

# Debugging Programs with **gdb**

CS 420

**gdb** is a utility for debugging and executing programs. In order to be able to debug a program written in C or C++, it has to be compiled with the **-g** option:

```
gcc -g -o filename filename.c...
```

```
g++ -g -o filename filename.cc...
```

To start off **gdb**, type `gdb filename` at the command line. A few messages are printed, and then you are left at the **(gdb)** prompt:

```
% gdb filename
<various messages>
(gdb)
```

Some of the important and most often used commands at the **gdb** prompt follow. Many of these commands can be abbreviated. See the man page for details.

- **list** *sourceline*

```
list sourcefile:sourceline
```

```
list function
```

```
list
```

Print 10 lines centered at *sourceline* or starting from the beginning of *function*. By itself, print 10 more lines.

- **break** *sourceline*

```
break function
```

Used to set a breakpoint at the *sourceline* or the *function*. In the case of the *sourceline*, execution is stopped before any code on the line is executed. In the case of the *function*, execution stops when the function is entered.

- **run**

Start execution of the program. If breakpoints are set, execution stops when the *sourceline* or *function* is reached. Otherwise, the program runs to completion. **gdb** prints a message stating the status of the program on termination.

- **print** *expression*

Print the value of *expression*. The contents of variables in the program can be viewed through this command.

```
– print i
```

```
Print the value of variable i.
```

```
– print func(args)
```

```
Print the value returned by calling the function func, passed args.
```

- **print** *\*p*  
Print the contents of memory pointed to by *p*, where *p* is a pointer variable.
  - **print** *x.field*  
Check the different members of a structure.
  - **print** *x*  
Check all the members of a structure, assuming *x* is a structure.
  - **print** *y->field*  
*y* is a pointer to a structure.
  - **print** *array[i]*  
Print the *i*'th element of array.
  - **print** *array*  
Print all the elements of array.
- **c**  
Continue execution from where it stopped.
  - **k**  
Kill execution of the program begin run. Typically used to prepare to re-start the program from the beginning.
  - **step**  
**step** [*n*]  
Execute the next or next *n* source line(s). This command steps *into* functions.
  - **next**  
**next** [*n*]  
Same as **step**, but the command steps *past* functions, treating them as if they were single statements.
  - **Removing Breakpoints**
    - **delete**  
Deletes all breakpoints
    - **clear** *sourceline*  
**clear** *function*  
Deletes any breakpoints set on the *sourceline* or at the entry of *function*.
  - **backtrace**  
Print a backtrace of all the active functions on the stack. This is very useful in determining the order in which functions call each other. Frame 0 is the top-of-stack frame. I.e., the currently executing function, called from frame 1.
  - **frame**  
Print a brief description of the currently selected frame.
  - **frame** *n*  
Select frame number *n*.

- **info args**  
Print the arguments of the selected frame.
- **info locals**  
Print the local variables of the selected frame.
- **help**  
Display the set of commands available in **gdb**.
- **quit**  
Exit **gdb**.

The commands in the file **.gdbinit** are executed as **gdb** initializes. **gdb** executes (if present) the file in the home directory. Then, this process is repeated using the current working directory. For more information on **gdb**, run the **info** (see the *man* page) facility from the shell prompt, then use the **m** command to enter the **gdb** documentation. You may also look at the *man* pages and **gdb**'s *help* system.