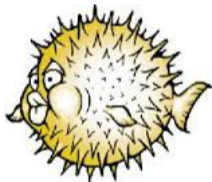


clang vs gcc: waaaaaat?

Marc Espie <espie@openbsd.org>, <espie@lse.epita.fr>



July 17, 2017

- December 17 - first arch using clang, arm64
- April 10 - addition of LIBCXX to bsd.port.mk
- April 17 - clang built by default on amd64/i386
- May 5 - emulated tls in clang
- July 1 - kill depend in Makefiles
- July 13 - introduce COMPILER_LIBCXX

Tools for us

We have dpb and proot! (so building with clang should be easy)

Tools against us

We already have gcc 4.2 in base, gcc 4.9 in ports, and clang in ports.

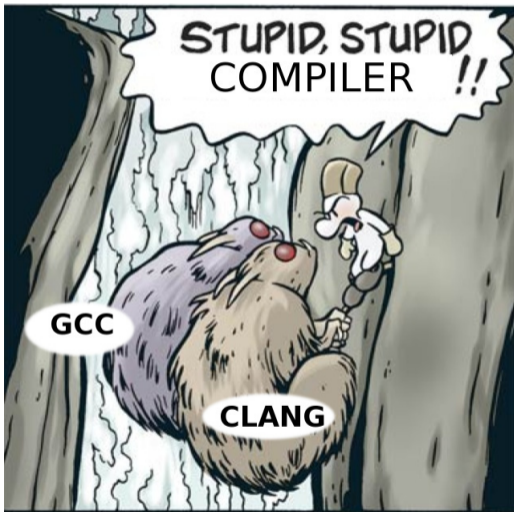
Plus, those are using *two different versions* of `libstdc++`. clang in base is using `libc++`.

And different linkers...









```
cc -O2 -pipe -Wall -g -I. -Werror -c ole.c
ole.c:677:26: error: comparison of unsigned expression < 0 is always false
    if (h->fat_sector_count < 0) insanity++;
        ~~~~~~~~~~~~~~~~~~~~~ ^ ~
ole.c:1070:25: error: comparison of unsigned expression < 0 is always false
    if (current_sector < 0) break;
        ~~~~~~~~~~~~~~~~~~~~~ ^ ~

2 errors generated.
```



```
u_heavy.o: In function 'Unpack_HEAVY':  
u_heavy.c:(.text+0x71): undefined reference to 'decode_c'  
u_heavy.c:(.text+0xbc): undefined reference to 'decode_p'  
u_deep.o: In function 'Unpack_DEEP':  
u_deep.c:(.text+0x643): undefined reference to 'update'  
cc: error: linker command failed with exit code 1 (use -v to see invocation)
```

C compiler doesn't work.

That's because `int main(int argc, char *argv[])`

clang -Whatever does warn, but does not error out.
So you add the option, add -Werror and it explodes!

Low-hanging fruits

- lots of missing headers
- functions that return void/int
- templates are not macros

```
oldfunction()
{
    whatever code
    return; /* note no value, but we don't have void */
}
```

```
oldfunction()
{
    whatever code
    return; /* note no value, but we don't have void */
}

void strangeshit()
{
    /* some code */
    return 0; /* WHY ???? */
}
```

In file included from mg_in.cc:22:

./mg_.h:109:17: error: variable has incomplete type 'C_Comment'

```
    C_Comment    dummy_c_comment;
    ^
```

./mg_.h:28:7: note: forward declaration of 'C_Comment'

```
class C_Comment;
    ^
```

./mg_.h:110:17: error: variable has incomplete type 'Cxx_Comment'

```
    Cxx_Comment  dummy_cxx_comment;
    ^
```

./mg_.h:29:7: note: forward declaration of 'Cxx_Comment'

```
class Cxx_Comment;
    ^
```

```
src/message.cpp:39:2: error: use of undeclared identifier 'time'  
    time(&timestamp);  
    ^
```




In file included from /usr/include/c++/v1/iostream:40:

In file included from /usr/include/c++/v1/istream:163:

In file included from /usr/include/c++/v1/ostream:140:

In file included from /usr/include/c++/v1/locale:220:

/usr/include/c++/v1/___bsd_locale_fallbacks.h:51:12: error: use of undeclared

```
    return wcsnrtombs(__dest, __src, __nwc, __len, __ps);
```

