WSL2の構成と「まともな」Linux環境として運用するためのアレコレ

EPNetFaN 2024年度第16回

佐々木洋平

連絡先: uwabami@gfd-dennou.org

2024年9月11日

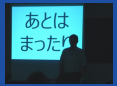
おはようございます

佐々木です

Important Caveat

- 疑問, 質問, ツッコミ, 茶々, **大歓迎**
  - その場でインタラクティブにどうぞ
- オンライン参加の方は質問の前に一言頂けるか, 挙手ボタンなど, ワンアクションお願いいたします.
  - チャットは瞬時には拾えないかもしれませんがご容赦下さい.

About me - https://about.me/uwabami



- 名前:
  - 佐々木洋平/Youhei SASAKI
    - 青森出身 → 北大地惑 → . . . → 北海道情報大
- 連絡先:
  - email: uwabami@gfd-dennou.org
  - X(旧 Twitter): ~~@uwabami~~ (凍結中)
  - Mastodon: uwabami@junkhub.org
- 所属:
  - 北海道情報大学 情報メディア学部
    - 計算機ネットワークの運用とセキュリティについて教えています.
  - Debian Project/Debian JP Project
    - 去年まで Debian JP の会長してました. 今年からは幹事やってます.

「予告編」はこんなんでした

- Windows Subsystem for Linux(WSL) は 2016 にベータ版がリリースされて以来, Windows 上で Linux のコマンド群を利用する環境として広まりつつある. しかしながら WSL1 および WSL2 が Windows 上でどの様に「Subsystem」として機能しているのかについては明確には整理公開されておらず, Linux 環境に慣れた人がメイン環境として利用しようとすると, しばしば予想とは異なる動作に悩んでしまう.
  - つまり, 私がハマったということで...
- 本講演では, 発表時点での WSL1 および WSL2 の構成について解説するとともに, 現状でよくあるハマり所について解説を試みる.

本日のお品書き

WSL とは

WSL1 の仕組み

WSL2 の仕組み

ここが変 (?) だよ, WSL2

本日のお品書き

WSL とは

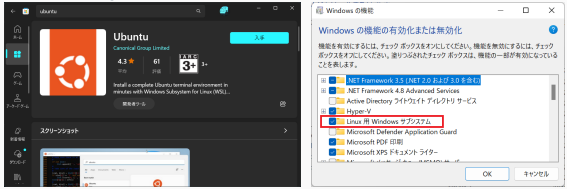
WSL1 の仕組み

WSL2 の仕組み

ここが変 (?) だよ, WSL2

WSL とは?

- **W**indows **S**ubsystem for **L**inux
  - Windows 上で Linux のプログラムを動作させる仕組み
  - Windows 10 Fall Creators Update から正式版



  - Windows Mobile で Android アプリを動かそうとした事が発端
  - 似た仕組みとしては Cygwin もあるが,
    WSL は再コンパイルすることなく Linux の 64bit elf バイナリが動作する.
- 現在は WSL1, WSL2 が提供されている.
  - 実装 (⇒ できること, できないこと) がだいぶ違う.

WSLの使い方...

- **導入とか使い方は今日はお話しません.**
  - マイクロソフト本家のドキュメント [1] を良く読みましょう.

1. MicrosoftStore で Ubuntu をインストール
2. 起動して, 初回はユーザ名とパスワードを入力
3. あとは普通の Linux と同じです
   ↑ じゃなかったのが面倒なんですよ.

WSLの使い方...

- **導入とか使い方は今日はお話しません.**
  - マイクロソフト本家のドキュメント [1] を良く読みましょう.

