# CSC4005
# Parallel Programming
# Tutorial 1

Bokai Xu, 119010355@link.cuhk.edu.cn

# Tutorial Schedule

- **Tuesday** 18:00-18:50 TA307
- **Wednesday** 20:00-20:50 TA307

- Other sessions are cancelled.

# Office Hour Schedule

| | **Time** | **Venue** |
|---|---|---|
| Weibin Chen (TA) | 9:00-10:00 Wednesday | SDS office on the floor 4 of ZhiXin building |
| Yihan Liu (TA) | 19:00-20:00 Thursday | Online, zoom id: 613-456-4907 |
| Yuxuan Liu (TA) | 20:00-21:00 Thursday | Online, zoom id: 394-118-1035 |
| Bokai Xu and Zhixin Luoyang (USTF) | 21:00-22:00 Wednesday | TA307 |

# Outline of Tutorial 1

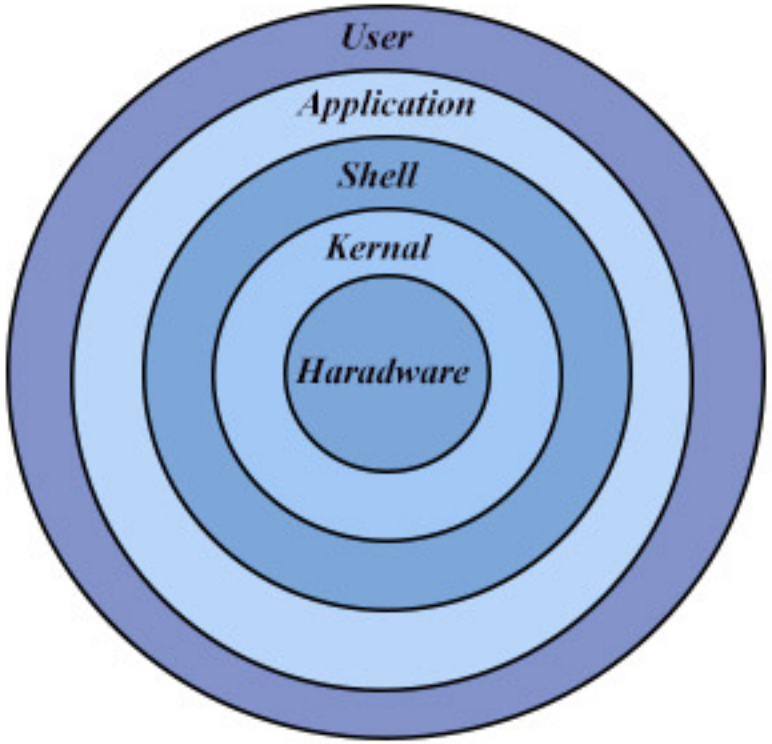
- About Linux
- Basic Linux Commands
- Development environment
- CSC4005 Virtual Machine setup
- C++ environment for parallel programs
- Compiler for parallel programs
- Run C++ parallel programs
- HPC (High Performance Computing) cluster
- IDE

# Why linux

- High performance computing clusters serve multiple users simultaneously,  linux works well.
- Linux are very friendly to cpp programming, a lot of cpp libraries provided on linux.
- Linux has a small size and high efficiency.
- ...

20% of interviewees told us that they don’t know how to use linux. So we add a linux part.

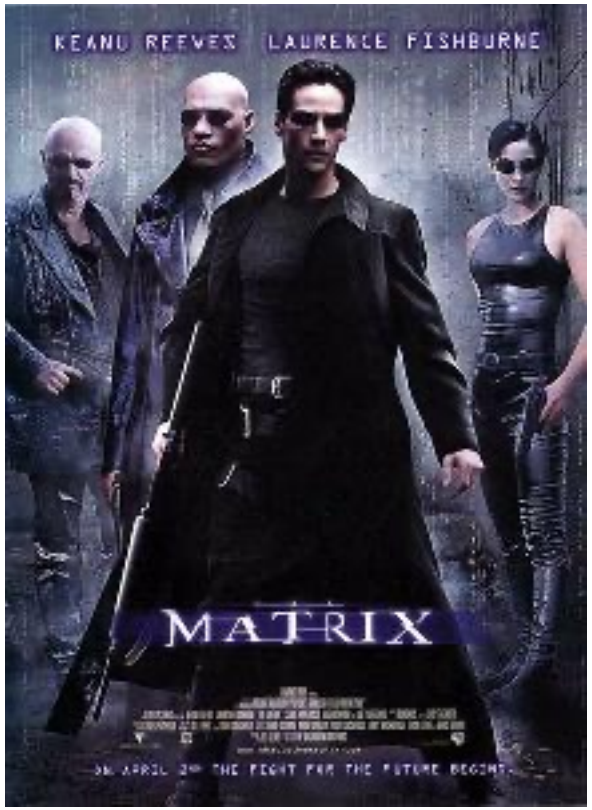
# How to use linux?

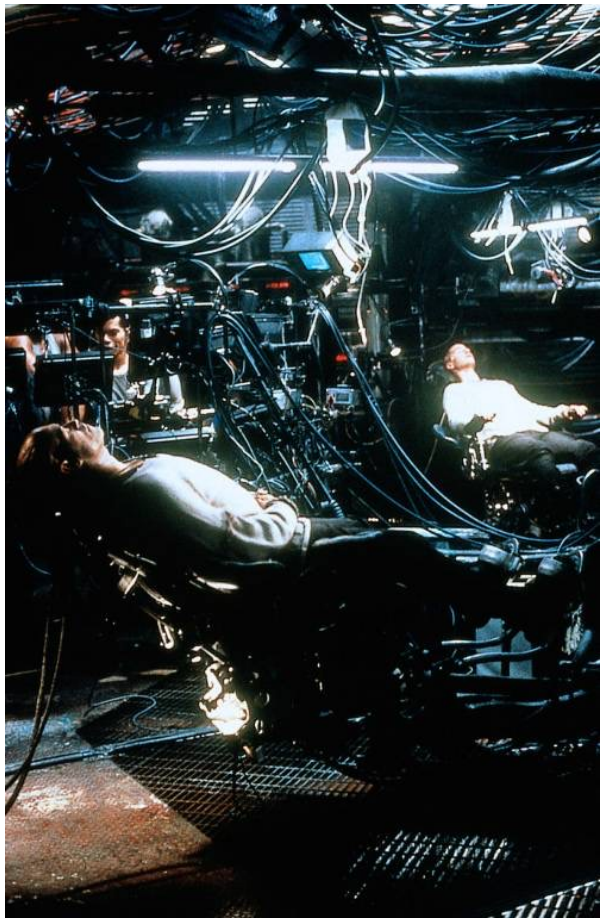



- What is your purpose to use Linux?
  - We want to utilize the capability of hardwares (like CPU, GPU, hard disk, network ...)
- How to use?
  - Hardware is wrapped by linux kernel, and kernel is wrapped by shell (terminal). Terminal is an interface layer.
  - We can only use terminal.

