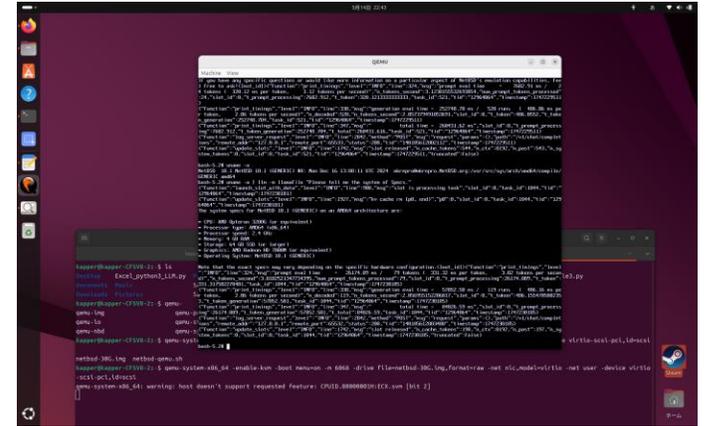


# 【初心者向け】生成AI SimonW/LLMとOllama・llamafire無料APIでコマンドラインをAI革命するセミナー。CPUでもGPUでも。NetBSDもOK

1. 自己紹介、生成AIとは
2. SimonW/LLMインストール
3. SimonW/LLM簡単な使い方
4. SimonW/LLMのネタ、応用事例
5. ハードウェア、GPU高速化



発表者: Kapper

ガジェットハッキング  
ユーザーグループ

This Presentation:

Slideshare & PDF files  
publication of my HP

<http://kapper1224.sakura.ne.jp>

Gadget Hacking User Group

Speaker: Kapper

## メインテーマ「ガジェット & 生成AI」

OSC京都2025 2025年8月3日

Place: 中小企業振興会館

# 自己紹介

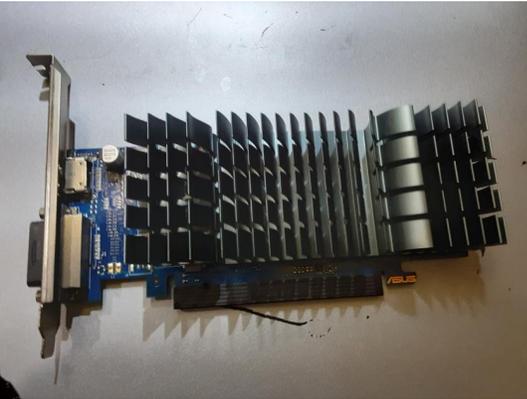
# Self Introduction



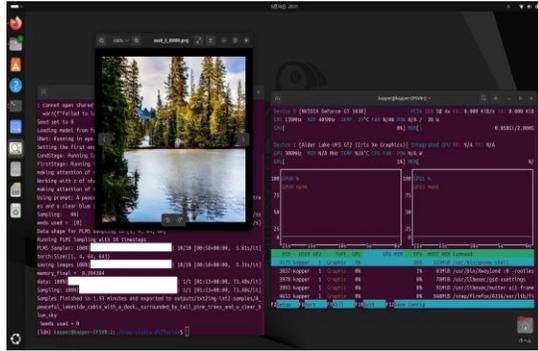
- My name: Kapper
- X(Twitter) account : @kapper1224
- HP : <http://kapper1224.sakura.ne.jp>
- Slideshare: <http://www.slideshare.net/kapper1224>
- Docsell: <https://pawoo.net/@kapper1224/>
- Mastodon: <https://pawoo.net/@kapper1224/>
- Facebook : <https://www.facebook.com/kapper1224/>
- My nobels : <https://ncode.syosetu.com/n7491fi/>
- My Posfie(Togetter) : <https://posfie.com/@kapper1224>
- My Youtube : <http://kapper1224.sakura.ne.jp/Youtube.html>
- My Hobby : Linux、\*BSD、Generative AI and Mobile Devices
- My favorite words : The records are the more important than the experiment. 「記録は実験より勝る」
- Test Model : Netwalker、Nokia N900、DynabookAZ、RaspberryPi、Nexus7、Nexus5、Chromebook、GPD-WIN、GPD-Pocket、Macbook、NANOTE、SteamDeck、Windows Tablet、SailfishOS、UBPorts、postmarketOS、NetBSD and The others...
- Recent my Activity :
  - Hacking Generative AI (Images and LLM) on a lot of devices.
  - Hacking Linux on Windows1x Tablet (Intel Atom and Gaming tablet) and Android Smartphone.
  - Hacking NetBSD and OpenBSD on UEFI and Windows Tablet.
  - Hacking SimonW/LLM, RAG and Finetuning LLM.
- 後、最近小説家になろうで異世界で製造業と産業革命の小説書いていますなう。

# Recently my activities

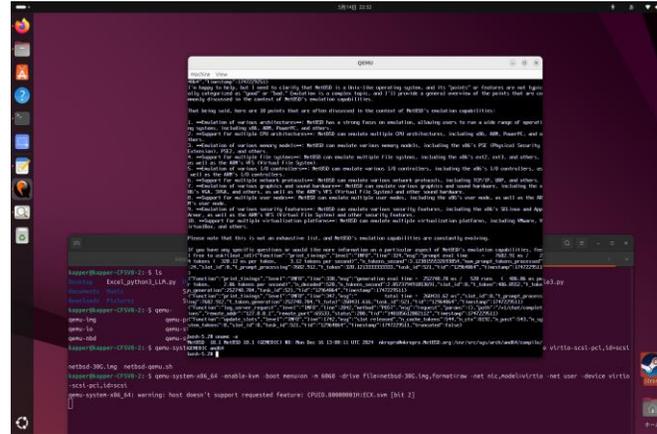
Generative AI  
on Junk GPU Boards(2GB)



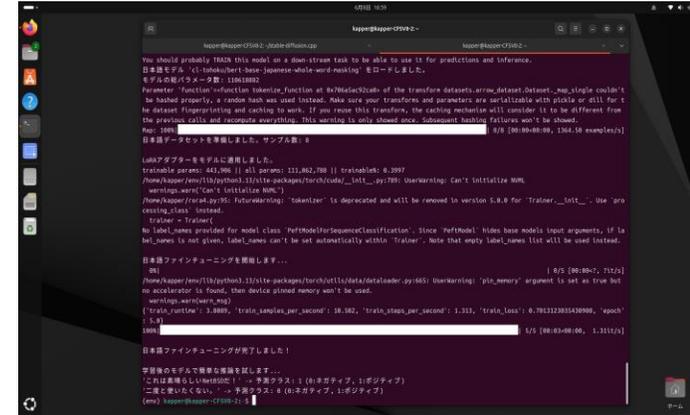
Tiny Stable-diffusion  
(VRAM 1GB)



SimonW/LLM and Python  
(CUDA,vulkan,CPU)



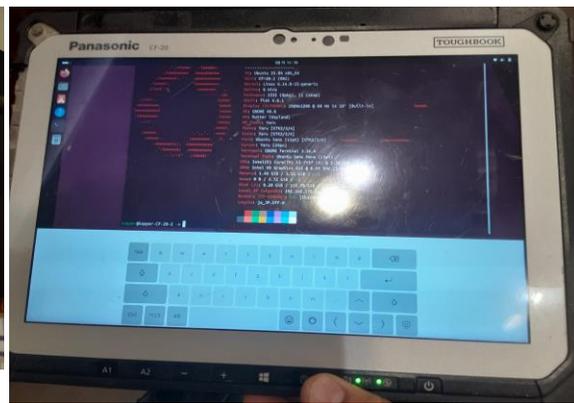
RAG,LLM Finetuning  
on Python



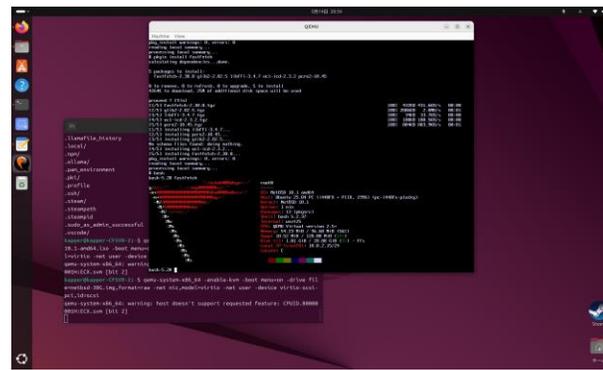
Linux on Gaming Tablet PC



Junk Windows Tablet



Generative AI on NetBSD



OSC,KOF参加



# OSC名古屋2024,OSC京都2024で発表したローカル生成AIの資料を公開中。よろしくお願ひします。

- Slideshare、ドクセル、Speakerdeck、PDFファイルなど

はじめての生成AIローカルLLM、llama.cppとllamafire。練習演習あり初心者向け講座

May 26, 2024 • 0 likes • 328 views

 Netwalker lab kapper

OSC名古屋2024の発表資料です。生成AIのローカルLLMを対象です。



2 of 47

 Download now

Download to read offline

Page 1



Page 2



Page 3



Page 4



Page 5



Page 6



はじめての生成AIローカルLLM、llama.cppとllamafire。練習演習あり初心者向け講座

1. 準備
2. ローカルLLMと生成AI
3. llamafire演習
4. llama.cpp,Ollama他
5. その他



発表者: Kapper

ガジェットハッキング  
ユーザーグループ

This Presentation:  
Slideshare & PDF files  
publication of my HP  
<http://kapper1224.sakura.ne.jp>  
Gadget Hacking User Group  
Speaker:Kapper

メインテーマ「ガジェット&生成AI」

OSC名古屋2024 2024年5月25日  
Place: 中小企業振興会館 15:00~15:45

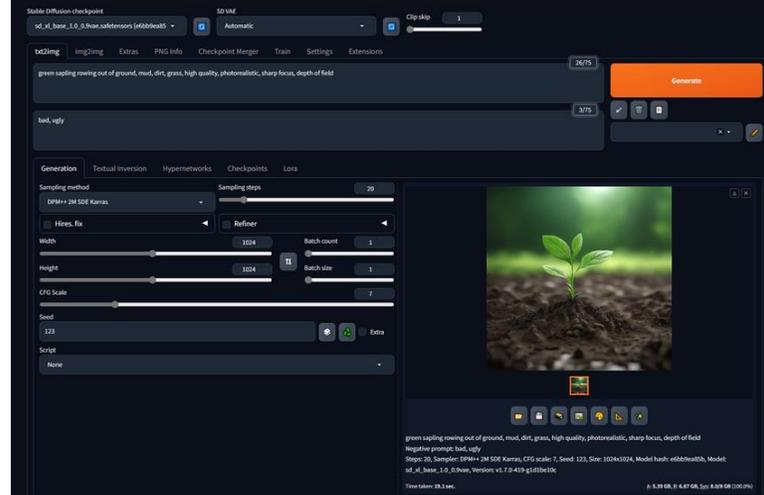
# 事前ご連絡

- 今回はコマンドラインでSimonW/LLMの使い方
- 可能であればOllama, llamafileとSimonW/LLMをインストールお願いします。
- Androidの人はTermuxかLinuxコンテナ。
- iOSの人はエミュレータで(適当なネイティブアプリなし?)
- GPUは必ずしも必須ではなく、CPUの計算が遅いだけAMD RadeonやIntel Arc、UHDでも動く
- CUDAが無いOSでもvulkanを入れればGPU高速化可能
- 低スペックPC・VRAMならQwen3 1.7B Q4\_K\_M(1.28GB) か Gemma3 1B Q4\_1(764MB)オススメ
  
- 内容がコマンドラインなので眠くなるかもしれませんがLLM作業しながらリラックスして頂きたいです

## Screenshots of Comparison

I tested with several devices, and this is a typical result from 8GB VRAM (3070ti laptop) with SDXL.

This is original WebUI:

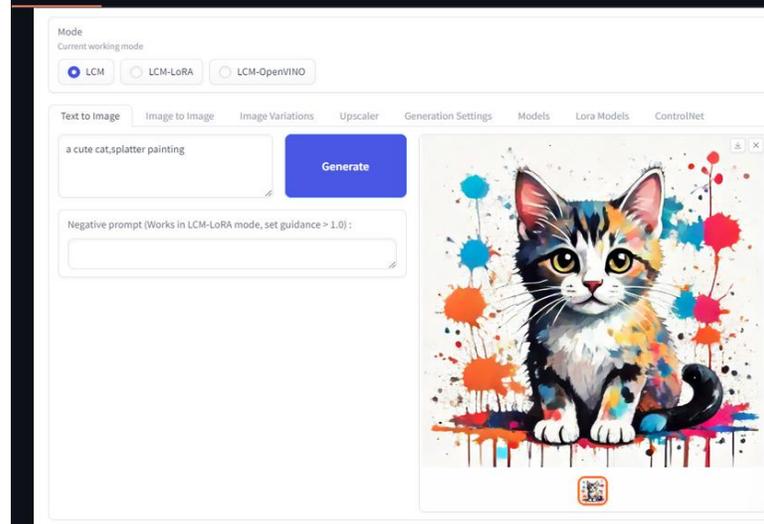


FastSD CPU  

 #6 Repository Of The Day

FastSD CPU is a faster version of Stable Diffusion on CPU. Based on [Latent Consistency Models](#) and [Adversarial Diffusion Distillation](#).

[README](#)  MIT license



# 事前アンケートお願い (挙手にてお願いいたします)

- ①生成AIをコマンドラインで使っている人、使ってみたい人？
- ②ローカル生成AIで無料API(llama.cpp,llamafile,Ollama)を使いたい人？
- ③IntelやAMD、AppleのMetalのNVIDIA以外のGPUで生成AI使いたい人？
- ④ローカルLLMのRAGやFinetuningに興味のある人？(KOF2025で予定)

今日のセミナーのペース時間配分と今後の参考とさせて下さい

# 生成AIとは?

- テキストなど 小さい情報から画像、動画、文章などを生成
- 短時間で複雑なデータを生成可能で生産性が良い
- 複雑な計算が必要なので高性能なCPU,GPU,NPUが必要
- 大規模言語モデル(Large Language models)が有名
- 最近ではAndroidやiPhoneでもOK
- 他にも画像生成AIなど  
定番モデル pony diffusion、Animage XL他  
参考としあき wiki [https://wikiwiki.jp/sd\\_toshiaki/SDXLモデル](https://wikiwiki.jp/sd_toshiaki/SDXLモデル)



# 最近GoogleがGemini CLIを出してきました

- 毎分60回、1日1,000回までのLLMリクエストが無料
- 無料アカウントでの使用はGoogleの学習用に利用される???
- 高性能ツールとしてユーザー数増えそう
- ただしコマンドラインはコスト優先で擬人化しなければ性能不問かも...

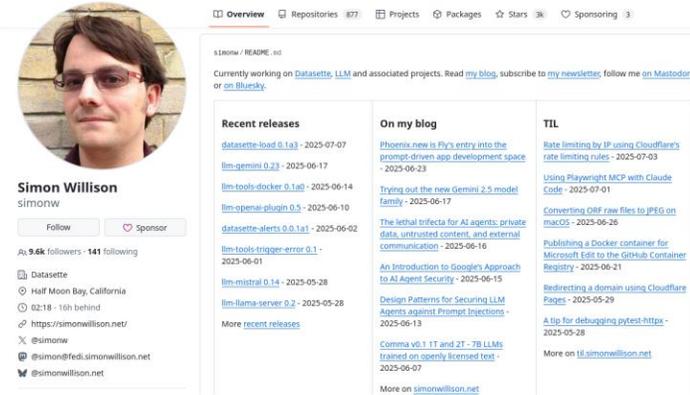
Gemini CLI: オープンソース AI エージェント

2025年6月26日

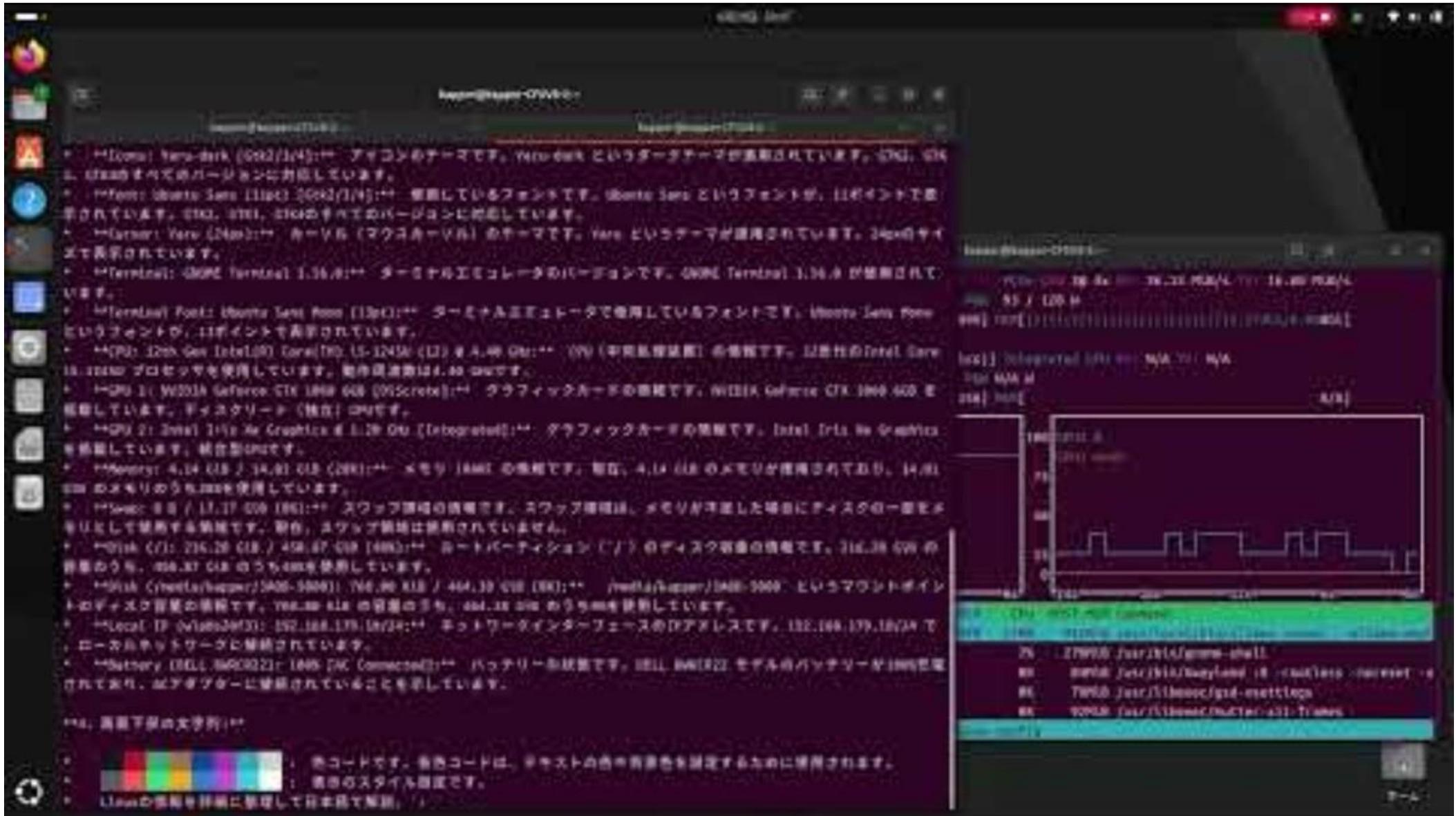


# SimonW/LLMとはそもそもなんぞや？

- Simon Willisonさんが作ったコマンドラインLLMツール。
- 星の数ほどありそうな、コマンドラインLLMツールの一つ。世界中で広く使用
- **多数のプラグインとOS対応**で広く使われている。無料APIに公式対応
- 挙動がシンプルで短いコマンドで使用可能なのが長所。**Shell script, Awk, Python**で拡張すると凄く応用性が高まる
- 基本的な使い方は
  - ① LLMに情報をインプットして出力する方法。日本語で指示や翻訳など
  - ② コマンドの出力をLLMに**パイプ**で送り解析、日本語化など
  - ③ バッチファルやAWKなどに組み込んで自動化
  - ④ PythonやJavascriptに組み込んで**多数の機能をAI化**
  - ⑤ Whisper.cppや音声合成AIと連携して**AIボイス出力**
- Pythonで書かれているが**依存関係が少なくNetBSDでも動作確認済**
- Python APIとしてimport llmと直接プログラムで叩いて使用も可能
- AIオートサジェスチョン機能としてはFish Shell-AIを使った方が良いかも



