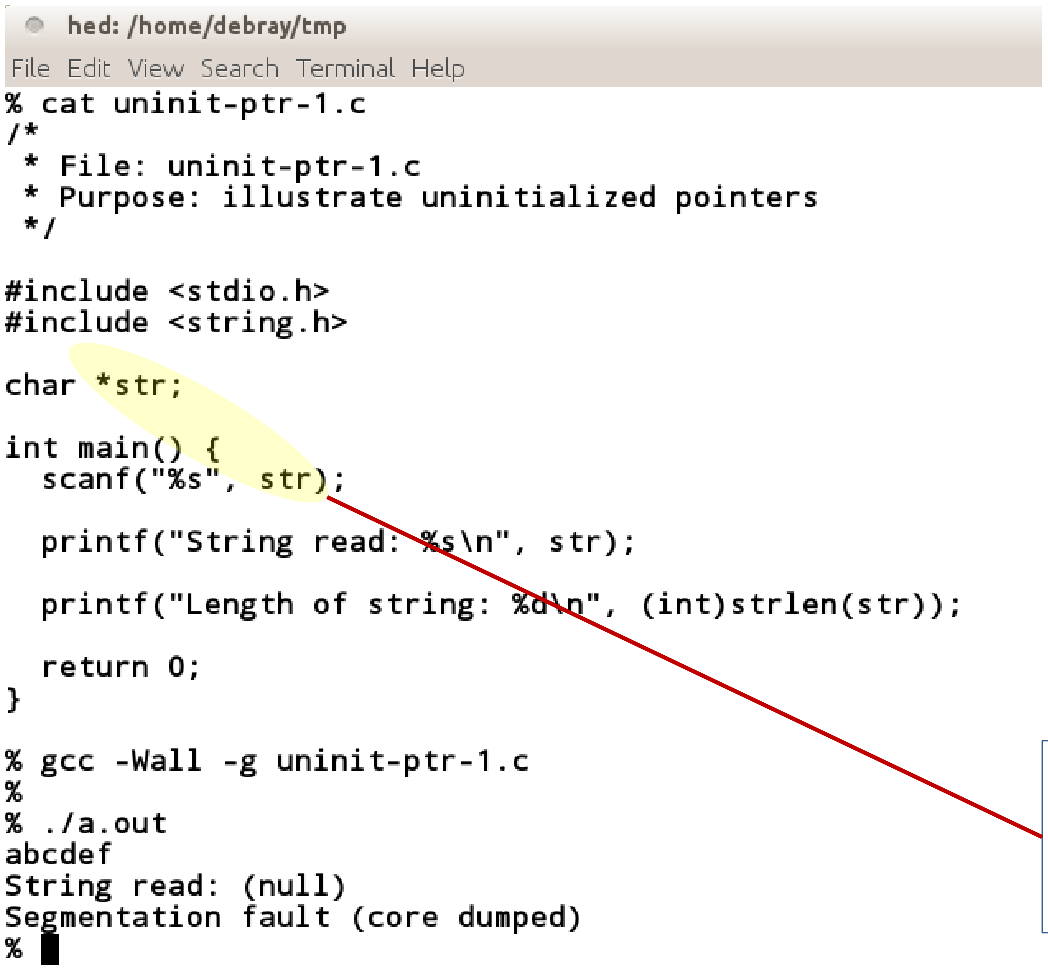
CSc 352
Debugging Tools

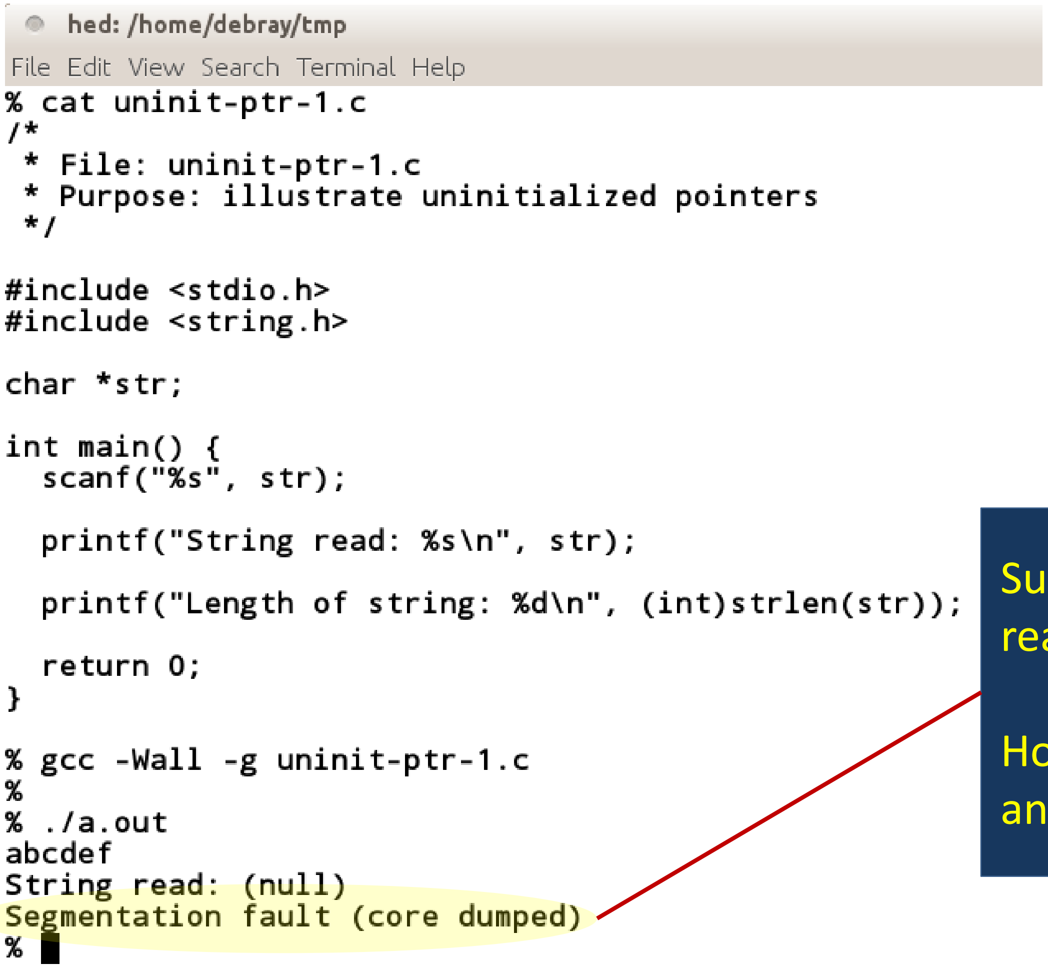
Uninitialized pointers



```
% cat uninit-ptr-1.c
/*
 * File: uninit-ptr-1.c
 * Purpose: illustrate 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;
}

% gcc -Wall -g uninit-ptr-1.c
%
% ./a.out
abcdef
String read: (null)
Segmentation fault (core dumped)
% 
```

str was never initialized to point to anything

Uninitialized pointers

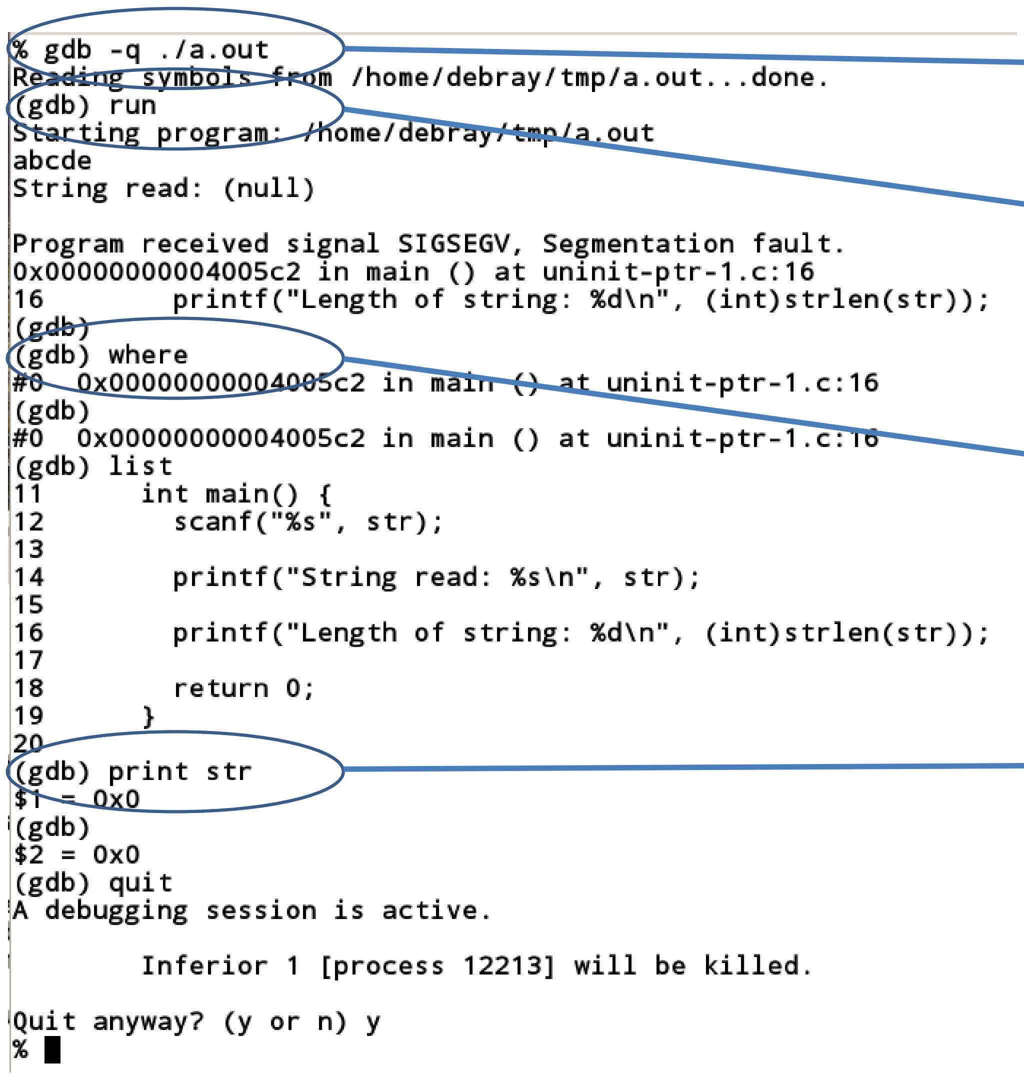


```
% cat uninit-ptr-1.c
/*
 * File: uninit-ptr-1.c
 * Purpose: illustrate uninitialized pointers
 */

#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;
}

% gcc -Wall -g uninit-ptr-1.c
%
% ./a.out
abcdef
String read: (null)
Segmentation fault (core dumped)
%
```

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  0x00000000004005c2 in main () at uninit-ptr-1.c:16
(gdb)
#0  0x00000000004005c2 in main () at uninit-ptr-1.c:16
(gdb) list
11        int main() {
12          scanf("%s", str);
13
14          printf("String read: %s\n", str);
15
16          printf("Length of string: %d\n", (int)strlen(str));
17
18          return 0;
19        }
20
(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
% 
```

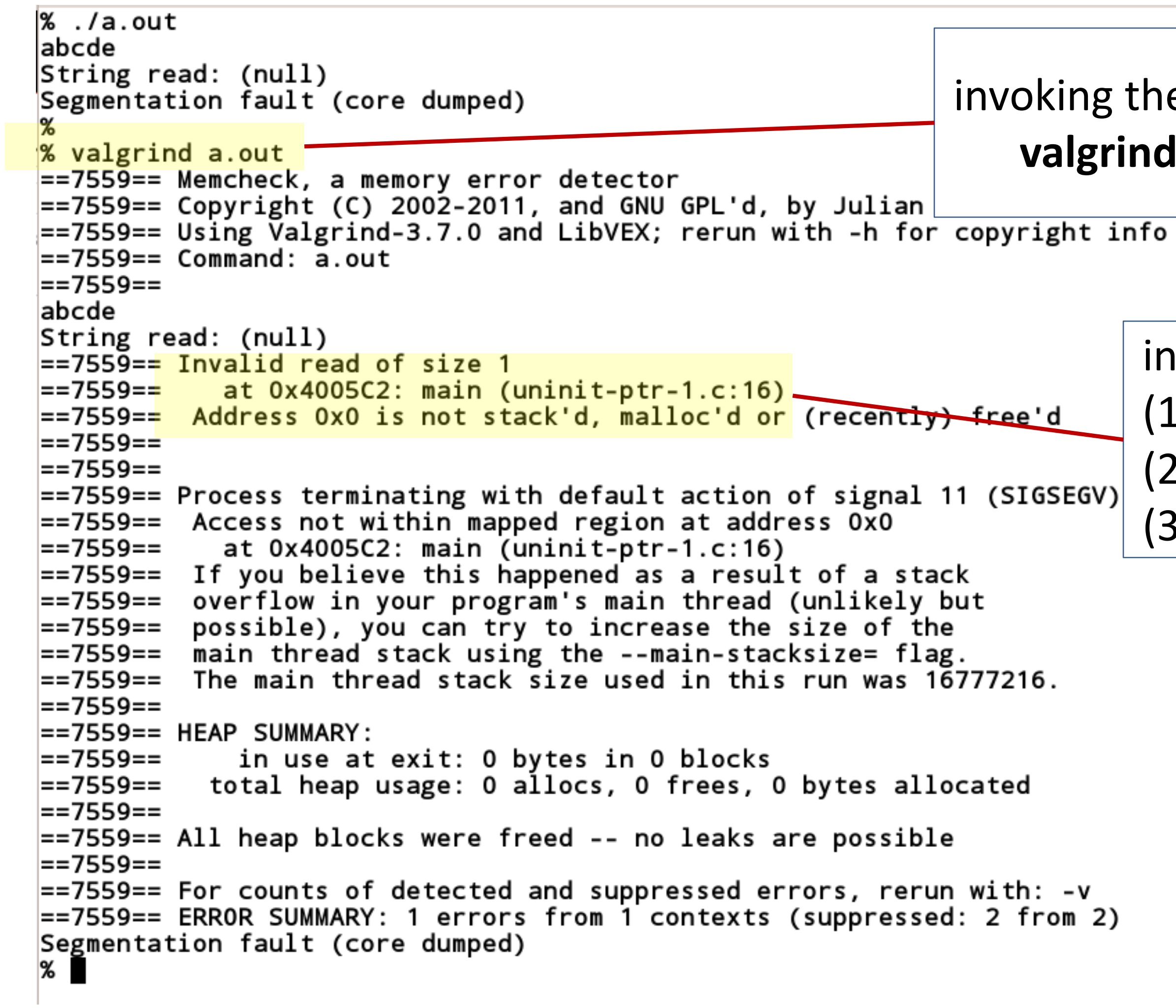
load the program into gdb

run the program within gdb

show where execution stopped

show values of variables

Memory error diagnosis: valgrind