^

/usr/include/c++/v1/___bsd_locale_fallbacks.h:66:12: error: use of undeclared

```
    return mbsnrtowcs(__dest, __src, __nms, __len, __ps);
```

```
//===----- __bsd_locale_fallbacks.h -----  
inline _LIBCPP_ALWAYS_INLINE  
size_t __libc_cpp_wcsnrtombs_l(char * __dest, const wchar_t ** __src, size_t __r  
                                size_t __len, mbstate_t * __ps, locale_t __l)  
{  
    __locale_raii __current( uselocale(__l), uselocale );  
    return wcsnrtombs(__dest, __src, __nwc, __len, __ps);  
}
```

```
//===----- _bsd_locale_fallbacks.h -----  
inline _LIBCPP_ALWAYS_INLINE  
size_t __libc_cpp_wcsnrtombs_l(char * __dest, const wchar_t ** __src, size_t __nwc,  
                               size_t __len, mbstate_t * __ps, locale_t __l)  
{  
    __locale_raii __current( uselocale(__l), uselocale );  
    return wcsnrtombs(__dest, __src, __nwc, __len, __ps);  
}  
  
#if __POSIX_VISIBLE >= 200809  
size_t wcsnrtombs(char * __restrict, const wchar_t ** __restrict, size_t,  
                 size_t, mbstate_t * __restrict)  
__attribute__((__bounded__(__wstring__,1,4)));  
#endif
```

```
qg_dialogfactory.cpp:192:44: error: ordered comparison between pointer and z
    while (layerList->find(layer_name) > 0)
           ~~~~~^~~~~~
```

Pointers everywhere

qg_dialogfactory.cpp:192:44: error: ordered comparison between pointer and zero

```
while (layerList->find(layer_name) > 0)
```

```
~~~~~ ^ ~
```

```
while (fgets(buf, sizeof buf, f) > 0) {
```

```
    ...
```

```
}
```

Define some types

```
typedef struct point {  
    int x;  
    int y;  
};
```

```
for (itSocket = m_Sockets.begin();  
     itSocket != m_Sockets.end(), socketcounter < socketmax ;  
     itSocket++, socketcounter++) {  
}
```


Show Stoppers 1

Some stuff wants Thread-Local-Storage.

Ports gcc has emulated TLS...

... turns out clang can have it too (thanks kettenis@)
and it's compatible!

Pthread functions

- `pthread_key_create`
- `pthread_setspecific`

old style

historically, `bsd.port.mk` was cut into small pieces.

So you do `MODULES = gcc4` to get to gcc. and it gets awful:

```
.include <bsd.port.arch.mk>
.if ${PROPERTIES:Mclang}
WANTLIB += c++ c++abi
.else
MODULES += gcc4
MODGCC_LANGS = c c++
.endif
```

So now we do

```
COMPILER = gcc
```

```
COMPILER_ARCHS = amd64
```

```
WANTLIB += ${COMPILER_LIBCXX}
```



```
struct
```

```
fast_resampler.cpp:40:52: error: declaration of 'S' shadows template parameter  
template<typename T, typename S, template<typename S> class Arithm>
```

```
AutoFilter.cc:72:13: error: call to 'div' is ambiguous
```

```
    div_t qr = div (frames, blocksize);  
                ^~~
```

```
/usr/include/stdlib.h:107:8: note: candidate function
```

```
div_t    div(int, int);  
    ^
```

```
/usr/include/c++/v1/stdlib.h:120:42: note: candidate function
```

```
inline _LIBCPP_INLINE_VISIBILITY ldiv_t div(    long __x,    long __y) _
```

global.cc:374:8: error: cannot initialize a variable of type 'char *' with a

```
char *ext = strchr(filename, '.');  
      ^      ~~~~~
```

1 error generated.

```
./MatrixBase.hpp:371:19: note: must qualify identifier to find this declarat
```

```
    inline void matSliceCheck(size_t sourceRowSize,
```

```
        ^
```

```
In file included from Bancroft.cpp:31:
```

```
In file included from ./Bancroft.hpp:34:
```

```
In file included from ./Matrix.hpp:35:
```

```
./Vector.hpp:117:13: error: use of undeclared identifier 'assignFrom'
```

```
    assignFrom(r);
```

```
    ^
```

```
    this->
```



```
./MatrixBase.hpp:371:19: note: must qualify identifier to find this declarat
    inline void matSliceCheck(size_t sourceRowSize,
        ^
In file included from Bancroft.cpp:31:
In file included from ./Bancroft.hpp:34:
In file included from ./Matrix.hpp:35:
./Vector.hpp:117:13: error: use of undeclared identifier 'assignFrom'
    assignFrom(r);
    ^
    this->
```

thisisnotmyplanetunderstandmonkeyboy

This is called "two phase dependent name lookup"

```
class is_convertible_basic_impl<From, To, false>
{
    typedef char one;
    typedef int two;

    template<typename To1>
    static void test_aux(To1);

    template<typename From1, typename To1>
    static decltype(test_aux<To1>(boost::declval<From1>()), one()) test(in

    template<typename, typename>
    static two test(...);

public:
    static const bool value = sizeof(test<From, To>(0)) == 1;
};
```

```
/usr/bin/ld: getopt_long.o: relocation R_X86_64_PC32 against 'optind'  
can not be used when making a shared object; recompile with -fPIC  
/usr/bin/ld: final link failed: Bad value cc: error: linker command  
failed with exit code 1 (use -v to see invocation)
```

- Led straight to nodepends
- More compiler fun! Why does `gcc -MD -MP a.s` behave differently ?