BRUCE DAWSON
VALVE

GETTING STARTED DEBUGGING ON LINUX
(MAKING IT EASY IN LESS THAN AN HOUR)

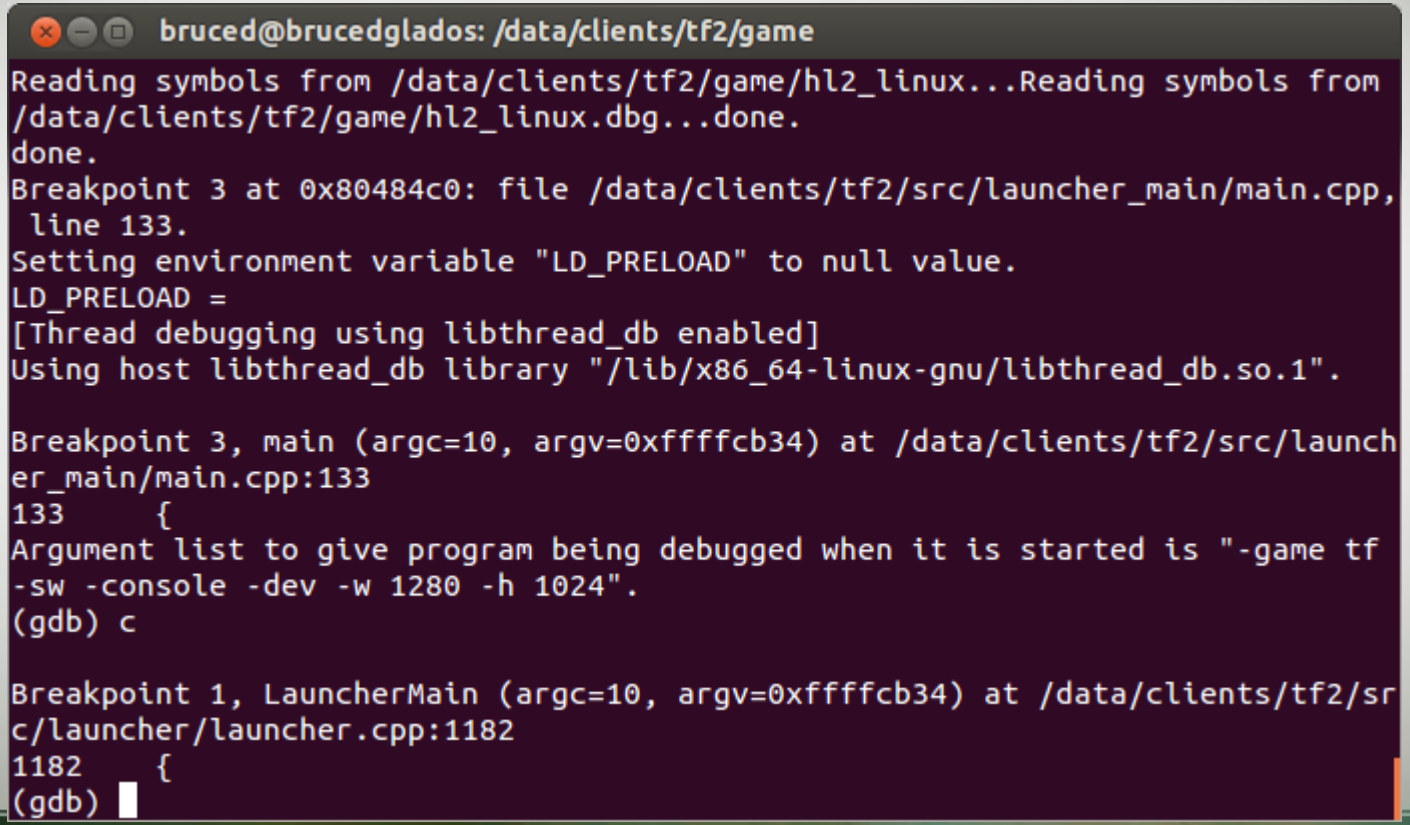
Linux Debugging

- Challenges:
  - Default debugger is intimidating to new users
  - Tough to get symbols and source to show up
  - Many tricks needed for efficient debugging
- You *can* be productive on Linux, quickly

