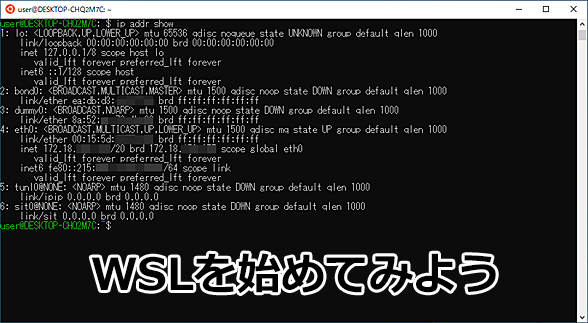
**Windows 10／11でLinux環境「WSL」を始める第一歩*これから始めるWSL*（1/3 ページ）**

Windows 10およびWindows 11上のLinux環境「WSL」をこれから始める読者のための、インストール方法やバックアップ方法をまとめて解説する。また、Windows 11上のWSLで新規サポートされた外部ドライブのマウント方法も合わせて紹介しよう。

2022年01月28日 05時00分 公開

[[塩田紳二](https://www.itmedia.co.jp/author/100454/)，著]

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic01.png)**WSLを始めよう**  
Linux環境「WSL」をこれから始める読者のための、インストール方法やバックアップ方法などをまとめて解説する。

1. **■記事内目次**

* [WSLとは](https://atmarkit.itmedia.co.jp/ait/articles/2201/28/news029.html#whatis)
* [WSLを設定する際の注意](https://atmarkit.itmedia.co.jp/ait/articles/2201/28/news029.html#attention)
* [WSL 2のインストールと設定の基本](https://atmarkit.itmedia.co.jp/ait/articles/2201/28/news029.html#install)
* [WSL設定の基本となるwsl.exeコマンド](https://atmarkit.itmedia.co.jp/ait/articles/2201/28/news029_2.html#wslexe)
* [WSLディストリビューションのバックアップ／リストア／環境復元](https://atmarkit.itmedia.co.jp/ait/articles/2201/28/news029_3.html#recover)
* [Windows 11で外部ドライブをマウントする](https://atmarkit.itmedia.co.jp/ait/articles/2201/28/news029_3.html#mount)

　「Windows 10」や「Windows 11」上でLinuxの実行環境を構築可能とする「Windows Subsystem for Linux（WSL）」が着々と更新されている。WSLは、Windows 10 Fall Creators Update（バージョン1709）から正式版が利用可能となり、その後、May 2020 Update（バージョン2004）では、Linuxとの互換性を高めた「WSL 2」が提供された（WSL 2については、Windows 10 The Latest「[前バージョンから大幅に性能向上した新Linux環境『WSL 2』の実力を探る](https://atmarkit.itmedia.co.jp/ait/articles/2005/15/news020.html" \t "_blank)」を参照のこと）。

　当初は、Windows 10の［コントロールパネル］で、［プログラムと機能］画面の［Windowsの機能の有効化または無効化］を実行してLinux用サブシステムのインストールを行い、その後、LinuxディストリビューションをMicrosoft Storeからインストールするといった少々面倒な手順が必要であった。

　Windows 10 May 2020 Update（バージョン2004）からは、Linux用サブシステムやLinuxディストリビューションがコマンドラインでインストール可能になり、インストールが容易になっている。

　そこで本稿では、これからWSLを始める読者のための、WSLのインストールからバックアップ方法などをまとめて解説する。

1. **WSLとは**

　WSLとは、Windows OS上で実行されるLinux環境のことである。ここでは、Windows OSに搭載されたWSL技術全般を指す用語として使う。以下ではWSLにまつわる基礎的な事柄を簡単に説明する。

1. **「WSL 1」「WSL 2」とは**

　WSLには実装方式によって、最初にリリースされたバージョン（ここでは区別のため「WSL 1」とする）と、その改良版である「WSL 2」に大別できる。WSL 2はLinuxカーネルと仮想マシンを使う方式で、WSL 1に比べ、Linuxとしての互換性が高くなっている。例えば、WSL 1ではDockerが動作しなかったが、WSL 2ではLinuxネイティブ版のDockerがそのまま動作できるようになっている。

[[コンピュータ が含まれている画像

自動的に生成された説明](https://dlv.itmedia.jp/rd/v1/j/on/c/chsm=90,a2c80ef7/2075501498/ISALM/343155/300966444.510324499.620249183/730319623/109319/2075501497:ISALM:*/l/sp=0;/g/B:aHR0cHM6Ly9hdG1hcmtpdC5pdG1lZGlhLmNvLmpwL2FpdC9hcnRpY2xlcy8yMjAxLzEyL25ld3MwMDUuaHRtbA)](https://dlv.itmedia.jp/rd/v1/j/on/c/chsm=90,a2c80ef7/2075501498/ISALM/343155/300966444.510324499.620249183/730319623/109319/2075501497:ISALM:*/l/sp=0;/g/B:aHR0cHM6Ly9hdG1hcmtpdC5pdG1lZGlhLmNvLmpwL2FpdC9hcnRpY2xlcy8yMjAxLzEyL25ld3MwMDUuaHRtbA)

**[オンプレみたいにストレージ管理できるコンテナが欲しい！](https://dlv.itmedia.jp/rd/v1/j/on/c/chsm=90,a2c80ef7/2075501498/ISALM/343155/300966444.510324499.620249183/730319623/109319/2075501497:ISALM:*/l/sp=0;/g/B:aHR0cHM6Ly9hdG1hcmtpdC5pdG1lZGlhLmNvLmpwL2FpdC9hcnRpY2xlcy8yMjAxLzEyL25ld3MwMDUuaHRtbA)**

　WSL 2は、専用の仮想マシンの上で動作し、利用にあたっては、PCに仮想マシン支援機能があり、BIOSなどで有効化されている必要がある。ただし、その手順はPCの機種によって異なるため、ここでは解説しない。PCのマニュアルなどを参照してほしい。

　この専用仮想マシンをMicrosoftは、「軽量ユーティリティーVM」と呼んでいる。これは、Windows OSでサポートする仮想マシン環境「Hyper-V」とは異なるもので、Hyper-Vとは異なり、Windows 10／11のHomeエディションでも動作できる。

