CSc 352
Debugging Tools
Uninitialized pointers

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

char *str;

int main() {
    scanf("%s", str);
    printf("String read: %s\n", str);
    printf("Length of string: %d\n", (int)strlen(str));
    return 0;
}
```

str was never initialized to point to anything
Uninitialized pointers

Suppose this was a program of realistic size.

How would we identify the location and reason for the problem?
Locating the problem: gdb

```
% gdb -q ./a.out
Reading symbols from /home/debray/tmp/a.out...done.
(gdb) run
Starting program: /home/debray/tmp/a.out
abcde
String read: (null)
Program received signal SIGSEGV, Segmentation fault.
0x00000000004005c2 in main () at uninit-ptr-1.c:16
16     printf("Length of string: %d\n", (int)strlen(str));
(gdb)
(gdb) where
#0  0x000000000004005c2 in main () at uninit-ptr-1.c:16
(gdb) #0  0x000000000004005c2 in main () at uninit-ptr-1.c:16
(gdb) list
11     int main() {
12         scanf("%s", str);
13         printf("String read: %s\n", str);
14         printf("Length of string: %d\n", (int)strlen(str));
15         return 0;
16     }
17     
(gdb) print str
$1 = 0x0
(gdb)
$2 = 0x0
(gdb) quit
A debugging session is active.
 Inferior 1 [process 12213] will be killed.

Quit anyway? (y or n) y
% ```
Memory error diagnosis: valgrind

invoking the tool:
valgrind progName arg₁ arg₂ ...

indicates:
(1) there was a problem;
(2) what happened
(3) where it happened
Dangling pointers

We looked at this code earlier:

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

// read_string(str) -- reads a string into buffer str. Returns
// str if a string was successfully read, NULL otherwise.
char *read_string(char *str) {
    int status = scanf("%s", str);
    if (status > 0) {
        return str;
    } else {
        return NULL;
    }
}

// my_read() -- reads a string into a buffer and returns a pointer
// to that buffer.
char *my_read() {
    char buf[128];
    return read_string(buf);
}

int main() {
    char *string = my_read();
    printf("> string: %s -- length = %d\n", string, strlen(string));
    return 0;
}
```

runtime stack

```
string
buf
str
```

main

```
dangling pointer!
```

```text
% cat dangling-ptr-1.c
// File: dangling-ptr-1.c
// Purpose: To illustrate dangling pointers

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

// read_string(str) -- reads a string into buffer str. Returns
// str if a string was successfully read, NULL otherwise.
char *read_string(char *str) {
    int status = scanf("%s", str);
    if (status > 0) {
        return str;
    } else {
        return NULL;
    }
}

// my_read() -- reads a string into a buffer and returns a pointer
// to that buffer.
char *my_read() {
    char buf[128];
    return read_string(buf);
}

int main() {
    char *string = my_read();
    printf("> string: %s -- length = %d\n", string, strlen(string));
    return 0;
}
% gcc -Wall dangling-ptr-1.c
% ./a.out
abcdef
>> string: -- length = 1
```
Dangling pointers

Minor variation on this code:

```c
% pwd
/cs/www/classes/cs352/spring10/Code/ex.3.Debugging
% cat dangling-ptr-2.c
// File: dangling-ptr-1.c
// Purpose: To illustrate dangling pointers

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

// read_string(str) -- reads a string into buffer str. Returns
// str if a string was successfully read, NULL otherwise.
char *read_string(char *str) {
    int status = scanf("%s", str);
    if (status > 0) {
        return str;
    } else {
        return NULL;
    }
}

