

Windows PC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
- WSL1 or WSL2を利用する

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○神拳に近いかも)
- 一騎打ち戦法から集団戦法へ

手順

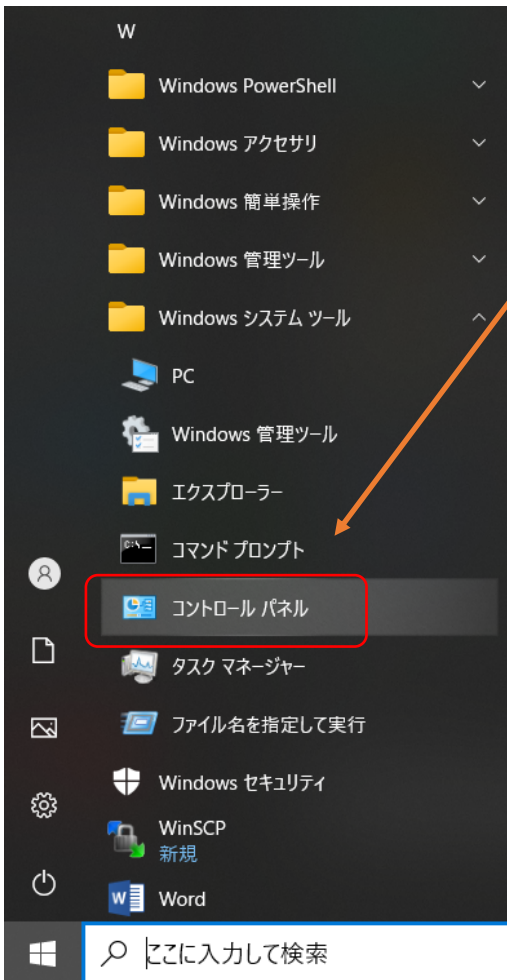
以下を実行すれば、自分のPCでMPIやOpenMPが練習できるようになります。

1. Windows Subsystem for LinuxでUbuntuをインストールする
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

Windows Subsystem for Linux

- Windows 10には、WSLという仕組みがあり、それを使うと簡単にLinuxがインストールできる
- WSL1かWSL2、どちらかを選びUbuntuというLinuxをインストールしてみよう

