Pro Windows Subsystem for Linux (WSL)

Powerful Tools and Practices for Cross-Platform Development and Collaboration

—

Hayden Barnes

Apress®

Pro Windows Subsystem for Linux (WSL)

Powerful Tools and Practices for Cross-Platform Development and Collaboration

Hayden Barnes

Apress®

Pro Windows Subsystem for Linux (WSL): Powerful Tools and Practices for Cross-Platform Development and Collaboration

Hayden Barnes
Columbus, GA, USA

ISBN-13 (pbk): 978-1-4842-6872-8 ISBN-13 (electronic): 978-1-4842-6873-5
https://doi.org/10.1007/978-1-4842-6873-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Joan Murray
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media LLC, 1 New York Plaza, Suite 4600, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484268728. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

This book is dedicated to the
Windows Subsystem for Linux community.

Table of Contents

About the Author

Hayden Barnes is the Senior Engineering Manager for Windows Containers at SUSE and a recognized Microsoft MVP. Hayden was previously Engineering Manager for Ubuntu on Windows Subsystem for Linux (WSL) at Canonical. Hayden regularly presents on the topic of WSL at conferences such as Microsoft Build and is the founder of WSLConf, the first community conference dedicated to WSL. He has consulted for enterprises, academic institutions, and government agencies to help them deploy WSL. Before joining Canonical, Hayden founded Whitewater Foundry, the first company to create a custom Linux distribution built specifically for WSL. He is passionate about WSL because it opens up a myriad of opportunities for cross-platform development, open source development, and collaboration between Linux and other communities.

About the Technical Reviewer

Nuno Do Carmo is an IT professional with 20 years of experience in various Windows domains, such as Windows OS support, Active Directory management, and application support.

He was also a Unix, HP/UX and Solaris 10, and Linux system administrator for HP for more than five years.

It is this mix of experiences that motivated him to start using WSL since its inception in 2016, and he was very happy to find other (crazy) persons who had the same interest in this incredible technology.

One of those persons was a certain Hayden Barnes.

Today, Nuno is a Microsoft MVP, Windows Insider MVP, CNCF Ambassador, and Docker Captain, and he is specially interested or invested in bringing the Cloud Native technologies to WSL to allow a broader user base to use these amazing projects.

On the personal side, Nuno lives in tiny Switzerland, the French side, with his wife, stepdaughter, stepson, and four cats.

He can be reached at

LinkedIn: www.linkedin.com/in/ndocarmo/

Twitter: https://twitter.com/nunixtech

Website: https://wsl.dev

Acknowledgments

I would like to acknowledge the following persons, without whom this book would not be possible:

Taylor Brown
Kayla Cinnamon
Sarah Cooley
Nuno Do Carmo
Yosef Durr
Sven Groot
Scott Hanselman
Ben Hillis
Dustin Howett
Igor Ljubuncic
Daniel Llewellyn
Craig Loewen
Kim Mullis
Tara Raj
Carlos Ramirez
Sohini Roy
Clint Rutkas
Mark Shuttleworth
John Starks
Rich Turner
Martin Wimpress
Patrick Wu

And everyone at Microsoft who contributed to the development of WSL and the community around WSL.

Introduction

Pro Windows Subsystem for Linux

This book will equip you with a wide breadth of WSL knowledge to tackle a range of challenges on WSL, from IT administration to development work, including:

- Connecting to popular Integrated Development Environments
- Building a custom Linux kernel for WSL 2
- Building a derivative Linux distro with your own packages
- Automating emails in Outlook from a bash script in WSL

It will also cover advanced settings, customization, and optimizations for both WSL and WSL 2, from the command line to the Registry. This will include each configuration option in wsl.conf and .wslconfig and recommendations for best performance.

First, we will cover the early development of WSL.

History of Windows Subsystem for Linux

What we now call WSL 1 began as an effort inside Microsoft, code-named Project Astoria, to support Android applications on the ill-fated Windows Phone. Project Astoria, known publicly as Windows Bridge for Android, was announced at the Microsoft Build conference for developers in 2015 (Figure I-1).