　WSL 1とWSL 2の利用上の相違点については、[この後に説明する](https://atmarkit.itmedia.co.jp/ait/articles/2201/28/news029.html" \l "attention)。

1. **「WSLディストリビューション」とは**

　このWSLで利用するLinuxディストリビューションを特に「WSLディストリビューション」と表記する。基本的にWSLディストリビューションは、Linuxのディストリビューションと同じだが、WSL 1／WSL 2内で動作させるためにカスタマイズされており、Microsoft Store経由で配布される。

　WSLディストリビューションは、それぞれ固有の「ディストリビューション名」を持つ。これは、WSLで複数のディストリビューションを扱うためのものだ。このディストリビューション名は、Microsoft Storeで配布されているWSLディストリビューションの名前に近いものになっているが、全く同じではない。

　コマンドライン中で引数として設定しやすいように空白が入らないようにし、かつ比較的短い記述ができるようにしてある。例えば、Microsoft Storeにある「[Ubuntu 18.04 LTS](https://www.microsoft.com/ja-jp/p/ubuntu-1804-lts/9n9tngvndl3q?activetab=pivot:overviewtab" \t "_blank)」には、「Ubuntu-18.04」というディストリビューション名が付けられている。

1. **「WSL」に対する「Win32」とは**

　WSLを利用する場合、その内部のLinux環境を「WSL」「WSL側」、WSLの外になるWindows OS環境を「Win32」や「Win32側」と表記されることが多い。例えば、WSL内からWindows OSのコマンドを起動する機能は「Win32相互運用性」と呼ばれている。

　ここでは、この表記に従い、必要に応じて、2つの環境を呼び分ける。この場合、Win32側のコマンドであることを明確にするため、「cmd.exe」などと必ず実行ファイル名を拡張子まで指定する。Windows OS側にも、「curl.exe」といったLinux由来のコマンドがあり、WSL側で単に「curl」とするとLinux側のcurlコマンドが実行されてしまうからだ。これを「curl.exe」とすれば、Win32側の同名のコマンドが実行される。

　なお、後述するようにWin32側のコマンドを実行ファイル名だけで起動するためには、PATH環境変数の設定が必要になる。標準では、Win32側のPATH環境変数に設定されているパスが全てWSL側にコピーされるため、Win32側と同じようにコマンドが実行できる（PATH環境変数の設定方法については、Tech TIPS「[【Windows 11対応】Path環境変数を設定／編集して、独自のコマンドを実行可能にする](https://atmarkit.itmedia.co.jp/ait/articles/1805/11/news035.html" \t "_blank)」参照のこと）。

1. **WSLを設定する際の注意**

　WSLの挙動などに関しては、Win32側にある「%userprofile%\.wslconfig」ファイルと、WSL側にある「/etc/wsl.conf」を使う。また、WSL自体やWSLディストリビューションのインストールは、「wsl.exe」というコマンドで行う。

　環境設定の問題もあるため、ここでは全てのコマンドをWindows Terminalで行う前提で説明する。他の端末エミュレーターなどを利用している場合には、その設定に注意してほしい。

　ここではWSLディストリビューションとして、Ubuntu-18.04を使っている。他のディストリビューションでは、WSL側のコマンドなどに一部違いが出る可能性がある。その場合にはインストールしているWSLディストリビューションに合わせて適宜読み替えていただきたい。

　Linuxでは、基本的に大文字／小文字が区別される点にも注意してほしい。文中に示す実行例などで、オプションの大文字／小文字による挙動の違いなどが起きるため、WSL側では、大文字／小文字を正確に入力する必要がある。

1. **WSL 1とWSL 2の違い**

　現在のWindows 10／11では、WSLディストリビューションを「WSL 1」または「WSL 2」として設定できる。インストールなどは同一で、後からWSL 1とWSL 2を変換することも可能だ。

　基本的には、WSL 2の方が対応できるソフトウェアも多く、より本物のLinuxに近い。だが、WSL 1には、Win32側のファイルアクセスが高速であること、Win32側と同じIPアドレスを利用するというメリットがある。また、仮想マシンを使わない関係もあり、メモリ割り当てに柔軟性があり、システム負荷も低い。

　そのため、現在でも利用方法によっては、あえてWSL 1を選択するメリットもある。あまり本格的に使わず、まずはLinuxを学んでみたい程度のことであれば、リソース割り当てなどに柔軟性のあるWSL 1を選択してもよい。

　あるいはコマンドラインで、Linuxのbashを主に使いたいといったcmd.exeやPowerShellの代用程度ならば、必ずしもWSL 2にする理由がない。また、タブレットや古いハードウェアなど、システム性能に余裕がない場合にもWSL 1を選択した方がよい場合が多い。

　逆に、コンテナソフトウェアや、Linux GUIアプリ（Windows 11のみ対応）をどうしても使いたいというのであれば、WSL 2を選択する必要がある。なお、現在のWSLでは、ディストリビューションインストール時の標準バージョンは「2」となっており、特に指定しなければ、WSL 2が選択される。これは、以下のコマンドで変更することができる。

wsl.exe --set-default-version **＜バージョン＞**

**デフォルトのWSLバージョンを変更するコマンド**

　標準は「2」なので、**＜バージョン＞**に「1」を指定すれば、以後、インストールされるWSLディストリビューションは全てWSL 1としてインストールされる。

1. **WSL 2のインストールと設定の基本**

　Windows 10 May 2020 Update（バージョン2004）から、WSL 2の設定は「wsl.exe」コマンドの「--install」オプションを利用して行えるようになった。このため、基本的には、従来のPowerShellコマンドや［コントロールパネル］を使う方法はもう必要ない（Windows 10 November 2019 UpdateのEnterprise／Educationエディションを除く）。インストールは以下のコマンドで行う。

