Freeceiv - Bug #855637

doc/HACKING: Outdated list of --enable-debug compiler options

2020-01-11 08:47 AM - Marko Lindqvist

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
<th>Start date:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assignee:</td>
<td>Marko Lindqvist</td>
<td>% Done: 0%</td>
<td></td>
</tr>
<tr>
<td>Category:</td>
<td>Documentation</td>
<td>Estimated time: 0.00 hour</td>
<td></td>
</tr>
<tr>
<td>Sprint/Milestone:</td>
<td>2.6.2.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description
doc/HACKING lists extra compiler warning options passed on --enable-debug builds. That list is outdated.

History

#1 - 2020-01-20 06:05 PM - Marko Lindqvist
- Sprint/Milestone set to 2.6.2

#2 - 2020-02-01 02:51 PM - Jacob Nevins
Feels like we would be better off telling people where to find the list in the configure code, than maintaining this parallel list. If they are reading this doc, they are HACKERS, after all.

#3 - 2020-02-02 08:56 PM - Marko Lindqvist
Jacob Nevins wrote:

we would be better off telling people where to find the list in the configure code

True.

Master is more complicated, as there this is relevant, at the moment, for autotools build only. We make no adjustments of our own to Meson based builds; they use what ever are meson defaults (and I don't know if meson adjusts its defaults by build type, compiler, etc)

#4 - 2020-02-03 04:19 AM - Chippo Elder
Speaking about that text in that document, where the --enable-debug caveats are mentioned, possibly we should add in another saying that if you try to compile with llvm, then don't go higher than 'some', otherwise you get stuff like:

```c
ldo.c:466:7: error: cast from 'char *' to 'TValue *' (aka 'struct lua_TValue *') increases required alignment from 1 to 8 [-Werror,-Wcast-align]
checkstack(L, 1, func); / ensure space for metamethod /
^-----------
ldo.c:403:9: note: expanded from macro 'checkstack'
\ldo.c:466:7: note: cast from 'char' to 'TValue *' (aka 'struct lua_TValue *') increases required alignment from 1 to 8 [-Werror,-Wcast-align]
^-----------
```

```
ldo.c:586:12: error: cast from 'char *' to 'TValue *' (aka 'struct lua_TValue *') increases required alignment from 1 to 8 [-Werror,-Wcast-align]
oldtop = restorestack(L, ci->extra);
^-----------
```

```
ldo.c:630:16: error: cast from 'char *' to 'TValue *' (aka 'struct lua_TValue *') increases required alignment from 1 to 8 [-Werror,-Wcast-align]
if (luaD_checkstackaux(L))
^-----------
```

```
ldo.h:33:28: note: expanded from macro 'restorstack'
#define restorestack(L,n)       ((TValue *)((char *)L->stack + (n)))
^-----------
```

```
ldo.h:33:28: note: expanded from macro 'luaD_checkstackaux'
{ pre; luaD_growstack(L, n); pos; } else { condmovestack(L,pre,pos); }
^-----------
```

```
ldo.h:33:28: note: expanded from macro 'luaD_checkstackaux'
#define restorestack(L,n)       ((TValue *)((char *)L->stack + (n)))
^-----------
```

```
ldo.h:33:28: note: expanded from macro 'luaD_checkstackaux'
#define restorestack(L,n)       ((TValue *)((char *)L->stack + (n)))
^-----------
```

```
ldo.h:33:28: note: expanded from macro 'luaD_checkstackaux'
#define restorestack(L,n)       ((TValue *)((char *)L->stack + (n)))
^-----------
```

```
ldo.h:33:28: note: expanded from macro 'luaD_checkstackaux'
#define restorestack(L,n)       ((TValue *)((char *)L->stack + (n)))
^-----------
```

```
ldo.h:33:28: note: expanded from macro 'luaD_checkstackaux'
#define restorestack(L,n)       ((TValue *)((char *)L->stack + (n)))
^-----------
```

```
ldo.h:33:28: note: expanded from macro 'luaD_checkstackaux'
#define restorestack(L,n)       ((TValue *)((char *)L->stack + (n)))
^-----------
```
ldo.c:731:20: error: cast from 'char *' to 'TValue *' (aka 'struct lua_TValue *') increases required alignment from 1 to 8 [-Werror,-Wcast-align]
StkId oldtop = restorestack(L, old_top);
^~~~~~~~~
.ldo.h:33:28: note: expanded from macro 'restorestack'
#define restorestack(L,n)       ((TValue *)((char *)L->stack + (n)))
^~~~~~~~~

5 - 2020-02-03 04:26 AM - Marko Lindqvist
For S2_6 & S3_0, install liblua-5.3-dev or whatever the package is called in your distribution to use lua from the system instead of building copy of it as part of freeciv. Building freeciv itself should work.
For master you would need liblua-5.4-dev, but that's not commonly available yet.

