600.120 Intermediate Programming, Spring 2017*

Misha Kazhdan

**Much of the code in these examples is not commented because it would otherwise not fit on the slides. This is bad coding practice in general and you should not follow my lead on this.*

Outline

- `Valgrind`

Valgrind

- Easy-to-use tool for finding memory leaks and other memory issues
- Compile with `-g` to get more helpful output from `valgrind`
- Then run using `valgrind`:

```
valgrind --leak-check=full ./myFile <arg1> <arg2> ...
```

```
#include <stdio.h>
int main( void )
{
    printf( "Hello world!\n" );
    return 0;
}
```

```
>> gcc -std=c99 -Wall -Wextra -g foo.c
>> ./a.out
Hello world!
>>
```

See also http://valgrind.org/docs/manual/QuickStart.html

Valgrind

- Easy-to-use tool for finding memory leaks and other memory issues
- Compile with `-g` to get more helpful output from `valgrind`
- Then run using `valgrind`:

va

```
>> valgrind --leak-check=full ./a.out
```

header

```
==12133== Command: ./a.out
==12133==
Hello World!
==12133==
==12133== HEAP SUMMARY:
==12133==     in use at exit: 0 bytes in 0 blocks
==12133==   total heap usage: 1 allocs, 1 frees, 1,024 bytes allocated
==12133==
==12133== All heap blocks were freed -- no leaks are possible
==12133==
==12133== For counts of detected and suppressed errors, rerun with: -v
==12133== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
>>
```

>

world!\n");

Wextra -g foo.c

See also http://va

Valgrind

- Your program output is interspersed with messages from `valgrind`
- Kinds of issues flagged by `valgrind`
  - Invalid reads or writes:
    Attempts to dereference pointers to memory that's not yours
  - Memory leaks:
    Failing to deallocate a block of memory you allocate previously
    - Info about leaks appears in HEAP SUMMARY section

Valgrind

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
char * string_copy( const char * in )
{
    char *out = malloc( strlen( in ) );
    assert( out!= NULL );
    return strcpy( out , in );
}
int main( void )
{
    char *str = string_copy( "hello" );
    assert( str!=NULL );
    printf( "%s\n" , str );
    return 0;
}
```

```
>> ./a.out
hello
>>
```

```
>> valgrind --leak-check=full ./a.out
```

header

```
==17647== Command: ./a.out
==17647==
==17647== Invalid write of size 1
==17647==    at 0x4C30CB7: strcpy (vg_replace_strmem.c:506)
==17647==    by 0x40067C: string_copy (foo.c:9)
==17647==    by 0x400690: main (foo.c:13)
==17647==  Address 0x5200045 is 0 bytes after a block of size 5 alloc'd
==17647==    at 0x4C2DB9D: malloc (vg_replace_malloc.c:299)
==17647==    by 0x400645: string_copy (foo.c:7)
==17647==    by 0x400690: main (foo.c:13)
==17647==
==17647== Invalid read of size 1
==17647==    at 0x4C30BC4: strlen (vg_replace_strmem.c:454)
==17647==    by 0x4EAAAA1: puts (in /usr/lib64/libc-2.24.so)
==17647==    by 0x4006C0: main (foo.c:15)
==17647==  Address 0x5200045 is 0 bytes after a block of size 5 alloc'd
==17647==    at 0x4C2DB9D: malloc (vg_replace_malloc.c:299)
==17647==    by 0x400645: string_copy (foo.c:7)
==17647==    by 0x400690: main (foo.c:13)
==17647==
hello
==17647==
```

first half of output

Valgrind

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
char * string_copy( const char * in )
{
    char *out = malloc( strlen( in ) );
    assert( out!= NULL );
    return strcpy( out , in );
}
int main( void )
{
    char *str = string_copy( "hello" );
    assert( str!=NULL );
    printf( "%s\n" , str );
    return 0;
}
```

```
>> ./a.out
hello
>>
```

second half of output

```
==17647== HEAP SUMMARY:
==17647==     in use at exit: 5 bytes in 1 blocks
==17647==   total heap usage: 2 allocs, 1 frees, 1,029 bytes allocated
==17647==
==17647== 5 bytes in 1 blocks are definitely lost in loss record 1 of 1
==17647==    at 0x4C2DB9D: malloc (vg_replace_malloc.c:299)
==17647==    by 0x400645: string_copy (foo.c:7)
==17647==    by 0x400690: main (foo.c:13)
==17647==
==17647== LEAK SUMMARY:
==17647==    definitely lost: 5 bytes in 1 blocks
==17647==    indirectly lost: 0 bytes in 0 blocks
==17647==      possibly lost: 0 bytes in 0 blocks
==17647==    still reachable: 0 bytes in 0 blocks
==17647==         suppressed: 0 bytes in 0 blocks
==17647==
==17647== For counts of detected and suppressed errors, rerun with: -v
==17647== ERROR SUMMARY: 3 errors from 3 contexts (suppressed: 0 from 0)
>>
```

Valgrind

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
char * string_copy( const char * in )
{
    char *out = malloc( strlen( in ) );
    assert( out!= NULL );
    return strcpy( out , in );
}

int main( void )
{
    char *str = string_copy( "hello" );
    assert( str!=NULL );
    printf( "%s\n" , str );
    return 0;
}
```