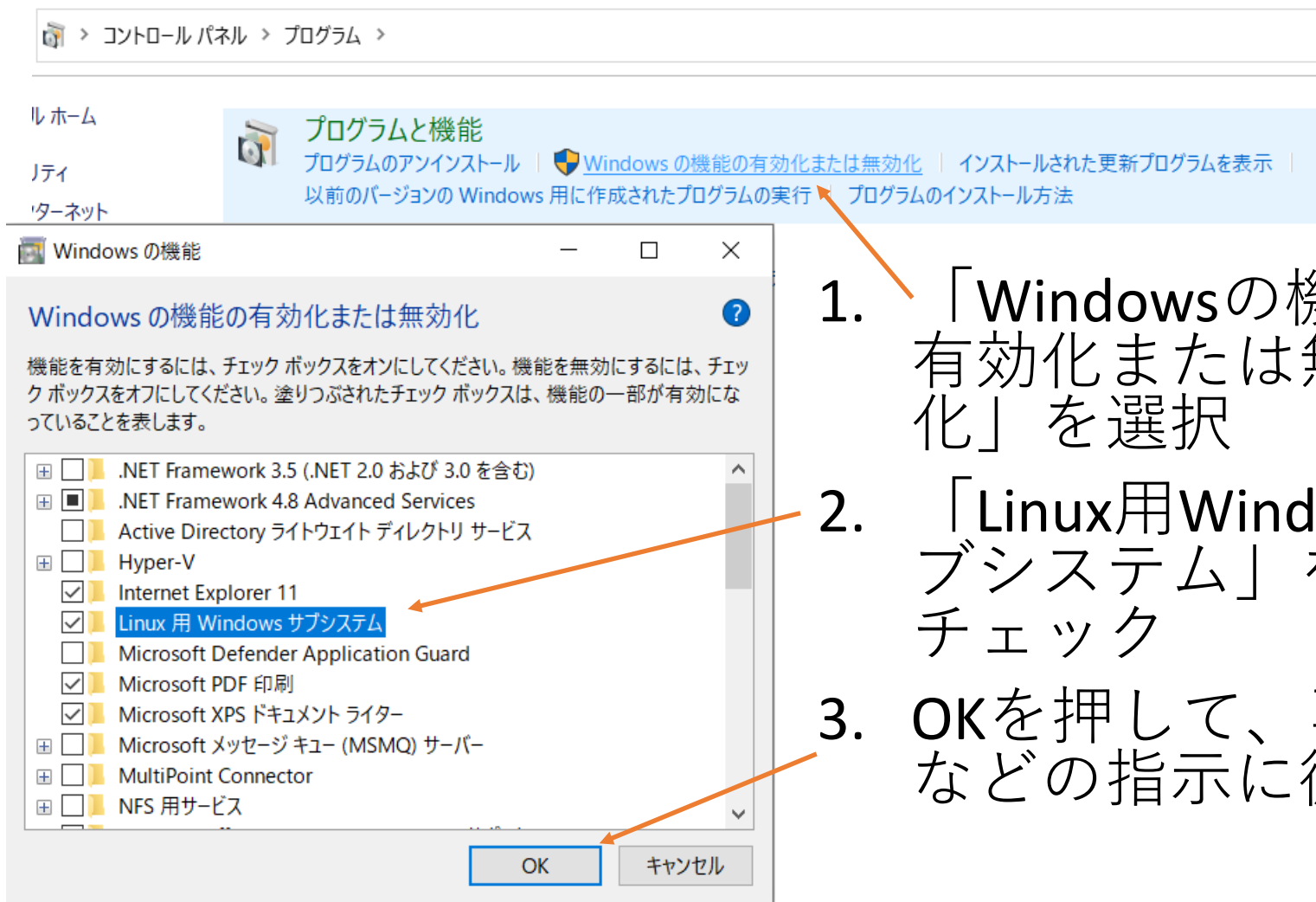
WSL1 -1



1. コントロールパネルをひらく
 1. スタートボタン
 2. Windowsシステムツール
2. プログラムをクリック

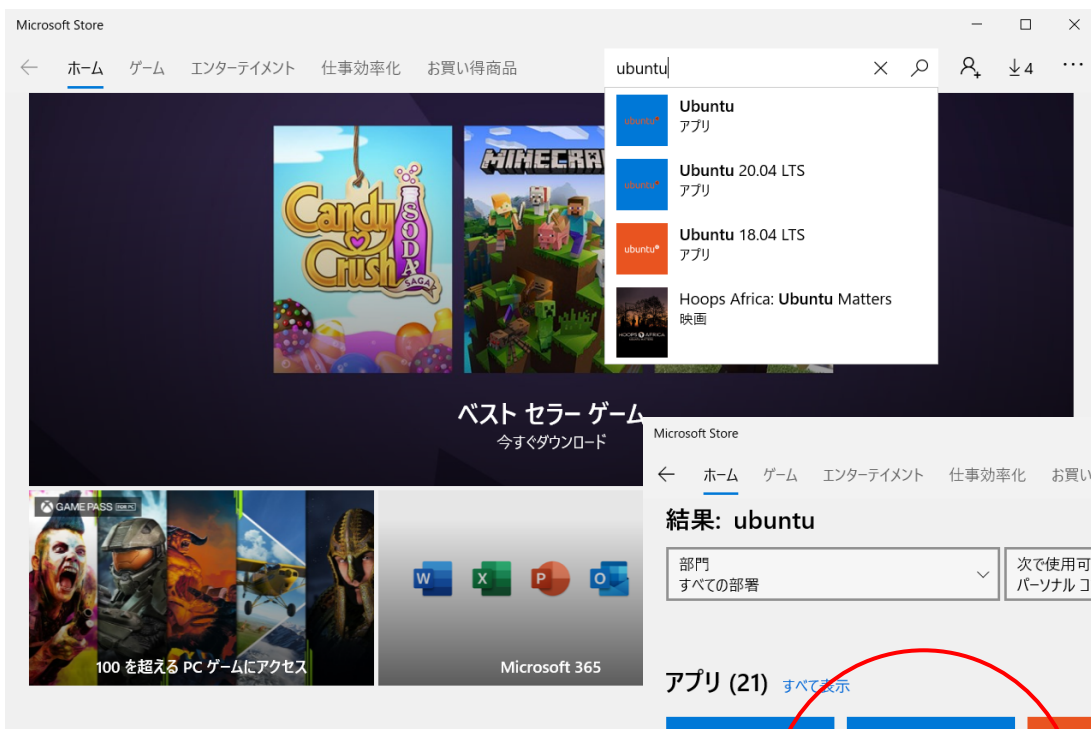


WSL1 -2



1. 「Windowsの機能の有効化または無効化」を選択
2. 「Linux用Windowsサブシステム」をチェック
3. OKを押して、再起動などの指示に従う

WSL1 -3



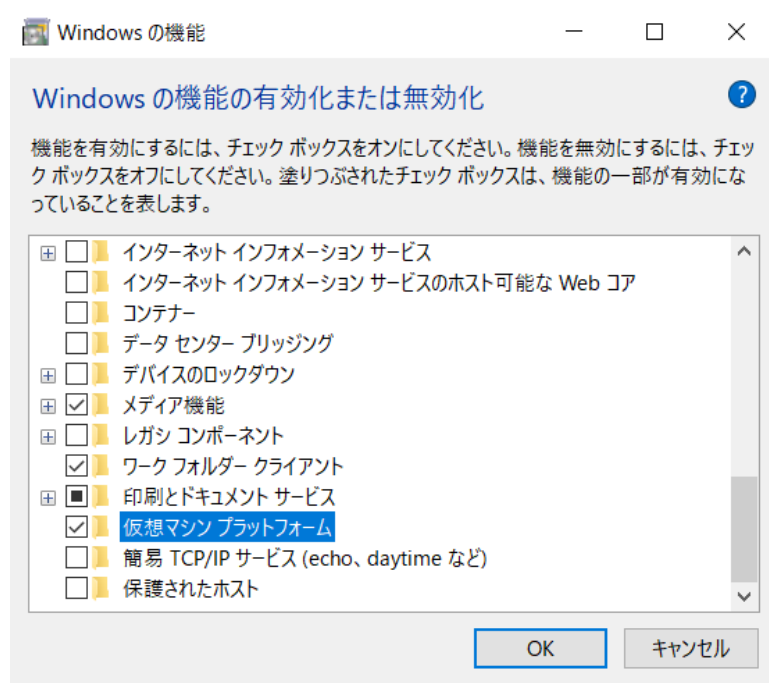
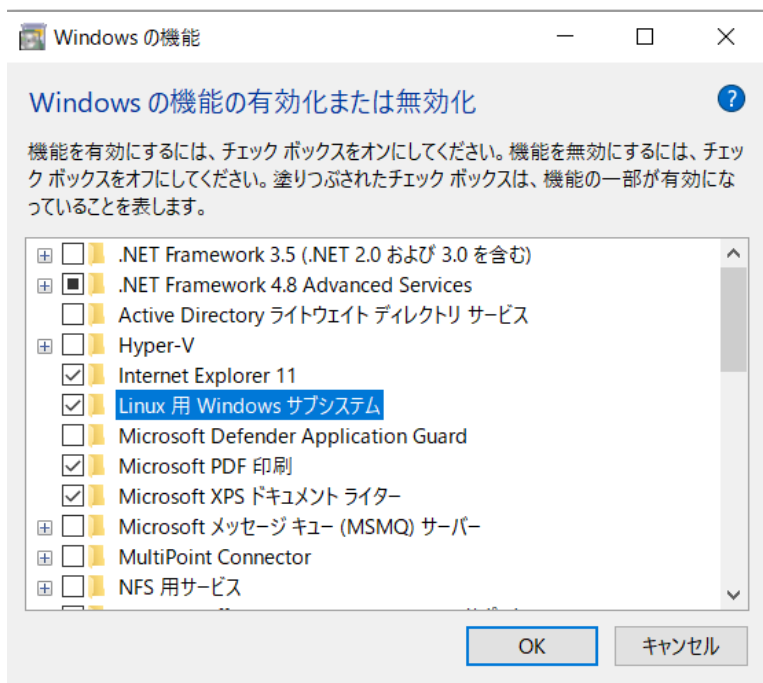
1. Microsoft StoreでUbuntuを検索し、入手、インストールする
2. パスワードを入れたりするとしばらくするとインストールが完了する

