rbp
   1004010a6:   c3                      ret
   1004010a7:   90                      nop
   1004010a8:   90                      nop
   1004010a9:   90                      nop
   1004010aa:   90                      nop
   1004010ab:   90                      nop
   1004010ac:   90                      nop
   1004010ad:   90                      nop
   1004010ae:   90                      nop
   1004010af:   90                      nop

00000001004010b0 <__cxa_atexit>:
   1004010b0:   ff 25 0a 70 00 00       jmp    QWORD PTR [rip+0x700a]        # 1
004080c0 <__IAT_start__>
   1004010b6:   90                      nop
   1004010b7:   90                      nop
   1004010b8:   90                      nop
   1004010b9:   90                      nop
   1004010ba:   90                      nop
   1004010bb:   90                      nop
   1004010bc:   90                      nop
   1004010bd:   90                      nop
   1004010be:   90                      nop
   1004010bf:   90                      nop
```

6-1) The compiler has translated the C statements in ex1.c into machine instructions and stored in ex1. Below is one such code (it might be different in your PC):

```
55              -- push rbp
48 89 e5        -- mov rbp, rsp
bf f0 05 40 00  -- mov edi, 0x40005f0
.........
```

Find and show the starting address of these machine instructions in ex1 with xxd. Use "/pattern" command in vi, e.g. /5548, or /55 48, etc., to search for them. Note that the actual code might be different. You should check it with objdump command as in 6-0).

$ xxd ex1 > x

$ vi x

7) Copy ex1.c to ex2.c, ex3.c, and ex4.c. Remove ex2.c. Rename ex3.c to y.c.

8) Make a subdirectory. Copy y.c in this subdirectory.

9) Redirect the output of ex1 to another file using ">" symbol and check its content with cat.

```
$ ./ex1 > f1
$ cat f1
```

10) Use grep to search "hello" in all files (use -nr option).

11) Find out what processes exist in your system. Use "ps -ef".

12) "ps -ef" shows all the processes in the Linux system (“ps -W” to see all processes including ones from Windows). How do you know which ones are running in the current terminal? Use "tty" for this purpose. Note that when a user logs in, the system allocates a terminal, and you can find the terminal number with "tty" command. What is your terminal number?

13) Modify ex1.c so that it receives two numbers from the user and prints the sum. Use scanf() for this.

14) Modify ex1.c so that it contains an infinite loop after printing "hello".

```
........
printf("hello");
fflush(stdout);   // to make it print hello immediately
for(;;);
```

.....

15) Run the program with & at the end, and use ps to check its status. "&" puts the process in the background so that you can type next command.
$ ./ex1 &
$ ps
16) Kill it with "kill" command.
17) Run the program again without & at the end. Open another login window, find out the process ID of the process running in the first window, and kill it.
18) Run following and tell the difference between gets and fgets

```
#include <stdio.h>
#include <string.h>
int main(){
    char buf[20];
    printf("enter a sentence₩n");
    gets(buf);
    printf("I got %s from you. length:%d₩n", buf, strlen(buf));
    printf("enter the same sentence again₩n");
    fgets(buf, 20, stdin);
    printf("I got %s from you. length:%d₩n", buf, strlen(buf));
}
```

19) Write a program to read a sentence and echo it as follows. Use gets() or fgets(). Search Internet to find out the usage of them (or Do "man gets" or "man fgets").

Enter a sentence
aa bcd e e ff aa bcd bcd hijk lmn al bcd
You entered aa bcd e e ff aa bcd bcd hijk lmn al bcd

20) Show the first 20 bytes of ex1.c in Problem 4 with xxd. Interpret them.

21) (For true Linux) [ELF format] An executable files in Linux follows ELF (Executable and Linkable Format) format as below. Show the first 20 bytes of ex1, the executable file (not ex1.c) in Problem 4, with xxd. Interpret them.

ELF format= ELF header + Program header table + Section 1 + Section 2 + ... + Section n + Section header table

ELF header =
e_ident(16)+ e_type(2)+ e_machine(2)+ e_version(4)+ e_entry(4)+ e_phoff(4)+ e_shoff(4)+ e_flags(4)+ e_ehsize(2)+ e_phentsize(2)+ e_phnum(2)+ e_shentsize(2)+ e_shnum(2)+ e_shstrndx(2)

e_ident=7f E L F + EI_CLASS(1) + EI_DATA(1) + EI_VERSION(1) + EI_OSABI(1) + EI_ABIVERSION(1) + EI_PAD(7)
EI_CLASS = 1 if 32bit application or 2 if 64bit application
EI_DATA = 1 if little endian or 2 if big endian
EI_VERSION = 1
EI_OSABI = 0 for System V, 1 for HP-UX, 2 for NetBSD, 3 for Linux, 4 for GNU Hurd, ...
EI_ABIVERSION = depends on ABI version
EI_PAD = 9 zero's

e_type= 1 for relocatable file, 2 for executable file, 3 for shared object file

e_machine = 3 for x386, 0x32 for IA-64, 0x3e for amd64, ...
e_version = 1
e_entry = program starting address
.....
(* Refer to https://en.wikipedia.org/wiki/Executable_and_Linkable_Format for the rest of ELF file format)

22) (For cygwin) [PE format] Executable file in PC is in PE format. Write ex1.c in cygwin and compile with gcc. Show the first 20 bytes of ex1.exe, the executable file (not ex1.c) with xxd. Interpret them. An executable file in Windows follows PE (Portable Executable) format as below (for detail of PE format, refer to Internet):

PE=dos header + image nt header + section table + sections

dos header = image dos header + dos stub

image dos header(64 byte)=

0,1: e_magic(2) : 4d 5a

2,3: e_cblp(2)

.....

3c-3f:e_lfanew(4) : offset of IMAGE_NT_HEADER

image nt header=pe signature(4) + file header + optional header

.....

23) (For Mac) Do 21) with Mach-O format.