```
% ./a.out
abcde
String read: (null)
Segmentation fault (core dumped)
%
% valgrind a.out
==7559== Memcheck, a memory error detector
==7559== Copyright (C) 2002-2011, and GNU GPL'd, by Julian
==7559== Using Valgrind-3.7.0 and LibVEX; rerun with -h for copyright info
==7559== Command: a.out
==7559==
abcde
String read: (null)
==7559== Invalid read of size 1
==7559==    at 0x4005C2: main (uninit-ptr-1.c:16)
==7559==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==7559==
==7559==
==7559== Process terminating with default action of signal 11 (SIGSEGV)
==7559==  Access not within mapped region at address 0x0
==7559==    at 0x4005C2: main (uninit-ptr-1.c:16)
==7559==  If you believe this happened as a result of a stack
==7559==  overflow in your program's main thread (unlikely but
==7559==  possible), you can try to increase the size of the
==7559==  main thread stack using the --main-stacksize= flag.
==7559==  The main thread stack size used in this run was 16777216.
==7559==
==7559== HEAP SUMMARY:
==7559==     in use at exit: 0 bytes in 0 blocks
==7559==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==7559==
==7559== All heap blocks were freed -- no leaks are possible
==7559==
==7559== For counts of detected and suppressed errors, rerun with: -v
==7559== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 2 from 2)
Segmentation fault (core dumped)
% █
```

invoking the tool:

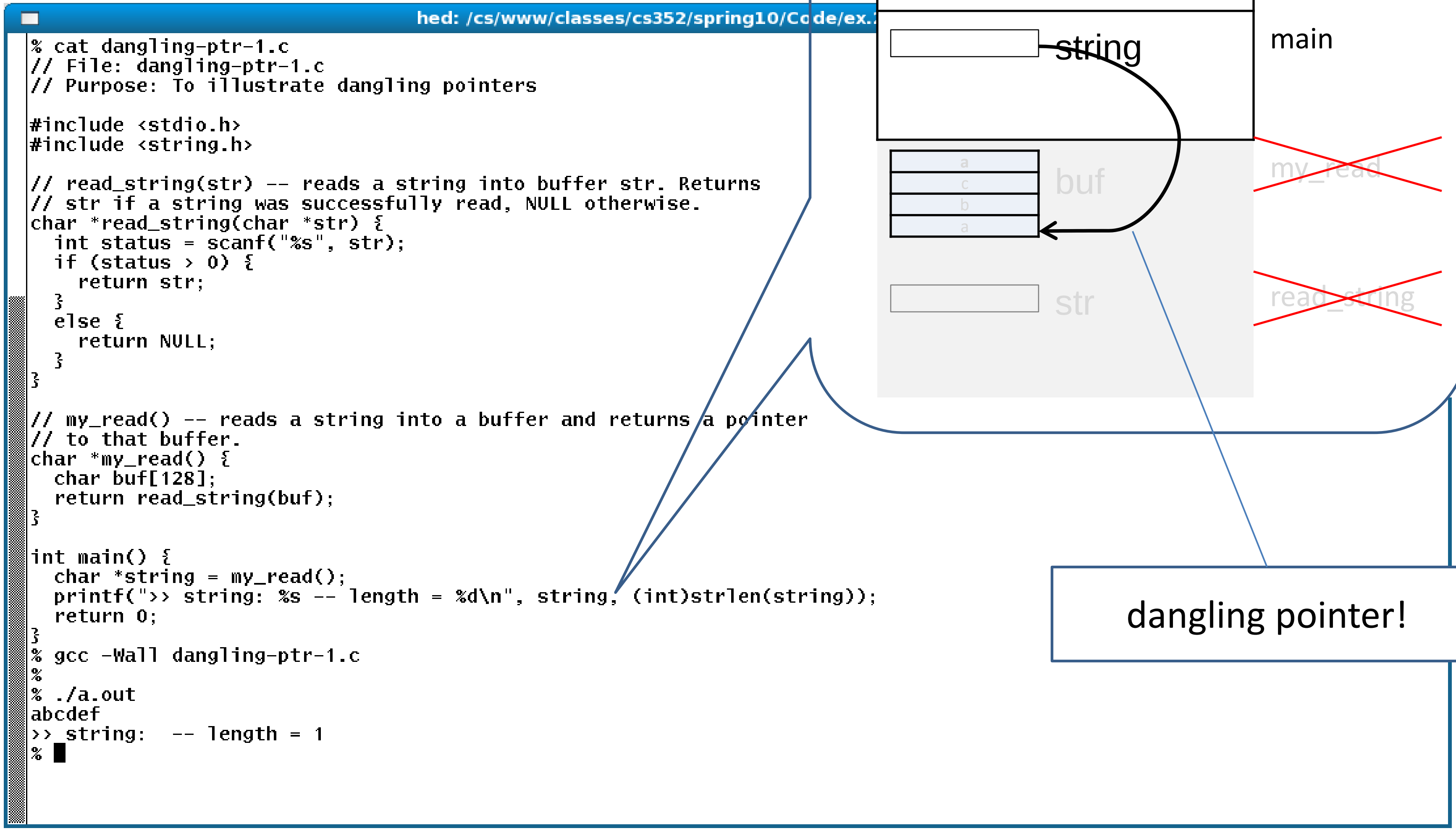
valgrind progName arg_1 arg_2 …

indicates:

1. there was a problem;
2. what happened
3. where it happened

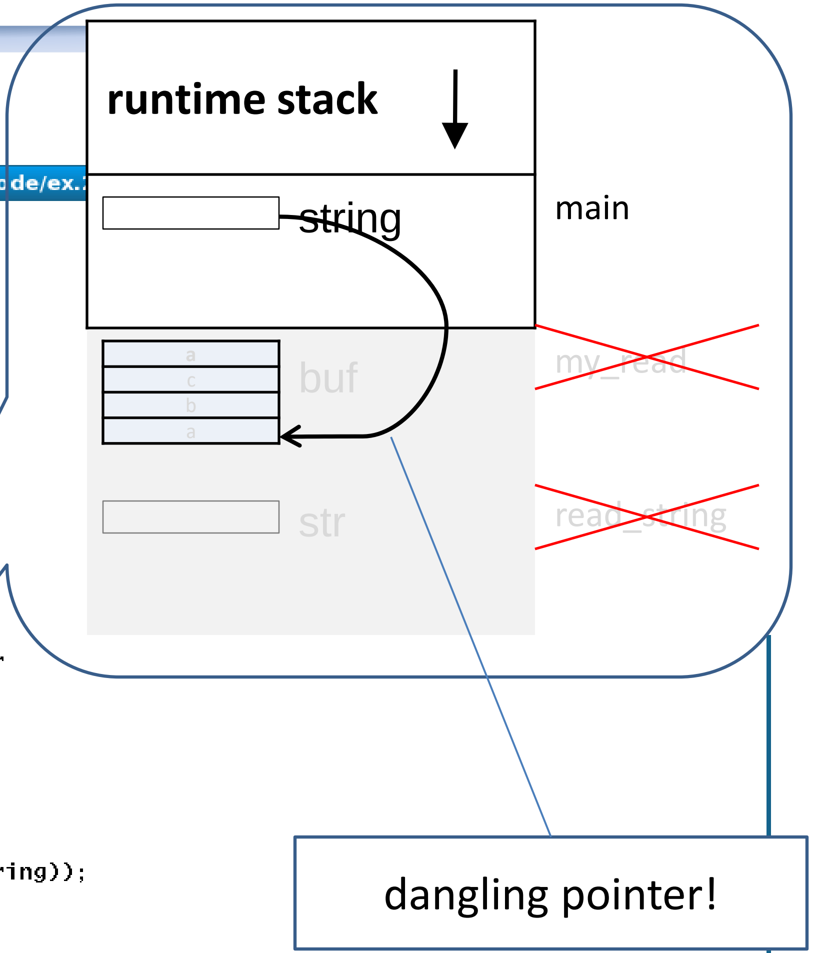
Dangling pointers

We looked at this code earlier:



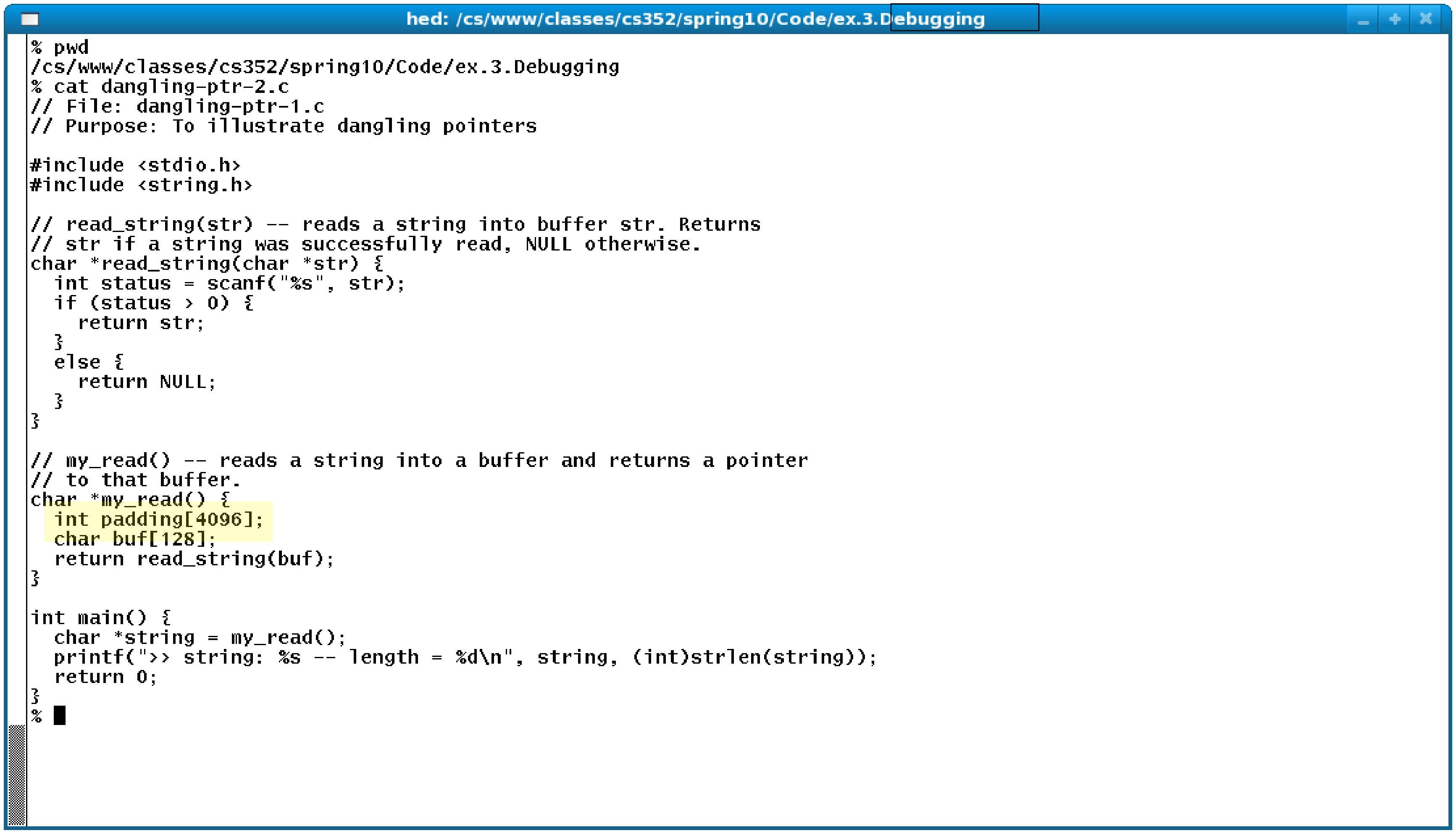
hed: /cs/www/classes/cs352/spring10/Code/ex.

