UNIX & LINUX

WSL 2 does not have /lib/modules/

Asked 1 year, 3 months ago Active 17 days ago Viewed 9k times

12

2

I have the source code of a hello world kernel module that works in Ubuntu 20 in a laptop. Now I am trying to compile the same code in Ubuntu 20 but inside WSL2. For that I am using this:

```
make -C /sys/modules/$(shell uname -r)/build M=$(PWD) modules
```

The problem is that `/lib/modules` is empty. It seems that WSL2 does not bring anything in `/lib/modules/4.19.104-microsoft-standard/build`

I tried getting the headers using:

```
sudo apt search linux-headers-`uname -r`

Sorting... Done
Full Text Search... Done
```

But nothing get's populated in the modules folder

Is there anything I need to do in order that folder contains all required modules?

[EDIT]

Getting closer thanks to @HannahJ.

I am doing:

```
> sudo make -C /home/<user>/WSL2-Linux-Kernel M=$(pwd) modules

SL2-Linux-Kernel M=$(pwd) modules
make: Entering directory '/home/<user>/WSL2-Linux-Kernel'
  CC [M]  /home/<user>/containers-assembly-permissionsdemo/demo-2/lkm_example.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/<user>/containers-assembly-permissionsdemo/demo-2/lkm_example.mod.o
  LD [M]  /home/<user>/containers-assembly-permissionsdemo/demo-2/lkm_example.ko
make: Leaving directory '/home/<user>/WSL2-Linux-Kernel'
```

At the end, I get the `lkm_example.ko` file created.

After that:

```
> sudo insmod lkm_example.ko
insmod: ERROR: could not insert module lkm_example.ko: Invalid module format
```

```
> dmesg
[200617.480635] lkm_example: no symbol version for module_layout
[200617.480656] lkm_example: loading out-of-tree module taints kernel.
[200617.481542] module: x86/modules: Skipping invalid relocation target, existing

value is nonzero for type 1, loc 0000000074f1d70f, val ffffffffc0000158


> sudo modinfo lkm_example.ko
filename:       /home/<user>/containers-assembly-permissionsdemo/demo-
2/lkm_example.ko
version:        0.01
description:    A simple example Linux module.
author:         Carlos Garcia
license:        GPL
srcversion:     F8B272146BAA2381B6332DE
depends:
retpoline:      Y
name:           lkm_example
vermagic:       4.19.84-microsoft-standard+ SMP mod_unload modversions
```

This is my Makefile

```
obj-m += lkm_example.o
all:
    make -C /home/<usr>/WSL2-Linux-Kernel M=$(PWD) modules
clean:
    make -C /home/<usr>/WSL2-Linux-Kernel M=$(PWD) clean
test:
    # We put a — in front of the rmmod command to tell make to ignore
    # an error in case the module isn’t loaded.
    -sudo rmmod lkm_example
    # Clear the kernel log without echo
    sudo dmesg -C
    # Insert the module
    sudo insmod lkm_example.ko
    # Display the kernel log
    dmesg
unload:
    sudo rm /dev/lkm_example
    sudo rmmod lkm_example
```

[Edit2] This is my kernel module:

```
    #include <linux/init.h>
    #include <linux/module.h>
    #include <linux/kernel.h>
    #include <linux/fs.h>
    #include <asm/uaccess.h>
    #include <linux/init_task.h>

    MODULE_LICENSE("GPL");
    MODULE_AUTHOR("Carlos Garcia");
    MODULE_DESCRIPTION("A simple example Linux module.");
    MODULE_VERSION("0.01");

    /* Prototypes for device functions */
    static int device_open(struct inode *, struct file *);
    static int device_release(struct inode *, struct file *);
    static ssize_t device_read(struct file *, char *, size_t, loff_t *);
    static ssize_t device_write(struct file *, const char *, size_t, loff_t *);
    static int major_num;
    static int device_open_count = 0;
    static char msg_buffer[MSG_BUFFER_LEN];
    static char *msg_ptr;
    /* This structure points to all of the device functions */
    static struct file_operations file_ops = {
        .read = device_read,
        .write = device_write,
        .open = device_open,
        .release = device_release
    };

    /* When a process reads from our device, this gets called. */
    static ssize_t device_read(struct file *flip, char *buffer, size_t len,
loff_t *offset)
    {
     ...
    }

    /* Called when a process tries to write to our device */
    static ssize_t device_write(struct file *flip, const char *buffer, size_t
len, loff_t *offset)
    {
       ...
    }

    /* Called when a process opens our device */
    static int device_open(struct inode *inode, struct file *file)
    {
        ...
        try_module_get(THIS_MODULE);

    }

    /* Called when a process closes our device */
    static int device_release(struct inode *inode, struct file *file)
    {
        ...
        module_put(THIS_MODULE);
    }

    static int __init lkm_example_init(void)
    {
        ...
        major_num = register_chrdev(0, "lkm_example", &file_ops);
        if (major_num < 0)
        {
            printk(KERN_ALERT "Could not register device: % d\n", major_num);
            return major_num;
        }
        else
        {
            printk(KERN_INFO "lkm_example module loaded with device major number
% d\n", major_num);
            return 0;
        }
    }

    static void __exit lkm_example_exit(void)
    {
        /* Remember - we have to clean up after ourselves. Unregister the
character device. */
        unregister_chrdev(major_num, DEVICE_NAME);
        printk(KERN_INFO "Goodbye, World !\n");
    }
    /* Register module functions */
    module_init(lkm_example_init);
    module_exit(lkm_example_exit);
```

linux-kernel kernel-modules c++ windows-subsystem-for-linux

Share Improve this question Follow

edited Apr 26 at 8:42

asked Jun 22 '20 at 22:54

Carlos Garcia
389 ● 2 ● 11

3 Answers

Active Oldest Votes

11

I had to do this for an assignment, so I figure I'll share my solution here.

