CSE 303
Lecture 13a

Debugging C programs

reading: *Programming in C* Ch. 13

slides created by Marty Stepp

http://www.cs.washington.edu/303/

gdb

- **gdb** : GNU debugger. Helps you step through C programs.
  - absolutely essential for fixing crashes and bad pointer code
  - your program must have been compiled with the `-g` flag

- usage:

```
$ gdb program
GNU gdb Fedora (6.8-23.fc9)
Copyright (C) 2008 Free Software Foundation, Inc...
(gdb) run parameters
...
```

- redirecting input:

```
$ gdb program
(gdb) run parameters < inputfile
```

gdb commands

command	description
`run` or `r` ***parameters***	run the program
`break` or `b` ***place***	sets a breakpoint at the given place: - a function's name - a line number - a source file : line number
`print` or `p` ***expression***	prints the given value / variable
`step` or `s`	advances by one line of code ("step into")
`next` or `n`	advances by one line of code ("step over")
`finish`	runs until end of function ("step out")
`continue` or `c`	resumes running program
`backtrace` or `bt`	display current function call stack
`quit` or `q`	exits gdb

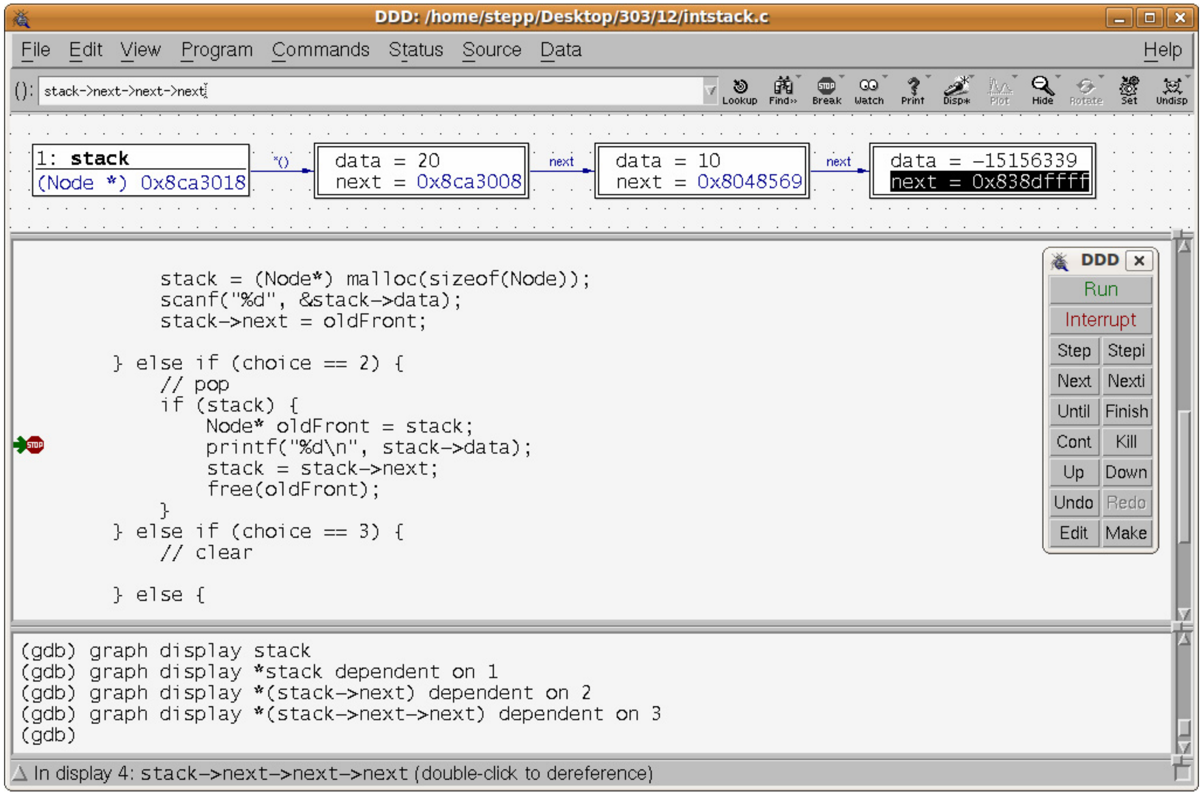
A gdb session

```
$ gdb intstack
GNU gdb 5.2.1
Copyright 2002 Free Software Foundation, Inc.
(gdb) b 34
Breakpoint 1 at 0x4010ea: file intstack.c, line 34.
(gdb) r
Starting program: /home/user/intstack

Breakpoint 1, main () at intstack.c:34
34                      Node* oldFront = stack;
(gdb) p stack
$1 = (Node *) 0x4619c0
(gdb) n
35                      printf("%d\n", stack->data);
(gdb) n
36                      stack = stack->next;
(gdb) n
37                      free(oldFront);
(gdb) p stack
$4 = (Node *) 0x462856
(gdb) p oldFront
$2 = (Node *) 0x4619c0
(gdb) p *oldFront
$3 = {data = 10, next = 0x462856}
(gdb) c
Continuing.
```