```
% 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, (int)strlen(string));
  return 0;
}
% gcc -Wall dangling-ptr-1.c
%
% ./a.out
abcdef
>> string:  -- length = 1
% 
```



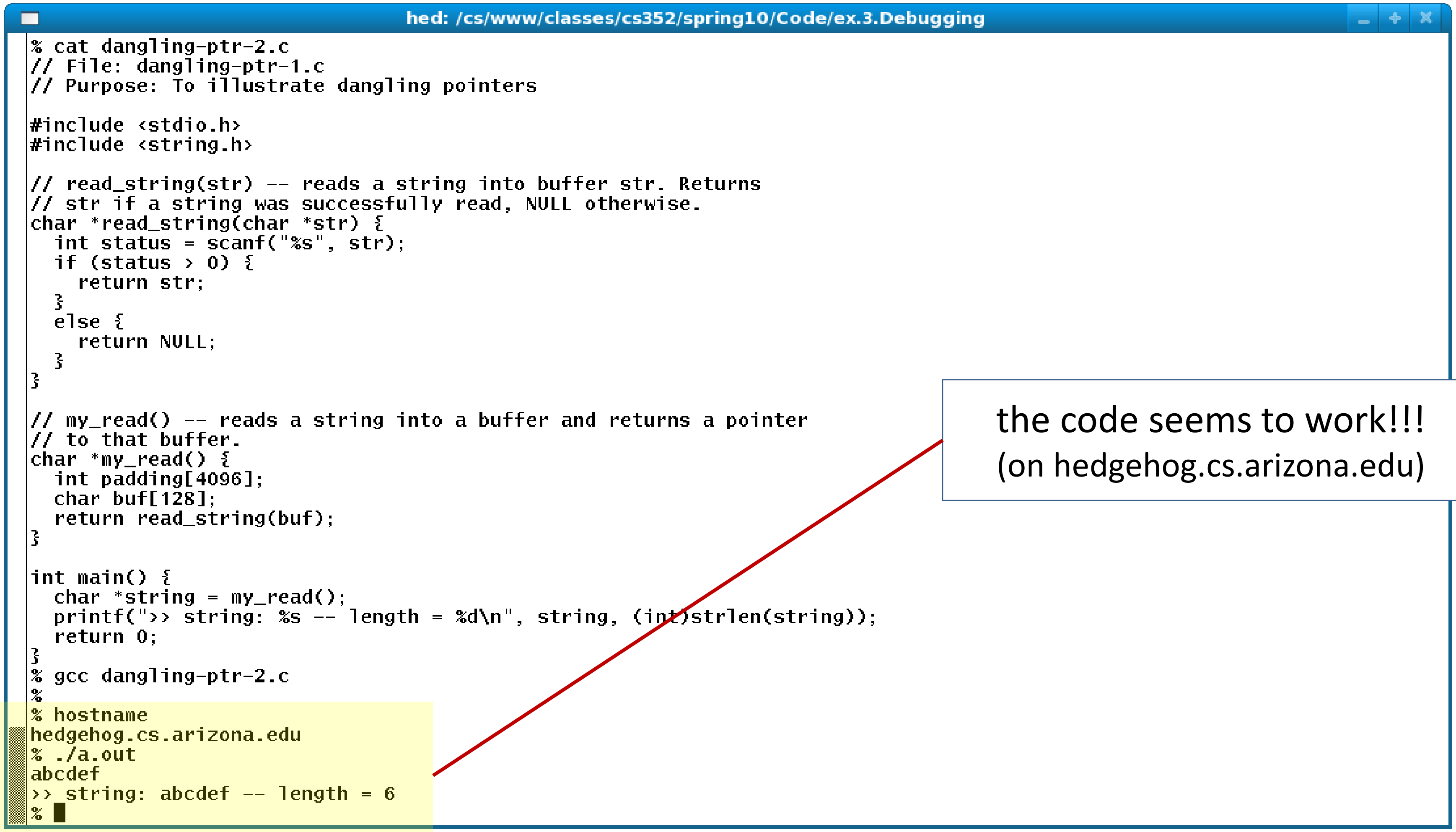
Dangling pointers

Minor variation on this code:



```
% 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(">> string: %s -- length = %d\n", string, (int)strlen(string));
  return 0;
}
% 
```

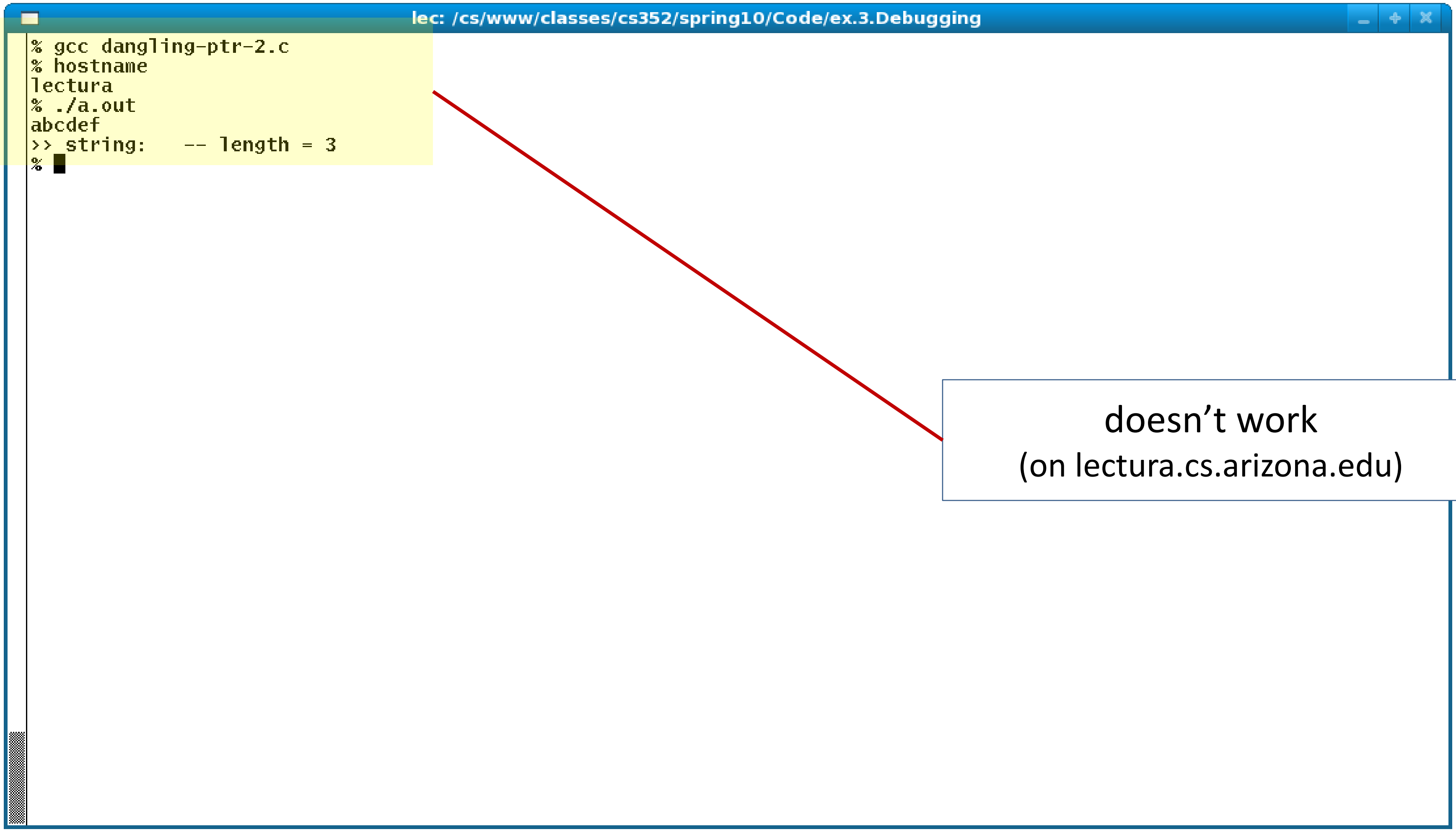
Dangling pointers



```
% 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(">> string: %s -- length = %d\n", string, (int)strlen(string));
  return 0;
}
% gcc dangling-ptr-2.c
%
% hostname
hedgehog.cs.arizona.edu
% ./a.out
abcdef
>> string: abcdef -- length = 6
%
```

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

Dangling pointers



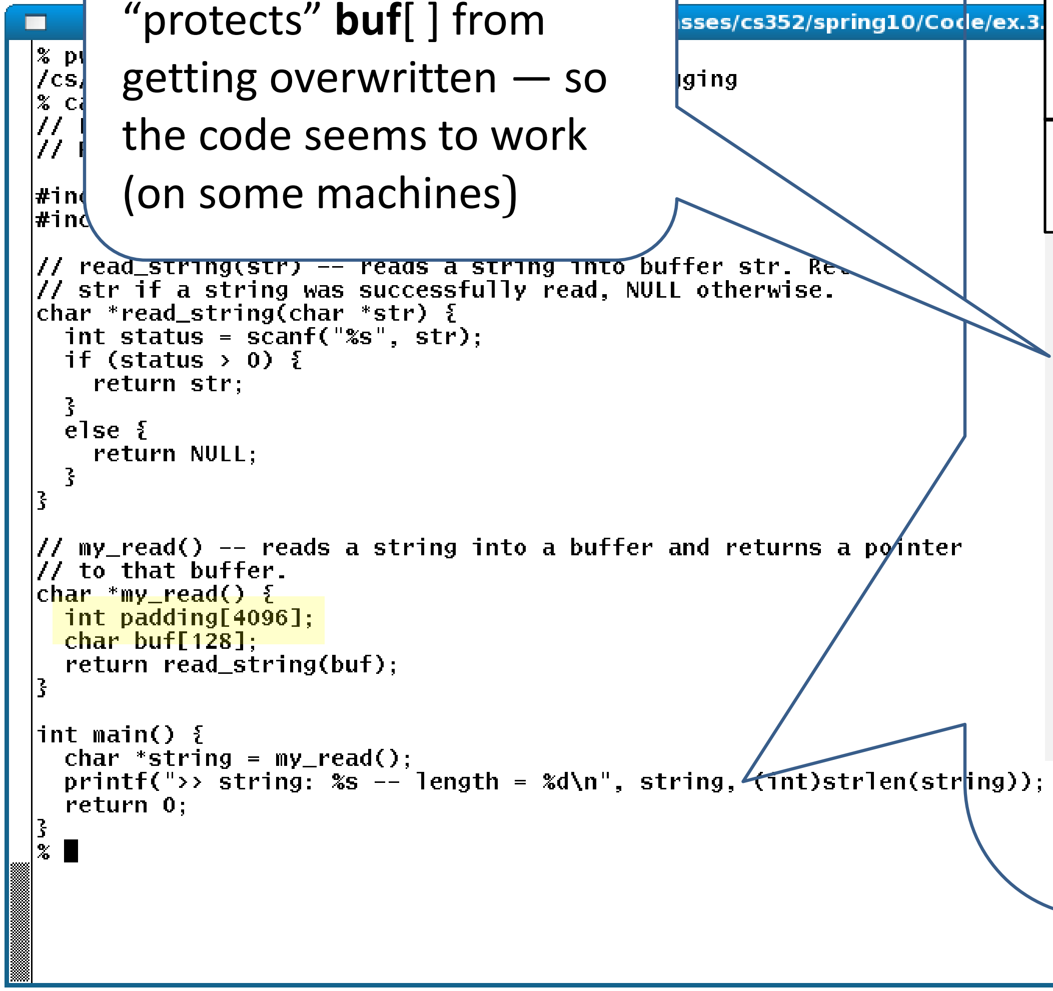
lec: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging

```
% gcc dangling-ptr-2.c
% hostname
lectura
% ./a.out
abcdef
>> string:    -- length = 3
%
```

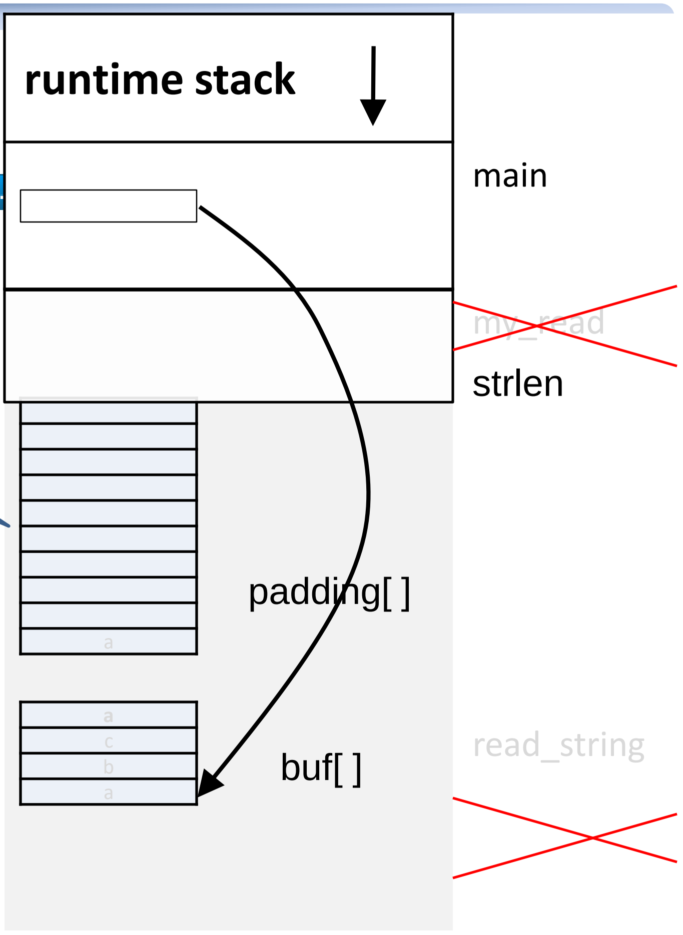
doesn’t work
(on lectura.cs.arizona.edu)

What's going on?

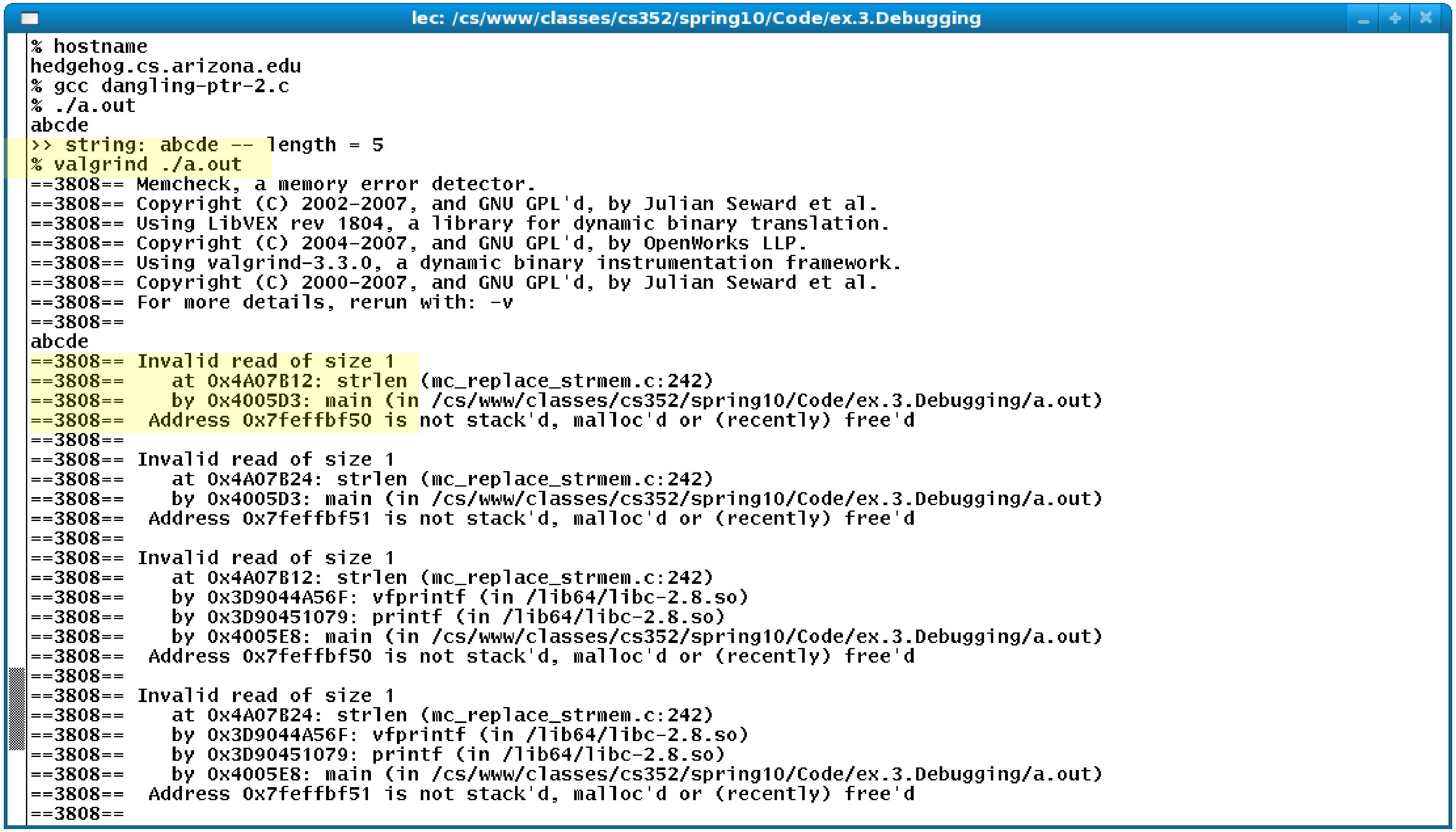
the array **padding**[ ] “protects” **buf**[ ] from getting overwritten — so the code seems to work (on some machines)



```
% p
/cs/
% c
// 
// 

#in
#in

// read_string(str) -- reads a string into buffer str. Re
// 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(">> string: %s -- length = %d\n", string, (int)strlen(string));
  return 0;
}
% 
```



More diagnosis



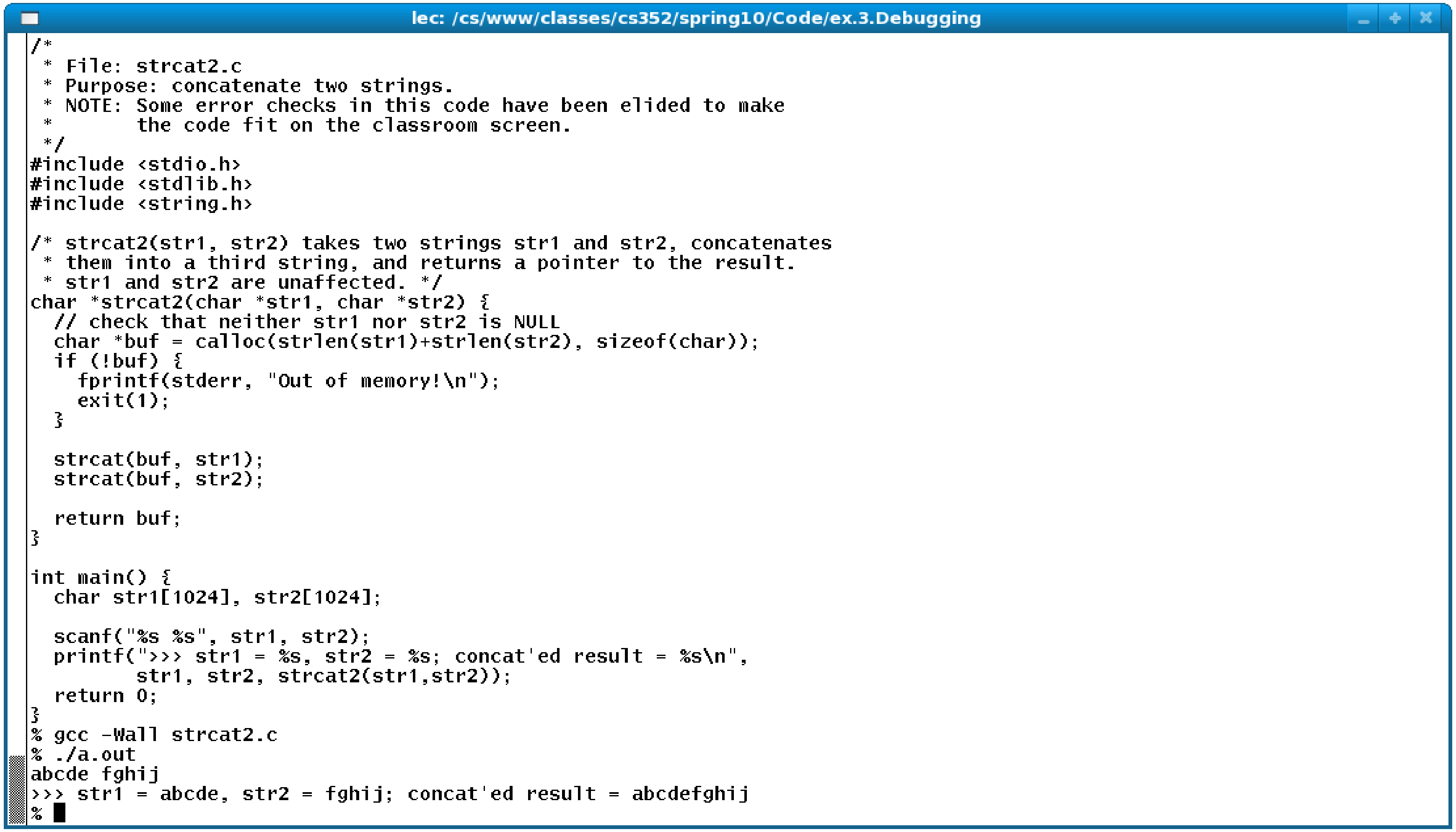
lec: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging

```
% hostname
hedgehog.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
==3808==
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==
==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==
==3808== Invalid read of size 1
==3808==    at 0x4A07B12: strlen (mc_replace_strmem.c:242)
==3808==    by 0x3D9044A56F: vfprintf (in /lib64/libc-2.8.so)
==3808==    by 0x3D90451079: 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==
==3808== Invalid read of size 1
==3808==    at 0x4A07B24: strlen (mc_replace_strmem.c:242)
==3808==    by 0x3D9044A56F: vfprintf (in /lib64/libc-2.8.so)
==3808==    by 0x3D90451079: 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
==3808==
```

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.

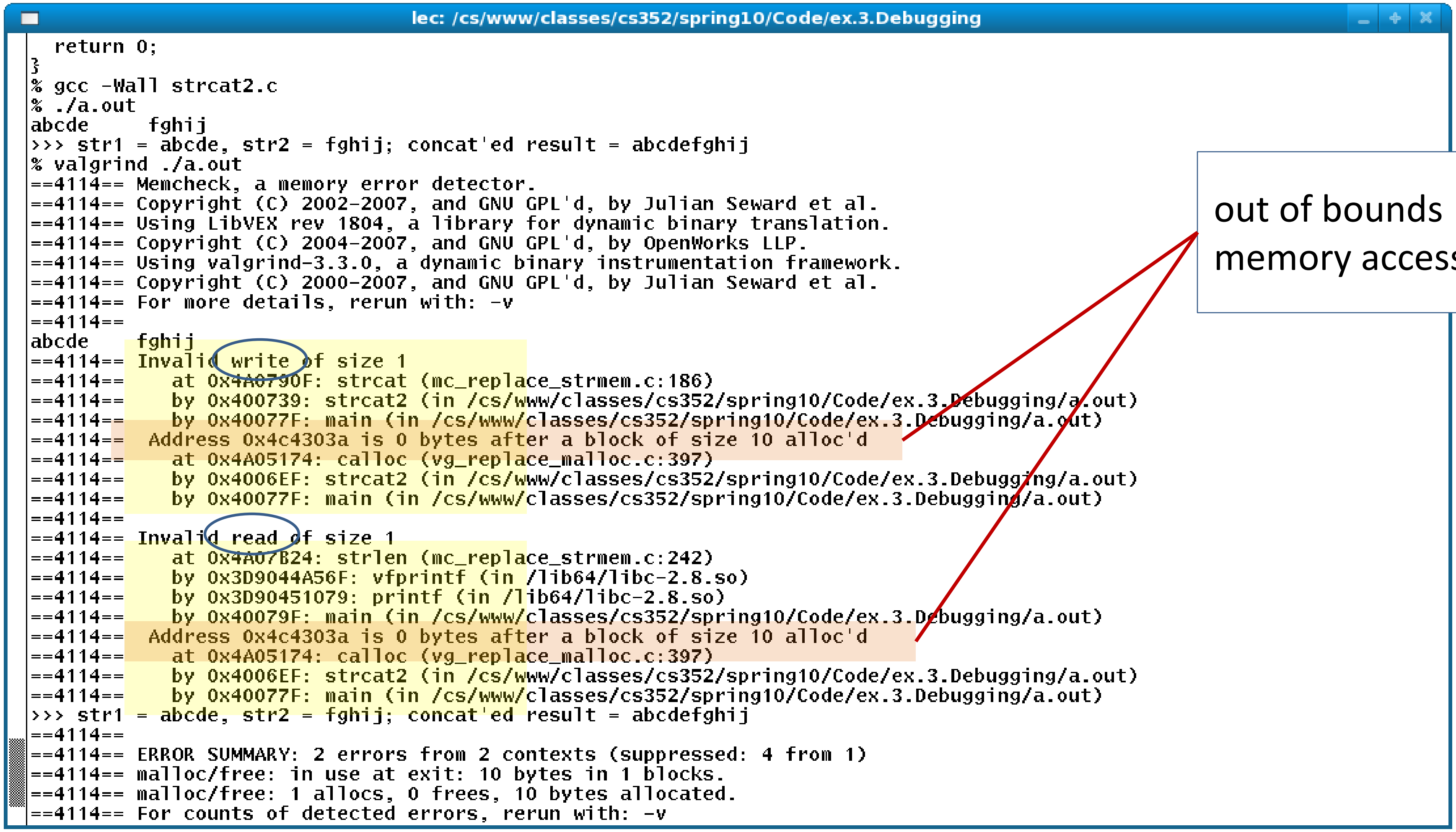
Another example



lec: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging

```
/*
 * 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
 * them 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 = calloc(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(">>> str1 = %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
% 
```

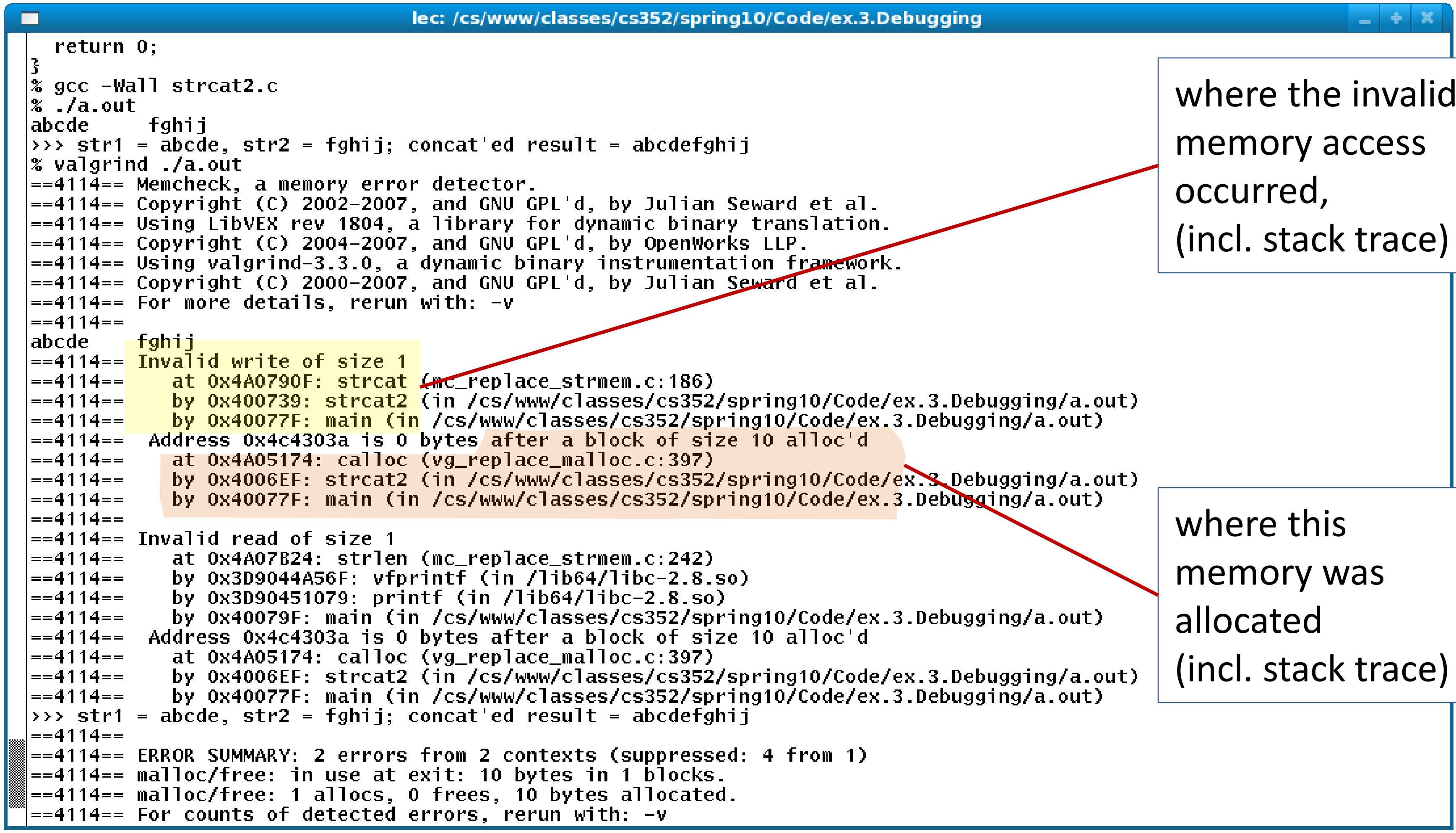
Example 2



```
  return 0;
}
% gcc -Wall strcat2.c
% ./a.out
abcde     fghij
>>> str1 = abcde, str2 = fghij; concat'ed result = abcdefghij
% 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==
abcde    fghij
==4114== Invalid write of size 1
==4114==    at 0x4A0790F: strcat (mc_replace_strmem.c:186)
==4114==    by 0x400739: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==4114==    by 0x40077F: 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 0x40077F: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==4114==
==4114== Invalid read of size 1
==4114==    at 0x4A07B24: strlen (mc_replace_strmem.c:242)
==4114==    by 0x3D9044A56F: vfprintf (in /lib64/libc-2.8.so)
==4114==    by 0x3D90451079: printf (in /lib64/libc-2.8.so)
==4114==    by 0x40079F: 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 0x40077F: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
>>> str1 = abcde, str2 = fghij; concat'ed result = abcdefghij
==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
```

out of bounds memory access

Example 2

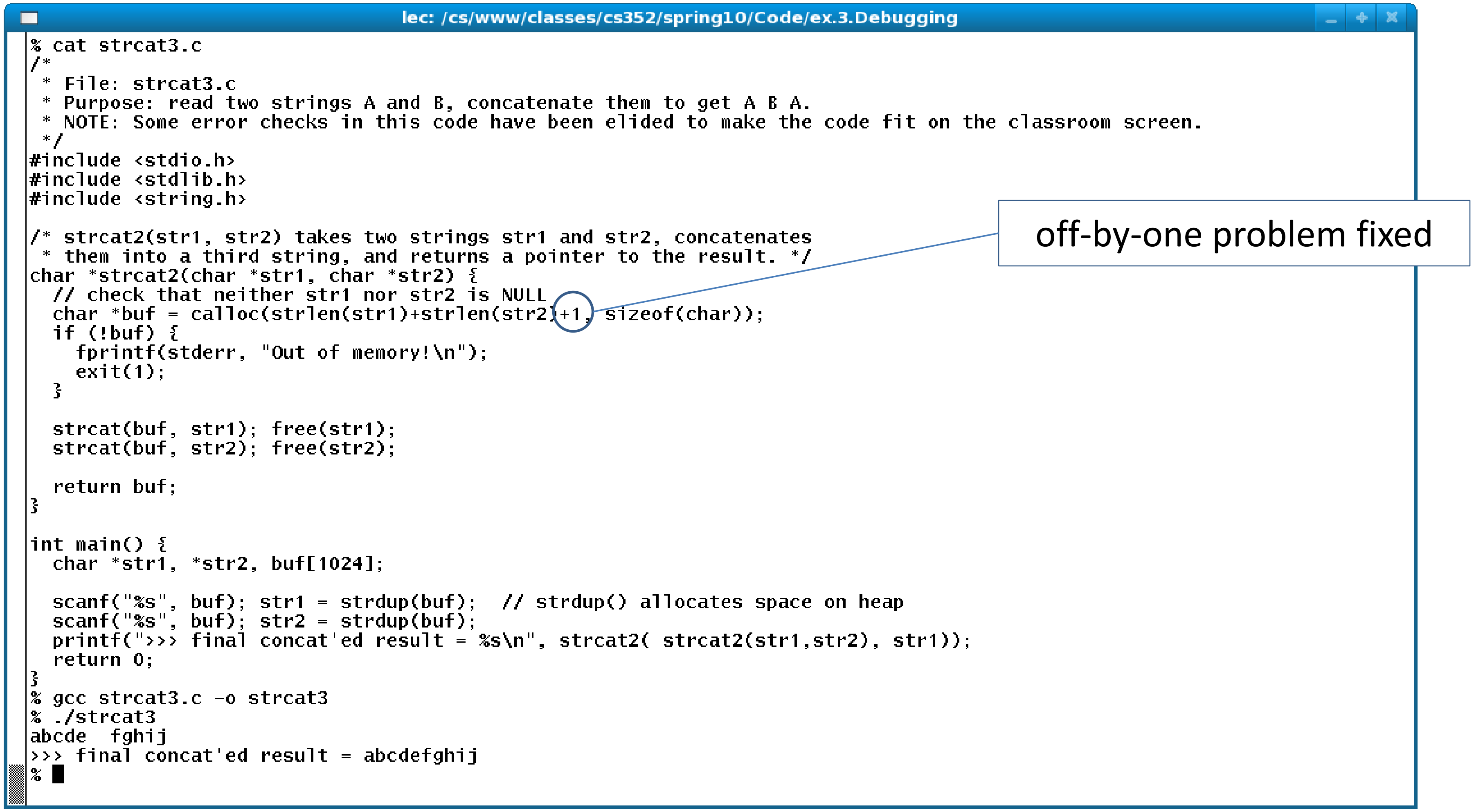


```
  return 0;
}
% gcc -Wall strcat2.c
% ./a.out
abcde     fghij
>>> str1 = abcde, str2 = fghij; concat'ed result = abcdefghij
% valgrind ./a.out
==4114== Memcheck, a memory error detector.
==4114== Copyright (C) 2002-2007, and GNU GPL'd, by Julian Seward et al.
==4114== Using LibVEX rev 1804, a library 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==
abcde     fghij
==4114== Invalid write of size 1
==4114==    at 0x4A0790F: strcat (mc_replace_strmem.c:186)
==4114==    by 0x400739: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==4114==    by 0x40077F: 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 0x40077F: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
==4114==
==4114== Invalid read of size 1
==4114==    at 0x4A07B24: strlen (mc_replace_strmem.c:242)
==4114==    by 0x3D9044A56F: vfprintf (in /lib64/libc-2.8.so)
==4114==    by 0x3D90451079: printf (in /lib64/libc-2.8.so)
==4114==    by 0x40079F: 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 0x40077F: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out)
>>> str1 = abcde, str2 = fghij; concat'ed result = abcdefghij
==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
```

where the invalid memory access occurred, (incl. stack trace)

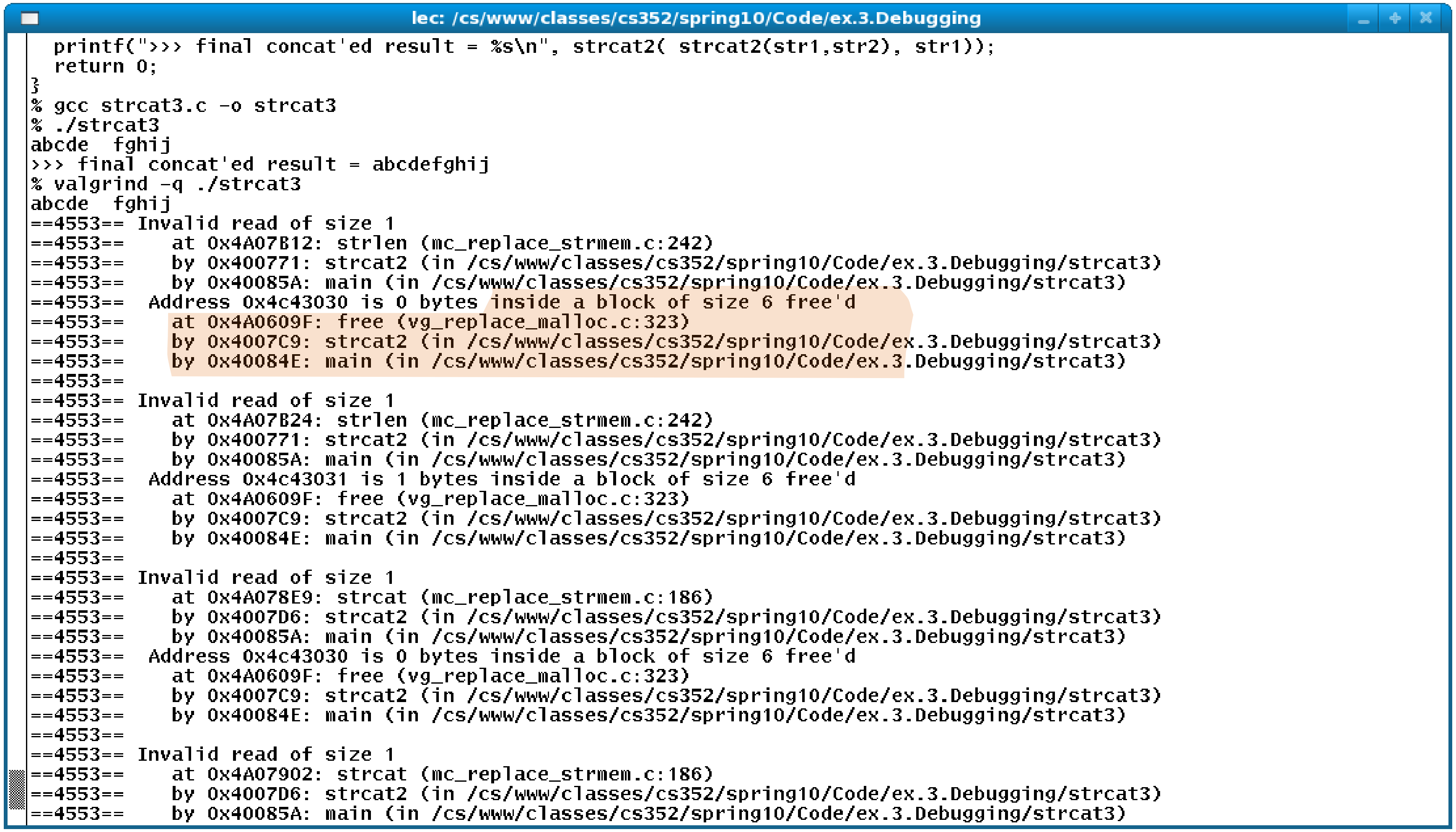
where this memory was allocated (incl. stack trace)

Example 3