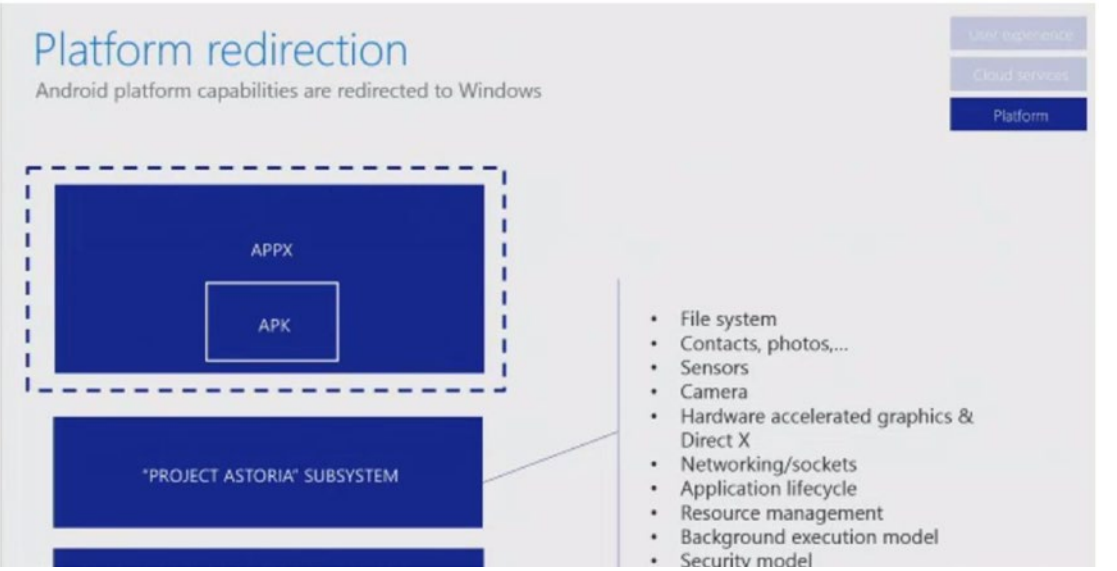

Figure I-1. *Screenshot of original presentation on Project Astoria, 2015. Source:* https://channel9.msdn.com/Events/Build/2015/2-702

Project Astoria built on virtualization concepts from a Microsoft Research project known as Project Drawbridge. Project Drawbridge was a prototype of a new form of virtualization for application sandboxing based on pico processes. Project Drawbridge included a version of Windows built to run inside pico processes. In Project Astoria, the pico process environment was modified to run Linux-based Android applications.

A single build of Windows 10 Phone was leaked from Microsoft containing Project Astoria. This build allowed rudimentary Android applications to be sideloaded and run on Windows Phone devices. Project Astoria was never officially released and was terminated by Microsoft along with Windows Phone in 2017. The underlying technology in Project Astoria survived though. After all, Android is based on Linux.

Bash on Ubuntu on Windows

In 2016, the technology underneath Project Astoria was recycled as Bash on Ubuntu on Windows. Instead of running Android applications, the technology was used to run a Bash terminal for developers on Windows (Figure I-2). The core of the technology was a binary translation layer that ran Linux binaries on an NT kernel inside pico processes, discussed in Chapter 1, "WSL Architecture."

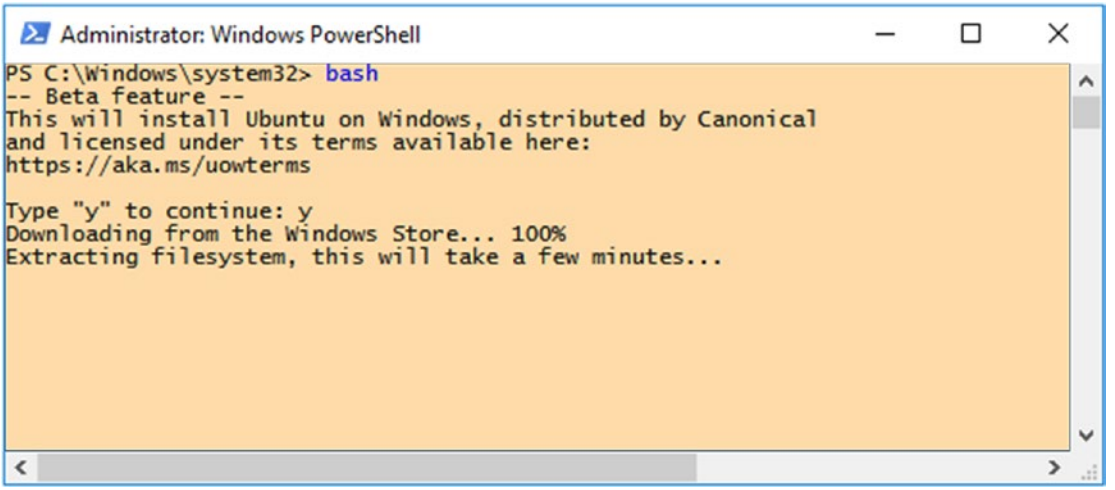

Figure I-2. *Screenshot of Bash on Ubuntu on Windows*

Microsoft partnered with Canonical, the publishers of Ubuntu, to bring this first version of WSL to Windows. Bash on Ubuntu on Windows shipped in Windows 10 Anniversary Update, also known as Windows 10 1607. The partnership between Microsoft and Canonical was a milestone in Microsoft's increasing adoption of Linux and open source software.