// my_read() -- reads a string into a buffer and returns a pointer
// to that buffer.
char *my_read() {
    int padding[4096];
    char buf[128];
    return read_string(buf);
}

int main() {
    char *string = my_read();
    printf("\nstring: %s -- length = %d\n", string, (int)strlen(string));
    return 0;
}
```
Dangling pointers

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

// read_string(str) -- reads a string into buffer str. Returns
// str if a string was successfully read. NULL otherwise.
char *read_string(char *str) {
    int status = sscanf("%s", str);
    if (status > 0) {
        return str;
    } else {
        return NULL;
    }
}

// my_read() -- reads a string into a buffer and returns a pointer
// to that buffer.
char *my_read() {
    int padding[4096];
    char buf[128];
    return read_string(buf);
}

int main() {
    char *string = my_read();
    printf("String: %s -- length = %d\n", string, (int)strlen(string));
    return 0;
}
```

the code seems to work!!!
(on hedgehog.cs.arizona.edu)
Dangling pointers

doesn’t work
(on lectura.cs.arizona.edu)
What’s going on?

the array \texttt{padding[\ldots]} “protects” \texttt{buf[\ldots]} from getting overwritten — so the code seems to work (on some machines)
More diagnosis

```c
% hostname
ehedgehog.cs.arizona.edu
% gcc dangling-ptr-2.c
% ./a.out
abcde
>> string: abcde -- length = 5
% valgrind ./a.out
==3808== Memcheck, a memory error detector.
==3808== Copyright (C) 2002-2007, and GNU GPL'd, by Julian Seward et al.
==3808== Using LibVEX rev 1804, a library for dynamic binary translation.
==3808== Copyright (C) 2004-2007, and GNU GPL'd, by OpenWorks LLP.
==3808== Using valgrind-3.3.0, a dynamic binary instrumentation framework.
==3808== Copyright (C) 2000-2007, and GNU GPL'd, by Julian Seward et al.
==3808== For more details, rerun with: -v

abcde
==3808== Invalid read of size 1
==3808== at 0x4A07B12: strlen (mc_replace_strmem.c:242)
==3808== by 0x4005D3: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==3808== Address 0x7feffbf50 is not stack'd, malloc'd or (recently) free'd

==3808== Invalid read of size 1
==3808== at 0x4A07B24: strlen (mc_replace_strmem.c:242)
==3808== by 0x4005D3: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==3808== Address 0x7feffbf51 is not stack'd, malloc'd or (recently) free'd

==3808== Invalid read of size 1
==3808== at 0x4A07B12: strlen (mc_replace_strmem.c:242)
==3808== by 0x3D944A50F: vprintf (in /lib64/libc-2.8.so)
==3808== by 0x3D9451079: printf (in /lib64/libc-2.8.so)
==3808== by 0x4005E8: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==3808== Address 0x7feffbf50 is not stack'd, malloc'd or (recently) free'd

==3808== Invalid read of size 1
==3808== at 0x4A07B24: strlen (mc_replace_strmem.c:242)
==3808== by 0x3D944A50F: vprintf (in /lib64/libc-2.8.so)
==3808== by 0x3D9451079: printf (in /lib64/libc-2.8.so)
==3808== by 0x4005E8: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==3808== Address 0x7feffbf51 is not stack'd, malloc'd or (recently) free'd
```
Summary

• Just because a program produces the expected output doesn’t mean that it’s correct
  – the observed behavior may be accidental
  – the observed behavior may be system-dependent

