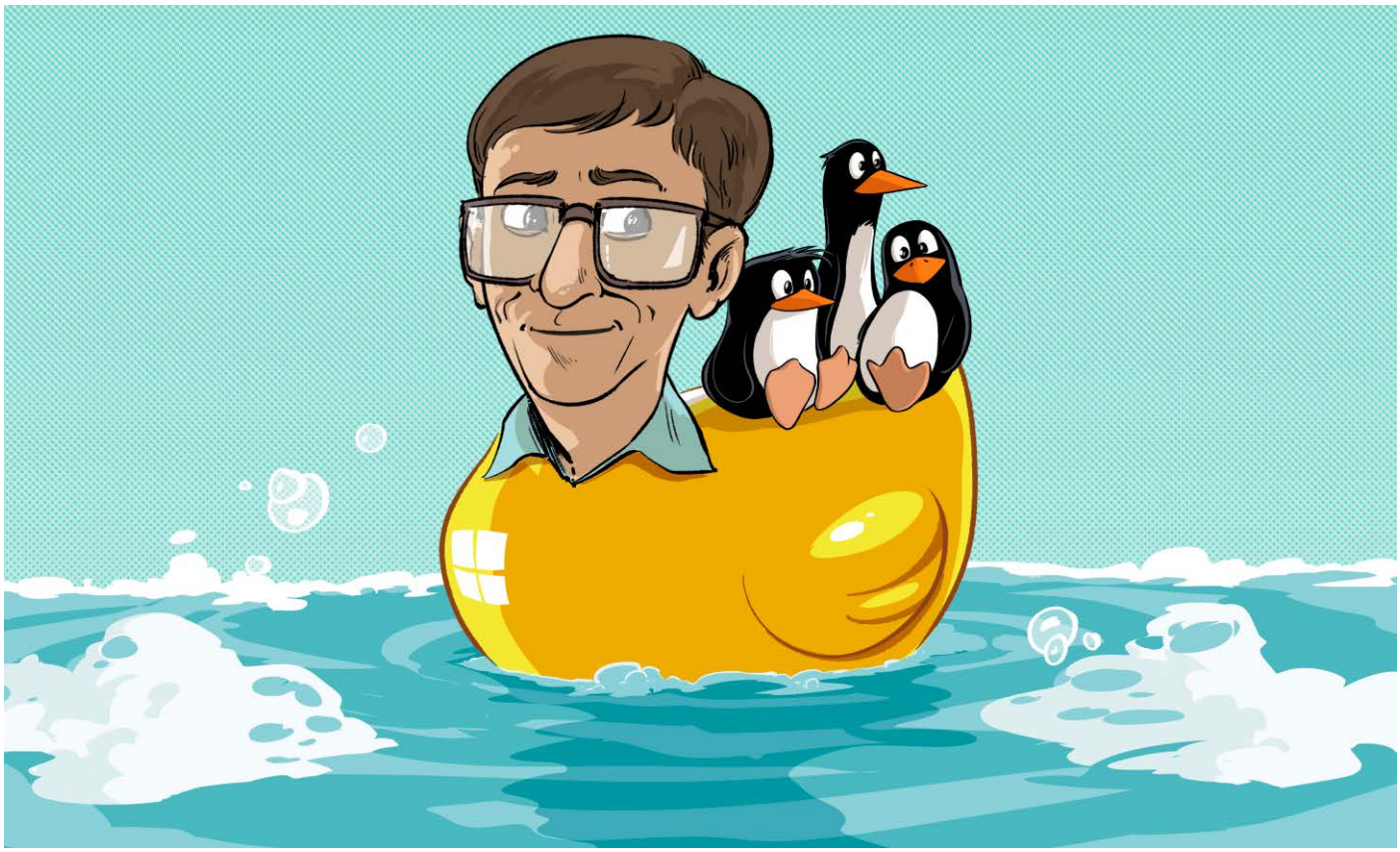


Bild: Albert Hulm

Windux

Entdeckungsreise durchs Windows Subsystem für Linux

Unter Windows 11 kann das Windows Subsystem für Linux (WSL) sogar grafische Linux-Anwendungen auf dem Desktop darstellen. Ein gründlicher Blick auf die Funktionsweise zeigt nicht nur, was das zuständige Subsystem in den verschiedenen Windows-Versionen leistet, sondern auch einige spannende Mechanismen, die in allen modernen Windows-Versionen stecken.

