12201922 이규민

2. Homework

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

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Pwd를 통해 현 위치 확인이 가능하고, ls를 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

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

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 text를 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.



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

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

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

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Ls -l 명령어를 사용하여 /usr/bin/passwd와 /etc/passwd의 차이점을 확인했다. Passwd.exe는 다른 그룹원이나 다른 사람들도 실행할 수 있는 권한이 있다. 하지만 passwd는 오직 오너만 실행할 수 있고, 그 외의 사람들은 읽기 권한밖에 없다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

File 명령어로 /usr/bin/passwd와 /etc/passwd의 차이점을 확인했다. Passwd는 텍스트 파일이지만 passwd.exe는 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

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Vi ex1.c명령어로 코맨드 모드 – 인서트 모드 진입 후 c언어로 코드를 작성하였다. 이후 gcc -o ex1 ex1.c 명령어로 ex1.c를 ex1으로 컴파일했다. 이후 ./ex1 명령어로 컴파일 된 프로그램을 실행했고, 정상적으로 메시지가 출력된다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Vi ex1.c명령어로 코맨드 모드 – 인서트 모드 진입 후 c++언어로 코드를 작성하였다. 이후 g++ -o ex1 ex1.c 명령어로 ex1.c를 ex1으로 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>".

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Cat ex1.c는 이전에 c++ 언어로 작성했던 코드 내용을 볼 수 있다. Xxd ex1.c는 작성했던 문자들을 16진수로 확인이 가능하다. 예를 들면 위 사진에서 초록색으로 2369 가 확인이 된다. 23 과 69가 16진수이고, 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?

텍스트, 스크린샷, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

cat ex1을하면 이상한 문자들이 출력된다. 그 이유는 ex1 파일은 ex1.c가 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>.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

objdump는 바이너리 분석 도구로 컴파일된 프로그램의 바이너리 코드를 분석하고 디버깅하는 데 사용된다. -D 옵션은 기계어로 변환된 내용을 어셈블리어로 볼 수 있게 하는 옵션 즉 디스어셈블하는 옵션이다. -M intel 옵션을 사용하여 인텔 문법으로 바꿔주었다.

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

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명

6-0번 문제에서 만든 x를 vi x로 진입하여 /rbp를하면 위 사진과 같이 검색이 된다.

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

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Cp 명령어를 사용하여 ex1.c 파일을 복사했다. Ex2, ex3, ex4가 복사된 파일이며, rm 명령어로 ex2.c 파일을 제거했다. Mv 명령어로 ex3.c파일의 이름을 y.c로 변경했다.

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

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Mkdir 명령어로 디렉토리를 만들었고 cp 명령어를 사용하여 하위 디렉토리 안에 y.c 파일을 복사하였다.

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

$ ./ex1 > f1

$ cat f1

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

./ex1 > f1은 ex1 실행프로그램에서 출력 되는 “hello” 메시지를 f1 파일의 내용으로 저장하라는 명령어다. F1에 다른 내용이 저장되어있었지만 위 사진과 같이 “hello” 메시지가 출력되는 것을 볼 수 있다.

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

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Grep 명령어를 통해 “hello” 메시지가 들어간 파일의 위치를 검색했다. 이전 문제에서 만들었던 f1 파일의 내용이 “hello”로 수정되었는데 그 내용도 확인이 된다.

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

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Ps -ef 를 이용하여 실행 중인 모든 프로세스를 자세하게 확인할 수 있다.

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?

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Ps -W 명령어를 사용하여 윈도우의 프로세스를 포함하여 모든 프로세스 확인이 가능하다. 또한 tty항목의 마지막 줄에 ‘pty0’라고 적혀있으며, tty 명령어를 사용해보면 동일하게 ‘pty0’라고 확인이 된다. 0번 터미널인 것을 알 수 있다.

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

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

Ex1.c의 코드는 a,b라는 수를 입력 받아서 더한 값을 출력하는 코드다. Ex2 파일로 컴파일했고, 프로그램 실행 시 정상적으로 덧셈이 되는 것을 확인했다.

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(;;);

.......

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Hello 출력 후 무한 루프에 빠지는 함수를 만들었다. For 문 안에는 아무 내용도 없다. 프로그램을 실행해보면 hello 출력 후 무한 루프에 빠져 ctrl+C 단축키로 강제 종료해주었다.