# デモ動画をYoutubeで公開中



# デモ動画をYoutubeで公開中



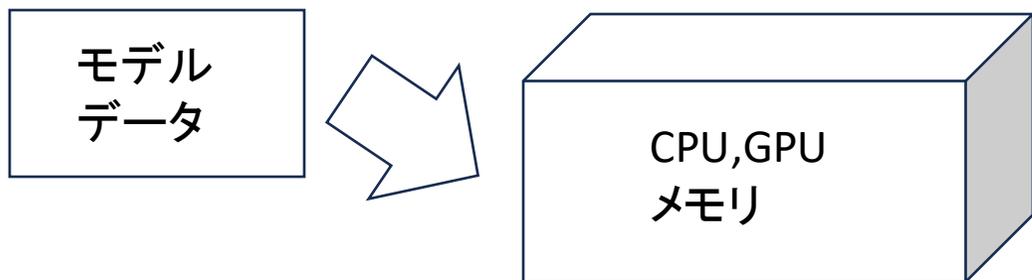
```
7月13日 08:00  
kapper@kapper-CFSH2-21 -  
kapper@kapper-CFSH2-21 -  
(new) kapper@kapper-CFSH2-21:~$ llm -n llama3.2:latest 'WindowsよりLinuxが優れている点を究極に長文で1文で説明' > /tmp/llm_output.txt  
llm && open_jtalk -x /var/lib/ncab/dlc/open-jtalk/ncab/ -n /usr/share/hts-voice/nel/nel_normal.htsvoice -ow /tmp/output.wav < /tmp/llm_output.txt && aplay --quiet /tmp/output.wav
```

# SimonW/LLMはとにかく応用力が無限大

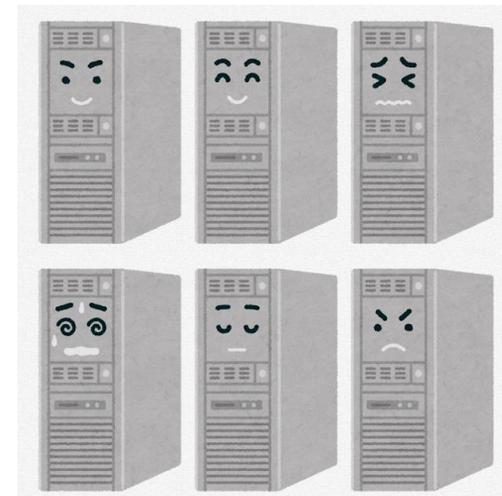
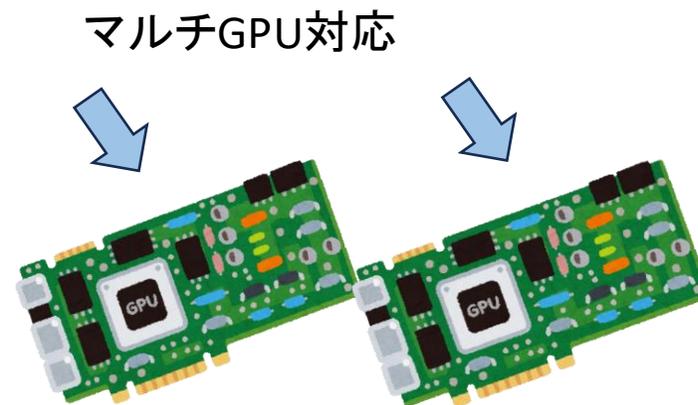
- コマンドにLLM。たったこれだけで何でも出来てしまう
- まだKapperの想像力がSimonW/LLMに追いついていませんw
- 皆さんの想像力でコマンドラインを無敵に改造しませんか？

# LLM動作イメージ

- 一旦モデルをメモリの中に入れる  
(モデルサイズのメモリが必要)



- CPU・GPUで計算

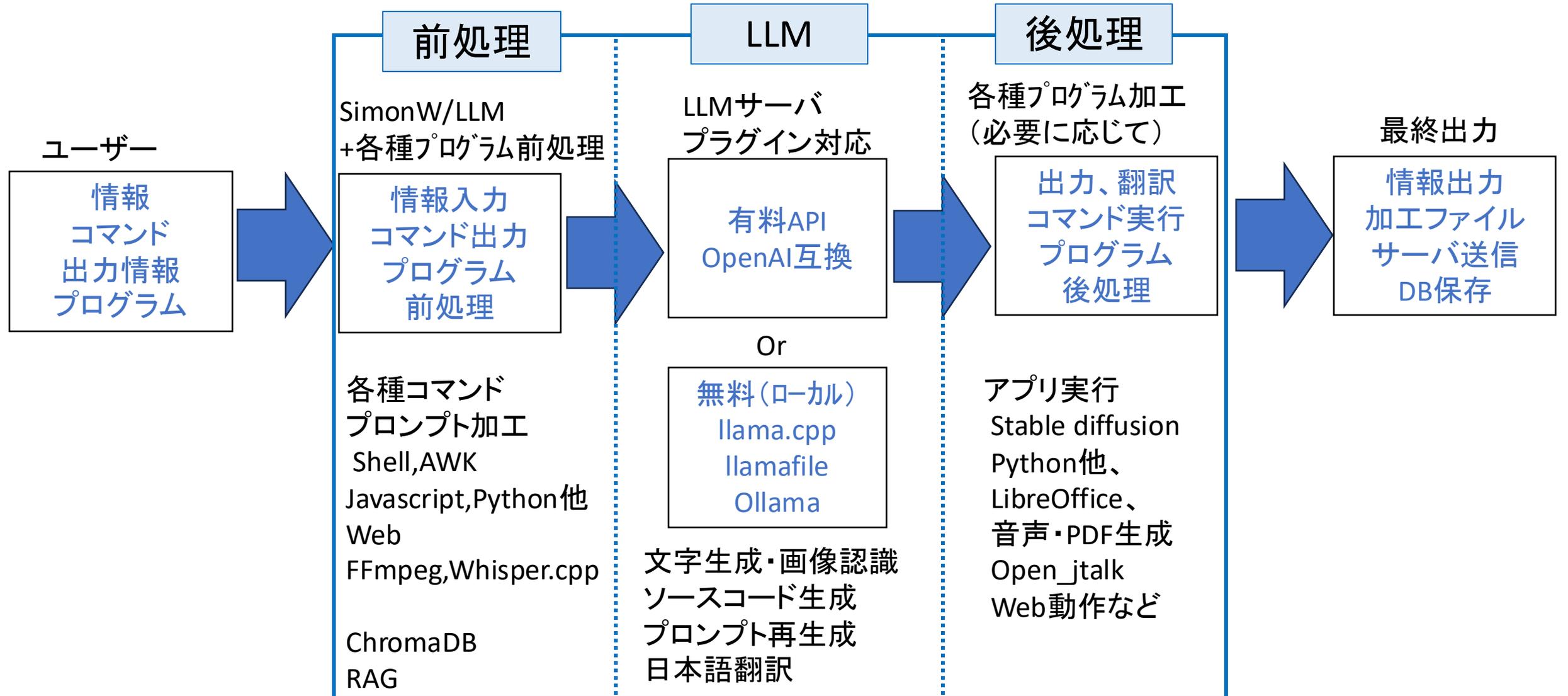


- LoRaがあれば追加



- 結果出力

# SimonW/LLMのイメージ図



繰り返し実行しても良い(時間がかかります)

フルスクラッチでPythonでLLMプログラムを組んだ方が速いかも

# SimonW/LLMで出来る面白いコマンドライン革命

- ①コマンドの入力、出力を日本語に変換。
- ②エラーメッセージをLLMで解析、日本語で対応
- ③スクリプトで分岐、変数など自動作成して簡単なAIプログラム作成
- ④犬・猫、各種キャラクターのAI擬人化出力(高性能モデル推奨)
- ⑤Web(Javascript,Python)に組み込んで自作AIサーバ
- ⑥FFmpeg,Whisper.cpp,Open\_jtalkなどと連携して音声入力と出力
- ⑦足りない機能はJavascriptやPythonから借りて、実は何でも出来る

まだ想像力の限りコマンドラインで生成AIは使えそうです。  
フルスクラッチでPythonコードを組むには負けますが、シンプル簡単です。  
ローカル生成AIで無料APIサーバを使えば、トークン使いたい放題

# SimonW/LLMのインストール

- Python3が必須です。pipx もしくはpython3-venv環境を使用

```
python3 -m venv env/  
source env/bin/activate  
pip install llm
```

NetBSDの場合は

```
pkgin install python3.11-pip rust  
pip3.11 install llm  
llm install llm-llamafie  
llm keys set openai
```

- OpenAIサーバを使う場合は

```
export OPENAI_API_KEY='(キー)'
```

を.bashrcか.bash\_profileなどに追記して実行

- llamafieやllama.cppのローカルAPIサーバはキー不要だがダミーの入力が必要

# SimonW/LLMをOllamaサーバで使う

- Ollamaをインストール。  
Linuxの場合、  
`curl -fsSL https://ollama.com/install.sh | sh`  
MacとWindowsの場合は公式HPよりダウンロード
- Ollamaモデルの実行、例えば①～③のどれか  
モデル① `ollama run llama3.2:latest` (要VRAM 2.0GB)  
モデル② `ollama run gemma3n:e2b` (要VRAM 5.6GB)  
モデル③ `ollama run gemma3n:e4b` (要VRAM 7.5GB)
- llm-ollamaプラグインのインストール(APIキー不要)  
`llm install llm-ollama`
- SimonW/llmを実行  
`llm -m llama3.2:latest "NetBSDの長所"`

注)Ollamaは起動と同時に  
`http://localhost:11435`を  
APIサーバとして起動します

# SimonW/LLMをllamafile,llama.cppサーバで使う

- llm-llamafileを入れる  
`python3 -m venv venv`  
`source venv/bin/activate`  
`llm install llm-llamafile`

- ① 適当なllamafileをダウンロードして実行  
`chmod +x google_gemma-3-4b-it-Q6_K.llamafile`  
`./google_gemma-3-4b-it-Q6_K.llamafile`

- ② もしくはllama.cppをビルドしてサーバーモードで起動  
`./llama-server -m (モデル名) -ngl 999`

- ダミーAPIキー登録。キーは適当で何でも良い  
`llm keys set openai`

- SimonW/llmを実行  
`llm -m llamafile "NetBSDの長所"`

注)llama.cppとllamafileは  
起動と同時に  
<http://localhost:8080> を  
APIサーバとして起動します



# llamafile, llama.cppの-ngl オプションとは？

- -ngl とはLLMモデルのレイヤー数をどれだけGPUに置くか？という設定項目。CPUとGPUの比率を調整できる
- VRAMが少ない場合-ngl 値を下げる事でCPUのメインメモリに置けるが当然処理速度は大幅に低下する
- VRAMを節約したい場合、-cオプションでモデルが設定出来るトークン長を調整する事も可能。default:4096 (VRAMによる)

```
load_tensors: offloading 28 repeating layers to GPU
load_tensors: offloading output layer to GPU
load_tensors: offloaded 29/29 layers to GPU
load_tensors:   CPU_Mapped model buffer size =   399.23 MiB
load_tensors:   SYCL0 model buffer size =  2009.36 MiB
```

# Ollama, llamafile, llama.cppサーバを止めてメモリを開放する

- Ollamaサーバを止めてメモリを開放  
ollama stop llama3.2:latest (←モデル名)
- llamafile, llama.cppサーバを止めてメモリを開放  
Ctrl+Zでアプリを停止  
ps でプロセス番号を確認  
kill -9 (プロセス番号) でllama-serverを停止

```
^Z
[1]+  停止                  ./build/build/bin/llama-server -m ../Llama-3.2-3B-Instruct-Q4_K_L.gguf
kapper@kapper-CFSV8-2:~/llama.cpp$ ps
  PID TTY          TIME CMD
  3250 pts/0    00:00:00 bash
  32610 pts/0    00:00:18 llama-server
  32661 pts/0    00:00:00 ps
kapper@kapper-CFSV8-2:~/llama.cpp$ kill -9 32610
kapper@kapper-CFSV8-2:~/llama.cpp$ ps
  PID TTY          TIME CMD
  3250 pts/0    00:00:00 bash
  32681 pts/0    00:00:00 ps
[1]+  強制終了              ./build/build/bin/llama-server -m ../Llama-3.2-3B-Instruct-Q4_K_L.gguf
```

# 簡単なSimonW/LLM使い方例

- `llm "NetBSDの説明"` ●通常のLLM(OpenAI API)
- `llm "Ubuntuの長所" -m llama3.2:latest` ●ローカルLLMモデル使用時
- `llm -c "会話しましょう" -m llama3.2:latest` ●チャットモード
- `llm -m llama3.2:latest "Linuxでストレージを表示. コマンドのみ出力."` ●モデル選択
- `cat sample.py | llm -s "日本語でソースコードを解析"` ●出力をパイプで処理+システムモード
- `echo "LinuxでSSHポート（22番）を開けるファイアウォール設定のコマンドとその説明を日本語で教えてください" | llm -m llama3.2:latest` ●Linuxコマンド解説
- `man curl | llm -s "以下を要約して、日本語で使用方法と代表的なオプションを簡潔に説明してください。" -m llama3.2:latest` ●Manを日本語訳と要約
- `cat script.sh | llm -s "このシェルスクリプトの問題点を修正して、修正後の完全なコードだけを出力してください" -m llama3.2:latest > fixed_script.sh` ●>で結果をファイルに出力

# SimonW/LLMのオプション

## ① プロンプト実行・対話系

- `prompt`: プロンプトを実行(基本コマンド)
- `chat`: モデルと対話モードでやり取り
- `Templates`: プロンプトテンプレートを管理

## ② モデル管理系

- `models`: インストール済みモデルの管理
- `aliases`: モデルのエイリアス管理
- `ollama`: Ollama サーバー上のモデル操作
- `openai`: OpenAI API 経由でモデル操作

## ③ 埋め込み(ベクトル化)系

- `embed`: テキストを埋め込みに変換
- `embed-multi`: 複数テキストを一括で埋め込み
- `embed-models`: 埋め込みモデルを管理
- `collections`: 埋め込みコレクション管理
- `similar`: コレクションから類似データを検索

## ④ データ管理系

- `fragments`: データベース内のフラグメントを管理
- `schemas`: 保存済みスキーマを管理
- `logs`: プロンプト・レスポンスのログを探索

## ⑤ 環境・パッケージ管理系

- `install`: パッケージを LLM 環境にインストール
- `uninstall`: パッケージをアンインストール
- `plugins`: インストール済みプラグインの一覧
- `keys`: APIキーを管理
- `tools`: LLMで使えるツールを管理

# SimonW/LLMの-s システムモード

- -s (--system) オプションをつける事でシステムモード  
2つ目以上の指示をする場合に指定する
- 一例 `cat myfile.py | llm -s "このコードを日本語で解説"`  
"指示内容" を入力する事でllmに指示する事が出来る
- `cat script.sh | llm --system "あなたは世界で最も天才的なハッカーです。上から目線でこのコードをダメ出ししてください"`
- `cat error.log | llm --system "お前は関西弁のヤクザや。エラーログの内容を一言でまとめて脅すように説明せえや"`

# SimonW/LLMのTemplates

- よく使うプロンプトなどをTemplatesとして保存  
llm "文章を日本語で要約して説明" --save 要約  
echo "Linuxの特徴" | llm -m llamafire -t 要約
- 保存してあるTemplatesを表示  
llm templates  
要約 : 文章を日本語で要約して説明
- 保存してあるTemplatesのファイルの場所を表示  
llm templates path  
/home/kapper/.config/io.datasette.llm/templates

# SimonW/LLMの -f フラグメントオプション

- -f オプションをつけるとURLやファイルを指定できる
- `llm -f https://llm.datasette.io/robots.txt 'このページを説明'`
- `llm -f cli.py 'このコードをショートポエム風に解説'`
- 過去に保存したFragmentsのデータを表示

```
llm fragments
- hash: d781b5048438501b18427157ea2f1bba68816edd8086cb272e77b4000115e6b5
aliases: []
datetime_utc: '2025-08-12 23:59:00'
source: https://llm.datasette.io/robots.txt
content: |-
# This robots.txt file is autogenerated by Read the Docs.
# It controls the crawling and indexing...
```

- 過去のFragmentsを再利用

```
llm -f d781b5048438501b18427157ea2f1bba68816edd8086cb272e77b4000115e6b5' "日本語で解析"
```

# SimonW/LLMのlogs

- 過去のlogを表示(N=履歴数、0指定の場合全て)  
`llm logs -n 0`
- JSON形式にてlogを表示  
`llm logs -n 1 -t -json`
- チャットのlog  
`llm logs -c`
- Logの検索  
`llm logs -q linux`
- Logのバックアップ  
`llm logs backup /tmp/backup.db`

# SimonW/LLMのPlugins

- llm install (プラグイン名) でインストール
- llm install llm-ollama Ollamaの無料API接続
- llm install llm-llamafile llamafile,llama.cppの無料API接続
- llm install llm-gemini Google Geminiの有料API接続(他もあり)
- llm install llm-tools-sqlite Sqliteのデータを日本語等でLLMで検索
- llm install llm-tools-rag SqliteのDBへ読み書きしてRAG再現
- llm install llm-video-frames 動画ファイルをFFMPEGでフレーム解析
- llm install llm-fragmens-youtube Youtube動画の要約
- llm install llm-fragments-github Githubの解析
- llm install llm-python llmの操作をPythonから実行
- <https://simonwillison.net/tags/plugins/>  
<https://llm.datasette.io/en/stable/plugins/directory.html>