Von Peter Siering

Während der Weiterentwicklung von Windows 10 schlug die Neuigkeit, dass das Windows Subsystem für Linux (WSL) im Microsoft-Betriebssystem normale Linux-Programme ausführen wird, hohe Wellen. Hatten die Microsoft-Granden vor 20 Jahren noch gewettert, Linux sei ein Krebsgeschwür, bauten die Redmonder Entwickler es nun direkt ins Betriebssystem ein. Seit den ersten Gehversuchen in den Insider Builds von Windows 10 im Jahr 2016 bis hin zur aktuellen Fassung in Windows 11 haben die Entwickler die Funktionsweise stark verändert und greifen tief in die Trickkiste der Betriebssystemtechnik.

c't Hardcore

Geschichte

Vorn angefangen: Für die erste reguläre mit Windows 10 veröffentlichte Fassung des WSL, die heute Version 1 heißt, hatten die Entwickler versucht, die Linux-Systemaufrufe auf Windows-Funktionen abzubilden. Beim Ausführen von Linux-Programmen, sogenannten ELF-Binaries, merkten die nicht, dass statt eines Linux-Kernels eine Software-Schicht auf dem Windows-Kernel die Systemaufrufe ausführte. Die Entwickler hatten zwar nicht 100 Prozent aller Aufrufe implementiert, aber genug zum Ausführen einer Linux-Umgebung, wie sie zum Beispiel Ubuntu bereitstellt – Umgebung meinte zunächst eine textbasierte Unix-Shell mit all ihren Begleitern wie `grep`, `awk` und Kollegen.

Mitte 2019 erschien dann Version 2 von WSL, zunächst wieder im Rahmen der Insider Previews. Anstatt Windows die Systemaufrufe nachbilden zu lassen, spannten die Entwickler jetzt eine Mini-

mal-VM ein, die einen Linux-Kernel ausführt. Für die kompakte VM braucht Microsoft nicht die vollständige eigene Virtualisierungskomponente Hyper-V, sondern nur eine abgespeckte Form derselben. Diesen „Host Compute Service“ (HCS) verwendet Windows auch an verschiedenen anderen Stellen, um Sandboxen zu realisieren.

WSL1 und WSL2 schließen einander nicht aus, sondern können nebeneinander laufen. Der Anwender entscheidet, unter welcher Variante er eine Linux-Umgebung betreiben möchte. Eine Windows-Installation kann die Linux-Umgebungen per App aus dem Microsoft-Store ziehen oder das ganz ohne Geklicke auch auf der Kommandozeile erledigen. Letztere bietet sich ohnehin an, denn Dreh- und Angelpunkt für Aufruf und Konfiguration der WSL-Umgebungen ist das Programm `wsl` auf der Kommandozeile, das zumeist Rechte eines Administrators erwartet.

Installation

Während der Entwicklung variierten die Tipps, wie die Installation von WSL(2) zu bewerkstelligen sei. Jetzt ist klar: Wenn Sie eine aktuelle Windows-Version am Start haben, ist dafür die Kommandozeile am bequemsten. Öffnen Sie als Admin eine Eingabeaufforderung oder ein Windows-Terminal und geben Sie dort ein:

```
wsl --install --distribution ubuntu
```

Der Befehl installiert nicht nur WSL, sondern auch gleich eine Ubuntu-Umgebung. Windows 11 braucht die Option für die Distribution nicht, es ergänzt von sich aus gleich Ubuntu. Unter Windows 11 holt der Befehl auch gleich die Komponenten, die notwendig sind, um grafische Linux-Programme auszuführen, von Microsoft WSLg getauft. Für Windows 10 wird Microsoft diese Funktion aller Voraussicht nach nicht anbieten. Sie wurde zwar in den Insider Preview unter Windows 10 getestet, bleibt aber Windows 11 vorbehalten.