```
% 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 concat'ed result = %s\n", strcat2( strcat2(str1,str2), str1));
  return 0;
}
% gcc strcat3.c -o strcat3
% ./strcat3
abcde  fghij
>>> final concat'ed result = abcdefghij
% 
```

Example 3



```
  printf(">>> final concat'ed result = %s\n", strcat2( strcat2(str1,str2), str1));
  return 0;
}
% gcc strcat3.c -o strcat3
% ./strcat3
abcde  fghij
>>> final concat'ed result = abcdefghij
% valgrind -q ./strcat3
abcde  fghij
==4553== Invalid read of size 1
==4553==    at 0x4A07B12: strlen (mc_replace_strmem.c:242)
==4553==    by 0x400771: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==    by 0x40085A: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==  Address 0x4c43030 is 0 bytes inside a block of size 6 free'd
==4553==    at 0x4A0609F: free (vg_replace_malloc.c:323)
==4553==    by 0x4007C9: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==    by 0x40084E: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==
==4553== Invalid read of size 1
==4553==    at 0x4A07B24: strlen (mc_replace_strmem.c:242)
==4553==    by 0x400771: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==    by 0x40085A: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==  Address 0x4c43031 is 1 bytes inside a block of size 6 free'd
==4553==    at 0x4A0609F: free (vg_replace_malloc.c:323)
==4553==    by 0x4007C9: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==    by 0x40084E: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==
==4553== Invalid read of size 1
==4553==    at 0x4A078E9: strcat (mc_replace_strmem.c:186)
==4553==    by 0x4007D6: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==    by 0x40085A: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==  Address 0x4c43030 is 0 bytes inside a block of size 6 free'd
==4553==    at 0x4A0609F: free (vg_replace_malloc.c:323)
==4553==    by 0x4007C9: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==    by 0x40084E: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==
==4553== Invalid read of size 1
==4553==    at 0x4A07902: strcat (mc_replace_strmem.c:186)
==4553==    by 0x4007D6: strcat2 (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
==4553==    by 0x40085A: main (in /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/strcat3)
```

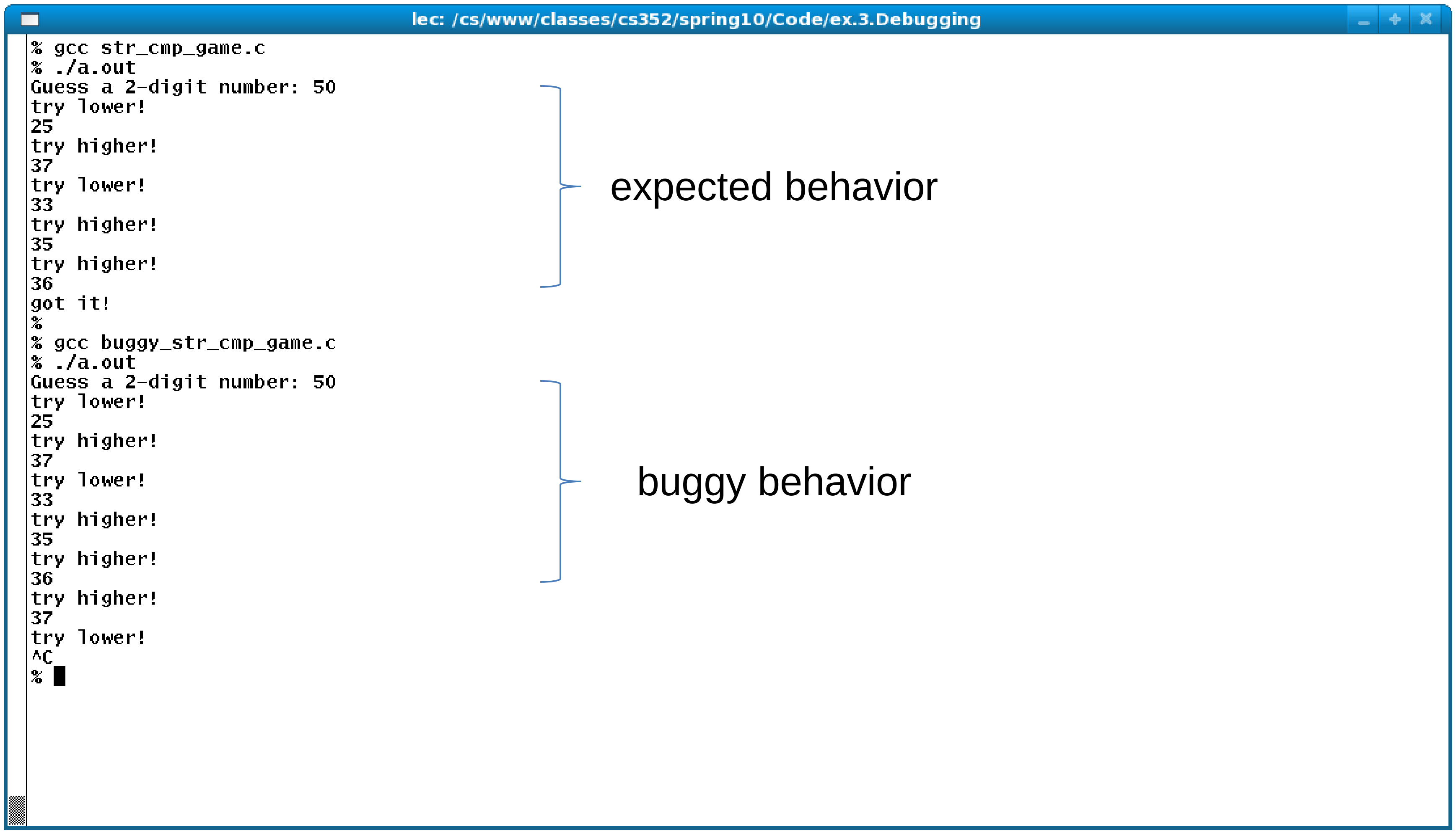
gdb: basic functionality

- 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

    gcc **–g** …
  - invoke the program as

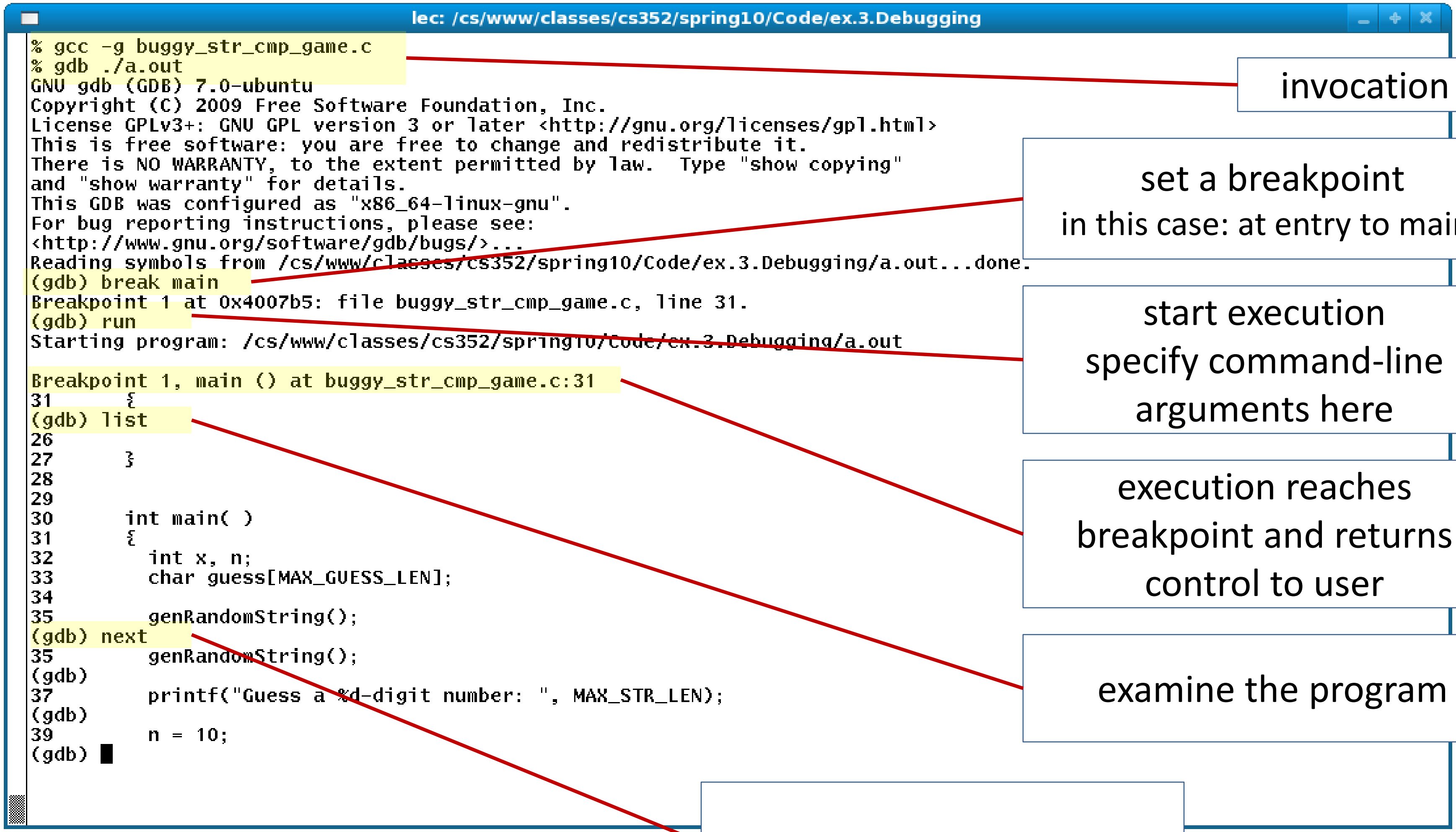
    gdb *prog* *(then supply arguments inside gdb)*

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
%
```