# 人気あるローカルLLMモデルデータ

- 主にHuggingfaceなどからGGUFファイルをダウンロードしてmodels以下の各フォルダに保存して使用
- 日本語対応: llama3.2 1B,3B(古いけどサイズが小さく1.5GB相当から)
- 高性能: Gemma3n n2b(5.6GB)、n4b(7.5GB)、Gemma3 4B(3.3GB)
- 中国系: Deepseek-r1(1.1~404GB)、Qwen3(1.2~142GB) <think>で高性能化
- 最軽量: **Qwen3 1.7B Q4\_K\_M(1.28GB)**、**Gemma3 1B Q4\_1(764MB)**、Gemma3 270M Q4 (253MB)

日本語可モデル。低スペックPC、低VRAMに特にオススメ

- マルチモーダル: Qwen2.5vl(3.2~49GB)、llava (4.7~20GB)
- コーダー系: Qwen2.5-coder(1.0~20GB)
- ベクトル埋め込みモデル: mxbai-embed-large

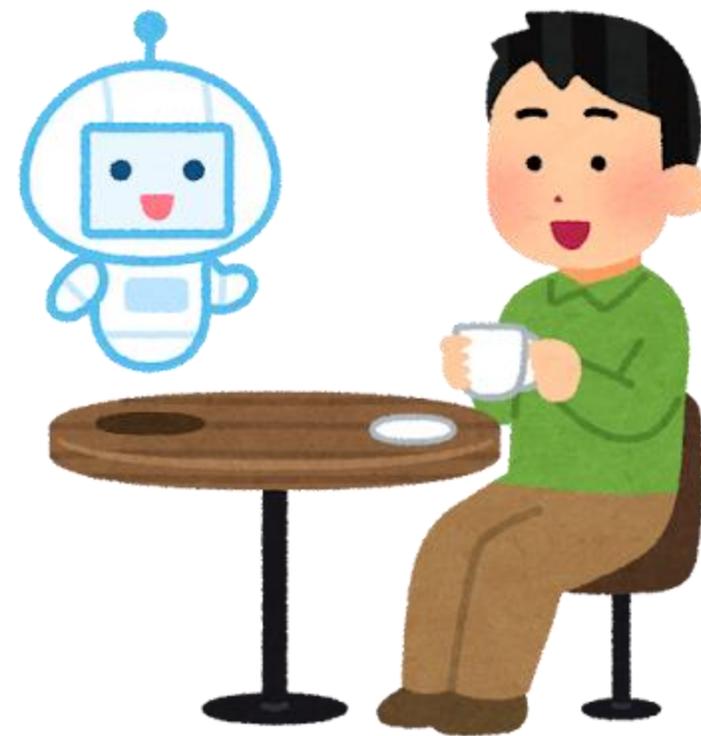
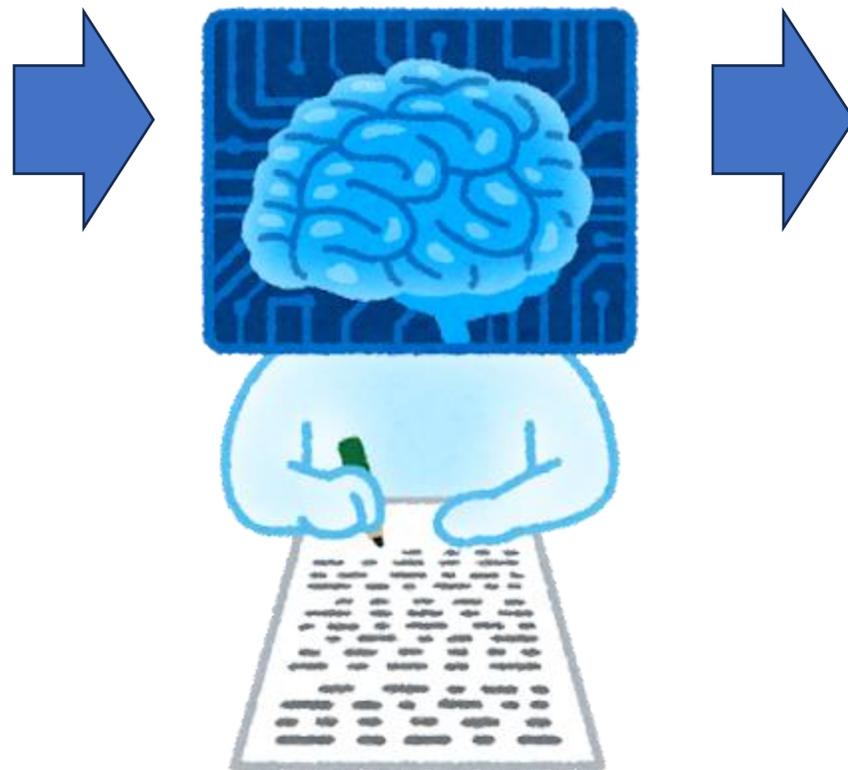
- ~1GB 殆ど会話出来ない、1~1.5GB 辛うじて会話が成立、  
1.5~2.0GB ギリギリ日本語可。英語混じり、2.0~3.0GB 日本語で多少会話出来る、  
3.0~4.0GB この位あると普通、5.0~6.0GB 犬猫レベルの簡単な擬人化、  
7.0~8.0GB 超最低限の人格の語尾を再現

# 生成AIでこんな使い方



①音声入力  
Whisper.cpp  
Windowsキー+H  
など

②生成AI文章作成  
ChtGPT  
llama.cpp  
など



③音声合成  
Open-Jtalkなど

# SimonW/LLMの練習①基本コマンド

- `llm -m llama3.2:latest "2*2"`  
`2 * 2 = 4`

- `llm -m llama3.2:latest "NetBSDの長所"`

NetBSDは、オペレーティングシステムの開発と使用にさまざまな利点を提供する、オープンソースなオペレーティングシステムです。以下に、NetBSDの長所がまとめられています。

- `export LLM_MODEL=llama3.2:latest`  
`llm "モデルのデフォルト設定"`

- `llm -f https://llm.datasette.io/robots.txt 'このページを日本語で説明'`

- `llm -f setup.py 'このプログラムを日本語で解析'`

# SimonW/LLMの練習②echoやパイプ,> ファイル出力

- echo "現在のディレクトリ以下のすべての空フォルダを削除するbashコマンドを教えてください" | llm -m llama3.2:latest  
find . -type d -empty | xargs rm -i
- cat \*.sh | llm -m llama3.2:latest --system "これらのシェルスクリプトの概要と全体構成を説明してください"  
> 全体説明.md

**\*\*README\*\***

=====

**\*\*JTalk Voice Assistant\*\***

-----

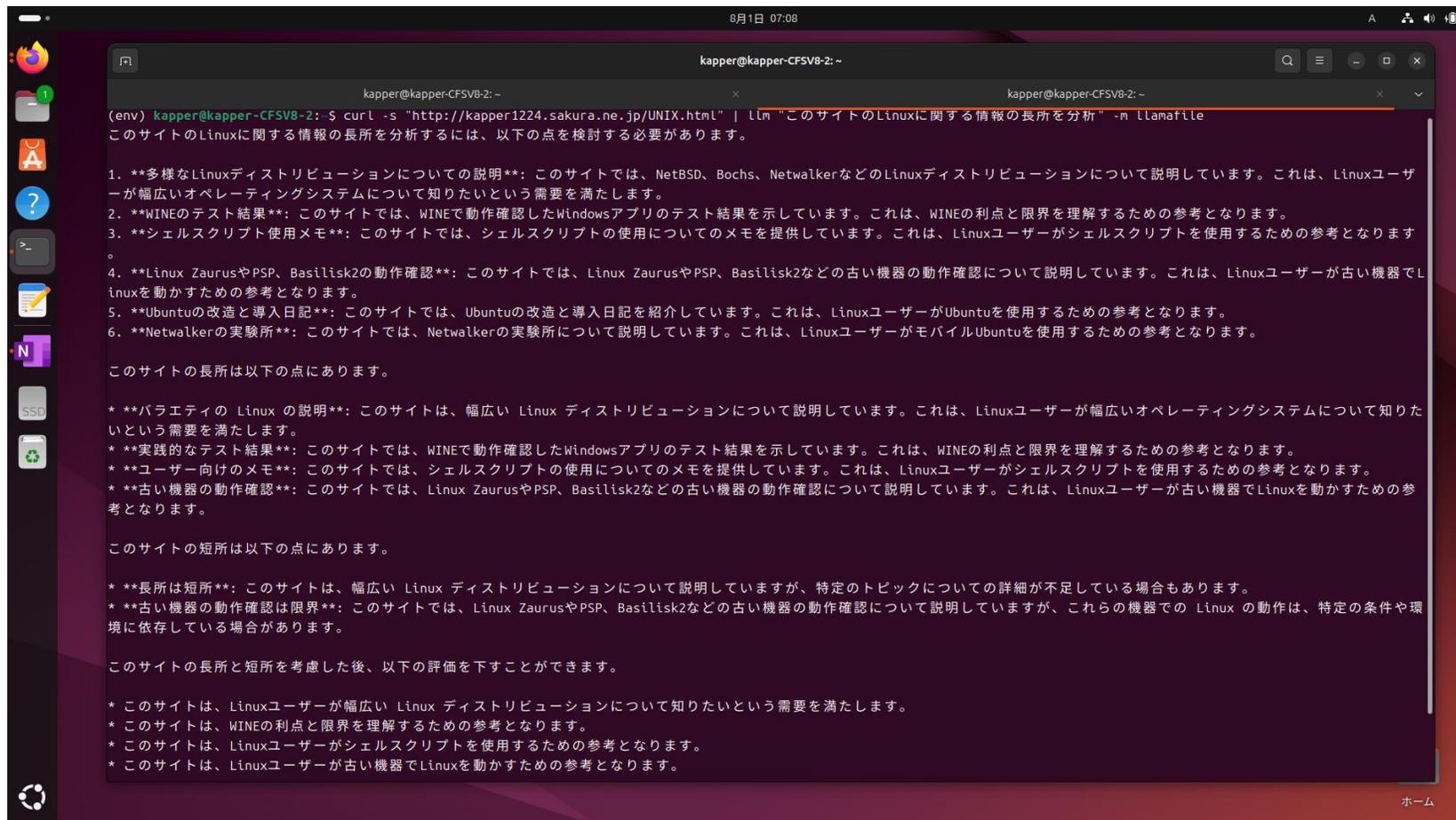
このコードは、現在の日付と時間を英語で発声するための shスクリプトです。

### how to use

# SimonW/LLMの練習③curl(カール)+Web情報

- CurlコマンドでWebから各種データを入手して解析

`curl -s "http://kapper1224.sakura.ne.jp/UNIX.html" | llm "このサイトのLinuxに関する情報の長所を分析" -m llamafire`



```
(env) kapper@kapper-CFSV8-2: ~$ curl -s "http://kapper1224.sakura.ne.jp/UNIX.html" | llm "このサイトのLinuxに関する情報の長所を分析" -m llamafire
このサイトのLinuxに関する情報の長所を分析するには、以下の点を検討する必要があります。

1. **多様なLinuxディストリビューションについての説明**：このサイトでは、NetBSD、Bochs、NetwalkerなどのLinuxディストリビューションについて説明しています。これは、Linuxユーザーが幅広いオペレーティングシステムについて知りたいという需要を満たします。
2. **WINEのテスト結果**：このサイトでは、WINEで動作確認したWindowsアプリのテスト結果を示しています。これは、WINEの利点と限界を理解するための参考となります。
3. **シェルスクリプト使用メモ**：このサイトでは、シェルスクリプトの使用についてのメモを提供しています。これは、Linuxユーザーがシェルスクリプトを使用するための参考となります。
4. **Linux ZaurusやPSP、Basilisk2の動作確認**：このサイトでは、Linux ZaurusやPSP、Basilisk2などの古い機器の動作確認について説明しています。これは、Linuxユーザーが古い機器でLinuxを動かすための参考となります。
5. **Ubuntuの改造と導入日記**：このサイトでは、Ubuntuの改造と導入日記を紹介しています。これは、LinuxユーザーがUbuntuを使用するための参考となります。
6. **Netwalkerの実験所**：このサイトでは、Netwalkerの実験所について説明しています。これは、LinuxユーザーがモバイルUbuntuを使用するための参考となります。

このサイトの長所は以下の点にあります。

* **バラエティのLinuxの説明**：このサイトは、幅広いLinuxディストリビューションについて説明しています。これは、Linuxユーザーが幅広いオペレーティングシステムについて知りたいという需要を満たします。
* **実践的なテスト結果**：このサイトでは、WINEで動作確認したWindowsアプリのテスト結果を示しています。これは、WINEの利点と限界を理解するための参考となります。
* **ユーザー向けのメモ**：このサイトでは、シェルスクリプトの使用についてのメモを提供しています。これは、Linuxユーザーがシェルスクリプトを使用するための参考となります。
* **古い機器の動作確認**：このサイトでは、Linux ZaurusやPSP、Basilisk2などの古い機器の動作確認について説明しています。これは、Linuxユーザーが古い機器でLinuxを動かすための参考となります。

このサイトの短所は以下の点にあります。

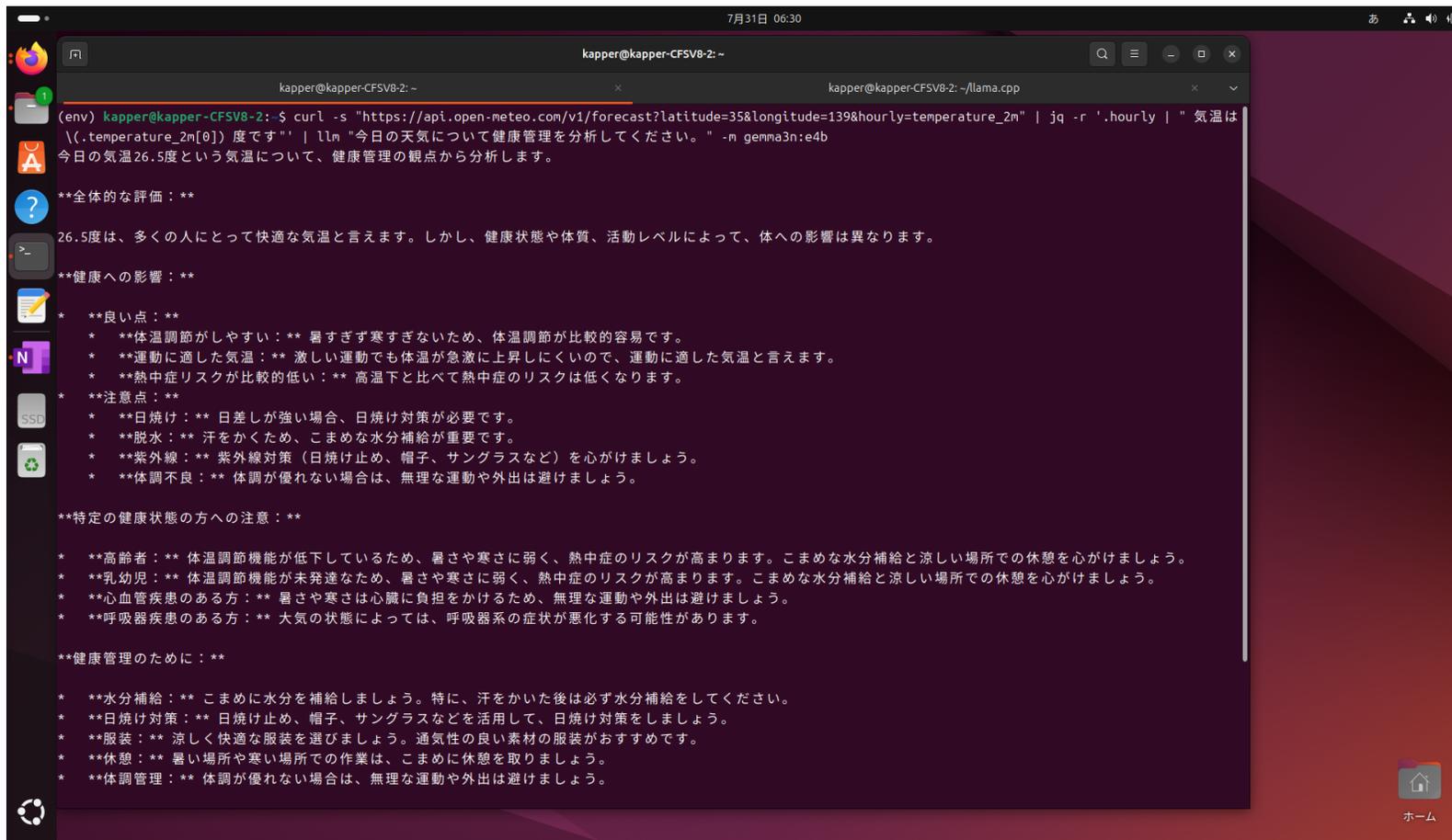
* **長所は短所**：このサイトは、幅広いLinuxディストリビューションについて説明していますが、特定のトピックについての詳細が不足している場合もあります。
* **古い機器の動作確認は限界**：このサイトでは、Linux ZaurusやPSP、Basilisk2などの古い機器の動作確認について説明していますが、これらの機器でのLinuxの動作は、特定の条件や環境に依存している場合があります。

このサイトの長所と短所を考慮した後、以下の評価を下すことができます。

* このサイトは、Linuxユーザーが幅広いLinuxディストリビューションについて知りたいという需要を満たします。
* このサイトは、WINEの利点と限界を理解するための参考となります。
* このサイトは、Linuxユーザーがシェルスクリプトを使用するための参考となります。
* このサイトは、Linuxユーザーが古い機器でLinuxを動かすための参考となります。
```

