Opening a Linux Kernel in CLion IDE

Installing the build tools

Before you start, it's a good idea to make sure you have all the necessary prerequisites installed, and that you can build the kernel from your console (without CLion). On Debian/Devuan/Ubuntu Linux systems, you need to run `apt-get install build-essential` as `root`.

Getting the source code

Next, head to https://www.kernel.org and pick a 4.x or 5.x kernel tarball. Building an ancient 2.2.x or 2.4.x kernel on modern systems is possible, but may require extra tinkering. Alternatively, you can pull the most up to date Git `master` from https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git.

Bootstrapping

Now you need to make sure the project is both *clean* (meaning it doesn't contain any binaries that have already been built from source) and *ready for build* (meaning you've carried out all required configuration steps):

- `make distclean` is necessary as it removes any previously built binaries and object files;
- `make defconfig` creates the default `.config` file in the project directory. You can go ahead and copy over the `.config` from your running Linux kernel, or run `make menuconfig` instead and make any customizations (like enabling a specific driver or file system module, e.g. Btrfs);
- running `make clean` after this is still desirable, as `make defconfig` and `make menuconfig` both build a small number of binaries;
- don't run `make distclean` once you have the `.config` file, as it will get deleted and you'll have to start over.
- don't build the kernel manually (`make bzImage` and/or `make modules`) once you've run `make clean`, as CLion needs a clean project state.

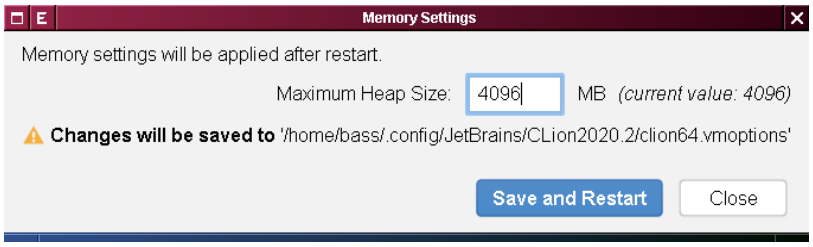
To summarize the above, use this shell one-liner and you'll be all set:

```
make distclean defconfig clean
```

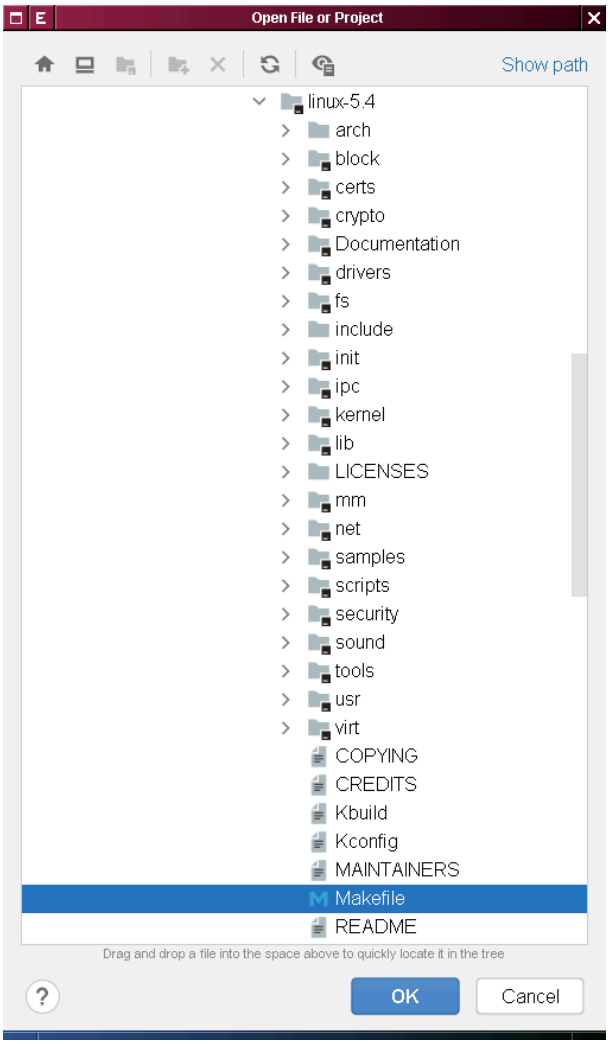
Creating a CLion project

Now we're ready to open the kernel in CLion (note you need version 2020.2+).