Windows Subsystem for Linux

In 2017, Bash on Ubuntu on Windows became Windows Subsystem for Linux with Windows 10 1709. Linux distributions could now be installed on WSL from the Microsoft Store (Figure I-3), and the number of available distributions expanded. A complete list of available WSL distributions is in Chapter 2, "Enabling WSL." This version of WSL based on the binary translation layer is what we now call WSL 1 following the announcement of WSL 2 in 2019.

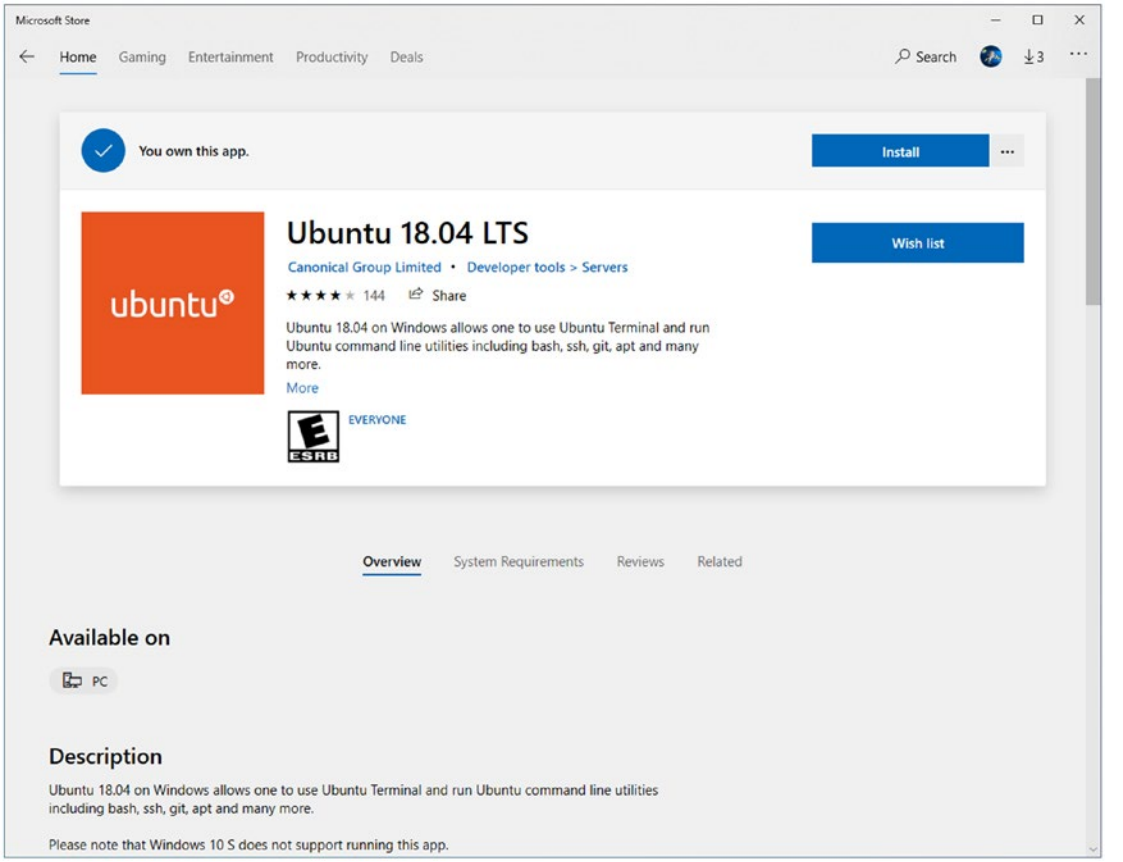

Figure I-3. *Ubuntu 18.04 LTS on the Microsoft Store*

Why "Windows Subsystem for Linux"?

