CSE306 Software Quality in Practice

Dr. Carl Alphonse
alphonce@buffalo.edu
343 Davis Hall
Sorry!
Sorry!

There were two issues with yesterday's 2:00 - 4:00 lab

1. Environment set-up was incorrect (my fault)

2. Bell 340 machines were cantankerous
Teachable moment

1. Environment set-up was incorrect (my fault)

Lesson: just because something works for you does not mean it works for others
Teachable moment

2. Bell 340 machines were cantankerous

Issue: users not exiting programs and not logging out.

Resolution: TAs can reboot machines if needed. Make sure you shut down programs and log out before leaving.

Do NOT reboot machines on your own. SENS staff are pretty insistent about this. For your protection, in case something goes wrong during reboot - you're in the clear if you didn't reboot.

NEVER power-cycle the machines (TAs can if necessary).
If you were affected:

- If you were in the 2-4 lab, and you were unable to do the lab work, we will arrange for a make-up opportunity.

- One opportunity is tonight @ 6:00 PM in the lab. If this doesn't work for you we'll find alternate arrangements.
Debugging with gdb
Common compiler options

-std set language standard
-o set output file name
-g debugging
-c compile/assemble do not link
-Wall report "all" warnings
-L library path ← search for library files here
-I include path ← where to find .h files
-l library ← search this library during linking
compiling and running

without debugger

- compile using gcc, with `-o` flag if you want to specify a name for the resulting executable (other than "a.out")
  - gcc -o factorial factorial.c main.c
- launch program using by running executable:
  - factorial 5

with debugger

- compile using gcc, with `-g` flag to include debugging information in executable (name of executable is up to you, but adding .debug is a reminder that debugging information is included).
  - gcc -g -o factorial.debug factorial.c main.c
- launch program using gdb
  - gdb factorial.debug

NB: no program argument supplied in gdb invocation
basic commands

- quit - get out of gdb
- help - on-line help system
- run (with program arguments)
Toggle between UI modes

- C-x C-a for TUI/standard mode toggle
- C-x 1 for code only
- C-x 2 for code and assembly

TUI mode commands:

https://sourceware.org/gdb/current/onlinedocs/gdb/TUI-Commands.html#TUI-Commands
TUI mode not always available

- Not all environments support the TUI mode
- All environments support the standard command-line mode: learn these commands
short demo

- bt (backtrace)
- up / down / frame N
- info frame / info args / info locals
- break <function> / break <line> / break <bp> if <expr>
- enable / disable
- ignore <bp> N
- tbreak (a once-only breakpoint)
- run / step / continue / next
Inspecting/changing variables

- `print var (= expr)`
- `set var var = expr`
- `print expr` — evaluate and print, carrying out function calls
- `call expr` — evaluate, do not print

**returning from a function call**

- `return` — discard frame (and subframes)
- `return expr` — as above, `expr` is returned
- `finish` — complete execution of this function normally
- `kill` — terminate execution of the program being debugged
factorial code from lecture

https://cse.buffalo.edu/faculty/alphonce/SP19/CSE306/slides/intro-gdb-factorial.zip

helpful resources

https://www.recurse.com/blog/7-understanding-c-by-learning-assembly

https://sourceware.org/gdb/current/onlinedocs/gdb/