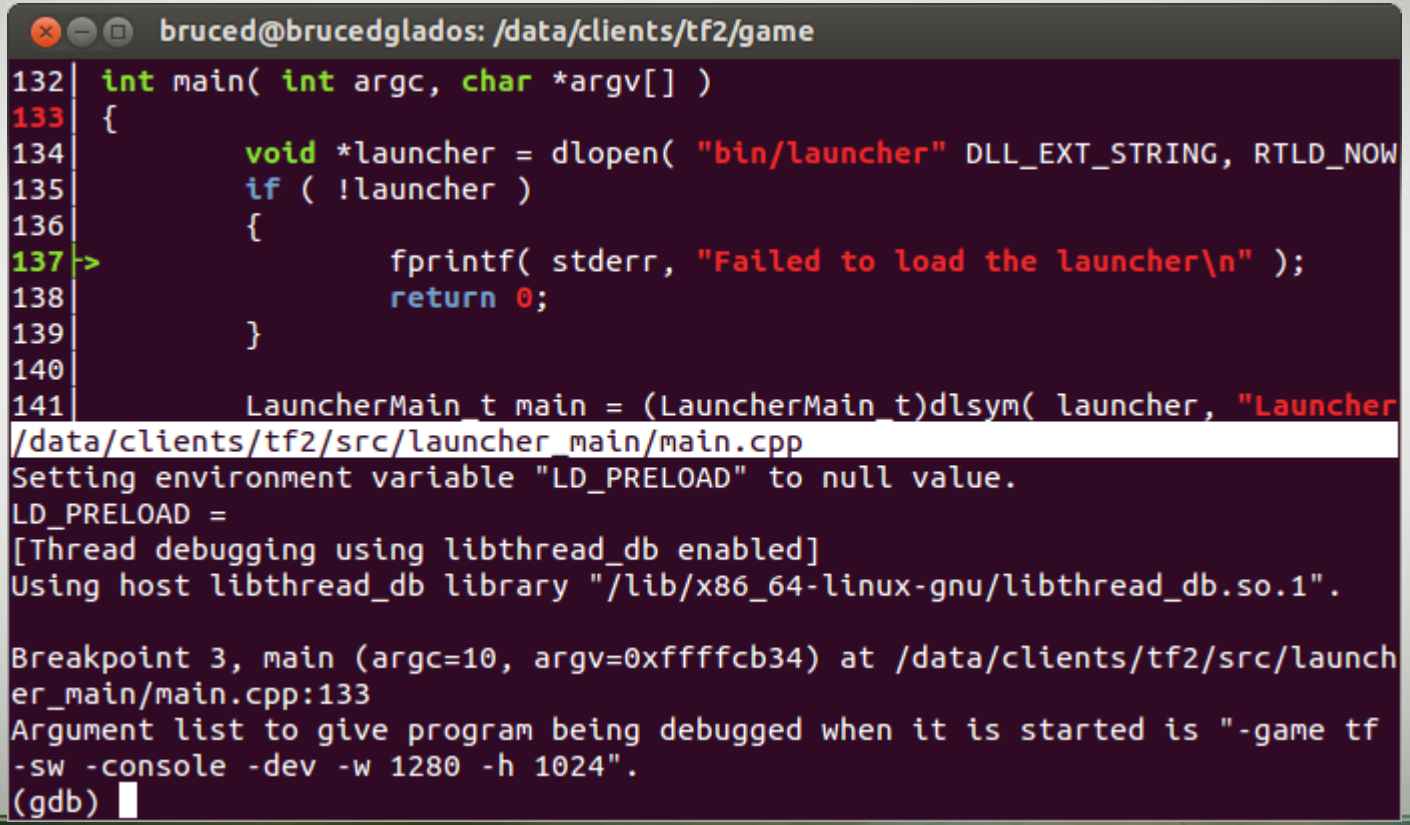
Main Topics

- Choosing a debugger
- Getting symbols to show up
- Getting source code to show up
- Tips and tricks

