1. 개요

리눅스 환경에서 Stack과 heap이 같은 메모리 segment를 쓰고 있음을 각각의 크기를

달리하여 확인해본다.

2. 프로그램 구조 설명

2.1 함수에 대한 설명

-segment.c

void mkHeap(int\* heapMax)

//while문으로 힙을 지속적으로 만드는 함수입니다. 쓰레드로 작동합니다.

//인자로 힙의 최대 크기(MB)가 담긴 변수의 주소를 받아옵니다.

//heapMax가 -1이면 힙의 최대 크기 제한이 없는 상태로 충돌이 일어날 때까지 힙을 생성합니다.

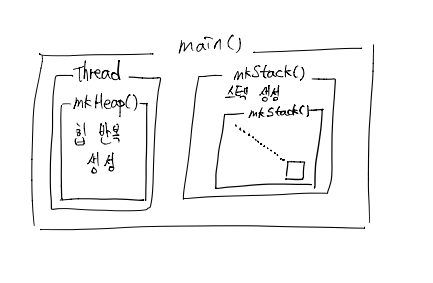
void mkStack(int stackMax)

//recursive 함수를 써서 stack을 지속적으로 생성합니다.

//인자로 stack의 최대 크기(MB)를 받아옵니다.

//stackMax가 -1이면 stack의 최대 크기 제한이 없는 상태로 충돌이 일어날 때까지 stack을 생성합니다.