• Use **valgrind** to check whether the execution was free of memory errors
  – provides information only about one execution
    • other executions may contain erroneous behaviors
  – provides some help in identifying where the error occurred.
Another example

```c
/*
 * File: strcat2.c
 * Purpose: concatenate two strings.
 * NOTE: Some error checks in this code have been elided to make
 * the code fit on the classroom screen.
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* strcat2(str1, str2) takes two strings str1 and str2, concatenates
 * then into a third string, and returns a pointer to the result.
 * str1 and str2 are unaffected. */
char *strcat2(char *str1, char *str2) {
    // check that neither str1 nor str2 is NULL
    char *buf = malloc(strlen(str1)+strlen(str2), sizeof(char));
    if (!buf) {
        fprintf(stderr, "Out of memory!\n");
        exit(1);
    }
    strcat(buf, str1);
    strcat(buf, str2);
    return buf;
}

int main() {
    char str1[1024], str2[1024];
    scanf("%s %s", str1, str2);
    printf("\n\tstr1 = %s, str2 = %s; concat'ed result = %s\n", str1, str2, strcat2(str1, str2));
    return 0;
}
% gcc -Wall strcat2.c
% ./a.out
abcde fghij
>>> str1 = abcde, str2 = fghij; concat'ed result = abcdefghij
```
Example 2

```
return 0;
}
% gcc -Wall strcat2.c
% ./a.out
abcdefgh

>>> str1 = abcd, str2 = fghij; concat'ed result = abcdfghij
% valgrind ./a.out

==4114== Memcheck, a memory error detector.
==4114== Copyright (C) 2002-2007, and GNU GPL'd, by Julian Seward et al.
==4114== Using LibVEX rev 1804, a library for dynamic binary translation.
==4114== Copyright (C) 2004-2007, and GNU GPL'd, by OpenWorks LLP.
==4114== Using valgrind-3.3.0, a dynamic binary instrumentation framework.
==4114== Copyright (C) 2000-2007, and GNU GPL'd, by Julian Seward et al.
==4114== For more details, rerun with: -v
==4114== abcd
==4114== fghij
==4114== Invalid write of size 1
==4114== at 0x4A05174: strcat (mc_replace_strmem.c:186)
==4114==  by 0x400739: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==4114==  by 0x40077C: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==4114== Address 0x4c4303a is 0 bytes after a block of size 10 alloc'd
==4114== at 0x4A05174: calloc (vg_replace_malloc.c:397)
==4114==  by 0x4006EF: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==4114==  by 0x40077C: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==4114==
==4114== Invalid read of size 1
==4114== at 0x4A07824: strlen (mc_replace_strmem.c:242)
==4114==  by 0x3D9944A56F: vprintf (in /lib64/libc-2.8.so)
==4114==  by 0x3D99451079: printf (in /lib64/libc-2.8.so)
==4114==  by 0x40079B: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==4114== Address 0x4c4303a is 0 bytes after a block of size 10 alloc'd
==4114== at 0x4A05174: calloc (vg_replace_malloc.c:397)
==4114==
==4114== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 4 from 1)
==4114== malloc/free: in use at exit: 10 bytes in 1 blocks.
==4114== malloc/free: 1 allocs, 0 frees, 10 bytes allocated.
==4114== For counts of detected errors, rerun with: -v

>>> str1 = abcd, str2 = fghij; concat'ed result = abcdfghij
```
Example 2

```
return 0;
}
% gcc -Wall strcat2.c
% ./a.out
abcde fghij
>> str1 = abcde, str2 = fghij; concat'ed result = abcdefghij
% valgrind ./a.out
---14--- Memcheck, a memory error detector.
---14--- Copyright (C) 2002-2007, and GNU GPL'd, by Julian Seward et al.
---14--- Using LibVEX rev 1804, a library for dynamic binary translation.
---14--- Copyright (C) 2004-2007, and GNU GPL'd, by OpenWorks LLP.
---14--- Using valgrind-3.3.0, a dynamic binary instrumentation framework.
---14--- Copyright (C) 2000-2007, and GNU GPL'd, by Julian Seward et al.
---14--- For more details, rerun with: -v
---14--- abcde fghij
---14--- Invalid write of size 1
---14--- at 0x4A0790F: strcat (mc_replace_strmem.c:186)
---14--- by 0x400739: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
---14--- by 0x400777f: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
---14--- Address 0x4c430a is 0 bytes after a block of size 10 alloc'd
---14--- at 0x4A05174: malloc (vg_replace_malloc.c:397)