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

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

&이란 명령어를 앞의 명령어를 백그라운드로 동작시킬 때 사용한다. 위 사진의 경우 hello 메시지가 출력되고, 현재 프로세스를 볼 수 있다. ex1들이 실행되고 있음을 볼 수 있다.

16) Kill it with "kill" command.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Kill 명령어를 사용하여 ex1 실행 중이던 프로세스를 제거 했고, ps를 통해 ex1이 조회되지 않는 것을 볼 수 있다.

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.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 전자제품, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Ex1 프로그램을 실행 중인 상태로 다른 터미널 창을 열어서 ps 명령어로 프로세스 확인을 해봤다. Ex1이 실행 중임을 볼 수 있다. Kill 명령어로 ex1을 종료했다. 기존 터미널 창에서는 무한루프가 종료된 것을 볼 수있고, 두 번째 터미널 창에서 ps로 ex1이 조회되지 않는 것을 볼 수 있다.

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);

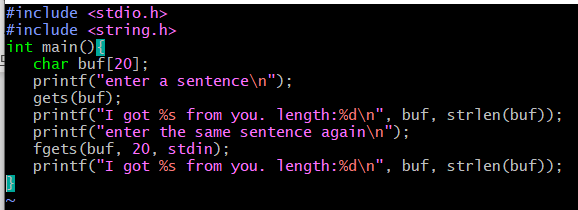
printf("I got %s from you. length:%d\n", buf, strlen(buf));

printf("enter the same sentence again\n");

fgets(buf, 20, stdin);

printf("I got %s from you. length:%d\n", buf, strlen(buf));

}

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

문자를 입력하고 엔터를 누른다면 \n(엔터)이 버퍼에는 줄바꿈으로 들어간다. Gets의 경우 줄바꿈을 \0(종료)로 바꿔준다. 그렇기 때문에 printf 사용 시 줄바꿈 없이 출력이 된다. 반면 fgets의 경우 \n 뒤에 \0(종료)가 들어간다. 그래서 printf를 해보면 줄바꿈이 생기는 것을 볼 수 있다.

위 사진의 경우 “hello”라는 문자를 입력 후 엔터를 눌렀다. Gets의 경우 “hello”만을 입력받았고, 문자 길이는 7로 확인이 된다. fgets의 경우 “message\n”을 입력받았기 때문에 줄바꿈이 되었고, 문자 길이도 8이다.

또한 20글자 이상 입력 시 gets는 모든 글자를 입력 받지만 fgets는 20글자 미만만 입력받는다.

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

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Gets 함수는 오버플로우가 발생할 수 있다는 단점이 있다. 그래서 fgets를 사용하여 코드를 작성했다.

우선 40글자까지 입력 받고 싶으나 fgets의 경우 버퍼에 \n(엔터)까지 입력이 되니 char buf[41]로 지정해줬다. Fgets로 입력을 받도록 했고, if문을 사용하여 버퍼 끝에 있는 \n(엔터)를 \0(종료)로 바꿔주었다. 이후 buf를 출력할 수 있도록 puts를 사용했다. 이후 정상적으로 출력되는 모습을 볼 수 있다.

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

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 평행, 다채로움, 스크린샷이(가) 표시된 사진

자동 생성된 설명

문제 4번의 코드 내용이 들어간 ex1.c를 xxd 명령어를 사용하여 확인했다. 첫 20바이트는 00000000: 2369 6e63 6c75 6465 203c이다. 여기서 한글자는 2바이트씩이다.

2369 = #i / 6e63 = nc / 6c75 = lu / 6465 = de / 203c = 공백< 이다.

Xxd를 통해 16진수로 내용을 확인할 수 있다.

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

.........................

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Ex1은 기계어로 변환된 내용이 들어갔기 때문에 내용이 매우 길다. 그래서 | head -5를 뒤에 붙여서 5줄만 출력되도록 했다. 그 중 20바이트인 부분은 00000000: 4d5a 9000 0300 0000 0400까지다. 16진수를 아스키 코드와 비교해보면 4d5a = MZ이지만 뒷부분은 90처럼 아스키코드에 없거나 00, 03처럼 제어문자라서 위 사진의 우측 부분에는 . 으로 표기된다.

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