Choosing a Debugger: gdb

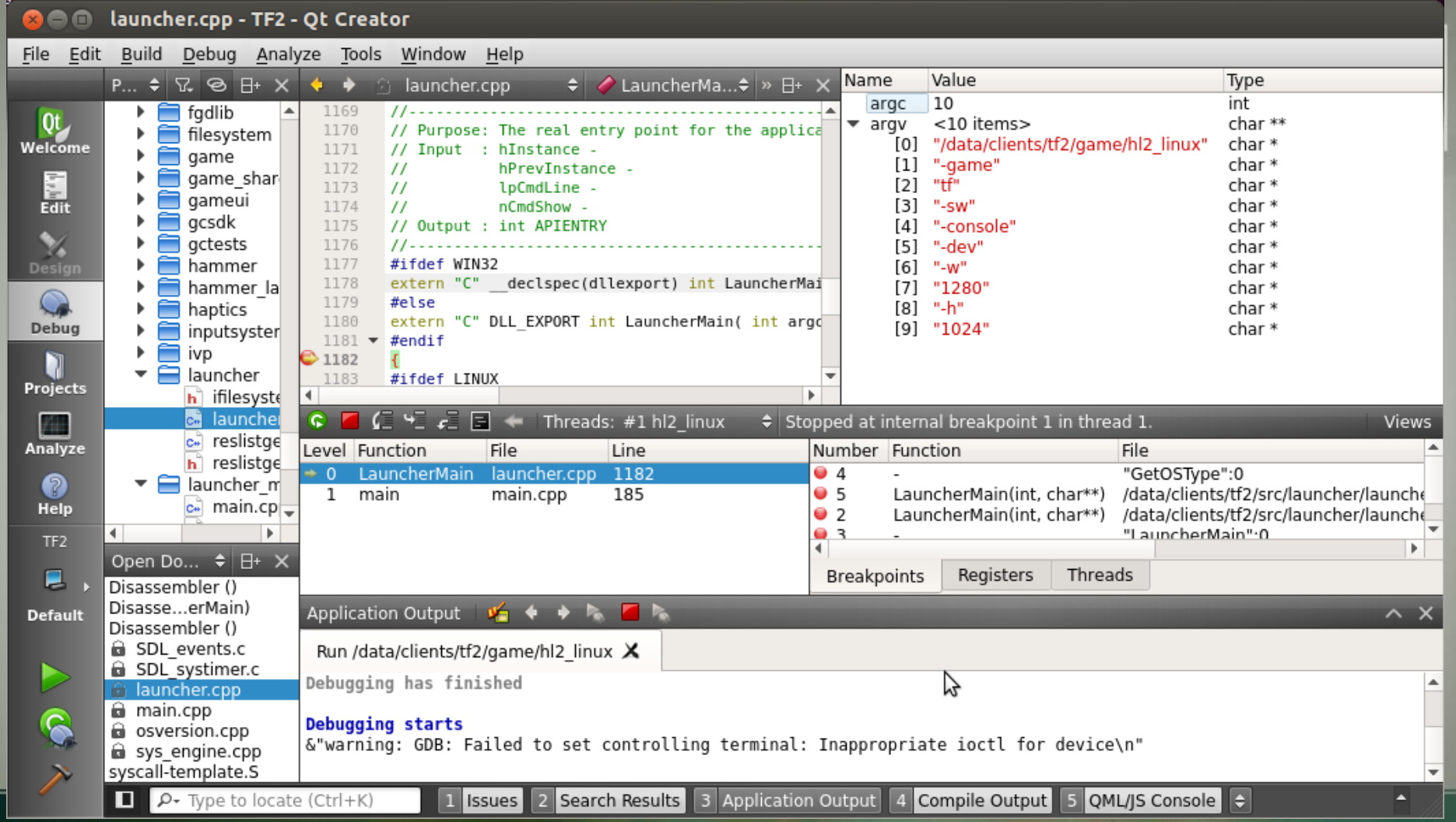


Choosing a Debugger: cgdb



Choosing a debugger: VisualGDB

- Integrates into VisualC++
- Remote build/debug
- Used by some Valve developers
- Commercial product


launcher.cpp - TF2 - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Welcome
Edit
Design
Debug
Projects
Analyze
Help
TF2
Default
fgdlib
filesystem
game
gameui
gcsdk
gctests
hammer
haptics
inputsyster
ivp
launcher
launcher.cpp
main.cpp
//----------
// Purpose: The real entry point for the applica
// Input : hInstance -
// hPrevInstance -
// lpCmdLine -
// nCmdShow -
// Output : int APIENTRY
#ifdef WIN32
extern "C" __declspec(dllexport) int LauncherMai
#else
extern "C" DLL_EXPORT int LauncherMain(int argc
#endif
{
#ifdef LINUX
Name Value Type
argc 10 int
argv <10 items> char **
[0] "/data/clients/tf2/game/hl2_linux" char *
[1] "-game" char *
[2] "tf" char *
[3] "-sw" char *
[4] "-console" char *
[5] "-dev" char *
[6] "-w" char *
[7] "1280" char *
[8] "-h" char *
[9] "1024" char *
Threads: #1 hl2_linux
Stopped at internal breakpoint 1 in thread 1.
Views
Level Function File Line
0 LauncherMain launcher.cpp 1182
1 main main.cpp 185
Number Function File
4 - "GetOSType":0
5 LauncherMain(int, char**) /data/clients/tf2/src/launcher/launche
2 LauncherMain(int, char**) /data/clients/tf2/src/launcher/launche
Breakpoints
Registers
Threads
Open Do...
Disassembler ()
Disasse...erMain)
Disassembler ()
SDL_events.c
SDL_systimer.c
launcher.cpp
main.cpp
osversion.cpp
sys_engine.cpp
syscall-template.S
Application Output
Run /data/clients/tf2/game/hl2_linux
Debugging has finished
Debugging starts
&"warning: GDB: Failed to set controlling terminal: Inappropriate ioctl for device\n"
Type to locate (Ctrl+K)
1 Issues
2 Search Results
3 Application Output
4 Compile Output
5 QML/JS Console