1. MicrosoftStore で Ubuntu をインストール
2. 起動して, 初回はユーザ名とパスワードを入力
3. あとは普通の Linux と同じです
   ↑ じゃなかったのが面倒なんですよ.

本日のお品書き

WSL とは

WSL1 の仕組み

WSL2 の仕組み

ここが変 (?) だよ, WSL2

WSLの仕組み

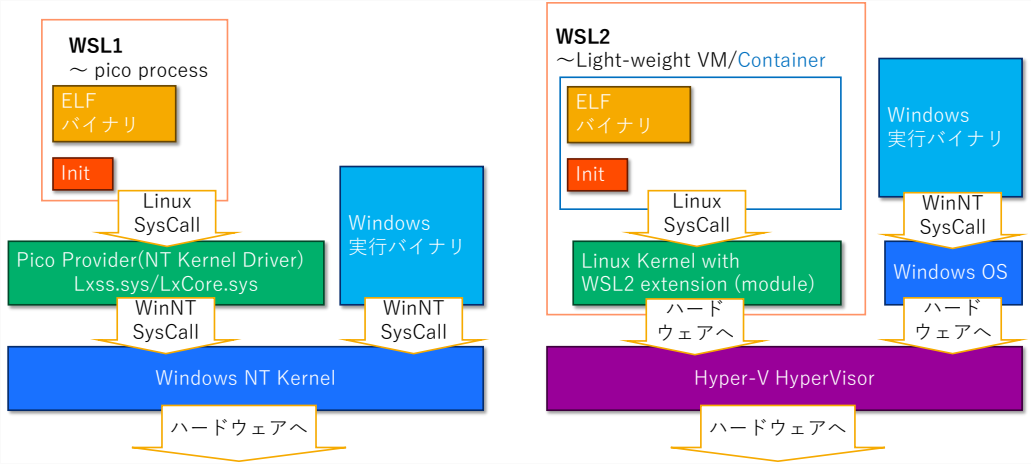


図 1: WSL1 と WSL2 の構造. LXSS の資料 [2] を元に作成

WSL1 の仕組み

- WSL1 を起動すると...
  - Windows 10 から導入された "Pico Process" として, WSL1 用の init が起動
    - PicoProcess: 権限を削ぎ落した隔離されたプロセス
  - WSL1 用の init は
    1. Windows とのファイルアクセス用の 9P サーバを起動
    2. Linux の daemon は起動しない
    3. `/etc/passwd` に記載されたログインシェルを実行 (ログイン処理は無い)
- WSL1 のプロセスは Windows から見るとそれぞれ 1 プロセス
  - タスクマネージャからそれぞれのプロセスが見える

NT Process, Minimal Process, PicoProcess

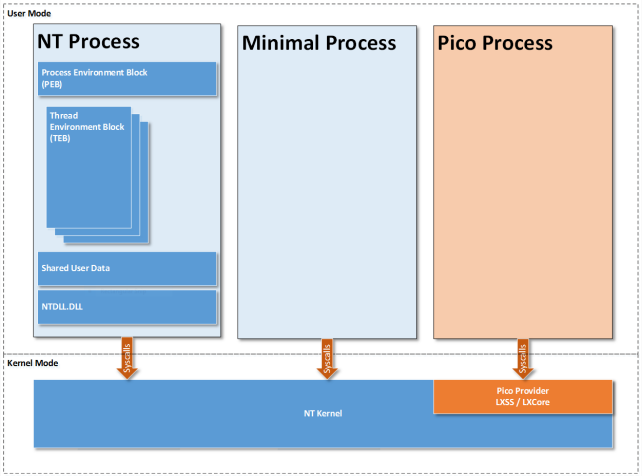


図 2: Pico Process Overview [3] より引用

WSL1 の仕組み

- WSL1 を起動すると...
  - Windows 10 から導入された "Pico Process" として, WSL1 用の init が起動
    - PicoProcess: 権限を削ぎ落した隔離されたプロセス
  - WSL1 用の init は
    1. Windows とのファイルアクセス用の 9P サーバを起動
    2. Linux の daemon は起動しない
    3. `/etc/passwd` に記載されたログインシェルを実行 (ログイン処理は無い)
- WSL1 のプロセスは Windows から見るとそれぞれ 1 プロセス
  - タスクマネージャからそれぞれのプロセスが見える