---14--- by 0x40066F: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
---14--- by 0x400777f: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
---14--- Invalid read of size 1
---14--- at 0x4A07824: strlen (mc_replace_strmem.c:242)
---14--- by 0x3D9044A56F: vprintf (in /lib64/libc-2.8.so)
---14--- by 0x3D90451079: printf (in /lib64/libc-2.8.so)
---14--- by 0x400779: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
---14--- Address 0x4c430a is 0 bytes after a block of size 10 alloc'd
---14--- at 0x4A05174: malloc (vg_replace_malloc.c:397)
---14--- by 0x40066F: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
---14--- by 0x400777f: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
>> str1 = abcde, str2 = fghij; concat'ed result = abcdefghij
---14--- ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 4 from 1)
---14--- malloc/free: in use at exit: 10 bytes in 1 blocks.
---14--- malloc/free: 1 allocs, 0 frees, 10 bytes allocated.
---14--- For counts of detected errors, rerun with: -v
```
Example 3

```c
/* cat strcat3.c
 * File: strcat3.c
 * Purpose: read two strings A and B, concatenate them to get A B A.
 * NOTE: Some error checks in this code have been elided to make the code fit on the classroom screen.
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* strcat2(str1, str2) takes two strings str1 and str2, concatenates
 * them into a third string, and returns a pointer to the result. */
char *strcat2(char *str1, char *str2) {
    // check that neither str1 nor str2 is NULL
    char *buf = calloc(strlen(str1)+strlen(str2)+1, sizeof(char));
    if (!buf) {
        fprintf(stderr, "Out of memory!\n");
        exit(1);
    }

    strcat(buf, str1); free(str1);
    strcat(buf, str2); free(str2);

    return buf;
}

int main() {
    char *str1, *str2, buf[1024];
    scanf("%s", buf); str1 = strdup(buf);  // strdup() allocates space on heap
    scanf("%s", buf); str2 = strdup(buf);
    printf(">> final concatenated result = %s\n", strcat2(strcat2(str1,str2), str1));
    return 0;
}
% gcc strcat3.c -o strcat3
% ./strcat3
abcdefghi
>> final concatenated result = abcdefghij
% 
```
Example 3

```c
printf(">>> final concat'ed result = %s\n", strcat2( strcat2(str1,str2), str1));
return 0;
}
```

```
% gcc strcat3.c -o strcat3
% ./strcat3
abced fghij
>>> final concat'ed result = abcdefghij
% valgrind -q ./strcat3
abced fghij
```

```c
---552---  Invalid read of size 1
---552---  at 0x4a07b12: strlen (mc_replace_strmem.c:242)
---552---  by 0x400771: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  by 0x40085a: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  Address 0x4c4303b is 0 bytes inside a block of size 6 free'd
---552---  at 0x4a0659f: free (vg_replace_malloc.c:333)
---552---  by 0x4007c9: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  by 0x40084t: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---
```

```
---552---  Invalid read of size 1
---552---  at 0x4a07b24: strlen (mc_replace_strmem.c:242)
---552---  by 0x400771: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  by 0x40085a: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  Address 0x4c4303c is 1 bytes inside a block of size 6 free'd
---552---  at 0x4a0659f: free (vg_replace_malloc.c:333)
---552---  by 0x4007c9: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  by 0x40084t: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---
```

```
---552---  Invalid read of size 1
---552---  at 0x4a078f9: strcat (mc_replace_strmem.c:186)
---552---  by 0x40076d: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  by 0x40085a: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  Address 0x4c4303d is 0 bytes inside a block of size 6 free'd
---552---  at 0x4a0659f: free (vg_replace_malloc.c:333)
---552---  by 0x4007c9: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  by 0x40084t: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---
```

```
---552---  Invalid read of size 1
---552---  at 0x4a07902: strcat (mc_replace_strmem.c:186)
---552---  by 0x40076d: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---  by 0x40085a: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
---552---
```
gdb: basic functionality