　まず、インストール可能なWSLディストリビューション名を以下のコマンドで確認する。好みや用途に合わせて、WSLディストリビューションを選択すればよい。

wsl.exe -l --online

**インストール可能なWSLディストリビューション名を調べるコマンド**

　次にWSLのインストールと同時にWSLディストリビューションのインストールを実行する。「-d」オプションでWSLディストリビューションを指定しない場合、「Ubuntu」が自動的にインストールされることになる。何をインストールしたのかを明確にするためにも、WSLのインストール時にWSLディストリビューションを指定した方がよいだろう。

wsl.exe --install ［-d **＜WSLディストリビューション名＞**］

**WSLとWSLディストリビューションをインストールするコマンド**

　上記のコマンドを入力し、WSLのインストールが完了したら、システムの再起動を行うこと。再起動後、自動的に指定したWSLディストリビューションのインストールが開始される。

1. **Microsoft StoreからWSLディストリビューションをインストールする**

　WSLディストリビューションは、Microsoft Storeアプリで「Linux」や「WSL」で検索してインストールすることも可能だ。

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic02.png)**Microsoft StoreからWSLディストリビューションをインストールする**  
Microsoft Storeアプリを起動し、「Linux」や「WSL」で検索を行うと、WSLディストリビューションが見つかる。ここからインストールしたいものをクリックして、［入手］ボタンをクリックすればよい。

　下表にMicrosoft Storeから入手可能なWSLディストリビューションを示す。これらは個人利用の場合には、無料で利用が可能なものだ。詳しいライセンス条件などは、Microsoft Storeの各情報ページで確認してほしい。

|  | **WSLでの名称** | **バージョン** | **標準シェル** | **パッケージ管理** | **man** | **--online** |
| --- | --- | --- | --- | --- | --- | --- |
| [**Alpine Linux**](https://www.alpinelinux.org/) | Alpine | 3.13.4 | /bin/ash | apk | オプション | × |
| [**Debian GNU/Linux**](https://www.debian.org/) | Debian | 11 | /bin/bash | apt/dpkg | オプション | ○ |
| [**Kali Linux**](https://www.kali.org/) | kali-linux | 2021.3 | /bin/bash | apt/dpkg | オプション | ○ |
| [**openSUSE Leap 42**](https://www.opensuse.org/) | openSUSE-42 | 15.2 | /bin/bash | zypper,yast/rpm | オプション | ○ |
| [**Ubuntu**](https://jp.ubuntu.com/) | Ubuntu | 20.04 | /bin/bash | apt/dpkg | 導入済み | ○ |
| [**Ubuntu 16.04 LTS**](https://jp.ubuntu.com/) | Ubuntu-16.04 | 16.04 | /bin/bash | apt/dpkg | 導入済み | ○ |
| [**Ubuntu 18.04 LTS**](https://jp.ubuntu.com/) | Ubuntu-18.04 | 18.04 | /bin/bash | apt/dpkg | 導入済み | ○ |
| [**Ubuntu 20.04 LTS**](https://jp.ubuntu.com/) | Ubuntu-20.04 | 20.04 | /bin/bash | apt/dpkg | 導入済み | ○ |
| **Microsoft Storeから入手可能な主なWSLディストリビューション** | | | | | | |

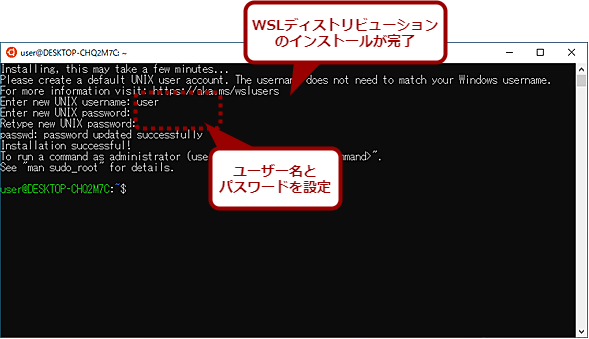
1. **Linuxのユーザー名とパスワードを設定する**

　WSLディストリビューションのインストールが行われるとき、Linuxのユーザー名とパスワードを指定する必要がある。WSLディストリビューションによって細かな違いがあるので注意してほしい（Linuxとして厳密な定義があるわけではなく、「adduser」などのディストリビューションが採用するユーザー追加プログラムの仕様に従う）。

　大まかなところでは、ユーザー名に利用できる文字は、半角アルファベット（ASCII文字）小文字と数字、「\_（アンダーバー）」と「-（ハイフン）」である。ユーザー名の先頭は、「\_（アンダーバー）」にすることは可能だが、数字や「-（ハイフン）」にすることはできない。Windows OS側と一致させる必要はなく、適当にユーザー名を決めてよい。

　ただし、初期状態で登録してあるユーザー名（root／bin／sysなど）は、使うことができない。確認するには、Linuxを起動して「/etc/passwd」ファイルを確認するしかない。初期設定では、WSLディストリビューションとして許可できないユーザー名は、指定してもエラーになるので、エラーになったらユーザー名を変更して改めて登録を行えばよい。

　また、WSLの場合、ログインは不要なため、パスワードは管理者権限でコマンドを実行するための「sudo」コマンドなどで必要になるだけだ。このため、セキュリティを強く考える必要はなく、打ちやすいものにしておいて構わない。WSLディストリビューションのインストールでは、ユーザー名に続いて、パスワードを2回入力すれば、処理は完了する。

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic03.png)**WSLディストリビューション上でユーザー名とパスワードを設定する**  
WSLディストリビューションのインストールが完了すると、ユーザー名とパスワードの設定が求められる。どちらもWindows OSとは異なるものでよい。WSLの場合、ログインは不要なため、パスワードは管理者権限でコマンドを実行するための「sudo」コマンドなどで必要になるだけだ。このため、セキュリティを強く考える必要はなく、打ちやすいものにしておいて構わない。