WSL1 のファイルアクセス

- 実データは全てそのまま NTFS として保存されている.
  - Linux で利用する際には **VolFS** としてマウントしている
    - 実ファイルとは別にメタデータ (permission 等) を保存しておいて利用
    - なお, `cmd` や `powershell` 等で直接触るのはオススメしません
- Linux → Windows: **DrvFS**
  - Windows 側のファイルを 777 or 555 でマウントして利用 (変更不可能)
    - `/mnt/c/Users/...` など
    - ネットワークドライブは非対応だが, USB メモリ等はマウントしてそのまま使える.
- Windows → Linux: **9P** でアクセスする
  - ベル研の Plan 9 由来のネットワークファイルプロトコル
    - WSL1 の `init` で 9P のサーバを起動. Windows 側のクライアントは `p9rdr.sys`.
    - アクセスするまでデータは見えない (←エクスプローラで表示が自動更新されない).

WSL1 の仕組み, まとめ

- WSL1 は Pico Process として起動 (→ Linux kernel は動いていない)
  - Pico Provider: `LxCore.sys`, `Lxss.sys`, `LxssManager.sys`
  - Pico Process: Microsoft 製 `/init`, + Linux ディストリビューション
  - データは VolFS として NTFS 上に保存
  - Linux → Windows のファイルアクセスは DrvFS
    - **9P に比して圧倒的に高速**. この用途では WSL1 を利用するのが吉.
- Windows 側のフロントエンド
  - `wsl.exe` で管理. 表示は `conhost.exe` (コマンドプロンプトと同じ).
    - 他に `wslhost.exe`, `p9rdr.sys`, `lxssManager.dll` も起動している
  - LxssManager サービスが Pico Provider と通信
  - Windows → Linux のファイルアクセスは 9P プロトコルで行なう

本日のお品書き

WSL とは

WSL1 の仕組み

WSL2 の仕組み

ここが変 (?) だよ, WSL2

WSLの仕組み

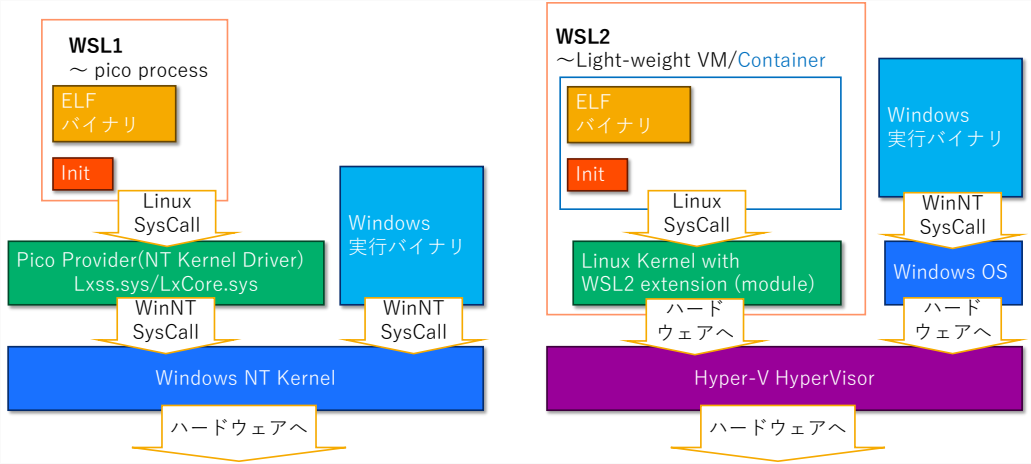


図 1: WSL1 と WSL2 の構造. LXSS の資料 [2] を元に作成

WSL2の仕組み

- WSL2 を導入すると...
  - Hyper-V HypverVisor が有効になる
  - Windows では仮想マシンの管理サービス (`vmwp.exe`) が起動.
    - 9P のサーバも同梱されている
  - WSL2 は 軽量 VM/Container として起動: Hypver-V 上で **Linux kernel が動作**.
    - コンテナ: Microsoft 製 init, 仮想 HDD (ext4.hdx), Linux ディストリビューション