QtCreator Demo

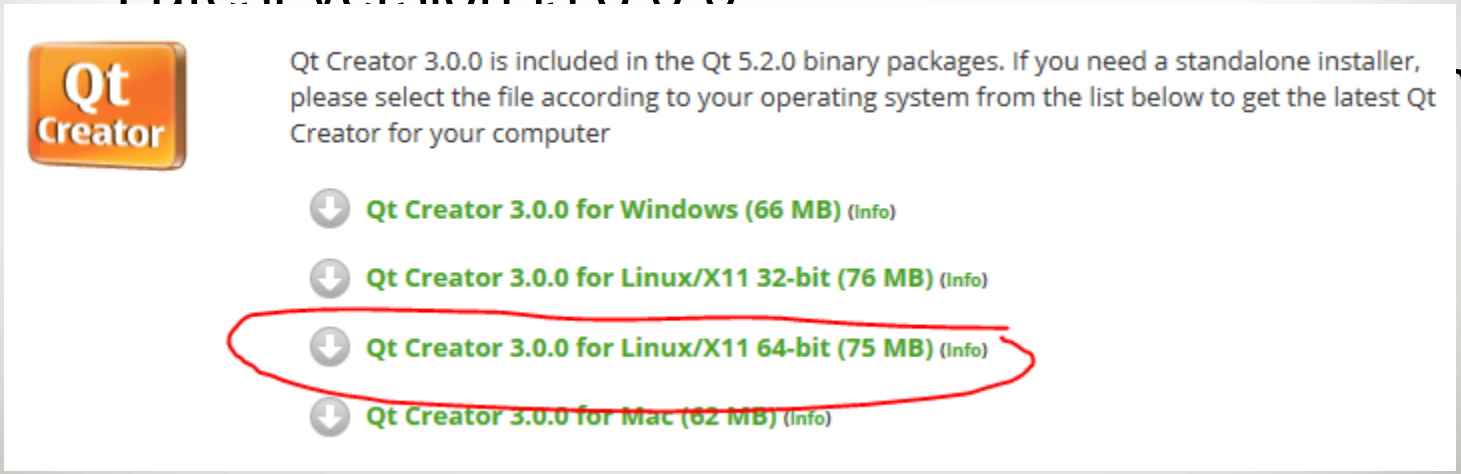
- Creating a project
- Building
- Fixing errors
- Debugging
- Code exploration

Getting QtCreator

- Install from http://qt-project.org/downloads#qt-creator
  - Latest version is 3.0.0
  - Must mark the .run file as executable before running

