Windowsで Linuxや並列計算の 練習をしよう (準備編)

大野

概要

- みなさんのPCは、複数のCPUコアが搭載されています
- スーパーコンピュータやワークステーションで並列計算をするための練習を自分のPCでやってみよう
- ラップトップPCで環境を作れば、家でもどこでも練習ができます
- WSLを利用する
- Windows11になって、だいぶ便利になったので、資料を部分的に作り直しました

概要(並列計算)

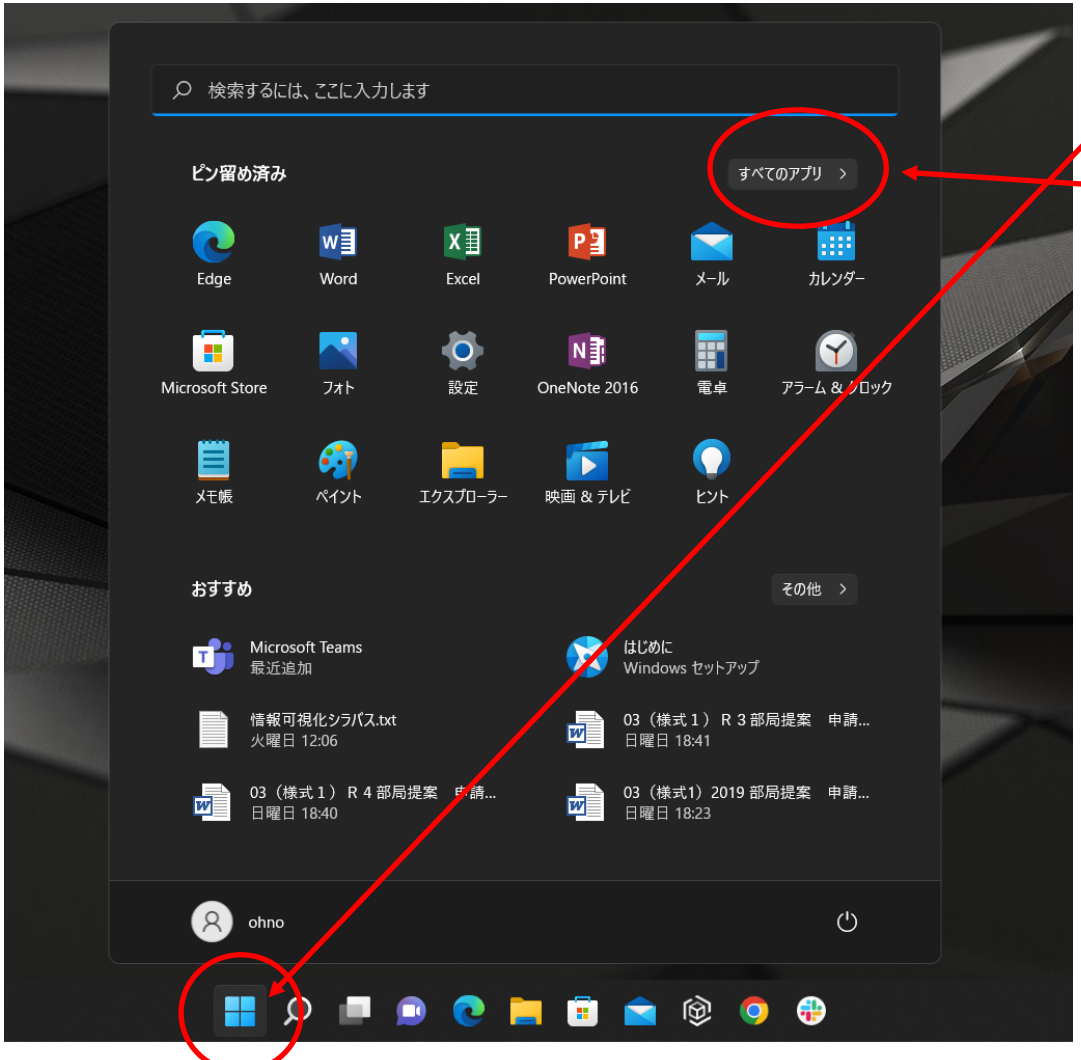
- 複数のCPUコアが搭載されているにもかかわらず、普通にプログラムを作ったのでは、1コアしか使用していない
- 例えば、4コアあるのに1コアしか使わないのでは、潜在能力の25%しか使わないようなもの
- 上の例では、普通のプログラムはPCの潜在能力の25パーセントしか使えないが、並列のプログラムでは残りの75パーセントも使用する(北〇神拳に近いかも)
- 一騎打ち戦法から集団戦法へ

手順

以下を実行すれば、自分のPCでMPIやOpenMPが練習できるようになります。

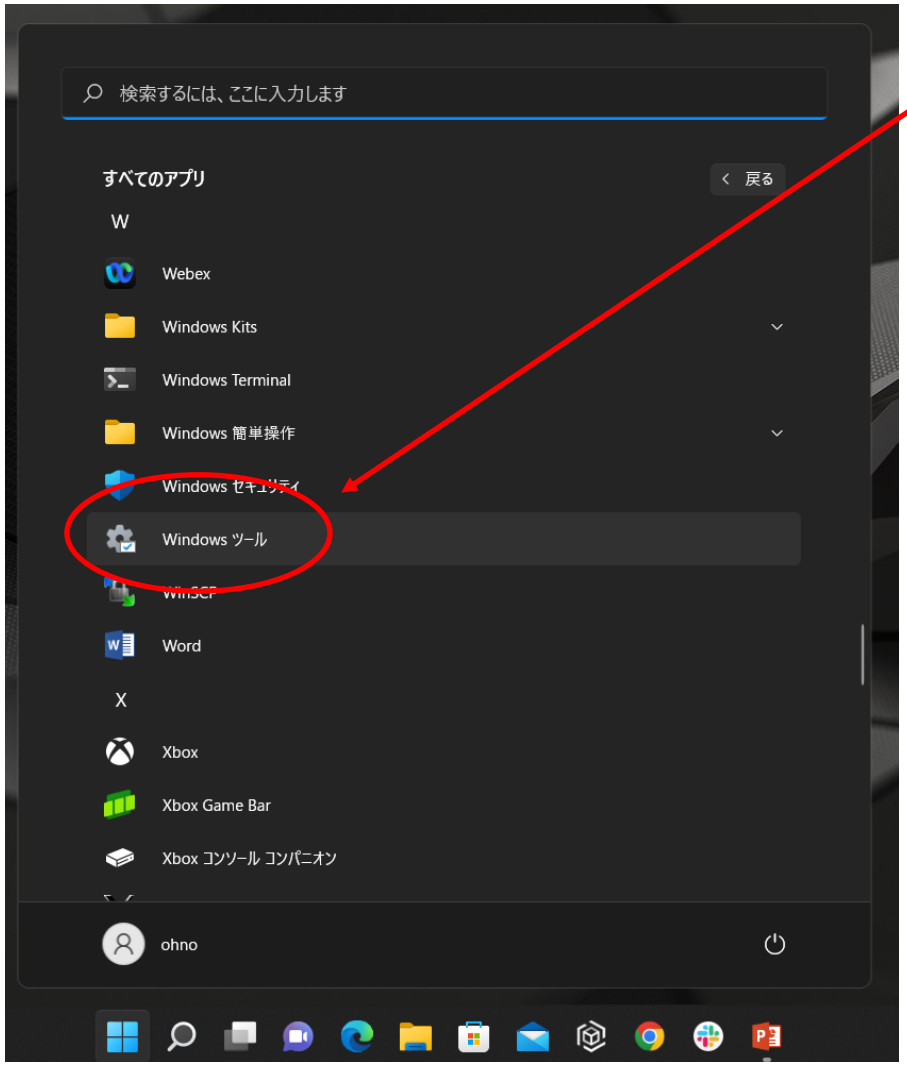
1. Windows Subsystem for LinuxでUbuntuをインストールする
   - コマンド一発でできるようになった！
2. X11系のアプリのインストール
3. gcc, g++, gfortran, makeのインストールと動作確認
4. OpenMPIのインストールと動作確認

