CSc 352

Valgrind

Benjamin Dicken

Valgrind

- A tool for debugging executables
- Provides a virtual CPU to run various “tools” for profiling your code
- Can use to
  - Detect memory leaks
  - Profile heap usage
  - Profile performance
  - Etc.

Check out the man page

Valgrind

- Use the **`--tool`** command-line option to specify what valgrind tool you want to use
- For this class, primarily use **`--tool=memcheck`** (the default)
  - You are welcome to experiment with others!

```c
#define LARGE 250

char* get_longest_line() {
  char* longest = NULL;
  char* line_buffer = malloc(LARGE);
  while(fgets(line_buffer, LARGE, stdin) != NULL) {
    int length = strlen(line_buffer);
    if (longest == NULL || length > strlen(longest)) {
      longest = malloc(length+1);
      longest[length] = '\0';
      strncpy(longest, line_buffer, length);
    }
  }
  return longest;
}

int main() {
  char* longest_line = get_longest_line();
  printf("The longest line from standard input is:\n");
  printf("%s\n", longest_line);
  free(longest_line);
  return 0;
}
```

Input:

```
abcdefghijk
abcdefghijklmnop
abcdefghijklmnopqrs
abcdefghijklmnopqrstuv
```

Try out Valgrind with this program from before

```
$ gcc -Wall -Werror -std=c11 test.c -o longestline
$ valgrind --tool=memcheck ./longestline
==494374== Memcheck, a memory error detector
==494374== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==494374== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==494374== Command: ./longestline
==494374==
```

Waiting for us to give it standard input

```
$ gcc -Wall -Werror -std=c11 test.c -o longestline
$ valgrind --tool=memcheck ./longestline
==494374== Memcheck, a memory error detector
==494374== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==494374== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==494374== Command: ./longestline
==494374==
abcdefghijk
abcdefghijklmnop
abcdefghijklmnopqrs
abcdefghijklmnopqrstuv
```

Ctrl-D to send EOF signal to program

```
. . . .
==494374==
==494374== HEAP SUMMARY:
==494374==     in use at exit: 302 bytes in 4 blocks
==494374==   total heap usage: 7 allocs, 3 frees, 2,374 bytes allocated
==494374==
==494374== LEAK SUMMARY:
==494374==    definitely lost: 302 bytes in 4 blocks
==494374==    indirectly lost: 0 bytes in 0 blocks
==494374==      possibly lost: 0 bytes in 0 blocks
==494374==    still reachable: 0 bytes in 0 blocks
==494374==         suppressed: 0 bytes in 0 blocks
==494374== Rerun with --leak-check=full to see details of leaked memory
==494374==
==494374== For lists of detected and suppressed errors, rerun with: -s
==494374== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Heap memory in use at program exit

Definitely lost this memory

Maybe get more info?

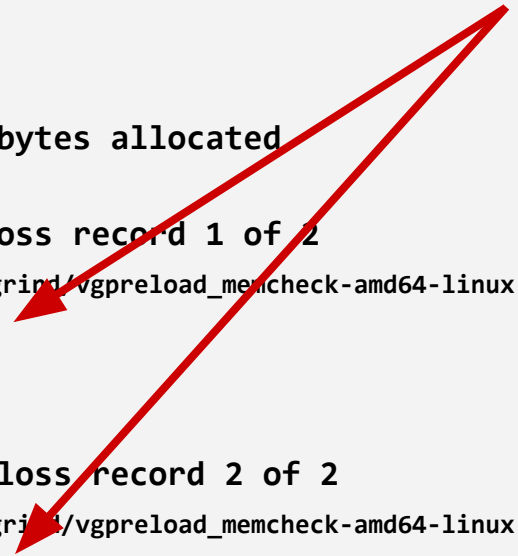
```
$ gcc -Wall -Werror -std=c11 test.c -o longestline
$ valgrind --tool=memcheck --leak-check=full ./longestline
. . . .
==495187==
==495187== HEAP SUMMARY:
==495187==     in use at exit: 302 bytes in 4 blocks
==495187==   total heap usage: 7 allocs, 3 frees, 2,374 bytes allocated
==495187==
==495187== 52 bytes in 3 blocks are definitely lost in loss record 1 of 2
==495187==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==495187==    by 0x10924A: get_longest_line (in /home/bddicken/test/longestline)
==495187==    by 0x1092B9: main (in /home/bddicken/test/longestline)
==495187==
==495187== 250 bytes in 1 blocks are definitely lost in loss record 2 of 2
==495187==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==495187==    by 0x109207: get_longest_line (in /home/bddicken/test/longestline)
==495187==    by 0x1092B9: main (in /home/bddicken/test/longestline)
==495187==
. . . .
```

Specific info on where the memory was lost

But we can do better with -g . . . .

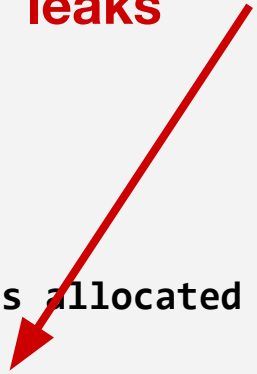
```
$ gcc -Wall -Werror -std=c11 test.c -g -o longestline
$ valgrind --tool=memcheck --leak-check=full ./longestline
. . . .
==495350== HEAP SUMMARY:
==495350==     in use at exit: 302 bytes in 4 blocks
==495350==   total heap usage: 7 allocs, 3 frees, 2,374 bytes allocated
==495350==
==495350== 52 bytes in 3 blocks are definitely lost in loss record 1 of 2
==495350==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==495350==    by 0x10924A: get_longest_line (test.c:14)
==495350==    by 0x1092B9: main (test.c:23)
==495350==
==495350== 250 bytes in 1 blocks are definitely lost in loss record 2 of 2
==495350==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==495350==    by 0x109207: get_longest_line (test.c:10)
==495350==    by 0x1092B9: main (test.c:23)
. . . .
```

File names and line numbers