Getting QtCreator

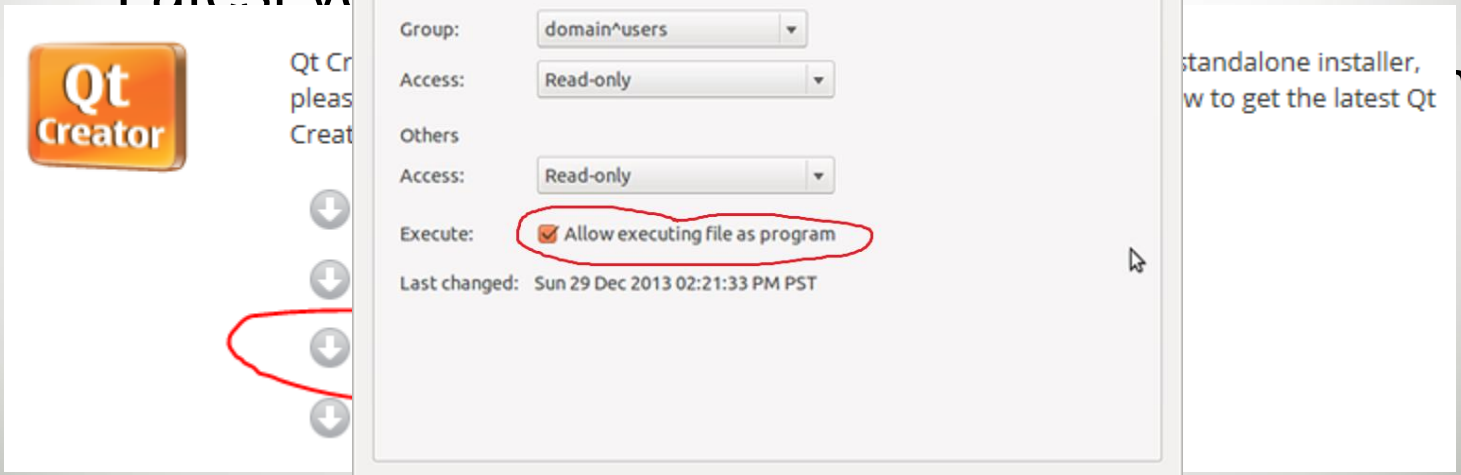
- Install from http://qt-project.org/downloads#qt-creator
  - Latest version is 3.0.0



nning

Getting QtCreator

- Install from ls#qt-creator
  - Latest ve



QtCreator

- Can use for full edit/build/run/debug cycle
- File-> New File or Project-> Import Project-> Import Existing Project
  - Imports all files from the specified directory
  - Will run ‘make’ in that directory, assumes ‘makefile’
  - Can use cmake or run any custom build command you want

QtCreator building

- Summarizes warnings and errors in Issues tab
- Can double-click to jump to location of error/warning

QtCreator Debugging

- VS compatible keyboard shortcuts (F5, F10, F11, etc.)
  - Important exception: Ctrl+F5
- Can load core files, attach to processes, launch processes, etc.
- Other debug windows available from Window-> Views
  - Threads window
  - Registers window
  - Debugger log (for invoking raw gdb commands)

QtCreator: Go-to Anything

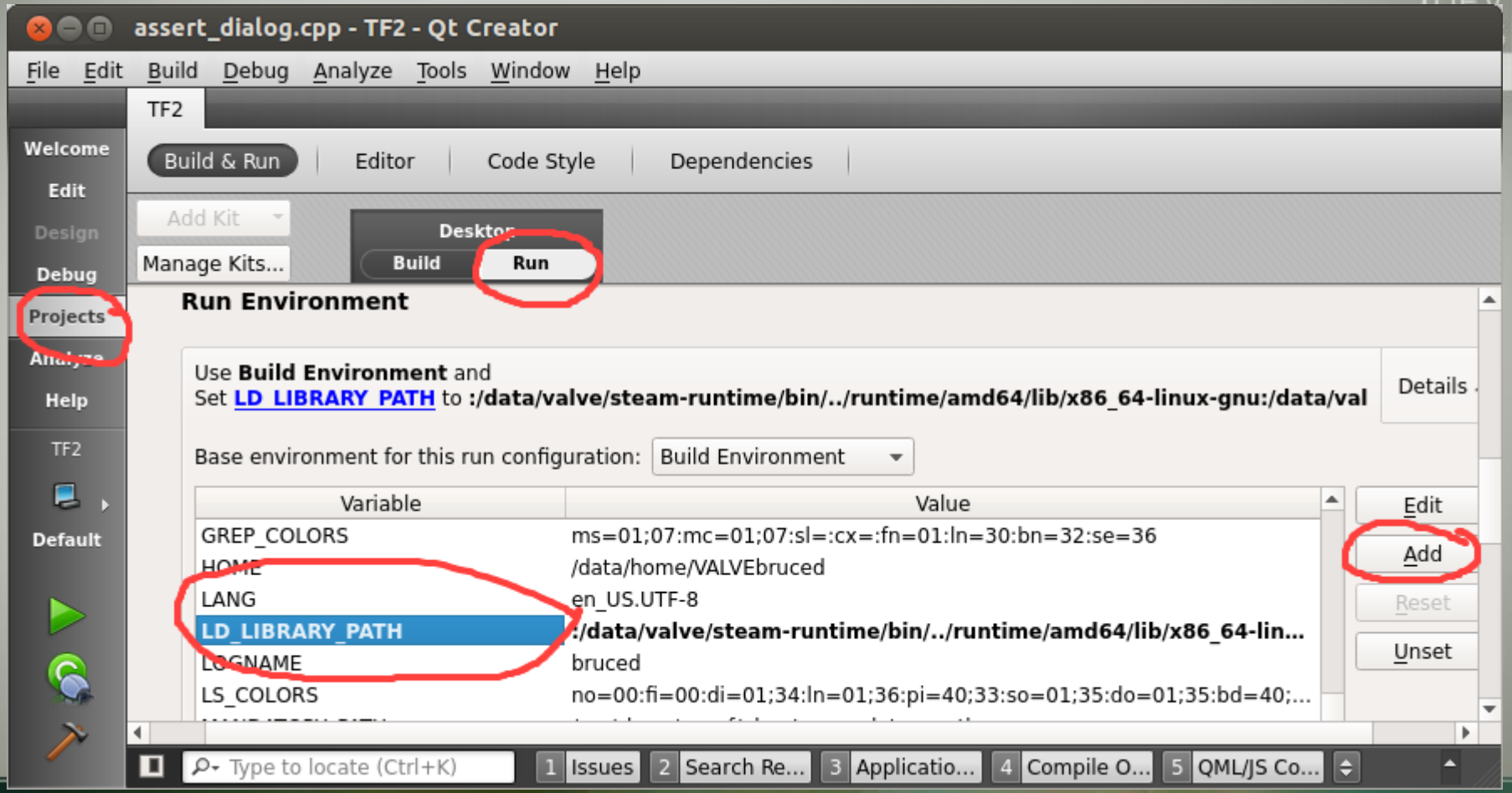
- Ctrl+K is the universal Go-To command
  - Similar to Ctrl+, in VS 2010+
  - <name> goes to source files
  - : <name> goes to C++ classes, enums, and functions
  - l <number> goes to line number
  - Fuzzy matching
- F2 goes to the definition of a symbol
- Alt left/right goes back/forward through navigation history