Ubuntuのインストール -1



1. スタートボタン
2. 「すべてのアプリ」を選ぶ

Ubuntuのインストール -2

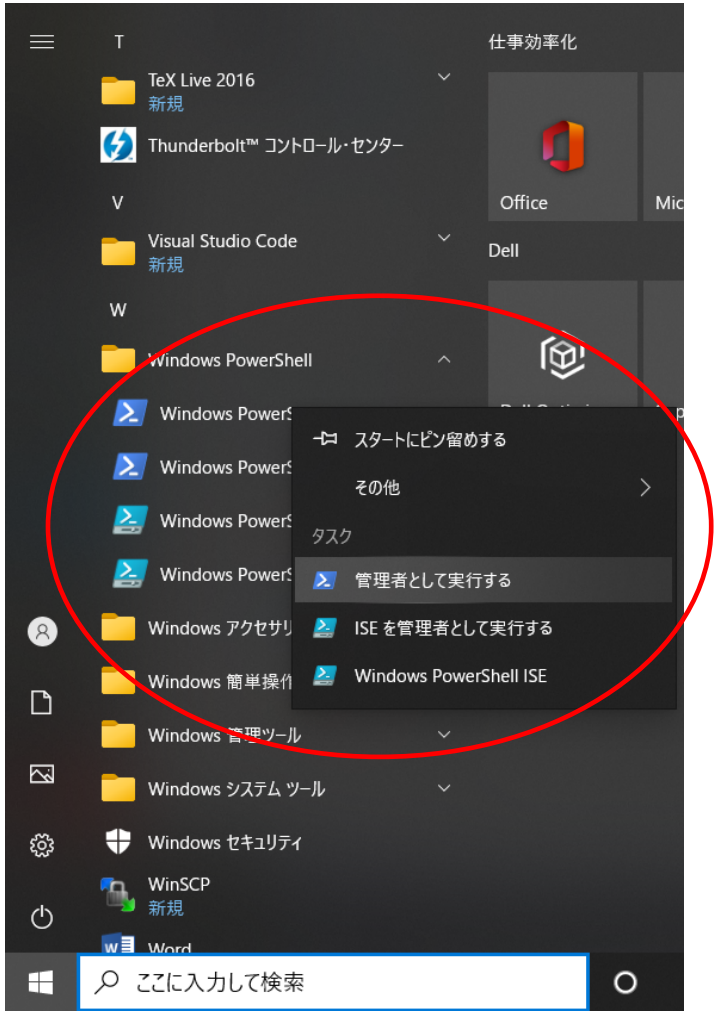


- 「Windowsツール」

Ubuntuのインストール -3

- Windows PowerShellを右クリック
- 「管理者として実行」

Windows10の場合



- 同様にWindows Power Shellを管理者として実行する

Ubuntuのインストール -4

```
PS C:¥WINDOWS¥system32> wsl --update
更新をチェック中...
更新をダウンロード中...
更新をインストール中...
この変更は、次回の WSL 再起動時に有効になります。強制的に再起動するには、'wsl --shutdown' を実行してください。
カーネル バージョン: 5.10.102.1
PS C:¥WINDOWS¥system32> wsl --shutdown
PS C:¥WINDOWS¥system32> wsl --install -d Ubuntu
ダウンロード中: Ubuntu
インストール中: Ubuntu
Ubuntu はインストールされました。
Ubuntu を起動しています...
PS C:¥WINDOWS¥system32>
```

1. wsl␣--update、wsl --shutdownでwslをアップデート、再起動する
2. wsl␣--install␣-d␣Ubuntuで Ubuntuをインストール

Ubuntuのインストール -5

```
ohno@Akagi: ~
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: ohno
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.72-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Apr 14 15:17:31 JST 2022

  System load:  0.0                Processes:             8
  Usage of /:   0.4% of 250.98GB   Users logged in:       0
  Memory usage: 0%                 IPv4 address for eth0: 172.17.95.47
  Swap usage:   0%

0 updates can be installed immediately.
0 of these updates are security updates.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


This message is shown once once a day. To disable it please create the
/home/ohno/.hushlogin file.
ohno@Akagi:~$
```

