

# REPORT

[Hw 2]



과 목 : 시스템소프트웨어 03

담당교수 : 석문기 교수님

학 과 : 컴퓨터공학과

학 번 : 2021111971

이 름 : 이재혁

# 문제 1

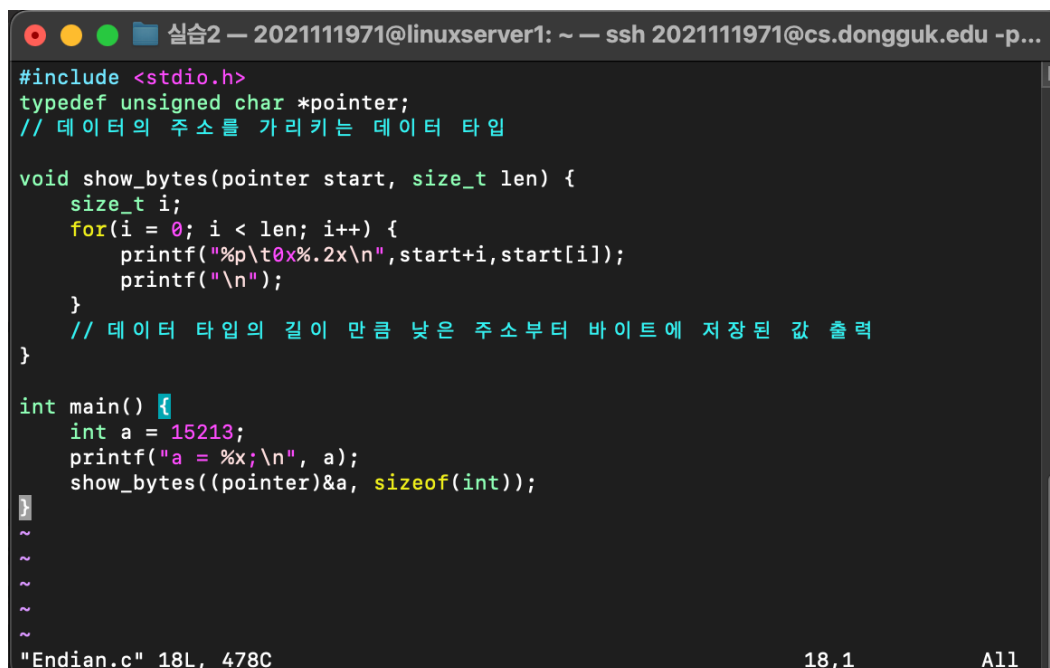
- c 코드를 작성하고 실행하여 현재 실습 머신이 리틀 엔디안인지 확인하고, 코드와 실행 결과 캡처를 첨부하세요.

소스코드

```
#include <stdio.h>
typedef unsigned char *pointer;
// 데이터의 주소를 가리키는 데이터 타입

void show_bytes(pointer start, size_t len) {
    size_t i;
    for(i = 0; i < len; i++) {
        printf("%p\t0x%.2x\n", start+i, start[i]);
        printf("\n");
    }
    // 데이터 타입의 길이 만큼 낮은 주소부터 바이트에 저장된 값 출력
}

int main() {
    int a = 15213;
    printf("a = %x;\n", a);
    show_bytes((pointer)&a, sizeof(int));
}
```



```
#include <stdio.h>
typedef unsigned char *pointer;
// 데이터의 주소를 가리키는 데이터 타입

void show_bytes(pointer start, size_t len) {
    size_t i;
    for(i = 0; i < len; i++) {
        printf("%p\t0x%.2x\n", start+i, start[i]);
        printf("\n");
    }
    // 데이터 타입의 길이 만큼 낮은 주소부터 바이트에 저장된 값 출력
}

int main() {
    int a = 15213;
    printf("a = %x;\n", a);
    show_bytes((pointer)&a, sizeof(int));
}
```

"Endian.c" 18L, 478C 18,1 A11

## 결과분석

```
[2021111971@linuxserver1:~$ vim Endian.c
[2021111971@linuxserver1:~$ gcc Endian.c -o Endian
[2021111971@linuxserver1:~$ ./Endian
a = 3b6d;
0x7ffe6d4a4a04  0x6d

0x7ffe6d4a4a05  0x3b

0x7ffe6d4a4a06  0x00

0x7ffe6d4a4a07  0x00

2021111971@linuxserver1:~$
```

a의 낮은 주소와 값을 출력할 때 하위 8비트가 먼저 저장되어 있는 것을 확인할 수 있습니다. 즉, Little Endian 방식으로 값을 저장합니다.