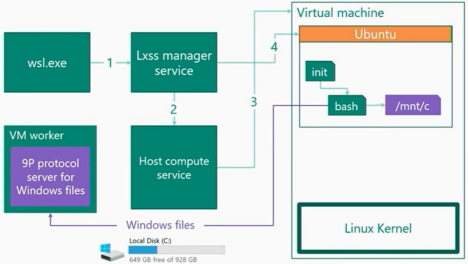


図 3: The new Windows subsystem for Linux architecture: a deep dive[4] より

WSL2のファイルアクセス

- Hyper-V の仮想ディスク上の ext4 ファイルシステム
  - WSL1 の VolFS (NTFS + メタデータ) に比して高速
- Windows ⇔ Linux は全て 9P で行なう
  - DrvFS は使わない: Linux → Windows は遅くなった... 😢
    - WSL2 のファイルシステムへは Explorer からアクセス可能. エントリも追加されている

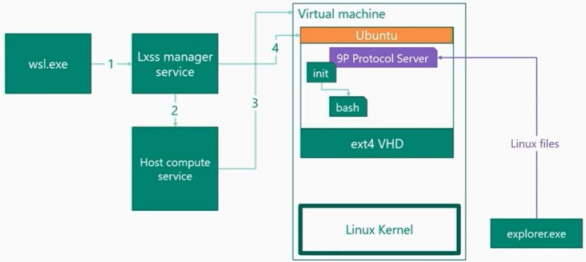


図 4: The new Windows subsystem for Linux architecture: a deep dive[4] より, explorer からアクセスする場合

WSL2 の仕組み, まとめ

- WSL2 は Hyper-V HyperVisor 上の仮想マシン/コンテナとして動作
  - Microsoft 製 `/init` + 仮想 HDD + Linux ディストリビューション
  - データは仮想 HDD へ保存.
    - Linux へのデータ読み書きは **VolFS に比して数 10 倍高速**. この用途では WSL2 が良い.
- Windows 側のフロントエンド
  - `wsl.exe` で管理. 表示は `conhost.exe` (コマンドプロンプトと同じ).
    - 他に `wslhost.exe`, `p9rdr.sys`, `lxssManager.dll` も起動している
    - Hyper-V サービス: `vmcompute.exe`, 仮想マシン管理 (含む 9P サーバ): `vmwp.exe`
  - LxssManager サービスが仮想マシンの管理サービスとやりとり
- Windows ⇔ Linux のファイルアクセスは 9P プロトコルで行なう