- 別なWindowが自動的に開くしばらくたつと
- Linuxで使うユーザー名とパスワードを聞いてくるので入力する（忘れないこと）

Ubuntuのインストール -6

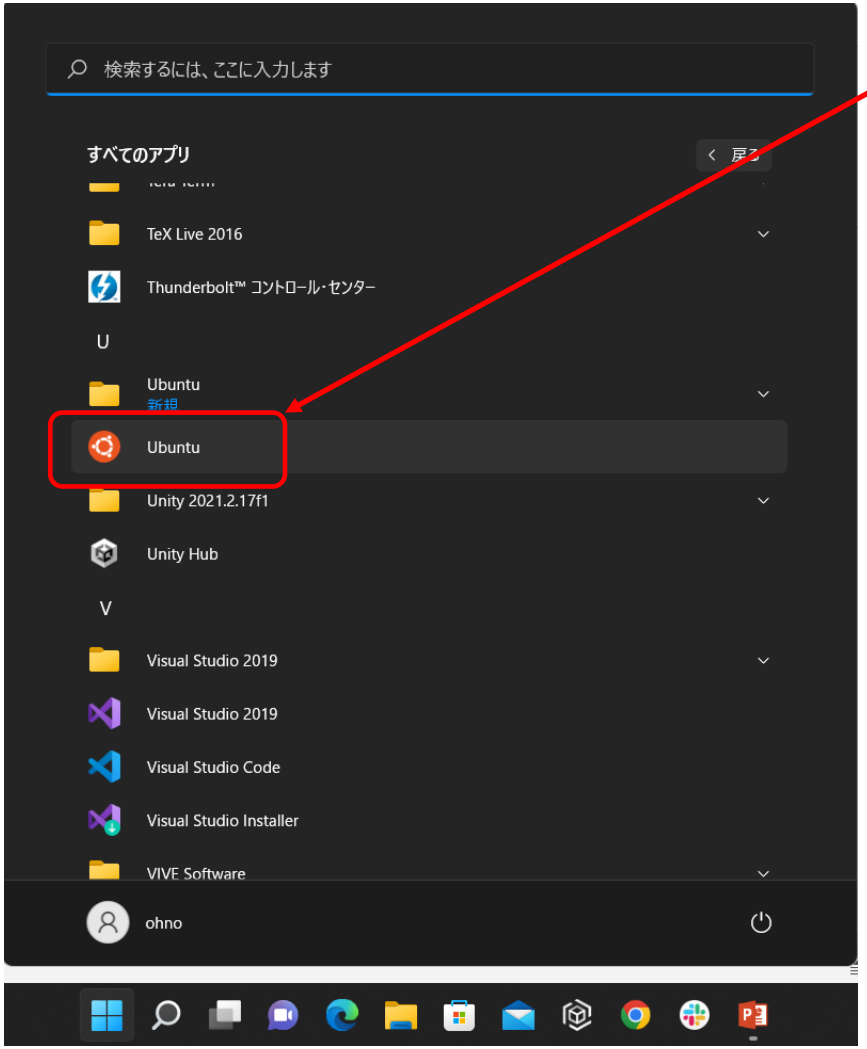
いろいろなソフトをインストールする前に、

- sudo apt-get update
- sudo apt-get upgrade

を入力する（重要！）

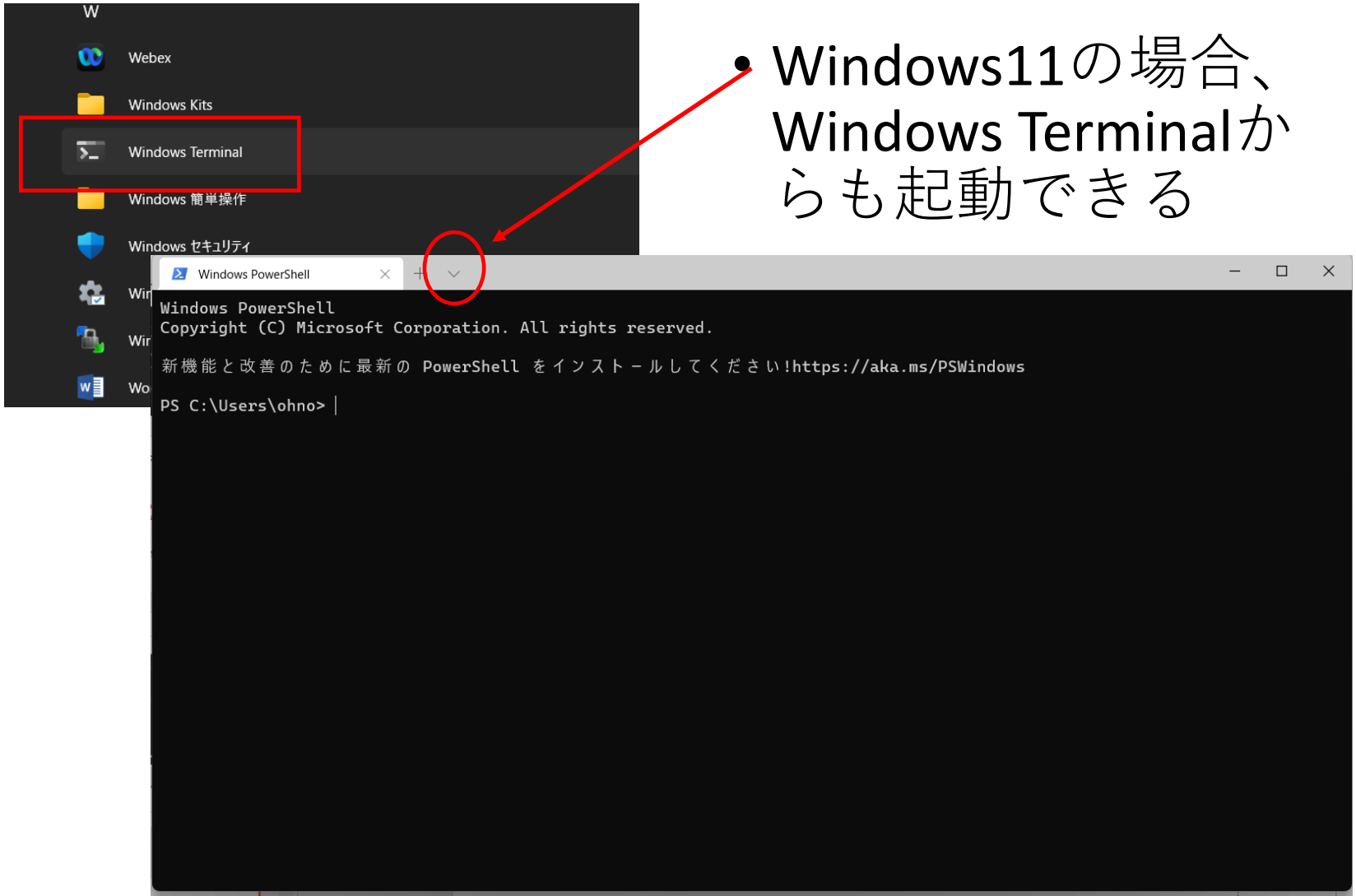
- 頻繁に実行しよう！

Ubuntuの起動 -1



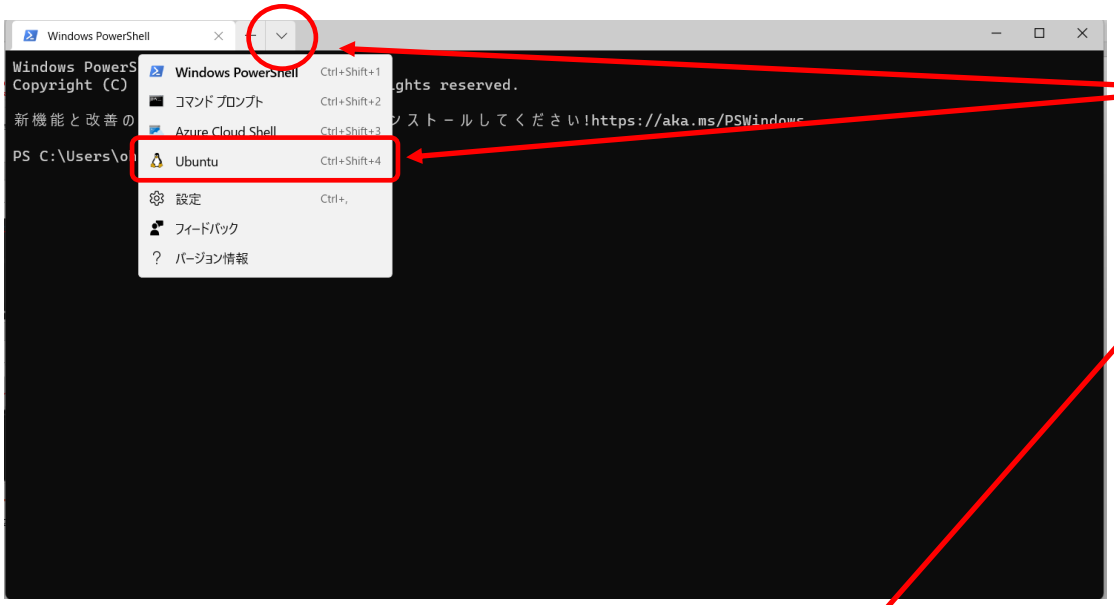
「すべてのアプリ」から探してみる

Ubuntuの起動 -2.1

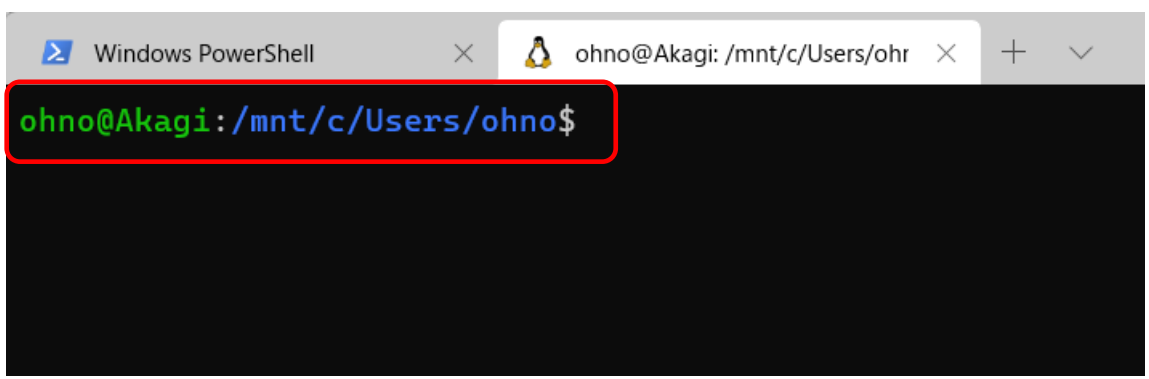


- Windows11の場合、Windows Terminalからも起動できる

Ubuntuの起動-2.2



- Ubuntuを選ぶ
- Ubuntuのタブが現れる
- ただし、カレントディレクトリがUbuntuのホームじゃないので、「cd」コマンドで直す



X11のアプリ -1

- グラフィカルなソフトが使えるようにしよう
- X11のアプリをインストール

ohno@Akagi: ~

```
ohno@Akagi:~$ sudo apt install x11-apps
```

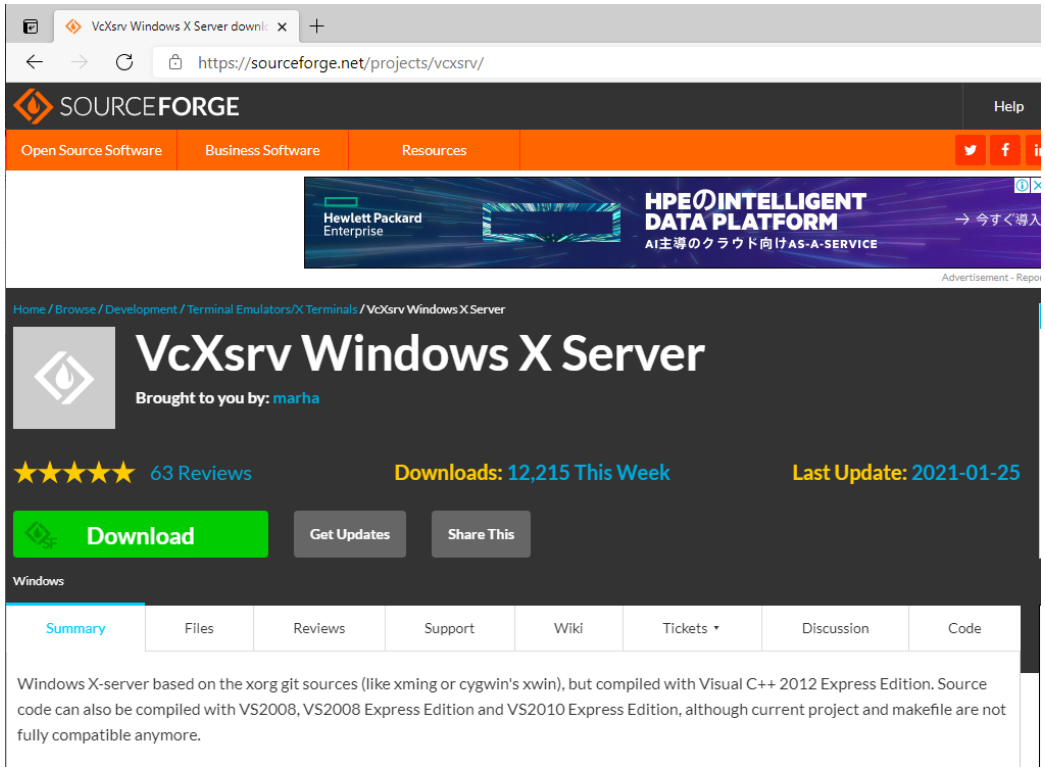
sudo apt install x11-apps

X11のアプリ-2

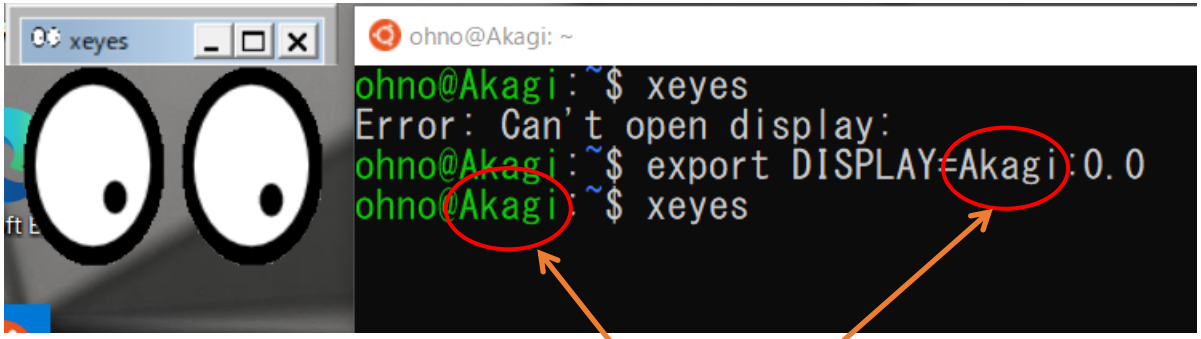
- xeyesで上のプログラムが起動する
- xcalc, xclockなども試してみる
- wslのアップデートを忘れてエラーが出て起動できない場合、Ubuntuを終了した後、Windows PowerShellで下記を実行して、Ubuntuを起動する
  - wsl --update
  - wsl --shutdown

Windows10の場合

- X11のアプリを使うためにVcXsrv Windows X serverというWindows用ソフトをインストールする。



Windows10の場合



VcXsrv Windows X serverを起動して

- export DISPLAY=[PCの名前]:0.0

として、環境変数を設定すると、x11のアプリのWindowがオープンできるようになる

- [PCの名前]は、localhostでもOK

X11のアプリ -3

- geditなど、GUIなエディタも利用できるので、インストールする
- xeyesを起動したままだと、次のコマンドが打てないので、右上の×を押して終了する
- onekoという可愛いソフトもある

gccなどのインストール -1

- gccとコマンドを打つと、not foundとでるが、インストールするためのコマンドも表示されるので、それをそのまま入力する
- sudo apt install gcc

gccなどのインストール -2

```
ohno@Akagi: ~
Setting up libgomp1:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libquadmath0:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libmpc3:amd64 (1.1.0-1) ...