# SimonW/LLMで練習④jqコマンドでjson解析

- `curl -s "https://api.open-meteo.com/v1/forecast?latitude=35&longitude=139&hourly=temperature_2m" | jq -r '.hourly | "気温は ¥(.temperature_2m[0]) 度です"' | llm "今日の天気について健康管理を分析してください。" -m gemma3n:e4b`



```
(env) kapper@kapper-CFSV8-2:~$ curl -s "https://api.open-meteo.com/v1/forecast?latitude=35&longitude=139&hourly=temperature_2m" | jq -r '.hourly | "気温は ¥(.temperature_2m[0]) 度です"' | llm "今日の天気について健康管理を分析してください。" -m gemma3n:e4b
今日の気温26.5度という気温について、健康管理の観点から分析します。

**全体的な評価：**
26.5度は、多くの人にとって快適な気温と言えます。しかし、健康状態や体質、活動レベルによって、体への影響は異なります。

**健康への影響：**
* **良い点：**
  * **体温調節がしやすい：** 暑すぎず寒すぎないため、体温調節が比較的容易です。
  * **運動に適した気温：** 激しい運動でも体温が急激に上昇しにくいので、運動に適した気温と言えます。
  * **熱中症リスクが比較的低い：** 高温下と比べて熱中症のリスクは低くなります。
* **注意点：**
  * **日焼け：** 日差しが強い場合、日焼け対策が必要です。
  * **脱水：** 汗をかくため、こまめな水分補給が重要です。
  * **紫外線：** 紫外線対策（日焼け止め、帽子、サングラスなど）を心がけましょう。
  * **体調不良：** 体調が優れない場合は、無理な運動や外出は避けましょう。

**特定の健康状態の方への注意：**
* **高齢者：** 体温調節機能が低下しているため、暑さや寒さに弱く、熱中症のリスクが高まります。こまめな水分補給と涼しい場所での休憩を心がけましょう。
* **乳幼児：** 体温調節機能が未発達なため、暑さや寒さに弱く、熱中症のリスクが高まります。こまめな水分補給と涼しい場所での休憩を心がけましょう。
* **心血管疾患のある方：** 暑さや寒さは心臓に負担をかけるため、無理な運動や外出は避けましょう。
* **呼吸器疾患のある方：** 大気の状態によっては、呼吸器系の症状が悪化する可能性があります。

**健康管理のために：**
* **水分補給：** こまめに水分を補給しましょう。特に、汗をかいた後は必ず水分補給をしてください。
* **日焼け対策：** 日焼け止め、帽子、サングラスなどを活用して、日焼け対策をしましょう。
* **服装：** 涼しく快適な服装を選びましょう。通気性の良い素材の服装がおすすめです。
* **休憩：** 暑い場所や寒い場所での作業は、こまめに休憩を取りましょう。
* **体調管理：** 体調が優れない場合は、無理な運動や外出は避けましょう。
```

# SimonW/LLMで練習⑤pdftotextで抽出・解析

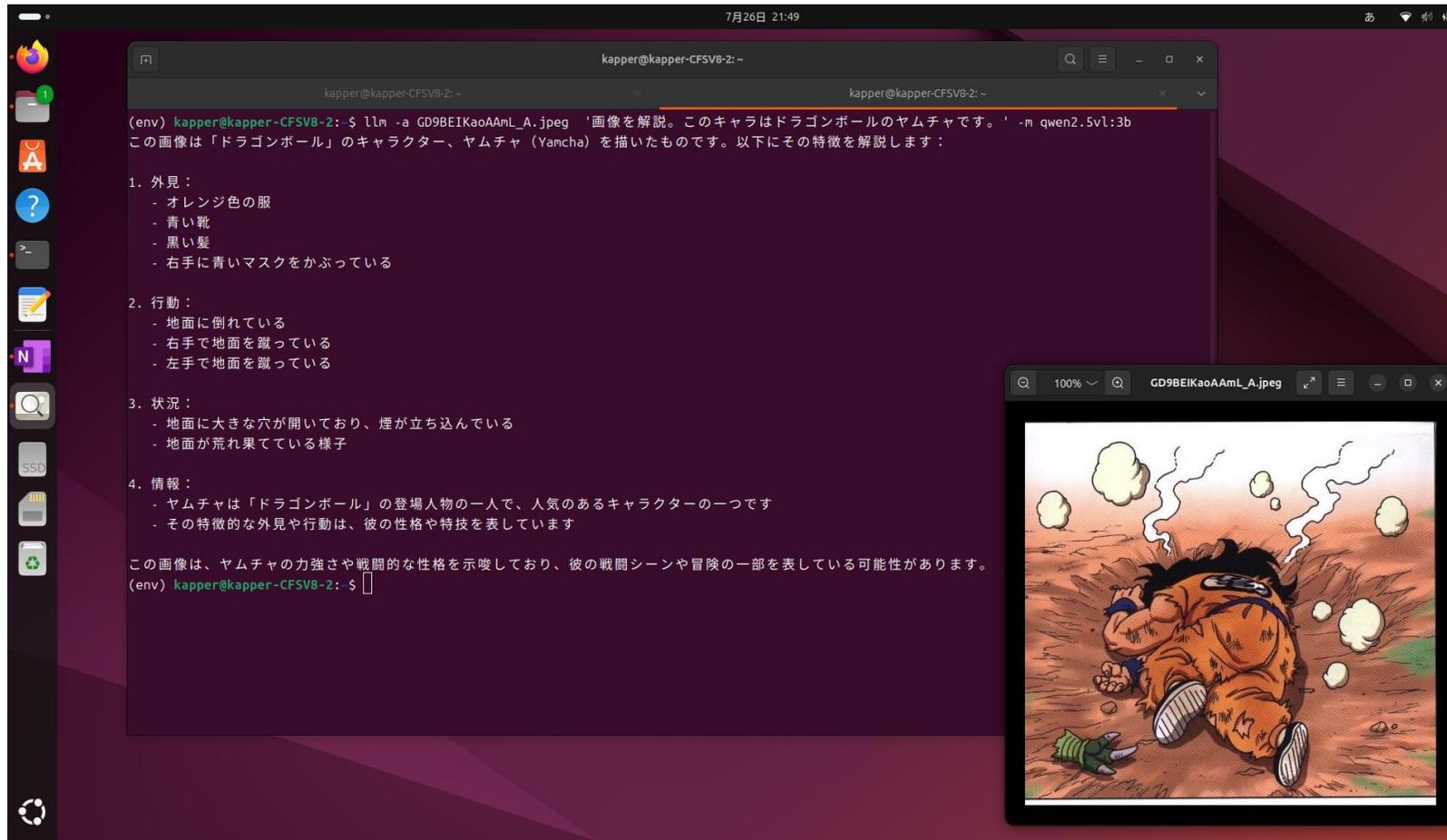
- PDFファイルをpdftotextで変換してSimonW/LLMで解析  
pdftotext -f 1 -l 1 Netwalker¥ osc¥ kyoto2025.pdf - | llm -m llamafire "この文章の改善方法"

```
(env) kapper@kapper-CFSV8-2:~$ pdftotext -f 1 -l 1 Netwalker\ osc\ kyoto2025.pdf - | llm -m llamafire "この文章の改善方法"  
以下は文章の改善方法です。
```

- \* 「初心者向け」は書き物としては適切ですが、セミナー名は「生成AI SimonW/LLMとOllama・llamafire無料APIでコマンドラインをAI革命するセミナー。CPUでもGPUでも。」にし、NetBSDもOKという表現を書き物から削除して「NetBSDもOK」と書き直す。
- \* 「1. 準備」は書き物としては適切ですが、「」に書き換える。
- \* 「2. SimonW/LLMインストール」は書き物としては適切ですが、「」に書き換える。
- \* 「3. SimonW/LLM簡単な使い方」は書き物としては適切ですが、「」に書き換える。
- \* 「4. SimonW/LLMのネタ、応用事例」は書き物としては適切ですが、「」に書き換える。
- \* 「5. ハードウェア、GPU高速化」は書き物としては適切ですが、「」に書き換える。
- \* 「メインテーマ」は書き物としては適切ですが、「」に書き換える。
- \* 「このプレゼンテーション：」は書き物としては適切ですが、「」に書き換える。
- \* 「Slideshare & PDF files」は書き物としては適切ですが、「」に書き換える。
- \* 「publication of my HP」は書き物としては適切ですが、「」に書き換える。
- \* 「http://kapper1224.sakura.ne.jp」は書き物としては適切ですが、「」に書き換える。
- \* 「Gadget Hacking User Group」は書き物としては適切ですが、「」に書き換える。
- \* 「Speaker：Kapper」は書き物としては適切ですが、「」に書き換える。

# SimonW/LLMの練習⑥マルチモーダル

- `llm -a GD9BEIKaoAAmL_A.jpeg '画像を解説。このキャラはドラゴンボールのヤムチャです。' -m qwen2.5vl:3b`



The screenshot shows a terminal window on a Linux desktop environment. The terminal displays the command `(env) kapper@kapper-CFSV8-2:~$ llm -a GD9BEIKaoAAmL_A.jpeg '画像を解説。このキャラはドラゴンボールのヤムチャです。' -m qwen2.5vl:3b` and its output. The output is a detailed analysis of the image, structured into four numbered sections: 1. 外見 (Appearance), 2. 行動 (Action), 3. 状況 (Situation), and 4. 情報 (Information). Below the analysis, there is a summary sentence and the prompt is repeated.

```
(env) kapper@kapper-CFSV8-2:~$ llm -a GD9BEIKaoAAmL_A.jpeg '画像を解説。このキャラはドラゴンボールのヤムチャです。' -m qwen2.5vl:3b
この画像は「ドラゴンボール」のキャラクター、ヤムチャ (Yamcha) を描いたものです。以下にその特徴を解説します：

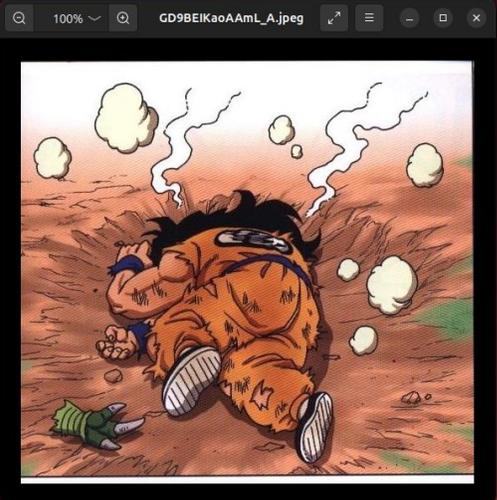
1. 外見：
- オレンジ色の服
- 青い靴
- 黒い髪
- 右手に青いマスクをかぶっている

2. 行動：
- 地面に倒れている
- 右手で地面を蹴っている
- 左手で地面を蹴っている

3. 状況：
- 地面に大きな穴が開いており、煙が立ち込んでいる
- 地面が荒れ果てている様子

4. 情報：
- ヤムチャは「ドラゴンボール」の登場人物の一人で、人気のあるキャラクターの一つです
- その特徴的な外見や行動は、彼の性格や特技を表しています

この画像は、ヤムチャの力強さや戦闘的な性格を示唆しており、彼の戦闘シーンや冒険の一部を表している可能性があります。
(env) kapper@kapper-CFSV8-2:~$
```



# SimonW/LLMのネタ的使い方①擬人化

- `sudo tcpdump -c 10 | llm -m llama3.2 "これを犬と猫たちが会話している様に変換してください"`

```
kapper@kapper-CFSV8-2:~/Downloads$ sudo tcpdump -c 10 | llm -m llama3.2 "これを犬と猫たちが会話している様に変換してください"
[sudo] kapper のパスワード:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on wlo1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
10 packets captured
13 packets received by filter
0 packets dropped by kernel
犬:
11:55:38 なんねん！私も待っててよかったね！
猫:
11:55:39 猫でも待っていたんだ。なーい？あらかじめお時間をいただいた。
犬:
11:55:39 ごめんね！何度も失敗してたから...
猫:
11:55:39 いいえ、いいえ。もうすでにお時間をいただいたんだ。
犬:
11:55:40 そう？あっすがーい！次は我慢してね！
猫:
11:55:40 うわーん！あなたのご指示をもって行こうよ。
犬:
11:55:41 ええ、でも我慢するためにお時間をいただかれたんだね？そうすると我慢できると思います！
猫:
11:55:41 いいえ、いいえ。もうすでにお時間をいただいたんだ。
など（続き）
kapper@kapper-CFSV8-2:~/Downloads$
```

# SimonW/LLMのネタ的使い方②Historyで精神鑑定

- history | tail -n 20 | llm "このコマンド履歴は、とある変人の心理状態と今日の行動を詳細に表しています。この人物の性格、趣味、そして今日最も興味を持っていたことを、精神科医の視点から分析し、結論としてこの人物に最適な治療法を提案してください。" -m gemma3n:e2b

```
kapper@kapper-CFSV8-2: ~  
(env) kapper@kapper-CFSV8-2:~$ history | tail -n 20 | llm "このコマンド履歴は、とある変人の心理状態と今日の行動を詳細に表しています。この人物の性格、趣味、そして今日最も興味を持っていたことを、精神科医の視点から分析し、結論としてこの人物に最適な治療法を提案してください。" -m gemma3n:e2b  
このコマンド履歴は、とある変人の心理状態と今日の行動を詳細に表しています。この人物の性格、趣味、そして今日最も興味を持っていたことを、精神科医の視点から分析し、結論としてこの人物に最適な治療法を提案してください。  
  
**分析と提案:**  
  
このコマンド履歴は、非常に多様な操作を行っており、いくつかの興味深いパターンが見られます。  
  
* **技術的な探索:** `llama3.2:latest` や `ollama run`、`nano` など、様々なツールやソフトウェアを試していることから、技術的な探求心と学習意欲が伺えます。特に、LLM（大規模言語モデル）の利用に積極的であることは、情報収集や知識獲得を重視していることを示唆します。  
* **コードの作成と分析:** `jtalk.py` や `tiny_sd.py` の作成、そしてそれらのコードをLLMに分析させる試みは、プログラミングスキルとコード理解への関心を示しています。LLMによるコードの概要、使い方、注意点といった分析は、効率的な開発や保守を目的としていると考えられます。  
* **システム管理と自動化:** `sudo tlp auto` や `sudo powertop --auto-tune` は、システムリソースの最適化や自動化に関心があることを示唆します。  
* **情報収集と整理:** `ls`、`cat`、`history`、`tail` など、ファイルやコマンド履歴を操作する操作は、情報収集と整理を重視していることを示唆します。  
* **心理的な探求:** `history | tail -n 20 | llm ...` は、自身のコマンド履歴を分析し、心理的な状態を把握しようとしていることを示しています。これは、自己理解や問題解決への意識の高さと関連している可能性があります。  
  
**精神科医の視点からの分析と提案:**  
  
このコマンド履歴から、以下の性格特性や潜在的な心理状態が推測できます。  
  
* **好奇心旺盛で探求心が高い:** 新しい技術や情報を積極的に試すことから、好奇心と探求心が高いと考えられます。  
* **学習意欲がある:** LLMの利用やコードの分析など、継続的に学習しようとする意欲が感じられます。  
* **問題解決能力が高い:** 複雑な問題を解決するために、様々なツールや手法を試すことから、問題解決能力が高いと考えられます。  
* **自己分析への意識が高い:** コマンド履歴を分析し、自身の心理状態を把握しようとする試みは、自己分析への意識が高いことを示しています。  
* **潜在的な不安やストレス:** コマンド履歴の分析に、ある程度の時間を費やしていることから、潜在的な不安やストレスを感じている可能性も考えられます。  
  
**最適な治療法:**  
  
この人物に最適な治療法は、個々の状況によって異なりますが、以下の可能性が考えられます。  
  
* **認知行動療法:** 自己分析や問題解決に役立つ認知行動療法は、潜在的な不安やストレスの軽減に効果的です。
```

# SimonW/LLMの応用①プログラムに組み込む

- ChatGPTの様な賢いLLMでシェルスクリプトやPythonプログラムを作成し、SimonW/LLMを実行する

```
修正後のシェルスクリプト: ai_command.sh

bash
# /bin/bash

# ----- 設定 -----
MODEL="llama2.2"

# ----- 関数定義 -----
function generate_command() {
  echo "💡 日本語で実行したい操作を入力してください:"
  read INPUT_TEXT

  if [ -z "$INPUT_TEXT" ]; then
    echo "❌ 入力为空です。処理を終了します。"
    exit 1
  fi

  PROMPT="次の日本語の操作をLinuxコマンドに変換してください。実行環境はubuntu Linuxです。コマンドは"

  echo "🕒 コマンドを生成中..."

  # llmコマンドで生成
  RESPONSE=$(echo -e "$PROMPT" | llm -m $MODEL)

  # Markdownのコードブロックからコマンド部分を抽出
  # "" で囲まれた部分の最初のブロックだけ抽出
  COMMAND=$(echo "$RESPONSE" | awk '/^```/{flag=1; next} /```/{flag=0} flag')

  if [ -z "$COMMAND" ]; then
    echo "❌ 適切なコマンドが生成されませんでした。"
    echo "生成された結果:"
    echo "$RESPONSE"
    exit 1
  fi

  echo ""
  echo "🌟 生成されたコマンド:"
  echo "$COMMAND"
  echo ""

  # 確認プロンプト
  read -p "このコマンドを実行しますか? (y/n): " CONFIRM

  if [ "$CONFIRM" == "y" ]; then
    echo "🚀 コマンドを実行中..."
  fi
}
```

```
kapper@kapper-CFSV8-2: ~

kapper@kapper-CFSV8-2:~$ ./ai_command.sh
💡 日本語で実行したい操作を入力してください:
このpcの情報を表示
🕒 コマンドを生成中...

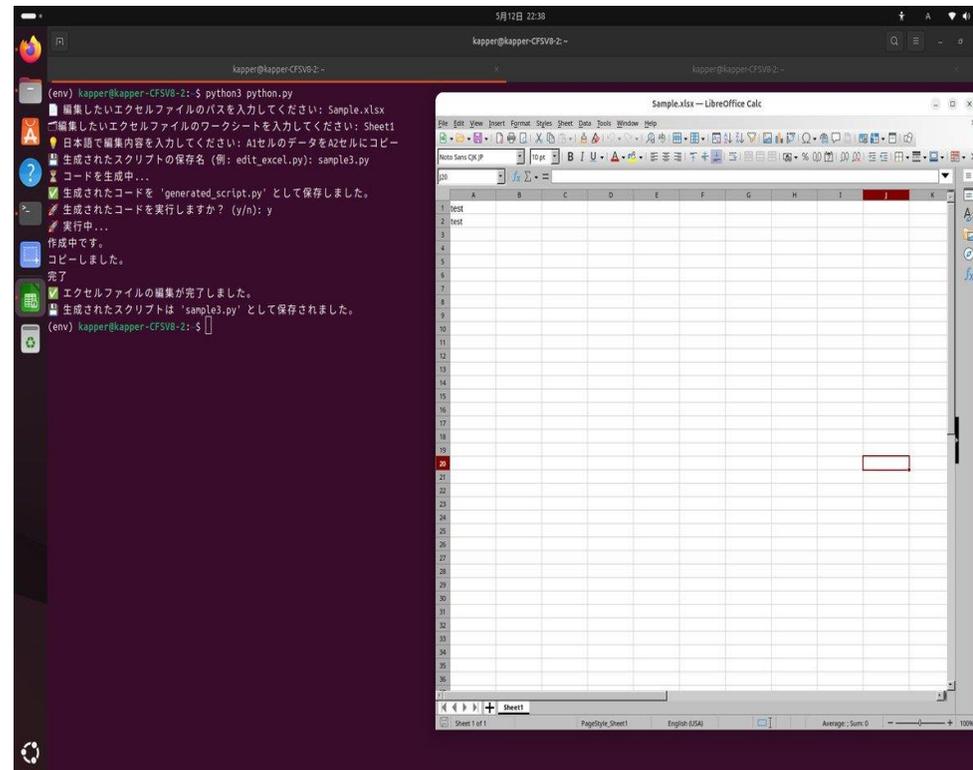
🌟 生成されたコマンド:
lsb_release -a

このコマンドを実行しますか? (y/n): y
🚀 コマンドを実行中...
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 25.04
Release:        25.04
Codename:       plucky
✅ 実行が完了しました。
kapper@kapper-CFSV8-2:~$
```