## 문제 2

- 다음을 지원하는 c 코드를 작성하고, 실행 결과를 출력하시오

- ▶ int a = -9; 일 때, 변수 a의 MSB(Most Significant Bit)가 1인지 확인
- ▶ a/8 와 a >> 3가 같은지를 확인
- 결과에 대한 설명 짧게 3줄 이내로 설명하시오.

## 소스코드

```
#include <stdio.h>

int main() {
    int a = -9;
    // 1을 4byte의 최상위 위치인 32번째 칸으로 이동
    // & 연산시 a의 최상위 비트의 결과가 출력된다.
    if (a & (1 << 31)) {
        // MSB == 1 : 음수
        printf("MSB : 1 (음수)\n");
    } else {
        // MSB == 0 : 양수
        printf("MSB : 0 (양수)\n");
    }
}
```

```
}

printf("a / 8: %d\n", a / 8);
printf("a >> 3: %d\n", a >> 3);
// division과 shift의 연산결과를 출력

if (a / 8 == a >> 3) { // 두 연산을 비교
    printf("a / 8 and a >> 3 are the same.\n");
} else {
    printf("a / 8 and a >> 3 are different.\n");
}
}

실습2 — 2021111971@linuxserver1: ~ — ssh 2021111971@cs.dongguk.edu -p...
#include <stdio.h>

int main() {
    int a = -9;
    // 1을 4byte의 최상위 위치인 32번째 칸으로 이동
    // & 연산시 a의 최상위 비트의 결과가 출력된다.
    if (a & (1 << 31)) {
        // MSB == 1 : 음수
        printf("MSB : 1 (음수)\n");
    } else {
        // MSB == 0 : 양수
        printf("MSB : 0 (양수)\n");
    }

    printf("a / 8: %d\n", a / 8);
    printf("a >> 3: %d\n", a >> 3);
    // division과 shift의 연산결과를 출력

    if (a / 8 == a >> 3) { // 두 연산을 비교
        printf("a / 8 and a >> 3 are the same.\n");
    } else {
        printf("a / 8 and a >> 3 are different.\n");
    }
}

1,1 All
```

## 결과 분석

```
[2021111971@linuxserver1:~$ nano Bit.c
[2021111971@linuxserver1:~$ vim Bit.c
[2021111971@linuxserver1:~$ gcc Bit.c -o Bit
[2021111971@linuxserver1:~$ ./Bit
MSB : 1 (음수)
a / 8: -1
a >> 3: -2
a / 8 and a >> 3 are different.
2021111971@linuxserver1:~$
```

[illegible]

## 결과분석

```
[2021111971@linuxserver1:~$ nano compare.c
[2021111971@linuxserver1:~$ vim compare.c
[2021111971@linuxserver1:~$ gcc compare.c -o compare
[2021111971@linuxserver1:~$ ./compare
x : 2147483647
y : 1
x + y < 0 입니다 .
-2147483648
```

x : 0111 1111 1111 1111 1111 1111 1111 1111 > 0

y : 0000 0000 0000 0000 0000 0000 0000 0001 > 0

x + y : 1000 0000 0000 0000 0000 0000 0000 0000 < 0 (overflow 발생)

## 문제 4

### • 아래의 항목이 항상 성립하는지 설명하시오.

#### ▶ 만족하지 않으면 반례만 제시

$$\bullet (x|-x) \gg 31 == -1$$

$$\bullet ux \gg 3 == ux/8$$

$$x \& (x-1) != 0$$

$$(x|-x) \gg 31 == -1$$

→ x가 0 일 때  $x|-x$  는 32개의 비트가 모두 0이다. 따라서 31번 shift연산을 시행해도 0 이다.