Im Store bietet Microsoft WSL neuerdings auch an, bei Redaktionsschluss in Form einer Preview. Langfristig soll das der empfohlene Installationsweg werden, steht in Microsofts Entwickler-Blog. Das Argument dafür: So wäre es leichter, WSL außer der Reihe zu aktualisieren, ohne dafür auf eine neue Windows-Version umsteigen zu müssen. Der auf der Kommandozeile dafür angebotene Befehl `wsl --update` aktualisiert nur den Kernel. Damit letzteres automatisch geschieht, muss man in der Konfiguration für Windows-Update auch das Aktualisieren anderer Microsoft-Software gestatten.

Was die WSL-App im Store im Unterschied zu `wsl --install` derzeit nicht leistet: Sie richtet nicht das zusätzlich nötige Windows-Feature, die „Virtual Machine Platform“ ein. Das muss der Nutzer zuvor von Hand erledigen. Das geht per `dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all` aus einer Kommandozeile mit Admin-Rechten oder über die Windows-Feature-Liste, die man über eine Suche im Startmenü nach „Feature“ findet (oder auch über die betagte und in Windows 11 noch immer enthaltene Systemsteuerung). Wie gesagt: `wsl --install` spart diese Handgriffe.

Erste Schritte

Sobald WSL eingerichtet ist, der PC eventuell neu gestartet und eine Linux-Umgebung hinzugefügt wurde, lässt sie sich durch Eingabe ihres Namens in die Startmenüsuche starten. Alternativ erledigt das auch der parameterlose Aufruf von `wsl` für die als Standard gesetzte Linux-Umgebung. Starten heißt, dass Windows eine Shell zeigt, also die Linux-Kommandozeile. Das dauert einige Sekunden, wenn zuvor keine Linux-Umgebung lief. Je nachdem, wie Sie diese aufrufen, stellt Windows sie in einer Windows-Konsole oder im neuen Windows-Terminal dar (letzteres empfiehlt sich, weil es komfortabler ist).

In der Shell können Sie beliebige Linux-Kommandos starten, etwa mit `ls` eine Liste der Dateien ausgeben lassen, die im aktuellen Verzeichnis liegen. Wenn Ihr PC mit Windows 11 läuft und grafische Linux-Programme in der Umgebung installiert sind, können Sie diese wie normale Windows-Programme benutzen. Windows stellt diese dann auf dem Desktop als Fenster neben den Windows-Anwendungen dar. WSL reiht sogar solche grafischen Anwendungen automatisch ins Windows-11-Startmenü ein. Wenn Sie Firefox in der Ubuntu-Umgebung mit

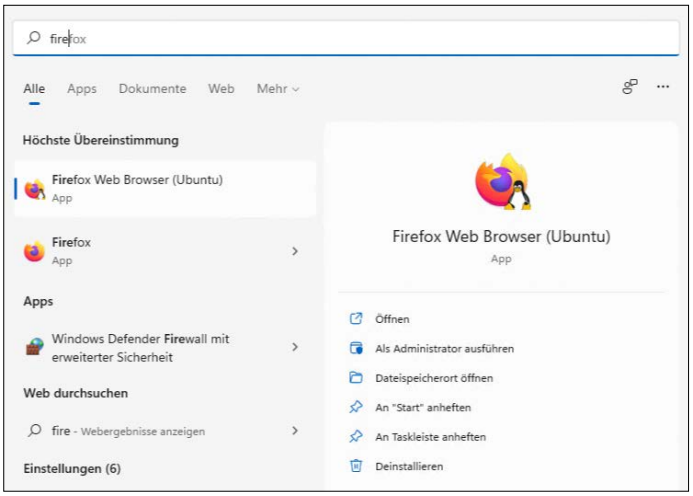
```
sudo apt-get update
sudo apt-get install firefox
```

nachinstalliert haben, würde Windows den als „Firefox (Ubuntu)“ bei einer Startmenüsuche anbieten. Beim Aufruf der vorgenannten Befehle fragt die Ubuntu-Umgebung das Passwort ab, das Sie beim Einrichten von WSL einzugeben aufgefordert worden sind. Den Benutzernamen, den Sie dabei festgelegt haben, verwendet WSL als Ihre Linux-Identität. Die WSL-Linux-Umgebungen gehören immer dem angemeldeten Windows-Nutzer. Das unter Linux angelegte Konto und Passwort sind nicht weiter mit dem Windows-Konto verknüpft.