WSL2 -1

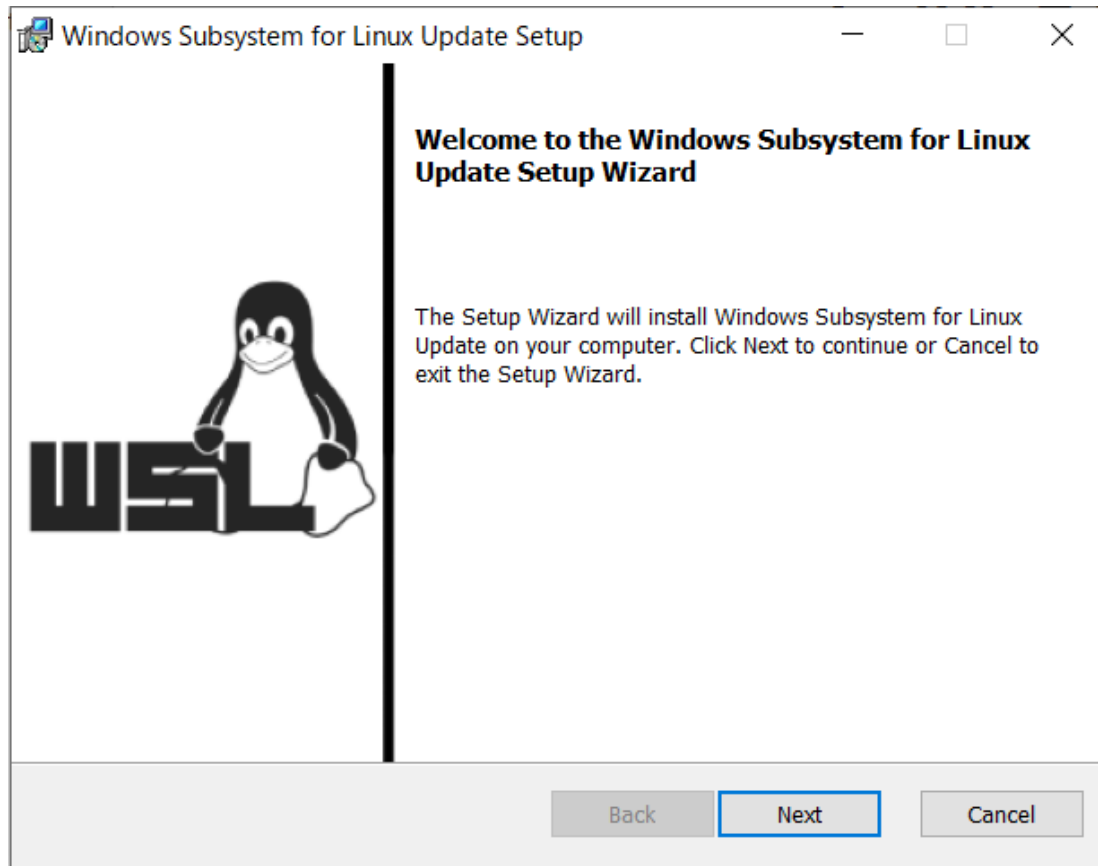
<https://docs.microsoft.com/ja-jp/windows/wsl/install-win10>