QtCreator Debug Environment

- LD_LIBRARY_PATH needed for many games
  - Required for Steam runtime
- run.sh sets up runtime environment, let's print it:

  $ *run.sh printenv LD_LIBRARY_PATH*

  /data/valve/steam-runtime/bin/../runtime/amd64/lib/x86_64-linux-gnu:/data/valve/steam-runtime/bin/../runtime/amd64/lib:/data/valve/steam-runtime/bin/../runtime/amd64/usr/lib/x86_64-linux-gnu:/data/valve/steam-runtime/bin/../runtime/amd64/usr/lib:/data/valve/steam-runtime/bin/../runtime/i386/lib/i386-linux-gnu:/data/valve/steam-runtime/bin/../runtime/i386/lib:/data/valve/steam-runtime/bin/../runtime/i386/usr/lib/i386-linux-gnu:/data/valve/steam-runtime/bin/../runtime/i386/usr/lib:/data/clients/tf2/game/bin:/data/valve/steam-runtime/bin/../runtime/amd64/lib/x86_64-linux-gnu:/data/valve/steam-runtime/bin/../runtime/amd64/lib:/data/valve/steam-runtime/bin/../runtime/amd64/usr/lib/x86_64-linux-gnu:/data/valve/steam-runtime/bin/../runtime/amd64/usr/lib:/data/valve/steam-runtime/bin/../runtime/i386/lib/i386-linux-gnu:/data/valve/steam-runtime/bin/../runtime/i386/lib:/data/valve/steam-runtime/bin/../runtime/i386/usr/lib/i386-linux-gnu:/data/valve/steam-runtime/bin/../runtime/i386/usr/lib:/data/valve/steam-runtime/bin/../runtime/amd64/lib/x86_64-linux-gnu:/data/valve/steam-runtime/bin/../runtime/amd64/lib:/data/valve/steam-runtime/bin/../runtime/amd64/usr/lib/x86_64-linux-gnu:/data/valve/steam-runtime/bin/../runtime/amd64/usr/lib:/data/valve/steam-runtime/bin/../runtime/i386/lib/i386-linux-gnu:/data/valve/steam-runtime/bin/../runtime/i386/lib:/data/valve/steam-runtime/bin/../runtime/i386/usr/lib/i386-linux-gnu:/data/valve/steam-runtime/bin/../runtime/i386/usr/lib:
- Copy LD_LIBRARY_PATH setting to QtCreator