Setting up libatomic1:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libubsan1:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libcrypt-dev:amd64 (1:4.4.10-10ubuntu4) ...
Setting up libisl22:amd64 (0.22.1-1) ...
Setting up libbinutils:amd64 (2.34-6ubuntu1.1) ...
Setting up libc-dev-bin (2.31-0ubuntu9.2) ...
Setting up libcc1-0:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up liblsan0:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libitm1:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up gcc-9-base:amd64 (9.3.0-17ubuntu1~20.04) ...
Setting up libtsan0:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libctf0:amd64 (2.34-6ubuntu1.1) ...
Setting up libasan5:amd64 (9.3.0-17ubuntu1~20.04) ...
Setting up cpp-9 (9.3.0-17ubuntu1~20.04) ...
Setting up libc6-dev:amd64 (2.31-0ubuntu9.2) ...
Setting up binutils-x86-64-linux-gnu (2.34-6ubuntu1.1) ...
Setting up binutils (2.34-6ubuntu1.1) ...
Setting up libgcc-9-dev:amd64 (9.3.0-17ubuntu1~20.04) ...
Setting up cpp (4:9.3.0-1ubuntu2) ...
Setting up gcc-9 (9.3.0-17ubuntu1~20.04) ...
Setting up gcc (4:9.3.0-1ubuntu2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
ohno@Akagi:~$
```

同様にして、g++, gfortran, makeをインストールする。

makeはmake-guileを選択する。

なお

- gcc: C Compiler
- g++: C++ Compiler
- gfortran: Fortran Compiler

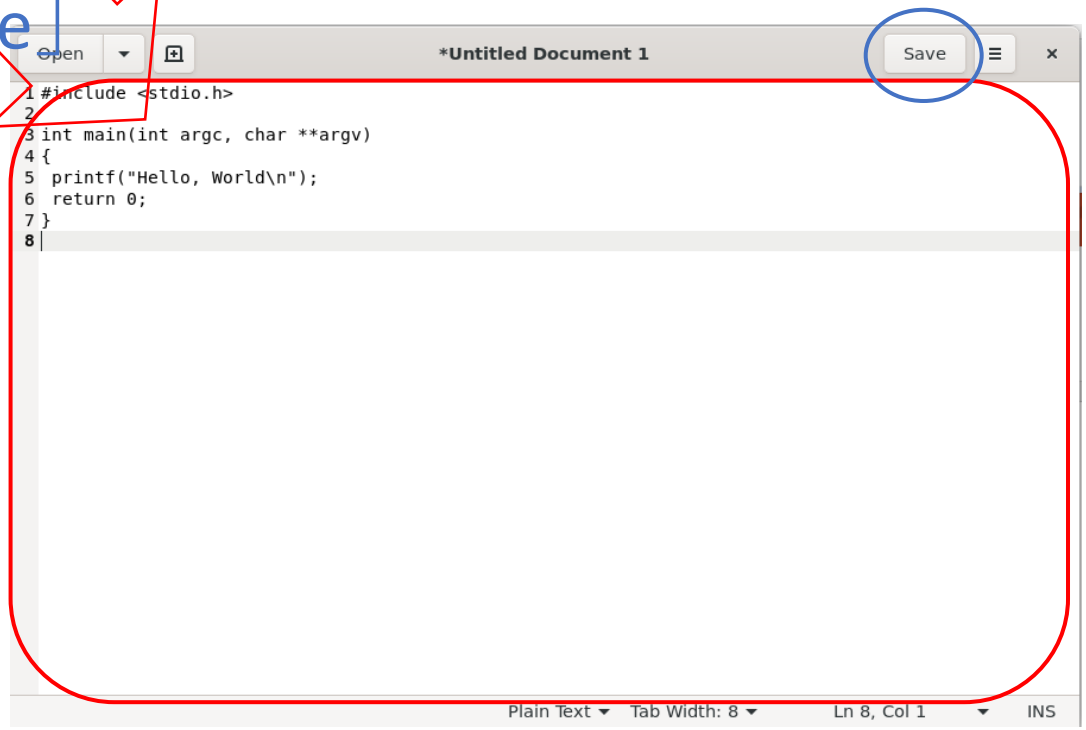
gccなどの動作テスト -1

C, C++, Fortranのプログラム入力と実行の手順

- geditなどのエディタでプログラムを入力する(テキストファイル)
- 入力したプログラムを保存する
- コンパイラでコンパイルする→実行ファイルができる
  - 文法エラーなどがあると、コンパイル時にエラーが出てコンパイルできない
  - 論理的なエラーは検出されない
- 実行ファイルを実行する