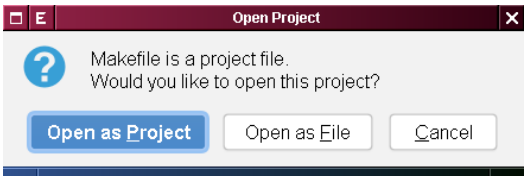
But first let's adjust the memory settings. This step is probably unnecessary if you only intend to build the kernel (`make bzImage`). But if you want to run `make modules` too (and you probably do), then increasing the heap size of the JVM is a good idea. Navigate to *Help → Change Memory Settings* and set the heap size to 4096 MB, or if you can, 8192 MB:



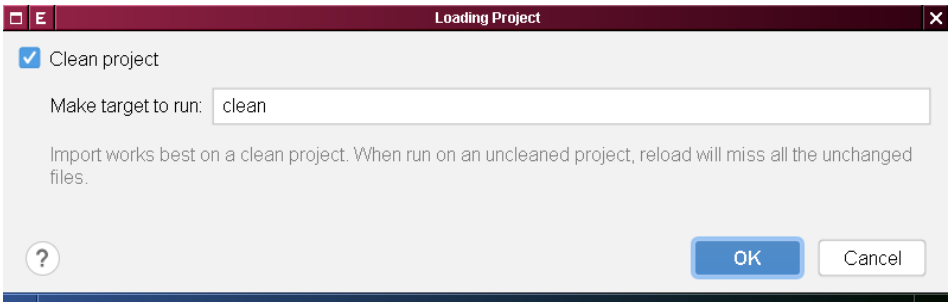
Once you've started CLion with the new memory settings, open `Makefile` from the project root:



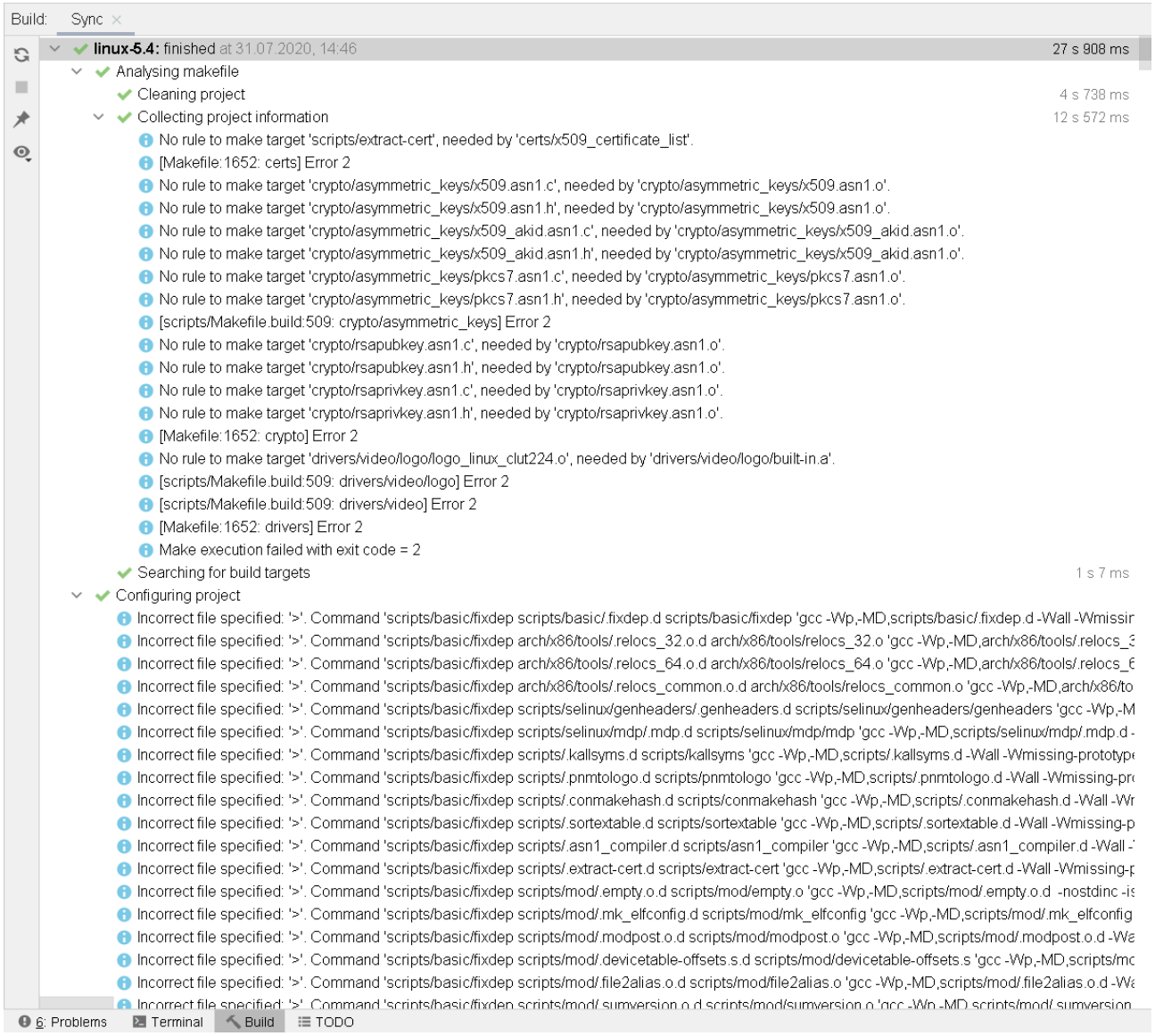
and select “*Open as Project*”:



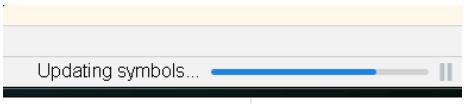
Next, CLion will ask whether you want to clean the project. Accept the defaults, and CLion will run `make clean` before analyzing the project structure:



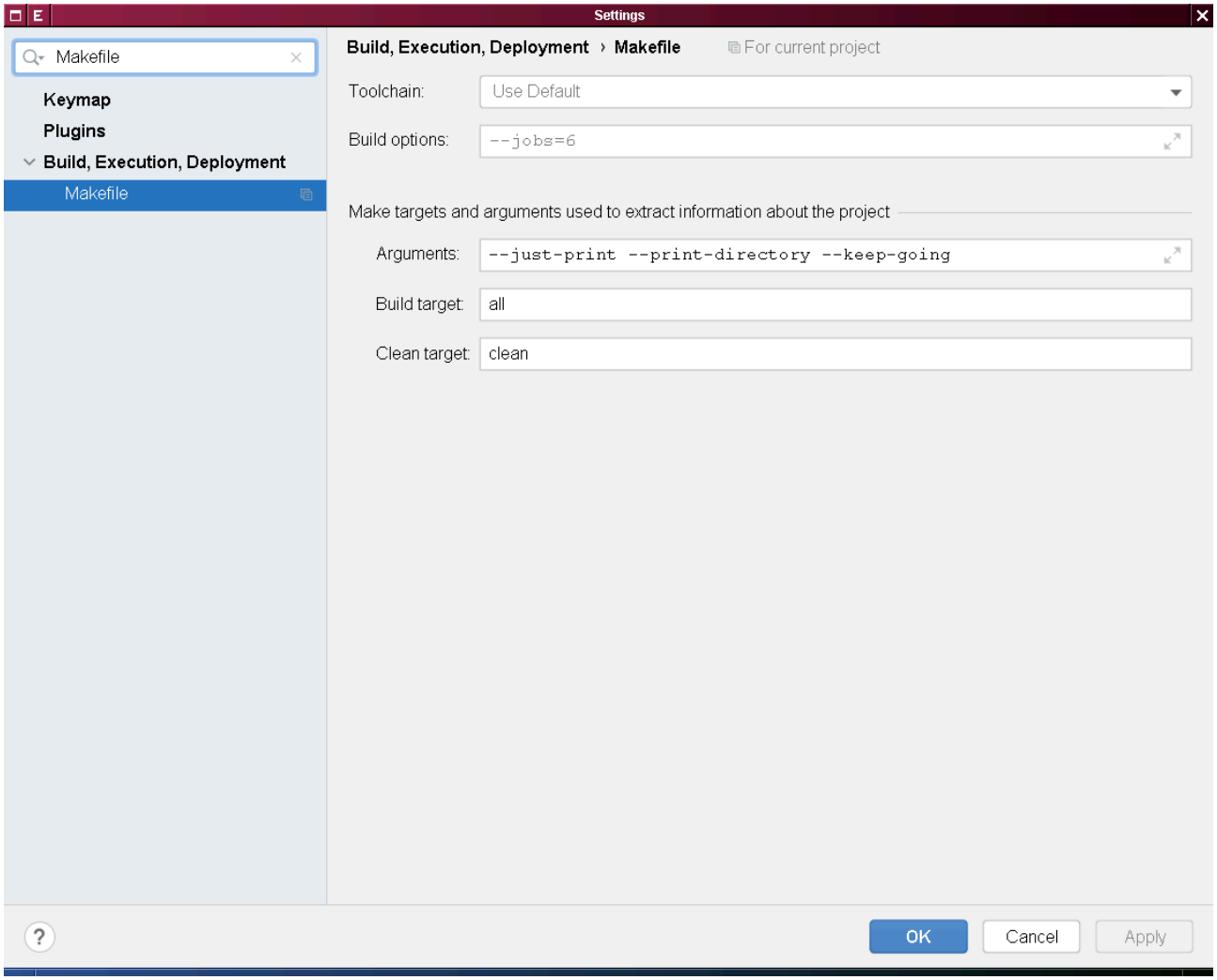
After a short while, the build tool window will tell you that the project analysis is complete. There will be multiple warnings, from both *GNU Make* and CLion itself, but you can ignore those:



Indexing symbols will take quite some time:



At this point, CLion has used the default Make targets to analyze the project. Let's adjust these values. Navigate to *File* → *Settings* → *Build, Execution, Deployment* → *Makefile*:

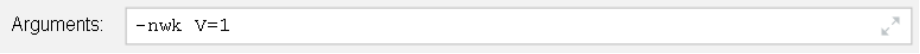


The *Arguments* field contains the Make switches used to analyze the project. Since the Linux kernel uses the *Kbuild* build system on top of GNU Makefiles, let's add `V=1` for extra verbosity: this will aid CLion in analyzing the project. This is not mandatory for the kernel, but will definitely help a lot with other *Kbuild*-based projects, such as QEMU, BusyBox, Git or Apache NuttX.

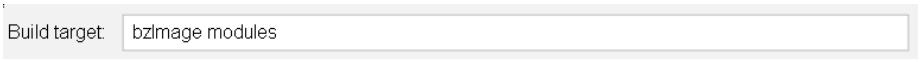
Also,

```
--just-print --print-directory --keep-going
```