$$x \& (x-1) != 0$$

→ x가 8일 때

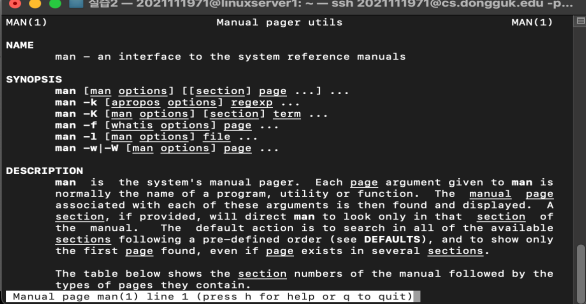
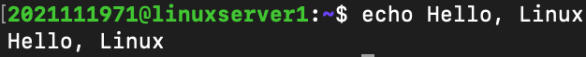
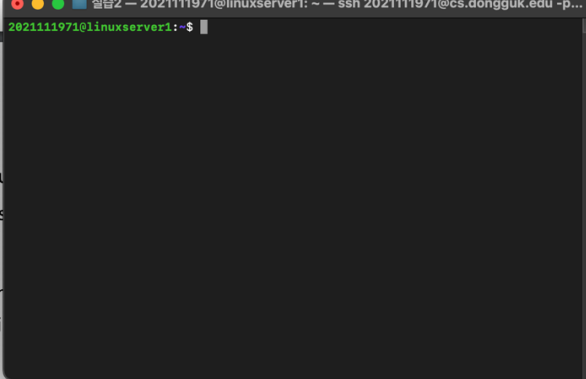
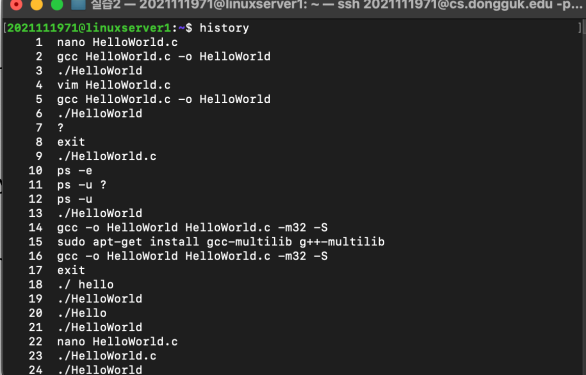
→ x : 0000 0000 0000 0000 0000 0000 0000 1000

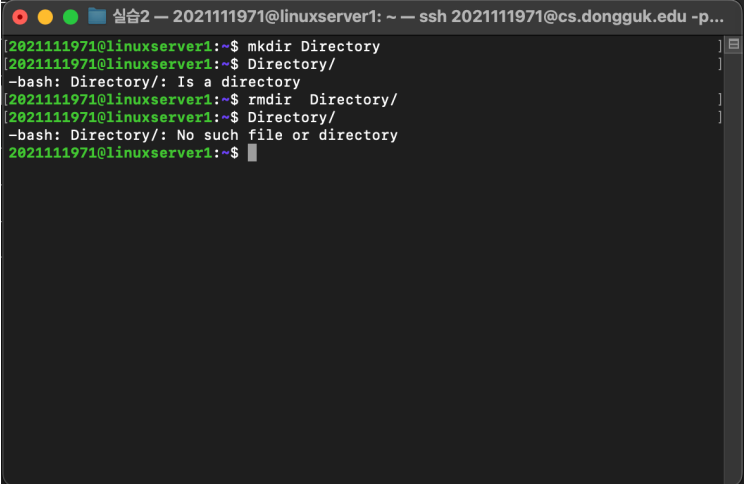
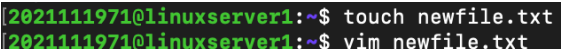
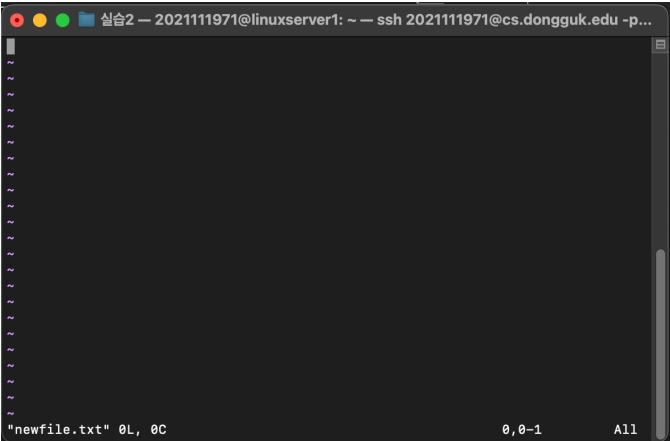
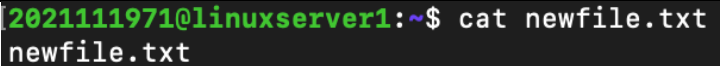
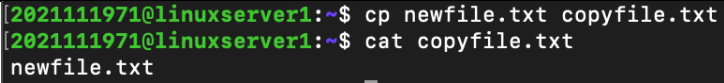
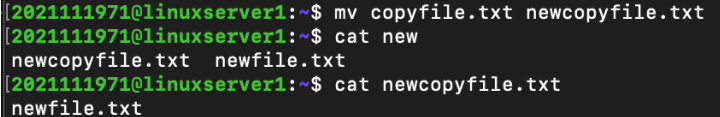
→ x - 1 : 0000 0000 0000 0000 0000 0000 0000 0111