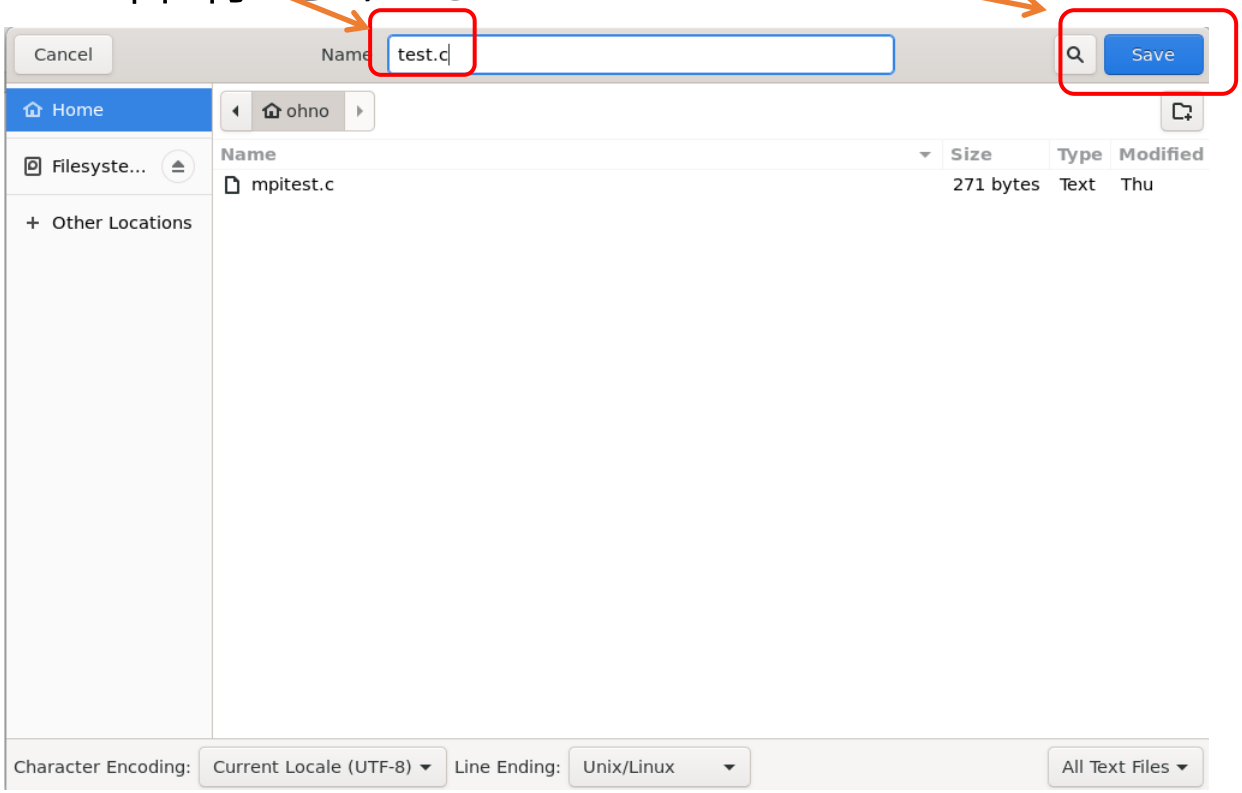
gccなどの動作テスト -2

1. gedit⏎として、geditを起動する
2. プログラムをここに入力
3. 終わったら「Save」ボタンを押す



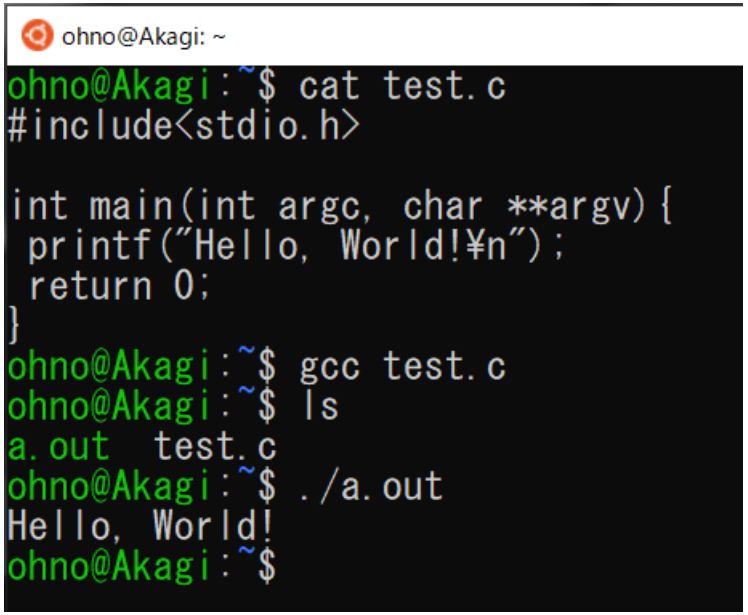
gccなどの動作テスト-3

- 名前を入力して、「Save」ボタンを押す
- ファイルが保存される



gccなどの動作テスト -4

```
#include<stdio.h>

int main(int argc,  char  **argv){
 printf("Hello, World!¥n");
 return 0;
}
```



1. これをgeditで入力して、test.cとして保存する
2. gcc ファイル名　とするとコンパイルできる
3. コンパイルすると、a.outという実行ファイルができる
4. カレントディレクトリを表す./をつけて実行する

- gcc -o test test.cとすると、a.outではなく、testという実行ファイルができる

gccなどの動作テスト -5

```
#include<iostream>

using namespace std;

int main(int argc, char **argv){
 cout << "Hello, World!(cpp)¥n";
 return 0;
}
```

test.cpp

- g++やgfortranでも、同じようにしてコンパイルできる
- これを入力してtest.cppとして保存
- g++ ファイル名

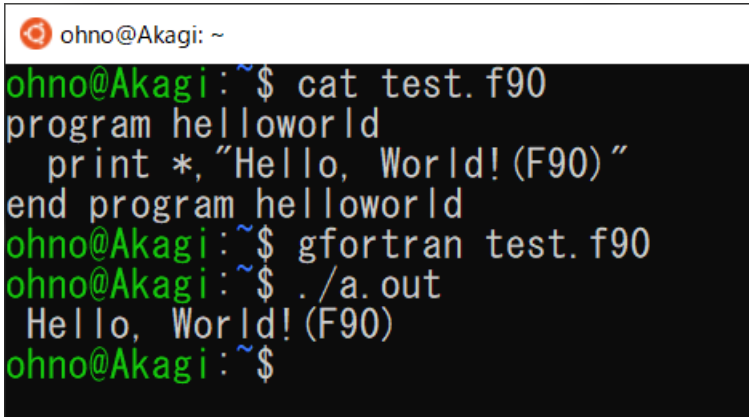
ohno@Akagi: ~

```
ohno@Akagi:~$ cat test.cpp
#include<iostream>

using namespace std;

int main(int argc, char **argv){
 cout << "Hello, World!(cpp)¥n";
 return 0;
}
ohno@Akagi:~$ g++ test.cpp
ohno@Akagi:~$ ./a.out
Hello, World!(cpp)
ohno@Akagi:~$
```

gccなどの動作テスト -6

```
program  helloworld
  print  *, "Hello, World!(F90)"
end  program  helloworld
```

test.f90



- g++やgfortranでも、同じようにしてコンパイルできる
- gfortran ファイル名

OpenMPの動作テスト -1

- 現在インストールされるgcc, g++, gfortranは、デフォルトでOpenMPの利用が可能
- コンパイル時に、-fopenmpオプションをつける

```
#include <stdio.h>
#include <omp.h>

int main(int argc, char **argv){
 int i;
#pragma omp parallel for private(i)
for(i=0; i<16; i++){
   printf("i=%d thread num = %d¥n",
          i, omp_get_thread_num());
}
 return 0;
}
```

omptest.c

```
ohno@Akagi: ~/mpitest
ohno@Akagi:~/mpitest$ cat omptest.c
#include <stdio.h>
#include <omp.h>

int main(int argc, char **argv){
 int i;
#pragma omp parallel for private(i)
for(int i=0; i<16; i++){
        printf("i=%d thread num = %d¥n",i, omp_get_thread_num());
}
 return 0;
}
ohno@Akagi:~/mpitest$
ohno@Akagi:~/mpitest$ gcc -fopenmp omptest.c
ohno@Akagi:~/mpitest$
```