ddd

- ddd (Data Display Debugger): Graphical front-end for gdb
  - allows you to view the values of your variables, pointers, etc.

    `$ ddd` ***`programName`***


DDD: /home/stepp/Desktop/303/12/intstack.c
File Edit View Program Commands Status Source Data Help
(): stack->next->next->next
Lookup Find» Break Watch Print Disp* Plot Hide Rotate Set Undisp
DDD
Run Interrupt Step Stepi Next Nexti Until Finish Cont Kill Up Down Undo Redo Edit Make

```
1: stack                 data = 20              data = 10              data = -15156339
(Node *) 0x8ca3018       next = 0x8ca3008       next = 0x8048569       next = 0x838dffff
```

```
        stack = (Node*) malloc(sizeof(Node));
        scanf("%d", &stack->data);
        stack->next = oldFront;

    } else if (choice == 2) {
        // pop
        if (stack) {
            Node* oldFront = stack;
            printf("%d\n", stack->data);
            stack = stack->next;
            free(oldFront);
        }
    } else if (choice == 3) {
        // clear

    } else {
```

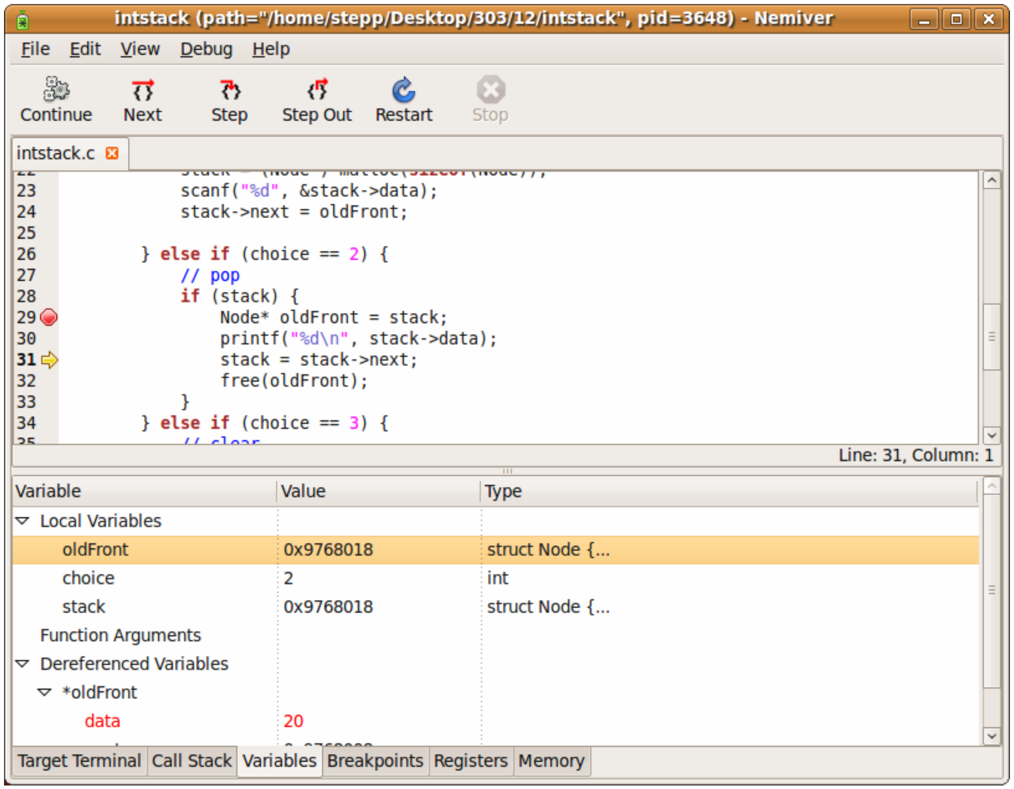
```
(gdb) graph display stack
(gdb) graph display *stack dependent on 1
(gdb) graph display *(stack->next) dependent on 2
(gdb) graph display *(stack->next->next) dependent on 3
(gdb)
```