2.다이어그램



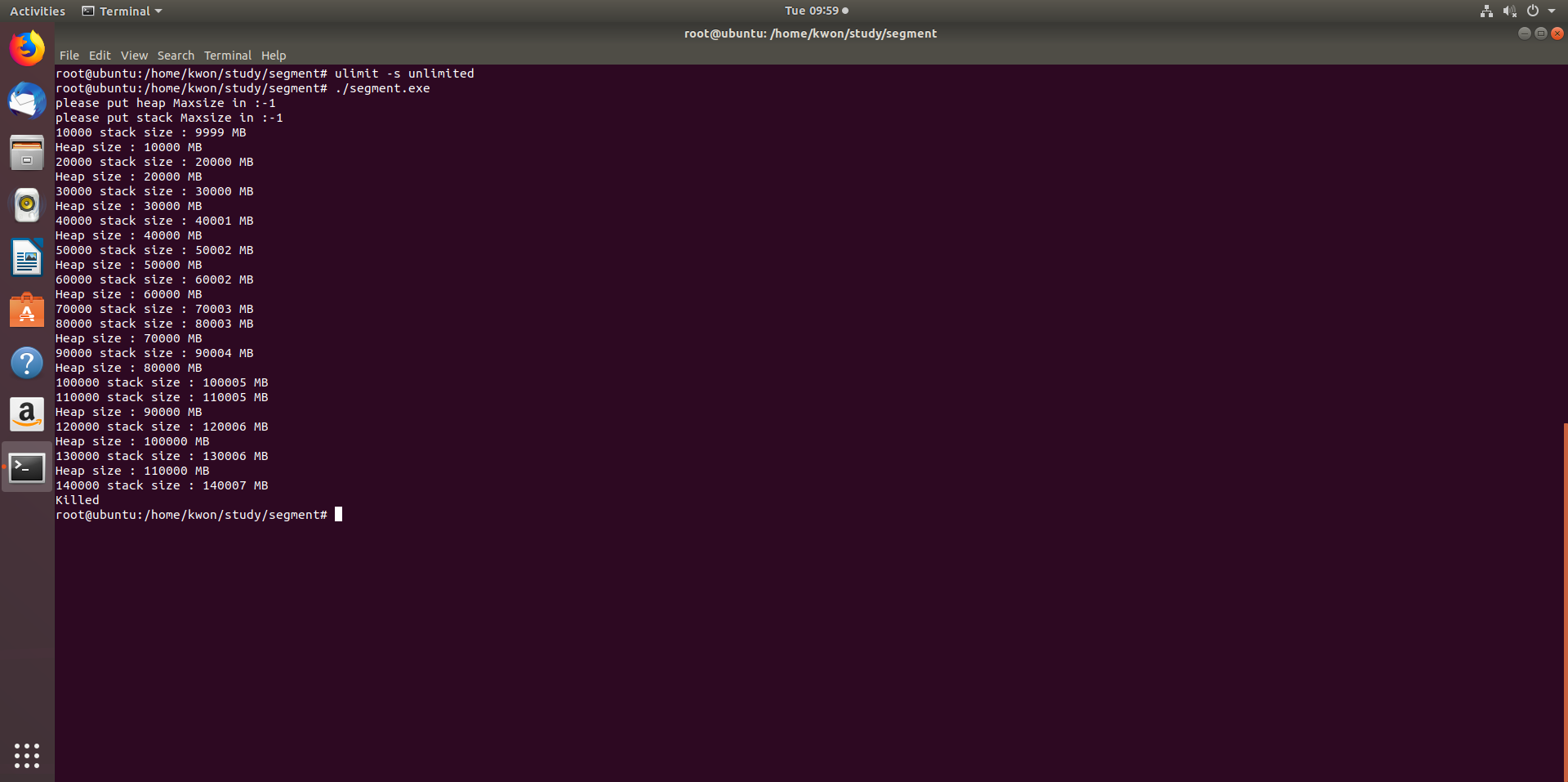
3. 실행 결과

저는 ulimit을 사용해서 스텍의 크기를 unlimited로 설정했습니다.

프로그램에서 입력값이 -1이면 최대크기 제한이 없는 것을 의미합니다.

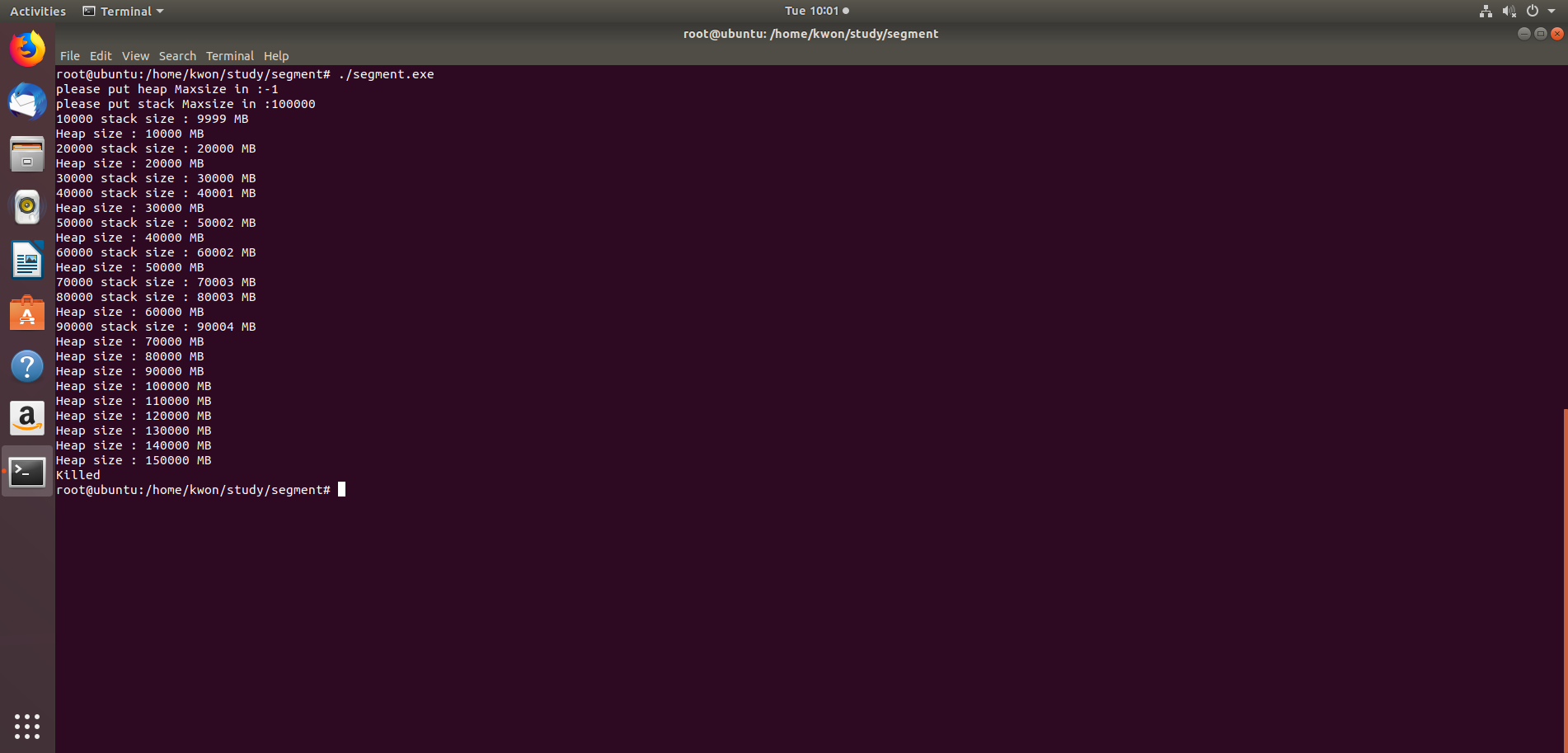
1.스텍과 힙을 제한 없이 생성합니다.