1. **WSLディストリビューションを起動する**

　WSLディストリビューションのインストールが完了したら、初期設定などを行う。そのためには、まず、［スタート］メニューでWSLディストリビューション名を選択するか、以下のコマンドを実行することでWSLディストリビューションを起動する。

wsl.exe -d **＜WSLディストリビューション名＞**

**WSLディストリビューションを起動するコマンド**

　終了時には、必ずコマンドラインなどから「exit」コマンドを使って抜けるようにすること。WSL自体がLinuxというOSを実行しており、WSLを実行中のコンソールウィンドウを閉じたり、wsl.exeの「--terminate」や「--shutdown」オプションで実行中のWSLを終了させたりすることは、Windows OSでいえば、いきなりコンセントを抜いて電源を切ることに相当するからだ。

　bashは、コマンドの実行履歴をファイルに記録しているなど、さまざまな処理をログアウト時に行うため、WSLのコマンドラインからは、正しい方法でログアウトしてからWSLを終了させることは心掛けた方がよい。Windows OSで簡易に起動できるとはいえ、WSLの中で動いているのはLinuxという普通のオペレーティングシステムであることは理解しておく必要がある。

[**2. wsl.exeコマンドの概要**](https://atmarkit.itmedia.co.jp/ait/articles/2201/28/news029_2.html)

page: [2](https://atmarkit.itmedia.co.jp/ait/articles/2201/28/news029_2.html)

1. **WSL設定の基本となるwsl.exeコマンド**

　WSL 2のインストールが終了すると、wsl.exeコマンドは、WSL環境の起動や制御などに利用できるようになる。なお、WSL初期に作られた「lxrun.exe」「wslconfig.exe」「bash.exe」に関しては、基本的な機能は全てwsl.exeに統合されており、これらのコマンドの利用は非推奨とされている。

　既に「lxrun.exe」に関しては搭載が廃止されている。「wslconfig.exe」「bash.exe」に関しては実行ファイルも残っているが、今後廃止される可能性もある。基本的には利用しないようにすること。特にスクリプトなどからの利用は行うべきではない。

　wsl.exeには、大きく「WSL実行機能（旧bash.exeの機能）」「WSL制御機能」「WSL管理機能（旧wslconfig.exeの機能）」の3種類の機能がある。各表の先頭にあるのがコマンドの指定方法で、その下は指定可能なオプションを示す。先頭の表にある「～オプション」の部分に後続の表のオプションを指定することができる。なお、表中の赤字は、Windows 11で有効になるものである。

**●WSL実行機能**

| **ロングオプション書式** | **ショート形式オプション** | **動作** |
| --- | --- | --- |
| **wsl.exe ［実行オプション］ --exec ＜コマンドライン＞** | -e | 既定のLinuxシェルを使用せずにコマンドラインを実行 |
| **wsl.exe ［実行オプション］ -- ＜コマンドライン＞** |  | 残りのコマンドラインをそのままWSL側に渡す |
| **WSL実行機能に関するWSLコマンド** | | |
| **実行オプション** | **ショート形式オプション** | **意味** |
| **--cd ＜ディレクトリ＞** |  | WSL側カレントディレクトリを指定 |
| **--distribution ＜ディストリビューション＞** | -d | 実行するディストリビューションを指定 |
| **--user ＜ユーザー名＞** | -u | コマンド実行ユーザーを指定 |
| **--system** |  | システムディストリビューションのシェルを起動 |
| **WSL実行機能にに関するWSLコマンドの実行オプション** | | |

**●WSL制御機能**

| **ロングオプション書式** | **ショート形式オプション** | **動作** |
| --- | --- | --- |
| **wsl.exe --help** |  | 使用法情報を表示します |
| **wsl.exe --install ［インストールオプション］** |  | WSLのインストール |
| **wsl.exe --mount ＜ディスク＞ ［マウントオプション］** |  | 外部ドライブのマウント |
| **wsl.exe --set-default-version ＜バージョン＞** |  | 新規ディストリビューションの既定のインストールバージョンを設定（1または2） |
| **wsl.exe --shutdown** |  | WSL 2を全て停止 |
| **wsl.exe --status** |  | WSLの状態表示 |
| **wsl.exe --umount ＜ディスク＞** |  | --mountでマウントしたドライブのアンマウント |
| **wsl.exe --update ［アップデートオプション］** |  | WSL 2カーネルの更新 |
| **wsl.exe --version** | -v | WSLのバージョン番号表示（Microsoft Store版WSL プレビューのみ） |
| **WSL制御機能に関するWSLコマンド** | | |
| **インストールオプション** | **ショート形式オプション** | **意味** |
| **--distribution ＜ディストリビューション＞** | -d | ディストリビューションの指定（大文字と小文字は区別されない） |
| **WSL制御機能に関するWSLコマンドのインストールオプション** | | |

| **マウントオプション** | **ショート形式オプション** | **意味** |
| --- | --- | --- |
| **--bare** |  | WSL側仮想マシンに接続するがマウント処理はしない |
| **--type ＜タイプ＞** |  | Linux Mountコマンドに伝達されるドライブのファイルシステム。既定はext4 |
| **--options ＜追加オプション＞** |  | マウントコマンドに伝達される追加のオプション |
| **--partition ＜インデックス＞** |  | マウント対象とするパーティションの番号。未指定ではドライブ全体 |
| **WSL制御機能に関するWSLコマンドのマウントオプション** | | |

**●WSL管理機能**

| **ロングオプション書式** | | **ショート形式オプション** | **動作** | |
| --- | --- | --- | --- | --- |
| **wsl.exe --export ＜ディストリビューション＞ ＜ファイル名＞|"-"** | |  | ディストリビューションをtarファイルにエクスポート | |
| **wsl.exe --import ＜ディストリビューション＞ ＜インストール先＞ ＜ファイル名＞|"-" ［インポートオプション］** | |  | tar ファイルを新しいディストリビューションとしてインポート | |
| **wsl.exe --list ［リストオプション］** | | -l | ディストリビューションの一覧を表示 | |
| **wsl.exe --set-default ＜ディストリビューション＞** | | -s | 既定のディストリビューションを設定 | |
| **wsl.exe --set-version ＜ディストリビューション＞ ＜バージョン＞** | |  | ディストリビューションのバージョンを変換（1または2） | |
| **wsl.exe --terminate ＜ディストリビューション＞** | | -t | 指定したディストリビューションを終了 | |
| **wsl.exe --unregister ＜ディストリビューション＞** | |  | ディストリビューションを登録解除、ファイルシステムを削除 | |
| **WSL管理機能に関するWSLコマンド** | | | | |
| **インポートオプション** | **ショート形式オプション** | **意味** | |
| **--version ＜バージョン＞** |  | 新しいディストリビューションに使用するバージョンを指定 | |
| **WSL管理機能に関するWSLコマンドのインポートオプション** | | | |

| **リストオプション** | **ショート形式オプション** | **意味** |
| --- | --- | --- |
| **--all** |  | インストール済みをすべて表示 |
| **--running** |  | 実行中ディストリビューションのみ表示 |
| **--quiet** | -q | ディストリビューション名のみ表示 |
| **--verbose** | -v | 詳細情報を表示 |
| **--online** | -o | インストール可能なディストリビューションの一覧 |
| **WSL管理機能に関するWSLコマンドのリストオプション** | | |