*Cite: http://generalnote.com/Linux-Tutorial/Introduction-to-Linux/basic-linux.php*

# The Matrix

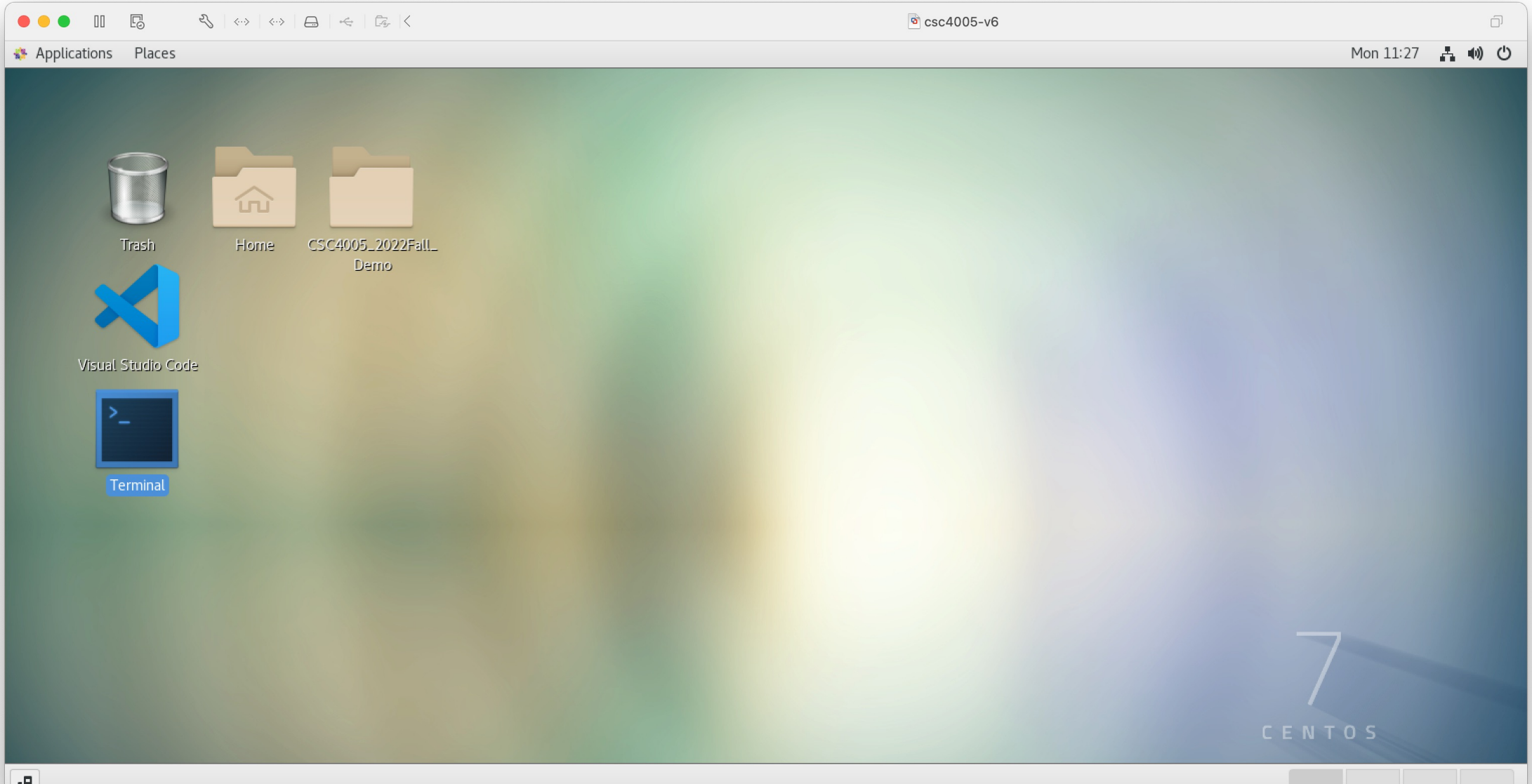


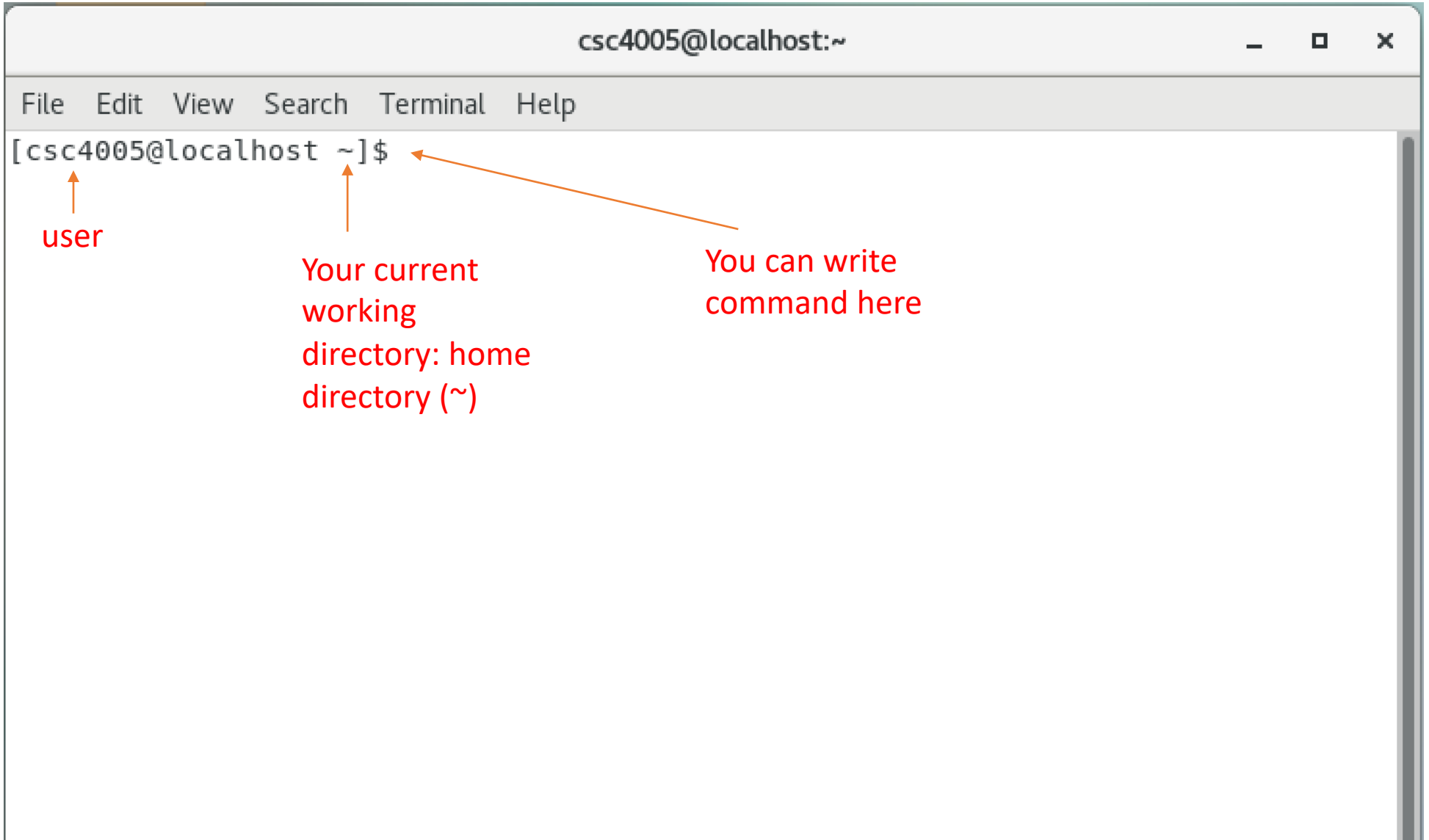
- The Matrix is a high performance computer, people who live in reality can only interact with the Matrix by using a terminal.

- GUI Desktop is indeed a program running on Linux. It is a enchanced version of terminal. But it is essentially a terminal.

# A classical terminal

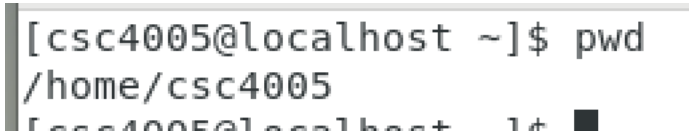


# A classical terminal



# pwd

- Use the **pwd** command to find out the path of the current working directory (folder) you're in. The