Rethinking the core OS in 2015

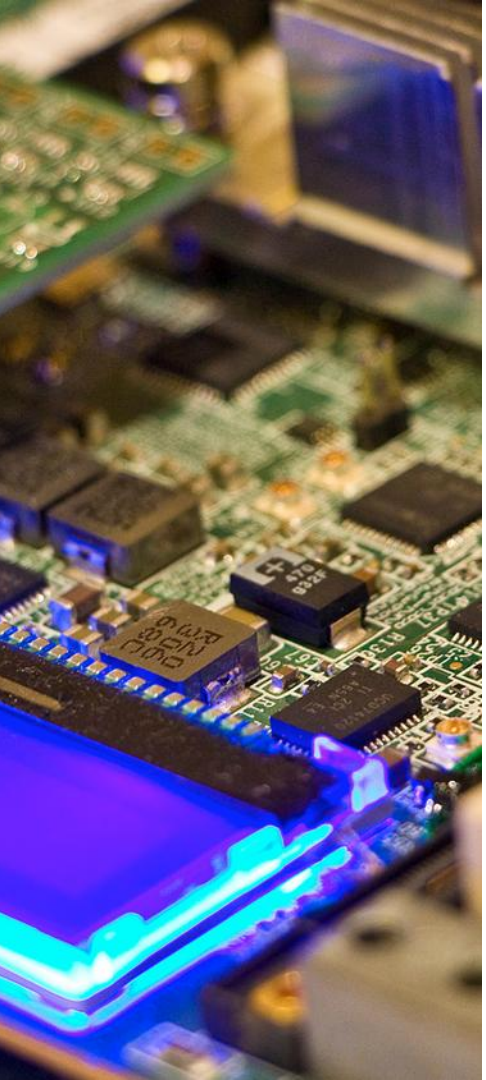
Are alternatives to gcc, libstdc++ and glibc viable yet?

Presented by

Bernhard "Bero" Rosenkränzer

Date

Linux Plumbers Conference 2015

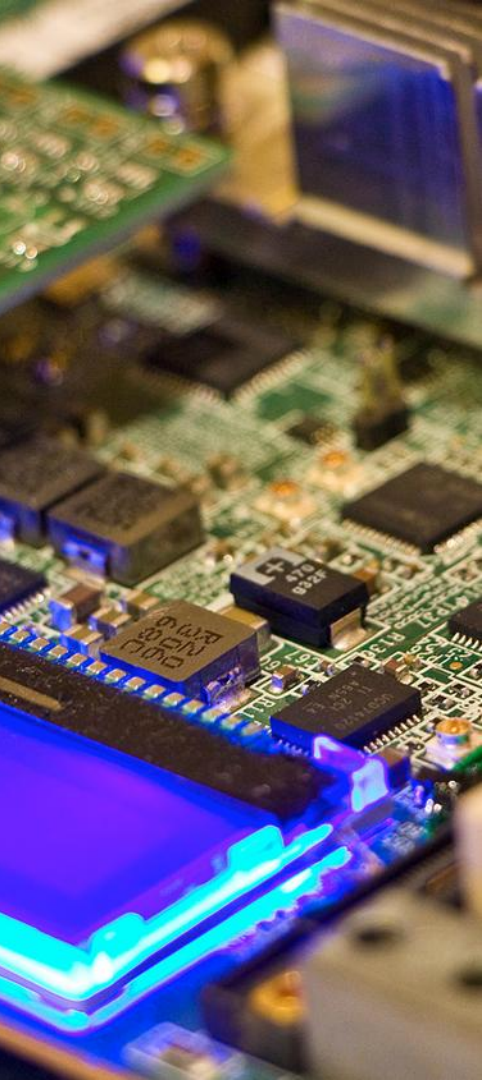


The traditional approach

Building a Linux system traditionally meant starting with building a core consisting of binutils, gcc and glibc (sometimes uClibc).

This is still a very viable approach, but now there are other options...

Linaro

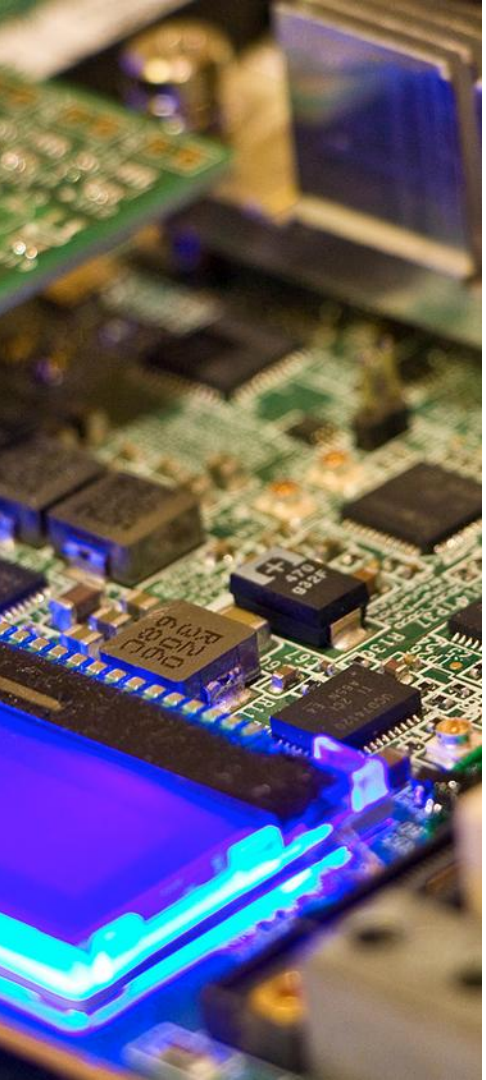


Binutils

Parts of binutils are still needed - in particular, a linker. (The traditional BFD ld can be replaced with gold, also part of binutils).

lld and mclinker are making some progress, but not quite there yet.