1. **WSL 1とWSL 2の変換方法**

　ここで簡単にWSL 1とWSL 2の変換について解説しておこう。インストールしたディストリビューションは、wsl.exeコマンドを使ってWSL 1とWSL 2を相互に変換することができる。

　まず、変換前にWSLディストリビューションを停止状態にしておく。これには、以下のコマンドを使う。

wsl.exe --terminate **＜WSLディストリビューション名＞**

**WSLディストリビューションを停止状態にするコマンド**

　次に、「--set-version」オプションを使って、変換を実行する。

wsl.exe --set-version **＜WSLディストリビューション名＞** **＜バージョン番号＞**

**WSL 1とWSL 2の変換を行うコマンド**

**＜WSLディストリビューション名＞**には、既にインストールされているWSLディストリビューションの名前を指定する。これは、「wsl.exe -l」コマンドで表示できる。バージョン番号には、半角数字で「1」または「2」を指定する。「1」を指定すればWSL 1、「2」を指定すればWSL 2となる。

　ただ、システム状態によって、「--set-version」コマンドが失敗することがあるようだ。その場合は、システムを再起動してから、再び「--set-version」コマンドを実行するとよい。

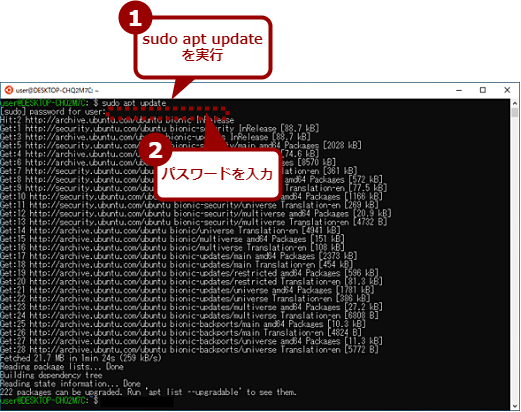
1. **WSLディストリビューションの更新**

　WSLディストリビューションのインストールが完了したら、インストールされているパッケージの更新を行っておく。Ubuntu-18.04にインストールされているパッケージの更新を行うには、「apt update」と「apt upgrade」というコマンドを組み合わせて使う。Windows OSのWindows Updateに例えると、apt updateが「更新プログラムのチェック」、apt upgradeが「更新プログラムのインストール」に相当するものだ。

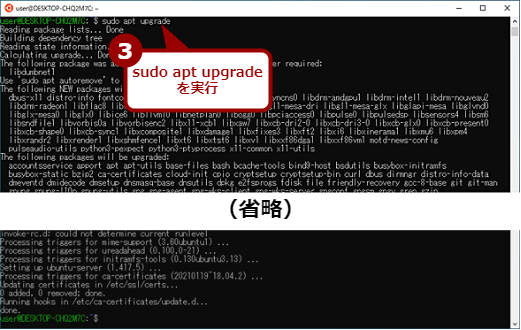
　なお、このパッケージ管理用のコマンドは、ディストリビューションにより違いがある。Ubuntu-18.04などUbuntu系やDebianなどのディストリビューションでは、aptコマンドを使うが、Open SUSEなど他のディストリビューションでは他のコマンドを使う点に注意してほしい。

　いずれもroot権限で実行する必要があるので、コマンド名の前に「sudo」を付けてから実行する。

　最初は「apt update」を実行して、更新プログラムがあるかどうかの確認を行う。更新すべきパッケージが見つかった場合は、さらに続けて「apt upgrade」を実行して、更新プログラムのインストールを行う。

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic04.png)**WSLディストリビューションを更新する（1）**  
「sudo apt update」で更新プログラムの有無を確認する。WSLディストリビューションを起動後、最初の「sudo」コマンドを実行すると、パスワードが求められるので、インストールの最後で入力したパスワードを入力する。

▼

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic05.png#_ga=2.263823631.244786985.1643411972-1474987053.1606544903)**WSLディストリビューションを更新する（2）**  
「sudo apt upgrade」を実行して、WSLディストリビューションを更新する。

　「apt update」の実行時に、「All packages are up to date.（パッケージは全て最新です）」と表示されていれば、実行する必要はない。

1. **WSLディストリビューションのバックアップ／リストア／環境復元**

　WSL自体は安定しているものの、その中でLinuxオペレーティングシステムを動作させることから、ユーザーの操作などによっては、WSLディストリビューションの動作が不安定になる可能性がある。この場合の復元方法についてまとめておく。なお、WSL 1で使われていたWSLを管理するツール「lxrun.exe」は、WSL 2には対応しておらず、最近のWindows OSには組み込まれていない。

　まず、WSLディストリビューションのバックアップ／リストアは、「wsl.exe --export」／「wsl.exe --import」で行える。バックアップするとき、WSLディストリビューションの実行イメージ（実際にはVHDX仮想ディスクファイル内にある。後述）内をLinuxのtar形式で出力する。tarファイルは、ディレクトリやファイルなどの構造を単一のファイル内に記録するアーカイブ形式の一種である。Linux内で利用可能な他、Windows 10／11にもtarファイルを扱える「tar.exe」コマンドが用意されている。

　「wsl.exe --export」ではディスクイメージをそのままアーカイブするため、完全なバックアップやリストアも可能な上、tarコマンドを使って一部を取り出すこともできる。

