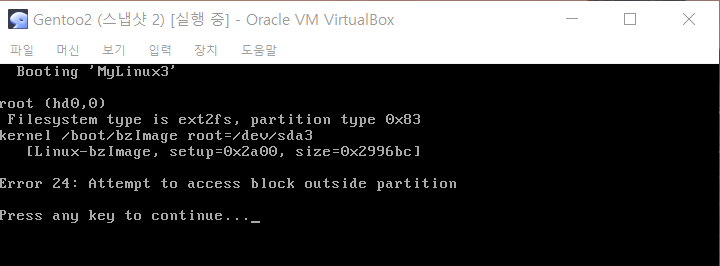
오퍼레이팅시스템 13주차 과제

1280626 성시열

(hw 1-1) Draw the memory map (process image) of the following program (ex1.cpp). What are the starting addresses of the code, data, heap, stack segment of this program and how many pages each segment occupies? What is the address of main function, the addresses of the global variables and local variables?



부팅창에서 에러가 발생하였다. grub.conf에 에러가 없는지 확인하기 바랍니다. mylinux2에서 우선 작업하였다.

ex1.cpp

#include <stdio.h>

int x;

int y[10000];

int main(){

int k;

int \*pk;

pk=new int;

printf(“ex1. &main:%p &x:%p &y:%p &y[9999]:%p &k:%p &pk:%p pk:%p\n”,

main,&x,&y,&y[9999],&k,&pk,pk);

for(;;); // to see memory map of this process

return 0;

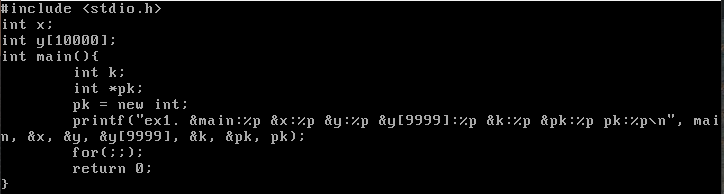
}

# g++ –o ex1 ex1.cpp

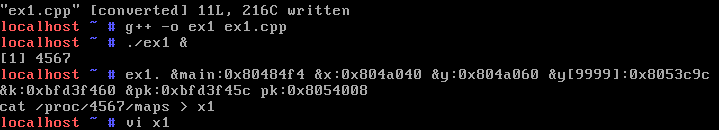
# ./ex1 &

# cat /proc/(pid of ex1)/maps > x1

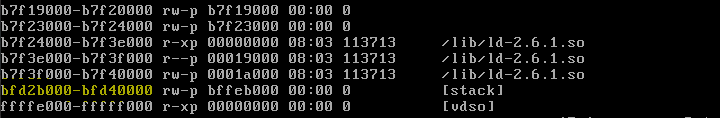
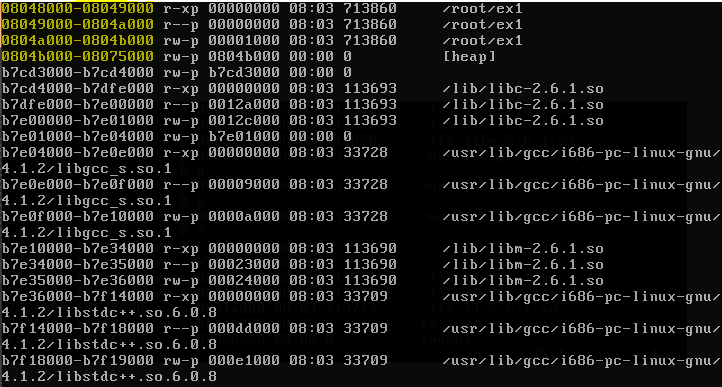
# vi x1



Ex1.cpp 코드를 작성하였다. For(;; );를 통해 무한 루프에 빠지게 한 모습이다.



이후 g++을 통해 컴파일 한 뒤 cat /proc/4567/maps > x1을 통해 x1에 ex1에 대한 정보가 저장된 모습이다.



