

Assignment 1

2015313254 노인호

September 28th 2019

1 Environmnet

코드는 다음과 같은 환경에서 실행되었다.

- Ubuntu 16.04 LTS 64bit
- gcc 5.4.0

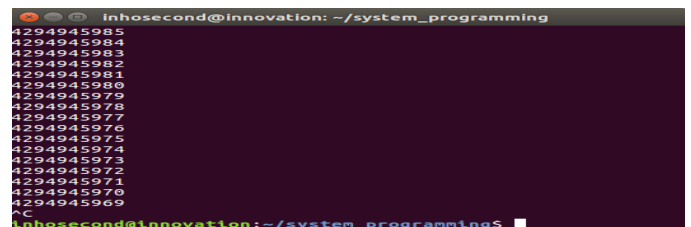
2 Code-1

2.1 Problem

먼저 문제가 되는 코드는 다음과 같다.

```
#include<stdio.h>
int main(){
    unsigned i;
    for (i=10;i>=0;i--){
        printf("%u\n",i);
    }
}
```

Listing 1: Code-1.c



```
Inhosecond@innovation: ~/system_programming
4294945985
4294945984
4294945983
4294945982
4294945980
4294945979
4294945978
4294945977
4294945976
4294945975
4294945974
4294945973
4294945972
4294945971
4294945970
4294945969
nC
Inhosecond@innovation:~/system_programming$
```

Figure 1: Executing result of Code-1.c

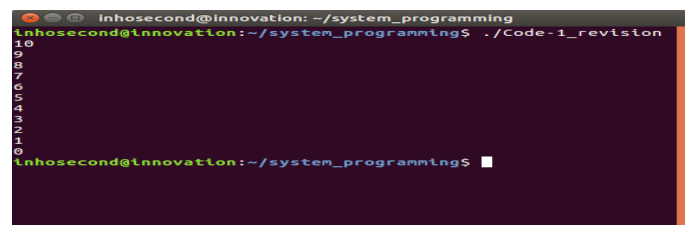
이 코드를 실행시키면 출력되는 값은 다음 그림과 같이 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 4294967295, 4294967294, 4294967293, ... , 5, 4, 3, 2, 1, 0, 4294967295, 4294967294, ... 이런식으로 무한으로 출력되게 된다. for문에서 i가 0이 될 때, 출력을 수행하고, 다음 조건식에서 1을 빼주어 i=-1이 되는데 i는 unsigned형 이므로 T2U 즉 2의 보수(signed)에서 비부호형(unsigned)으로 형변환이 일어나게 된다. 그래서 $i=-1+2^{32}=4294967295$ 가 되고(unsigned는 32bit, 64bit 상관없이 4byte=32bit) 이 값은 for문의 조건문을 만족시키기 때문에 for을 무한히 돌게 된다.

2.2 Revision

원래 의도한 대로 코드를 unsigned i 에서 int i로 수정하고 실행한 결과는 다음과 같다.

```
#include<stdio.h>
int main(){
    int i;
    for (i=10;i>=0;i--){
        printf("%d\n",i);
    }
}
```

Listing 2: Code-1_revision.c



```
Inhosecond@innovation: ~/system_programming
Inhosecond@innovation:~/system_programming$ ./Code-1_revision
10
9
8
7
6
5
4
3
2
1
0
Inhosecond@innovation:~/system_programming$
```

Figure 2: Executing result of Code-1_revision.c

3 Code-2

3.1 Problem

먼저 문제가 되는 코드는 다음과 같다.

```
#include <stdio.h>

int sum_array(int a[], unsigned len){
    int i;
    int result=0;
    for(i=0;i<=len-1;i++){
        result +=a[i];
    }
    return result;
}

int main(){
    int A[10];
    int i;
    for(i=0;i<10;i++){
        A[i]=i+1;
    }
    int final=sum_array(A,0);
    printf("%d\n", final);
}
```