　具体的には、以下のようなコマンドをWin32側で実行すると、「c:/temp/U1804.tar」というファイルにバックアップが格納される（「c:/temp/U1804.tar」の部分は、Win32側の適当なフォルダとファイルを個々に指定する必要がある）。

wsl.exe --export Ubuntu-18.04 c:/temp/U1804.tar

**WSLディストリビューションをバックアップするコマンド**

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic06.png)**WSLディストリビューションをバックアップする**  
「wsl.exe --export」コマンドを実行することで、WSLディストリビューションがバックアップできる。バックアップは、tar形式で出力される。

　WSLディストリビューション自体は、前述の通り、Microsoft Storeから入手可能であり、Windows OS側からみると、Microsoft Storeアプリケーションとして扱われる。このため、WSLディストリビューションの修復やリセットは、Microsoft Storeと同じ方法が利用できる。これに関しては、Tech TIPS「[Windows 10のWSL環境を初期化してクリーンな状態に戻す](https://atmarkit.itmedia.co.jp/ait/articles/1807/06/news028.html" \t "_blank)」を参照していただきたい。

　また、wsl.exeの以下のコマンドを使うことで、WSLディストリビューションの削除も可能になる。このコマンドでは、実行イメージも削除されるため、その後同じディストリビューションを使うには、再インストールが必要なる。

wsl.exe --unregister **＜ディストリビューション＞**

**WSLディストリビューションを削除するコマンド**

　再度インストールするには、以下のコマンドを使う。

wsl.exe --install -d **＜WSLディストリビューション名＞**

**WSLディストリビューションをインストールするコマンド**

　なお、どの場合も同一WSLディストリビューションであれば、「wsl --export」で出力したtarファイルを使って、「wsl --import」コマンドで環境を復元することが可能だ。

　ただし、「--import」オプションは、WSLディストリビューションを新規にインストールして、tarファイルの内容を書き戻すものであるため、同じWSLディストリビューションをあらかじめアンインストール（「--unregister」コマンドを使う）しておく必要がある。

　また、WSLディストリビューションを構成するファイルが全て元に戻ってしまうため、再度問題が起きる可能性がある。これを回避するには、Microsoft StoreからWSLディストリビューションを新規にインストールしたのち、WSLディストリビューション内でtarコマンドを使って、必要なファイルのみ取り込むようにすること。

1. **tarコマンドによるtarファイルからのファイルの取り出し**

　tarコマンドには、下表のようなオプションがある。基本的な使い方としては、「アーカイブ復元」「アーカイブ内のファイル一覧」「アーカイブ作成」の3つだ。それぞれ、以下のオプション指定で行う。

| **オプション** | **短い形式** | **必須** | **意味** |
| --- | --- | --- | --- |
| **--catenate, --concatenate** | -A | ○ | 既存のtarファイルに追加 |
| **--create** | -c | ○ | 新規tarファイルを作る |
| **--diff、--compare** | -d | ○ | tarファイルとファイルを比較 |
| **--delete** |  | ○ | tarファイルから削除 |
| **--append** | -r | ○ | tarファイルの最後に追加 |
| **--list** | -t | ○ | tarファイル内のファイルリスト |
| **--update** | -u | ○ | tarファイル内のファイルを更新 |
| **--extract、--get** | -x | ○ | tarファイルからファイルを抽出 |
| **--file ＜tarファイル＞** | -f |  | tarファイル/出力デバイスを指定 |
| **--verbose** | -v |  | 処理を詳しく表示する |
| **--files-from ＜ファイル＞** | -T |  | ファイルFで抽出ファイルを指定する |
| **--exclude ＜ファイルリスト＞** |  |  | **＜ファイルリスト＞**を除外 |
| **--exclude-from ＜ファイル＞** | -X |  | 除外するファイルを**＜ファイル＞**から読み込み |
| **--help、--usage** |  |  | オンラインヘルプ |
|  | | | |

**tarコマンドのオプション**  
**※**ショートコマンドはオプション先頭に文字だけを並べることができる。  
**※**必須コマンドのどれか1つを必ず指定する必要がある。  
**※**原則対象のtarファイルは「--file」／「-f」オプションで指定する。

　なお、tarコマンドはWindows 10／Windows 11にも搭載されている。以下のコマンドラインは、どちらでも有効であるが、同じパスであっても、WSL側とWin32側ではパス指定方法が異なることに注意してほしい。

**■復元**  
tar -x -v -f **＜tarファイル＞** ［**＜復元するファイル、ディレクトリ＞**……］  
  
**■一覧**  
tar -t -v -f **＜tarファイル＞** ［**＜表示するファイル、ディレクトリ＞**……］  
  
**■作成**  
tar -c -v -f **＜tarファイル＞** **＜アーカイブするファイル、ディレクトリ＞**……

**tarコマンドの主な書式**

　原則、tarファイルは、全て相対パスで記録されており、コマンドを実行したワーキングディレクトリを起点にファイルが復元される。WSLコマンドでエクスポートしたtarファイルは、全て「./」からの記録になっている。これは、システムを復元するときに、絶対パスで書き戻してしまうと問題が起きる可能性があるからだ。WSLの場合、このtarファイルのパスは、WSL側のパスで記述されている点にも注意されたい。

　こうしたtarファイルには、bashなどの挙動を変更するファイルも含まれており、迂闊（うかつ）に上書きしてしまうと、次回以降の利用で問題が起きてしまう可能性もある。このため、いったん、ユーザーディレクトリや/tmpディレクトリなど既存ファイルを上書きする心配のない場所に復元させるなどしてから、中身を確認した方がよい。例えばtarファイルに含まれる「./home/shio」ディレクトリのファイルを確認するには、以下のコマンドを実行する。

tar -t -v -f /mnt/c/temp/U1804.tar ./home/shio

**「./home/shio」ディレクトリのファイルを確認するコマンド**

　Linuxでは、単独で「cd」コマンドを実行すると、ユーザーのホームディレクトリに移動するので、ファイル関連のコマンドを扱い、ホームディレクトリにファイルを置きたい場合には、その前に「cd」コマンドを実行するとよい。