• Interactive debugger
  – allows the user to run a program and interactively examine its execution. Features include:
    • breakpoints (“run until control reaches here, then prompt user”)
    • stack backtrace (chain of calls leading to some point in the code)
    • examination of program variables

• Usage:
  – compile program using
    \texttt{gcc \ -g \ ...}
  – invoke the program as
    \texttt{gdb \ prog \ (then \ supply \ arguments \ inside \ gdb)}
Interactive debugging: gdb

```
% gcc str_cmp_game.c
% ./a.out
Guess a 2-digit number: 50
try lower!
25
try higher!
37
try lower!
33
try higher!
35
try higher!
36
got it!
%
% gcc buggy_str_cmp_game.c
% ./a.out
Guess a 2-digit number: 50
try lower!
25
try higher!
37
try lower!
33
try higher!
35
try higher!
36
try higher!
37
try lower!
^C
%
```

expected behavior

buggy behavior
gdb: example usage

```
% gcc -g buggy_str_cmp_game.c
% gdb ./a.out
GNU gdb (GDB) 7.0-ubuntu
Copyright (C) 2009 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".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>
Reading symbols from /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out...done.
(gdb) break main
Breakpoint 1 at 0x4007b5: file buggy_str_cmp_game.c, line 31.
(gdb) run
Starting program: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out
Breakpoint 1, main () at buggy_str_cmp_game.c:31
31  
(gdb) list
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
int main()
{ int x, n;
  char guess[MAX_GUESS_LEN];
genRandomString();
gdb) next

genRandomString();
gdb) next

printf("Guess a %d-digit number: ", MAX_STR_LEN);
gdb) n = 10;
```

**invocation**

set a breakpoint
in this case: at entry to main()

**start execution**

specify command-line
arguments here

**execute reaches**
breakpoint and returns
control to user

**examine the program**

**move to next statement**
gdb: Looking at the program

```
% gdb ./a.out
GNU gdb (GDB) 7.0-ubuntu
Copyright (C) 2009 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 "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out...done.
(gdb) list 30
25    randomstr[MAX_STR_LEN] = '\0';
26
27    }
28
29    int main( )
30    {
31        int x, n;
32        char guess[MAX_GUESS_LEN];
33
34    (gdb) list 45
35        while ( scanf("%s", guess) !- EOF & n > 0 ) {
36            n--;
37            x = strcmp(randomstr, guess);
38            if (x < 0) {
39                printf("try lower!\n");
40                continue;
41            } else if (x - 0) {
(gdb)
```

“list the program source around the line number specified”
% gdb ./a.out
GNU gdb (GDB) 7.0-ubuntu
Copyright (C) 2009 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".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>
Reading symbols from /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out...done.
(gdb) list 30
25     randomstr[MAX_STR_LEN] = '\0';
26
27 }
28
29
30     int main( )
31 {
32         int x, n;
33     char guess[MAX_GUESS_LEN];
34
(gdb) list 45
40     While ( scanf("%s", guess) != EOF & & n > 0 ) {
41         n--;
42         x = strcmp(randomstr, guess);
43         if (x < 0) {
44             printf("try lower!\n");
45             continue;
46         }
47         else if (x - 0) {
48     (gdb) b 45
Breakpoint 1 at 0x400812: file buggy_str_cmp_game.c, line 45.
(gdb) run
Starting program: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out
Guess a 2-digit number: 

The execution reaches the breakpoint and returns control to the user.