Some people wonder why Windows Subsystem for Linux is so awkwardly named. From a historical perspective, Windows Subsystem for Linux matches the naming structure of Windows Services for UNIX, the previous POSIX compatibility layer for Windows NT. According to Rich Turner, Senior Program Manager at Microsoft, it was trademark concerns that prevented Microsoft from starting the product name with Linux. At the time, WSL did not contain the Linux kernel like it would with WSL 2. We ended up with Windows Subsystem for Linux, which you can think about as a subsystem of Windows to run Linux.

Windows Subsystem for Linux 2

WSL 2 was announced at Microsoft Build 2019 and reached general availability in Windows 10 2004. The core of WSL 2 was not a binary translation layer like WSL 1 but a full Linux kernel and environment running in a lightweight Hyper-V container. Unlike traditional Hyper-V, which is limited to Windows 10 Enterprise, Professional, Education, and Server, WSL 2 is available for all Windows 10 editions, including Windows 10 Home. WSL 2 offered significant improvements in application compatibility and performance over WSL 1. WSL 2 enabled several highly requested features to be brought to WSL, including GPU acceleration, official GUI support, and nested virtualization for KVM guests.

CHAPTER 1

WSL Architecture

To get the most out of Windows Subsystem for Linux, it is useful to understand its underlying architecture and history. If you plan to hack on WSL or just deploy it in your enterprise, it is necessary to know. This chapter will cover the architecture of WSL 1 and WSL 2, how we got here, and then dive into some of the bleeding edge features on the WSL platform.

WSL 1 vs. WSL 2

WSL 1 creates a Linux environment on Windows through the use of a Linux binary translation layer. WSL 2 does so with a lightweight virtualization platform based on Hyper-V. Both are unique and fascinating approaches to achieving Linux and Windows interoperability.

Kernel Drivers

When WSL is enabled on Windows 10, two NT core kernel drivers are loaded by Windows 10 (Figure 1-1). These drivers are Lxss.sys, a stub driver loaded early in the boot process, and LxCore.sys, the full WSL driver, which is loaded later in the boot process.

H. Barnes, *Pro Windows Subsystem for Linux (WSL)*, https://doi.org/10.1007/978-1-4842-6873-5_1

Pico Processes

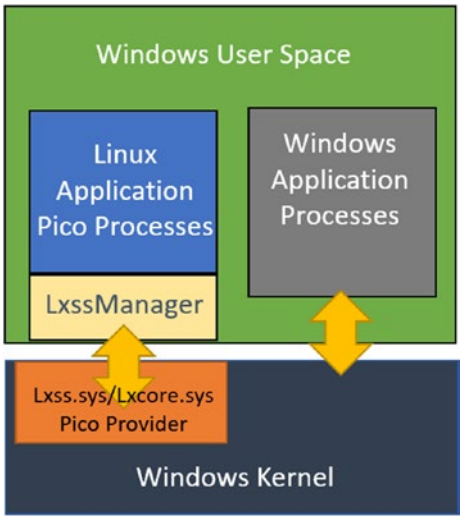

Figure 1-1. *Diagram of WSL 1 architecture*

Linux applications in WSL are executed in *pico processes*, lightweight virtual spaces created in Windows user space. LxCore.sys acts as a *pico provider* simulating a Linux environment in the virtual space inside WSL pico processes. LxCore.sys also performs the syscall translation in WSL 1, which is discussed in more detail below.

Linux applications running in a pico process are completely unaware they are running on Windows. Pico process technology can theoretically be used to simulate *any* operating system environment. Similar pico process technology is used by Microsoft to allow Windows 10 IoT to run legacy Windows CE applications and was used, in reverse, to port Microsoft SQL Server to Linux.

LxssManager

LxssManager is a Windows Service (Figure 1-2) that serves as a broker to LxCore.sys. An NT call to execute a Linux binary is routed by LxssManager to LxCore.sys. LxssManager also monitors the WSL user state and ensures smooth installation and uninstallation of WSL distributions (Figure 1-2).