　エクスポートしたtarファイルから自身のユーザーディレクトリ（ユーザー名「shio」）内のファイルを取り出すには、自身のユーザーディレクトリなど書き込みが可能な場所に移動して、以下のコマンドを実行すればよい。

tar -x -v -f /mnt/c/temp/U1804.tar ./home/shio/

**tarファイルからファイルを取り出すコマンド**

　これで現在のディレクトリに「home/shio」ディレクトリが作成され、そこにtarファイルにアーカイブされていたユーザーディレクトリのファイルが復元される。後は、必要に応じて実際の場所に書き戻せばよい。

　Linuxの場合、システム関連の設定のほとんどが「/etc」にあり、ユーザー関連の設定などはユーザーのホームディレクトリ（「/home」以下）にある。環境を復元するには、最低限、この2つのディレクトリは戻す必要がある。

　しかし、「/etc」ディレクトリは、インストールされているプログラムにも関係するため、一回、ユーザーディレクトリの下などに置き、必要に応じて、「/etc」フォルダのファイルを変更していく。いきりなりtarから復元した「/etc」ディレクトリをシステムの「/etc」ディレクトリに上書しないようにする。

　その他の部分は、ディストリビューションに付属するファイルもしくは、ディストリビューションのパッケージ管理コマンドでインストールしたプログラムだ。これらは、ディストリビューションが動作していれば、いつでも復元可能なので、あまり気にする必要はない。

1. **Windows 11で外部ドライブをマウントする**

　Windows 11のWSLでは、外部ドライブを直接マウントすることが可能になった。これにより、ext4形式など、Linux固有のファイルシステムでフォーマットされたディスクを直接扱うことが可能になる。

　WSL側では、標準的にext4形式のストレージを使うが、実際には仮想ハードディスクファイル（vhdx）の中に実現されたものだ。また、Win32側のファイルアクセスも可能だが、これは「9P（Plan 9 Filesystem Protocol）」と呼ばれるプロトコルを使って擬似（ぎじ）的にネットワークアクセスの形で行われるため、それほど効率的ではない。

　しかし、WSLで大量データの処理などを行いたい場合、どうしても大容量のネイティブファイルシステムのドライブを扱いたくなることがある。このような場合、「wsl.exe」コマンドで外部ドライブをマウントすることができる。

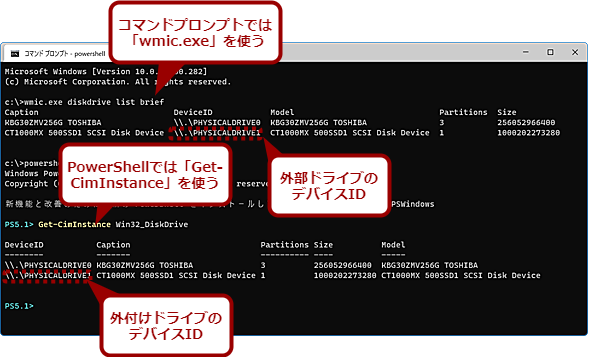
　ただし、通常外付けディスクとして販売されているものは、あらかじめNTFSなどでフォーマットしてあるものが少なくない。このため、一度パーティションを削除するなどして、ドライブをext4形式でフォーマットする必要がある。ここでは、既にNTFSなどでパーティションが確保されているドライブであるとして、外部ドライブの接続からフォーマット、マウントまでの手順を示す。

　まずは、外部ドライブを接続し、そのデバイス名を調べる。それには以下のコマンドが利用できる。

**■cmd.exeの場合**  
wmic.exe diskdrive list brief  
  
**■PowerShellの場合**  
Get-CimInstance Win32\_DiskDrive

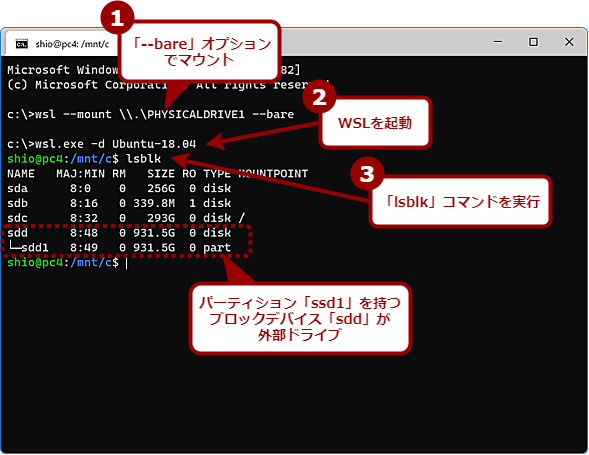
**外部ドライブのデバイス名を調べるコマンド**

　ここで、外部ドライブが接続されているとき、下画面のように「\\.\PHYSICALDRIVE0」から始まるDevice IDが表示される。ここから外部ドライブのDevice IDを調べる。

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic07.png)**外部ディスクのデバイス名を調べる**  
外部ドライブをWSL側に接続するには、「wmic.exe」（コマンドプロンプト）や「get-CimInstance」（PowerShell）コマンドで記憶デバイスを表示させ、デバイスIDを調べる。この例では、「PHYSICALDRIVE1」が対象の外部ドライブである。どれが外部ドライブか分からないときには、外して同じコマンドを実行し、外す前の結果と比較する。

　たいていの場合、末尾の数字は、接続順に付けられるため、大きな番号のものが外部ドライブに相当するはずだ。分からなければ、一回外部ドライブを外して、前記のコマンドを実行すれば、表示されなくなったドライブが外部ドライブである。以後の説明では「\\.\PHYSICALDRIVE1」が外部ドライブだったとする。