command will return an absolute (full) path, which is basically a path of all the directories that starts with a forward slash **(/)**. An example of an absolute path is **/home/username**.



# cd

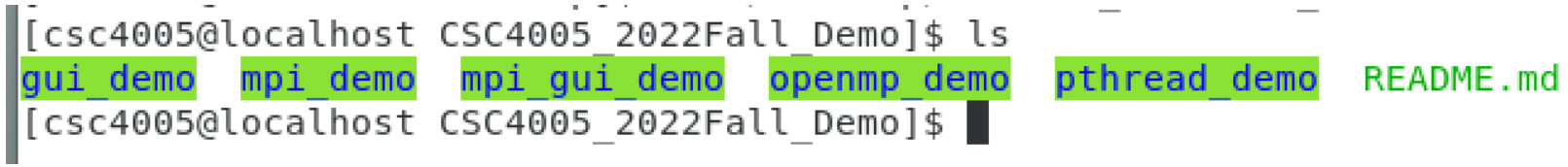
- To navigate through the Linux files and directories, use the **cd** command. It requires either the full path or the name of the directory, depending on the current working directory that you're in.

```
[csc4005@localhost ~]$ cd ~/Desktop/CSC4005_2022Fall_Demo
[csc4005@localhost CSC4005_2022Fall_Demo]$
```

```
[csc4005@localhost CSC4005_2022Fall_Demo]$ cd ..
[csc4005@localhost Desktop]$
```