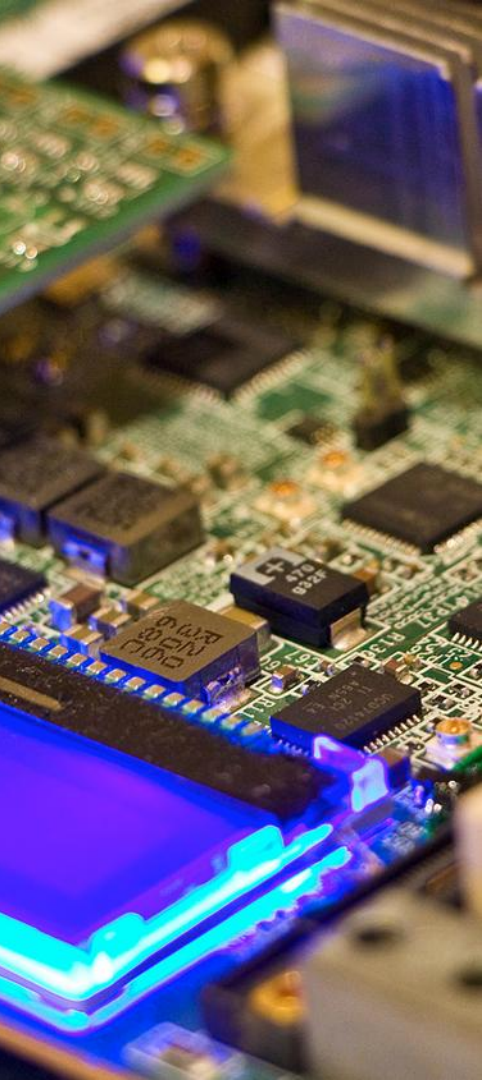
Linaro



Binutils

gas is frequently needed because clang's integrated as doesn't support legacy constructs in common use (e. g. pre-unified syntax on ARM)

Linaro

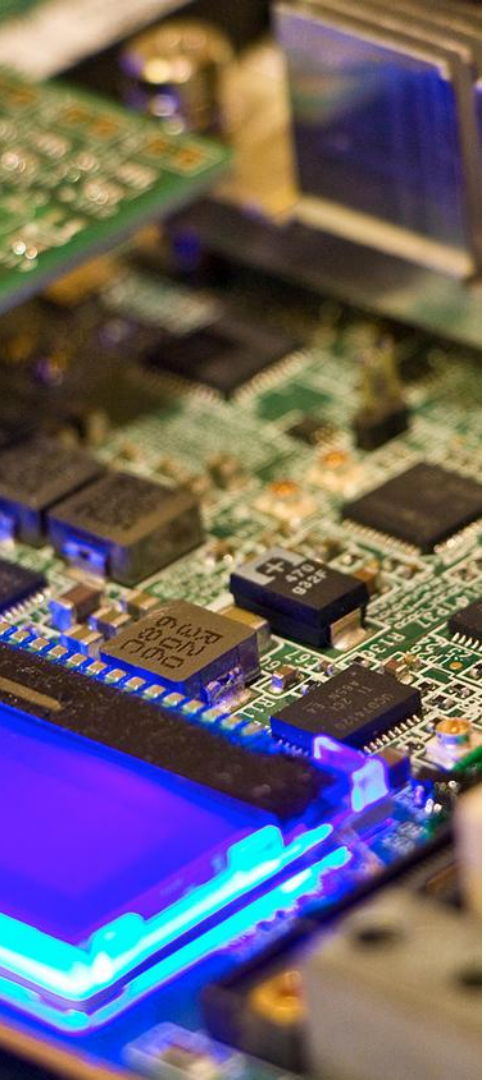


Binutils

Tools like nm need to get more complex: They should now deal with 3 types of input:

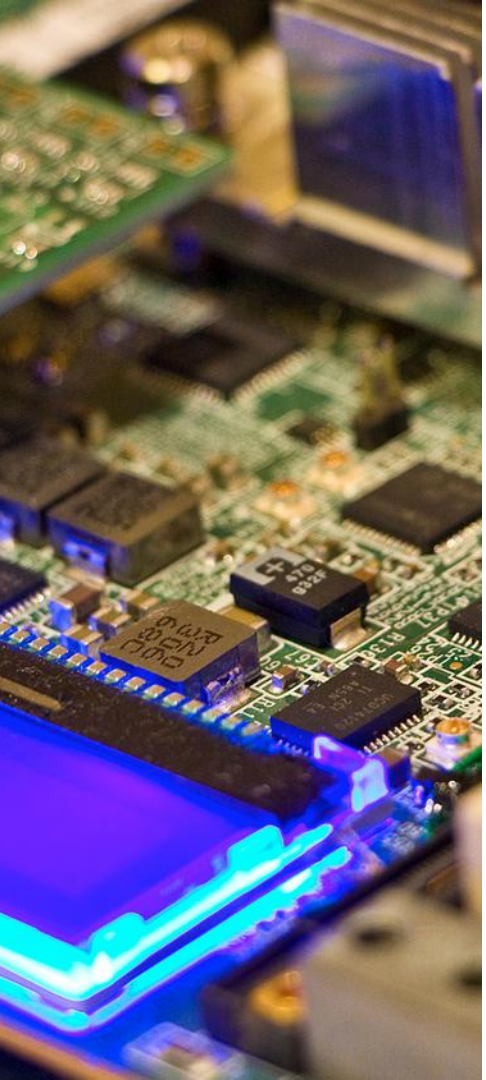
- regular object files
- LLVM bytecode (clang -flto)
- gcc interim code (gcc -flto)

Linaro



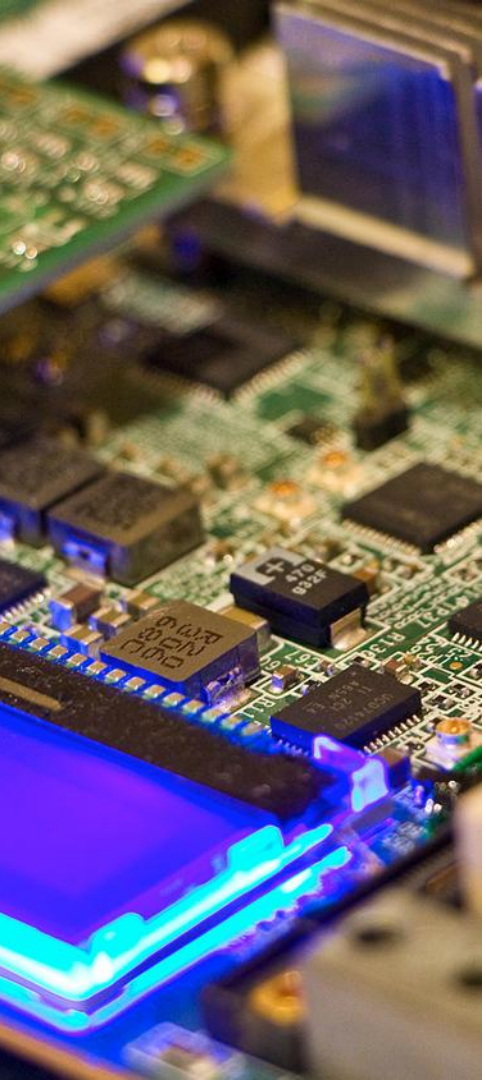
```
#!/bin/sh
REAL_NM=binutils-nm
PARENT="`readlink /proc/$PPID/exe`"
WRAPPED=false
# If /proc isn't mounted, let's do the least evil thing we can
if [ -z "$PARENT" ]; then
    WRAPPED=true
elif echo $PARENT |grep -qE -- '-nm$'; then
    # If we're being called by gcc-nm or llvm-nm, we're already
    # wrapped (and need to make sure we don't call ourselves recursively)
    WRAPPED=true
elif echo $PARENT |grep -qE -- 'qemu'; then
    # Fun... We're running inside qemu binfmt_misc emulation,
    # so we have to determine our parent the evil and less
    # reliable way...
    if grep -qP -- '-nm\x00' /proc/$PPID/cmdline; then
        WRAPPED=true
    fi
fi
```