Starting addresses of code -> 0804 8000

Starting addresses of data -> 0804 a000

Starting addresses of heap -> 0804 b000

Starting addresses of stack -> bfd2 b000

Page는 총 25개이다.

Main의 주소는 0804 84f4 이었고, global variable인 x는 0804 a040, y는 0804 a060부터 0805 3c9c이었다. Local variable인 k는 bfc0 0300, pk 주소는 bfc0 02fc 이었다. Pk의 포인터 주소는 0805 4008이었다. 즉 heap 공간 안에 있었다.

Main, 전역변수, 지역변수가 메모리 맵에 나오는 것과 일치함을 알 수 있다.

(hw 1-2) Write another simple program, ex2.cpp (see below), and run ex2, ex1 at the same time. Confirm they have the same address for main function. How can they run at the same location at the same time?

#include <stdio.h>

int x1;

int main(){

int \*pk1;

pk1 = new int;

printf(“ex2. &main:%p &x1:%p\n”, main,&x1);

for(;;); // to see memory map of this process

return 0;

}

# g++ -o ex2 ex2.cpp

#./ex2 &

...........

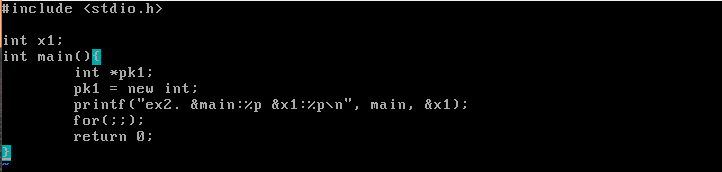
#./ex1 &

...........

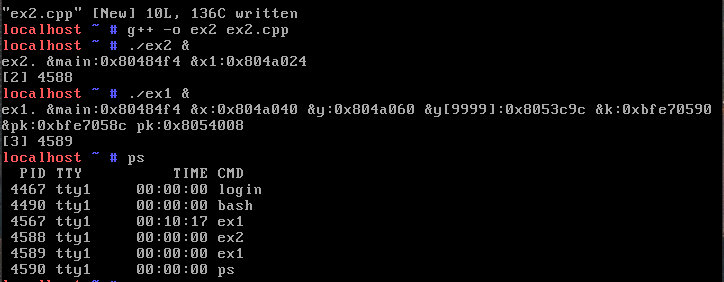
# ps

.............. ex2

.............. ex1



ex2.cpp를 다음과 같이 작성하였다.



이전 ex1 실행을 중지하지 않아 다음과 같은 결과가 발생하였다. 출력 결과를 확인하면 main의 주소는 같음을 알 수 있으나, 각자 logical 주소이기 때문에 두 개의 process가 같은 주소를 가질 수 있다. Logical address에서 각자 서로 다른 physical address를 가진다.

(hw 2) Show the memory map of the following program. Which pages does the program access during the run time? Show the page numbers that the program accesses in the order they are accessed. Indicate which pages are for code and which are for global data and which are for local data. To predict page numbers, you need to print the address of main, variable i and j, and the address of A[0], A[1], .., A[4].

ex4.c

int A[5][1024];

int main(){

int i,j;

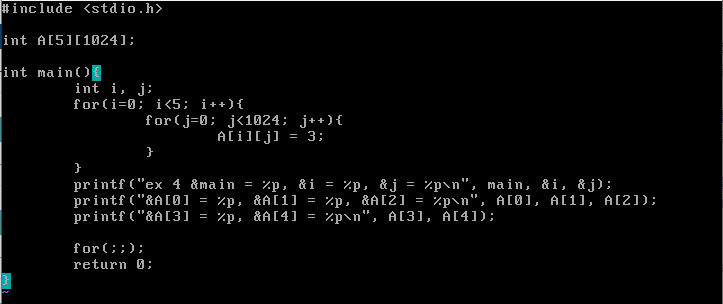
for(i=0;i<5;i++){

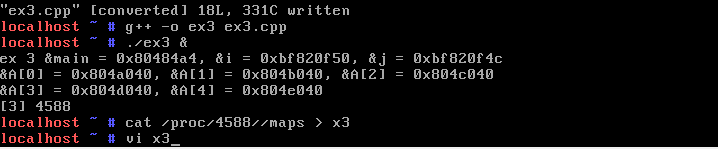
for(j=0;j<1024;j++)