```
>> ./a.out
hello
>>
```

```
>> valgrind --leak-check=full ./a.out
```

header

```
==17647== Command: ./a.out
==17647==
==17647== Invalid write of size 1
==17647==    at 0x4C30CB7: strcpy (vg_replace_strmem.c:506)
==17647==    by 0x40067C: string_copy (foo.c:9)
==17647==    by 0x400690: main (foo.c:13)
==17647==  Address 0x5200045 is 0 bytes after a block of size 5 alloc'd
==17647==    at 0x4C2DB9D: malloc (vg_replace_malloc.c:299)
==17647==    by 0x400645: string_copy (foo.c:7)
==17647==    by 0x400690: main (foo.c:13)
==17647==
==17647== Invalid read of size 1
==17647==    at 0x4C30BC4: strlen (vg_replace_strmem.c:454)
==17647==    by 0x4EAAAA1: puts (in /usr/lib64/libc-2.24.so)
==17647==    by 0x4006C0: main (foo.c:15)
==17647==  Address 0x5200045 is 0 bytes after a block of size 5 alloc'd
==17647==    at 0x4C2DB9D: malloc (vg_replace_malloc.c:299)
==17647==    by 0x400645: string_copy (foo.c:7)
==17647==    by 0x400690: main (foo.c:13)
==17647==
hello
==17647==
```

first half of output

Valgrind

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
char * string_copy( const char * in )
{
    char *out = malloc( strlen( in ) );
    assert( out!= NULL );
    return strcpy( out , in );
}
int main( void )
{
    char *str = string_copy( "hello" );
    assert( str!=NULL );
    printf( "%s\n" , str );
    return 0;
}
```

```
>> ./a.out
hello
>>
```

```
>> valgrind --leak-check=full ./a.out
```

header

```
==17647== Command: ./a.out
==17647==
==17647== Invalid write of size 1
==17647==    at 0x4C30CB7: strcpy (vg_replace_strmem.c:506)
==17647==    by 0x40067C: string_copy (foo.c:9)
==17647==    by 0x400690: main (foo.c:13)
==17647==  Address 0x5200045 is 0 bytes after a block of size 5 alloc'd
==17647==    at 0x4C2DB9D: malloc (vg_replace_malloc.c:299)
==17647==    by 0x400645: string_copy (foo.c:7)
==17647==    by 0x400690: main (foo.c:13)
==17647==
==17647== Invalid read of size 1
==17647==    at 0x4C30BC4: strlen (vg_replace_strmem.c:454)
==17647==    by 0x4EAAAA1: puts (in /usr/lib64/libc-2.24.so)
==17647==    by 0x4006C0: main (foo.c:15)
==17647==  Address 0x5200045 is 0 bytes after a block of size 5 alloc'd
==17647==    at 0x4C2DB9D: malloc (vg_replace_malloc.c:299)
==17647==    by 0x400645: string_copy (foo.c:7)
==17647==    by 0x400690: main (foo.c:13)
==17647==
hello
==17647==
```

first half of output

Valgrind

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
char * string_copy( const char * in )
{
    char *out = malloc( strlen( in ) );
    assert( out!= NULL );
    return strcpy( out , in );
}
int main( void )
{
    char *str = string_copy( "hello" );
    assert( str!=NULL );
    printf( "%s\n" , str );
    return 0;
}
```

```
>> ./a.out
hello
>>
```

second half of output

```
==17647== HEAP SUMMARY:
==17647==     in use at exit: 5 bytes in 1 blocks
==17647==   total heap usage: 2 allocs, 1 frees, 1,029 bytes allocated
==17647==
==17647== 5 bytes in 1 blocks are definitely lost in loss record 1 of 1
==17647==    at 0x4C2DB9D: malloc (vg_replace_malloc.c:299)
==17647==    by 0x400645: string_copy (foo.c:7)
==17647==    by 0x400690: main (foo.c:13)
==17647==
==17647== LEAK SUMMARY:
==17647==    definitely lost: 5 bytes in 1 blocks
==17647==    indirectly lost: 0 bytes in 0 blocks
==17647==      possibly lost: 0 bytes in 0 blocks
==17647==    still reachable: 0 bytes in 0 blocks
==17647==         suppressed: 0 bytes in 0 blocks
==17647==
==17647== For counts of detected and suppressed errors, rerun with: -v
==17647== ERROR SUMMARY: 3 errors from 3 contexts (suppressed: 0 from 0)
>>
```

Valgrind

- So what was wrong?
  - An invalid write
  - An invalid read
  - A block of memory that wasn't `free`d

- On `ugradx`, this program didn't crash and seemed to work properly

⇒ Not every bad memory access leads to error (or bad output)

⇒ `valgrind` is a useful tool to help us find problematic code

Valgrind

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
char * string_copy( const char * in )
{
    char *out = malloc( strlen( in ) +1 );
    assert( out!= NULL );
    return strcpy( out , in );
}
int main( void )
{
    char *str = string_copy( "hello" );
    assert( str!=NULL );
    printf( "%s\n" , str );
    free( str );
    return 0;
}
```

```
>> ./a.out
hello
>>
```

```
>> valgrind --leak-check=full ./a.out
                                                    header
==30398== Command: ./a.out
==30398==
hello
==30398==
==30398== HEAP SUMMARY:
==30398==     in use at exit: 0 bytes in 0 blocks
==30398==   total heap usage: 2 allocs, 2 frees, 1,030 bytes allocated
==30398==
==30398== All heap blocks were freed -- no leaks are possible
==30398==
==30398== For counts of detected and suppressed errors, rerun with: -v
==30398== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
>>
```

Piazza → Resources section → Resources tab → Exercise 6-1*

*Add the `-lm` command to include the math library when compiling primes.c