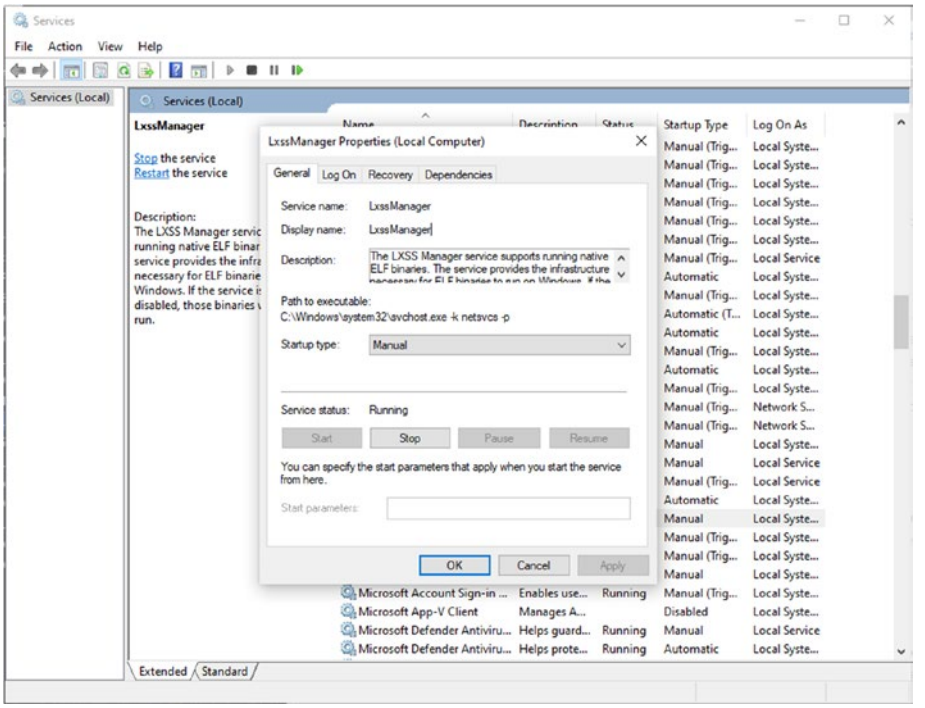

Figure 1-2. *Properties of the LxssManager service*

Syscall Translation in WSL 1

LxCore.sys must translate Linux kernel system calls from Linux applications into NT system calls. System calls are the low-level requests made by compiled binaries to an operating system kernel to perform tasks, such as apportion memory, open files, and read from devices.

To read the system calls created by a simple application, install `strace` on Ubuntu on WSL:

```
$ sudo apt install strace -y
```

Then run a simple application with strace, outputting the system call trace output to a file called output.txt:

```
$ strace -o output.txt echo 'hello world'
```

You can then read the strace output with cat:

```
$ cat output.txt
```

And you can see the Linux system calls made from simply running echo 'hello world' (Figure 1-3):

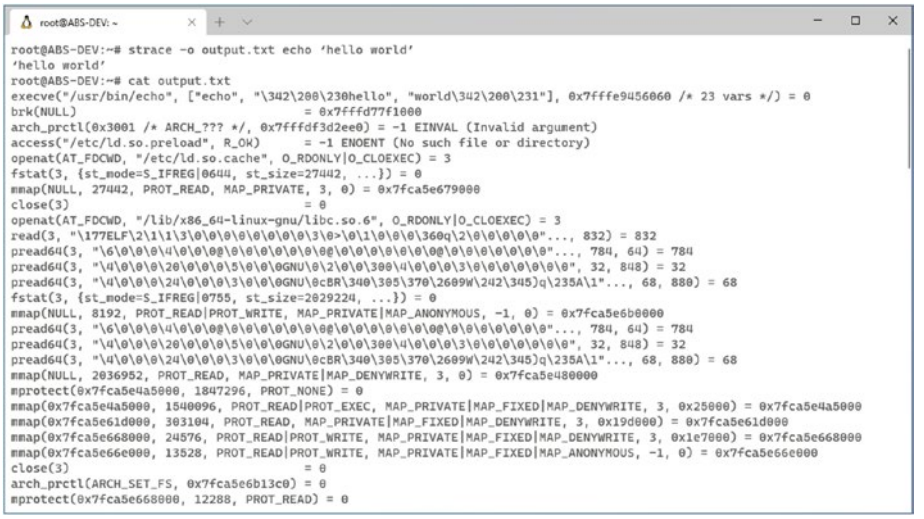

Figure 1-3. *System calls made by the command echo 'hello world'*

These system calls involve executing the binary, opening files, assigning memory, and writing the output to the console.

The Linux and Windows NT kernels were both created in the early 1990s. Linux was created by Linus Torvalds in 1991, inspired by MINIX, a research operating system, which was itself inspired by UNIX from Bell Labs. The Windows NT kernel was designed in 1993 by a team at Microsoft under David N. Cutler whose background at Digital Equipment Corporation on the VMS operating system heavily influenced the design of NT. Like Linux, NT was originally designed to be POSIX standards compatible. Like Linux, NT also had distinct modes of execution, a kernel privileged mode and a user unprivileged mode. Despite their very distinct implementations, the NT and Linux kernels were influenced by the same operating system theories and trends in the early to mid-1990s.