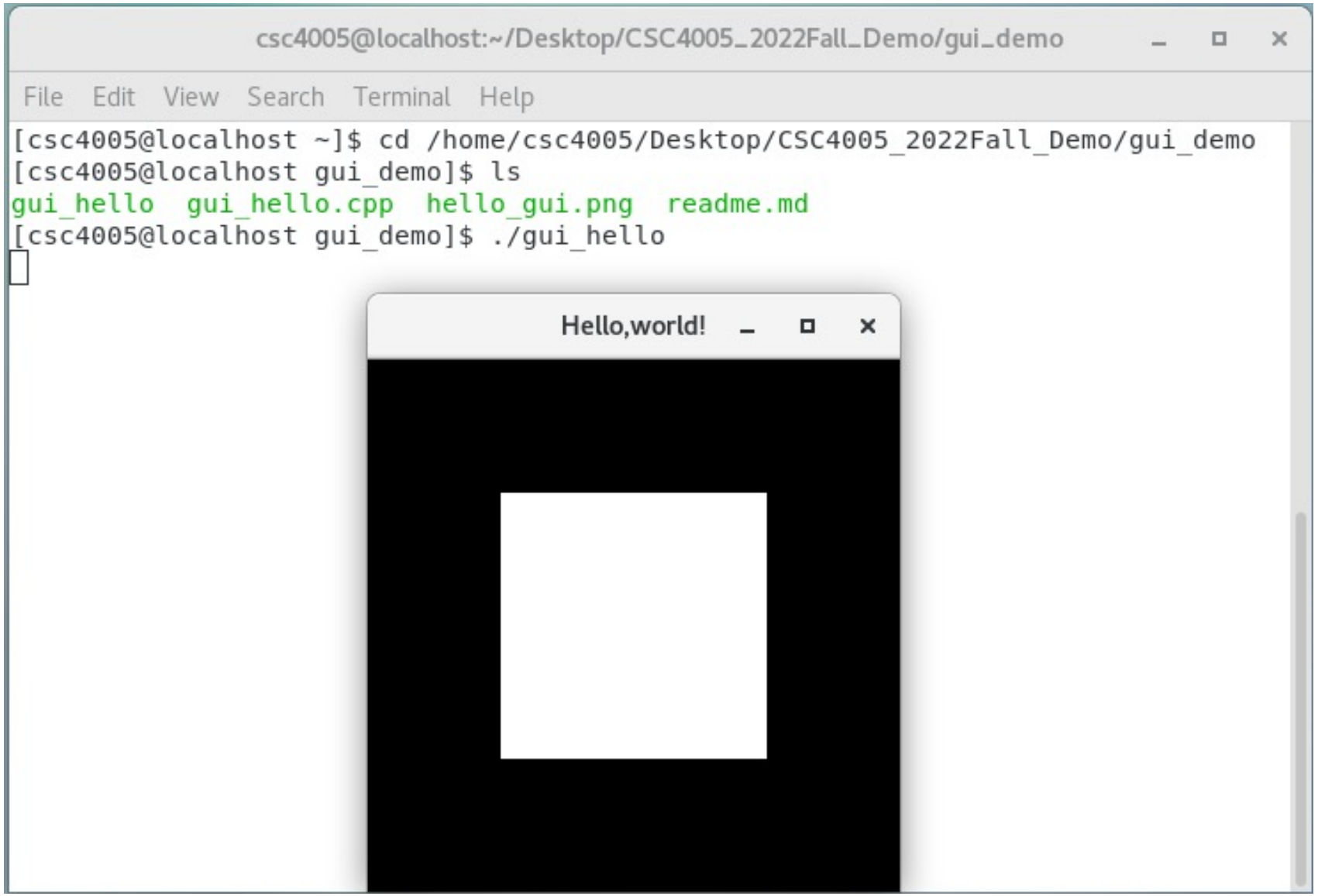
# ls

- The **ls** command is used to view the contents of a directory. By default, this command will display the contents of your current working directory.



# Run an executable

- Go to a proper directory
- Make sure you refer to the right executable
- Some program are global (you have to add it to PATH env variable)



# Interrupt a program

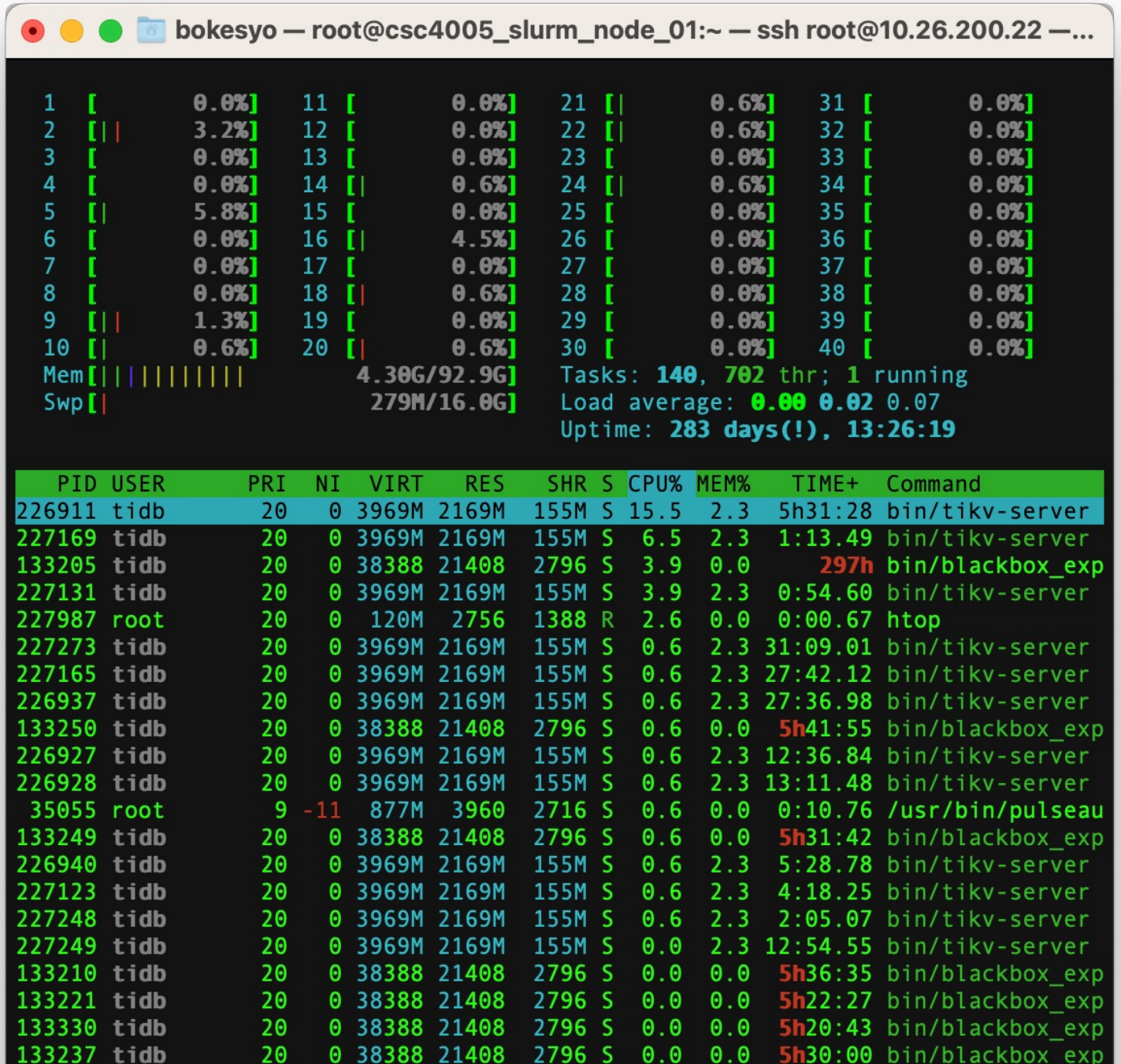
- so it will kill itself:
- Ctrl+C

```
[csc4005@localhost ~]$ cd /home/csc4005/Desktop/CSC
[csc4005@localhost gui_demo]$ ls
gui_hello  gui_hello.cpp  hello_gui.png  readme.md
[csc4005@localhost gui_demo]$ ./gui_hello
^C
[csc4005@localhost gui_demo]$
```

