C: separate files, make, gdb

CSE 413, Autumn 2007
10-31-2007

Topics

- multiple files
- make
- gdb

Compiling Multiple Files

1. gcc –c cpx.c => cpx.o
2. gcc –c test.c => test.o
3. gcc –o test test.o cpx.o => test.exe

or

1. gcc –c cpx.c => cpx.o
2. gcc –o test test.c cpx.o => test.exe

gcc options

- -o filename => name for output file
- -c => don't link, gives .o files
- -g => include extra debugging info in the output
- -Wall => give *warnings* about all possible errors

- gcc –g –Wall *whatever* => a good habit

Compiling with make

```
target: sources
        command
```

Example:

```
foo.o: foo.c foo.h
       gcc -g -Wall -c foo.c
```

- Note: command lines must start with a TAB not spaces!
- Uses:
  - » make –f myMakefile aTarget
  - » make aTarget (assumes Makefile is file name)
  - » make (makes the first target in the file)

Debugging with gdb

- Need to compile with –g option
- To run:
  - » gdb *executable*
- Once in gdb, to run your program:
  - » run *args*

Useful commands in gdb

- backtrace
  - » shows a trace of the stack, including the parameters passed in
- print *expression* (can use &, *, cast)
- info args, or info locals
- break *function* or break *filename:line #*
  - » also: info break, and delete *breaknum*
- step – runs next line of source (goes into subroutine calls)
- next – same as step, doesn't go into subroutines
- finish – run until current function returns

7