The base WSL2 kernel does not allow modules to be loaded. You have to compile and use your own kernel build.

How to compile and use a kernel in WSL2:

1. `sudo apt install build-essential flex bison libssl-dev libelf-dev git`
2. `git clone https://github.com/microsoft/WSL2-Linux-Kernel.git`
3. `cd WSL2-Linux-Kernel`
4. `cp Microsoft/config-wsl .config`
5. `make -j $(expr $(nproc) - 1)`
6. From Windows, copy `\\wsl$\DISTRO\home\USER\WSL2-Linux-Kernel\arch\x86\boot\bzimage` to `C:\Users\WIN10_USER`
7. Create the file `C:\Users\WIN10_USER\.wslconfig` that contains:

```
[wsl2]
kernel=C:\\Users\\WIN10_USER\\bzimage
```

Note: The double slashes (\\) are intentional. Also, to avoid a potential old bug, make sure not to leave any trailing whitespace on either line.

8. In PowerShell, run `wsl --shutdown`
9. Reopen your flavor of WSL2

How to compile the module:
Note: You'll want to do these from /home/USER/ or adjust the Makefile to match your location.

1. Create a `Makefile` that contains:

```
obj-m:=lkm_example.o

all:
    make -C $(shell pwd)/WSL2-Linux-Kernel M=$(shell pwd) modules

clean:
    make -C $(shell pwd)/WSL2-Linux-Kernel M=$(shell pwd) clean
```

2. `make`

Source for the .wslconfig file steps: https://www.bleepingcomputer.com/news/microsoft/windows-10-wsl2-now-allows-you-to-configure-global-options/

Share Improve this answer Follow

edited Sep 7 at 9:10
Massimo
895 ● 1 ● 9 ● 23

answered Apr 14 at 23:00
Hannah J
126 ● 1 ● 3

Wow Hannah! Thank you very much. I will give it a try :) – Carlos Garcia Apr 16 at 9:06

Still debugging, I am getting: `insmod: ERROR: could not insert module lkm_example.ko: Invalid module format make: *** [Makefile:14: test] Error 1` – Carlos Garcia Apr 16 at 23:23

When i run `uname -r` I get `5.4.72-microsoft-standard-WSL2`. Is that the right version when compiling the kernel? – Carlos Garcia Apr 16 at 23:50

Hm. So if I'm understanding correctly, you compiled a kernel and switched to that, but you're now getting complaints about a Makefile, is that correct? Could you share your Makefile content? The one that holds this line: `make -C $(shell pwd)/WSL2-Linux-Kernel M=$(shell pwd) modules` – Hannah J Apr 17 at 11:09

1 crap! i had an embarrassing mistake. I had a typo in .wslconfig, even when you have explicitly mentioned :S - Your solution works very well, and it also allows me to make changes to the kernel itself, so I love it x2 :) Thanks a lot for your help – Carlos Garcia Apr 26 at 9:51

9

The Windows Subsystem for Linux version 2 uses a custom Linux kernel from Microsoft that contains all of its drivers compiled in. While it has support for modules, it doesn't contain any, as you can see from the configuration file. Therefore, there's no reason to ship a `/lib/modules` directory.

In addition, most of the Linux distros available for WSL don't ship their WSL versions with a kernel at all. Since Microsoft ships their own, there's no reason to. Those Linux distros don't ship build packages for Microsoft's kernel because they aren't responsible for it; Microsoft is, and you'd need to talk to them about the packages.

It may be the case that you can load the modules into the kernel if you use the standard tools, but you'll likely need to build against an appropriate source tree. You can either try to find the appropriate version in the GitHub repository I linked to above, or you may need to contact Microsoft and ask for the source under the GPLv2, which they are required to provide you upon request.

I will note that WSL is not designed to allow loading custom kernel modules; it's not designed to be a full Linux environment, but rather to allow folks to develop and run standard Linux applications on Windows. If you want to do Linux kernel development, you'll probably need a full Linux installation.

Share Improve this answer Follow

answered Jun 28 '20 at 15:47
bk2204
2,738 ● 3 ● 7

1

WLS2 does not have a Linux Kernel.

> there are no kernel headers for WSL's "kernel" (actually a Windows driver)

Source

You might be able to get away with compiling still by getting another kernel

```
# If asked to install grub, you can simply skip using Esc
sudo update && sudo apt install linux-generic -y


ls /lib/modules
5.7.2-arch1-1
```

You will probably have different output, but you should have kernel headers installed. You can modify your original command using whatever you get in `lib/modules`

```
make -C /lib/modules/<kernel source from previous step>/build M=$(PWD) modules
```

Share Improve this answer Follow edited Jun 23 '20 at 4:00

answered Jun 23 '20 at 3:51
mbtamuli
81 ● 8

3 Thank you very much for your answer! I believe the biggest difference from WSL and WSL2 is that the latest has a real kernel: github.com/microsoft/WSL2-Linux-Kernel. I still don't know how to get the headers, I am too new to this :) – Carlos Garcia Jun 23 '20 at 9:00