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        int main( )
31        {
32          int x, n;
33          char guess[MAX_GUESS_LEN];
34
35          genRandomString();
(gdb) next
35          genRandomString();
(gdb)
37          printf("Guess a %d-digit number: ", MAX_STR_LEN);
(gdb)
39          n = 10;
(gdb) █
```

invocation

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

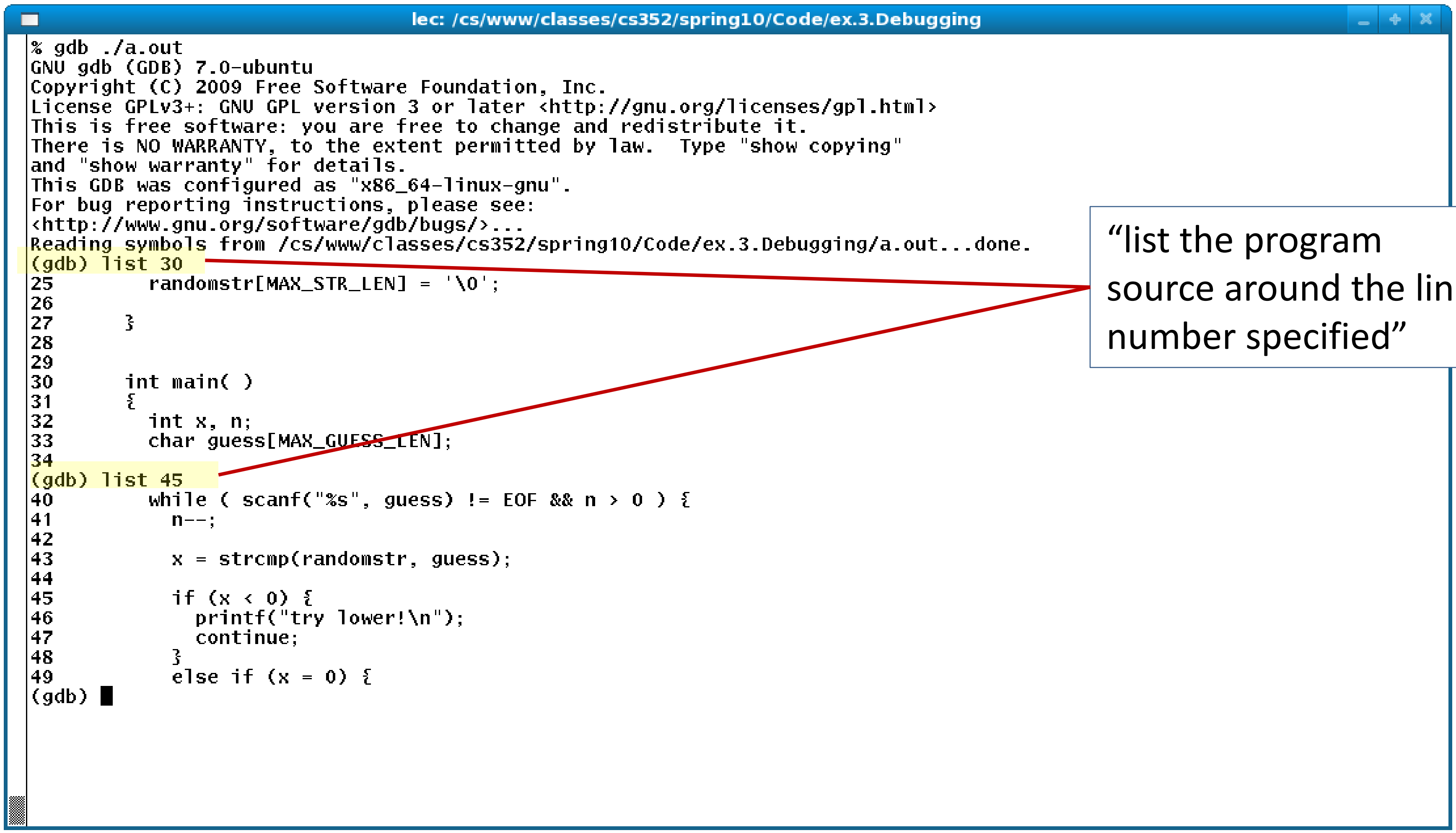
start execution
specify command-line arguments here

execution reaches breakpoint and returns control to user

examine the program

move to next statement

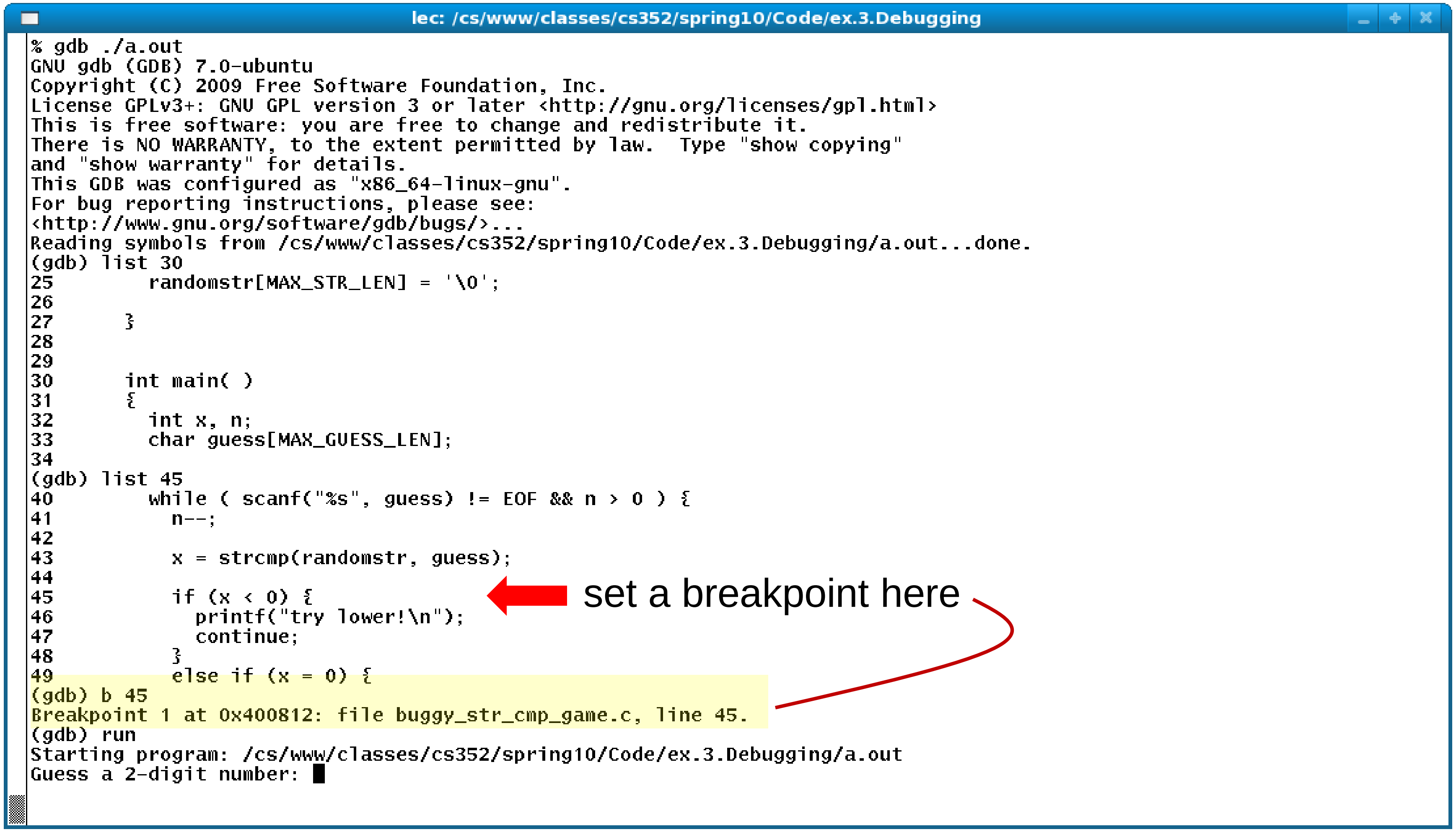
gdb: Looking at the program



```
% 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 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
43            x = strcmp(randomstr, guess);
44
45            if (x < 0) {
46              printf("try lower!\n");
47              continue;
48            }
49            else if (x = 0) {
(gdb) █
```

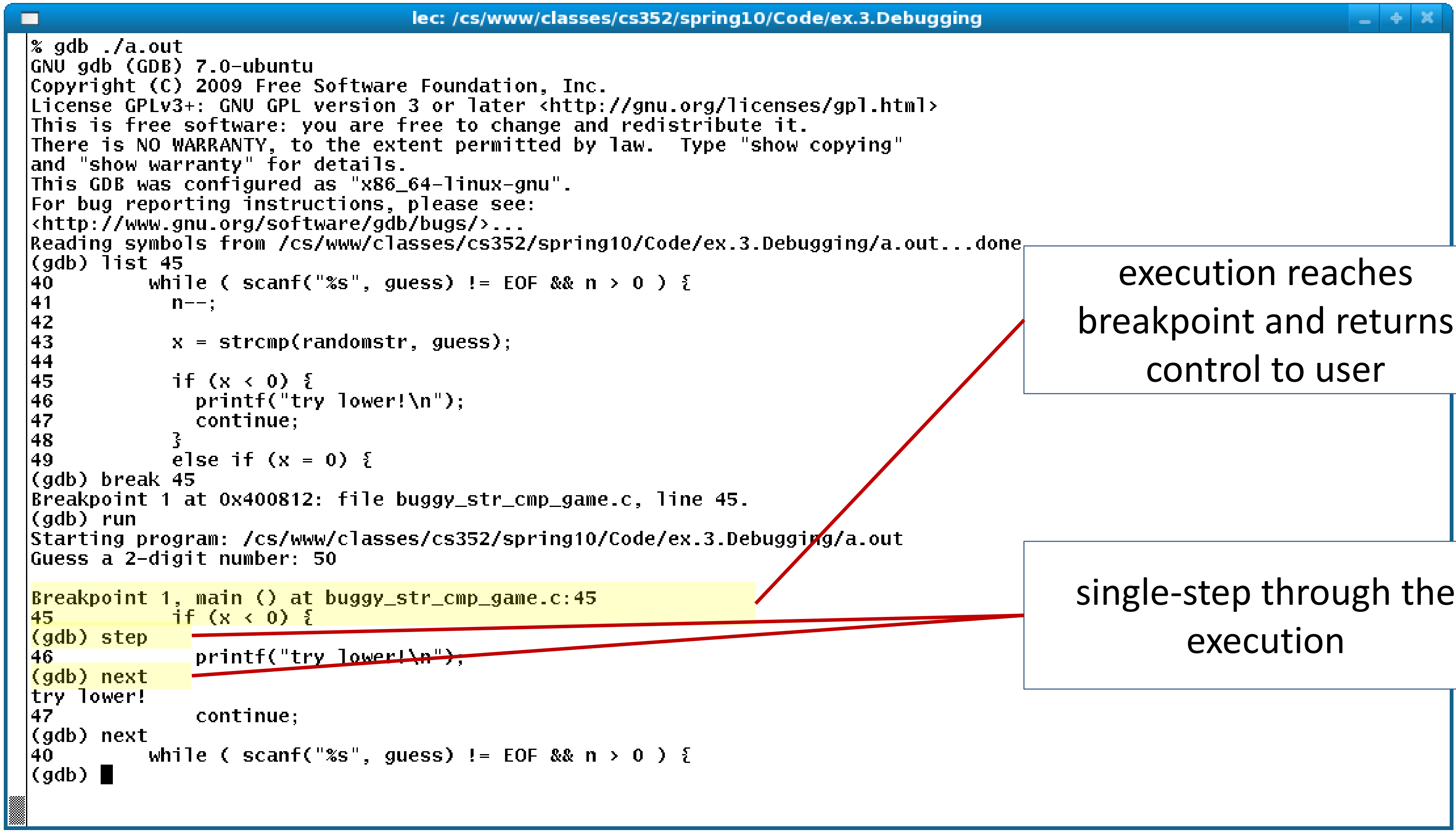
“list the program source around the line number specified”

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 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
43            x = strcmp(randomstr, guess);
44
45            if (x < 0) {
46              printf("try lower!\n");
47              continue;
48            }
49            else if (x = 0) {
(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: 
```

gdb



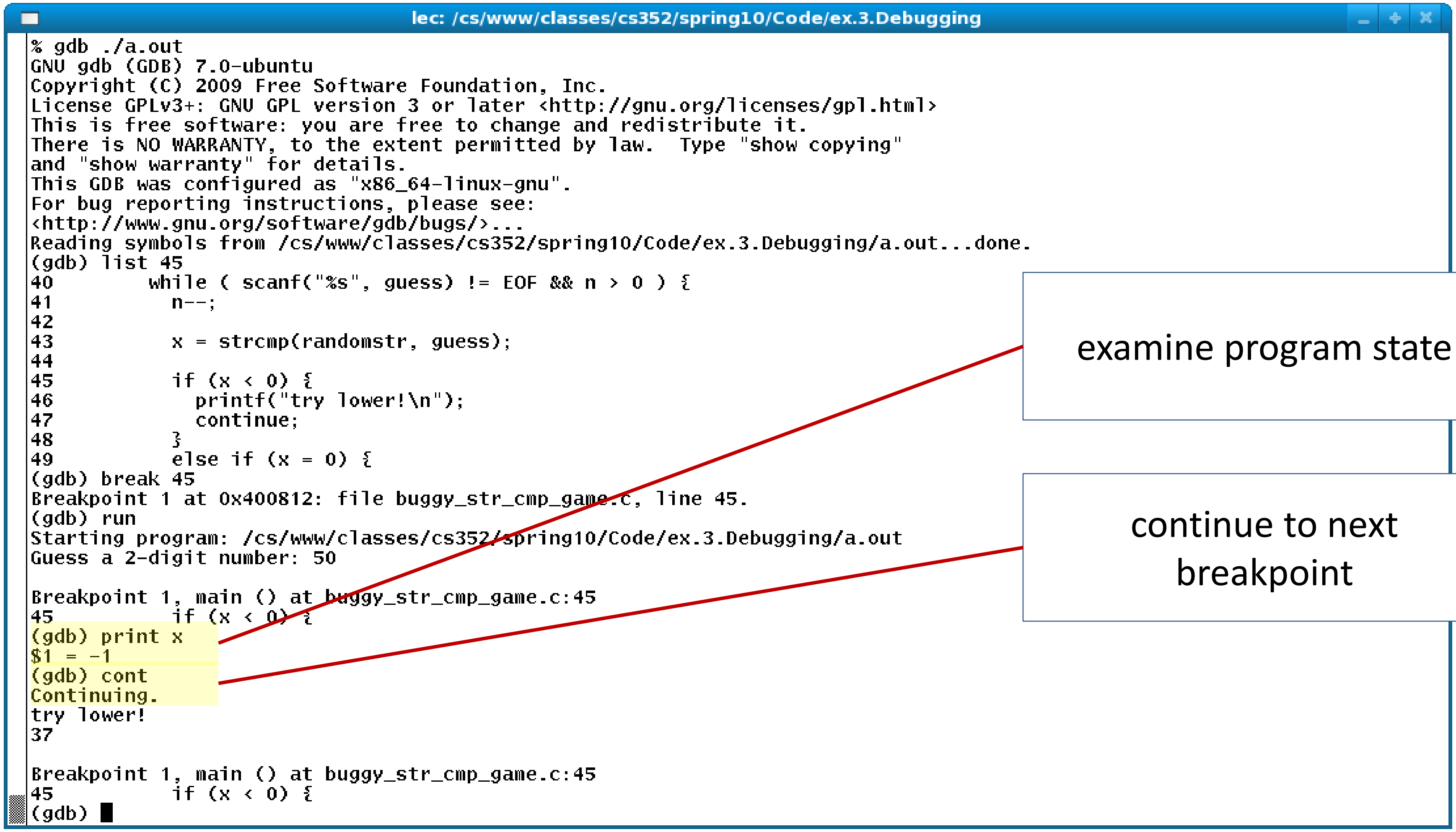
lec: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging

```
% 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
43            x = strcmp(randomstr, guess);
44
45            if (x < 0) {
46              printf("try lower!\n");
47              continue;
48            }
49            else if (x = 0) {
(gdb) break 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: 50

Breakpoint 1, main () at buggy_str_cmp_game.c:45
45            if (x < 0) {
(gdb) step
46              printf("try lower!\n");
(gdb) next
try lower!
47              continue;
(gdb) next
40          while ( scanf("%s", guess) != EOF && n > 0 ) {
(gdb) █
```