# SimonW/LLMの応用②Python+openpyxlでエクセル生成

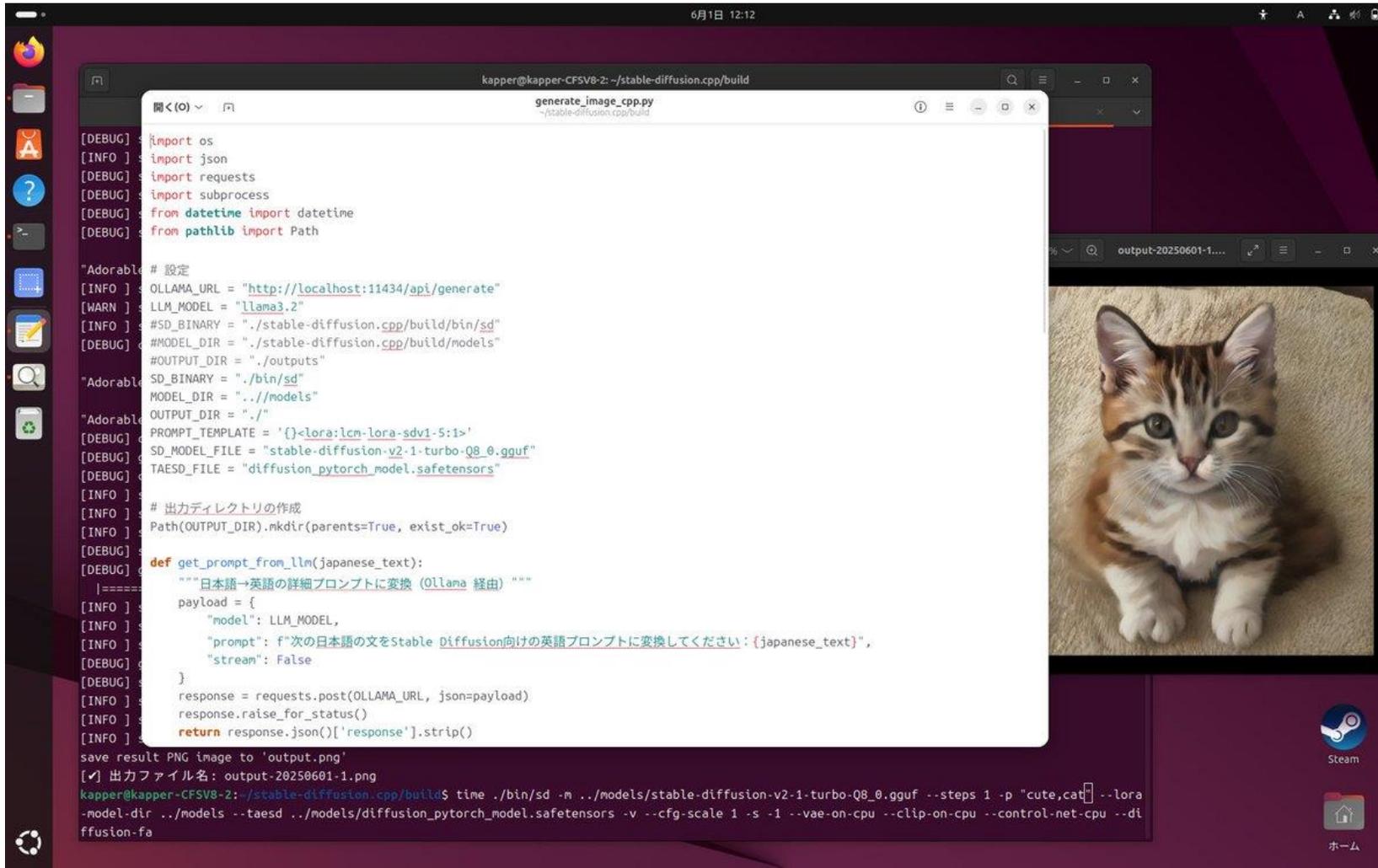
・ LLMアプリを使ってエクセルファイルを編集する。次のPythonプログラムのソースコードを作成

- ① LLMにローカルモデル-m llama3.2オプションを使用。
- ② 日本語の文章とエクセルファイルを入力してLLMに読み込ませる。
- ③ LLMにてPython3とopenpyxlを使用してソースコードを作成
- ④ 作成したソースコードを実行してエクセルファイルを編集して保存
- ⑤ 作成したソースコードにファイル名を付けて保存。



# SimonW/LLMの応用③Stable-Diffusion.cppのプロンプトを日本語から変換して生成

- PythonとSimonW/LLMでStable-Diffusion.cppのプロンプトを日本語から生成



```
kapper@kapper-CFSV8-2: ~/stable-diffusion.cpp/build
generate_image_cpp.py
~/stable-diffusion.cpp/build

[DEBUG] import os
[INFO] import json
[DEBUG] import requests
[DEBUG] import subprocess
[DEBUG] from datetime import datetime
[DEBUG] from pathlib import Path

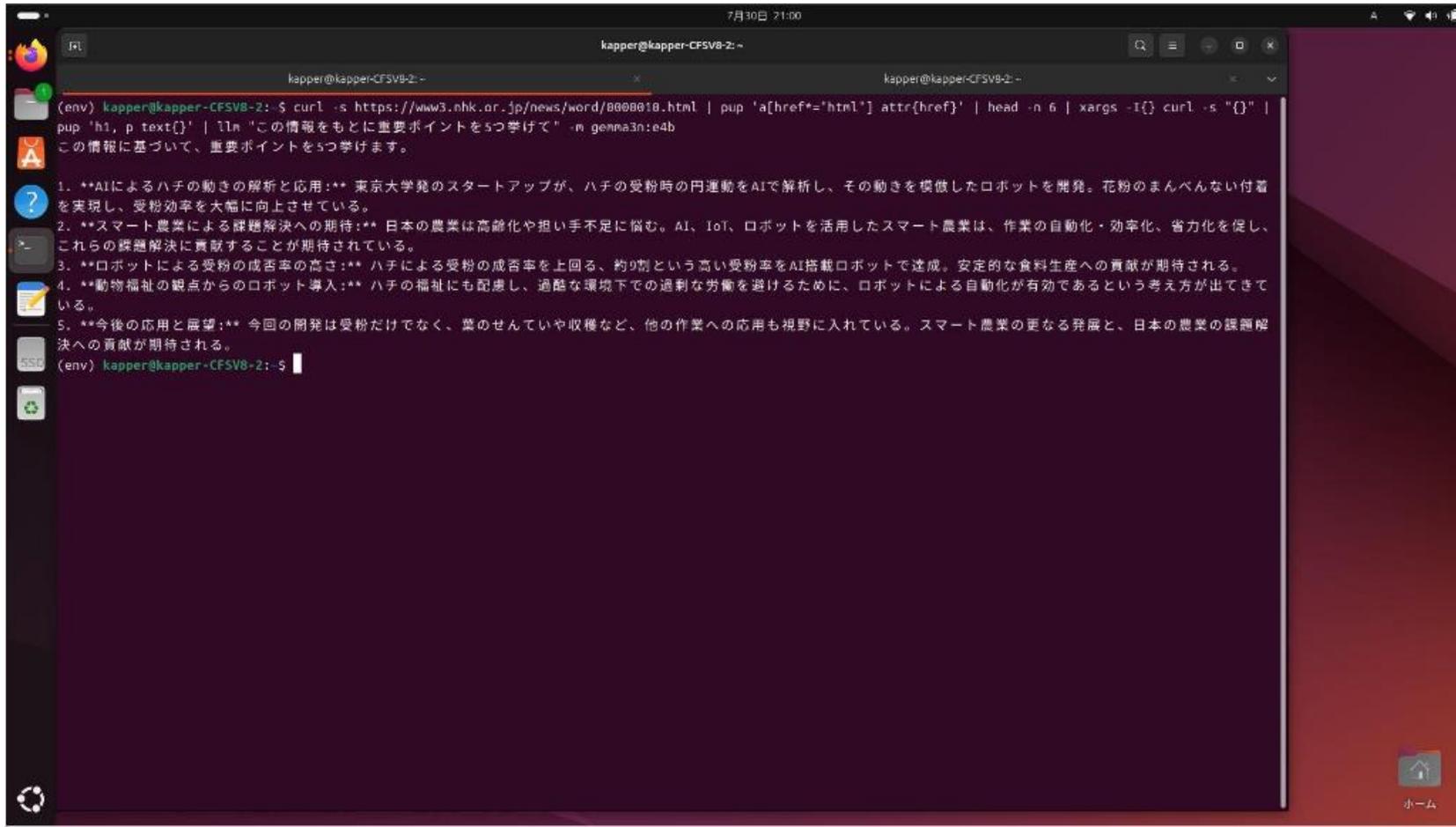
"Adorable" # 設定
[INFO] OLLAMA_URL = "http://localhost:11434/api/generate"
[WARN] LLM_MODEL = "llama3.2"
[INFO] #SD_BINARY = "./stable-diffusion.cpp/build/bin/sd"
[DEBUG] #MODEL_DIR = "./stable-diffusion.cpp/build/models"
[INFO] #OUTPUT_DIR = "./outputs"
[INFO] SD_BINARY = "./bin/sd"
[INFO] MODEL_DIR = "../models"
[INFO] OUTPUT_DIR = "."
[DEBUG] PROMPT_TEMPLATE = '{}<loralora:lcmlora-sdv1-5:1>'
[DEBUG] SD_MODEL_FILE = "stable-diffusion-v2-1-turbo-Q8_0.gguf"
[DEBUG] TAESD_FILE = "diffusion_pytorch_model.safetensors"
[INFO] # 出力ディレクトリの作成
[INFO] Path(OUTPUT_DIR).mkdir(parents=True, exist_ok=True)
[DEBUG]
[DEBUG] def get_prompt_from_llm(japanese_text):
[DEBUG] """日本語→英語の詳細プロンプトに変換 (Ollama 経由) """
[INFO] payload = {
[INFO]     "model": LLM_MODEL,
[INFO]     "prompt": f"次の日本語の文をStable Diffusion向けの英語プロンプトに変換してください: {japanese_text}",
[INFO]     "stream": False
[DEBUG] }
[DEBUG] response = requests.post(OLLAMA_URL, json=payload)
[INFO] response.raise_for_status()
[INFO] return response.json()['response'].strip()

save result PNG image to 'output.png'
[✓] 出力ファイル名: output-20250601-1.png
kapper@kapper-CFSV8-2: ~/stable-diffusion.cpp/build$ time ./bin/sd -m ../models/stable-diffusion-v2-1-turbo-Q8_0.gguf --steps 1 -p "cute,cat" --lora
-model-dir ../models --taesd ../models/diffusion_pytorch_model.safetensors -v --cfg-scale 1 -s -1 --vae-on-cpu --clip-on-cpu --control-net-cpu --di
ffusion-fa
```



# SimonW/LLMの応用④1行簡易RAG Retrieval-Augmented Generation (検索拡張生成)

- ネットから必要なデータを検索してSimonW/LLMで解析  
`curl -s https://www3.nhk.or.jp/rss/news/cat0.xml | llm "このRSSのタイトルと概要を日本語で5件にまとめて" -m gemma3n:e4b`



```
(env) kapper@kapper-CFSV8-2:~$ curl -s https://www3.nhk.or.jp/news/word/0000010.html | pup 'a[href*="html"] attr(href)' | head -n 6 | xargs -I{} curl -s "{}" | pup 'h1, p text{}' | llm "この情報をもとに重要ポイントを5つ挙げて" -m gemma3n:e4b
この情報に基づいて、重要ポイントを5つ挙げます。

1. **AIによるハチの動きの解析と応用:** 東京大学発のスタートアップが、ハチの受粉時の円運動をAIで解析し、その動きを模倣したロボットを開発。花粉のまんべんない付着を実現し、受粉効率を大幅に向上させている。
2. **スマート農業による課題解決への期待:** 日本の農業は高齢化や担い手不足に悩む。AI、IoT、ロボットを活用したスマート農業は、作業の自動化・効率化、省力化を促し、これらの課題解決に貢献することが期待されている。
3. **ロボットによる受粉の成否率の高さ:** ハチによる受粉の成否率を上回る、約9割という高い受粉率をAI搭載ロボットで達成。安定的な食料生産への貢献が期待される。
4. **動物福祉の観点からのロボット導入:** ハチの福祉にも配慮し、過酷な環境下での過剰な労働を避けるために、ロボットによる自動化が有効であるという考え方が出てきている。
5. **今後の応用と展望:** 今回の開発は受粉だけでなく、葉のせんていや収穫など、他の作業への応用も視野に入れている。スマート農業の更なる発展と、日本の農業の課題解決への貢献が期待される。
(env) kapper@kapper-CFSV8-2:~$
```

# SimonW/LLMの応用⑤Arduinoを操作

- プログラムを自動作成。スクリプトで自動化も出来る

```
echo "LEDを13番ピンで1秒ごとに点滅させるArduinoコードを書いて" | llm -m llamafire -s "
あなたはArduino開発の専門家です。ユーザーの要求をもとに、コンパイル可能なC++の
Arduinoスケッチだけを出力してください。説明文やMarkdown、余分なテキストは一切不
要です。" | sed '/^```/d' > sketch/sketch.ino
```

```
arduino-cli compile --fqbn arduino:avr:uno sketch
```

```
arduino-cli upload -p /dev/ttyUSB0 --fqbn arduino:avr:uno sketch
```

- 音声入力と組み合わせても可能

# SimonW/LLMの応用⑥Open\_jtalk音声出力

```
▪ llm -m llama3.2:latest 'NetBSDの解説を完結に一文で長文で説明' > /tmp/llm_output.txt && open_jtalk -x /var/lib/mecab/dic/open-jtalk/naist-jdic/ -m /usr/share/hts-voice/mei/mei_normal.htsvoice -ow /tmp/output.wav < /tmp/llm_output.txt && aplay --quiet /tmp/output.wav
```

- さらにFFmpeg(音声をwav保存) + whisper.cpp (wavをテキスト変換) + llmでローカルLLMに文字出力 + Open\_jtalkで音声出力  
みたいな事をすれば疑似会話出来ます。処理が遅いけど。

# さらに応用。生成AI会話アプリ作成

- aplayで音声をWav保存  
whisper.cppでwavをテキストに変換  
SimonW/LLMとllama.cppでテキストを生成AIで変換  
Open\_jtalkで変換したテキストを読み上げ

```
whisper_model_load: A_Loas      = 89
whisper_model_load:             CPU total size = 487.81 MB
whisper_model_load: model size  = 487.81 MB
whisper_backend_init_gpu: no GPU found
whisper_init_state: kv self size = 18.87 MB
whisper_init_state: kv cross size = 16.42 MB
whisper_init_state: kv pad size  = 4.72 MB
whisper_init_state: compute buffer (conv) = 22.42 MB
whisper_init_state: compute buffer (encode) = 128.82 MB
whisper_init_state: compute buffer (cross) = 4.28 MB
whisper_init_state: compute buffer (decode) = 97.28 MB

system_info: n_threads = 4 / 12 | MACSPES : CORE0L = 0 | OPENVINO = 0 | CPU : SSE3 = 1 | SSE42 = 1 | AVX = 1 | AVX_512 = 1 | AVX2 = 1 | F16C = 1
| FMA = 1 | BMI2 = 1 | DPDP = 1 | RDPACK = 1

main: processing 'input.wav' (33363 samples, 1.7 sec), 4 threads, 1 processors, 5 beams + best of 5, lang = ja, task = transcribe, timestamps = 0
...

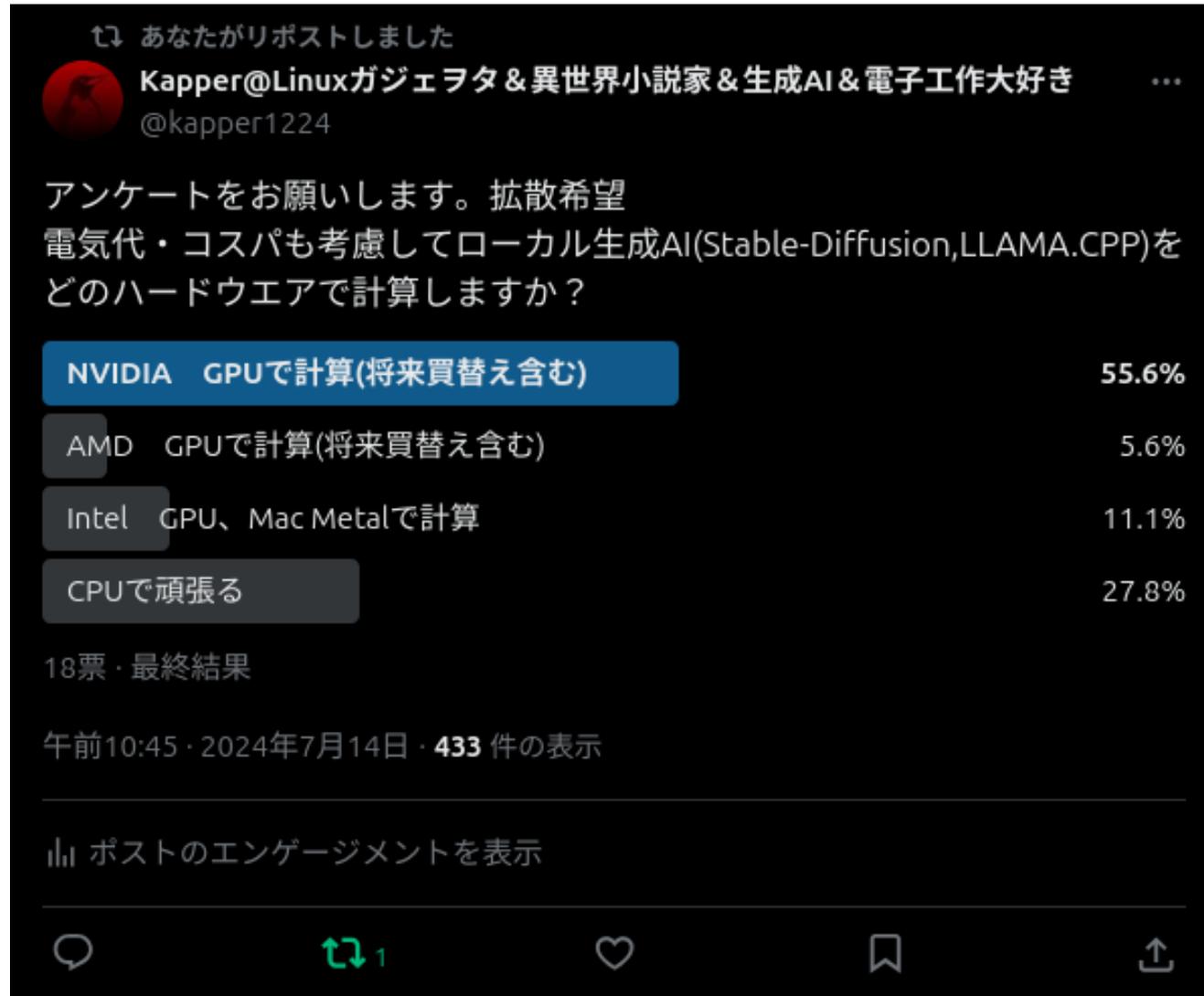
whisper_print_timings: load time = 645.59 ms
whisper_print_timings: fallbacks = 0 p / 8 h
whisper_print_timings: mel time = 5.90 ms
whisper_print_timings: sample time = 29.44 ms / 34 runs ( 0.86 ms per run)
whisper_print_timings: encode time = 5823.86 ms / 1 runs ( 5823.86 ms per run)
whisper_print_timings: decode time = 0.86 ms / 1 runs ( 0.86 ms per run)
whisper_print_timings: batchd time = 383.33 ms / 33 runs ( 11.62 ms per run)
whisper_print_timings: prompt time = 0.86 ms / 1 runs ( 0.86 ms per run)
whisper_print_timings: total time = 6268.56 ms
[Whisper 生成結果 read]

ネット8500の生成
あなた: ネット8500の生成
```