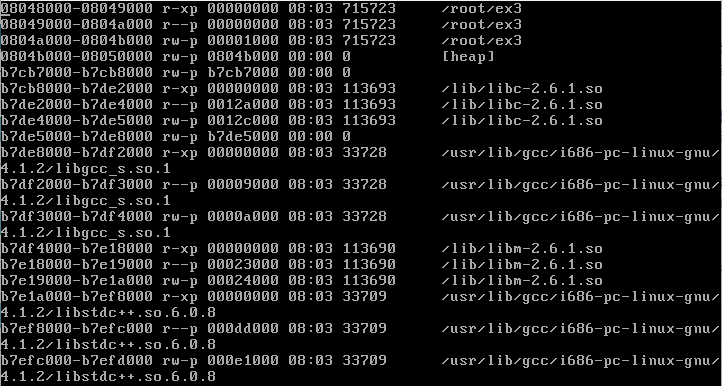
A[i][j]=3;

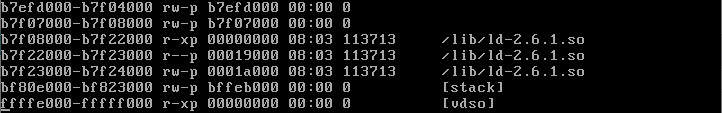
}

}









🡪code, main at 08048000 -08049000 page : 1개

🡪static data at 08049000-0804a000 page : 1개

🡪data,A[0] at 0804a000-0804b000 page : 1개

🡪heap, A[1] ~ A[5] at 0804b000-0805000 page : 5개

🡪stack, i, j at bfd3c000-bfd51000 page : 21개

|  |  |  |
| --- | --- | --- |
| &main | 함수 | 0x80483b4 |
| &i | 지역 변수 | 0xbfd50450 |
| &j | 지역 변수 | 0xbfd5044c |
| &A[0] | 전역 변수 | 0x804a040 |
| &A[1] | 전역 변수 | 0x804b040 |
| &A[2] | 전역 변수 | 0x804c040 |
| &A[3] | 전역 변수 | 0x804d040 |
| &A[4] | 전역 변수 | 0x804e040 |

해당하는 주소 안에 들어가 있음을 알 수 있다.

8048 h (main) → bf954h(i = 0) → bf954h(j = 0) → 804ah(A[0][0] = 3) → bf954h (j = 1) → 804ah(A[0][1] = 3) → … → bf954h (i = 1) → bf954h (j = 0) → 804bh(A[1][0] = 3) → … → bf954h (i = 2) → bf954h (j = 0) → 804ch(A[2][0] = 3) → … → bf954h (i = 3) → bf954h (j = 0) → 804dh(A[3][0] = 3) → … → bf954h (i = 4) → bf954h (j = 0) → 804eh(A[4][0] = 3) → … → 804fh(A[4][1023] = 3)

hw 3) How many page faults will the program in hw 2) generate? Explain your reasoning. Remember in the beginning the system has no page of the current process in the memory.

804a (A[0][0]=3)

804b(A[1][0]=3)

804c(A[2][0]=3)

804d(A[3][0]=3)

804e([4][0]=3)

총 5번 발생할 것이다.