Listing 3: Code-2.c

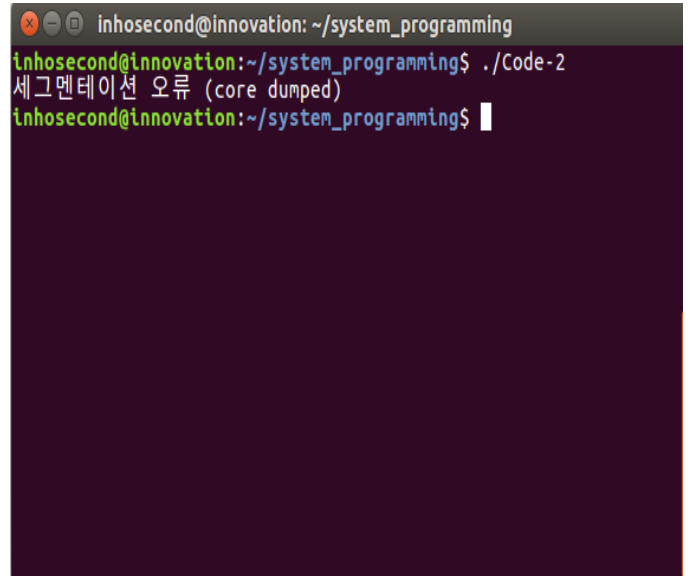


Figure 3: Executing result of Code-2.c

이 코드를 실행시키면 세그멘테이션 오류가 뜬다. 그 이유는 sum_array함수의 len인자가 unsigned형이기 때문이다. for문에서 len-1이 비부호형과 부호형의 연산이므로 결과는 비부호형으로 변환되어 -1은 T2U 즉 2의 보수(signed)에서 비부호형(unsigned)으로 형변환이 일어나게 된다. 그래서 $len-1 = -1 + 2^{32} = 4294967295$ 가 되고 결국 행렬의 범위를 초과하는 index값을 갖게 되면서 메모리 오류가 난다.

3.2 Revision

이런 문제를 방지하기 위해 코드를 unsigned len 에서 int len으로 수정하고 실행한 결과는 다음과 같다.

```
#include <stdio.h>

int sum_array(int a[], int len){
    int i;
    int result=0;
    for(i=0;i<=len-1;i++){
        result +=a[i];
    }
    return result;
}

int main(){
    int A[10];
    int i;
    for(i=0;i<10;i++){
        A[i]=i+1;
    }
    int final=sum_array(A,0);
    printf("%d\n", final);
}
```

Listing 4: Code-2_revision.c

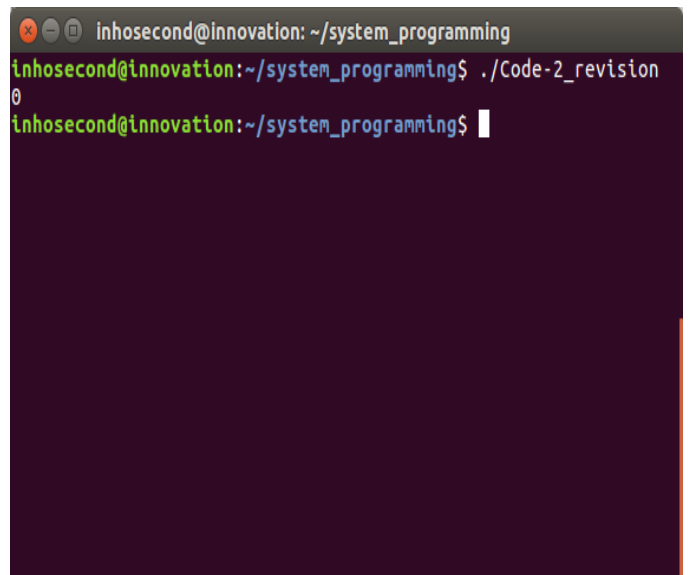


Figure 4: Executing result of Code-2_revision.c

4 Code-3

4.1 Problem

먼저 문제가 되는 코드는 다음과 같다.

```
#include <stdio.h>
#include <string.h>

int strlonger(char *s, char *t){
    return strlen(s)-strlen(t)>0;
}

int main(){
    char s1[10]="Korea";
    char t1[10]="America";

    int diff=strlonger(s1,t1);
    printf("%d\n",diff);
}
```

Listing 5: Code-3.c

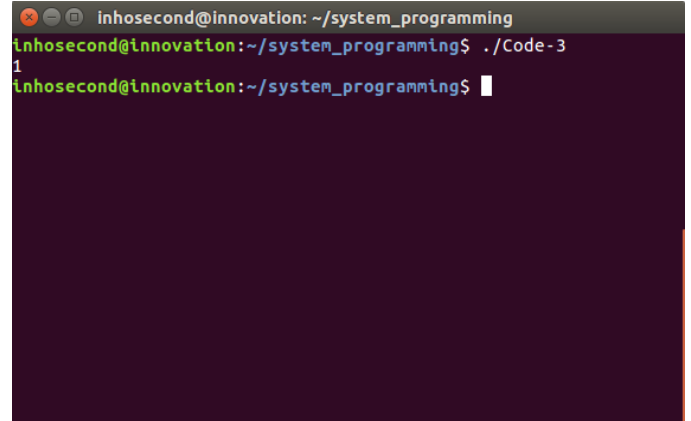


Figure 5: Executing result of Code-3.c

s1의 길이가 5고 t1의 길이가 7이어서 $5-7 > 0$ 이 거짓이 되어 strlonger함수에서 0을 리턴하고 diff는 0을 출력해야 하지만 1이 나왔다. 이유는 strlen은 size_t데이터 타입을 사용하는데 size_t가 unsigned로 정의되어 있어 비부호형-비부호형으로 연산결과가 비부호형이 나오기 때문이다. 여기서는 $5-7 = -2$ 는 T2U 즉 2의 보수(signed)에서 비부호형(unsigned)으로 형변환이 일어나게 된다. 그래서 $-2 + 2^{32} = 4294967294$ 가 되고 조건이 참이 되므로 strlonger함수는 1을 리턴하게 된다.