# ハードウェア目安

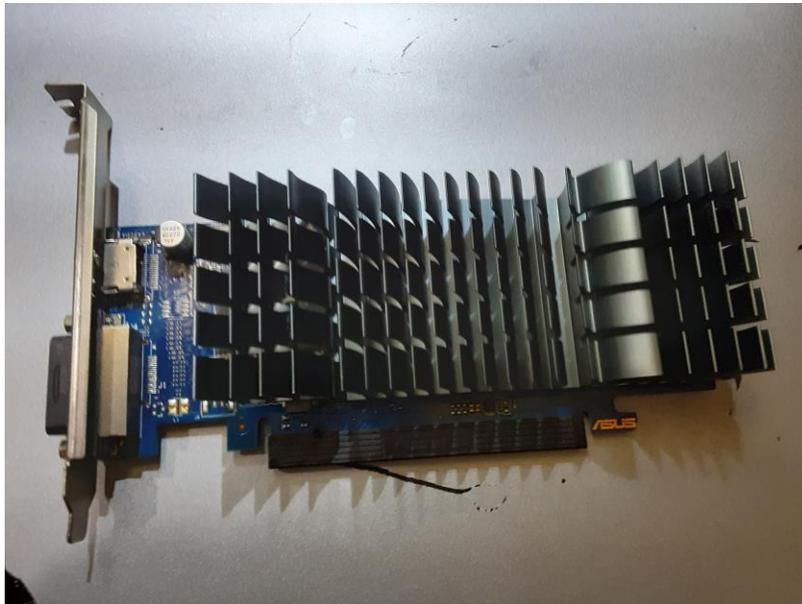
- 最低(動くだけ。速度は一切考慮しない)  
x86:SSE3搭載CPU、メモリ+Swap:1GB以上(モデル量子化の例外あり)  
Android メモリ+Swap 1GB以上(モデル量子化の例外あり)  
RaspberryPi4,5 メモリ+Swap 1GB以上
- 必要  
x86:第8世代Intel、AMD Ryzen以降、メモリ+Swap 3GB以上  
Android メモリ+Swap 3GB以上  
GPU:VRAMメモリ2GB以上 (Intel内蔵GPU iris Xe、UHD以上)
- 推奨(GPU前提)  
x86:第8世代Intel、AMD Ryzen以降、メモリ8GB以上  
GPU:NVIDIA GTX1060、AMD RX580以降 VRAMメモリ6GB以上(gpt-oss 20Bなどで12GB以上が望ましい)
- できれば10 tokens / sec 以上の速度を狙う(推奨15~25 tokens / sec)
- 速度を上げるには小さいモデル+高速GPU+オプション(-B 512 -C 2024 -t (コア数))など使用
- 低スペックPC、低VRAMならGemma3 1B Q4\_1(764MB)かQwen3 1.7B Q4\_K\_M(1.28GB)のモデルを推奨
- 賢い擬人化させる場合には出来るだけGPU VRAMを増やしてデータサイズの大きいモデルを使用
- 邪道であるが、CPUメインメモリを増やして内蔵GPUかNPUで処理する方法もあり

# アンケート結果(ご協力ありがとうございました)

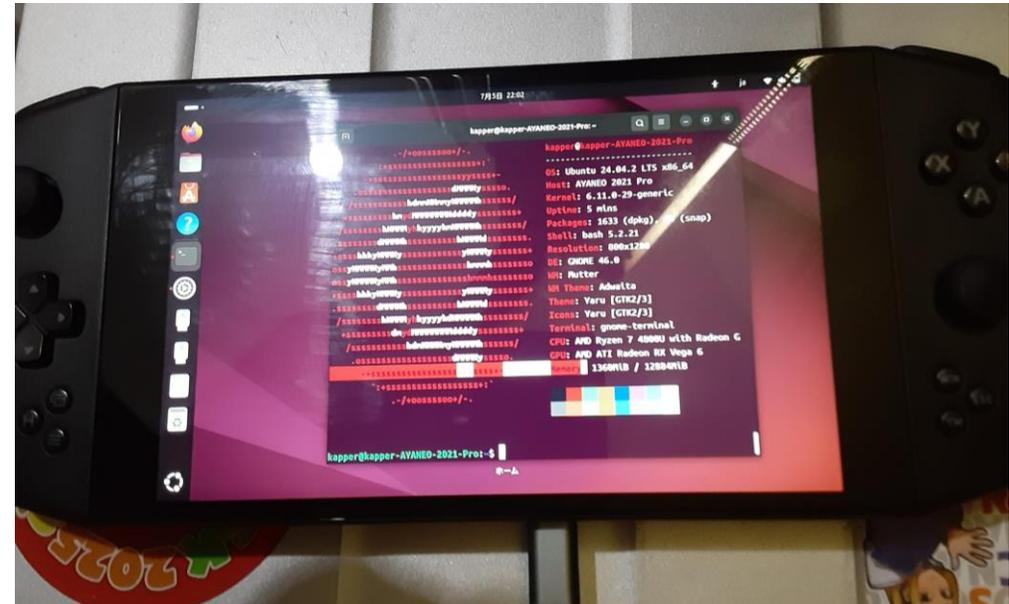


# 実は爆安ジャンクGPUボードLLMに使えます

- 日本語対応のLLAMA3.2 3Bの容量が1.5GBなのでVRAM2GBで動きます。
- AMDのAPU内蔵GPUでもOK。ゲーミングタブレットPCなど。
- 情報収集には物足りなくてもSimonW/LLMなら十分な性能



GTX1030 VRAM 2GB



Ayaneo 2021 Pro VRAM 3GB

# 実は爆安ジャンクGPUボードLLMに使えます



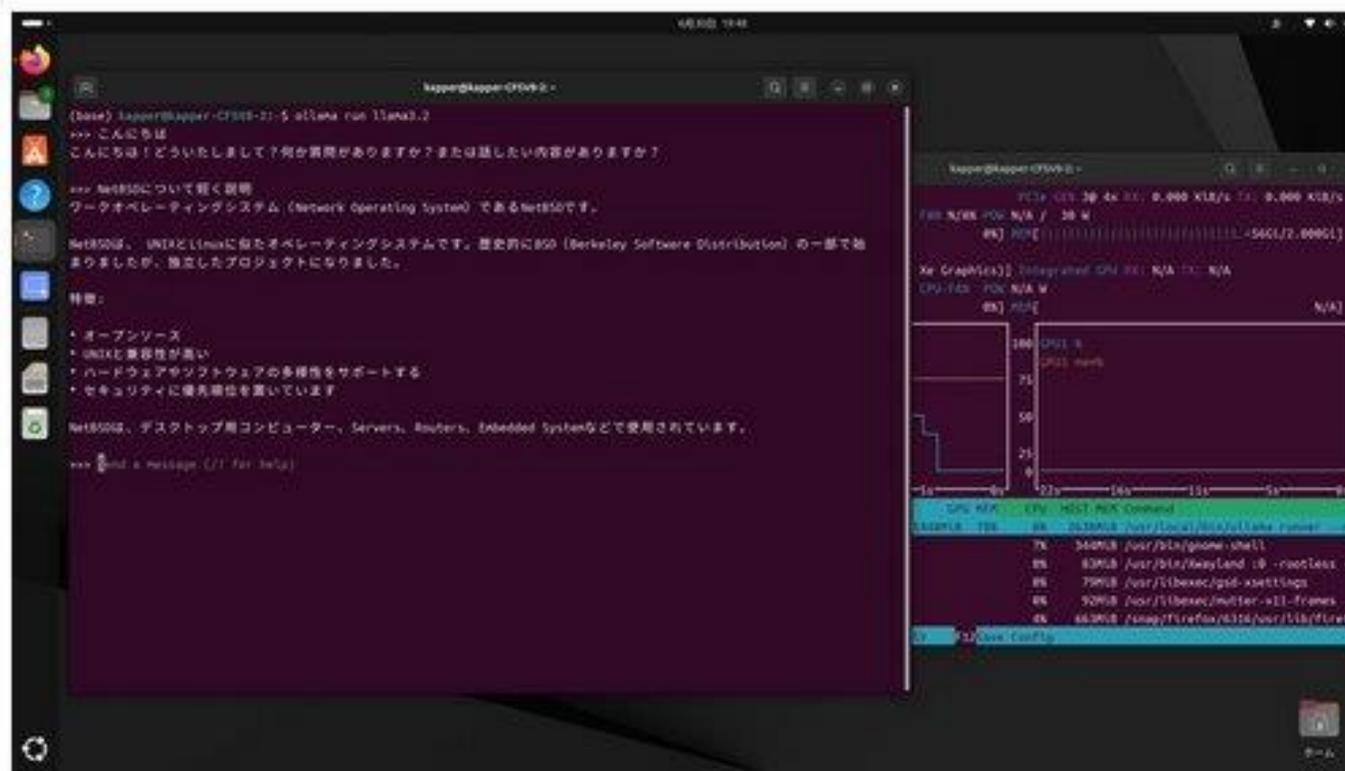
Kapper@Linuxガジェット家&異世界小説家&生成AI&電子工作大好き@kapper1224

GTX1030でOllamaとllama3.2

VRAM 1.456GB

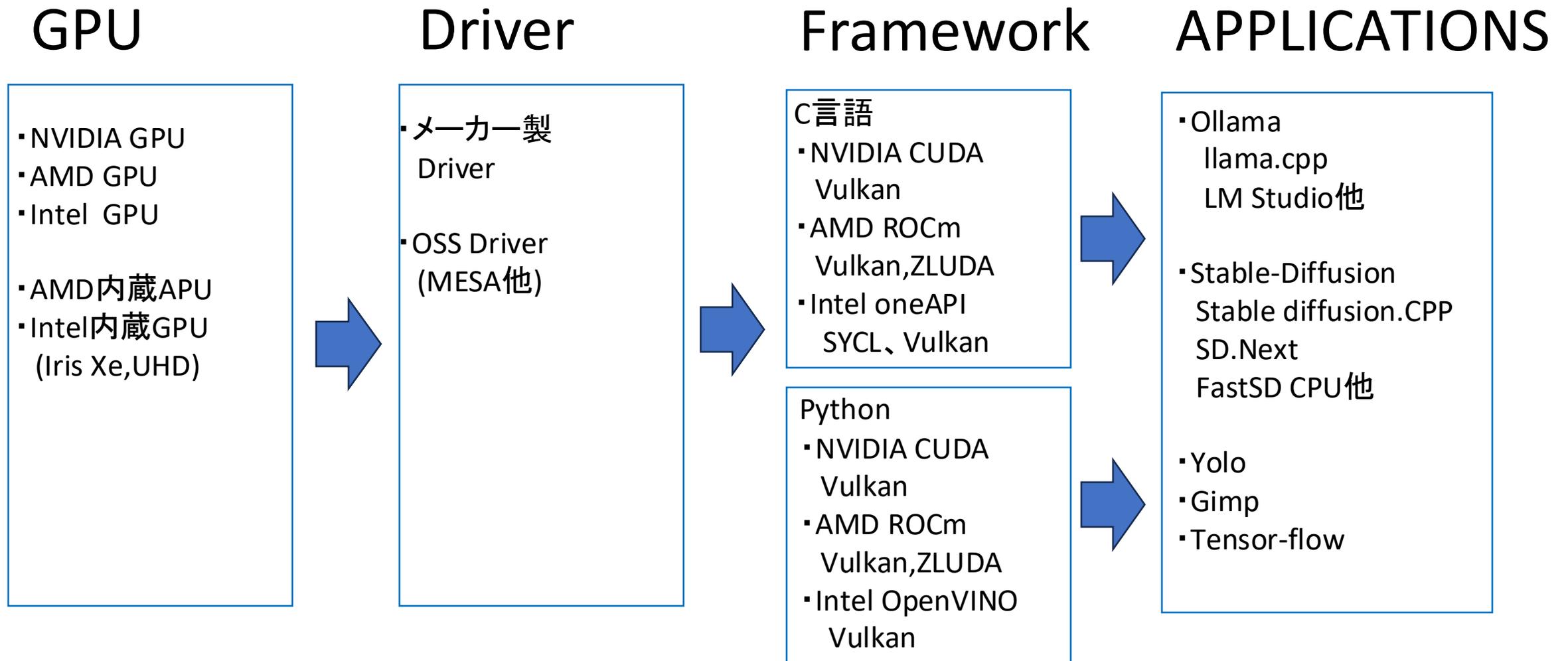
速度は遅いけど普通に動くいいね [pic.x.com/6LYiC96JXH](https://pic.x.com/6LYiC96JXH)

2025-06-30 19:49:57



# GPGPU

- GPUで高速化するにはそれぞれのツール、ハードのセットアップが必要



# GPUベンチマーク

- llama.cpp公式GithubにVulkanベンチマーク結果。  
おおよそ他のベンチマークと類似(CUDA,ROCm,SYCLはvulkanより高速)  
<https://github.com/ggml-org/llama.cpp/discussions/10879>

- 代表例(llama 2 7B)

NVIDIA RTX4090:170.95 tokens/sec

NVIDIA RTX3090:136.81 tokens/sec

AMD RX 7900XT:123.18 tokens/sec

Apple M3 Ultra:115.54 tokens/sec

AMD RX 9060XT:70.13 tokens/sec

NVIDIA RX3060:64.76 tokens/sec

Intel Arc A770:49.43 tokens/sec

AMD Ryzen AI Max+395:46.75 tokens/sec

NVIDIA GTX1070Ti:42.86 tokens/sec

Intel Core Ultra 7 258V:21.86 tokens/sec

AMD Ryzen AI 9 HX 370:21.23 tokens/sec

AMD Ryzen Z1 Extreme:18.77 tokens/sec

Intel Core i7 1100:7.28 tokens/sec

Intel Core i5 8000:3.23 tokens/sec

Vulkan Scoreboard for Llama 2 7B, Q4\_0 (no FA)

Chip	pp512 t/s	tg128 t/s	Commit	Comments
Nvidia RTX 4090	8534.56 ± 200.32	170.95 ± 0.32	<a href="#">bb4f7a9</a>	coopmat2
AMD Radeon RX 7900 XTX	3489.67 ± 82.17	145.00 ± 0.89	<a href="#">d1aa0cc</a>	
Nvidia RTX 3090	4543.96 ± 73.80	136.81 ± 3.63	<a href="#">bb4f7a9</a>	coopmat2
Nvidia RTX 5070 Ti	6213.63 ± 27.72	135.63 ± 0.18	<a href="#">d13d0f6</a>	coopmat2
AMD Radeon RX 9070 XT	3831.64 ± 1.82	130.57 ± 0.03	<a href="#">fd1234c</a>	
AMD Radeon RX 7900 XT	2941.58 ± 17.17	123.18 ± 0.40	<a href="#">71e74a3</a>	
Nvidia A100 (80GB)	3103.32 ± 4.21	121.83 ± 0.54	<a href="#">d394a9a</a>	
Apple M3 Ultra Mac Studio	1116.83 ± 0.55	115.54 ± 0.78	<a href="#">2d451c8</a>	MoltenVK
AMD Radeon RX 6900 XT	1257.98 ± 1.55	101.42 ± 0.02	44e18ef	
AMD Radeon RX 7800 XT	2145.60 + 23.14	96.89 + 0.22	<a href="#">baad948</a>	
AMD Radeon RX 6800 XT	1533.60 ± 2.47	95.56 ± 0.72	N/A	
Nvidia RTX 4070	3179.37 ± 46.16	92.29 ± 0.28	<a href="#">9a48399</a>	
AMD Radeon PRO W6800X	510.80 ± 0.13	86.47 ± 0.46	<a href="#">13b4548</a>	MoltenVK

# VRAM量を考慮するとこのくらいのグラボがお買い得(VRAM $\leq$ 8GBはゲームでは使いにくいいためお安い)eGPUもあり



ASUS RX480-8G  
RadeonRX480/8GB(GDDR5)/PCI-E

1個の在庫  
中古¥5,980~



AMD Radeon RX Vega 56  
8GB(HBM2)/PCI-E

20個の在庫  
中古¥6,980~



NVIDIA GeForce GTX980Ti  
6GB(GDDR5)/PCI-E

1個の在庫  
中古¥6,980~



ASUS GTX1070-8G  
GTX1070/8GB(GDDR5)/PCI-E

1個の在庫  
中古¥9,980~



MSI Radeon RX Vega 64 Air  
Boost 8G OC RX VEGA  
64/8GB(HBM2)/PCI-E

2個の在庫  
中古¥10,990~



ZOTAC GeForce GTX 1080 AMP  
Edition (ZT-P10800C-10P)  
GTX1080/8GB(GDDR5X)/PCI-E

3個の在庫  
中古¥12,980~



ZOTAC GeForce GTX 1080 Ti  
Blower(ZT-P10810B-10P)  
GTX1080Ti/11GB(GDDR5X)/PCI-E

2個の在庫  
中古¥16,980~



AKiTiO AKiTiO Node (500W)  
Thunderbolt3対応 外付けGPU  
ボックス

1個の在庫  
中古¥17,980~



ASRock Phantom Gaming X  
Radeon VII 16G Radeon  
VII/16GB(HBM2)/PCI-E

1個の在庫  
中古¥28,980~



Acer Predator BiFrost Intel Arc  
A770 OC APBF-IA770-16G-OC  
ArcA770/16GB(GDDR6)

2個の在庫  
中古¥31,980~

# 中古デスクトップならこの価格。将来性より今安く生成AI使える物。 生成AIではCPUはあまり重要ではない。グラボ代+OS代+おまけ扱い

福岡筑紫通り店 ☎092-436-4781 [店舗に問い合わせる](#)

状態：外装にスレあり  
付属品：本体のみ

ディスプレイ (画面) サイズ	-
付属ディスプレイ仕様	-
搭載OS	Windows 10 Pro 64bit
CPU	Core i5 6600K
CPUクロック	3.5GHz
標準メモリ容量	8GB
内蔵ストレージ容量	500G(SSD)
ドライブ	-
グラフィックコントローラ名称	nVidia Geforce GTX 1070, 8GB GDDR 5
TVチューナ	-
有線LAN	10/100/1000BASE-T
無線LAN/その他	-