を参考にインストールしてみる

「Linux用サブシステム」と「仮想マシンプラットフォーム」をチェックし、OK。再起動等



WSL2 -2

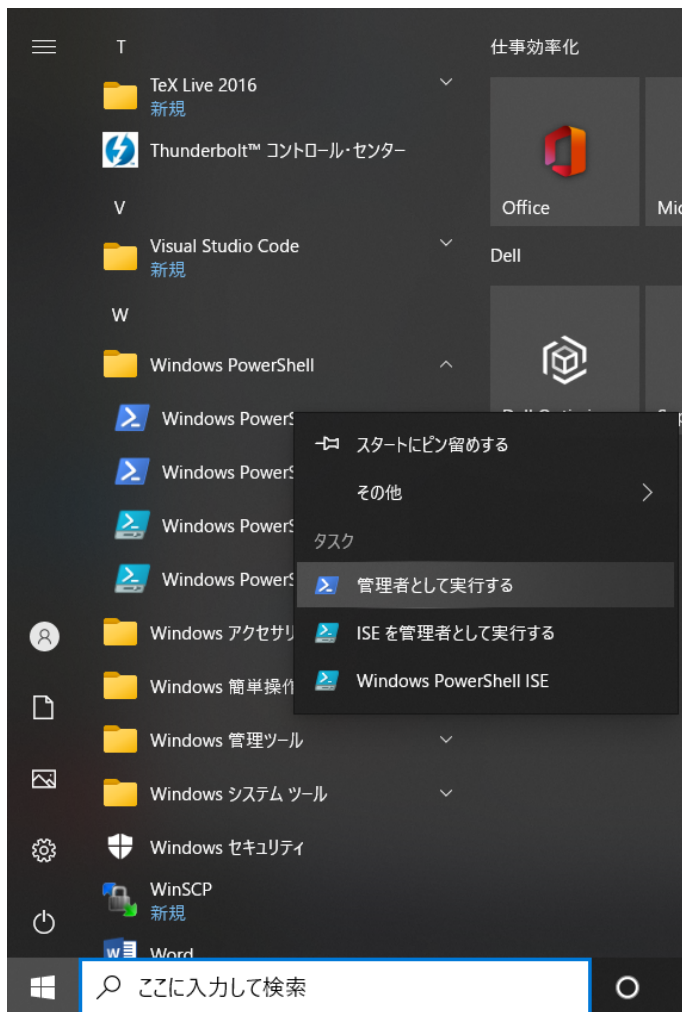


- LINUXカーネル更新プログラムのインストール

https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi

これをダウンロード、実行

WSL2 -3



管理者: Windows PowerShell

```
PS C:\Windows\system32> wsl --set-default-version 2
WSL 2 との主な違いについては、https://aka.ms/wsl2 を参照してください
PS C:\Windows\system32>
```

- Windows Power Shellを管理者として実行し

```
wsl --set-default-version 2
```

WSL2 -4

- WSL1と同様にストアでubuntuを検索、インストールする



↓ この製品はインストール済みです。 起動 ...

Ubuntu 20.04 LTS

Canonical Group Limited • 開発者ツール > ユーティリティ

★★★★★ 7 共有

Ubuntu 20.04 LTS on Windows allows you to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.

[表示数を増やす](#)

IA RC 3+ 欲しい物リスト

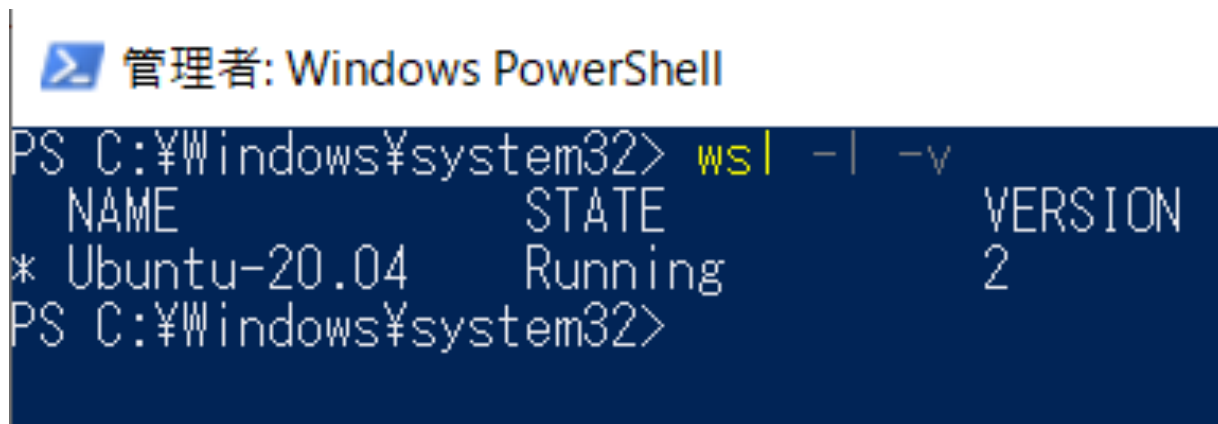
起動して、ユーザー名
パスワードを設定

WSL2 -5

- Power Shellでバージョンを確認

```
wsl -l -v ↵
```

確かに、Version 2になっている



```
管理者: Windows PowerShell
PS C:\Windows\system32> wsl -l -v
  NAME                STATE      VERSION
* Ubuntu-20.04        Running    2
PS C:\Windows\system32>
```