WSLの仕組み

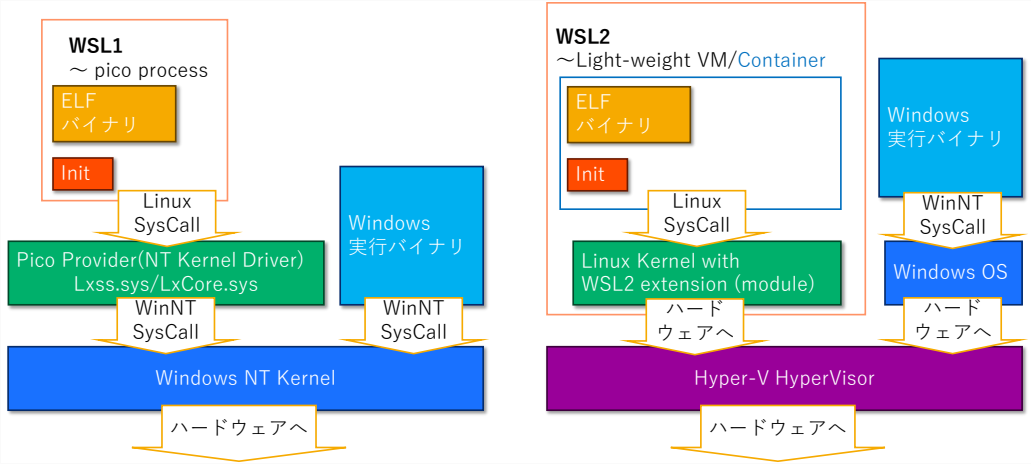


図 1: WSL1 と WSL2 の構造. LXSS の資料 [2] を元に作成

本日のお品書き

WSL とは

WSL1 の仕組み

WSL2 の仕組み

ここが変 (?) だよ, WSL2

WSL2 を使い始めた.

- 昨年の夏に Let's Note CF-SR3 を購入しました. . . が.
  - SecureBoot できない.
  - Linux 使用時, idle 時の CPU FAN の回転数が下がらない.
    - ちょっと騒々しすぎて, 静かな所ではおいそれと開けない.
  - Hack する時間が取れず. . .
- 学生に WSL2 を使わせる機会が増えてきた.
  - Eating your own dog food[5]
- Microsoft が 「WSL2 で systemd が使えるよ! [6] 」と言ったので.
  - 普通に daemon 動きそうだな → なら常用環境にできるかな, と.

WSL2 で systemd　i

- 結論: systemd の version をちゃんと確認して下さい
  - **systemd status fails output on Ubuntu #8879**
    https://github.com/microsoft/WSL/issues/8879
    - systemd ≤ 249 or ≥ 255.7 なら OK
  - 現時点では
    - Debian は testing 以上, かつ cgroup2 を有効化すれば OK
      → 私は普段使いが sid なので問題無いけど...
    - Ubuntu は 22.04 なら OK
    - Ubuntu-24.04 は cgroup2 を有効化すれば OK
    - 他のディストリビューションでもイロイロと対応が必要, かもね.

WSL2 で systemd ii

- cgroup2
  - Control Group version 2 のこと
  - プロセスをグループ化し, グループ内に存在するプロセスに対して共通の管理を行う機能.
    - 例: CPU やメモリなどのリソースの制限, アクセスの制御など.
- 新しめの systemd は cgroup2 が必要
  - WSL の Linux kernel は cgroup1 で動作している
- cgroup2 を有効にしたい場合は `/etc/fstab` にて以下の様に

```
cgroup2 /sys/fs/cgroup  cgroup2 rw,nosuid,nodev,noexec,relatime,nsdelegate      0       0
```

時計合わせ

- suspend/resume で結構な頻度で時計がズレます.
  - 仮想環境だと良くある話ですね.
  - Windows 側との時刻合わせをしておきたい.
    - WSL 側で時刻合わせをしても良いけど, ファイルアクセスの際の時刻ズレが困る
- 結論: /dev/ptp0 が生えてます. これを使いましょう.
  - Time sync for Linux VMs in Azure:
    - https://github.com/MicrosoftDocs/azure-docs/blob/main/articles/virtual-machines/linux/time-sync.md
- chrony などを導入して, /dev/ptp0 を参照するように.
  - /etc/chrony.conf の末尾あたりに以下を追記しておきます

```
## use ptp0              # 以下, 追記
refclock PHC /dev/ptp_hyperv poll 3 dpoll -2 offset 0 stratum 2
```

WSLg を試してみる

- WSLg: Windows Subsystem for Linux GUI
  - WSL 環境で X サーバと Wayland サーバを提供
    - GUI アプリケーションを動作させる.
- **キー配列が US 配列になったり JP 配列に戻ったり, と謎挙動**
  - 特にログイン時の keyring の unlock で困る
    - どこで制御されているのか不明 (要調査).
  - VcXsrv や X410 などを使った方が良い, かも.