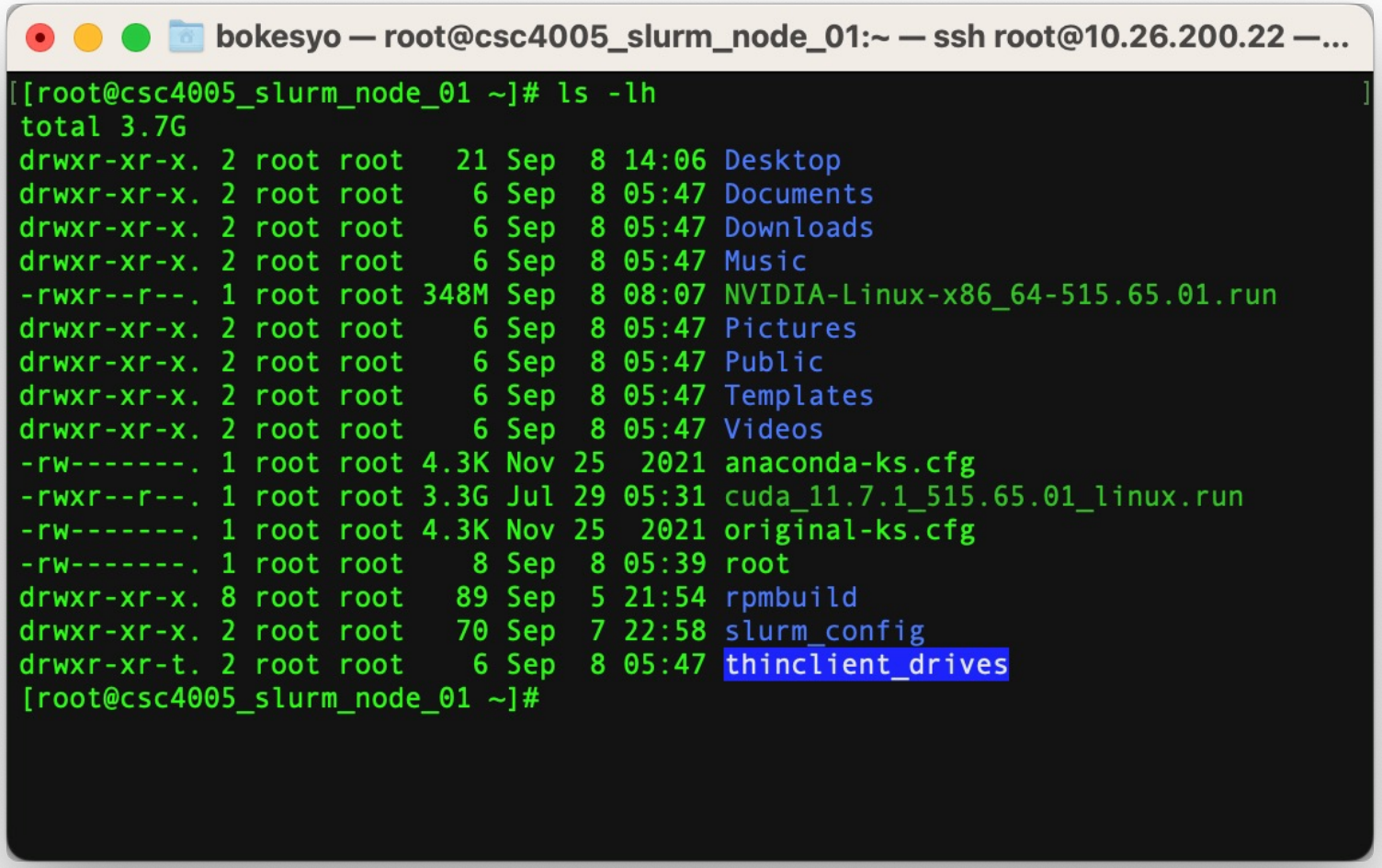
# Kill a process

- pkill [process_name]
- Kill -9 [pid]