Windows Subsystem for Linux

いろいろなソフトをインストールする前に、

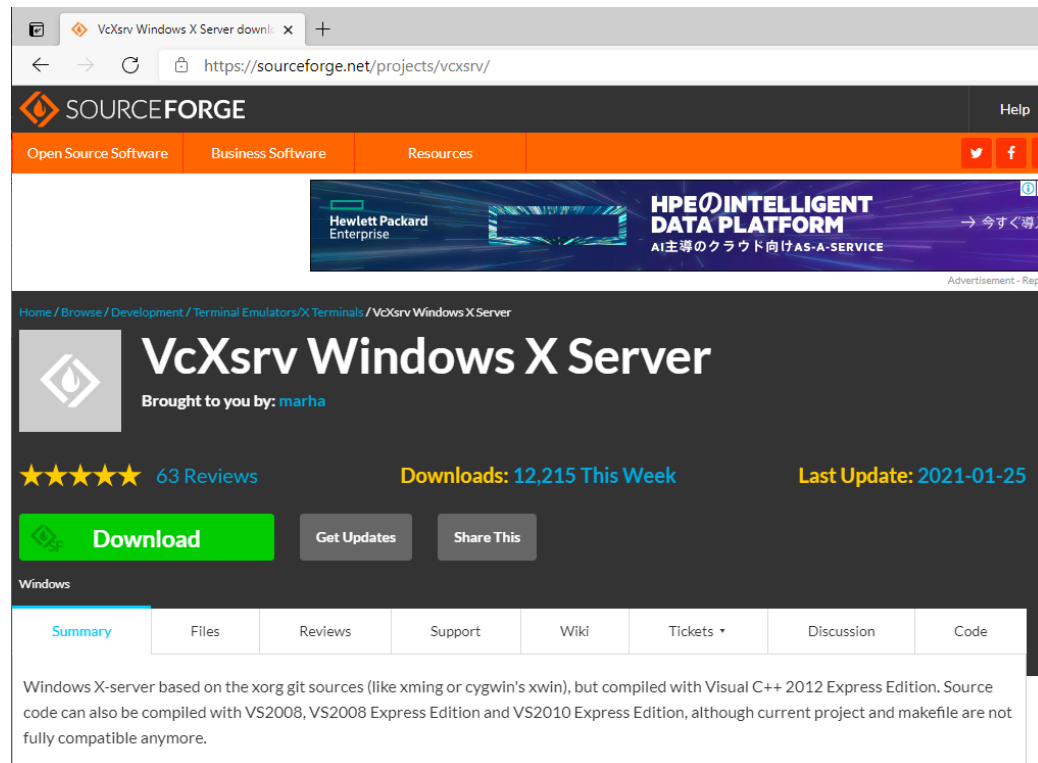
- `sudo apt-get update`
- `sudo apt-get upgrade`

を入力する（重要！）

- 頻繁に実行しよう！

X11のアプリ


- 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。



The screenshot shows a web browser window displaying the SourceForge project page for VcXsrv Windows X Server. The browser's address bar shows the URL <https://sourceforge.net/projects/vcxsrv/>. The SourceForge logo is visible at the top left, and the project title "VcXsrv Windows X Server" is prominently displayed. Below the title, it says "Brought to you by: marha". The page features a green "Download" button, a "Get Updates" button, and a "Share This" button. The project has a rating of 5 stars (4.5 shown) based on 63 reviews, and it has 12,215 downloads this week. The last update date is 2021-01-25. The page also includes a navigation menu with options like Summary, Files, Reviews, Support, Wiki, Tickets, Discussion, and Code. A description at the bottom states: "Windows X-server based on the xorg git sources (like xming or cygwin's xwin), but compiled with Visual C++ 2012 Express Edition. Source code can also be compiled with VS2008, VS2008 Express Edition and VS2010 Express Edition, although current project and makefile are not fully compatible anymore."

Xのアプリ (WSL1) -1

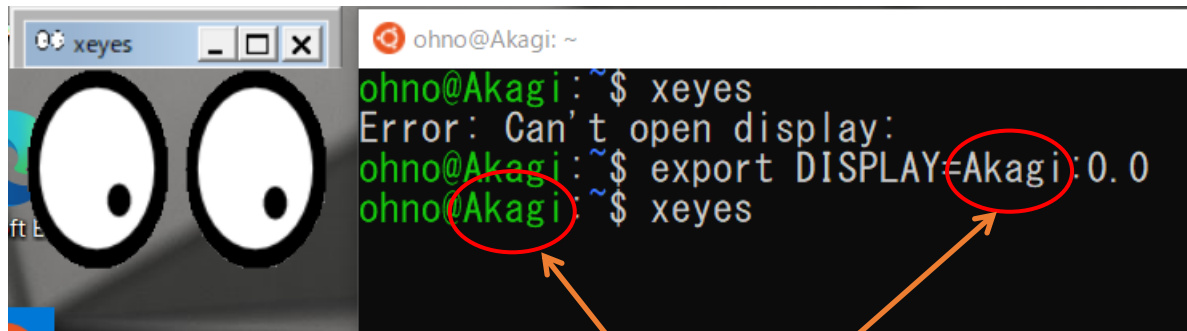
- グラフィカルなソフトが使えるようにしよう
- X11のアプリをインストール

 ohno@Akagi: ~

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

`sudo apt install x11-apps`

Xのアプリ (WSL1) -2



```
ohno@Akagi: ~  
ohno@Akagi: ~$ xeyes  
Error: Can't open display:  
ohno@Akagi: ~$ export DISPLAY=Akagi:0.0  
ohno@Akagi: ~$ xeyes
```