WSLg の構造

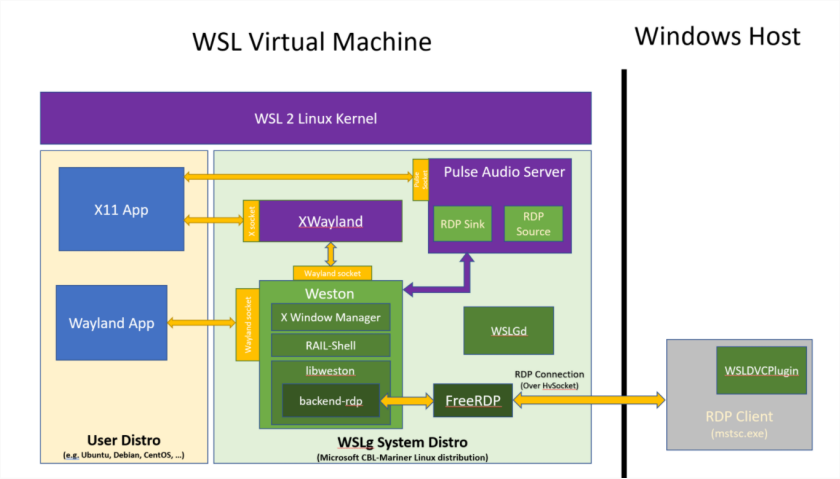


図 5: WSLg の構造 [7]

WSLg を試してみる

- WSLg: Windows Subsystem for Linux GUI
  - WSL 環境で X サーバと Wayland サーバを提供
    - GUI アプリケーションを動作させる.
- **キー配列が US 配列になったり JP 配列に戻ったり, と謎挙動**
  - 特にログイン時の keyring の unlock で困る
    - どこで制御されているのか不明 (要調査).
  - VcXsrv や X410 などを使った方が良い, かも.

まとまらないまとめ

- WSL1 と WSL2 の内部構造についてお話しました.
  - 用途に応じて使い分けるのが良いかと

バージョン	ベースとなる技術	メリット	デメリット
WSL1	システムコールの変換	メモリ消費が少ない	▶ 命令変換によるオーバーヘッド ▶ 不完全なシステムコール対応 ▶ Windows → WSL のアクセスが遅い
WSL2	Hyper-V 仮想化	▶ システムコール完全互換 ▶ WSL 内でのディスクアクセス速度の改善	メモリ消費量の増大

- 「普通の Linux」だと思うと結構ハマる.
  - 特に network まわり (今回触れてない). Hyper-V の Switch を弄ればなんとか...

Reference i

[1] Windows Subsystem for Linux Documentation. https://learn.microsoft.com/en-us/windows/wsl/, 2022/06/27. LastAccess: 2024/09/10.

[2] lxss – Fun with the Windows Subsystem for Linux (WSL/LXSS). https://github.com/ionescu007/lxss, 2017/03/09. LastAccess: 2024/08/17.

[3] Pico Process Overview. https://learn.microsoft.com/ja-jp/archive/blogs/wsl/pico-process-overview, 2016/05/23. LastAccess: 2024/08/15.

[4] The new Windows subsystem for Linux architecture: a deep dive - BRK3068. https://www.youtube.com/watch?v=1whMThePdIo. LastAccess: 2024/08/17.

[5] ドッグフーディング. https://ja.wikipedia.org/wiki/%E3%83%89%E3%83%83%E3%82%B0%E3%83%95%E3%83%BC%E3%83%87%E3%82%A3%E3%83%B3%E3%82%B0. LastAccess: 2024/09/10.

[6] Systemd support is now available in WSL! https://devblogs.microsoft.com/commandline/systemd-support-is-now-available-in-wsl/. LastAccess: 2024/08/17.

[7] Steve Pronovost. WSLg Architecture. https://devblogs.microsoft.com/commandline/wslg-architecture/, 2021/04/19. LastAccess: 2024/08/17.