4.2 Revision

strlen(t)부분을 부등호 뒤로 넘겨서 비부호형간의 연산이 일어나지 않도록 하면 문제가 해결된다. 실행한 결과는 다음과 같다.

```
#include <stdio.h>
#include <string.h>

int strlonger(char *s, char *t){
    return strlen(s)>strlen(t);
}

int main(){
    char s1[10]="Korea";
    char t1[10]="America";

    int diff=strlonger(s1,t1);
    printf("%d\n",diff);
}
```

Listing 6: Code-3_revision.c

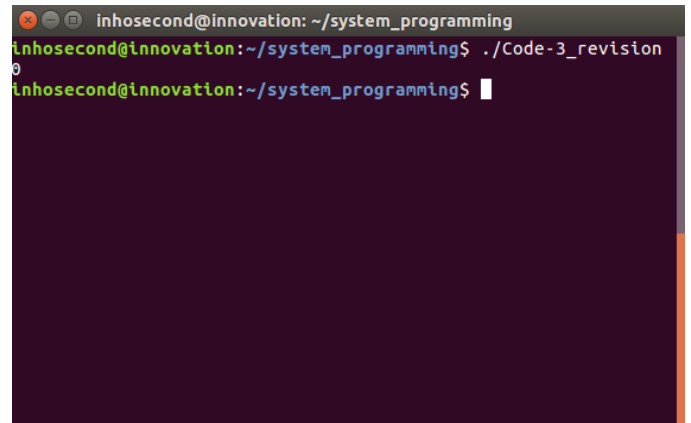


Figure 6: Executing result of Code-3_revision.c

5 Code-4

5.1 Problem

먼저 문제가 되는 코드는 다음과 같다.

```
#include<stdio.h>

int main(){
    unsigned char c;

    while((c=getchar())!=EOF){
        putchar(c);
    }
}
```

Listing 7: Code-4.c

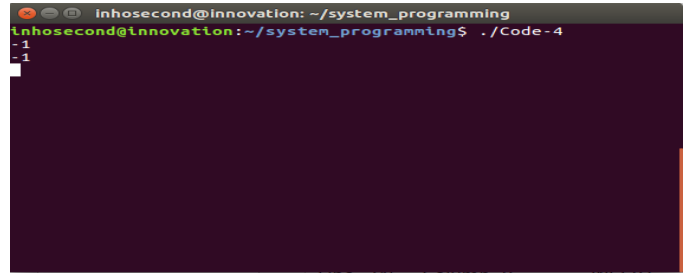


Figure 7: Executing result of Code-4.c

일단 /usr/include/libio.h 에 #define EOF (-1) 로 EOF는 -1로 정의되어 있다. 그런데 unsigned char는 절대 음수값을 가질 수 없으므로 위의 코드는 무한루프에 빠지게 된다. 또한 getchar는 들어온 값을 int형으로 리턴하도록 되어있다.(EOF=-1로 되어있기 때문에 -1을 포함시키기 위해) 따라서 getchar는 들어온 값이 0-255와 -1로 총 257가지 경우가 있는데 c의 타입인 char는(unsigned든 signed든) 1byte=8bit로 총 256가지 경우 밖에 가지지 못한다. 만약 unsigned char로 정의되어 있다면 EOF(-1) 값이 입력되었을때 255로 변하여 while문은 항상 참이 되어 위와 같이 무한루프에 빠지게 된다.

또한 만약 signed char로 정의되어 있다면 문자값 0xff(255)이 입력되었을때 -1이 되므로 EOF를 입력받은 것과 같은 방식으로 동작하는 문제가 생긴다.

5.2 Revision

이런 문제를 방지하기 위해 unsigned c를 int c로 변경한다. EOF(ctrl-d)를 눌렀을 때 종료가 잘 됨을 확인할 수 있다.

```
#include<stdio.h>

int main(){
    int c;

    while((c=getchar())!=EOF){
        putchar(c);
    }
}
```

Listing 8: Code-4_revision.c

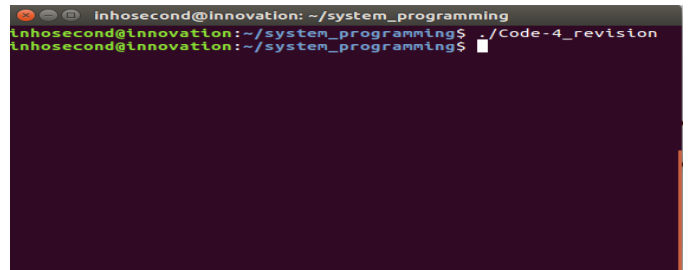


Figure 8: Executing result of Code-4_revision.c