VcXsrv Windows X serverを起動して

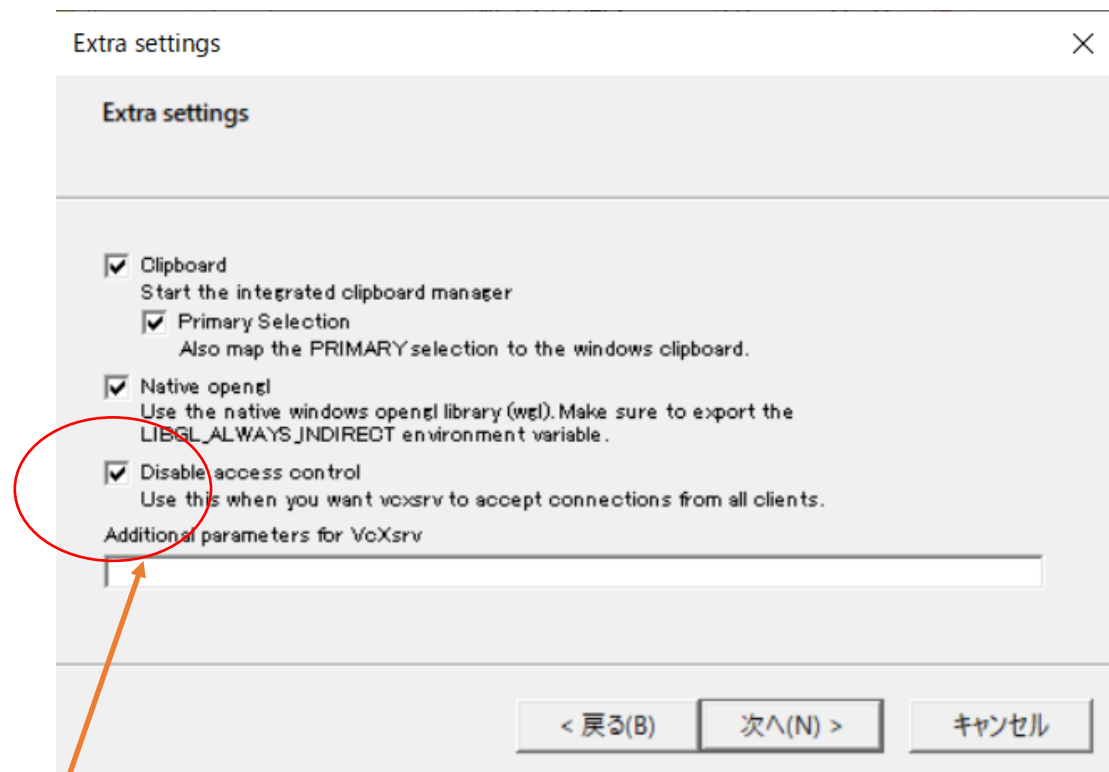
- export DISPLAY=[PCの名前]:0.0

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

Xのアプリ (WSL2) -1

VcXsrv Windows X Serverの起動時に



ここにチェックを入れる

Xのアプリ (WSL2) -2

以下をUbuntuのWindowで入力

```
export DISPLAY=$(cat /etc/resolv.conf | grep nameserver  
| awk '{print $2}'):0.0
```

A terminal window titled 'xeyes' is shown. The prompt is 'ohno@Akagi: ~'. The user enters the command 'export DISPLAY=\$(cat /etc/resolv.conf | grep nameserver | awk '{print \$2}'):0.0' and then 'xeyes'. The output shows two large, cartoonish eyes following the cursor.

```
ohno@Akagi: ~  
$ export DISPLAY=$(cat /etc/resolv.conf | grep nameserver | awk '{print $2}'):0.0  
ohno@Akagi: ~  
$ xeyes
```

WSL2のXについては、
いろいろなサイトを見て回りました。
例えば、

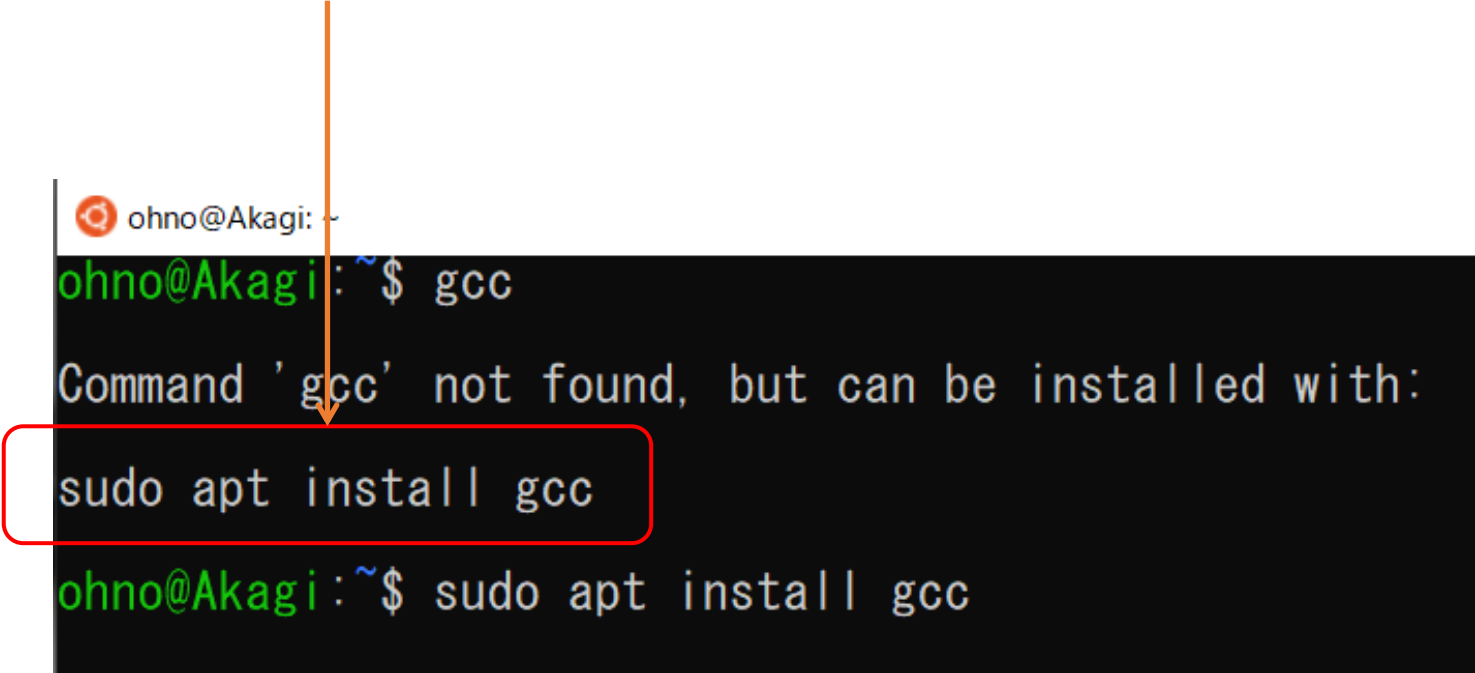
<https://github.com/microsoft/WSL/issues/4106#issuecomment-501885675>