assert_dialog.cpp - TF2 - Qt Creator
File Edit Build Debug Analyze Tools Window Help
TF2
Welcome
Edit
Design
Debug
Projects
Help
TF2
Default
Build & Run
Editor
Code Style
Dependencies
Add Kit
Manage Kits...
Desktop
Build
Run
Run Environment
Use Build Environment and
Set LD_LIBRARY_PATH to :/data/valve/steam-runtime/bin/../runtime/amd64/lib/x86_64-linux-gnu:/data/val
Details
Base environment for this run configuration:
Build Environment
Variable
Value
GREP_COLORS
ms=01;07:mc=01;07:sl=:cx=:fn=01:ln=30:bn=32:se=36
HOME
/data/home/VALVEbruced
LANG
en_US.UTF-8
LD_LIBRARY_PATH
:/data/valve/steam-runtime/bin/../runtime/amd64/lib/x86_64-lin...
LOGNAME
bruced
LS_COLORS
no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:do=01;35:bd=40;...
Edit
Add
Reset
Unset
Type to locate (Ctrl+K)
1 Issues
2 Search Re...
3 Applicatio...
4 Compile O...
5 QML/JS Co...

Loading Files from the Terminal

- To open <filename> in an existing QtCreator instance:

  $ *qtcreator.sh -client <filename>*

- Can wrap this in an alias:

  $ *alias qtedit='~/qtcreator-3.0.0/bin/qtcreator.sh -client'*

Symbols!

Symbol Sanity: Others' Code

- Getting symbols for libc6
  - $ *sudo apt-get install libc6-dbg* (or libc6-dbg:i386)
  - Puts symbols in /usr/lib/debug/CopyOfSoPath
- Assume libc6 is in /lib/x86_64-linux-gnu/libc-2.15.so
- Installed symbols go to /usr/lib/debug/**lib/x86_64-linux-gnu/libc-2.15.so**
- gdb and other debuggers automatically look there
  - Will download and use debug version of runtime (symbols and source)
  - Or, download steam-runtime SDK

Symbol Stripping

- Our convention is:
  - <bin>.so is code and minimal symbols
  - <bin>.so.dbg is code and full debug info (everything)
  - <bin>.so is shipped, <bin>.so.dbg is archived
- Archiving a file with full symbols and full code is useful
  - One file provides everything
  - Works with tools that can’t handle stripped symbols

Symbol Stripping

- Copy symbols from <bin>.so to <bin>.so.dbg
  - *$ objcopy <bin>.so <bin>.so.dbg*
  - Optionally add *--only-keep-debug* to strip code
- Add a debug link from <bin>.so to <bin>.so.dbg
  - *$ objcopy --add-gnu-debuglink=<bin>.so.dbg <bin>.so*
- Remove debug information from <bin>.so
  - *$ strip -S <bin>.so*
  - Optionally add -x to strip more information
- See gendbg.sh in the source-sdk-2013 for examples

Symbol Sanity: Your Code

- You can put your symbols in /usr/lib/debug/SoPath
- Or side-by-side with your .so files
- Or leave debuginfo in your .so files
- Better yet, use a symbol server*

* apologies for Microsoft-speak

Symbol Servers (on Linux)

- Just a file server and a convention
- Based on build IDs (40 hex digits)
- Step 1: tell gdb to look for symbols in a second location
  - Assume symbol server directory is **/mnt/syms**
  - *(gdb) set debug-file-directory /usr/lib/debug:**/mnt/syms***
  - Put command in ~/.gdbinit

Adding to Symbol Servers

- Extract the build ID

  $ readelf -n <bin>.so

  …

  Build ID: **6d5f7575de387ed72286** (shortened for slide purposes)

- Copy the .so.dbg file somewhere and make a link to it

  $ cp <bin>.so.dbg (somewhereonserver)

  $ mkdir -p /mnt/syms/.build-id/6d

  *$ ln -s (somewhereonserver) /mnt/syms/.build-id/**6d/5f7575de387ed72286***

  *$ ln -s (somewhereonserver) /mnt/syms/.build-id/**6d/5f7575de387ed72286.debug***

Our Symbol Server

- File paths made from product name, file name, build ID, then file name again
- Archived files contain both binary code and debug info (symbols)
- Two links point to each file