Heap size : 110000MB , Stack size : 140007MB, Data Segment : 250000MB (추정치)



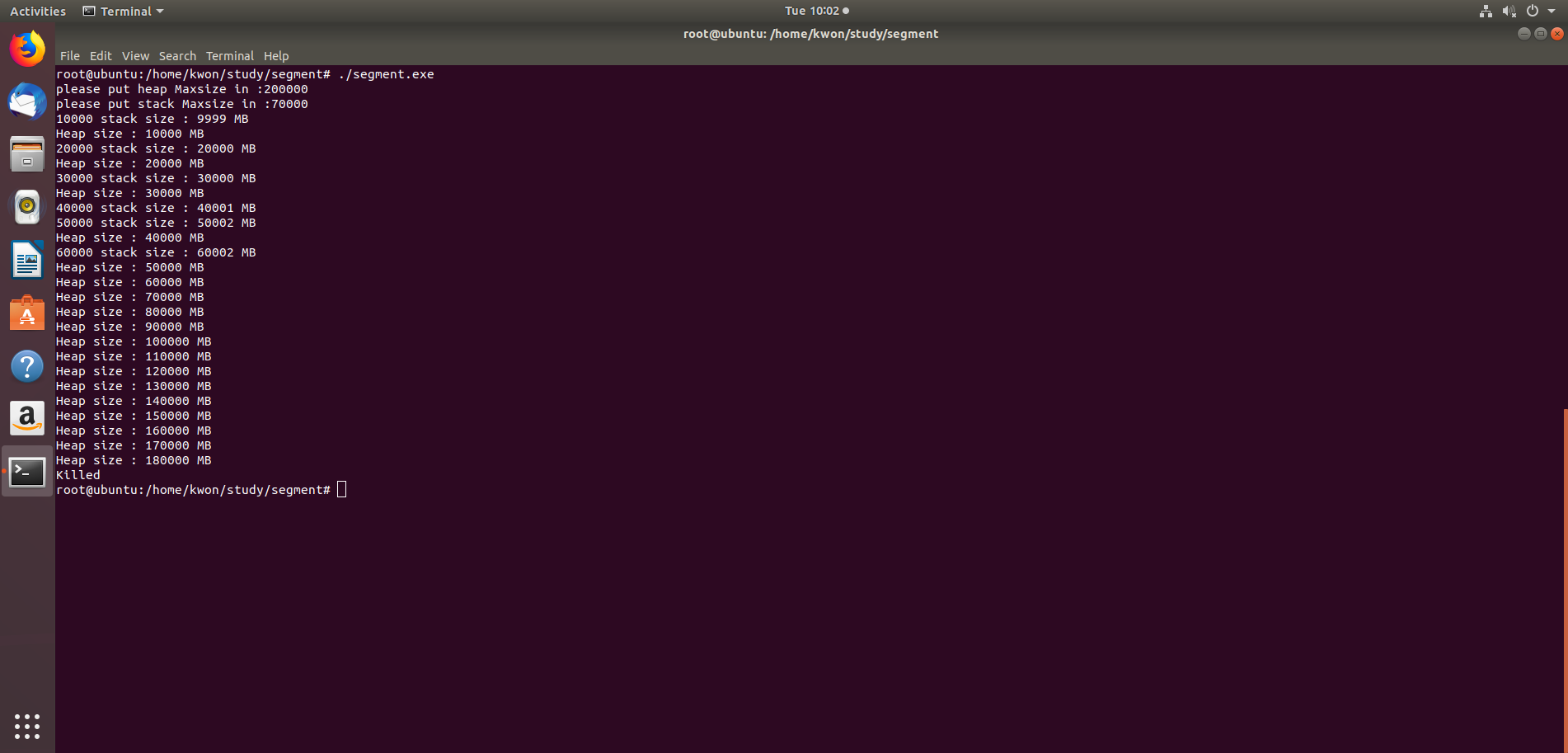
2. 힙의 크기를 제한없이, 스텍의 크기를 100000MB로 뒀을 때

Heap size : 150000MB , Stack size : 90004MB, Data Segment : 240000MB (추정치)



3. 힙의 크기를 200000, 스텍의 크기를 70000MB로 뒀을 때

Heap size : 180000MB , Stack size : 60002MB, Data Segment : 240000MB (추정치)