execution reaches breakpoint and returns control to user

single-step through the execution

gdb



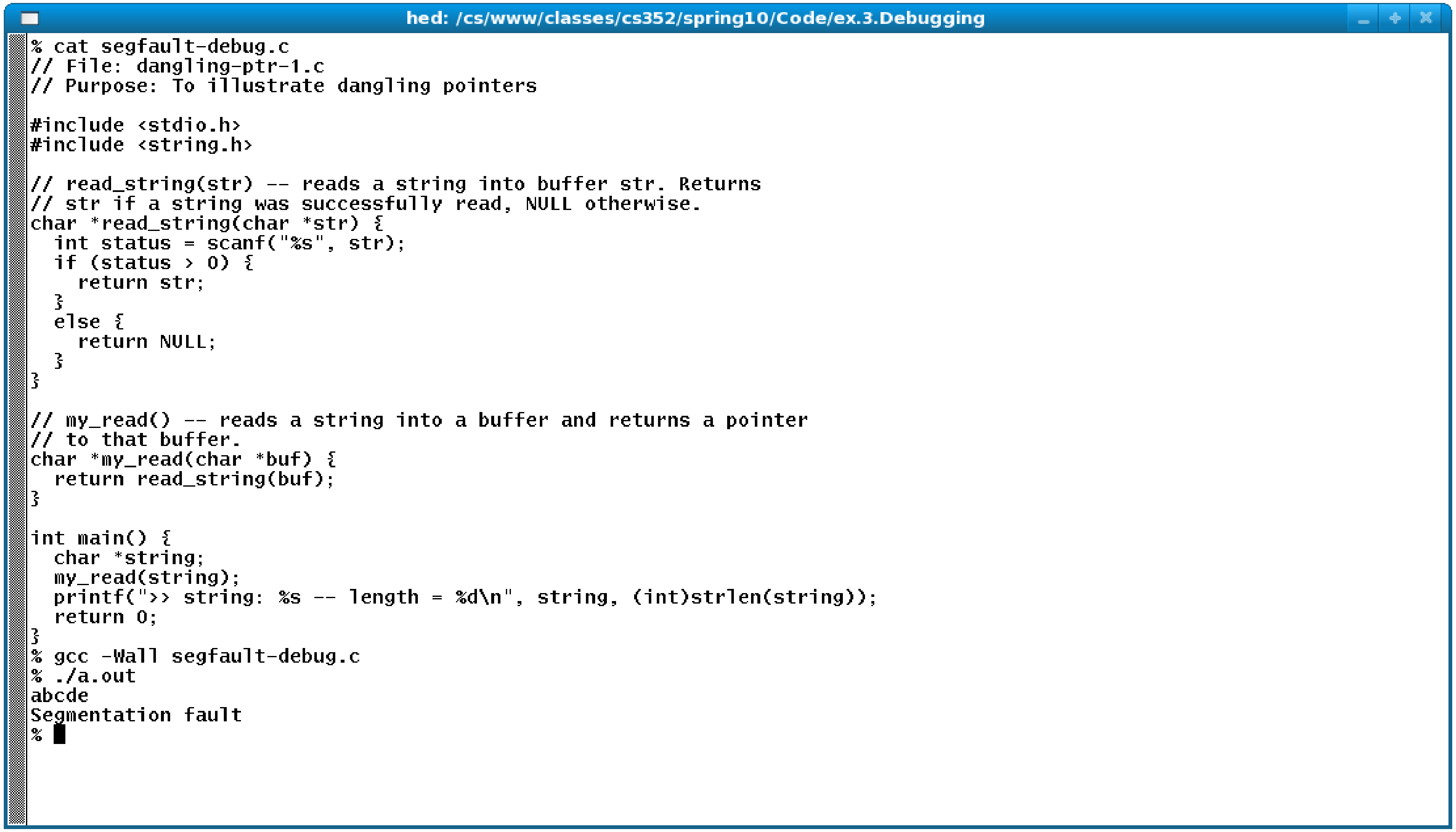
lec: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging

```
% 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
43            x = strcmp(randomstr, guess);
44
45            if (x < 0) {
46              printf("try lower!\n");
47              continue;
48            }
49            else if (x = 0) {
(gdb) break 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: 50

Breakpoint 1, main () at buggy_str_cmp_game.c:45
45            if (x < 0) {
(gdb) print x
$1 = -1
(gdb) cont
Continuing.
try lower!
37

Breakpoint 1, main () at buggy_str_cmp_game.c:45
45            if (x < 0) {
(gdb) █
```

examine program state

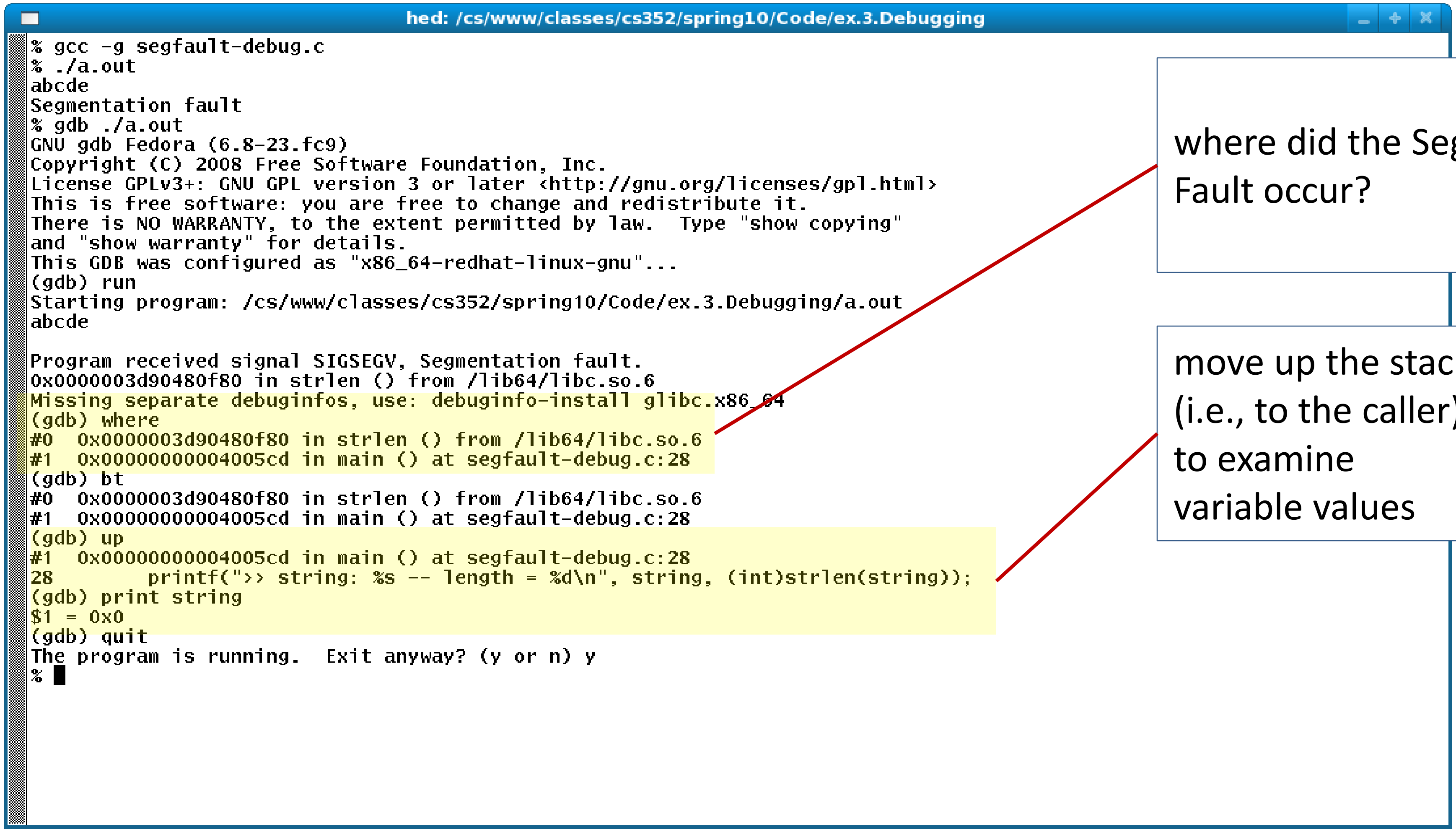
continue to next breakpoint

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(">> string: %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



hed: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging

```
% gcc -g segfault-debug.c
% ./a.out
abcde
Segmentation fault
% gdb ./a.out
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"...
(gdb) run
Starting program: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging/a.out
abcde

Program received signal SIGSEGV, Segmentation fault.
0x0000003d90480f80 in strlen () from /lib64/libc.so.6
Missing separate debuginfos, use: debuginfo-install glibc.x86_64
(gdb) where
#0  0x0000003d90480f80 in strlen () from /lib64/libc.so.6
#1  0x00000000004005cd in main () at segfault-debug.c:28
(gdb) bt
#0  0x0000003d90480f80 in strlen () from /lib64/libc.so.6
#1  0x00000000004005cd in main () at segfault-debug.c:28
(gdb) up
#1  0x00000000004005cd in main () at segfault-debug.c:28
28        printf(">> string: %s -- length = %d\n", string, (int)strlen(string));
(gdb) print string
$1 = 0x0
(gdb) quit
The program is running.  Exit anyway? (y or n) y
% 
```

where did the Seg Fault occur?

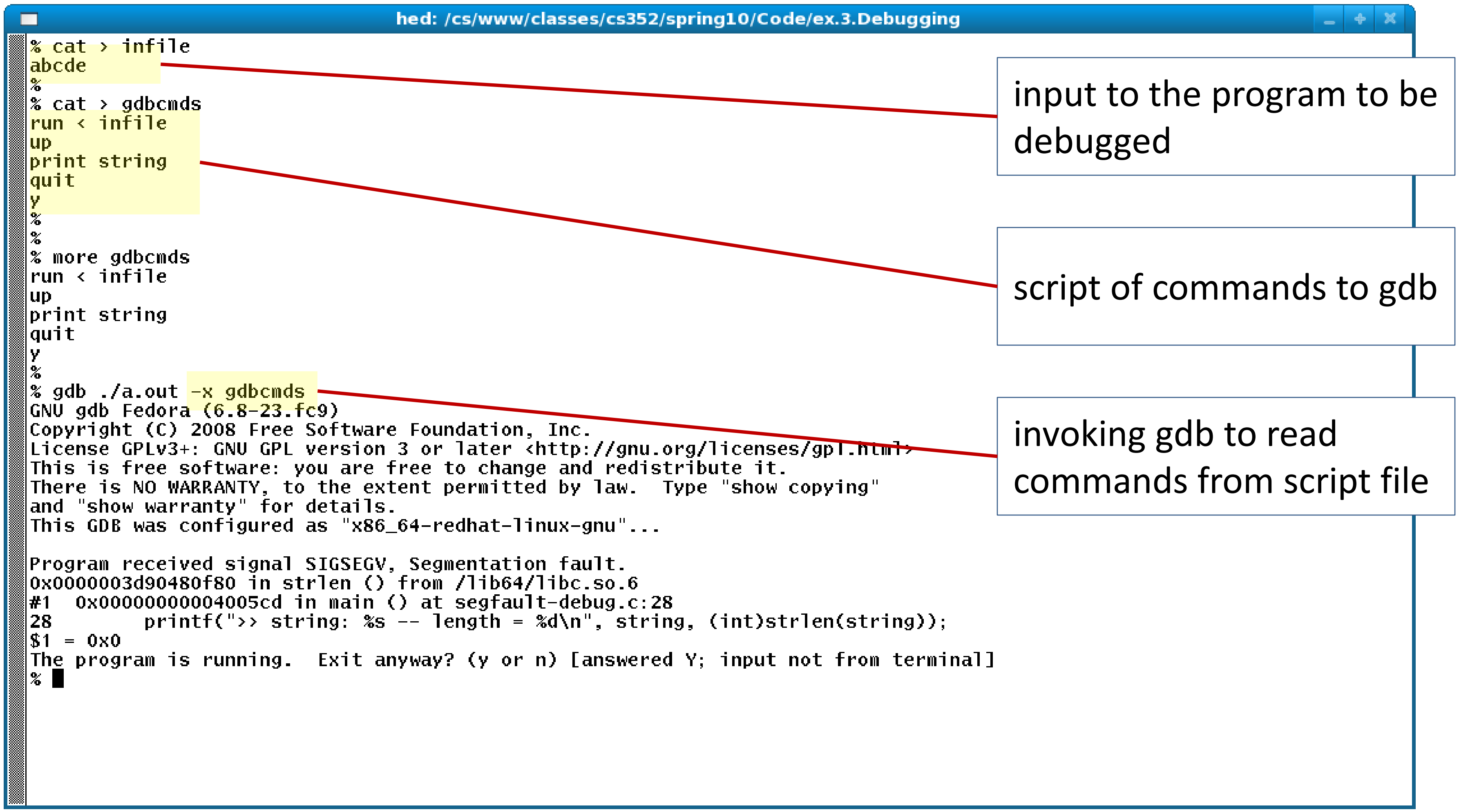
move up the stack (i.e., to the caller) to examine variable values

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



hed: /cs/www/classes/cs352/spring10/Code/ex.3.Debugging

```
% cat > infile
abcde
%
% 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.
0x0000003d90480f80 in strlen () from /lib64/libc.so.6
#1  0x00000000004005cd in main () at segfault-debug.c:28
28          printf(">> string: %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]
% 
```

input to the program to be debugged

script of commands to gdb

invoking gdb to read commands from script file