# htop

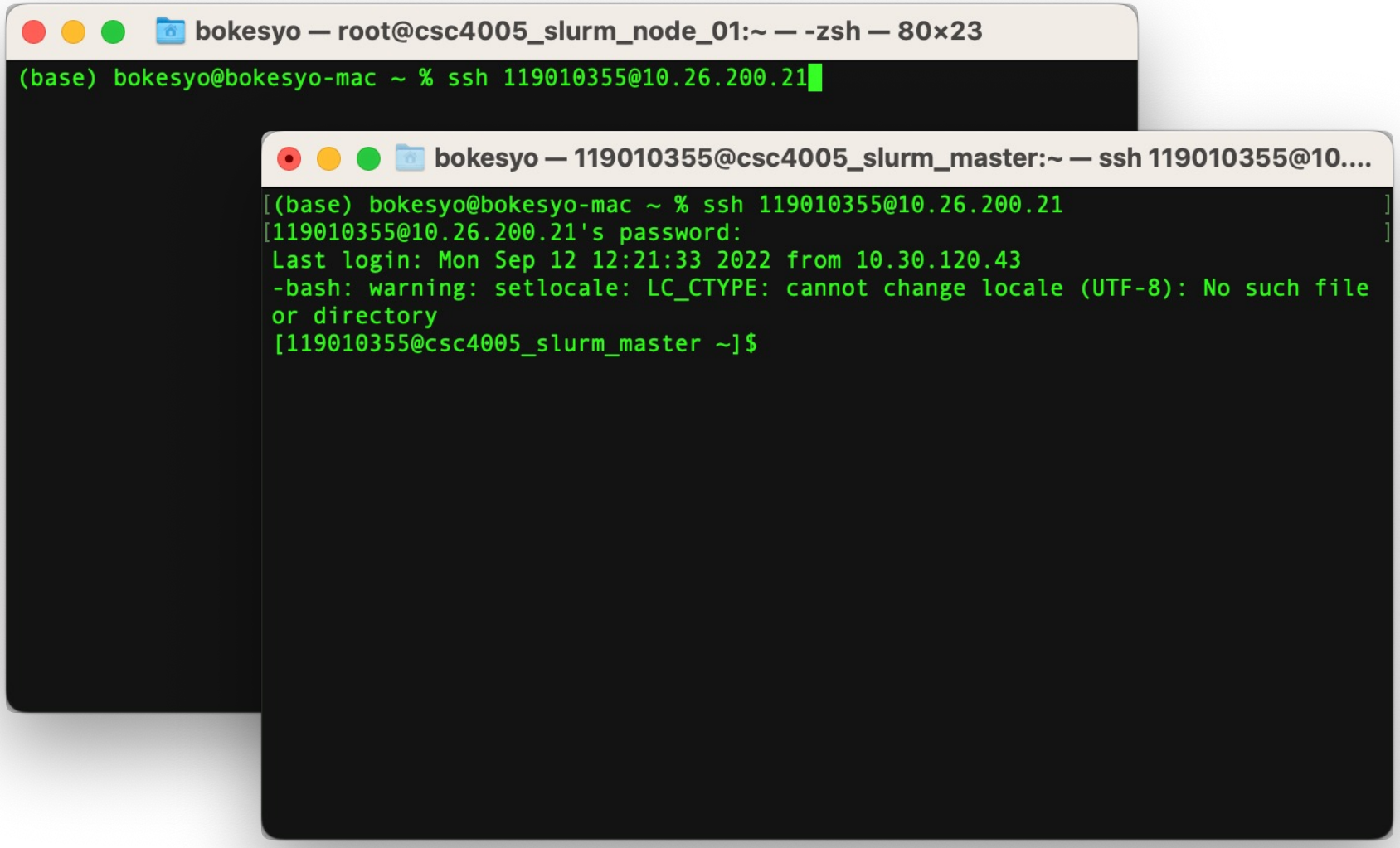

bokesyo — root@csc4005_slurm_node_01:~ — ssh root@10.26.200.22 —...
1 [ 0.0%] 11 [ 0.0%] 21 [| 0.6%] 31 [ 0.0%]
2 [|| 3.2%] 12 [ 0.0%] 22 [| 0.6%] 32 [ 0.0%]
3 [ 0.0%] 13 [ 0.0%] 23 [ 0.0%] 33 [ 0.0%]
4 [ 0.0%] 14 [| 0.6%] 24 [| 0.6%] 34 [ 0.0%]
5 [| 5.8%] 15 [ 0.0%] 25 [ 0.0%] 35 [ 0.0%]
6 [ 0.0%] 16 [| 4.5%] 26 [ 0.0%] 36 [ 0.0%]
7 [ 0.0%] 17 [ 0.0%] 27 [ 0.0%] 37 [ 0.0%]
8 [ 0.0%] 18 [| 0.6%] 28 [ 0.0%] 38 [ 0.0%]
9 [|| 1.3%] 19 [ 0.0%] 29 [ 0.0%] 39 [ 0.0%]
10 [| 0.6%] 20 [| 0.6%] 30 [ 0.0%] 40 [ 0.0%]
Mem[|||||||||| 4.30G/92.9G] Tasks: 140, 702 thr; 1 running
Swp[| 279M/16.0G] Load average: 0.00 0.02 0.07
Uptime: 283 days(!), 13:26:19
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
226911 tidb 20 0 3969M 2169M 155M S 15.5 2.3 5h31:28 bin/tikv-server
227169 tidb 20 0 3969M 2169M 155M S 6.5 2.3 1:13.49 bin/tikv-server
133205 tidb 20 0 38388 21408 2796 S 3.9 0.0 297h bin/blackbox_exp
227131 tidb 20 0 3969M 2169M 155M S 3.9 2.3 0:54.60 bin/tikv-server
227987 root 20 0 120M 2756 1388 R 2.6 0.0 0:00.67 htop
227273 tidb 20 0 3969M 2169M 155M S 0.6 2.3 31:09.01 bin/tikv-server
227165 tidb 20 0 3969M 2169M 155M S 0.6 2.3 27:42.12 bin/tikv-server
226937 tidb 20 0 3969M 2169M 155M S 0.6 2.3 27:36.98 bin/tikv-server
133250 tidb 20 0 38388 21408 2796 S 0.6 0.0 5h41:55 bin/blackbox_exp
226927 tidb 20 0 3969M 2169M 155M S 0.6 2.3 12:36.84 bin/tikv-server
226928 tidb 20 0 3969M 2169M 155M S 0.6 2.3 13:11.48 bin/tikv-server
35055 root 9 -11 877M 3960 2716 S 0.6 0.0 0:10.76 /usr/bin/pulseau
133249 tidb 20 0 38388 21408 2796 S 0.6 0.0 5h31:42 bin/blackbox_exp
226940 tidb 20 0 3969M 2169M 155M S 0.6 2.3 5:28.78 bin/tikv-server
227123 tidb 20 0 3969M 2169M 155M S 0.6 2.3 4:18.25 bin/tikv-server
227248 tidb 20 0 3969M 2169M 155M S 0.6 2.3 2:05.07 bin/tikv-server
227249 tidb 20 0 3969M 2169M 155M S 0.0 2.3 12:54.55 bin/tikv-server
133210 tidb 20 0 38388 21408 2796 S 0.0 0.0 5h36:35 bin/blackbox_exp
133221 tidb 20 0 38388 21408 2796 S 0.0 0.0 5h22:27 bin/blackbox_exp
133330 tidb 20 0 38388 21408 2796 S 0.0 0.0 5h20:43 bin/blackbox_exp
133237 tidb 20 0 38388 21408 2796 S 0.0 0.0 5h30:00 bin/blackbox_exp