Linaro



```
# If we're being called by gcc-nm or llvm-nm, we're
# already wrapped...
if ! $WRAPPED; then
    for i in "$@"; do
        [ "`echo $i |cut -b1`" = "-" ] && continue
        if echo $i |grep -qE '\.(o|a)$' && [ -e $i ]; then
            if LANG=C gcc-nm $i 2>&1 |grep -q "File format not recognized";
then
                which llvm-nm &>/dev/null && REAL_NM=llvm-nm
                break
            fi
        fi
    done
    if [ "$REAL_NM" = "binutils-nm" ] && which gcc-nm &>/dev/null; then
        REAL_NM=gcc-nm
    fi
fi
exec $REAL_NM "$@"
```

Linaro

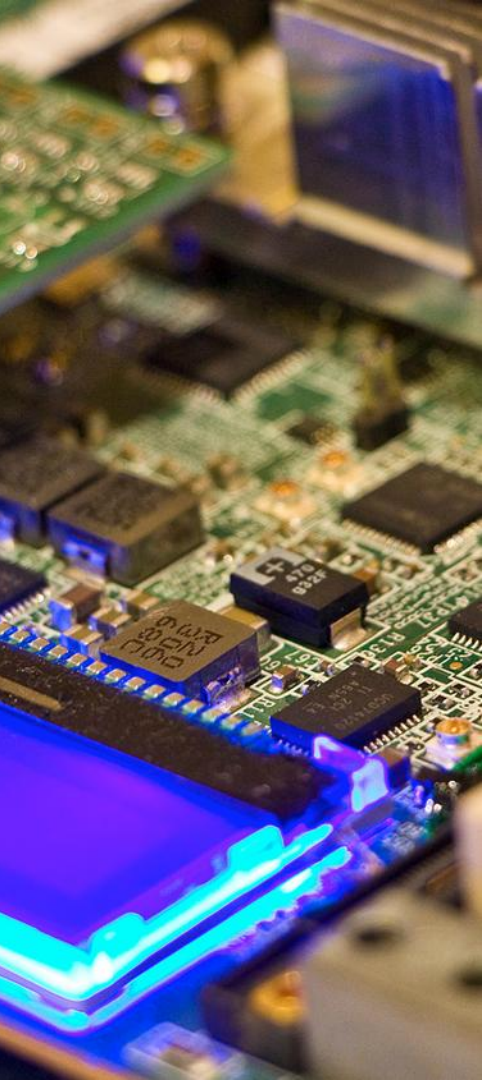


gcc

gcc can, for the most part, be replaced with clang these days.

OpenMandriva switched to clang as its primary compiler last year. OpenMandriva 3 (soon to be released) is almost fully built with clang 3.7.

Linaro

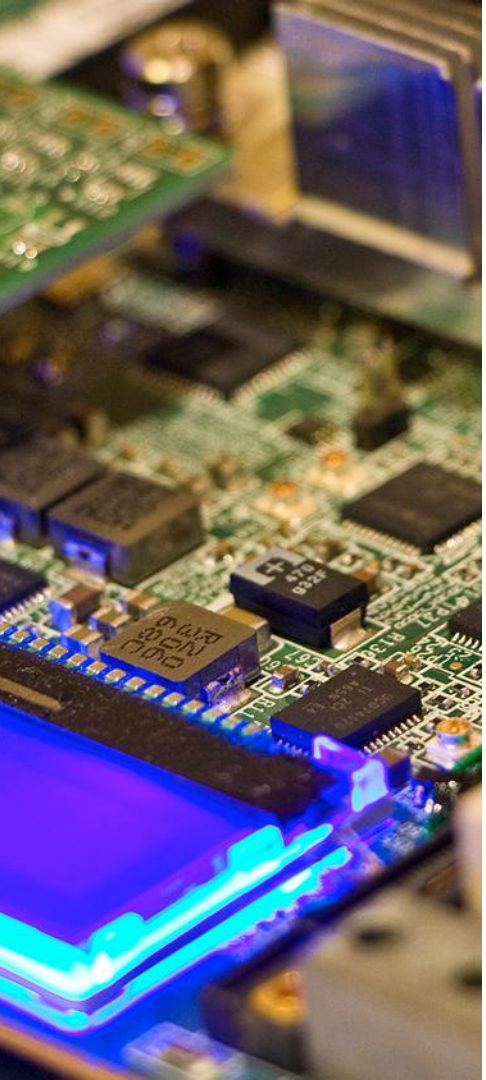


gcc

The transition was unproblematic, most packages that failed failed due to bad code or use of nonstandard gcc extensions.

We force some packages to build with CC=gcc CXX=g++.

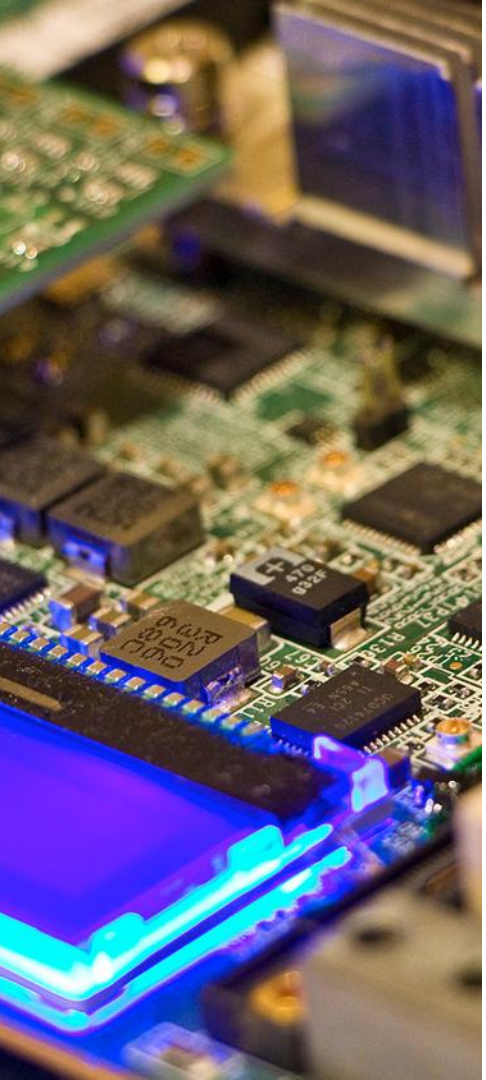
Linaro



gcc

We still need to build gcc even if we don’t want to use it as a compiler though: We need libgcc, libgcc_s, libatomic and friends (and potentially libstdc++)