C言語

OpenMPの動作テスト-2

- 現在インストールされるgcc, g++, gfortranは、デフォルトでOpenMPの利用が可能
- コンパイル時に、-fopenmpオプションをつける

```
program  omptest
 use   omp_lib
 integer  ::  i
!$omp  parallel   do  private(i)
do  i=0, 15
  print *,"i and thread num ", i, omp_get_thread_num()
end  do

end  program  omptest
```

omptest.f90

ohno@Akagi: ~/mpitest

```
ohno@Akagi:~/mpitest$ cat omptest.f90
program omptest
 use omp_lib
 integer :: i
!$omp parallel do private(i)
do i=0, 15
  print *,"i and thread num ",i,omp_get_thread_num()
end do

end program omptest
ohno@Akagi:~/mpitest$
ohno@Akagi:~/mpitest$ gfortran -fopenmp omptest.f90
ohno@Akagi:~/mpitest$
```

OpenMPの動作テスト-3

- 6コアのPCで実行したところ、スレッドが12(0～11)も生成された(Hyper Threading)
- 環境変数やOpenMPの関数でスレッド数を制御できる

```
ohno@Akagi:~/mpitest$ gcc -fopenmp omptest.c
ohno@Akagi:~/mpitest$ ./a.out
i=4 thread num = 2
i=5 thread num = 2
i=12 thread num = 8
i=2 thread num = 1
i=3 thread num = 1
i=11 thread num = 7
i=0 thread num = 0
i=1 thread num = 0
i=8 thread num = 4
i=13 thread num = 9
i=15 thread num = 11
i=10 thread num = 6
i=14 thread num = 10
i=9 thread num = 5
i=6 thread num = 3
i=7 thread num = 3
ohno@Akagi:~/mpitest$
```

C言語

```
ohno@Akagi:~/mpitest$ gfortran -fopenmp omptest.f90
ohno@Akagi:~/mpitest$ ./a.out
 i and thread num            0           0
 i and thread num            1           0
 i and thread num           14          10
 i and thread num            2           1
 i and thread num            3           1
 i and thread num           11           7
 i and thread num           13           9
 i and thread num            6           3
 i and thread num            7           3
 i and thread num           12           8
 i and thread num            4           2
 i and thread num            5           2
 i and thread num            8           4
 i and thread num           10           6
 i and thread num           15          11
 i and thread num            9           5
ohno@Akagi:~/mpitest$
```

Fortran

MPI -1

- MPIを利用するために、OpenMPIのインストールを行う
- mpirunとコマンドを打ってみる
- openmpi-binを選ぶ

ohno@Akagi: ~

```
ohno@Akagi:~$ mpirun

Command 'mpirun' not found, but can be installed with:

sudo apt install lam-runtime       # version 7.1.4-6build2, or
sudo apt install mpich             # version 3.3.2-2build1
sudo apt install openmpi-bin       # version 4.0.3-0ubuntu1
sudo apt install slurm-wlm-torque  # version 19.05.5-1

ohno@Akagi:~$ sudo apt install openmpi-bin
```

MPI -2

MPIを用いたプログラムのコンパイルには、

- mpicc：C言語のプログラム
- mpic++：C++言語のプログラム
- mpif90：Fortran言語のプログラム

を利用する。

後述するが、-fopenmpと組み合わせることも可能で、この場合、MPIとOpenMPのハイブリッド並列化される

MPIの動作テスト-1

ohno@Akagi: ~

```
ohno@Akagi:~$ cat mpitest.c
#include <stdio.h>
#include "mpi.h"

int main(int argc, char **argv){
  int np, myrank;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD,&np);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  printf("Hello, World myrank=%d size = %d¥n", myrank, np);
  MPI_Finalize();
  return 0;
}
ohno@Akagi:~$ mpicc mpitest.c
ohno@Akagi:~$
```

- C言語のMPIプログラムをmpiccでコンパイル
- 各プロセスが、Hello,Worldを表示する

MPIの動作テスト-1

```
#include <stdio.h>
#include  "mpi.h"

int  main(int  argc,  char  **argv){
  int  np,  myrank;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD,&np);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  printf("Hello, World myrank=%d size = %d¥n", myrank, np);
  MPI_Finalize();
  return  0;
}
```

mpitest.c

MPIの動作テスト-1

```
#include <iostream>
#include "mpi.h"

using  namespace  std;

int  main(int  argc,  char  **argv){
  int  np, myrank;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD,&np);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  cout  <<  "Hello, World myrank="<< myrank << " size = "<<np<<"¥n";
  MPI_Finalize();
  return  0;
}
```

mpitest.cpp

MPIの動作テスト-2

```
ohno@Akagi:~$ mpicc mpitest.c
ohno@Akagi:~$ mpirun -np 4 ./a.out
--------------------------------------------------------------------------
WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

  Local host: Akagi
--------------------------------------------------------------------------
Hello, World myrank=1 size = 4
Hello, World myrank=2 size = 4
Hello, World myrank=0 size = 4
Hello, World myrank=3 size = 4
[Akagi:00057] 3 more processes have sent help message help-btl-vader.txt / cma-permission-denied
[Akagi:00057] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
ohno@Akagi:~$
```

実行には、mpirunを使う

- mpirun –np [プロセス数] [program]

[プロセス数]のところには、MPIプロセスの数、[program]のところにはプログラム名

例では、プロセス数4で、カレントディレクトリのa.outを実行している（何か警告が出ている）

MPIの動作テスト-3

```
ohno@Akagi: ~
ohno@Akagi:~$ cat mpitest.f90
program mpitest
use mpi
implicit none


integer :: np, myrank
integer :: ierr

  call mpi_init(ierr)
  call mpi_comm_size(mpi_comm_world,np,ierr)
  call mpi_comm_rank(mpi_comm_world,myrank,ierr)
  print *,"Hello, World myrank and size = ", myrank, np
  call mpi_finalize(ierr)

end program mpitest
ohno@Akagi:~$ mpif90 mpitest.f90
```

Fortranでも同様に、プログラム実行にはmpirunを使う

```
ohno@Akagi:~$ mpirun -np 4 ./a.out
--------------------------------------------------------------------------
WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

  Local host: Akagi
--------------------------------------------------------------------------
 Hello, World myrank and size =           2           4
 Hello, World myrank and size =           3           4
 Hello, World myrank and size =           0           4
 Hello, World myrank and size =           1           4
[Akagi:00256] 3 more processes have sent help message help-btl-vader.txt / cma-permission-denied
[Akagi:00256] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
ohno@Akagi:~$
```

MPIの動作テスト-3

```
program  mpitest
use  mpi
implicit  none



integer :: np,  myrank
integer :: ierr

  call  mpi_init(ierr)
  call  mpi_comm_size(mpi_comm_world,np,ierr)
  call  mpi_comm_rank(mpi_comm_world,myrank,ierr)
  print  *,"Hello, World myrank and size = ", myrank, np
  call  mpi_finalize(ierr)

end  program  mpitest
```

mpitest.f90

MPIの動作テスト-4

ohno@Akagi: ~

```
ohno@Akagi:~$ mpirun -np 8 ./a.out
--------------------------------------------------------------------------
There are not enough slots available in the system to satisfy the 8
slots that were requested by the application:

  ./a.out

Either request fewer slots for your application, or make more slots
available for use.

A "slot" is the Open MPI term for an allocatable unit where we can
launch a process.  The number of slots available are defined by the
environment in which Open MPI processes are run:

  1. Hostfile, via "slots=N" clauses (N defaults to number of
     processor cores if not provided)
  2. The --host command line parameter, via a ":N" suffix on the
     hostname (N defaults to 1 if not provided)
  3. Resource manager (e.g., SLURM, PBS/Torque, LSF, etc.)
  4. If none of a hostfile, the --host command line parameter, or an
     RM is present, Open MPI defaults to the number of processor cores

In all the above cases, if you want Open MPI to default to the number
of hardware threads instead of the number of processor cores, use the
--use-hwthread-cpus option.

Alternatively, you can use the --oversubscribe option to ignore the
number of available slots when deciding the number of processes to
launch.
--------------------------------------------------------------------------
ohno@Akagi:~$
```