Xのアプリ(共通) -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`



```
ohno@Akagi: ~$ gcc
Command 'gcc' not found, but can be installed with:
sudo apt install gcc
ohno@Akagi: ~$ 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

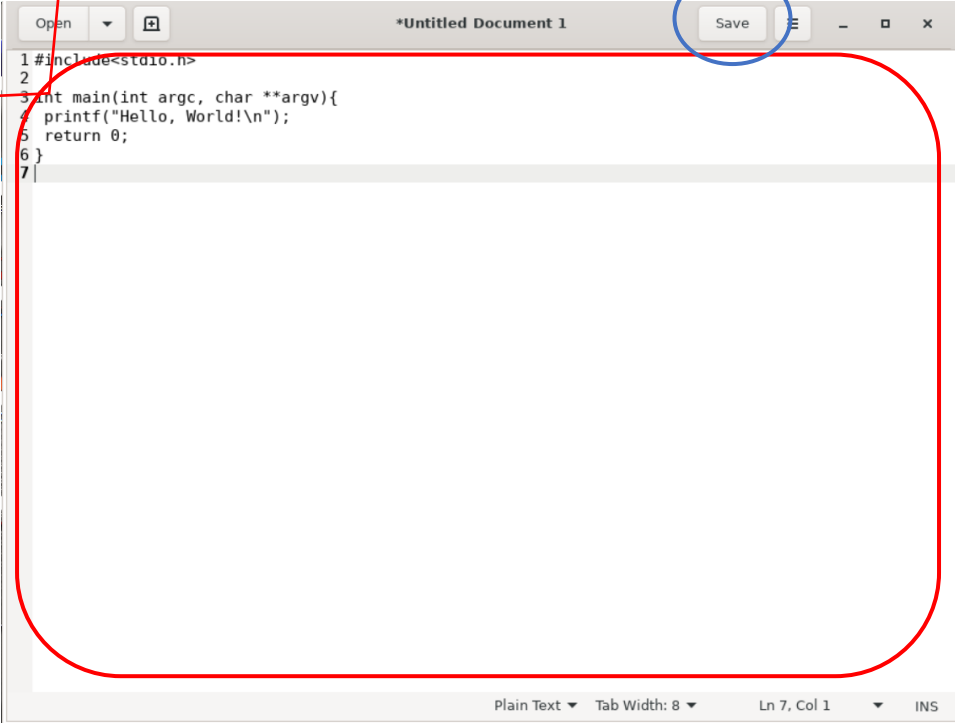
gccなどの動作テスト -1

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



The screenshot shows the gedit text editor window titled '*Untitled Document 1'. The editor contains the following C code:

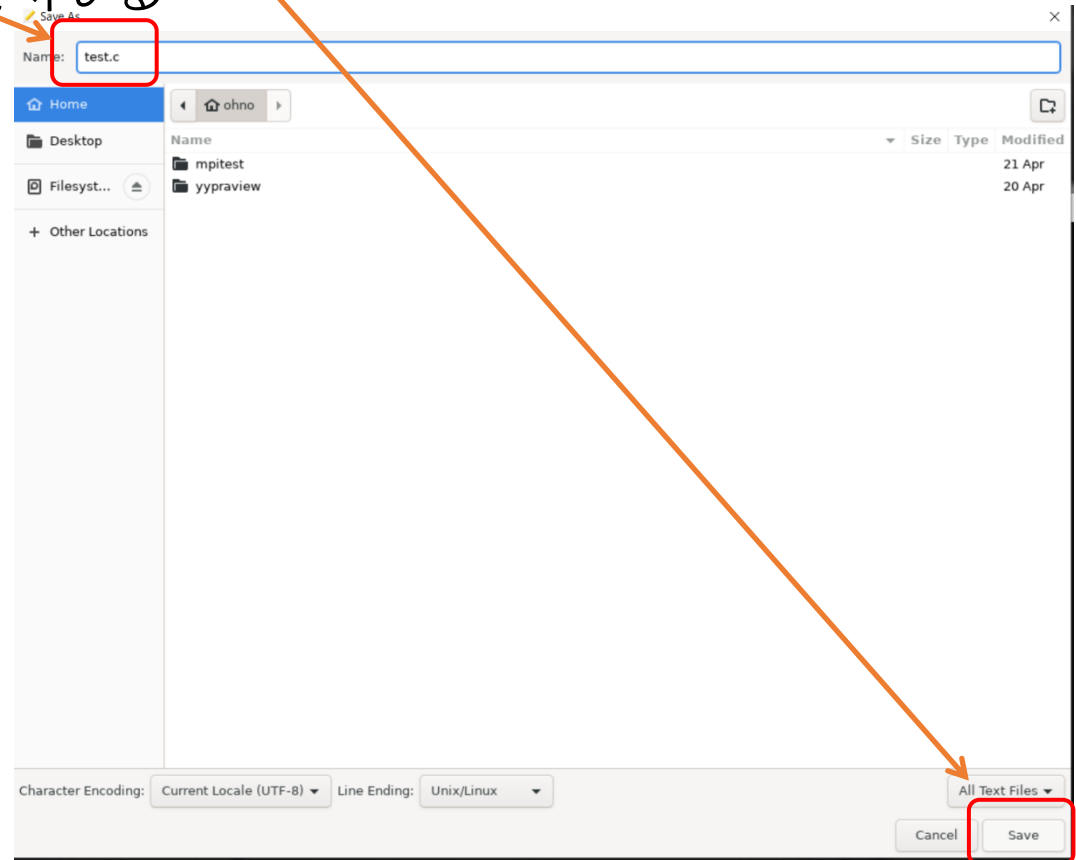
```
1 #include<stdio.h>
2
3 int main(int argc, char **argv){
4     printf("Hello, World!\n");
5     return 0;
6 }
7 |
```

The 'Save' button in the top right corner of the window is circled in blue. A red arrow points from the 'Save' button in the list to the 'Save' button in the screenshot. A red rounded rectangle highlights the code area of the editor.

Plain Text ▾ Tab Width: 8 ▾ Ln 7, Col 1 ▾ INS

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

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

int main(int argc, char **argv) {
    printf("Hello, World!¥n");
    return 0;
}
ohno@Akagi: ~$ gcc test.c
ohno@Akagi: ~$ ls
a.out  test.c
ohno@Akagi: ~$ ./a.out
Hello, World!
ohno@Akagi: ~$
```