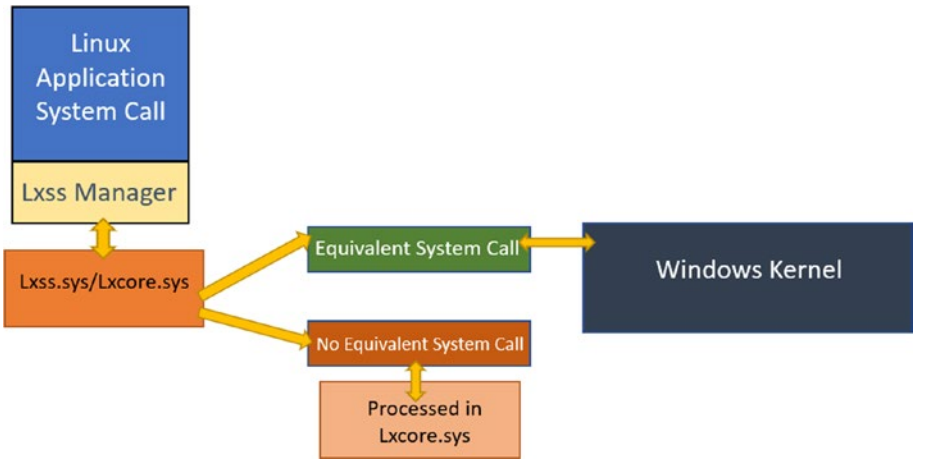

Figure 1-4. *Diagram of system call handling in WSL 1*

Despite their distinct implementations, thanks to their common influences in some cases there are direct translations of Linux to NT kernel system calls. When a Linux binary is executed in a pico process and a 1-1 Linux to NT system call exists, these can be passed directly by LxCore.sys to the NT kernel (Figure 1-4).

When a 1-1 Linux to NT system call does not exist, but a similar NT system call exists for a Linux system call, LxCore will translate the Linux system call into an NT system call by reordering or refactoring the call from a Linux to an NT call.

In other cases, where there is no equivalent for a Linux system call. Here, LxCore must handle the system call itself. This is handled in a clean room implementation of the Linux kernel API and contains no Linux kernel code.

Not all possible Linux system calls are implemented in WSL 1, and some of the more obscure system calls will never be. WSL 1 achieves close to 90% binary compatibility with Linux with this translation layer. Early in WSL 1 development, the Linux Test Project (`https://linux-test-project.github.io/`) was used to validate Linux compatibility.

WSL 2

WSL 2 is a vastly different architectural approach from WSL 1. By leveraging a Linux kernel and a lightweight Hyper-V container, WSL 2 addresses many of the issues users encountered with WSL 1, such as syscall incompatibility.

Hyper-V

WSL 2 addresses the challenge of implementing complete system call translation support for every possible Linux system call by implementing a true Linux kernel in a lightweight virtualization platform built on Hyper-V (Figure 1-5).

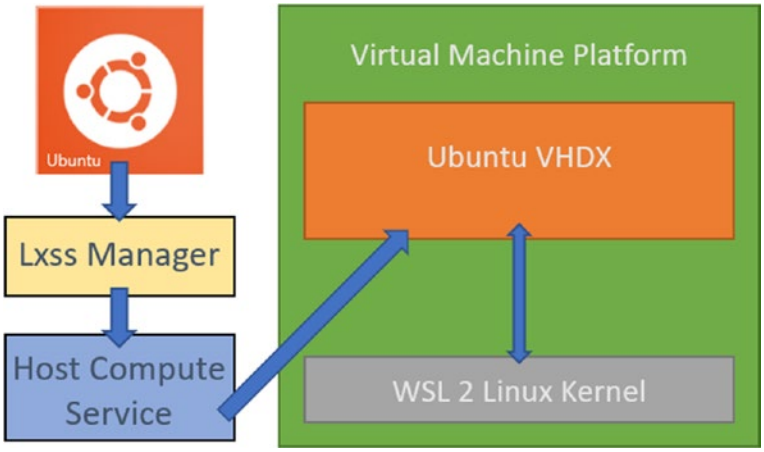

Figure 1-5. *Diagram of WSL 2 architecture*

Hyper-V is the native virtualization technology built into the Windows NT kernel, equivalent to the native virtualization implementations such as KVM on Linux or Hypervisor.Framework on macOS.