# ls -lh


bokesyo — root@csc4005_slurm_node_01:~ — ssh root@10.26.200.22 —...
[[root@csc4005_slurm_node_01 ~]# ls -lh
total 3.7G
drwxr-xr-x. 2 root root   21 Sep  8 14:06 Desktop
drwxr-xr-x. 2 root root    6 Sep  8 05:47 Documents
drwxr-xr-x. 2 root root    6 Sep  8 05:47 Downloads
drwxr-xr-x. 2 root root    6 Sep  8 05:47 Music
-rwxr--r--. 1 root root 348M Sep  8 08:07 NVIDIA-Linux-x86_64-515.65.01.run
drwxr-xr-x. 2 root root    6 Sep  8 05:47 Pictures
drwxr-xr-x. 2 root root    6 Sep  8 05:47 Public
drwxr-xr-x. 2 root root    6 Sep  8 05:47 Templates
drwxr-xr-x. 2 root root    6 Sep  8 05:47 Videos
-rw-------. 1 root root 4.3K Nov 25  2021 anaconda-ks.cfg
-rwxr--r--. 1 root root 3.3G Jul 29 05:31 cuda_11.7.1_515.65.01_linux.run
-rw-------. 1 root root 4.3K Nov 25  2021 original-ks.cfg
-rw-------. 1 root root    8 Sep  8 05:39 root
drwxr-xr-x. 8 root root   89 Sep  5 21:54 rpmbuild
drwxr-xr-x. 2 root root   70 Sep  7 22:58 slurm_config
drwxr-xr-t. 2 root root    6 Sep  8 05:47 thinclient_drives
[root@csc4005_slurm_node_01 ~]#

# ssh

- ssh [user]@[IP]



# scp

- Copy files from one machine to another.
- scp [user]@[IP of remote machine]:[source file address on remote machine] [target file address on local machine]

- scp [target file address on local machine] [user]@[IP of remote machine]:[source file address on remote machine]

# Suggested Development Environment

| | **MPI, pthreads, openMP, GUI** | **CUDA** |
|---|---|---|
| If you are using Windows | Virtual Machine for coding and debug,<br>ssh+HPC cluster for large scale experiment | ssh+HPC cluster for everything |
| If you are using Mac (intel chip) | Virtual Machine for coding and debug,<br>ssh+HPC cluster for large scale experiment | ssh+HPC cluster for everything |
| If you are using Mac (Apple Silicon) | ssh+HPC cluster for everything | ssh+HPC cluster for everything |

# CSC4005 Virtual Machine Setup

- You need a virtual machine software
- For windows users:
  - You can use either VirtualBox (free) or VMware Workstation (paid)
- For Mac (intel) users:
  - You can try VMware Fusion Player (free) or VirtualBox (free)
- For Mac (Apple silicon) users:
  - You are suggested to use HPC cluster for everything (we will provide special guidance for you)

# C++ libraries on linux

- Compiling parallel programs requires external libraries (previously we only use C++ standard library in CSC3002)
- Install external libraries
  - sudo yum install xxxx
- External libraries are stored in
  - **Include directory** (.h):
  - /usr/include
  - **Library directory** (.so binary compiled code):
  - /usr/lib
  - /usr/lib64

mpi_demo > mpi_hello.cpp > ...

```cpp
#include <mpi.h>
#include <stdio.h>
#include <math.h>

int main(int argc,char** argv)
{
    int myid,numproces;
    int namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    MPI_Comm_size(MPI_COMM_WORLD,&numproces);
    MPI_Get_processor_name(processor_name,&namelen);
    fprintf(stdout,"hello world! Process %d of %d on %s\
        myid,numproces,processor_name);
    MPI_Finalize();

    return 0;
}

```

# C++ compilers on linux

- Suggested: g++ (for both C and C++)
- **Function1: Compilation**
  - Make your code binary code for CPU to execute.
- **Function2: Linking**
  - Link your compiled program with external libraries (*.so) to make the program executable. Otherwise some functions will be undefined. Error will occurs.

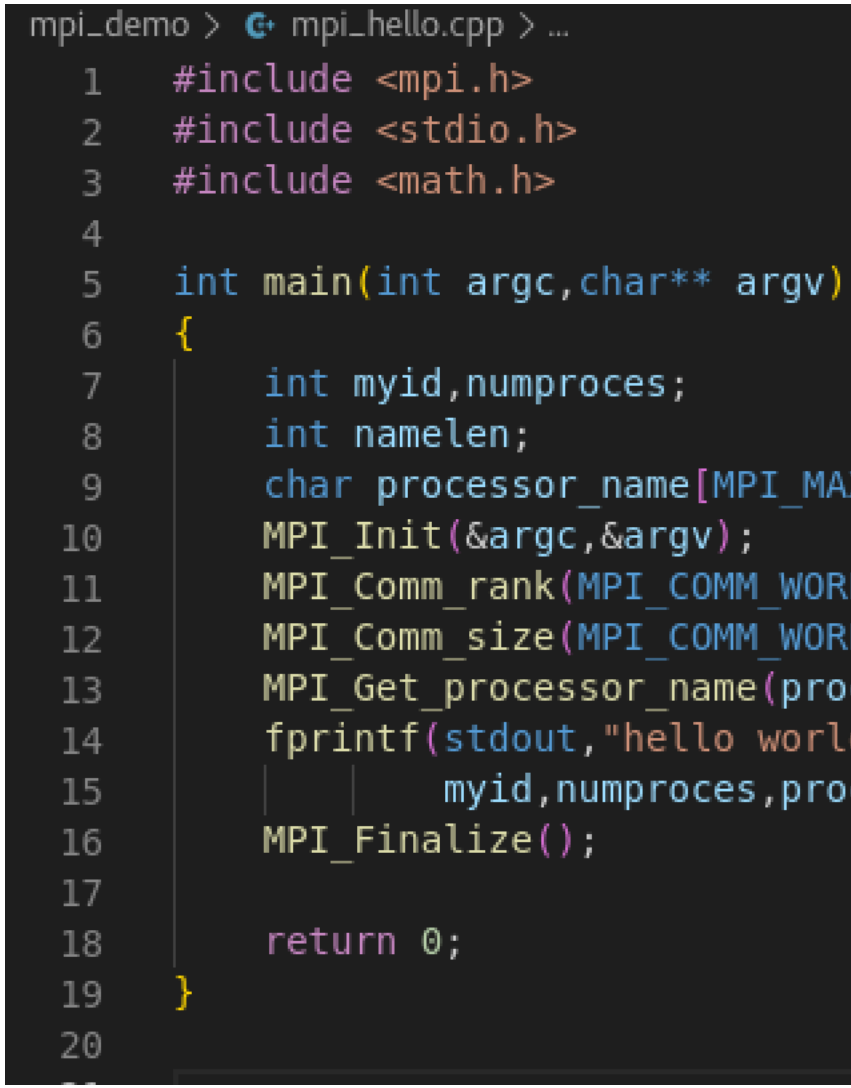
# C++ compilers for CSC4005

We will have four programming projects in CSC4005, all frameworks needed are listed here:

| | **compiler** | |
|---|---|---|
| MPI | mpic++ | It is a wrapped g++ command |
| OpenMP, pthread | g++ | |
| GUI | g++ | |
| CUDA | nvcc | |
| Hybrid Compilation (CUDA & C++) | nvcc & g++ -> g++ linker | |