In display 4: stack->next->next->next (double-click to dereference)

nemiver

- `nemiver` : Another graphical debugger front-end
  - design goal: Be usable even if you don't know gdb commands

```
$ nemiver programName arguments
```


intstack (path="/home/stepp/Desktop/303/12/intstack", pid=3648) - Nemiver
File Edit View Debug Help
Continue Next Step Step Out Restart Stop
intstack.c
Target Terminal Call Stack Variables Breakpoints Registers Memory

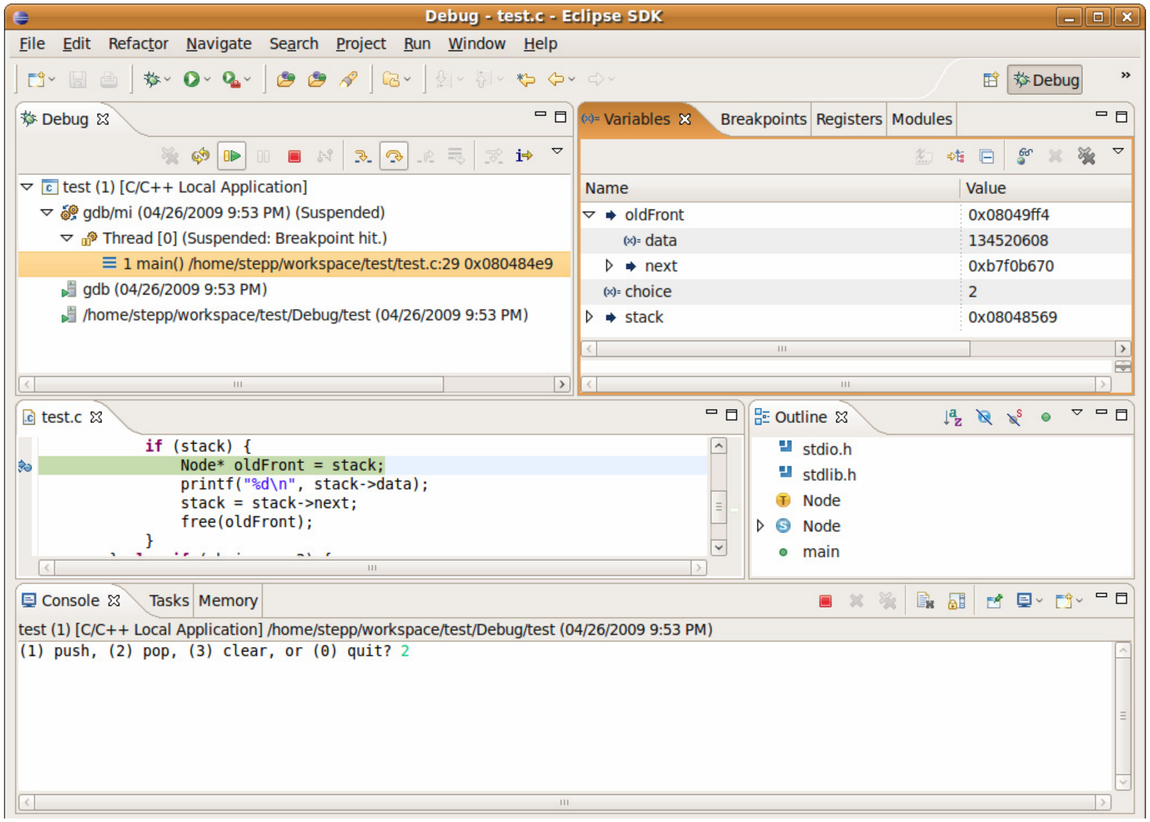
```
            scanf("%d", &stack->data);
            stack->next = oldFront;

        } else if (choice == 2) {
            // pop
            if (stack) {
                Node* oldFront = stack;
                printf("%d\n", stack->data);
                stack = stack->next;
                free(oldFront);
            }
        } else if (choice == 3) {
```