Hyper-V is a Type 1 hypervisor, which means it runs at the NT kernel level. Third-party hypervisors, like VirtualBox and VMware, are Type 2 hypervisors which load kernel-level drivers but are mostly implemented user space.

This is why Hyper-V and third-party hypervisors clash. You cannot run VirtualBox and WSL 2 on the same Windows installation.

Virtual Machine Platform

WSL 2 uses the Windows Host Compute Service, an API built on Hyper-V and exposed by enabling the Virtual Machine Platform in Windows 10.

WSL 2 defines a lightweight Linux environment through a series of API calls to the Host Compute Service, including attaching virtual file systems and virtual network adapters.

In contrast with WSL 1, where calls to open files or to open a networking port on WSL 1 are handled directly by the NT kernel, calls to open files or a networking port on WSL 2 are handled by the Linux kernel which then interacts with virtual Hyper-V devices.

Calls to open files on WSL 2 are directed to the Linux kernel which interacts with a virtual file system emulated by Hyper-V. The virtual file system is a hard disk image mounted as a virtual block device upon which the file system is stored. This provides greater performance and lower overhead than emulating a physical IDE, SATA, or NVMe device.

Calls to open a networking port on WSL 2 are directed to the Linux kernel which interacts with a virtual network adapter emulated by Hyper-V.

You can view the virtual file systems and network adapters in WSL 2 by running

```
$ lshw
```

The device drivers for these virtual Hyper-V file systems and devices have been included in the upstream Linux kernel since 2009. Here is an example of a virtual network adapter:

```
  *-network:0
       description: Ethernet interface
       physical id: 1
       logical name: eth0
       serial: 00:10:4d:eb:01:ab
       size: 10Gbit/s
       capabilities: ethernet physical
       configuration: autonegotiation=off broadcast=yes driver=hv_netvsc
       duplex=full firmware=N/A ip=172.24.18.219 link=yes multicast=yes
       speed=10Gbit/s
```

WSL 2 Kernel

The Linux kernel in WSL 2 is a slightly modified Linux kernel optimized to run in the Hyper-V-based WSL 2 environment. The source is made available under GPL 2.0 at https://github.com/microsoft/WSL2-Linux-Kernel. Updates to the Linux kernel in WSL 2 are provided by Windows Update (Figure 1-6).

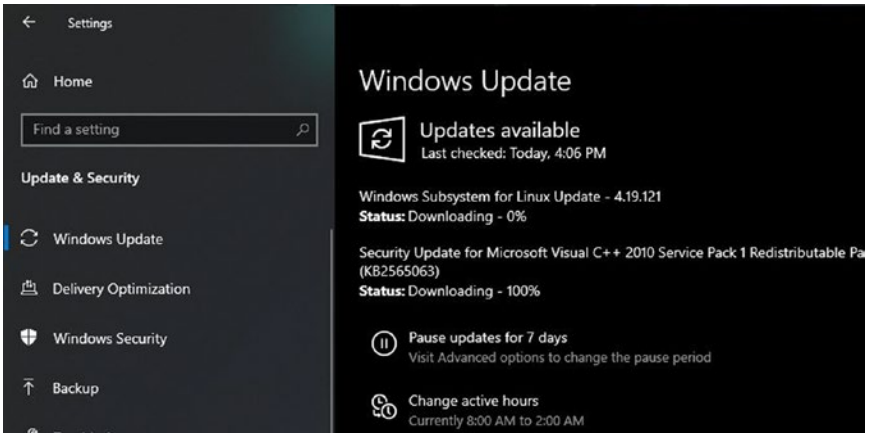

Figure 1-6. *Screenshot of Windows Update in Windows 10 Settings*

Some users will need to manually install the WSL 2 kernel (Figure 1-7) from an installer when upgrading from previous version of Windows 10 by downloading it from `https://aka.ms/wsl2kernel`. A version of the WSL 2 kernel for ARM64 devices is also available.