Linaro

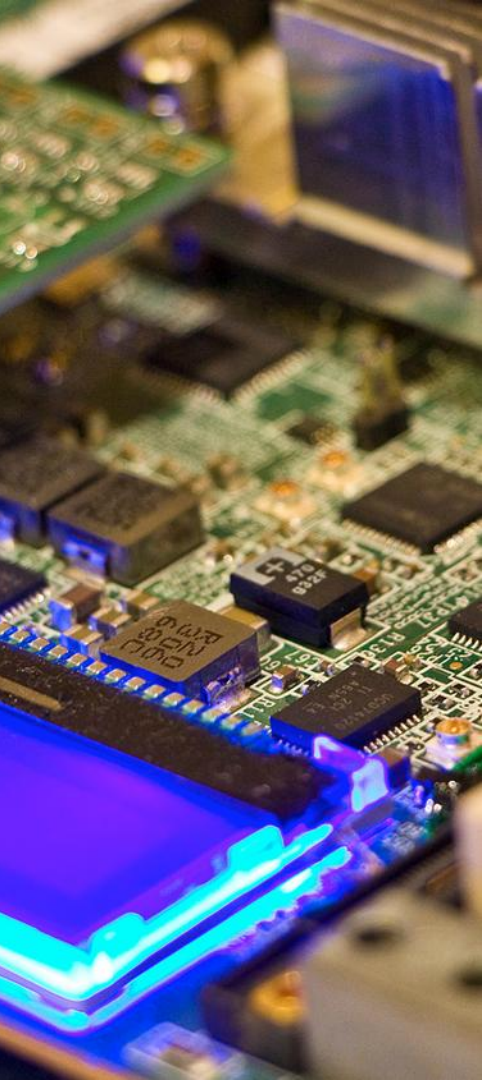


gcc

clang's __GNUC__ macro definitions are too conservative, claiming to be gcc 4.2.1, causing code that checks __GNUC__ to leave out optimizations.

Patching it to say 4.9 produces better code. (real fix is to check for features instead of compiler versions)

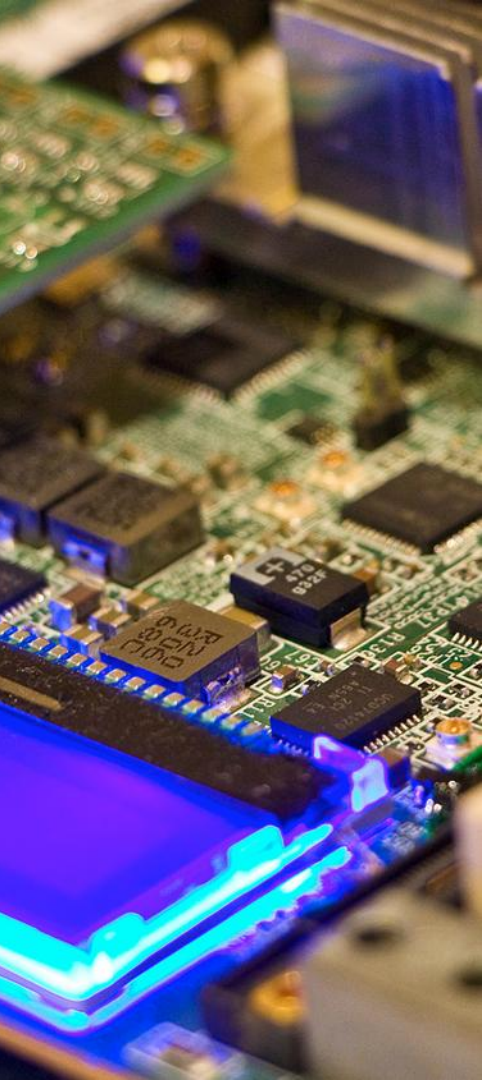
Linaro



Things to avoid for compatibility

- Nested functions
- Variable length arrays in structs
- Variable length arrays of non-POD types
- Empty structs
- Array subscripts of type “char” (`value['0']=0;`)
- Reserved words (“_Nullable” defined by both clang and Qt)

Linaro

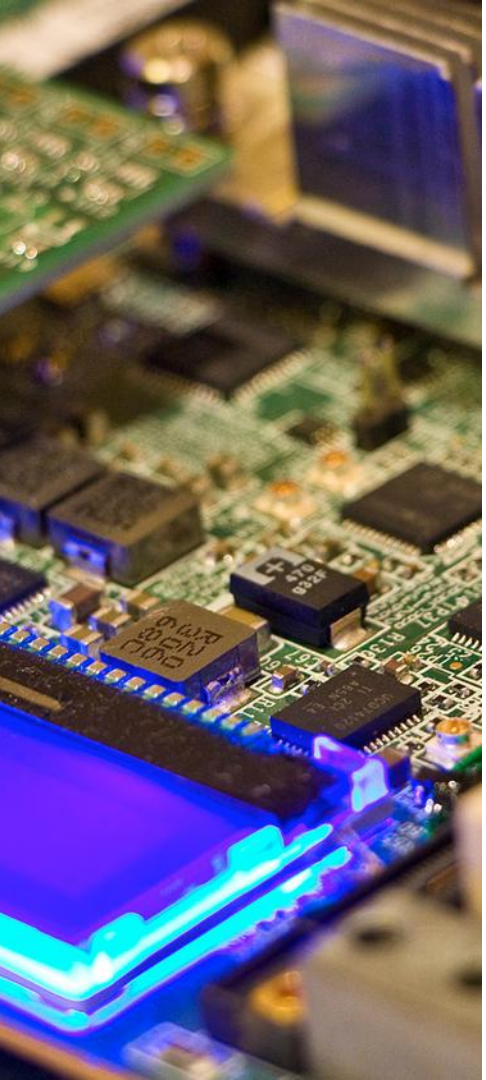


Things to avoid for compatibility

- Undefined internal functions and variables -- even if they aren't used:

```
static void a();
void b() {
    if (0)
        a();
}
```