- 6コアのパソコンで8プロセスで起動しようとしたら、エラーが出た。コアの数を超えるプロセスは生成できない設定のようだ

MPIの動作テスト-5

ohno@Akagi: ~

```
ohno@Akagi:~$ mpirun -oversubscribe --mca btl_vader_single_copy_mechanism none -np 4 ./a.out
 Hello, World myrank and size =            2           4
 Hello, World myrank and size =            3           4
 Hello, World myrank and size =            0           4
 Hello, World myrank and size =            1           4
ohno@Akagi:~$ mpirun -oversubscribe --mca btl_vader_single_copy_mechanism none -np 8 ./a.out
 Hello, World myrank and size =            3           8
 Hello, World myrank and size =            5           8
 Hello, World myrank and size =            6           8
 Hello, World myrank and size =            1           8
 Hello, World myrank and size =            2           8
 Hello, World myrank and size =            4           8
 Hello, World myrank and size =            7           8
 Hello, World myrank and size =            0           8
ohno@Akagi:~$ alias mpirun='mpirun -oversubscribe --mca btl_vader_single_copy_mechanism none'
ohno@Akagi:~$ mpirun -np 4 ./a.out
 Hello, World myrank and size =            1           4
 Hello, World myrank and size =            2           4
 Hello, World myrank and size =            3           4
 Hello, World myrank and size =            0           4
ohno@Akagi:~$ mpirun -np 8 ./a.out
 Hello, World myrank and size =            0           8
 Hello, World myrank and size =            3           8
 Hello, World myrank and size =            4           8
 Hello, World myrank and size =            5           8
 Hello, World myrank and size =            6           8
 Hello, World myrank and size =            7           8
 Hello, World myrank and size =            1           8
 Hello, World myrank and size =            2           8
ohno@Akagi:~$
```

(解決法は検索して見つけました)

おまじないをつけることで、警告やエラーをでなくできる。

aliasという機能で、mpirun自体をおまじない付きにする

mpirun -oversubscribe --mca btl_vader_single_copy_mechanism none

MPI+OpenMPの動作テスト (C言語) -1

ohno@Akagi: ~/mpitest

```
ohno@Akagi:~/mpitest$ cat hybtest.c
#include <stdio.h>
#include <omp.h>
#include "mpi.h"

int main(int argc, char **argv){
  int i, np, myrank;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD,&np);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  omp_set_num_threads(3);
#pragma omp parallel for private(i)
for(i=0;i<3;i++){
  printf("Hello, World myrank=%d size = %d thread =%d¥n", myrank, np, omp_get_thread_num());
}
  MPI_Finalize();
  return 0;
}
ohno@Akagi:~/mpitest$ mpicc -fopenmp hybtest.c
ohno@Akagi:~/mpitest$
```

- MPIとOpenMP両方で並列化するプログラムのコンパイルには、mpiccやmpif90に-fopenmpオプションを付け加えればよい

MPI+OpenMPの動作テスト (C言語) -1

```
#include <stdio.h>
#include <omp.h>
#include "mpi.h"

int  main(int  argc,  char  **argv){
  int i, np, myrank;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD,&np);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  omp_set_num_threads(3);
#pragma omp parallel for private(i)
for(i=0;i<3;i++){
  printf("Hello, World myrank=%d size = %d thread =%d¥n",
        myrank, np, omp_get_thread_num());
}
  MPI_Finalize();
  return  0;
}
```

hybtest.c

MPI+OpenMPの動作テスト (C言語) -2

```
ohno@Akagi:~/mpitest$ mpicc -fopenmp hybtest.c
ohno@Akagi:~/mpitest$ mpirun -np 2 ./a.out
Hello, World myrank=1 size = 2 thread =2
Hello, World myrank=1 size = 2 thread =1
Hello, World myrank=1 size = 2 thread =0
Hello, World myrank=0 size = 2 thread =0
Hello, World myrank=0 size = 2 thread =2
Hello, World myrank=0 size = 2 thread =1
ohno@Akagi:~/mpitest$
```

- このプログラムでは、omp_set_num_threads(3)でスレッド数を3にしている
- 実行すると、各MPIプロセス(myrank=0, 1)で、OpenMPのスレッドが3(thread = 0, 1, 2)つ生成される
- このプログラムでは、MPIプロセスx2、各MPIプロセスでOpenMPスレッドx3なので、合計2x3=6コア使う

MPI+OpenMPの動作テスト (Fortran) -3

ohno@Akagi: ~/mpitest

```
ohno@Akagi:~/mpitest$ cat hybtest.f90
program hybridtest
use mpi
use omp_lib
implicit none


integer :: i, np, myrank
integer :: ierr

  call mpi_init(ierr)
  call mpi_comm_size(mpi_comm_world,np,ierr)
  call mpi_comm_rank(mpi_comm_world,myrank,ierr)
  call omp_set_num_threads(3)
  !$omp parallel do private(i)
  do i=0, 2
    print *,"Hello, World myrank, size, threadn = ", myrank, np, omp_get_thread_num()
  end do
  call mpi_finalize(ierr)

end program hybridtest
ohno@Akagi:~/mpitest$ mpif90 -fopenmp hybtest.f90
ohno@Akagi:~/mpitest$ mpirun -np 2 ./a.out
 Hello, World myrank, size, threadn =           1           2           1
 Hello, World myrank, size, threadn =           1           2           0
 Hello, World myrank, size, threadn =           1           2           2
 Hello, World myrank, size, threadn =           0           2           1
 Hello, World myrank, size, threadn =           0           2           0
 Hello, World myrank, size, threadn =           0           2           2
ohno@Akagi:~/mpitest$
```

- これで並列計算の練習ができるようになった！

MPI+OpenMPの動作テスト (Fortran) -3

```
program hybridtest
use mpi
use omp_lib
implicit none

integer :: i, np, myrank
integer :: ierr

  call mpi_init(ierr)
  call mpi_comm_size(mpi_comm_world,np,ierr)
  call mpi_comm_rank(mpi_comm_world,myrank,ierr)
  call omp_set_num_threads(3)
  !$omp parallel do private(i)
  do i=0, 2
    print *,"Hello, World myrank, size, threadn = ", myrank, np, omp_get_thread_num()
  end do
  call mpi_finalize(ierr)

end program hybridtest
```

hybtest.f90

.bashrcの編集 -1

- mpirunをubuntu起動時に自動で設定したい
- .bashrcに付け加えることで実現できる

ohno@Akagi: ~

```
alias l='ls -CF'

# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|s
s/^¥s*[0-9]¥+¥s*//;s/[;&|]¥s*alert$//'¥'')"'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

alias mpirun='mpirun -oversubscribe -mca btl_vader_single_copy_mechanism none'

ohno@Akagi:~$
```

mpirunのaliasを.bashrcに書いてしまう

Windows10の場合

- displayもmpirunもubuntu起動時に設定したい
- .bashrcに付け加えることで実現できる

ohno@Akagi: ~

```
# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(histor
y|tail -n1|sed -e '¥''s/^¥s*[0-9]¥+¥s*//;s/[;&|]¥s*alert$//'¥'')"'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export DISPLAY=Akagi:0.0
alias mpirun='mpirun -oversubscribe --mca btl_vader_single_copy_mechanism none'

".bashrc" 121L, 3878C written                                    118,0-1       Bot
```

mpirunだけでなくDISPLAYのことも.bashrcに書いてしまう

.bashrcの編集 -2

- .bashrcは、ホームディレクトリにある
- lsでは見えない。ls -a で見える
- gedit␣.bashrc⏎とすると、geditで編集できるようになる
- 「Save」ボタンを押して上書き保存する