/mnt/syms/.build-id/6d/5f7575de387ed72286

/mnt/syms/.build-id/6d/5f7575de387ed72286.debug

/mnt/syms/tf2/client.so.dbg/**6d5f7575de387ed72286**/client.so.dbg

Symbol Server Uses

- Debuggers automatically retrieve binary and debug info
- You can put libc6 symbols in symbol server
- You can write scripts to retrieve unstripped symbol files
  - Handy for tools that can’t handle stripped symbols or ignore symbol servers

Source Sanity

- Source for locally built binaries will just work
- Build machine binaries need remapping
  - *(gdb) set substitute-path /home/buildbot/tf2/build/src /data/clients/tf2/src*
  - Put in ~/.gdbinit
- Get libc6 source and add to gdb search paths:
  - *$ apt-get source libc6*
  - *(gdb) directory /data/home/bruced/libcsource/eglibc-2.15/stdio-common/*
  - *(gdb) directory /data/home/bruced/libcsource/eglibc-2.15/malloc/*
  - Put in ~/.gdbinit

Tips and Tricks

Linux Library Loading

- ldd
  - prints shared library dependencies
  - Used to diagnose why a module won't load

tf2/game$ *ldd hl2_linux*

```
linux-gate.so.1 =>  (0xf7780000)
libtcmalloc_minimal.so.4 => not found
libdl.so.2 => /lib/i386-linux-gnu/libdl.so.2
libc.so.6 => /lib/i386-linux-gnu/libc.so.6
/lib/ld-linux.so.2
```

  - Often fixed by setting LD_LIBRARY_PATH

Linux Library Loading

- LD_PRELOAD – specify shared objects to load first, can override symbols

  $ LD_PRELOAD="/usr/lib/libtcmalloc.so“ ls

- LD_DEBUG – debug process loading. Example:

  $ LD_DEBUG=all ls 2>out.txt

Better On Linux

- ValGrind – runs process on a virtual CPU, analyzes every memory access. Finds leaks, overruns, and uninitialized variables
- strace – trace system calls. Sample usage:

  $ *strace -p $(pidof procname)* Attach to process

  $ *strace -o out.txt ls* Launch process

Dumpbin Replacements

- nm – list symbols in a shared object
- objdump -d – disassemble an object file

More Tips and Tricks

- Forcing old compilers to add build IDs:

  -Wl,--build-id
- Getting build IDs from a core file (Linux crash dump)

  eu-unstrip -n --core corefile

Ptrace hardening (security)

- Attaching to processes may require root privileges or disabling of ptrace hardening
- ptrace hardening is a security feature to stop debuggers from attaching to running processes
- Either elevate gdb before attaching or disable ptrace hardening:

  $ *sudo -i*

  # echo 0 > /proc/sys/kernel/yama/ptrace_scope

lsof – LiSt Open Files

- List all files opened by a particular process:

  $ lsof -p $(pidof steam)

- List all processes that have a file open

  $ lsof /lib/i386-linux-gnu/libc-2.15.so

References

- Blogging about symbols:
  http://randomascii.wordpress.com/2013/01/19/symbols-on-linux-part-two-symbols-for-other-versions/
- QtCreator:
  http://qt-project.org/downloads#qt-creator
  http://richg42.blogspot.com/2013/10/a-shout-out-to-qtcreator-28x-on.html
  http://richg42.blogspot.com/2013/10/qtcreators-python-debug-visualizers.html
  http://linux-debugger-bits.blogspot.com/2014/01/qtcreator-projects.html
- Steam run-time SDK:
  https://github.com/ValveSoftware/steam-runtime/blob/master/sdk/README.txt
- Symbol ‘servers’ on Linux:
  http://fedoraproject.org/wiki/Releases/FeatureBuildId#Find_files_by_build_ID
- Ptrace hardening:
  https://wiki.ubuntu.com/SecurityTeam/Roadmap/KernelHardening#ptrace_Protection

Questions?

- bruced@valvesoftware.com
- Ask questions now
- Or drop by the Linux break-out session at 5:00 in this room (6C)

Extra slides follow

QtCreator Disassembly Quirks

- Disassembly is shown when *Operate by Instruction*

```
40057c:  mov    edi,0x400747
400581:  mov    eax,0x0
     12 printf("kFooBar == %u\n", kFooBar);
40058b:  mov    esi,0x10
400590:  mov    edi,0x40075d
400595:  mov    eax,0x0
     14 double pi = 3.14159265358979323;
40059f:  movabs rax,0x400921fb54442d18
4005a9:  mov    QWORD PTR [rbp-0x8],rax
     16 InlineDebugTest();
4005b2:  movsd  xmm0,QWORD PTR [rbp-0x8]
```