충돌 지점은 계속 달라지지만 총 segment는 거의 같다는 것을 알 수 있음

**(실행결과는 자신의 컴퓨터환경에서 실행한 결과 캡쳐(그림파일)해서 문서에 첨부하여 제출합니다. 실행결과가 성공한 경우뿐만 아니라 실패한 경우도 포함되어야 함.**

**만약 과제로 올려서 실행한 결과와 실행 결과로 캡쳐한 결과가 다를 경우 감점합니다.)**

4. 고찰 (제일 중요함 - **점수배점이 가장 높음!** 자세히 쓸 것)

이번 과제는 운영체제 수업 중에 했던 프로그래밍 중에 가장 힘들었습니다. 어떻게 짜야 과제의 취지에

맞게 코드를 짤 수 있을까 고민을 많이 했습니다. 하지만 가장 힘들었던 것은 스텍이 8MB이상

커지지 않는다는 것이었습니다. 처음에는 이론과 실제가 다를 수도 있다는 생각을 했었습니다.

고민을 많이 했지만 뒤늦게 공지사항에 올라온 HW4 추가설명을 보고 프로그래밍을 진전 시킬

수 있었습니다. 또 저를 힘들게 했던 것 중 하나는 분명 프로그램을 다 만들고 정상실행되는 것

을 확인했었으나 우분투를 재부팅한 후에 다시 실행하니 segmentation fault 오류가 뜨는 것이

었습니다. 한참을 헤매고 헤매다 ulimit는 한번 설정하면 영구 지속이 아니라 새로 부팅 할 때

마다 설정해줘야 한다는 것을 알았습니다. 이번 과제를 하면서 정말 직접 코딩을 해보는 것이

굉장히 중요하다는 것을 다시 한번 깨닫았습니다. 이번 코딩을 하지 않았더라면 우분투에서 스텍

크기가 제한되는 것도 몰랐을 것이고 ulimit에 대한 것도 몰랐을 것입니다. 항상 배운 것이

있으면 바로 실습을 해보는 것을 습관으로 삼도록 하겠습니다.

5. 프로그램 소스 파일 (주석을 꼭 달 것)

-main.c

#include <stdio.h>

#include <pthread.h>

#define ONE\_MEG 1048576

long\* firstHeap;

//쓰레드로 동작하는, 반복적으로 힙을 만드는 함수입니다.

void mkHeap(int\* heapMax) { // MB

int heapSize = 0;

while (\*heapMax >= heapSize || \*heapMax == -1) { //힙 최대치보다 작으면 힙생성

firstHeap = (long\*)malloc(ONE\_MEG);

if (firstHeap == 0) { //힙 생성 실패

printf("The program is ending because we could allocate no more memory.\n");

printf("final Heap size : %d\n",heapSize);

exit(0);

}

heapSize++;

if(heapSize%10000 == 0){ //10000개 단위로 출력합니다.

printf("Heap size : %d MB\n",heapSize);

}

}

pthread\_exit(0);

}

unsigned long firstStackLocation;

int recursiveDepth;

#define STACK\_ALLOC ONE\_MEG

//recursive 함수로 스텍을 반복 생성합니다.

void mkStack(int stackMax) {//MB

recursiveDepth++;

char temp[STACK\_ALLOC];

unsigned long size = firstStackLocation-(unsigned long)&(temp[STACK\_ALLOC]);

if(recursiveDepth%10000 == 0){ // 10000개 단위로 출력

printf("%d stack size : %d MB\n",recursiveDepth,size/ONE\_MEG);

}

if(size/ONE\_MEG < stackMax || stackMax == -1){ //스텍 최대치보다 작으면 실행

mkStack(stackMax);

}else{

while(1){ //스텍 최대치보다 커지면 대기합니다.

}

}

}

void main() {

int stack; // 최초의 스텍

recursiveDepth = 0; //recursive 횟수

int stackMax;

int heapMax;

pthread\_t id;

//최대치 입력 받기

printf("please put heap Maxsize in :");

scanf("%d",&heapMax);

printf("please put stack Maxsize in :");

scanf("%d",&stackMax);

firstStackLocation = &stack;

pthread\_create(&id,NULL,mkHeap,&heapMax); //힙 쓰레드 시작

mkStack(stackMax); // 스텍 함수 시작

pthread\_join(id,NULL);

}

6. 자료 출처 (어떤 책을 참고했으며 어떤 웹사이트를 참고했는지 출처를 적습니다)

<https://adnoctum.tistory.com/387>

<https://www.morenice.kr/66>

등등을 참고했습니다.