Debuging Report

Serhii Donetskyi

GDB:

Compile:

```
student@24bf42cbf919:/df/HomeWork/debug$ make
g++ -Wall -fexceptions -Weffc++ -pedantic-errors -Wextra -std=c++17 -g -Iinclude -c main.cpp -o main.o
g++ main.o -o main
student@24bf42cbf919:/df/HomeWork/debug$ echo $?
0
```

Run:

```
student@24bf42cbf919:/df/HomeWork/debug$ ./main
free(): double free detected in tcache 2
Aborted
```

Run gdb:

```
student@24bf42cbf919:/df/HomeWork/debug$ gdb ./main
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./main...
```

Put breakpoint on main() func and evaluate line by line:

```
Reading symbols from ./main...
(gdb) b 5
Breakpoint 1 at 0x11a9: file main.cpp, line 5.
(gdb) r
Starting program: /df/HomeWork/debug/main
warning: Error disabling address space randomization: Operation not permitted

Breakpoint 1, main () at main.cpp:5
5       int main(){
(gdb) n
7               char* arr = new char[128];
(gdb) n
8               arr = new char[128];
(gdb) n
10              delete[] arr;
(gdb) n
11              delete[] arr;
(gdb) n
free(): double free detected in tcache 2

Program received signal SIGABRT, Aborted.
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:50
50      ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
(gdb)
```

It is double free. Fix the issue:

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main(){
6
7       char* arr = new char[128];
8       arr = new char[128];
9
10      //delete[] arr;
11      delete[] arr;
12
13      return 0;
14  }
```

Recomplie and run:

```
student@24bf42cbf919:/df/HomeWork/debug$ make
g++ -Wall -fexceptions -Weffc++ -pedantic-errors -Wextra -std=c++17 -g -Iinclude -c main.cpp -o main.o
g++ main.o -o main
student@24bf42cbf919:/df/HomeWork/debug$ ./main
student@24bf42cbf919:/df/HomeWork/debug$ echo $?
0
```

No runtime errors.

valgrind:

Run valgrind:

```
student@24bf42cbf919:/df/HomeWork/debug$ valgrind ./main
==3876== Memcheck, a memory error detector
==3876== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3876== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3876== Command: ./main
==3876==
==3876==
==3876== HEAP SUMMARY:
==3876==     in use at exit: 128 bytes in 1 blocks
==3876==   total heap usage: 3 allocs, 2 frees, 72,960 bytes allocated
==3876==
==3876== LEAK SUMMARY:
==3876==    definitely lost: 128 bytes in 1 blocks
==3876==    indirectly lost: 0 bytes in 0 blocks
==3876==      possibly lost: 0 bytes in 0 blocks
==3876==    still reachable: 0 bytes in 0 blocks
==3876==         suppressed: 0 bytes in 0 blocks
==3876== Rerun with --leak-check=full to see details of leaked memory
==3876==
==3876== For lists of detected and suppressed errors, rerun with: -s
==3876== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@24bf42cbf919:/df/HomeWork/debug$
```

find some memory leaks. Fix the issue:

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main(){
6
7       char* arr = new char[128];
8       delete[] arr;
9
10      arr = new char[128];
11      delete[] arr;
12
13      return 0;
14  }
```

Recompile and run:

```
student@24bf42cbf919:/df/HomeWork/debug$ make
g++ -Wall -fexceptions -Weffc++ -pedantic-errors -Wextra -std=c++17 -g -Iinclude -c main.cpp -o main.o
g++ main.o -o main
student@24bf42cbf919:/df/HomeWork/debug$ valgrind ./main
==3884== Memcheck, a memory error detector
==3884== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==3884== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==3884== Command: ./main
==3884==
==3884==
==3884== HEAP SUMMARY:
==3884==     in use at exit: 0 bytes in 0 blocks
==3884==   total heap usage: 3 allocs, 3 frees, 72,960 bytes allocated
==3884==
==3884== All heap blocks were freed -- no leaks are possible
==3884==
==3884== For lists of detected and suppressed errors, rerun with: -s
==3884== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@24bf42cbf919:/df/HomeWork/debug$
```

No leaks.

gprof:

Edit code:

```cpp
#include <iostream>

using namespace std;

int test1(){
    int i = 0;
    while(i >= 0){
        ++i;
    }
    return i;
}

int test2(int init){
    int i = init;
    while(i < 0){
        ++i;
    }
    return i;
}

int main(){

    char* arr = new char[128];
    delete[] arr;

    arr = new char[128];
    delete[] arr;

    cout << test1() << endl;
    cout << test2(test1()) << endl;

    return 0;
}
```

Use "gcc -pg" for performance compiler:

```
student@24bf42cbf919:/df/HomeWork/debug$ make pg
g++ -Wall -fexceptions -Weffc++ -pedantic-errors -Wextra -std=c++17 -g -Iinclude -c main.cpp -o main.o
g++ main.o -pg -o main
student@24bf42cbf919:/df/HomeWork/debug$ ./main
-2147483648
0
```

use gprof tool for parse:

```
student@24bf42cbf919:/df/HomeWork/debug$ gprof ./main
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
 67.58      5.08     5.08                             test1()
 33.86      7.63     2.55                             test2(int)
```

we can also use "valgrind --tool=callgrind" for that purpose.