6 - 2020-02-03 05:12 AM - Chippo Elder
It's called liblua5.3-dev on my (ubuntu) system. Now clang/++ will compile freeciv at level 'yes' and at level 'checks' will bomb out with:

In file included from /usr/include/gtk-2.0/gtk/gtk.h:234:
/usr/include/gtk-2.0/gtk/gtkitemfactory.h:47:41: error: this function declaration is not a prototype [-Werror,-Wstrict-prototypes]
typedef void (*GtkItemFactoryCallback) ()
^ void

which is already discussed in the caveats. Cool!

Why are my system's header-files (and pulling in that package didn't drag any dependencies with it) more palatable to clang than the ones we ship with freeciv? And why didn't gcc complain about the freeciv headers (at the same be-pedantic level)?

7 - 2020-02-03 03:17 PM - Chippo Elder
I started thinking about this, because to get good BTs from ASAN, one needs the options "-fsanitize=address -fno-omit-frame-pointer -O1 -fno-optimize-sibling-calls -g -fPIE" to end up in the final clang/++ command, and they need to be compatible with the other options that are going to end up there.

If one specifies --enable-debug=yes or checks, a bunch of gadgets are inserted into the code (to display unit-IDs all over the guis, to auto-create BTs on assert fails, etc.). I like these gadgets.

I want to refer to this text.

Debugging has to be activated on compile time by adding the option
'--enable-debug=no/some/yes/checks' to the configure script. The different options have the following meaning:

no: no debugging enabled (FREECIV_NDEBUG and NDEBUG are defined)
additional compiler flags for optimisation (-O3 -fomit-frame-pointer)
some: default build (neither FREECIV_NDEBUG nor FREECIV_DEBUG is defined)
yes: debugging enable (FREECIV_DEBUG is defined)
additional compiler flags:
-Werror -Wmissing-prototypes -Wmissing-declarations
-Wformat -Wformat-security -Wnested-externs
-Wl,--no-add-needed
checks: same as 'yes' but with one additional compiler check:
-Wstrict-prototypes
This is OK for the server and most clients but in the case
of the gtk client there is a problems in its main external library
gtk2 which prevent the compilation of the client using the
extended flags. A fix is available but will not be applied due to
compatibility issues. To compile freeciv with this option,
temporary patch the file gtkitemfactory.h
(see http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=148766).

We're trying to do too many things with a single variable, which I feel are orthogonal:

- whether or not to insert debugging symbols (-g)
- what optimization level to use (-O1 and -O3)
- whether to insert the debugging gadgets
- what level of strictness to warn about
- what level of strictness to err on

I would like to warn about everything, but still have the compile succeed. I can't simultaneously have -Wextra and -Werror with the current code.

The best compromise I've found (which I'm using) is to set --enable-debug=some, specify my ASAN flags manually, and include -Wextra. I only miss out on the debugging gadgets.
In another ticket I learned that at levels 'yes' and 'checks', the current directory is searched when looking for the server binary. I'm going to label this as another of the debugging gadgets.

Can't I get the gadgets by manually defining FREECIV_DEBUG in compiler flags, and that way get all the benefits of --enable-debug=yes, without getting the brutally uncompromising -Werror?

Override free civ -Werror with user CFLAGS/CXXFLAGS
CFLAGS="-Wno-error"

Oh, thank you. Fantastic! Already using this with much happiness and gratitude.

But as a matter of interest, since I'm already compiling with -Wextra (I like seeing the purple and green warnings scrolling past) and hence 'yes' and 'checks' doesn't give me much else, would my original plan have worked? I.e. leave --enable-debug=some (for least interference with the eventual flags) and add in -DFREECIV_DEBUG to get all of:

- will run the server from the current directory
- will honour the -F flag and stop on failed asserts
- will auto-print BTs

- Sprint/Milestone changed from 2.6.2 to 2.6.3

- Status changed from New to Resolved

- Assignee set to Marko Lindqvist

- Sprint/Milestone changed from 2.6.3 to 2.6.2.1

<table>
<thead>
<tr>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006-HACKING-Replace-list-of-enable-debug-set-flags-with-.patch</td>
</tr>
<tr>
<td>0006-HACKING-Replace-list-of-enable-debug-set-flags-with-.patch</td>
</tr>
</tbody>
</table>