Single-step through the execution.
gdb

```
% gdb ./a.out
GNU gdb (GDB) 7.0-ubuntu
Copyright (C) 2009 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".
For bug reporting instructions, please see:
(http://www.gnu.org/software/gdb/bugs/)
Reading symbols from /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out...done.
(gdb) list 45
  40      while ( scanf("%s", guess) != EOF && n > 0 ) {
  41          n--; 
  42          x = strcmp(randomstr, guess);
  43          if (x < 0) {
  44              printf("try lower\n");
  45              continue;
  46          } else if (x - 0) {
(gdb) break 45
Breakpoint 1 at 0x400812: file buggy_strcmp_game.c, line 45.
(gdb) run
Starting program: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out
Guess a 2-digit number: 50
Breakpoint 1, main () at buggy_strcmp_game.c:45
  45      if (x < 0) {
(gdb) print x
$1 = -1
(gdb) cont
Continuing.
try lower!
37
Breakpoint 1, main () at buggy_strcmp_game.c:45
  45      if (x < 0) {
(gdb) ```
% cat segfault-debug.c
// File: dangling-ptr-1.c
// Purpose: To illustrate dangling pointers

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

// read_string(str) -- reads a string into buffer str. Returns str if a string was successfully read. NULL otherwise.
char *read_string(char *str) {
    int status = scanf("%s", str);
    if (status > 0) {
        return str;
    } else {
        return NULL;
    }
}

// my_read() -- reads a string into a buffer and returns a pointer
// to that buffer.
char *my_read(char *buf) {
    return read_string(buf);
}

int main() {
    char *string;
    my_read(string);
    printf("\%s -- length = %d\n", string, (int)strlen(string));
    return 0;
}
% gcc -Wall segfault-debug.c
% ./a.out
abcde
Segmentation fault
%
gdb: moving around the runtime stack

where did the Seg Fault occur?
move up the stack (i.e., to the caller) to examine variable values

1. gcc -g segfault-debug.c
2. ./a.out
3. abcde
4. Segmentation fault
5. gdb ./a.out
6. GNU gdb Fedora (6.8-23.fc9)
7. Copyright (C) 2008 Free Software Foundation, Inc.
8. License GPLv3+: GNU GPL Version 3 or later <http://gnu.org/licenses/gpl.html>
9. This is free software: you are free to change and redistribute it.
10. There is NO WARRANTY, to the extent permitted by law. Type "show copying"
11. and "show warranty" for details.
12. This GDB was configured as "x86_64-redhat-linux-gnu"...
13. (gdb) run
15. abcde
16. Program received signal SIGSEGV, Segmentation fault.
17. 0x0000003d90480f80 in strlen () from /lib64/libc.so.6
18. Missing separate debuginfos, use: debuginfo-install glibc.x86_64
19. (gdb) where
20. #0 0x0000003d90480f80 in strlen () from /lib64/libc.so.6
21. #1 0x00000000004005cd in main () at segfault-debug.c:28
22. (gdb) bt
23. #0 0x0000003d90480f80 in strlen () from /lib64/libc.so.6
24. #1 0x00000000004005cd in main () at segfault-debug.c:28
25. (gdb) up
26. #1 0x00000000004005cd in main () at segfault-debug.c:28
27. 28  printf("%s -- length = %d\n", string, (int)strlen(string));
28  (gdb) print string
29. $1 = 0x0
30. (gdb) quit
31. The program is running. Exit anyway? (y or n) y
32. %
gdb: other features

• Gdb provides many other debugging features, e.g.:
  – conditional breakpoints
    • “break execution at some point in the code and return control to the user if some condition holds”
  – watchpoints
    • “break execution and return control to user if a variable is read or written’
    – change the value of a variable in the program state

• See tutorials in the DOCS area of class website
gdb: reading commands from a file

```
% cat > infile
abcede
%
% cat > gdbcmds
run < infile
up
print string
quit
Y
%
% more gdbcmds
run < infile
up
print string
quit
Y
%
% gdb ./a.out -x gdbcmds
GNU gdb Fedora (6.8-23.fc9)
Copyright (C) 2008 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-redhat-linux-gnu"...

Program received signal SIGSEGV, Segmentation fault.
0x00000000000400f8 in strlen () from /lib64/libc.so.6
#1 0x000000000000005cd in main () at segfault-debug.c:28
28   printf("%s -- length = %d\n", string, (int)strlen(string));
%1 = 0x0
The program is running. Exit anyway? (y or n) [answered Y; input not from terminal]
```