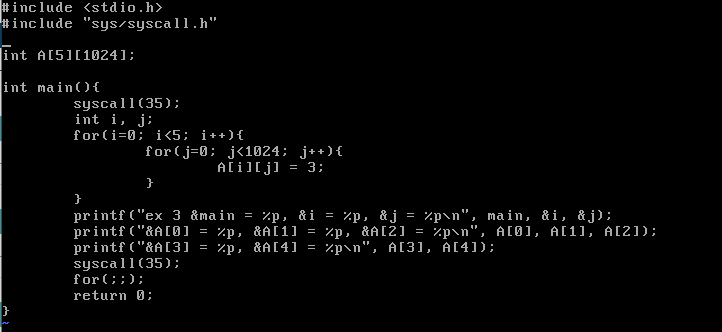
hw 4) Confirm your answer in hw 3) by defining a new system call, sys\_show\_pfcnt(), in mm/mmap.c, which displays the number of page faults generated so far.

extern int pfcnt;

void sys\_show\_pfcnt(){

printk("page fault count so far:%d\n", pfcnt);

}



Syscall 함수 등록하고 찾기부터 진행하지 못하였습니다.

hw4-1) You can display the exact address where page fault has happend. Make ex3.c and insert following code (in italic) in arch/mm/fault.c:do\_page\_fault(). When you run ex3, the kernel will display the page fault addresses generated by ex3. Explain the result. Also confirm your answer in hw3) again by examining page fault addresses.

숙제 4-1)

어떤 페이지 폴트가 어떤 코드/데이타 를 메모리에 반입했는지 밝혀야 합니다. 예를 들어 main은 어떤 폴트 주소때문에 반입되었는지, x 는 어떤 폴트 주소때문에 반입되었는지 등등. 폴트 주소가 많으므로 출력을 옆으로 하는 등 dmesg 로 모두 볼 수 있도록 하기 바랍니다. 이 작업은 hw3)에 있는 코드에 대해서도 수행해야 합니다.

ex3.c:

int x;

void main(){

x=3;

}

In kernel arch/x86/mm/fault.c:

void do\_page\_fault(...........){

...........

/\* get the address \*/

address = read\_cr2();

*if (strcmp(tsk->comm, "ex3")==0){*

*printk("pg fault for ex3 at:%p\n", address);*

*}*

*..............*

hw4-2) Repeat hw4) with modified hw2) code as below. Why is pfcnt increased?

int A[5][1024];

int main(){

int i,j;

for(i=0;i<5;i++){

printf("&A[%d][0]:%p\n", i, &A[i][0]); // add this line

for(j=0;j<1024;j++)

A[i][j]=3;

}

}

hw 5) Make a system call that prints vma information of the current process, and write a user program that displays the VMA list with it. Confirm that this result matches to those in /proc/xxxx/maps.

- Use system call 31 which is not used currently.

- Modify the system call table so that system call 31 is redirected to sys\_get\_VMAlist.

- Provide sys\_get\_VMAlist() in mm/mmap.c (**not** arch/x86/mm/mmap.c). This function will display all vma's of the current process.

struct vm\_area\_struct \*temp=current->mm->mmap;

for(;;){

if (temp==NULL) break; // we are done

display temp->vm\_start, temp->vm\_end,

temp->vm\_file->f\_dentry->d\_name.name;

temp=temp->vm\_next;

}

- Write a user program (e.g. xx.c) that invokes system call 31

hw 6) Count the number of page faults when you run following ex1 and ex2 by using sys\_show\_pfcnt(). Explain the results. Also compare the running time of each code (use gettimeofday() function) and explain why they differ. Run several times and compute the average.

double getUnixTime(){

struct timeval tv;

gettimeofday(&tv, (void \*)NULL); // get current time

return (tv.tv\_sec + tv.tv\_usec/1.0e6);// return it in seconds

}

……

double stime, etime, diff;

stime=getUnixTime(); // starting time

…… code …….

etime=getUnixTime(); // ending time

diff=etime-stime; // the difference

printf(“the elapsed time:%f\n”, diff);

ex1.c

#include <unistd.h>

#include <sys/time.h>

int A[8192][8192];

double getUnixTime(){

…………….

}

void main(){

int i,j;

call getUnixTime() and remember the stime…….

syscall(17); // display pfcnt

for(i=0;i<8192;i++){

for(j=0;j<8192;j++){

A[i][j]=3;

}

}

syscall(17); // display pfcnt again

call getUnixTime() and compute diff and print it…….

ex2.c

same as ex1.c except

change A[i][j]=3; to A[j][i]=3;

(If your vm dies, reduce the array size)