→  $x \& (x-1) = 0$

## 문제 5

리눅스 서버 command 실습

man	명령어에 대한 상세한 설명	 <pre>MAN(1)                                Manual pager utils                                MAN(1)  NAME   man - an interface to the system reference manuals  SYNOPSIS   man [man options] [(section) page ...] ...   man -k [apropos options] regexp ...   man -K [man options] [section] term ...   man -f [whatis options] page ...   man -l [man options] file ...   man -w -W [man options] page ...  DESCRIPTION   man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order (see DEFAULTS), and to show only the first page found, even if page exists in several sections.    The table below shows the section numbers of the manual followed by the types of pages they contain.    Manual page man(1) line 1 (press h for help or q to quit)</pre>
echo	terminal 에 text 출력	 <pre>2021111971@linuxserver1:~\$ echo Hello, Linux Hello, Linux</pre>
clear	terminal 내용 지우기	 <pre>2021111971@linuxserver1:~\$ clear</pre>
history	이전에 작성한 command 출력	 <pre>2021111971@linuxserver1:~\$ history 1  nano HelloWorld.c 2  gcc HelloWorld.c -o HelloWorld 3  ./HelloWorld 4  vim HelloWorld.c 5  gcc HelloWorld.c -o HelloWorld 6  ./HelloWorld 7  ? 8  exit 9  ./HelloWorld.c 10 ps -e 11 ps -u ? 12 ps -u 13 ./HelloWorld 14 gcc -o HelloWorld HelloWorld.c -m32 -S 15 sudo apt-get install gcc-multilib g++-multilib 16 gcc -o HelloWorld HelloWorld.c -m32 -S 17 exit 18 ./ hello 19 ./HelloWorld 20 ./Hello 21 ./HelloWorld 22 nano HelloWorld.c 23 ./HelloWorld.c 24 ./HelloWorld</pre>

mkdir	Directory 생성	 <pre> [2021111971@linuxserver1:~\$ mkdir Directory [2021111971@linuxserver1:~\$ Directory/ -bash: Directory/: Is a directory [2021111971@linuxserver1:~\$ rmdir Directory/ [2021111971@linuxserver1:~\$ Directory/ -bash: Directory/: No such file or directory [2021111971@linuxserver1:~\$ </pre>
rmdir	Directory 삭제	
touch	새로운 파일 생성 or 파일의 time stamp 갱신	 <pre> [2021111971@linuxserver1:~\$ touch newfile.txt [2021111971@linuxserver1:~\$ vim newfile.txt </pre>  <pre> "newfile.txt" 0L, 0C 0,0-1 All </pre>
cat	주로 파일 내용 확인에 사용	 <pre> [2021111971@linuxserver1:~\$ cat newfile.txt newfile.txt </pre>
cp	파일 내용을 새로운 파일에 복사	 <pre> [2021111971@linuxserver1:~\$ cp newfile.txt copyfile.txt [2021111971@linuxserver1:~\$ cat copyfile.txt newfile.txt </pre>
mv	파일의 디렉토리 혹은 이름 변경	 <pre> [2021111971@linuxserver1:~\$ mv copyfile.txt newcopyfile.txt [2021111971@linuxserver1:~\$ cat new newcopyfile.txt newfile.txt [2021111971@linuxserver1:~\$ cat newcopyfile.txt newfile.txt </pre>