Week4 HW

12201922 이규민

1) Make a file, "f1", and fill it with more than 20 bytes.

$vi f1

I have a dream

that one day this nation

will rise up and

live out the true

meaning of its creed

that all men are created equal.

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

Vi f1으로 코맨드 모드에 진입하여 내용을 입력하였다.

2) Try the code in 6-0), 6-1), 6-2), 6-3), 6-4), 6-5). For 6-3) explain the strange output.

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

6-3) 코드를 실행하면 마지막에 “ are create”라는 문자가 추가로 붙는다. 그 이유는 write(1, buf, 20);에서 y가 아닌 20이라고 작성했기 때문이다.

Buf 배열은 저장되어있는 스트링보다 저장할 스트링의 길이가 짧다면 이전 문자열이 기록에 남는다. 즉 기존에 “t all men are create”라고 저장되어있다가 “d equal.\0 all ceate”라고 저장이 된다면 buf를 출력할 때 끝에 “ are create”가 추가로 출력되는 것이다.

3) Find the byte size of f2 with “ls –l f2”. Use xxd to find out the actual data stored in f2.

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명

Ls -l f2명령어를 사용하여 확인한 크기는 129 바이트다. Xxd를 사용하여서도 확인해봤다. 16진수로 하나에 1바이트다. 예를들어 4920 에서는 49와 20 이렇게 2바이트다. 총 개수를 세어보면 129 바이트로 확인된다.

3-1) Write a program that counts the number of bytes in f2. Compare it with the output of “ls –l f2”.

텍스트, 스크린샷, 폰트, 디스플레이이(가) 표시된 사진

자동 생성된 설명 텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

우선 읽은 문자열을 저장할 buf 배열을 만들었다. f2파일을 읽기모드로 열었다. 그리고 x 파일을 읽은 내용을 buf에 저장하고, 읽은 길이를 변수 R에 더해준다. 만약 읽은 글자 수가 200 이하라면 읽은만큼만 y에 더해줄 수 있다. 이렇게 무한 for문을 돌려주는데 위 코드처럼 y=R; / if(y==R) break; 명령어를 사용해줬는데 이는 R의 값에 변화가 없다면 무한 for문을 멈추라는 의미다. Y 값을 출력해보면 f1파일이 몇 바이트인지 알 수 있다.

4) Write a program "hw4.c" that opens f2 and shows each byte of it in hexadecimal number, decimal number, and character. Use printf("%x %d %c\n", ...............) to display a number in various format.

int x, y; char buf[20];

x=open("f2", O\_RDONLY, 00777); // open f2 for reading

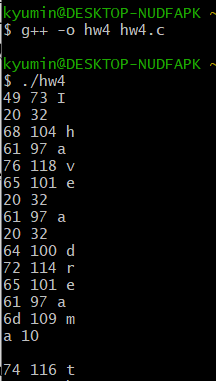
for(;;){

y=read(x, buf, 1); // read next byte

if (y==0) break; // we read all, exit the loop

printf("%x %d %c\n", buf[0], buf[0], buf[0]); // display

}



printf("%x %d %c\n",buf[0]…)을 사용하여 16진수 / 10진수 / 문자 형태로 출력이 가능하다.

5) Compile hw4.c with –g option and run gdb to execute each instruction one by one. Use “p” or “x” to check the value of a variable. For m1 mac, use lldb instead of gdb.

$ gcc -g -o hw4 hw4.c

$ gdb hw4

gdb) b main -- stop at main

gdb) r -- run

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

9 x=open("f2", O\_RDONLY, 00777); -- next line to execute

gdb) list -- show code list

gdb) n -- execute current line

11 y=read(x, buf, 1); -- line 9 has been executed. next is line 11

gdb) p x -- show x

$1 = 7 -- f2 is now file number 7

gdb) n

..........

gdb) p y

$2 = 1 -- we have read 1 byte

gdb) p buf[0]

$4 = 73 'I' -- assume we have 'I' in buf[0]

gdb) x/4xb buf -- show 4 bytes at buf in hexadecimal num

0x7fffffffe470: 0x49 0x06 0x40 0x00 -- we have 0x49=73='I' in buf[0]

(for lldb, you need an address to use x command.

p &buf -- print the address of buf

0x0016fdff496

x/4xb 0x0016fdff496 -- show 4 bytes at addr 0x0016fdff496

)

gdb) n

............ –- repeat a few times

gdb) list -- show rest of code

gdb) b 15 -- break point at line 15 (after loop)

gdb) c -- continue to that break point

gdb) q -- stop gdb

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Gdb 명령어는 디버깅 시 사용된다. 위 사진처럼 b main이라고 입력하면 main에 브레이크 포인트를 걸 수 있다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

R을 입력하여 프로그램을 실행했고, 이전에 브레이크 걸었던 main에서 멈춘 것을 알 수 있다.