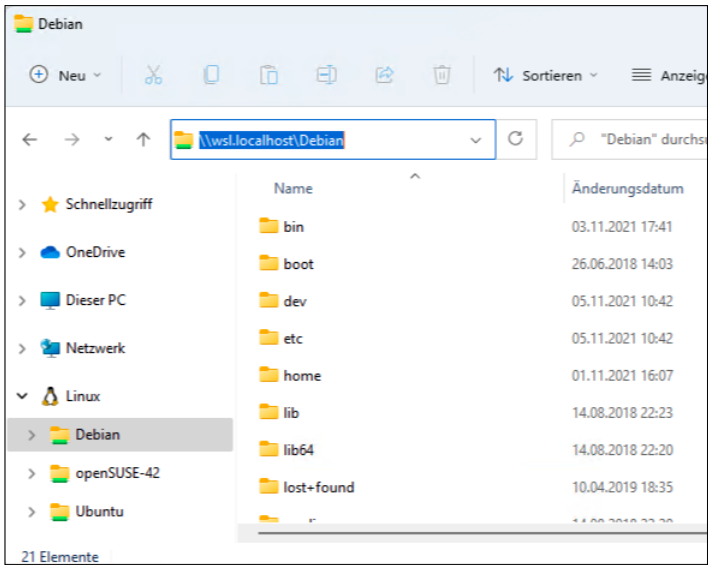
In der Linux-Shell können Sie aber nicht nur Linux-Programme starten, sondern auch Windows-Programme. Es ge-

c't kompakt

- WSL führt unter Windows Linux-Binärdateien direkt aus.
- Seit Windows 11 stellt WSL die Fenster grafischer Linux-Awendungen wie die von Windows Anwendungen auf dem Desktop dar.
- Als Komponente zeigt WSL auf, welche technischen Möglichkeiten heute in Windows stecken.



Windows 11 findet in einer WSL-Umgebung installierte grafische Linux-Anwendungen und trägt sie ins Startmenü ein. Für die Darstellung von deren Fenstern verwendet Microsoft die Funktionen des Remote Desktop.



Wer hätte gedacht, dass im Explorer mal Pinguine herumspringen: Die Dateien der Linux-Umgebungen sind per Klick erreichbar. Bei WSL2 liegen sie in VHDX-Dateien, also virtuellen Platten, wie sie Hyper-V verwendet.

nügt, den Namen der Programmdatei einzugeben, etwa `notepad.exe` oder `powershell.exe`. Microsoft selbst zeigt gern den Aufruf von Visual Studio Code, das unter Windows installiert ist. Der gelingt dabei interessanterweise durch Eingabe von `code` – damit das klappt, hinterlegt Microsoft bei der Installation der Windows-App im Suchpfad ein Shellskript, das aus der WSL-Shell das eigentliche Windows-Binary `code.exe` aufruft.

Weitere Distributionen lassen sich per `wsl --install --distribution` mit hinten angestelltem Distributionsnamen hinzufügen. Welche Microsoft anbietet, verrät der Aufruf `wsl --install --online`. Aber nicht nur diese Umgebungen stehen zur Verfügung. Microsoft hat Anleitungen veröffentlicht, wie Linuxe zu paketieren sind, damit sie sich in WSL integrieren. Es läuft letztlich darauf hinaus, ein Tar-Archiv zu bauen und eine EXE-Datei bereitzustellen, die diese auf Zuruf bearbeitet. Über ct.de/y2ue finden Sie viele weitere solcher Distributionen zum Download.