Winとのファイルのやりとり-1

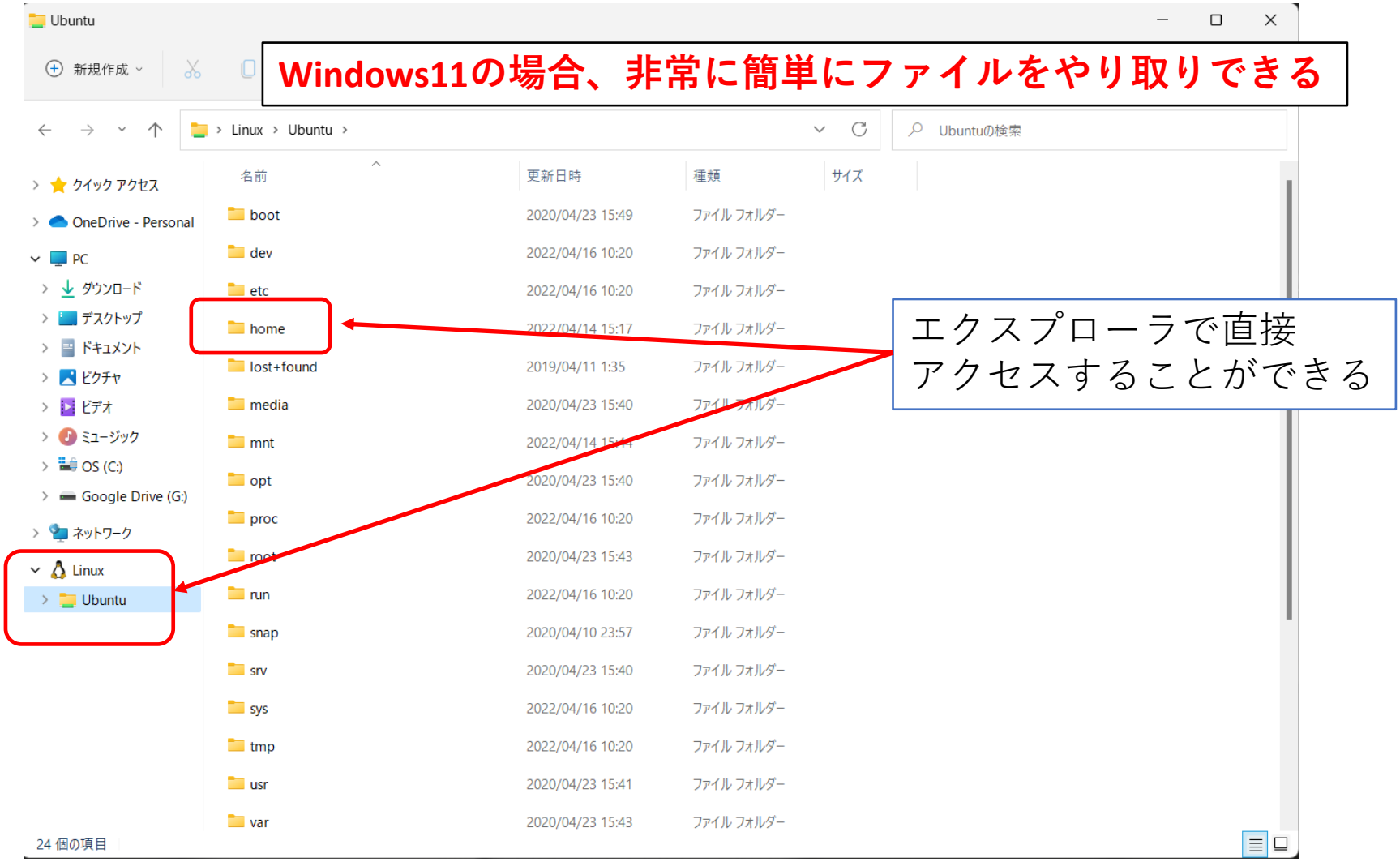
- Ubuntuから、Cドライブなどにアクセスしたい場合は、/mnt/ドライブ とする
- 例えば、c:¥dataにあるcのプログラム(拡張子がc)のファイルをカレントディレクトリにコピーしたい場合は、

cp␣/mnt/c/data/*.c␣.⏎

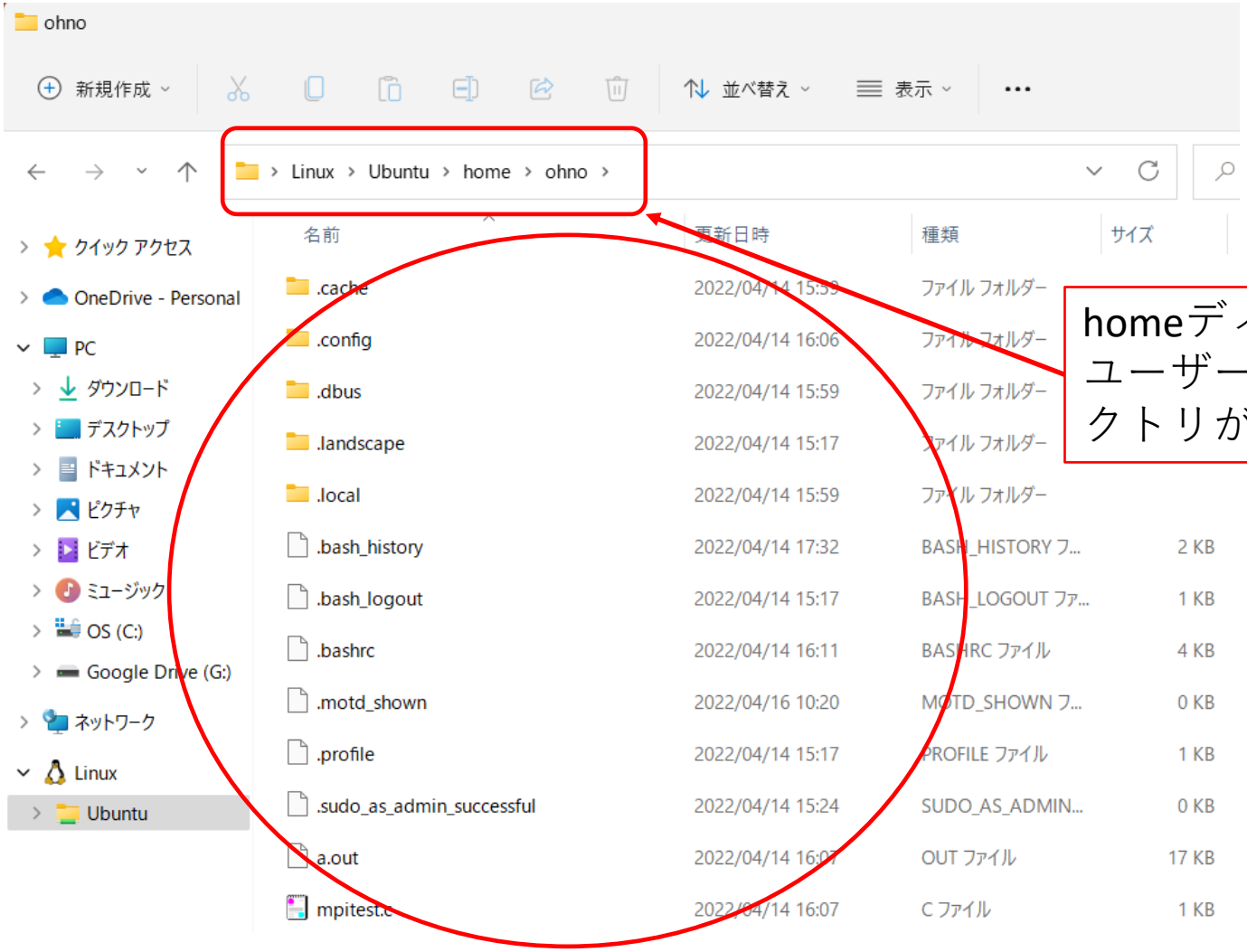
とすればよい。

これでWindows側とファイルのやり取りができる。

Winとのファイルのやりとり-2



Winとのファイルのやりとり-3



homeディレクトリの中にユーザーのホームディレクトリがある

付録

アンインストールしたはずのUbuntuがエクスプローラやWindows Terminalに現れる場合

- Windows PowerShellで
  - 「wsl -l -v」としてインストールされたUbuntuのバージョンやステイタスを見る
  - アンインストールしたものは「STATE」がSTOPPEDとなっているので、それを「wsl --unregister XXXX」として削除する。XXXXはUbuntuのディストリビューション名(NAMEのところでわかる)を入れる。