Linaro



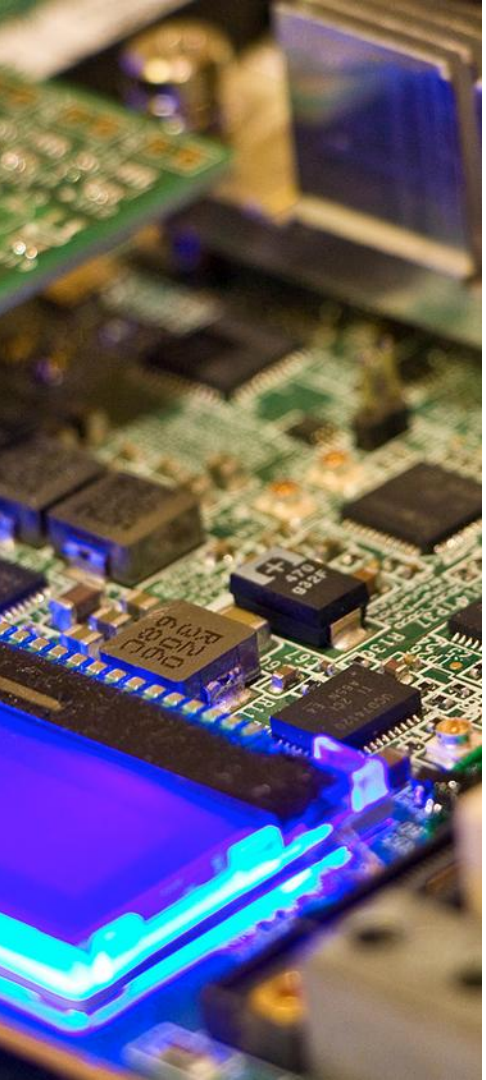
Things to avoid for compatibility

- gcc 5.x's changed libstdc++ ABI

  https://llvm.org/bugs/show_bug.cgi?id=23529

  - clang doesn't implement gcc's __attribute__((abi_tag)), needed by gcc 5.x's libstdc++ built in new ABI mode
  - build gcc with `--with-default-libstdcxx-abi=gcc4-compatible` for now

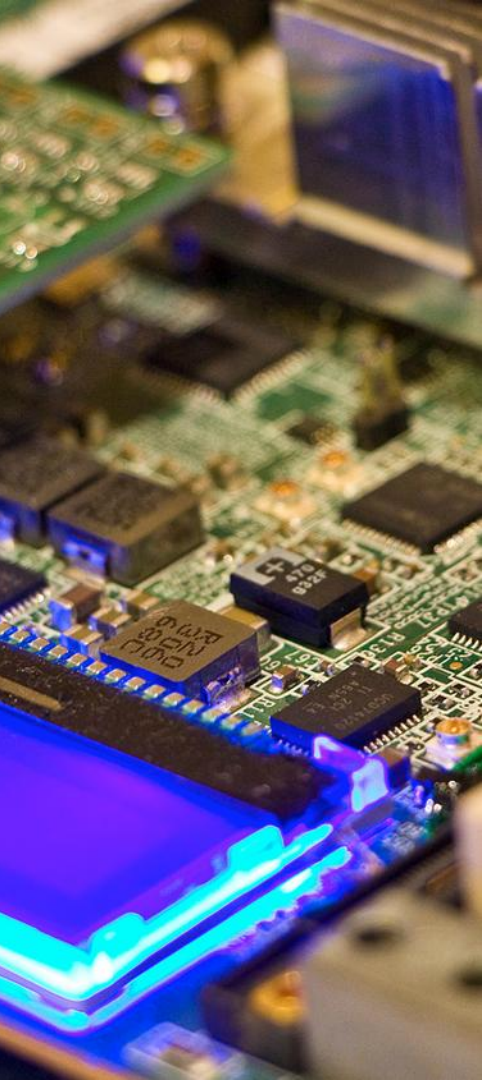
Linaro



Things to avoid for compatibility

- C89-isms and C++98-isms, e.g. changed meaning of “`extern inline`”

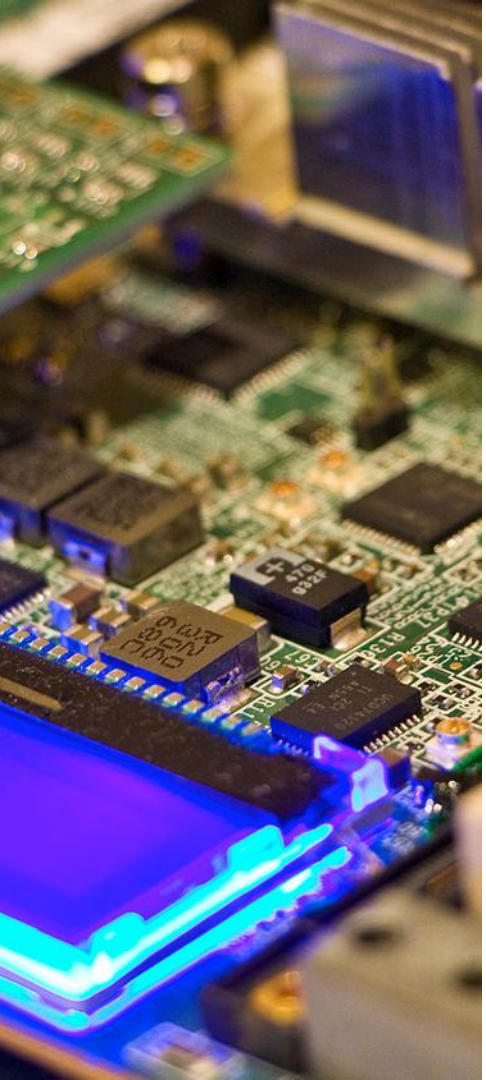
Linaro



Interesting bugs found by clang

```
void something(char n[30]) {
   if(!memcmp(buffer, n, sizeof(n))) {
      …
   }
}
```

Linaro

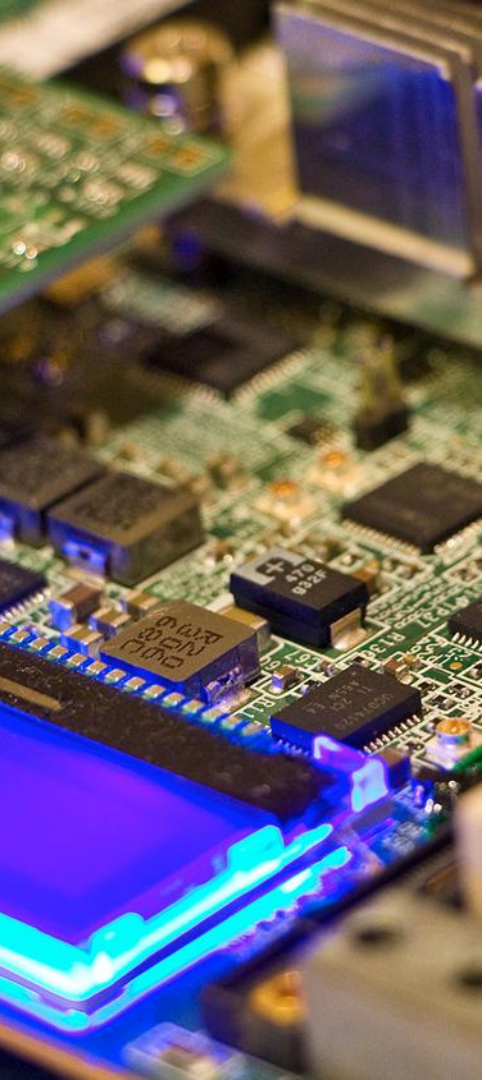


Interesting bugs found by clang

```
void something(char n[30]) {
   if(!memcmp(buffer, n, sizeof(n))) {
      …
   }
}
```