List를 입력하면 해당 프로그램의 코드를 현재 위치부터 10줄까지 볼 수 있다.

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

자동 생성된 설명

N은 다음 코드를 실행하는 명령어다. P x는 x의 변수를 보여주는 명령어다. x/4xb는 Buf 배열에서 4바이트 즉 4글자를 16진수로 보여주는 명령어다.

다시 n을 입력하여 다음 코드를 실행하였다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

List는 현위치부터 10줄까지의 코드를 보여주는 명령어지만 남은 코드가 10줄이 되지 않으므로 8줄만 출력된 것을 볼 수 있다.

B 15는 15행에 브레이크 포인트를 거는 명령어다.

C는 다음 브레이크 포인트까지 실행하는 명령어다.

Q는 프로그램을 종료하는 명령어고 y를 한 번 더 눌러 종료했다.

6) Write a program that creates a file and writes “how are you doing?”in it. Use open() and write(). Confirm the result with "cat".

x = open("f3", O\_RDWR | O\_CREAT | O\_TRUNC, 00777); // create f3

write(x, "how are you doing?", 18); // write 18 bytes in f3

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

X = open(“f3”,O\_RDWR | O\_CREAT은 f3 파일을 읽기 및 쓰기 모드로 열고, 만약 파일이 존재하지 않는다면 파일을 생성하라는 의미다. O\_TRUNC는 파일에 이미 데이터가 존재한다면 데이터를 제거하라는 의미다.

Write는 x 파일에 문자열을 18 바이트만큼 입력하라는 의미다. 위에서 f3 파일을 열 때 읽기 및 쓰기 모드로 열었기 때문에 입력이 가능하다.

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명 텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

프로그램을 실행한 후 f3 파일의 내용을 확인해보면 정상적으로 입력이 된 것을 볼 수 있다.

6-1) Repeat Problem 6 but pass a string variable to "write" this time.

x = open("f3", O\_RDWR | O\_CREAT | O\_TRUNC, 00777); // create f3

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

write(x, y, strlen(y)); // y is a string variable that has "how ..." string

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

6번과 다른 점은 y라는 스트링 배열을 만들었고, y에 스트링을 입력한 후 그 내용을 x에 입력했다는 점이다. F3의 내용을 다른 내용으로 바꾼 후 다시 ex1프로그램을 실행해보면 f3의 내용을 “how are you doing?”으로 바뀐 것을 볼 수 있다.

7) Write a program that makes a copy for file "hw4.c" into another file "cphw4.c". Use open(), read(),and write(). Confirm that they are same with "cat" and "ls -l".

x1 = open hw4.c as O\_RDONLY

x2 = open cphw4.c as O\_RDWR | O\_CREAT | O\_TRUNC

for(;;){

y = read max 20 bytes from x1 into buf

if y is 0, break

write y bytes of buf into x2

}

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

X1은 읽기모드로 x2는 읽기 및 쓰기 모드로 열었다. 그리고 무한 for문을 만들어서 x1의 파일 내용을 buf 배열에 저장한 후 x2 파일에 저장하도록 만들었다. 그리고 읽는 글자가 없을 때 무한 for문을 빠져나오도록 만들었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

cat명령어를 사용해서 cphw4.c 파일의 내용을 확인해보면 Hw4.c 파일에 작성했던 코드랑 동일한 내용임을 알 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명



Ls -l 명령어로 파일을 확인해보면 동일한 크기로 파일이 생성된 것을 볼 수 있다.

8) Write a program that makes a copy for file "hw4" (the executable file for "hw4.c) into another file cphw4. Confirm that they are same with "xxd" and "ls -l".

Execute cphw4 to see if it runs ok.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Hw4파일을 읽고 buf 배열에 저장한다음 cphw4에 복사하는 코드이다.

텍스트, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

Xxd 명령어로 cphw4의 내용과 hw4의 내용을 16진수로 비교해봤을 때 동일한 것을 볼 수 있다.(길이가 길어 8줄만 출력하였다.)

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

두 파일의 크기가 동일한 것을 볼 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

두 파일 모두 정상적으로 실행되고 동일한 결과가 나오는 것을 볼 수 있다.

9) Repeat 7). But get the name of the files from the user. Confirm that the result of copy with "cat" and "ls -l".

Enter src file name

hw4.c

Enter dest file name

newhw4.c

hw4.c is copied into newhw4.c successfully.

텍스트, 스크린샷, 폰트, 디스플레이이(가) 표시된 사진

자동 생성된 설명

캐릭터 배열 S를 만들어 “new”라는 스트링을 저장하고, buf 배열에 파일명을 입력받았다. 그리고 strcat 명령어로 S 배열에 buf의 내용을 더했고, 그 S의 스트링을 파일 명으로 하도록 만들었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

파일을 실행한 후 hw4.c라고 입력하니 newhw4.c라는 파일이 생성되었고, 그 내용을 보면 hw4.c랑 동일한 내용임을 알 수 있다.

텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진

자동 생성된 설명

Ls -l 명령어로 hw4.c와 newhw4.c를 비교해보니 파일 크기가 동일한 것을 볼 수 있다.

10) Write "mycat" that displays the contents of a user-input file in the terminal in characters. Give a text file and a non-text file to mycat and explain the difference.

$./mycat

Enter file name

f1

The content of f1 is :

I have a dream

that one day this nation

will rise up and

live out the true

meaning of its creed

that all men are created equal.

$./mycat

Enter file name

hw4

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

텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

코드는 스트링을 입력 받은 후 buf 배열에 저장하고, 파일명이 buf의 스트링인 파일을 오픈한다. 그리고 그 x 파일을 20글자씩 읽어서 화면에 출력하도록 write(1,buf,y) 명령어를 사용했다. 여기서 1은 화면에 출력한다는 뜻이다.

그리고 프로그램 실행 후 f1 파일을 실행한 후 f1을 입력하면 f1의 내용들이 출력되는 것을 볼 수 있다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

프로그램 실행 후 Hw4입력하면 hw4의 내용이 출력된다. 그 파일은 실행파일이고, 그 안에는 기계어가 들어가있으므로 이상한 문자들이 출력된다.

11) Write "myxxd" that displays the contents of a user-input file in the terminal in hexadecimal numbers. Give a text file and a non-text file to myxxd and explain the difference. You need to use printf(“%x “, buf[i]) to display a byte in a hexadecimal number. Also declare the buffer as an array of unsigned char. Compare the result from the result of xxd.

$./myxxd

Enter file name

f1

The content of f1 is :

49 20 68 61 ..........

$ xxd f1

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

$./myxxd

Enter file name

hw4

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

$ xxd hw4

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

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Buf는 캐릭터 형태이므로 음수가 필요 없어 unsigned로 선언해줬다. 그리고 입력받은 내용을 buf에 저장하고, buf에 저장된 스트링의 파일명인 파일을 열었다. 그리고 그 파일의 내용을 buf에 저장하고, printf를 통해 문자를 16진수 형태로 출력했다.

텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

프로그램을 통해 f1을 16진수형태로 확인했고, xxd를 통해 16진수 형태로 출력했다. 두 내용은 동일한 것을 볼 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

프로그램을 통해 hw4을 16진수형태로 확인했고, xxd를 통해 16진수 형태로 출력했다. 두 내용은 동일한 것을 볼 수 있다.

12) Run following code and display f8 with cat and xxd respectively. Explain the results.

int x;

x=open("f8", O\_CREAT|O\_RDWR|O\_TRUNC, 00777);

write(x, "ab", 2);

int y=10;

write(x, &y, 4);

write(x, "cd", 2);

y=235;

write(x, &y, 4);

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

우선 코드에 대한 분석을 해보면 write(x, "ab", 2)는 ab 스트링을 가리키는 주소에서 2글자만큼 x에 저장하라는 의미다. 즉 x에 ab라는 문자를 저장하라는 뜻이다. write(x, &y, 4)도 y의 주소에서 4개의 글씨를 x에 저장하라는 의미다.

위 코드를 실행해보면 f8 파일이 생성된다. Cat 명령어로 그 파일의 내용을 확인해보면 ab가 출력이 되고 줄바꿈이 일어난 것을 볼 수 있다. Y의 값을 10으로 저장했었기 때문에 x에는 10 즉 개행문자(LF)가 추가되었기 때문에 줄바꿈이 일어난 것이다. 그리고 write(x, &y, 4)에서 4바이트를 저장하라고했으나 저장된 문자가 없어 내용이 입력되지는 않았다. 그리고 cd입력 후 문자는 아니지만 235라는 숫자가 저장되었다. 그래서인지 깨진 문자가 보인다.

F8 파일을 xxd로 확인해보면 0a00 0000과 eb00 0000이 보이는데 이는 y에 저장된 값은 하나인데 저장하라고 명령을 한 것은 4글자이니 00이 입력되었기 때문이다. 여기서 0a와 eb를 10진수로 환산해보면 10과 235다. 이는 코드에서 저장했던 y의 값이다.

12-1) Run following code and display f8 with cat and xxd respectively. Explain the results.

int x;

x=open("f8", O\_CREAT|O\_RDWR|O\_TRUNC, 00777);

write(x, "ab", 2);

int y=10;

write(x, &y, 8);

write(x, "cd", 2);

y=235;

write(x, &y, 8);

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이번 코드는 write(x, &y, 8)을 입력했기 때문에 xxd로 결과화면의 우측부분에 . 표시가 8개로 늘었다. 중간에 0300에서 03 이 어떤 의미인지는 알 수 없다.