```
$ gcc -Wall -Werror -std=c11 test.c -g -o longestline
$ valgrind --tool=memcheck ./longestline
. . . .
==497142==
==497142== HEAP SUMMARY:
==497142==     in use at exit: 0 bytes in 0 blocks
==497142==   total heap usage: 7 allocs, 7 frees, 2,374 bytes allocated
==497142==
==497142== All heap blocks were freed -- no leaks are possible
==497142==
==497142== For lists of detected and suppressed errors, rerun with: -s
==497142== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
. . . .
```

After fixing the leaks



```c
int main() {
  int sum;
  int i;
  while (i > 0) {
    sum += rand();
    i--;
  }
  printf("%d\n", sum);
  return 0;
}
```

Valgrind can also detect other memory-related issues

```
$ valgrind --tool=memcheck --leak-check=full ./a.out
==499172== Memcheck, a memory error detector
==499172== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==499172== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==499172== Command: ./a.out
==499172==
==499172== Conditional jump or move depends on uninitialised value(s)
==499172==    at 0x109187: main (test2.c:7)
==499172==
==499172== Conditional jump or move depends on uninitialised value(s)
==499172==    at 0x48DD958: __vfprintf_internal (vfprintf-internal.c:1687)
==499172==    by 0x48C7D3E: printf (printf.c:33)
==499172==    by 0x10919E: main (test2.c:11)
==499172==
==499172== Use of uninitialised value of size 8
==499172==    at 0x48C169B: _itoa_word (_itoa.c:179)
==499172==    by 0x48DD574: __vfprintf_internal (vfprintf-internal.c:1687)
==499172==    by 0x48C7D3E: printf (printf.c:33)
==499172==    by 0x10919E: main (test2.c:11)
. . . .
```

What is wrong? What does valgrind say?

```c
#include <stdlib.h>
int main() {
  int x = -1;
  char * data = malloc(x);
  free(data);
  return 0;
}
```

```
==506388==
==506388== Argument 'size' of function malloc has a fishy (possibly negative) value: -1
==506388==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==506388==    by 0x109188: main (test2.c:4)
==506388==
```

https://valgrind.org/docs/manual/mc-manual.html