　これをWSL側に接続するが、この時点では、NTFSでフォーマットされているため、マウントはせず接続だけにする。このコマンドには管理者権限が必要だ。

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic08.png)**外部ドライブをマウントする**  
外部ドライブがNTFSや未フォーマットなどWSLのext4ファイルシステム以外の場合、最初に「--bare」オプションを付けて外部ドライブをマウントする。その後WSLを起動し、「lsblk」コマンドで外部ドライブのデバイス名（/dev/sddなど）を調べる。

wsl.exe --mount \\.\PHYSICALDRIVE1 --bare

**外部ドライブをマウントするコマンド**

　その後「wsl.exe」コマンドで、wslに入り、WSL側でのブロックデバイス名を確認する。それには、「lsblk」コマンドを使う。

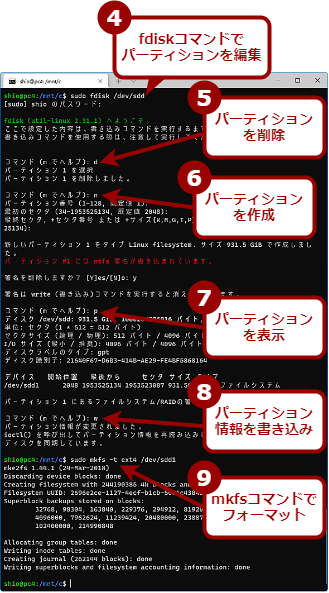
wsl.exe  
lsblk

**WSLでブロックデバイス名を調べる**

　Linuxのブロックデバイスは、「/dev/sda」「/dev/sdb」、「/dev/sdc」と「/dev/sd」の末尾に「a」「b」「c」「d」が付けられる。こちらも多くの場合は接続順になる。だが、WSL起動のタイミングによっては、外部ドライブが前に来てしまうことがある。ここで「MOUNTPOINT」に“/”があるブロックデバイスは、Linuxのルートファイルシステムにマウントされているもので、外部ドライブではない。

　また、Windows 11のWSL 2では、「/dev/sda」と「/dev/sdb」はシステム側で先に利用されているようだ。ここでは、WSL 2側のデバイス名が「/dev/sdd」であったとしよう。「lsblk」の出力では、下に「/dev/sdd1」と表示されるはずだ。これが外部ドライブにあるNTFSパーティションである。

　では、NTFSパーティションを削除して、ext4パーティションを作成する。それには、「fdisk」コマンドを使う。

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic09.png)**外部ドライブにパーティションを作成する**  
「fdisk」コマンドで外部ドライブにext4フォーマット用のパーティションを作る。NTFSのパーティションはext4ファイルシステムには変換できないため、パーティションを削除して、新規に作成する。「fdisk」コマンド中では、「m」コマンドを使うとヘルプを表示できる。パーティションを作成したら「w」コマンドで情報を書き込んで「fdisk」コマンドを抜け、「mkfs」コマンドで論理フォーマットを行う。

sudo fdisk /dev/sdd

**ext4パーティションを作成するコマンド**

　具体的な手順としては、「d」コマンドでパーティションを削除し、「n」コマンドで新規にパーティションを作成、その後「w」コマンドでディスクに書き込みを行って終了する。途中「p」コマンドでパーティションの状態を表示できる。終わったら、次は「mkfs」コマンドでファイルシステムを作る（Windows OSでいうフォーマットに相当）。このときには、パーティションのデバイス名（/dev/sdd1）を指定すること。コマンドの実行完了までは時間がかかる点にも注意してほしい。

sudo mkfs -t ext4 /dev/sdd1

**ファイルシステムを作るコマンド**

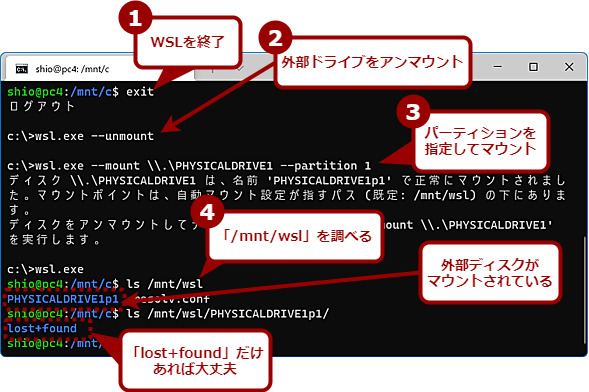
　ファイルシステムの作成が終了したら、一回WSLを抜け、「\\.\PHYSICALDRIVE1」をマウントし直し、WSLに戻る。

wsl.exe --unmount \\.\PHYSICALDRIVE1  
wsl.exe --mount \\.\PHYSICALDRIVE1 --partition 1  
wsl.exe

**WSLでマウントのし直しを行う**

　次回からは、2行目のコマンドだけでドライブの準備が完了する。

　では、WSLからマウントされたドライブを確認する。メッセージにあったように、WSL側では、「/mnt/wsl」の下に外部ドライブがマウントされている。「ls」コマンドや「lsblk」コマンドで確認ができる。「/mnt/wsl/PHYSICALDRIVE1p1/」の下に「lost+found」があれば、基本的に準備は完了だ。

[](https://image.itmedia.co.jp/l/im/ait/articles/2201/28/l_wi-wsl2basic10.png)**パーティション指定して外部ドライブをマウントする**  
「mkfs」コマンドが終了したら、WSLを抜け、Win32側で外部ドライブをマウントし直す。このとき「--partition」オプションを使って、マウントするパーティションを指定する。これで作業は完了したので、WSL側で「/mnt」ディレクトリの下に外部ドライブがマウントされているかを確認する。

　本稿では、コマンドラインを使ってWSLをインストールする方法などを紹介したが、Windows 11では、WSLプレビュー版の一般向け公開も始まった。これは、Microsoft Storeからインストールを行うものだ。「Windows Subsystem for Linux Preview」という名称で公開されている。このプレビュー版をインストールすると、更新もMicrosoft Store経由で自動的に行われる。

　いままでは、Windows Insider Programに登録し、Devチャネルで配付されるプレビュー版のWindows 11をインストールしないと、WSLのプレビュー版を使うことはできなかった。しかし、Devチャネルのプレビュー版は月に数回アップデートがある上、不具合が生じる可能性もあるため、日常的に利用するは厳しい。

　これに対してMicrosoft Store版では、OS自体は安定版のWindows 11を利用するため利用しやすい。

　もし、WSLの今後に興味があるようなら、Windows 11でMicrosoft StoreからWSLプレビュー版をインストールしてみるといいだろう。