✕ ポスト



パソコン本体

商品番号：85112460

**¥21,980**

[カートに入れる](#)

✕ ポスト



パソコン本体

福岡筑紫通り店 ☎092-436-4781 [店舗に問い合わせる](#)

【夏の感謝祭】  
※2T (HDD) にスマートエラーあり  
状態：外装キズ  
付属品：本体のみ

ディスプレイ (画面) サイズ	-
搭載OS	Windows 10 Home
CPU	Core i7 6700
CPUクロック	3.4GHz
標準メモリ容量	16GB
内蔵ストレージ容量	2TB(HDD)+250GB(SSD)
ドライブ	CD-R/-RW・DVD±R DL/±RW・BD-R/-RE
グラフィックコントローラ名称	nVIDIA GeForce GTX 1080
TVチューナ	-
有線LAN	10/100/1000BASE-T
無線LAN/その他	-

商品番号：85110886

**¥21,990**

[店頭限定販売](#)

✕ ポスト



パソコン本体

千葉店 ☎043-204-2142 [店舗に問い合わせる](#)

nVIDIA GeForce GTX 1060 6GB  
状態：使用感スレ  
付属品：電源ケーブル  
※箱欠品の為店頭限定の販売になります  
※お渡しの際は簡易包装でのお渡しになります  
※お持ち帰りする際運搬手段をご留意お願いいたします

搭載OS	なし
CPU	Core i5 6500
CPUクロック	3.2GHz
標準メモリ容量	16GB
内蔵ストレージ容量	256GB(SSD)
ドライブ	DVDマルチドライブ
グラフィックコントローラ名称	nVIDIA GeForce GTX 1060
TVチューナ	-
有線LAN	10/100/1000
無線LAN/その他	-

商品番号：39245012

**¥24,980**

[店頭限定販売](#)

大阪本店 ☎06-6645-0416 [店舗に問い合わせる](#)

【夏の感謝祭】  
☆★店頭限定商品★★  
箱がなく発送困難なため発送不可、店頭限定となります。

状態：キズ・汚れがあります  
付属品：電源ケーブルのみ

搭載OS	Windows 10 Home
CPU	Core i7 7700
CPUクロック	3.6GHz
標準メモリ容量	64GB
内蔵ストレージ容量	500G(SSD)
ドライブ	DVDマルチドライブ
グラフィックコントローラ名称	nVIDIA GeForce GTX 1070 TI
有線LAN	1Gbps
無線LAN/その他	-

✕ ポスト



パソコン本体

商品番号：127300646

**¥30,990**

[店頭限定販売](#)

✕ ポスト



パソコン本体

新宿店 ☎03-5321-6553 [店舗に問い合わせる](#)

CPU：Ryzen5 3500X/3.6GHz z  
状態：外装キズスレ  
付属品：本体のみ (電源ケーブルも付属いたしません)  
※箱無しのため配送不可

ディスプレイ (画面) サイズ	-
搭載OS	Windows 11 Home
Officeソフト	なし
CPU	AMD Ryzen 5 3500
CPUクロック	3.6GHz
標準メモリ容量	32G
内蔵ストレージ容量	512G(SSD)
ドライブ	-
グラフィックコントローラ名称	nVIDIA GeForce GTX 1660 SUPER
TVチューナ	-
有線LAN	○
無線LAN/その他	-

商品番号：49304449

**¥37,980**

[店頭限定販売](#)

✕ ポスト



パソコン本体

札幌店 ☎011-222-0655 [店舗に問い合わせる](#)

状態：スレキズ、シール痕、内部ホコリ汚れ (軽度) あり  
付属品：本体のみ

【CPU】 Core i7 8700  
【CPUクロック】 3.2GHz  
【搭載OS】 Windows 11 Pro 64bit  
【標準メモリ容量】 16GB  
【内蔵ストレージ容量】 500G(SSD)  
【ドライブ】 DVDマルチドライブ  
【グラフィックコントローラ名称】 nVIDIA GeForce RTX 2070 SUPER  
【TVチューナ】 -  
【有線LAN】 ○  
【無線LAN/その他】 -

搭載OS	Windows 11 Pro 64bit
CPU	Core i7 8700
CPUクロック	3.2GHz
標準メモリ容量	16GB
内蔵ストレージ容量	500G(SSD)
ドライブ	DVDマルチドライブ
グラフィックコントローラ名称	nVIDIA GeForce RTX 2070 SUPER
TVチューナ	-
有線LAN	○

商品番号：83284619

**¥50,980**

[店頭限定販売](#)

# NVIDIA CUDAインストール

- Ubuntu22.04の場合  
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86\_64/cuda-ubuntu2204.pin
- sudo mv cuda-ubuntu2204.pin /etc/apt/preferences.d/cuda-repository-pin-600
- wget https://developer.download.nvidia.com/compute/cuda/12.0.0/local\_installers/cuda-repo-ubuntu2204-12-0-local\_12.0.0-525.60.13-1\_amd64.deb
- sudo dpkg -i cuda-repo-ubuntu2204-12-0-local\_12.0.0-525.60.13-1\_amd64.deb
- sudo cp /var/cuda-repo-ubuntu2204-12-0-local/cuda-\*-keyring.gpg /usr/share/keyrings/
- sudo apt-get update
- sudo apt-get -y install cuda

```
sudo nano .bashrc
```

テキストエディタで下記を保存

```
export PATH="/usr/local/cuda/bin:$PATH"
```

- export LD\_LIBRARY\_PATH="/usr/local/cuda/lib64:\$LD\_LIBRARY\_PATH"
- <https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html>

# GTX1060 6GBでもまずまずの速度。十分実用

```
kapper@kapper-CFSV8-2:~/llama.cpp$ ./build/bin/llama-bench -m ../Llama-3.2-3B-Instruct-Q4_K_L.gguf -ngl 999 -b 512
```

```
ggml_cuda_init: GGML_CUDA_FORCE_MMQ: no
```

```
ggml_cuda_init: GGML_CUDA_FORCE_CUBLAS: no
```

```
ggml_cuda_init: found 1 CUDA devices:
```

```
Device 0: NVIDIA GeForce GTX 1060 6GB, compute capability 6.1, VMM: yes
```

model	size	params	backend	ngl	n_batch	test	t/s
llama 3B Q4_K - Medium	1.96 GiB	3.21 B	CUDA	999	512	pp512	823.37 ± 1.94
llama 3B Q4_K - Medium	1.96 GiB	3.21 B	CUDA	999	512	tg128	42.12 ± 0.11

# GTX1030 2GBでも頑張ればここまでは

```
kapper@kapper-CFSV8-2:~/llama.cpp$ ./build/bin/llama-bench -m ../Llama-3.2-3B-Instruct-Q4_K_L.gguf -ngl 20 -b 64
```

```
ggml_cuda_init: GGML_CUDA_FORCE_MMQ: no
```

```
ggml_cuda_init: GGML_CUDA_FORCE_CUBLAS: no
```

```
ggml_cuda_init: found 1 CUDA devices:
```

```
Device 0: NVIDIA GeForce GT 1030, compute capability 6.1, VMM: yes
```

model	size	params	backend	ngl	n_batch	test	t/s
llama 3B Q4_K - Medium	1.96 GiB	3.21 B	CUDA	20	64	pp512	123.42 ± 0.84
llama 3B Q4_K - Medium	1.96 GiB	3.21 B	CUDA	20	64	tg128	10.26 ± 0.48

# Vulkan AMD GPUインストール(\*BSD推奨)

- AMDのサイトからamdgpu-install~.debをダウンロード  
`sudo apt-get install ./amdgpu-install-VERSION.deb`
- `sudo apt-get update`  
`sudo amdgpu-install`
- Vulkan-SDKのインストール(SDKに必要ファイルあり)  
`wget -q0- https://packages.lunarg.com/lunarg-signing-key-pub.asc`  
`| sudo tee /etc/apt/trusted.gpg.d/lunarg.asc`
- `sudo wget -q0 /etc/apt/sources.list.d/lunarg-vulkan-noble.list`  
`http://packages.lunarg.com/vulkan/lunarg-vulkan-noble.list`
- `sudo apt update`
- `sudo apt install vulkan-sdk`
- GPU Driverはメーカー公式でもOSS版でもどちらでも使えます(MESA、OpenCL)  
申し訳ありませんが、\*BSDへのVulkanインストールは勉強中です

# SteamDeck(AMD Zen2)でVulkan GPU

- AMD APU内蔵GPUの割に侮れない性能。GTX1060の半分程度？  
小さいLLMモデルなら快適に動く(問題はVRAM)

```
kapper@kapper-Jupiter:~/llama.cpp$ ./build/bin/llama-bench -m models/gemma-3-4b-it.Q4_K_M.gguf
ggml_vulkan: Found 1 Vulkan devices:
ggml_vulkan: 0 = AMD Custom GPU 0405 (RADV VANGOGH) (radv) | uma: 1 | fp16: 1 | warp size: 32 | shared memory: 65536 | i
nt dot: 1 | matrix cores: none
```

model	size	params	backend	ngl	test	t/s
gemma3 4B Q4_K - Medium	2.31 GiB	3.88 B	Vulkan	99	pp512	119.98 ± 0.30
gemma3 4B Q4_K - Medium	2.31 GiB	3.88 B	Vulkan	99	tg128	19.62 ± 0.10

# ipex-llm(oneAPI+SYCL)Intel GPUインストール

- Ubuntu24.04の場合。Intel GPUドライバインストール  
下記の内容をコピー

<https://dgpu-docs.intel.com/driver/client/overview.html>

- oneAPIのインストール。コピー

<https://www.intel.com/content/www/us/en/docs/oneapi/installation-guide-linux-apt>

- Ipex-LLMバイナリをダウンロードして展開

llama-cpp-ipex-llm-2.3.0b20250612-ubuntu-core.tgz を解凍

<https://github.com/ipex-llm/ipex-llm/releases/tag/v2.3.0-nightly>

- ExportしてSYCLを許可

```
export GGML_SYCL_DISABLE_OPT=0
export GGML_SYCL_DISABLE_GRAPH=0
export ZES_ENABLE_SYSMAN=1
```

- 対応モデルをダウンロードして実行

```
echo "NetBSDの使い方事例" |./llama-cli -m ../llama-3.2-3B-Instruct-Q4_K_L.gguf -ngl 999 -b 512 -t 12
```

## Installing Client GPUs on Ubuntu Desktop

Ubuntu Desktop 25.04 and 24.04 Ubuntu Desktop 22.04

For Ubuntu 25.04 and 24.04, we offer the intel-graphics Personal Package Archive (PPA). This PPA provides early access to the latest packages, along with additional tools and features, such as EU debugging. Follow these steps to install the intel-graphics PPA and the required compute and media packages.

### Note

Using Ubuntu 24.04 with Lunar Lake or Battlemage requires your system to be running the hardware enablement (HWE) kernel. By default, Ubuntu Desktop 24.04 tracks the HWE stack. However, if your system is instead using the general availability (GA) kernel, you must switch to the HWE kernel before proceeding with the Client GPU installation.

1. Refresh the local package index and install the package for managing software repositories.

```
sudo apt-get update
sudo apt-get install -y software-properties-common
```

2. Add the intel-graphics PPA.

```
sudo add-apt-repository -y ppa:kobuk-team/intel-graphics
```

3. Install the compute-related packages.

```
sudo apt-get install -y libze-intel-gpu1 libze-intel-metrics-discovery intel-opencl-icd cLinfo intel-gsc
```

4. Install the media-related packages.

```
sudo apt-get install -y intel-media-va-driver-non-free libbfx-gen1 libvpl12 libvpl-tools libva-glx2 va-driver-a
```

### Assets 20

<a href="#">llama-cpp-ipex-llm-2.3.0b20250424-win-npu.zip</a>	8.16 MB	Apr 27
<a href="#">llama-cpp-ipex-llm-2.3.0b20250424-win.zip</a>	92.2 MB	Apr 27
<a href="#">llama-cpp-ipex-llm-2.3.0b20250430-ubuntu-core.tgz</a>	124 MB	Apr 30
<a href="#">llama-cpp-ipex-llm-2.3.0b20250430-ubuntu-xeon.tgz</a>	126 MB	Apr 30
<a href="#">llama-cpp-ipex-llm-2.3.0b20250605-ubuntu-xeon.zip</a>	125 MB	Jun 6
<a href="#">llama-cpp-ipex-llm-2.3.0b20250605-win.zip</a>	92.3 MB	Jun 6
<a href="#">llama-cpp-ipex-llm-2.3.0b20250611-ubuntu-xeon.tgz</a>	177 MB	Jun 12
<a href="#">llama-cpp-ipex-llm-2.3.0b20250612-ubuntu-core.tgz</a>	126 MB	Jun 13
<a href="#">llama-cpp-ipex-llm-2.3.0b20250612-ubuntu-xeon.tgz</a>	176 MB	Jun 13
<a href="#">llama-cpp-ipex-llm-2.3.0b20250612-win.zip</a>	92.3 MB	Jun 13
<a href="#">llama-cpp-ipex-llm-2.3.0b20250722-win.zip</a>	92.4 MB	4 days ago
<a href="#">llama-cpp-ipex-llm-2.3.0b20250724-ubuntu.tgz</a>	124 MB	2 days ago
<a href="#">ollama-ipex-llm-2.3.0b20250612-ubuntu.tgz</a>	137 MB	Jun 13
<a href="#">ollama-ipex-llm-2.3.0b20250612-win.zip</a>	105 MB	Jun 13
<a href="#">ollama-ipex-llm-2.3.0b20250630-ubuntu.tgz</a>	140 MB	last month
<a href="#">ollama-ipex-llm-2.3.0b20250630-win.zip</a>	106 MB	last month
<a href="#">ollama-ipex-llm-2.3.0b20250725-ubuntu.tgz</a>	140 MB	2 days ago
<a href="#">ollama-ipex-llm-2.3.0b20250725-win.zip</a>	108 MB	2 days ago
<a href="#">Source code (zip)</a>		Apr 18
<a href="#">Source code (tar.gz)</a>		Apr 18

# Intel oneAPI対応ハード

- 第11世代以降CoreシリーズCPU、N100系 UHD、Iris Xe GPU
- Intel Arc GPU
- Intel XeonサーバCPU
- Intel Datacenter GPU
- NVIDIA,AMD の GPU (Codeplay の oneAPI プラグインを使用)
  
- 注) Core 第8～10世代のGPU(UHD)は未対応  
→とても遅いけどvulkanを使うしか無い
- LLMモデルサイズが同じでも計算速度に差が出るので注意

# ipex-llm ベンチマーク Intel 12世代 1245U Iris Xe

- Core第12世代 Iris XeでもoneAPI+SYCLでGPU動作しました
- CPU12コアの2倍程度の速度で動作。小さいモデルならそこそこ動く。メインメモリとVRAM共有なので大きなモデルも動きます

7月27日 12:05

kapper@kapper-CFSV8-2: ~/llama-cpp-ipex-llm-2.3.0b20250724-ubuntu

```
kapper@kapper-CFSV8-2:~/llama-cpp-ipex-llm-2.3.0b20250724-ubuntu$ ./llama-bench -m ../Llama-3.2-3B-Instruct-Q4_K_L.gguf -ngl 999 -b 512 -t 12
| model | size | params | backend | ngl | threads | n_batch | test | t/s |
| -----|-----|-----|-----|---|-----|-----|-----|-----|
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
| llama 3B Q4_K - Medium | 1.96 GiB | 3.21 B | SYCL | 999 | 12 | 512 | pp512 | 96.29 ± 0.55 |
| llama 3B Q4_K - Medium | 1.96 GiB | 3.21 B | SYCL | 999 | 12 | 512 | tg128 | 10.74 ± 0.37 |
```

build: d2c8ed1 (1)

```
kapper@kapper-CFSV8-2:~/llama-cpp-ipex-llm-2.3.0b20250724-ubuntu$ ./llama-bench -m ../Llama-3.2-3B-Instruct-Q4_K_L.gguf -ngl 0 -b 512 -t 12
| model | size | params | backend | ngl | threads | n_batch | test | t/s |
| -----|-----|-----|-----|---|-----|-----|-----|-----|
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
| llama 3B Q4_K - Medium | 1.96 GiB | 3.21 B | SYCL | 0 | 12 | 512 | pp512 | 23.10 ± 5.34 |
| llama 3B Q4_K - Medium | 1.96 GiB | 3.21 B | SYCL | 0 | 12 | 512 | tg128 | 6.37 ± 1.21 |
```

Intel Iris Xe GPU

CPU

# ipex-llm ベンチマーク Intel 12世代 1245U Iris Xe

- Phi-3,3.5,4のベンチマーク比較。容量同じでも速度違いあり

```
kapper@kapper-CFSV8-2:~/llama-cpp-ipex-llm-2.3.0b20250724-ubuntu$ ./llama-bench -m ../Phi-3-mini-4k-instruct-q4.gguf -ngl 99
| model | size | params | backend | ngl | test | t/s |
| -----|-----|-----|-----|---|-----|-----|
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
| phi3 3B Q4_K - Medium | 2.23 GiB | 3.82 B | SYCL | 99 | pp512 | 72.88 ± 3.85 |
| phi3 3B Q4_K - Medium | 2.23 GiB | 3.82 B | SYCL | 99 | tg128 | 8.43 ± 0.53 |

build: d2c8ed1 (1)
kapper@kapper-CFSV8-2:~/llama-cpp-ipex-llm-2.3.0b20250724-ubuntu$ ./llama-bench -m ../phi-3.5-mini-instruct-q4_k_m.gguf -ngl 99
| model | size | params | backend | ngl | test | t/s |
| -----|-----|-----|-----|---|-----|-----|
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
| phi3 3B Q4_K - Medium | 2.23 GiB | 3.82 B | SYCL | 99 | pp512 | 57.94 ± 4.90 |
| phi3 3B Q4_K - Medium | 2.23 GiB | 3.82 B | SYCL | 99 | tg128 | 6.52 ± 0.75 |

build: d2c8ed1 (1)
kapper@kapper-CFSV8-2:~/llama-cpp-ipex-llm-2.3.0b20250724-ubuntu$ ./llama-bench -m ../Phi-4-mini-instruct-abliterated-Q4_K_M.gguf -ngl 99
| model | size | params | backend | ngl | test | t/s |
| -----|-----|-----|-----|---|-----|-----|
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
get_memory_info: [warning] ext_intel_free_memory is not supported (export/set ZES_ENABLE_SYSMAN=1 to support), use total memory as free memory
| phi3 3B Q4_K - Medium | 2.31 GiB | 3.84 B | SYCL | 99 | pp512 | 55.23 ± 0.60 |
| phi3 3B Q4_K - Medium | 2.31 GiB | 3.84 B | SYCL | 99 | tg128 | 5.03 ± 0.12 |
```

