

Offset	Topic
00:17	<ul style="list-style-type: none"> ● Intro <ul style="list-style-type: none"> ● Reminder about Farpoint <ul style="list-style-type: none"> ● http://www.farpointcon.com/ ● No news show this Sunday as a consequence ● Game review, Infernal Contraption <ul style="list-style-type: none"> ● http://www.boardgamegeek.com/game/29456 ● Simple, quick to learn ● Idea is you are a goblin engineer building a contraption ● Three phases per turn <ul style="list-style-type: none"> ● Add to your machine ● Aim at an opponent and turn on ● Draw fresh parts ● Last player with parts wins ● The contraption effects allow you to restore parts to your pile ● Burn parts from your opponents pile ● One expansion out, another on the way ● Part 2 adds sabotage parts, wreck opponents machine, burn up parts ● Rules mostly deal with what parts can go where in the machine ● The five year can play though doesn't get strategy ● Needs help reading cards ● Eight year old gets strategy a little better ● Cartoons on the card add a humorous flavor ● Totally family safe, may help encourage simple math, reading
04:51	<ul style="list-style-type: none"> ● Word of the Week: crock <ul style="list-style-type: none"> ● http://catb.org/jargon/html/C/crock.html
06:13	<ul style="list-style-type: none"> ● Inner Chapter: Debugging <ul style="list-style-type: none"> ● What is a bug? <ul style="list-style-type: none"> ● A programming error in your own code ● An error in someone else's code ● An edge case or counter example not considered ● Bad user input not handled well ● Unusually circumstances, like server outage, not handled well ● Bugs are usually multivariate <ul style="list-style-type: none"> ● The simple, obvious stuff gets tested during development ● Even automated unit testing only exercises what hacker anticipated ● Fully understanding a bug requires asking questions ● Need a way to answer those questions

Offset

Topic

- Debugging is the set of practices used to answer questions about a bug
 - Usually with the goal of fixing
 - Not always, bugs vary in frequency and severity
 - If a bug is not very bad or happens rarely, may not be worth fixing
- Similar to scientific method
 - Observe something unusual
 - Form a hypothesis about how, why it is happening
 - Craft an experiment
 - Send particular inputs into the application
 - Stress it in a specific way
 - Change its configuration
 - Invalidate or confirm hypothesis
 - Iterate if the hypothesis is invalid
 - Make a plan if you have a working theory
 - Important to control the experiment
 - Create a base line, first
 - Only change one variable
 - If you don't have a baseline, no idea whether things are improving
 - If you change more than one variable, no idea what causes what difference
 - Run the experiment repeatedly to smooth out fluctuations
 - Some applications have one time conditions on startup
 - Also smooths out small differences in time if working on performance
- Advanced tools
 - Logging
 - Sometimes called print or printf
 - printf refers to a C function for formatted output
 - Usually involves confirming via output statements that a program hit some code
 - Originally, could only really write to standard out or standard error
 - Even then, idea of verbosity added some control
 - Turn on debug, more verbose, get more output
 - Ability to turn off lets application run faster
 - Logging refers to built in facility to write directly to a log file
 - Some systems, like log4j, are very sophisticated, flexible
 - Many try to optimize performance so it is as close to a non-concern as possible
 - Logging is useful if you would want to output something anyway
 - Drawback is it requires code changes to uncover new things
 - Expensive for watching variables change
 - After a bug is resolved, do you still need that logging?

- Interactive debuggers
 - Usually application is run inside another, custom execution environment
 - Can inspect all data of a running application as it runs
 - Can pause the application, can step line by line through it
 - Usually can step into functions, procedures
 - Break points, places where code pauses, can often have conditions
 - Changed execution environment can alter program behavior
 - Debuggers require a way to map to source
 - Variable, function names usually stripped by compiler
 - Can usually choose to leave debug symbols in
 - This can inflate the size of the executable
 - It often makes de-compiling easier, if secrecy is a concern
- Permanent, semi-permanent instrumentation
 - Some languages, environments exploring permanent instrumentation
 - Java has debugging hooks built in
 - Makes builder debuggers easier
 - Means a debugger can attach to a regular JVM, as it runs
 - Less likely to alter runtime behavior
 - Also means debugger can more easily attach over a network
 - gdb, traditional debugger, can also support remote debugging
 - If debugger is detached, little or no overhead
 - dtrace is similar, allows investigation of running application
 - Doesn't require intrusive instrumentation
 - Can investigate on demand, use code like queries, expressions
- Like any other practice, need to understand all aspects, variations
 - You may not need them for each case
 - Should be able to make appropriate choices
 - If you cannot successfully debug, you cannot maintain software
 - Software spends more of its life in maintenance than any other phase of its life cycle

• Outro

- Contact me
 - Email to feedback@thecommandline.net
 - Web site at <http://thecommandline.net/>
 - IM to [command.line@skype](https://www.skype.com/en/contacts/commandline)
 - Listener comment line is 240-949-2638
 - del.icio.us tag is "for:cmdln"
 - <http://twitter.com/cmdln>

Offset**Topic**

- I'd like to thank libsyn.com for AAC hosting and Wouter de Bie for MP3 hosting
- These notes and the show audio and music are covered by a Creative Commons license
 - <http://creativecommons.org/licenses/by-nc-sa/3.0/us/>
 - Attribution, non-commercial, share alike