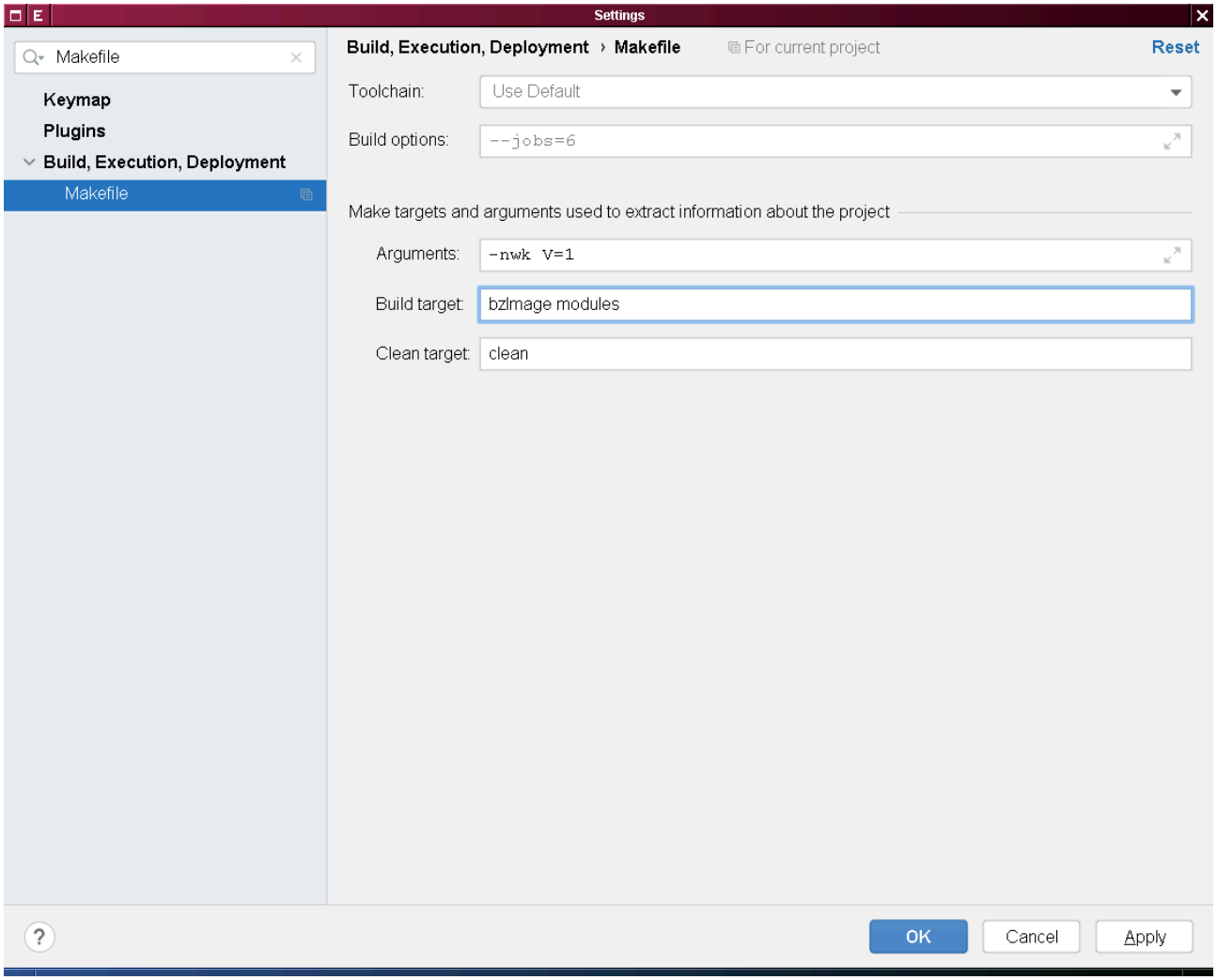
can be shortened to `-nwk`, so the resulting Make arguments will look like this:



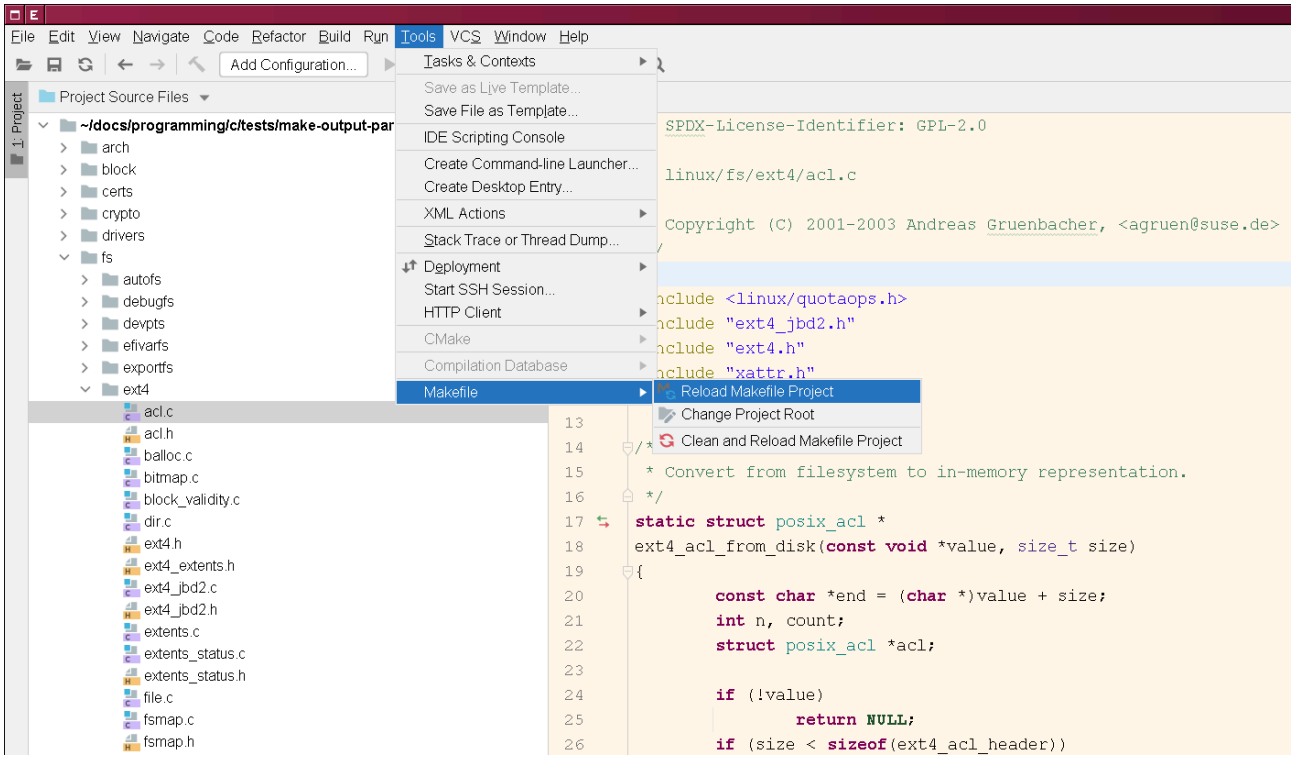
The `all` make target is just an alias for `bzImage`, so by default the project will include the kernel image but not the modules. Let's correct this (note the uppercase `I` in `bzImage`):



Here’s the overall view of the settings panel with all our changes:



In order for our changes to be applied to the project, invoke *Tools → Makefile → Reload Makefile Project*. The project will be reloaded with the updated settings applied:



If you modify any kernel flags, updating your `.config` file, this is the action you should invoke in order for CLion to reflect the changes.

Troubleshooting

If Makefile support in CLion doesn't work for you, feel free to contact us at clion-support@jetbrains.com.

You can also try opening the Linux kernel as a *compilation database* project instead. You'll need a *compilation database* generator like `compiledb` or `bear` (`bear` is best for the Linux kernel).

Just as you did with Makefile, run

```
make distclean defconfig clean
```

Next, from the project root, generate the `compile_commands.json` file by running

```
bear make V=1 bzImage modules
```

This will take a while, as the whole project will be built.

Finally, open the resulting `compile_commands.json` as a project in CLion.