4 minutes

Building the WSL 2 kernel with Clang

Recently, I built a computer for school that I installed Windows 10 Pro on (link to the current specs if you are curious). I was a little bummed about leaving Chrome OS because I was going to lose my local Linux development environment; however, Windows Subsystem for Linux is a thing and it has gotten even better with WSL 2, as it is actually running a Linux kernel so there is full Linux compatibility going forward. I also learned that it is possible to replace the Linux kernel that Microsoft ships with your own. This is a mini guide for what I uncovered because there is not a ton of information around for how to do this.

Building the kernel

You can either build the kernel on another machine and download it locally or you can build it within WSL 2 itself. This is a guide for how to do the latter because it is simpler and does not mess with an existing environment you might have working. I am giving the instructions for Debian but these should work for Ubuntu (I just do not want to tear down my current working environment to test…). I am also going to assume that you are familiar with Linux in general so I am not going to explain every single command.

1. Download and install the needed tools.

First, we are going to grab the utilities needed to build the kernel, along with some other tools to use for installing clang-10 and lld-10 from apt.llvm.org.

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install -y --no-install-recommends \
bc \
bison \
build-essential \
ca-certificates \
flex \
git \
gnupg \
libelf-dev \
libssl-dev \
lsb-release \
software-properties-common \
wget
```

Next, we will install a rolling release version of clang-10 and lld-10 from apt.llvm.org. There has been a lot of work done on the LLVM side to make building the Linux kernel a good experience and it is important to have all of those fixes available.

```
$ cd ~
$ wget https://apt.llvm.org/llvm.sh
$ chmod +x llvm.sh
$ sudo ./llvm.sh 10
```

If you are not on an Ubuntu or Debian based operating system, you can build `clang` and `ld.lld` from source easily with my `tc-build` script. Read the information in that README for how to use it and file issues at that repo if you run into any, I want to fix them!

2. Download the WSL 2 source code.

```
$ git clone https://github.com/microsoft/WSL2-Linux-Ker
$ cd WSL2-Linux-Kernel
```

3. Build the kernel.

I chose to use an out folder for easy clean up.

```
$ make -j$(nproc) -s CC=clang-10 KCONFIG_CONFIG=../Micr
```

If you want to just build it in tree:

```
$ make -j$(nproc) -s CC=clang-10 KCONFIG_CONFIG=Microso
```

You do not have to use `clang` and `ld.lld`, you can omit the `CC=clang-10` and `LD=ld.lld-10` to use `gcc` and `ld` respectively but I think that using `clang` is neat since I work on ClangBuiltLinux.

Using the kernel

To boot from the kernel we just compiled, we need to move it out of WSL 2 and add a `.wslconfig` to say that we want to boot from it.

1. Move the kernel to the Windows file system.

I have a separate folder for my kernels within my user folder (e.g., `C:\Users\natec\Linux`). It does not have to be there. Assuming that you used the out folder like I did above, the command will look something like this:

```
$ cp out.x86_64/arch/x86/boot/bzImage /mnt/c/Users/nate
```

2. Tell WSL 2 to use that kernel when booting up.

This was the tricky part that I had to do some reading about, as their documentation is not the best. This `.wslconfig` file has to be in your user folder's root (e.g., `C:\Users\natec\.wslconfig`). This is mine:

```
$ cat /mnt/c/Users/natec/.wslconfig
[wsl2]
kernel = C:\\Users\\natec\\Linux\\kernel-4.19-clang
```

You can create this with Visual Studio Code or a terminal editor like `vim`. Basically, you pass it the full path to your kernel binary with the "`\`" characters escaped. After that, you need to shutdown your WSL 2 VM in Powershell or `cmd` and just reopen it.

```
$ wsl --shutdown
```

After you have done that, you can run `cat /proc/version` to verify that WSL booted from that kernel.

Before:

```
$ cat /proc/version
Linux version 4.19.84-microsoft-standard (oe-user@oe-ho
```

After:

```
$ cat /proc/version
Linux version 4.19.84-microsoft-standard+ (nathan@Ryzen
```

I have noticed that the VM gets stuck starting with a custom kernel at times; if that happens, run `wsl --shutdown` and try reopening one of the distributions again, it will usually work after a couple of times.

Feel free to reach out to me with issues or questions on Twitter or my WSL 2 kernel source on Github.

#clang #linux #wsl2

803 Words

2019-12-17 12:53 -0800

READ OTHER POSTS

← Building and usin…