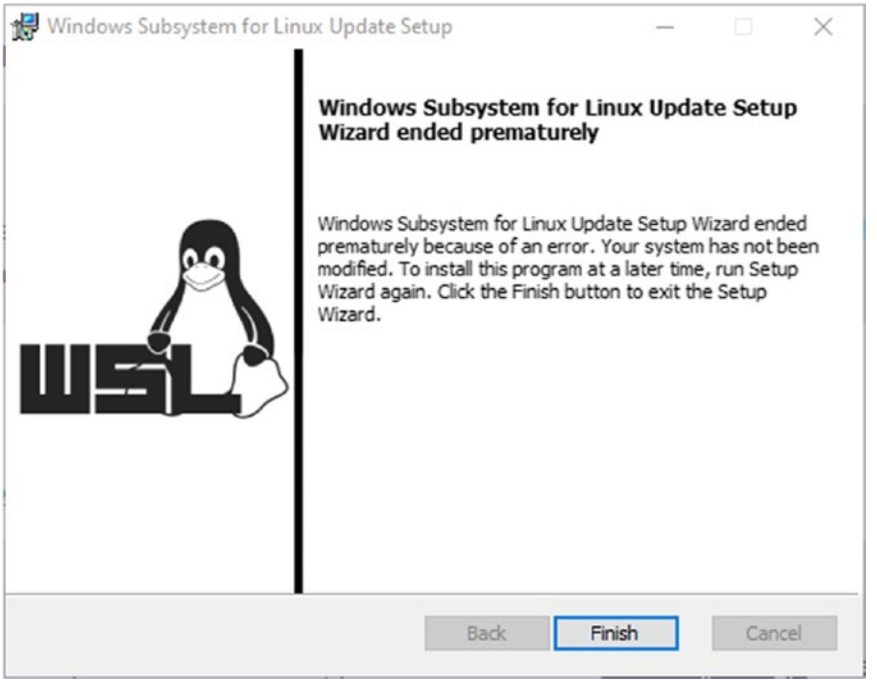

Figure 1-7. *Screenshot of Windows Subsystem for Linux 2 kernel installer*

Building your own kernel for WSL is detailed in Chapter 8 "Going Further with WSL2".

WSL 1 vs. WSL 2

Availability

WSL 1 is currently available on more versions of Windows 10, including Windows Server 2019. WSL 1 is your only choice on Windows 10 versions 1709 to 1809.

WSL 2 is available on the Windows 10 May 2019 update, known as version 1903, with recent updates applied, specifically build 18362.1049 and higher. If you are running this version of Windows 10 or higher, I strongly recommend you try WSL 2.

WSL 2 was originally launched on Windows 10 May 2020 Update, version 2004, but was then backported to Windows 10 May 2019, version 1903, later in 2020.

Why You Would Choose WSL 1

WSL 1 has lower system overhead than WSL 1. It may make a better choice on a lower-resource machine, such as a Surface Go, on which power usage takes precedence over performance.

WSL 1 also has simplified networking compared to WSL 2. WSL 1 simply adopts the networking configuration of your host Windows machines, whereas WSL 2 must implement an entire NAT network inside the virtual environment which can complicate some workflows.

Note NAT stands for Network Address Translation. Each WSL 2 distro has its own individual IP address that is only accessible from the Windows device on which it is running. Windows translates outgoing connections to look like they are coming from the Windows device.

WSL 1 has compatibility with many Linux applications. If you are limited to WSL 1, you should not be dismayed. It is still a viable option for some users. Note though it is not expected to get new features.

Why You Would Choose WSL 2

You should choose WSL 2 if your application requires it, such as Docker or microk8s. WSL 2 is also more performant than WSL 1, by a significant factor. If you want to maximize WSL performance, then WSL 2 is a breakthrough for your workflow.

WSL 2 has a more complicated networking setup than WSL 1. WSL 2 implements an entire NAT network inside the virtual environment. Extra steps are required to connect to WSL 2 services from other devices, something to keep in mind while planning your WSL deployment or migration from WSL 1 to WSL 2.

WSL 2 also allows you to compile and run your own Linux kernel if you need kernel features not provided by the default Microsoft WSL 2 kernel.

The Future of WSL

2020 brought announcements of more architectural changes to WSL 2. Chief among them is support for GPU computing tasks, which are accelerated by DirectX, Direct3D, and DirectML. Support for OpenGL, OpenCL, and Vulkan is expected later. These new GPU compute features required updates to the WSL 2 kernel and how GPU devices are handled by WSL 2. These represent the next major architectural advances since the introduction of WSL 2.

GPU Compute

Windows 10 builds with GPU compute support were released to the Insider Dev Channel in June 2020 and are expected to be a feature in a release of Windows 10 in 2021.

The new GPU compute functionality is based on a para-virtualized GPU in the WSL 2 environment. GPU acceleration will allow a whole new category of GPU-driven compute, artificial intelligence, machine learning, and statistical analysis workloads on WSL.

A GPU-accelerated workflow setup is detailed in Chapter 10, “Using WSL for Enterprise Development.”