# Vulkan ベンチマーク Intel Core第8世代 UHD

- Core第8世代 UHDでもvulkanでGPU動作しましたが遅い
- CPUプロセスを少ししか食わないのでブラウザを使いながらllama.cppを使う程度には良いですが...

```
kapper@kapper-CFSV8-2: ~/llama.cpp/build
kapper@kapper-CFSV8-2:~/llama.cpp/build$ ./build/bin/llama-bench -m ../../Llama-3.2-3B-Instruct-Q4_K_L.gguf -ngl 999 -t 4 -b 512
ggml_vulkan: Found 1 Vulkan devices:
ggml_vulkan: 0 = Intel(R) UHD Graphics 620 (WHL GT2) (Intel open-source Mesa driver) | uma: 1 | fp16: 1 | bf16: 0 | warp size: 32 | s
36 | int dot: 0 | matrix cores: none
| model | size | params | backend | ngl | n_batch | test | t/s |
|-----|-----|-----|-----|----|-----|-----|-----|
| llama 3B Q4_K - Medium | 1.96 GiB | 3.21 B | Vulkan | 999 | 512 | pp512 | 25.50 ± 0.02 |
| llama 3B Q4_K - Medium | 1.96 GiB | 3.21 B | Vulkan | 999 | 512 | tg128 | 3.21 ± 0.02 |
build: 1dc9614e (5999)
kapper@kapper-CFSV8-2:~/llama.cpp/build$ ./build/bin/llama-bench -m ../../Llama-3.2-3B-Instruct-Q4_K_L.gguf -ngl 0 -t 4 -b 512
ggml_vulkan: Found 1 Vulkan devices:
ggml_vulkan: 0 = Intel(R) UHD Graphics 620 (WHL GT2) (Intel open-source Mesa driver) | uma: 1 | fp16: 1 | bf16: 0 | warp size: 32 | s
36 | int dot: 0 | matrix cores: none
| model | size | params | backend | ngl | n_batch | test | t/s |
|-----|-----|-----|-----|----|-----|-----|-----|
| llama 3B Q4_K - Medium | 1.96 GiB | 3.21 B | Vulkan | 0 | 512 | pp512 | 12.75 ± 0.00 |
| llama 3B Q4_K - Medium | 1.96 GiB | 3.21 B | Vulkan | 0 | 512 | tg128 | 3.57 ± 0.01 |
```

3.21 ± 0.02

Intel UHD GPU

CPU

3.57 ± 0.01

# Vulkan ベンチマーク Intel Core第8世代 UHD

- Gemma3 1B Q4\_1なら8.10 tokens/sec  
日本語怪しいですが Gemma3 270M Q4で17.75 tokens/sec
- これだけモデルサイズが小さければRaspberryPiでも動きそうです。

```
kapper@kapper-CFSV8-2:~/llama.cpp$ ./build/bin/llama-bench -m ../gemma-3-1b-it-Q4_1.gguf -ngl 99 -b 512 -mmp 1
ggml_vulkan: Found 1 Vulkan devices:
ggml_vulkan: 0 = Intel(R) UHD Graphics 620 (WHL GT2) (Intel open-source Mesa driver) | uma: 1 | fp16: 1 | bf16: 0 | warp size: 32 | shared memory: 65536 |
int dot: 0 | matrix cores: none
| model | size | params | backend | ngl | n_batch | test | t/s |
| -----|-----|-----|-----|---|-----|-----|-----|
| gemma3 1B Q4_1 | 722.41 MiB | 999.89 M | Vulkan | 99 | 512 | pp512 | 60.87 ± 0.76 |
| gemma3 1B Q4_1 | 722.41 MiB | 999.89 M | Vulkan | 99 | 512 | tg128 | 8.10 ± 0.01 |
```

Intel UHD GPU

```
build: 228f724d (6129)
kapper@kapper-CFSV8-2:~/llama.cpp$ ./build/bin/llama-bench -m ../gemma-3-270m-it-Q4_K_M.gguf -ngl 99 -b 512 -mmp 1
ggml_vulkan: Found 1 Vulkan devices:
ggml_vulkan: 0 = Intel(R) UHD Graphics 620 (WHL GT2) (Intel open-source Mesa driver) | uma: 1 | fp16: 1 | bf16: 0 | warp size: 32 | shared memory: 65536 |
int dot: 0 | matrix cores: none
| model | size | params | backend | ngl | n_batch | test | t/s |
| -----|-----|-----|-----|---|-----|-----|-----|
| gemma3 ?B Q4_K - Medium | 235.16 MiB | 268.10 M | Vulkan | 99 | 512 | pp512 | 313.54 ± 0.30 |
| gemma3 ?B Q4_K - Medium | 235.16 MiB | 268.10 M | Vulkan | 99 | 512 | tg128 | 17.75 ± 0.08 |
```

Intel UHD GPU

```
build: 228f724d (6129)
```

# 最軽量日本語対応モデル Qwen3 1.7B Q4 K\_M

- 僅か1.28GBなのにギリギリ日本語が使える。とても軽量・高速CPUやIntel UHDなど限られたリソースに最適。ipex\_LLMで高速おおよそllama3.2 3Bの1.5倍高速モデル。オススメ

[https://huggingface.co/Alcoft/Qwen\\_Qwen3-1.7B-GGUF/tree/main](https://huggingface.co/Alcoft/Qwen_Qwen3-1.7B-GGUF/tree/main)

N100内蔵UHD+ipex-LLM(oneAPI+SYCL): 7.68 tokens/sec

```
llama_perf_sampler_print: sampling time = 8.97 ms / 49 runs ( 0.18 ms per token, 5459.61 tokens per second)
llama_perf_context_print: load time = 8542.45 ms
llama_perf_context_print: prompt eval time = 716.42 ms / 16 tokens ( 44.78 ms per token, 22.33 tokens per second)
llama_perf_context_print: eval time = 4168.22 ms / 32 runs ( 130.26 ms per token, 7.68 tokens per second)
llama_perf_context_print: total time = 4973.81 ms / 48 tokens
kapper@kapper-CFSV8-2:~/llama-cpp-ipex-llm-2.3.0b20250724-ubuntu$ echo "今日の天気。 no_think" | ./llama-cli -m ../Qwen_Qwen3-
```

Core 第8世代 内蔵UHD+Vulkan: 5.66 tokens/sec

```
kapper@kapper-CFSV8-2:~/llama.cpp$ ./build/build/bin/llama-bench -m ../Qwen_Qwen3-1.7B_Q4_K_M.gguf -ngl 999
ggml_vulkan: Found 1 Vulkan devices:
ggml_vulkan: 0 = Intel(R) UHD Graphics 620 (WHL GT2) (Intel open-source Mesa driver) | uma: 1 | fp16: 1 | bf16: 0 | warp s
int dot: 0 | matrix cores: none
| model | size | params | backend | ngl | test | t/s |
|-----|-----|-----|-----|----|-----|-----|
| qwen3 1.7B Q4_K - Medium | 1.19 GiB | 2.03 B | Vulkan | 999 | pp512 | 5.66 ± 0.17 |
| qwen3 1.7B Q4_K - Medium | 1.19 GiB | 2.03 B | Vulkan | 999 | tg128 | 5.66 ± 0.17 |
build: 1dc9614e (5999)
```

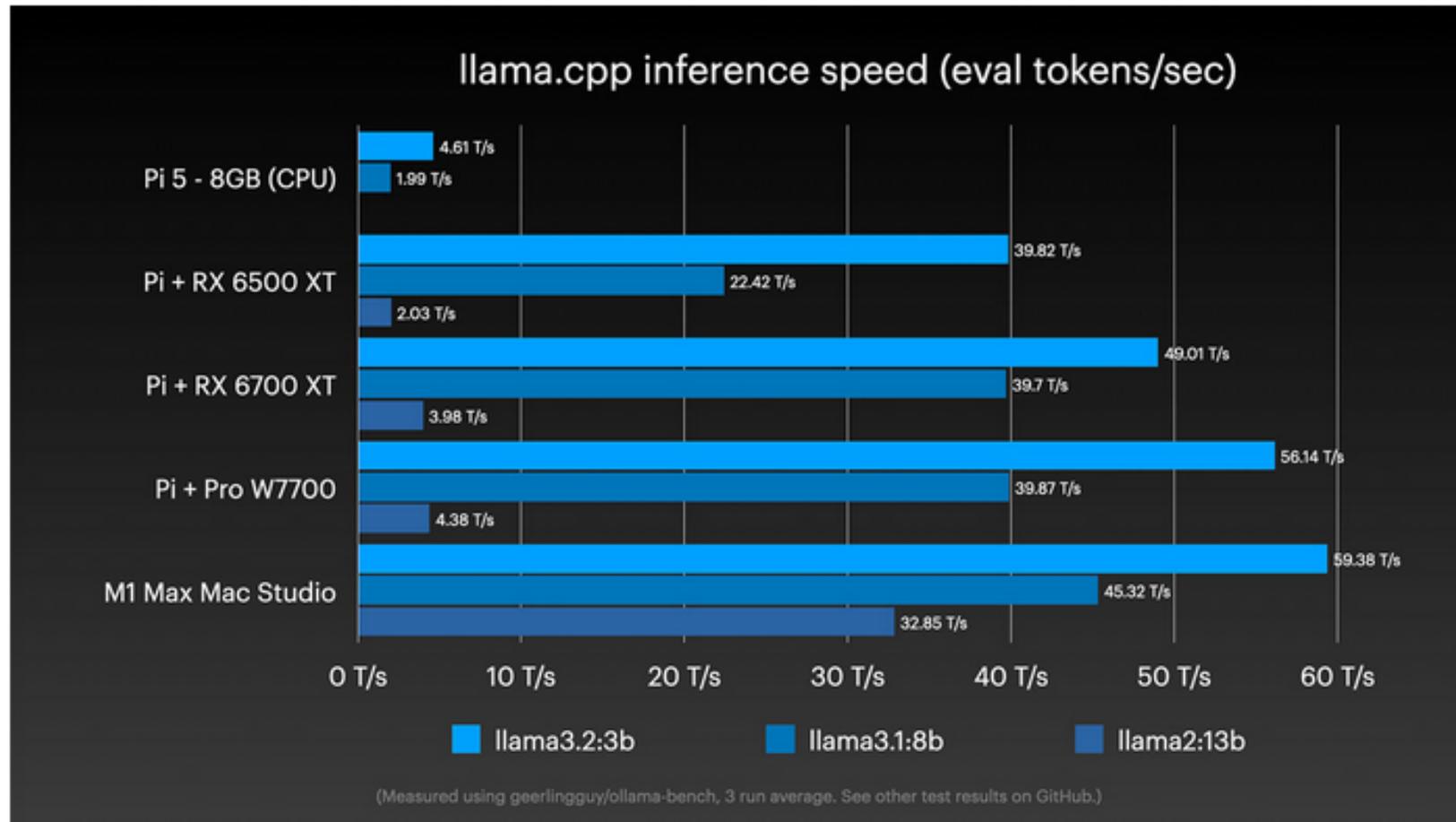
The screenshot shows the Hugging Face model card for 'Alcoft/Qwen\_Qwen3-1.7B-GGUF'. It lists various GGUF files for different quantization levels and data types. The file 'Qwen\_Qwen3-1.7B\_Q4\_K\_M.gguf' is highlighted with a red box, showing a size of 1.28 GB. Other files include 'Qwen\_Qwen3-1.7B\_Q4\_K\_S.gguf' (1.24 GB) and 'Qwen\_Qwen3-1.7B\_Q4\_K\_XL.gguf' (2.1 GB).

File Name	Size	Download Link
Qwen_Qwen3-1.7B_Q4_K_M.gguf	1.28 GB	Download
Qwen_Qwen3-1.7B_Q4_K_S.gguf	1.24 GB	Download
Qwen_Qwen3-1.7B_Q4_K_XL.gguf	2.1 GB	Download

# RaspberryPi5 のベンチマーク報告例

## Performance

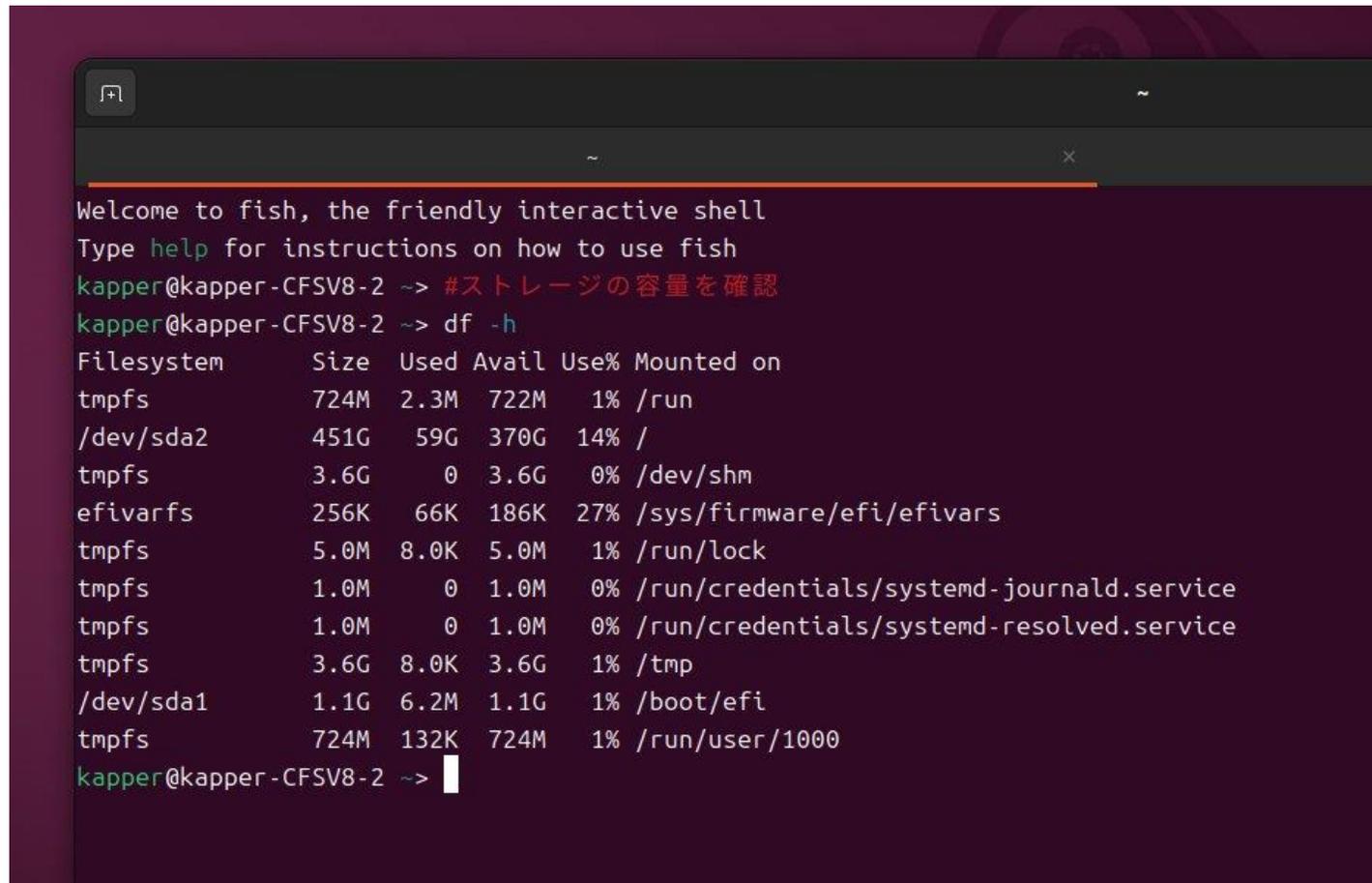
I went to Micro Center and bought a couple more consumer graphics cards for testing, and matched that up with the cards I already own, as well as my M1 Max Mac Studio, which has 64 GB of shared RAM and 24 GPU cores:



# AIサジェスチョンならFish shell AIも

- AIオートサジェスチョン機能ならFish-AIの方が操作系が良いです。

<https://github.com/Realiserad/fish-ai>



```
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
kapper@kapper-CFSV8-2 ~> #ストレージの容量を確認
kapper@kapper-CFSV8-2 ~> df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           724M  2.3M  722M   1% /run
/dev/sda2       451G   59G  370G  14% /
tmpfs           3.6G    0  3.6G   0% /dev/shm
efivarfs        256K   66K  186K  27% /sys/firmware/efi/efivars
tmpfs           5.0M   8.0K  5.0M   1% /run/lock
tmpfs           1.0M    0  1.0M   0% /run/credentials/systemd-journald.service
tmpfs           1.0M    0  1.0M   0% /run/credentials/systemd-resolved.service
tmpfs           3.6G   8.0K  3.6G   1% /tmp
/dev/sda1       1.1G   6.2M  1.1G   1% /boot/efi
tmpfs           724M  132K  724M   1% /run/user/1000
kapper@kapper-CFSV8-2 ~> |
```

# よくある質問・疑問点

- CPUではだめなんですか？ --- 遅いだけです。その分CPUメモリは安い。
- 古いPCあるんですが・・・---ジャンクGPUかノートパソコンならeGPUで高速化。
- AndroidスマホのGPUは動きますか？ --- VRAMが機種により差があるがVulkan Termuxとllama.cppで標準でvulkanが動くそうですが問題もあり。  
Subtle Vulkan shader compilation bug when running on Adreno GPUs (Samsung Galaxy S23 Ultra) #5186  
<https://github.com/ggml-org/llama.cpp/issues/5186>
- NVIDIA以外のGPUは動くのですか？ --- ROCm,DirectML,Metal,oneAPI+SYCLでも動く
- なぜIntel Arcは生成AIであまり使われないの？ --- 動作は問題なし。ipex-llm推奨
- Intel NPUでも動きますか？ --- OpenVINOかoneAPI+SYCL推奨(ipex-llm)
- ノートパソコンの内蔵GPUは使えますか？ ---遅いけどIntel内蔵GPUは動く。Intel Iris XeやUHD動作確認  
AMDのデスクトップやゲーミングPCは内蔵VRAMをBIOSで変更できるかも。
- Ollamaのモデルを追加したけどSimonW/LLMで動かない？ ---llm install llm-ollamaをもう一度実行
- Ollamaは何故\*BSDに対応していないの？ ---vulkanを強制指定してエラーが出るバグあり。どうも開発者が\*BSDのサポートに興味が無いらしく修正されていません。FreeBSDコミュニティが過去に大騒ぎしていました。  
<https://github.com/ollama/ollama/issues/1102>
- そもそもPythonフルスクラッチの方が良くない？ --- もちろんYES。SimonW/LLMは手軽さと汎用性重視

# まとめ

- 簡単なSimonW/LLMの使い方と説明
- SimonW/LLMのインストール方法を説明
- 簡単なコマンドラインを実例に各OSで動作
- 各種GPUをLinuxで動作確認。ノートパソコン内蔵GPUもOK
- 現時点最軽量日本語対応モデル  
Gemma3 1B Q4\_1かQwen3 1.7B Q4\_K\_M を低スペックPCでも推奨