size of a pointer - not quite 30

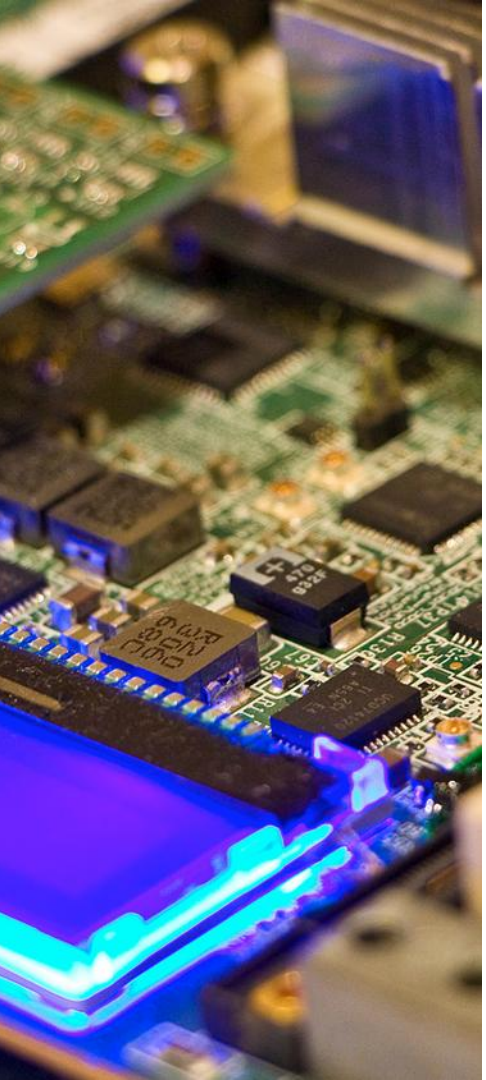
Linaro



Interesting bugs found by clang

```
unsigned char a[X];
for(int i=0; i<X; i++)
   b = a ? tagCpe++ : tagSce++;
```

Linaro

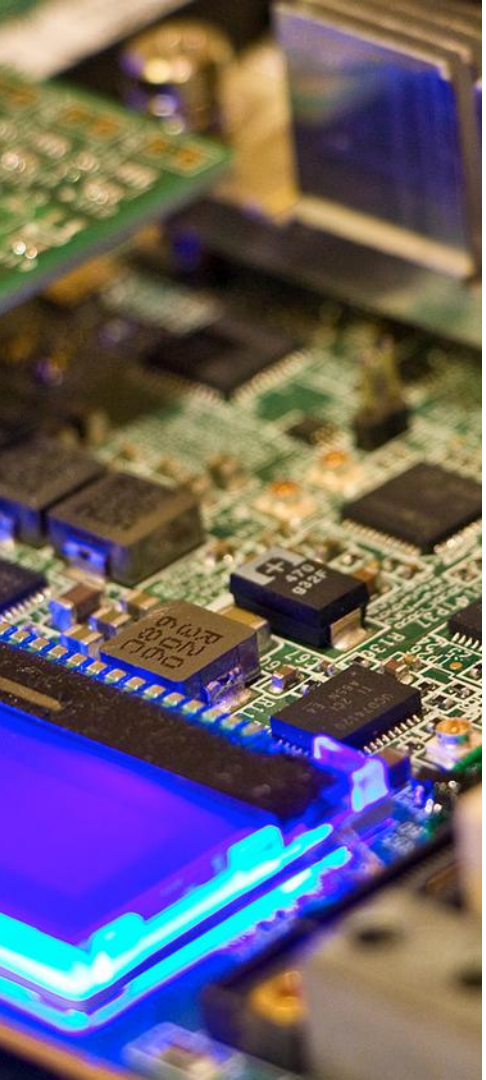


Interesting bugs found by clang

```
unsigned char a[X];
for(int i=0; i<X; i++)
   b = a ? tagCpe++ : tagSce++;
```

always true -- address of an array. This should have been a[i]

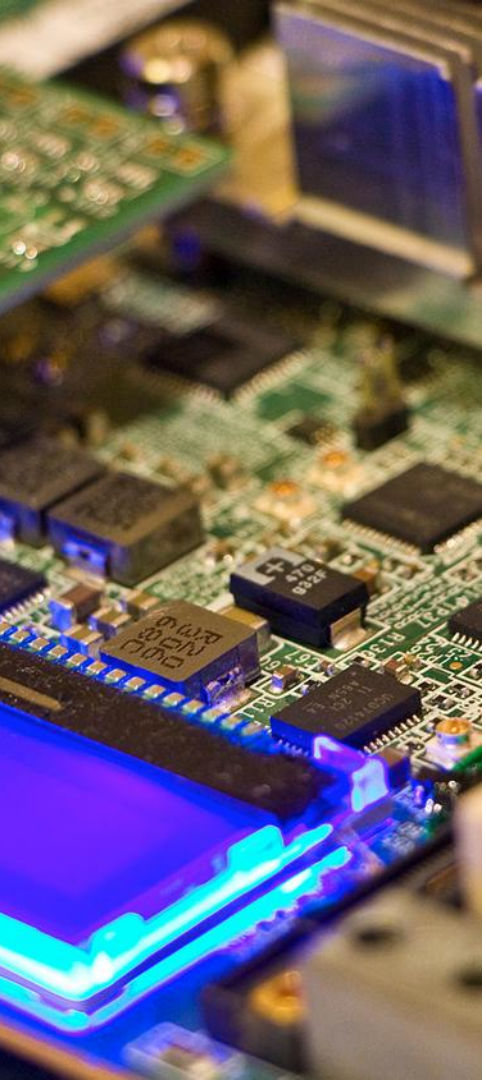
Linaro



glibc

musl is at a point where using it as the sole system libc is viable (if you don’t care about binary compatibility with other distributions).