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

```
ohno@Akagi: ~
ohno@Akagi:~$ cat test.f90
program helloworld
  print *, "Hello, World!(F90)"
end program helloworld
ohno@Akagi:~$ gfortran test.f90
ohno@Akagi:~$ ./a.out
Hello, World!(F90)
ohno@Akagi:~$
```

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;
}
```

```
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

omptest.c

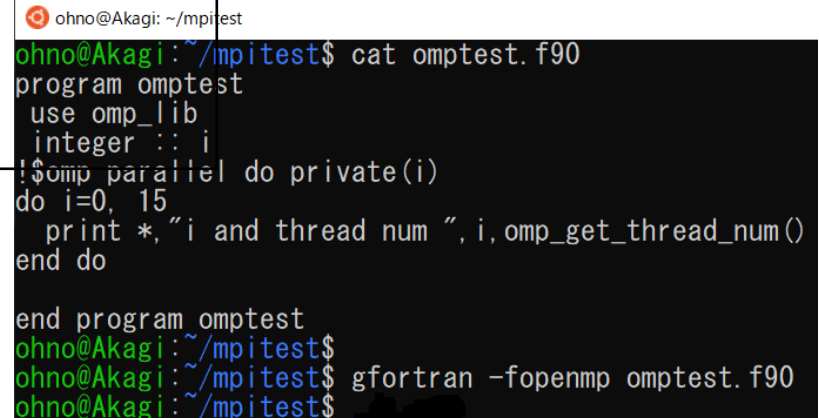
OpenMPの動作テスト-2

- 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 ompctest
  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 ompctest
```

ompctest.f90



```
ohno@Akagi: ~/mpitest
ohno@Akagi:~/mpitest$ cat ompctest.f90
program ompctest
  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 ompctest
ohno@Akagi:~/mpitest$
ohno@Akagi:~/mpitest$ gfortran -fopenmp ompctest.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 ompctest.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 ompctest.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;
}
```

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
```

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;
}
```

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 $2 \times 3 = 6$ 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
```

.bashrcの編集(WSL1)

- 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 "${[ $? = 0 ]} && echo terminal || echo error" "$(history | tail -n1 | sed -e '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  
  
export DISPLAY=Akagi:0.0  
alias mpirun='mpirun -oversubscribe --mca btl_vader_single_copy_mechanism none'  
".bashrc" 121L, 3878C written
```

DISPLAYやmpirun
のあたりのこと
を.bashrcに書いて
しまう

.bashrcの編集(WSL2)

- 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 "${[ $? = 0 ]} && echo terminal || echo error  
s/^%s*[0-9]+%s*//;s/[:&]%%s*alert$//'\''''"  
  
# 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=$(cat /etc/resolv.conf | grep nameserver | awk '{print $2}'):0.0  
alias mpirun='mpirun -oversubscribe --mca btl_vader_single_copy_mechanism none'  
".bashrc" 121L, 3933C
```

DISPLAYやmpirun
のあたりのこと
を.bashrcに書いて
しまう

.bashrcの編集(共通)

- .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側とのファイルのやりとり

- 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。