13) Write a program that divides a given file into three small files of roughly equal size. Use fstat() to find out the size of a file.

Enter file name to split

f9

f9 is split into f91, f92, and f93.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

하나의 파일을 비슷한 사이즈인 세 개의 파일로 나누는 문제다.

그러기 위해서 우선 fName이라는 포인터에 메모리를 할당하였고, 원본 파일명과 나뉘어진 파일명을 fName에 저장했다.

원본 파일은 x로, 나뉘어진 파일들은 x1/x2/x3로 열었다.

Fstat 함수를 사용하여 원본 파일의 길이를 확인한 후 3등분이 되도록하는 글자 위치를 a, b, c에 저장해줬다. 만약 원본 파일 x의 길이가 129라면 a는 43, b는 86, c는 129라는 값이 저장되고, 그 값으로 x1/x2/x3에 3등분된 크기만큼 쓰도록 만들었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

프로그램을 실행한 후 f1을 입력하였다. 그 파일은 위 사진처럼 3등분되었고, ls -l을 통해 f1파일의 1/3 크기인 43 크기인 것을 확인할 수 있다.

13-1) Modify your code in Problem 13 so that the user can specify the number of small files.

Enter file name to split

f9

How many smaile files you want to split it into?

5

f9 is split into f91, f92, f93, f94, f95

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

몇 등분을 할지 사용자의 입력을 받아야하니 N이라는 변수를 만들어 입력을 받았다.

분할된 파일 이름은 fName배열에 저장하기 위해서 sprintf를 사용했다. 여기서 sprintf(fName[i],"%s%d",fName[0],i);는 기존 파일 이름에 숫자를 더하고, fName[i]에 저장하라는 의미다.

그리고 분할될 위치를 S변수에 저장해줬다. 원본 파일의 길이를 N으로 나누어 이전 S를 더해 저장해주고, 마지막인 S[N]은 원본의 길이를 저장한다. 만약 131글자의 파일을 3으로 나누어준다면 S[1]=43 S[2]=86 S[3]=131다.

변수가 j인 for문을 만들어서 분할 위치를 저장했던 S를 이용하여 분할된 파일에 한 글자씩 저장해준다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

프로그램을 실행해서 f1을 5등분했다. Cat을 통해 분할된 파일의 내용을 확인해보면 정상적으로 분할된 것을 볼 수 있고, ls -l을 이용하여 크기를 보면 5등분이 된 것을 볼 수 있다.

14) What is wrong with following program?

char temp0[20], \*temp1[10], \*temp2[10];

printf(“enter src file name\n”);

gets(temp0);

temp1[0]=temp0;

printf(“enter dest file name\n”);

gets(temp0);

temp2[0]=temp0;

printf(“src file:%s dest file:%s\n”, temp1[0], temp2[0]);

temp1과 temp2는 포인터고, 동적 메모리 할당이 되지 않았다. 그래서 temp1[0]=temp0;는 단순히 temp0의 스트링의 주소를 가르킬 뿐이다. Temp1과 temp2가 가리키는 주소는 temp0으로 동일하다. 그래서 temp1과 temp2를 출력해보면 동일한 결과가 나올 뿐이다. 이 문제를 해결하기 위해서는 포인터에 메모리 할당을 한 후 스트링을 저장하거나 temp0이 아닌 다른 캐릭터 배열을 만들어 입력을 따로 받으면 문제를 해결할 수 있다.

15) What is wrong with following program. Find the problem with GDB and fix it.

int x, x1, y;

x=open(“f1”, O\_RDONLY, 00777);

x1=open(“f2”, O\_WRONLY|O\_CREAT|O\_TRUNC,00777);

char buf[512];

int cnt=0;

for(;;){

y=read(x,buf,1);

if (y==0) break;

cnt++;

}

write(x1, buf, cnt);

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

위 코드는 buf를 충분히 큰 512 크기만큼 만들었다. read 명령어 사용 시 한 글자씩 읽도록만들었으나 buf에는 모든 내용이 저장되지 않는다. 단순히 buf[0]에만 글자가 저장될 뿐이다. 그래서 위 코드를 실행하면 f1 파일의 마지막 글자만 f2에 저장된다.

위 사진처럼 f1의 마지막 글자인 개행 문자만 저장되었다. 이상한 문자들이 출력된 것은 buf에는 첫 글자만 계속 수정이 되었기 때문에 f2에는 쓰레기 값이 들어간 것이다.

텍스트, 전자제품, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명

buf에는 첫글자만 개행문자가 들어가고, 이후는 쓰레기 값이 들어가있는 것을 볼 수 있다.

F1의 길이인 131이 cnt에 저장된 것을 볼 수 있다.