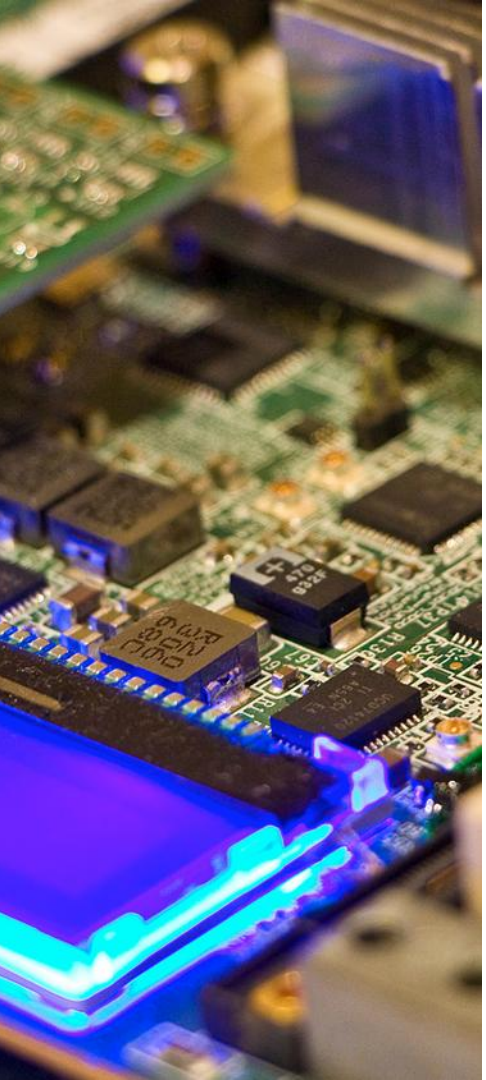
Linaro



glibc

clang currently doesn’t support musl, but that’s fixable: https://abf.io/openmandriva/llvm

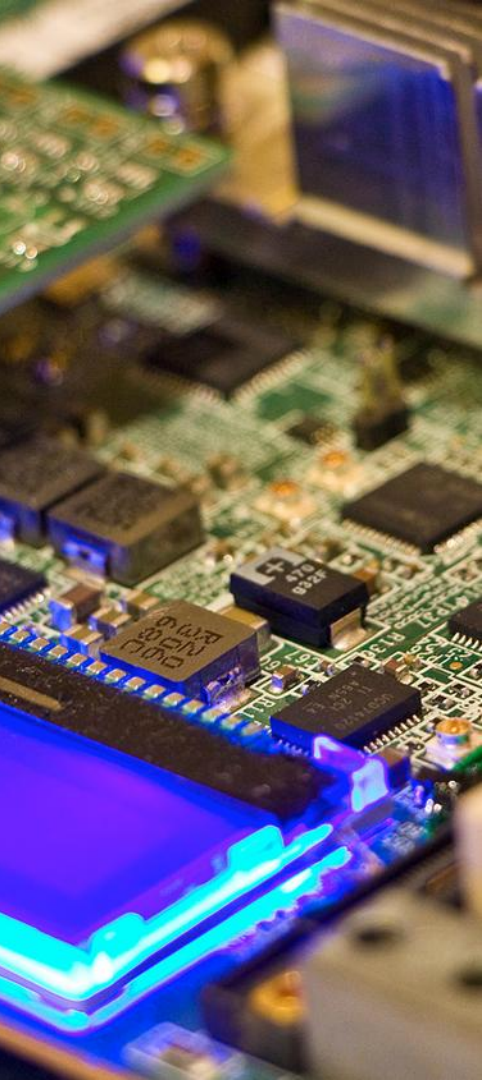
Linaro



libstdc++

LLVM’s libc++ is ready to replace libstdc++ where binary compatibility is not a concern.

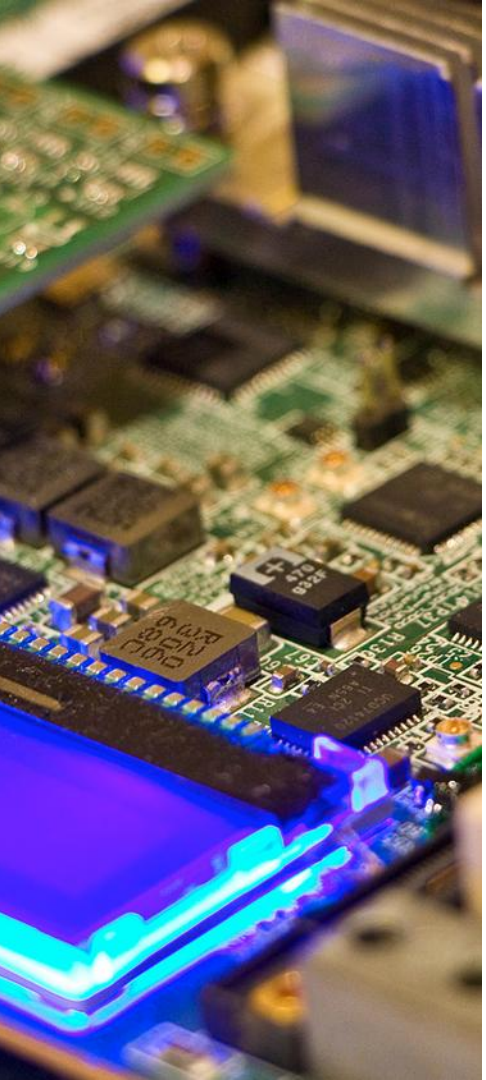
Linaro



libstdc++

Unfortunately, binary compatibility is a concern for many uses -- and while libstdc++ and libc++ can coexist, problems start showing up with other libraries (Qt linked to libc++, binary-only application uses Qt and links to libstdc++ → crash)

Linaro

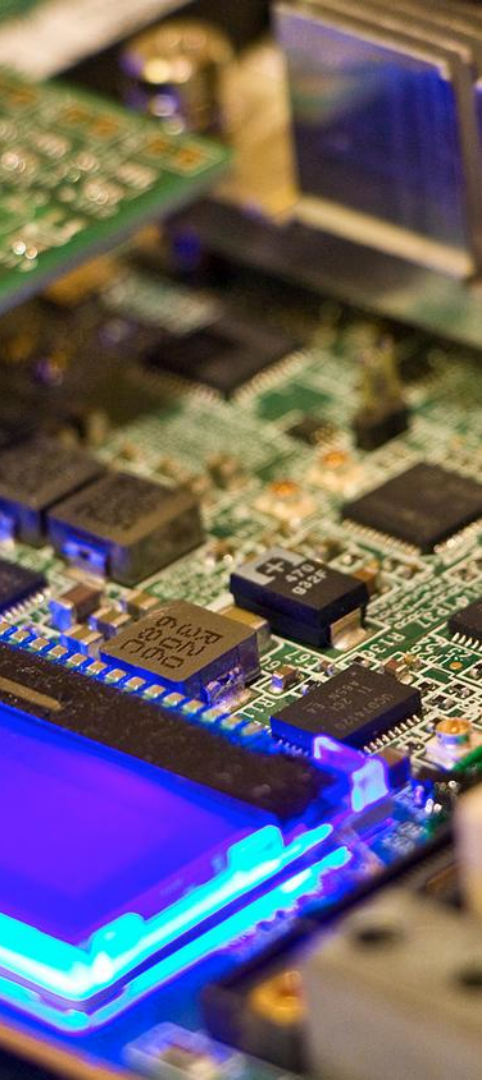


libstdc++

libc++ is the better choice if binary compatibility is not a concern -- roughly 50% space saved, full C++14 support.