Line: 31, Column: 1

Variable	Value	Type
Local Variables		
oldFront	0x9768018	struct Node {...
choice	2	int
stack	0x9768018	struct Node {...
Function Arguments		
Dereferenced Variables		
*oldFront		
data	20	

Other debuggers

- Eclipse CDT (C/C++ Development Toolkit)
  - create a new **Managed Make C Project**
  - right-click project name, choose **Debug As, Local C/C++ Application**



```
test (1) [C/C++ Local Application]
  gdb/mi (04/26/2009 9:53 PM) (Suspended)
    Thread [0] (Suspended: Breakpoint hit.)
      1 main() /home/stepp/workspace/test/test.c:29 0x080484e9
  gdb (04/26/2009 9:53 PM)
  /home/stepp/workspace/test/Debug/test (04/26/2009 9:53 PM)
```

Name	Value
oldFront	0x08049ff4
data	134520608
next	0xb7f0b670
choice	2
stack	0x08048569

```
        if (stack) {
            Node* oldFront = stack;
            printf("%d\n", stack->data);
            stack = stack->next;
            free(oldFront);
        }
```

```
test (1) [C/C++ Local Application] /home/stepp/workspace/test/Debug/test (04/26/2009 9:53 PM)
(1) push, (2) pop, (3) clear, or (0) quit? 2
```

valgrind

- `valgrind` : A memory-leak detector and debugging tool.

  `valgrind` ***`programName arguments`***

```
(1) push, (2) pop, (3) clear, or (0) quit? 2
==3888== Conditional jump or move depends on uninitialised value(s)
==3888==    at 0x80484E7: main (intstack.c:28)
==3888==
==3888== Use of uninitialised value of size 4
==3888==    at 0x80484F2: main (intstack.c:30)
-15156339
==3888==
==3888== Use of uninitialised value of size 4
==3888==    at 0x8048507: main (intstack.c:31)
==3888==
==3888== Invalid free() / delete / delete[]
==3888==    at 0x4025DFA: free (vg_replace_malloc.c:323)
==3888==    by 0x8048517: main (intstack.c:32)
==3888==  Address 0x8048569 is in the Text segment of /home/stepp/intstack
```

more valgrind

- `valgrind` dumps stats about leaked memory on program exit

```
(1) push, (2) pop, (3) clear, or (0) quit? 1
Number to push? 10
(1) push, (2) pop, (3) clear, or (0) quit? 1
Number to push? 20
(1) push, (2) pop, (3) clear, or (0) quit? 2
20
(1) push, (2) pop, (3) clear, or (0) quit? 2
10
(1) push, (2) pop, (3) clear, or (0) quit? 0

==5162== LEAK SUMMARY:
==5162==    definitely lost: 16 bytes in 2 blocks.
==5162==      possibly lost: 0 bytes in 0 blocks.
==5162==    still reachable: 0 bytes in 0 blocks.
==5162==         suppressed: 0 bytes in 0 blocks.
```

lint / splint

- `lint` (or more recently, `splint`) checks code for possible errors
  - famously picky (sometimes should be ignored)
  - but good for helping you find potential sources of bugs/errors
  - not installed on `attu`, but can install it on your Linux:

    `$ sudo apt-get install splint`

```
$ splint *.c
Splint 3.1.2 --- 07 May 2008

part2.c: (in function main)
part2.c:8:2: Path with no return in function declared to return int
 There is a path through a function declared to return a value on which there
 is no return statement. This means the execution may fall through without
 returning a meaningful result to the caller. (Use -noret to inhibit warning)

use_linkedlist.c:5:5: Function main defined more than once
 A function or variable is redefined. One of the declarations should use
 extern. (Use -redef to inhibit warning)

part2.c:8:1: Previous definition of main
```