Technik dahinter

Die in WSL „ausgeführten“ Linux-Umgebungen entsprechen in einigen Punkten nicht einer regulären Distribution, wie sie auf einem PC oder in einer virtuellen Maschine (VM) läuft. Es fehlen die typischen Funktionen zum Starten von Diensten, zur Anmeldung und die Software zum Bereitstellen einer grafischen Bedienoberfläche mit Desktop und so weiter. Aber die Umgebungen enthalten alle zur Distribution gehörenden Bibliotheken und Konfigurationsdateien, sodass Programme keinen Unterschied bemerken, solange sie nicht auf weitere Dienste angewiesen sind.

Wie heute zumeist in Containern führt die Umgebung typischerweise ein Programm aus. Sie kann aber durchaus auch mehrere Programme starten. Da jedoch die Infrastruktur zum automatischen Starten von Diensten fehlt, sind das eher Eintagsfliegen. Sobald sich der Nutzer abmeldet, die Umgebung oder Windows beendet, stoppen alle darin gestarteten Linux-Prozesse. Letztlich haben die Microsoft-Entwickler alles darauf abgestimmt, dass Linux-Binärdateien interaktiv nutzbar sind – den dauerhaften Betrieb von Serverdiensten hatten sie nicht im Auge.

WSL1 hat Microsoft als Dienst in Windows verankert, um Linux-Binärdateien auszuführen. Bei WSL2 führt eine leichtgewichtige virtuelle Maschine (VM) einen von Microsoft angepassten Linux-Kernel aus. Unter diesem Kernel in dieser VM betreibt WSL2 alle Linux-Umgebungen eines Nutzers. Es nutzt Namespaces, um die Distributionen in der VM voneinander zu isolieren. Fürs Ausführen von grafischen Linux-Anwendungen (WSLg) stellt Microsoft in Windows 11 der vom Nutzer eingerichteten Linux- eine spezielle System-Umgebung zur Seite, die deren Bildschirmausgaben so umlenkt, dass sich deren Fenster per RDP auf dem Desktop neben regulären Windows-Fenstern darstellen lassen.

Das Gebilde kann man recht gut erforschen. Dass eine leichtgewichtige VM beteiligt ist, sieht man zwar nicht in den Hyper-V-Verwaltungswerkzeugen, aber sehr wohl auf der Kommandozeile mit einem Aufruf von `hcsdiag list` (dafür sind Admin-Rechte erforderlich); so lassen sich übrigens auch andere solcher Mini-VMs sichtbar machen, die schon Windows 10 zum Beispiel für Sicherheitsfunktionen einbindet. Das Android Subsystem in Windows nutzt diese ebenso (mehr darüber in einer kommenden c’t und schon jetzt auf c’t 3003, siehe ct.de/y2ue). Auch für die virtuellen Hyper-V-Switches liefert ein solches Kommando (`hnsdiag`) Informationen.

Unter Windows 11 lässt sich die Forschungsarbeit noch weiter treiben: Dort kann man mit `wsl --system` sogar eine Shell innerhalb der von Microsoft für das Abfangen der grafischen Ausgaben konstruierten System-Umgebung erlangen. In dieser Umgebung steht mit yum sogar ein Paketmanager bereit, um für die Forschung weitere Software zu installieren. Solche Änderungen sind allerdings nur

So viel Linux steckt in jedem Windows

Nicht nur mit dem Ausführen von Linux-Anwendungen schickt Microsoft Windows auf Tuchfühlung mit der Unix-Welt. Schon zu frühen Zeiten von Windows 10 begannen die Entwickler, typische Werkzeuge aus der Unix-Kommandozeile einzubauen: So finden Windows-Nutzer dort das Programm **Curl** vor, mit dem sich automatisiert Dateien herunterladen, aber auch komplexere Web-Zugriffe in Skripten realisieren lassen. Außerdem kann Windows schon länger die unter Unix gebräuchlichen **Tar**-Archive lesen und schreiben – das allerdings „nur“ auf der Kommandozeile.

Einen großen Gefallen haben die Entwickler der IT-Welt getan, indem Sie Windows auch einen **OpenSSH**-Client und -Server spendiert haben. Der ist so nah am in der Open-Source-Welt verbreiteten Original, dass sich die für die Anmeldung nötigen Schlüssel problemlos austauschen lassen. Das war nicht immer so: Das oft auch heute noch von vielen unnötigerweise empfohlene Putty generierte diese in einer speziellen Form, sodass sie (Open-)SSH-Server erst nach Konvertierung schlucken. Diese Pfriemelei hat mit den in Windows eingebauten SSH-Komponenten gottlob ein Ende.

temporär und schlagen sich nicht in der beim Start bemühten VHDX-Datei der System-Umgebung nieder.