Android is doing the right thing by switching to libc++ (from STLport)

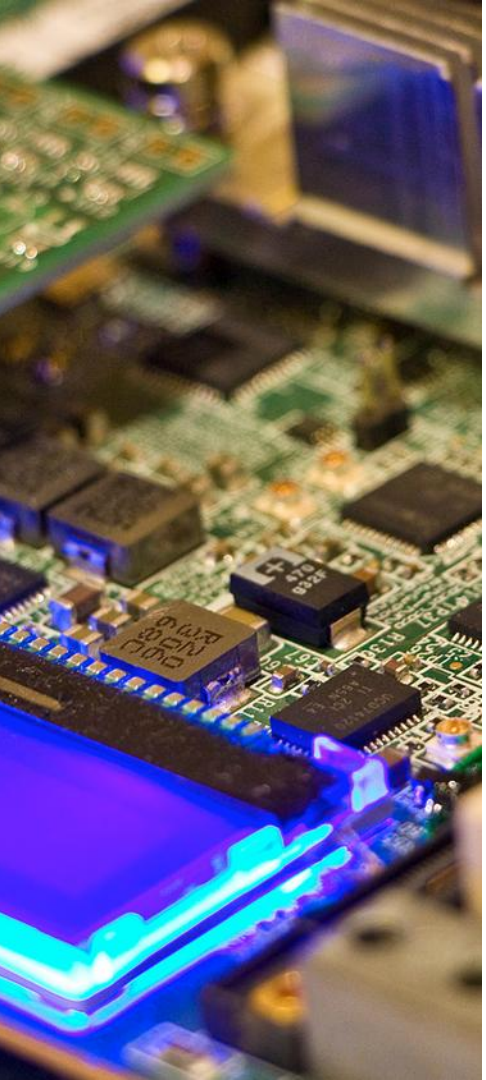
Linaro



crosscompiling

Switching to an LLVM/clang based toolchain is interesting for crosscompiling - a regular clang already has crosscompiling support built in, no need to build a fresh compiler for every new target

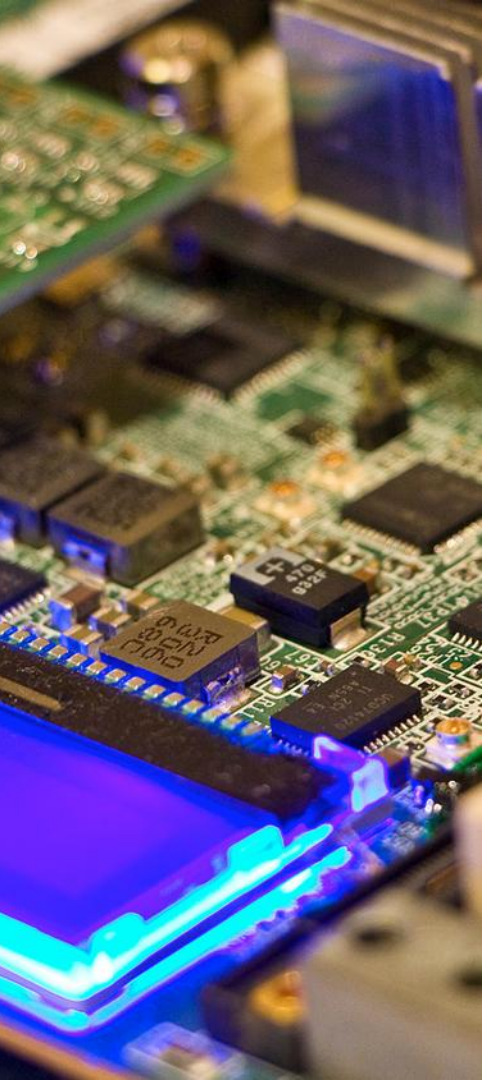
Linaro



crosscompiling

--sysroot in clang needs work: Still sees host system headers.
Wrapper scripts can be used to work around this.

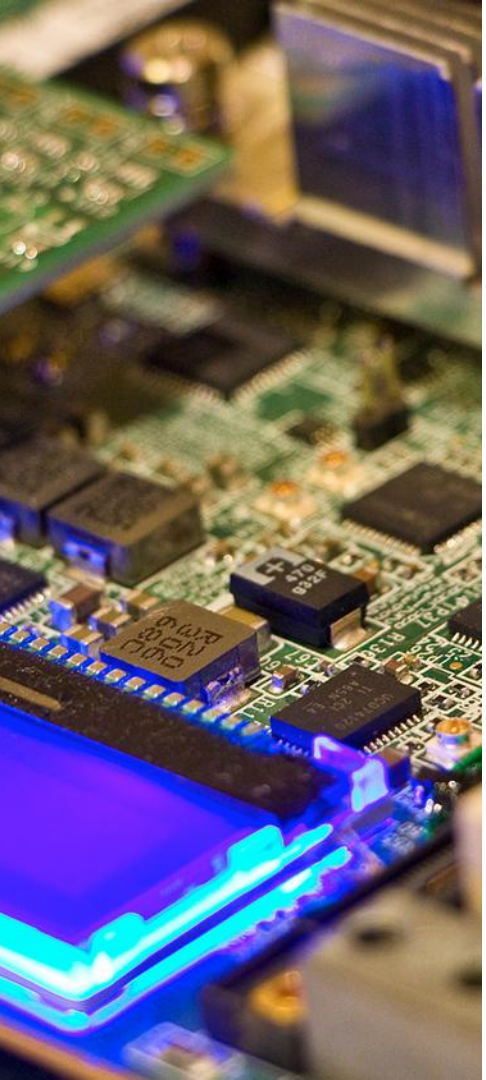
Linaro



crosscompiling

```
Lopts="-L$SYSROOT/usr/lib -L$SYSROOT/lib"
# Warnings like "argument unused during compilation"
# can break configure scripts
for i in "$@"; do
    if [ "$i" = "-E" -o "$i" = "-c" ]; then
        Lopts=""
        break
    fi
done
exec clang -target $TARGET \
    --sysroot=$SYSROOT -nostdinc \
    -isysroot $SYSROOT \
    -isystem $SYSROOT/usr/include \
    $Lopts \
    -ccc-gcc-name $TARGET-gcc "$@"
```

Linaro



crosscompiling

Automated toolchain and core system bootstrapping being worked on:
https://abf.io/openmandriva/crossbuild/blob/master/build-clang-musl.sh

Linaro

Questions? Comments?

bero@linaro.org