Programming #3 -- digging into the runtime stack

Keller Sedillo-Garrido

10/13/22

Problem Description:

For this program, we have been given a program in C that will skip over a print function in main. We are to look into the program and see how it's able to do so and then edit the code so that we are able to skip the second print statement.

First Run(No edits to the code):

```
(base) keller@debianLaptop:~/Documents/Fall-2022/CS 471 - Prog Lang Structure/Programming #3$ make all
gcc -o runTimeStack runTimeStack.c
./runTimeStack
main is at 94377853075966
f is at 94377853075781
I am about to call f
0 1025205776
1 32764
2 0
3 3
4 1025206240
5 32764
6 241713817
7 21974
8 100
9 200
10 300
A is 1025205776
-4 0
-3 0
-2 241713648
-1 21974
0 1025205776
1 32764
2 0
3 3
4 1025206240
5 32764
6 241713827
7 21974
8 100
9 200
10 300
make: *** [Makefile:7: run] Illegal instruction
```

Runtime error: Illegal Instruction meaning:

After running the code for the first time, we get this error that the program spits out which is an "Illegal Instruction" error. After some google searching, I found a website that explains the reason why the error occurs is because there is some data in the program that is "corrupting other data on the stack such as the return address of a stack frame. " This is the case if we set the address to an invalid location.

Programming #3 -- digging into the runtime stack

Keller Sedillo-Garrido

10/13/22

Jumping Over the first print statement:

I have edited the program under the function f to where we are able to skip the first printf statement.

```c
void f() {

    unsigned int *A;
    int i;

   A =(unsigned int *) &A;

    for (i=0;i<=10; i++)
       printf("%d %u\n",i,A[i]);

   A[6]=A[6]+12;
   printf("A is %u \n",A);

   for (i=-4;i<=10; i++)
    printf("%d %u\n",i,A[i]);
}
```

```
(base) keller@debianLaptop:~/Documents/Fall-2022/CS 471 - Prog Lang Structure/Programming #3$ make all
gcc -o runTimeStack runTimeStack.c
./runTimeStack
main is at 94190851568126
f is at 94190851567941
I am about to call f
0 913191904
1 32764
2 0
3 3
4 913192368
5 32764
6 2218767001
7 21930
8 100
9 200
10 300
A is 913191904
-4 0
-3 0
-2 2218766832
-1 21930
0 913191904
1 32764
2 0
3 3
4 913192368
5 32764
6 2218767013
7 21930
8 100
9 200
10 300
 I am here
```

Programming #3 -- digging into the runtime stack

Keller Sedillo-Garrido

10/13/22

Why, on the return of f(), the first printf() is skipped.

Looking at the program, the f() seems to edit A[6]. A[6] seems to hold the return address of the function. This means that we can alter the return values directly and make the program jump to any location of our choosing.

Segmentation faults:

Another error that pops out is Segmentation Faults. Here is my code when it happened.

```c
void f() {

    unsigned int *A;
    int i;

   A =(unsigned int *) &A;

    for (i=0;i<=10; i++)
       printf("%d %u\n",i,A[i]);

   A[6]=A[6]+11;
   printf("A is %u \n",A);

   for (i=-4;i<=10; i++)
    printf("%d %u\n",i,A[i]);
}
```

```
(base) keller@debianLaptop:~/Documents/Fall-2022/CS 471 - Prog Lang Structure/Programming #3$ make all
gcc -o runTimeStack runTimeStack.c
./runTimeStack
main is at 94127872868862
f is at 94127872868677
I am about to call f
0 1491574576
1 32766
2 0
3 3
4 1491575040
5 32766
6 3664577177
7 21915
8 100
9 200
10 300
A is 1491574576
-4 0
-3 0
-2 3664577008
-1 21915
0 1491574576
1 32766
2 0
3 3
4 1491575040
5 32766
6 3664577188
7 21915
8 100
9 200
10 300
make: *** [Makefile:7: run] Segmentation fault
```

Programming #3 -- digging into the runtime stack

Keller Sedillo-Garrido

10/13/22

Looking into the occurrence of segmentation faults, this kind of error happens when there are errors in the memory. According to Tutorialspoint, they state “A segmentation fault occurs when your program attempts to access an area of memory that it is not allowed to access.” We could be trying to access memory that we have no business accessing. After incrementing some more, The program finally worked again.

```c
void f() {

    unsigned int *A;
    int i;

   A =(unsigned int *) &A;

    for (i=0;i<=10; i++)
       printf("%d %u\n",i,A[i]);

   A[6]=A[6]+12;
   printf("A is %u \n",A);

   for (i=-4;i<=10; i++)
    printf("%d %u\n",i,A[i]);
}
```

```
(base) keller@debianLaptop:~/Documents/Fall-2022/CS 471 - Prog Lang Structure/Programming #3$ make all
gcc -o runTimeStack runTimeStack.c
./runTimeStack
main is at 94474519323134
f is at 94474519322949
I am about to call f
0 3460017792
1 32766
2 0
3 3
4 3460018256
5 32766
6 2418680473
7 21996
8 100
9 200
10 300
A is 3460017792
-4 0
-3 0
-2 2418680304
-1 21996
0 3460017792
1 32766
2 0
3 3
4 3460018256
5 32766
6 2418680485
7 21996
8 100
9 200
10 300
 I am here
```

Programming #3 -- digging into the runtime stack

Keller Sedillo-Garrido

10/13/22

Reference:

https://www.gnu.org/software/libc/manual/html_node/Program-Error-Signals.html

https://www.tutorialspoint.com/What-is-a-segmentation-fault-in-C-Cplusplus