Dateiaustausch

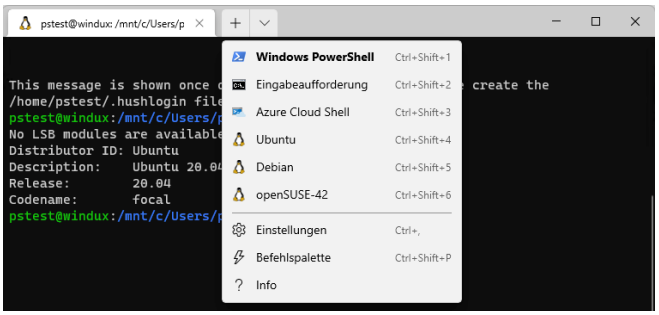
Ein wesentlicher Unterschied zwischen WSL1 und 2, den Microsoft auch sehr deutlich dokumentiert hat, liegt in der Art und Weise, wie „weltübergreifende“ Dateizugriffe realisiert sind. WSL2 legt alle Linux-Dateien zusammen mit denen der Distribution in einer VHDX-Datei im Benutzerprofilverzeichnis ab (unter c:\Users\<Name>). Die Zugriffe auf diesen Datenbestand aus WSL heraus erfolgen rasend schnell. Um die Windows-eigenen Laufwerke unter Linux zugänglich zu machen, bindet WSL2 sie in den Linux-Dateibaum unter „/mnt“ ein.

Zum Einbinden nutzt es eine spezielle Zugriffstechnik, um sicherzustellen, dass konkurrierende Zugriffe von Windows und Linux nicht das Dateisystem beschädigen (ein gleichzeitiges Einbinden eines Gerätes unter beiden Systemen würde das im Zweifel provozieren): das für das Experimentierbetriebssystem Plan 9 entworfene 9P. WSL2 selbst agiert dabei als Server, Windows als Client. Benchmarks zeigen, dass Dateizugriffe auf die so in Linux erreichbaren Dateien eines NTFS-Dateisystems viel langsamer erfolgen. Je mehr kleine Dateien dabei bearbeitet werden, desto stärker soll sich das auswirken.

Die Empfehlungen, die man daraus ableiten kann, sind sehr eindeutig: Wenn gemeinsame Zugriffe von Windows und Linux auf einen großen Bestand kleiner Dateien gefragt sind, dann empfiehlt sich WSL1. Hauptsächlich unter Windows bearbeitete Bestände sollten mit WSL2 auf einem NTFS-Datenträger liegen, im Wesentlichen unter Linux bearbeitete Dateien idealerweise im Linux-Dateisystem (nicht unter /mnt), also in der VHDX-Datei. Selbst wenn Windows diese Dateien nicht standardmäßig mountet, lässt sich ein Zugriff realisieren. Auch hier kommt 9P zum Einsatz: Microsoft hat Visual Studio Code um ein Modul ergänzt, das so Dateien im Linux-Dateisystem erreichen kann.

Netzwerk

Die Netzwerkanbindung der Linux-Umgebungen unterscheidet sich je nach WSL-Version. WSL1 klemmt diese Umgebungen hinter die Netzwerkkarte des PCs, das heißt, Linux und Windows teilen sich die IPv4- und IPv6-Adressen. Entsprechend



Das Windows Terminal trägt viel zum Charme von WSL bei. Die Linux-Umgebungen tauchen nach dem Hinzufügen automatisch dort auf.

ist die in den Linux-Umgebungen angezeigte Hardware-Ethernet-Adresse (MAC) mit der in Windows identisch. Dadurch sind innerhalb von WSL ausgeführte Dienste erreichbar, beispielsweise ein Web-Server. Öffnet eine Linux-Anwendung einen Port, fragt Windows gleich nach, ob es einen Firewall-Port öffnen soll.

WSL2 verpasst ebenfalls allen laufenden Umgebungen eine identische IPv4-Adresse, leitet die aber nicht von der des PCs ab. Stattdessen entnimmt es sie einem Adresspool eines virtuellen Switches, der „WSL“ heißt und bei Nachinstallation der Hyper-V-Verwaltungswerkzeuge und -Dienste auch sichtbar ist, wenn WSL2-Umgebungen laufen. Alle WSL2-Umgebungen nutzen dann gemeinsam eine IP-Adresse aus einem privaten IPv4-Netz. IPv6-Adressen erhalten sie nicht. Auf dem virtuellen Switch ist die Windows-Komponente zum Teilen einer Internet-Verbindung (Internet Connection Sharing, ICS) aktiv.

Netzwerkzugriffe von außen auf Dienste, die in einer WSL2-Umgebung laufen, sind dadurch schwieriger möglich. Im Bug-Tracker zu WSL gibt es dazu lang anhaltende Diskussionen, die allerhand Probleme aufzeigen (siehe ct.de/y2ue): Da eine Umgebung mit jedem Windows- und WSL-Neustart eine andere IP-Adresse zugeteilt bekommt, genügen statische Portweiterleitungen via `netsh` (mit `netsh interface portproxy add`) und Ausnahmen in der Windows-Firewall nicht, um Dienste in den Linux-Umgebungen dauerhaft zugänglich zu machen. Doch wie gesagt war dies auch nicht Microsofts Ziel.

Wem es nützt

WSL spielt da seine Vorzüge aus, wo Windows-Nutzer viel mit Linux-Umgebungen in Berührung kommen. Besonders gilt das für Entwickler, die Visual Studio Code benutzen, um zum Beispiel Docker-Container zu bauen. Sofern die nicht allzu umfangreiche Abhängigkeiten enthalten, genügt ein Laptop, um das Ganze zum Testen nötige Labor mitzuschleppen. Nach einer Zeit merkt man gar nicht mal mehr, auf welchem Betriebssystem man arbeitet.

Das unterscheidet den WSL-Ansatz von herkömmlicher Technik zu Virtualisierung. Theoretisch lässt sich all das auch mit virtuellen Maschinen realisieren, aber der Aufwand ist deutlich größer. Man muss die VMs einrichten, sich Gedanken über die zu verteilenden Ressourcen machen und selbst aufräumen. Mit Windows-Bordmitteln gelingt das nur mit der Pro-Version. WSL steht auch in Home zur Verfügung und sogar in der ARM-Variante (vorausgesetzt, die Hardware erlaubt Virtualisierung, was nicht für jedes ARM-Gerät gilt).

Unterm Strich ist mit WSL die Schwelle, eine Linux-Umgebung zu installieren, wieder zu entsorgen (mit `wsl --unregister`, was alle Daten löscht) und neu einzurichten, deutlich niedriger. Zum Charme trägt das Windows-Terminal bei: Der Ersatz für die stumpfen Konsolenfenster, in denen Windows bisher Kommandozeilenwerkzeug bereitgestellt hat, glänzt mit skalierbarer, bunter und Tab-fähiger Oberfläche. Obendrein integriert es die verschiedenen Linux-Umgebungen schon bei der Installation, sodass PowerShell, klassische Eingabeaufforderung und Bash immer nur einen Klick entfernt sind.

Im WSL-Kosmos ist das Frickelfieber ausgebrochen. Auf GitHub sprießen Projekte aus dem Boden: grafische Programme zum Verwalten der Linux-Umgebungen, Netzwerkkonfigurationshilfen und Ergänzungen zum dauerhaften Betrieb von Diensten. Obendrein gibt es zur Orientierung Sammlungen spannender WSL-Themen (einige Perlen haben wir unter ct.de/y2ue zusammen getragen). Das alles rückt sogar cronjobs in greifbare Nähe, die aus der Linux-Umgebung heraus Windows-Dateien bearbeiten. *(ps@ct.de)* c't

WSL-Zusätze, ausgewählte Bug-Reports, Architekturinfos, c't 3003: **ct.de/y2ue**