# C++ libraries on linux

- **Using external libraries**
- When you are writing your MPI parallel programs, you have to include <mpi.h>
- When you compile your mpi code, you should use mpic++ as compiler, it will help you find mpi.h and link mpi runtime library to your program.
- You can also use
- **g++** -m64 -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong --param=ssp-buffer-size=4 -grecord-gcc-switches -m64 -mtune=generic -fPIC -Wl,-z,noexecstack **-I/usr/include/mpich-3.2-x86_64 -L/usr/lib64/mpich-3.2/lib** -lmpicxx -Wl,-rpath -Wl,/usr/lib64/mpich-3.2/lib -Wl,--enable-new-dtags -**lmpi**



# C++ libraries on linux

- g++ **-I/usr/include -L/usr/local/lib -L/usr/lib -lglut -lGLU -lGL** -lm gui_hello.cpp -o gui_hello

gui_hello.cpp

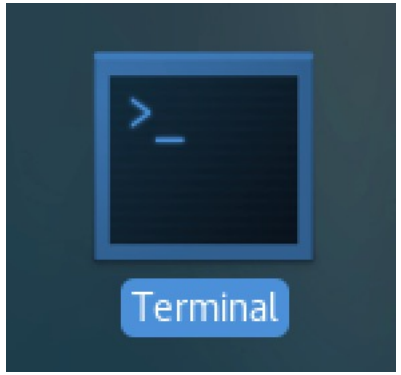
gui_demo > gui_hello.cpp > ...

```
#include <GL/glut.h>

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_POLYGON);
     glVertex2f(-0.5,-0.5);
     glVertex2f(-0.5,0.5);
     glVertex2f(0.5,0.5);
     glVertex2f(0.5,-0.5);
    glEnd();

    glFlush();
}

int main(int argc,char **argv)
{
    glutInit(&argc,argv);
    glutCreateWindow("Hello,world!");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

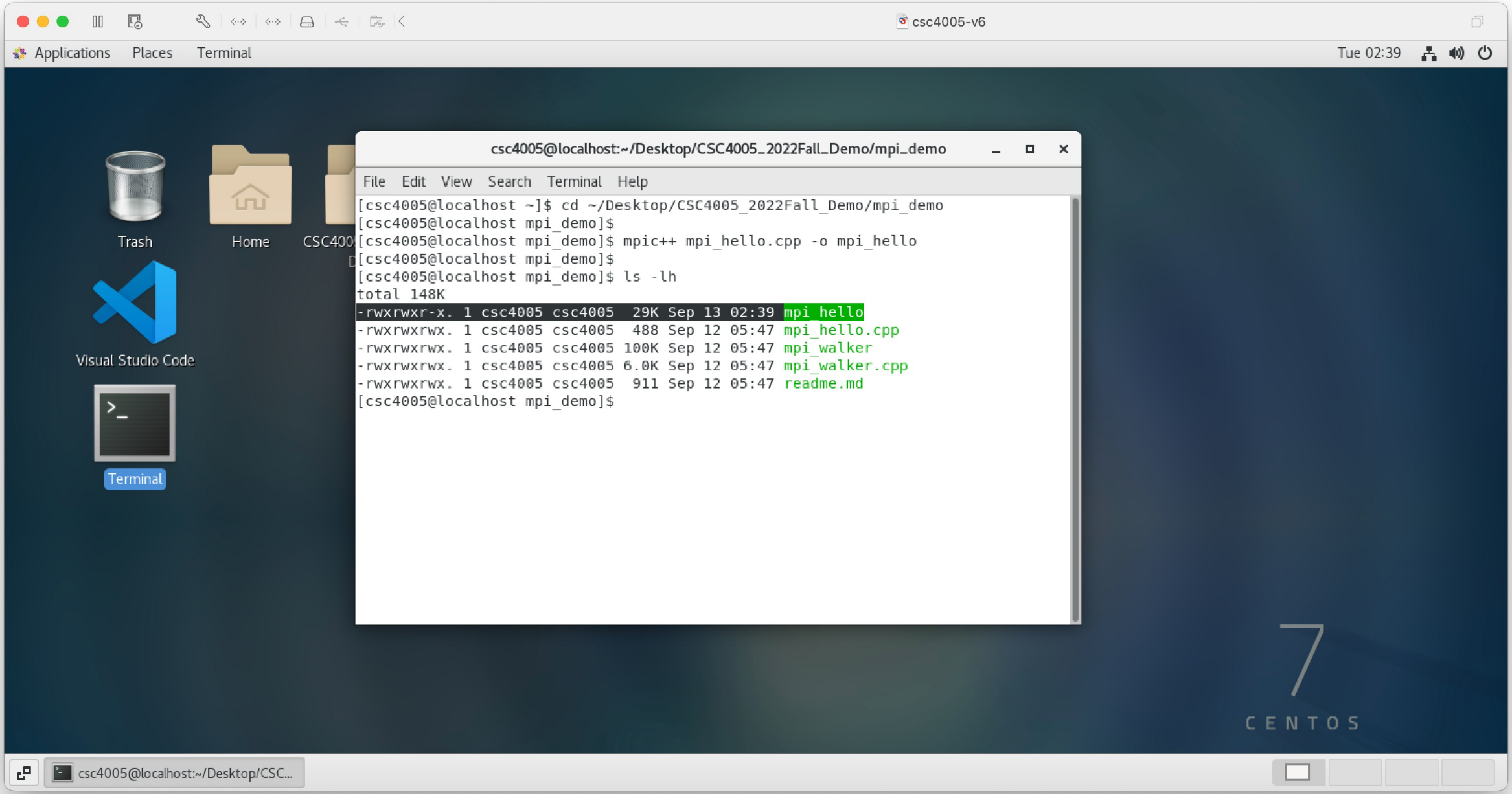
# C++ parallel program compilation

You need to use CSC4005 VM or HPC cluster to compile.

- If you have finsihed setting up your VM, open Terminal on your VM.



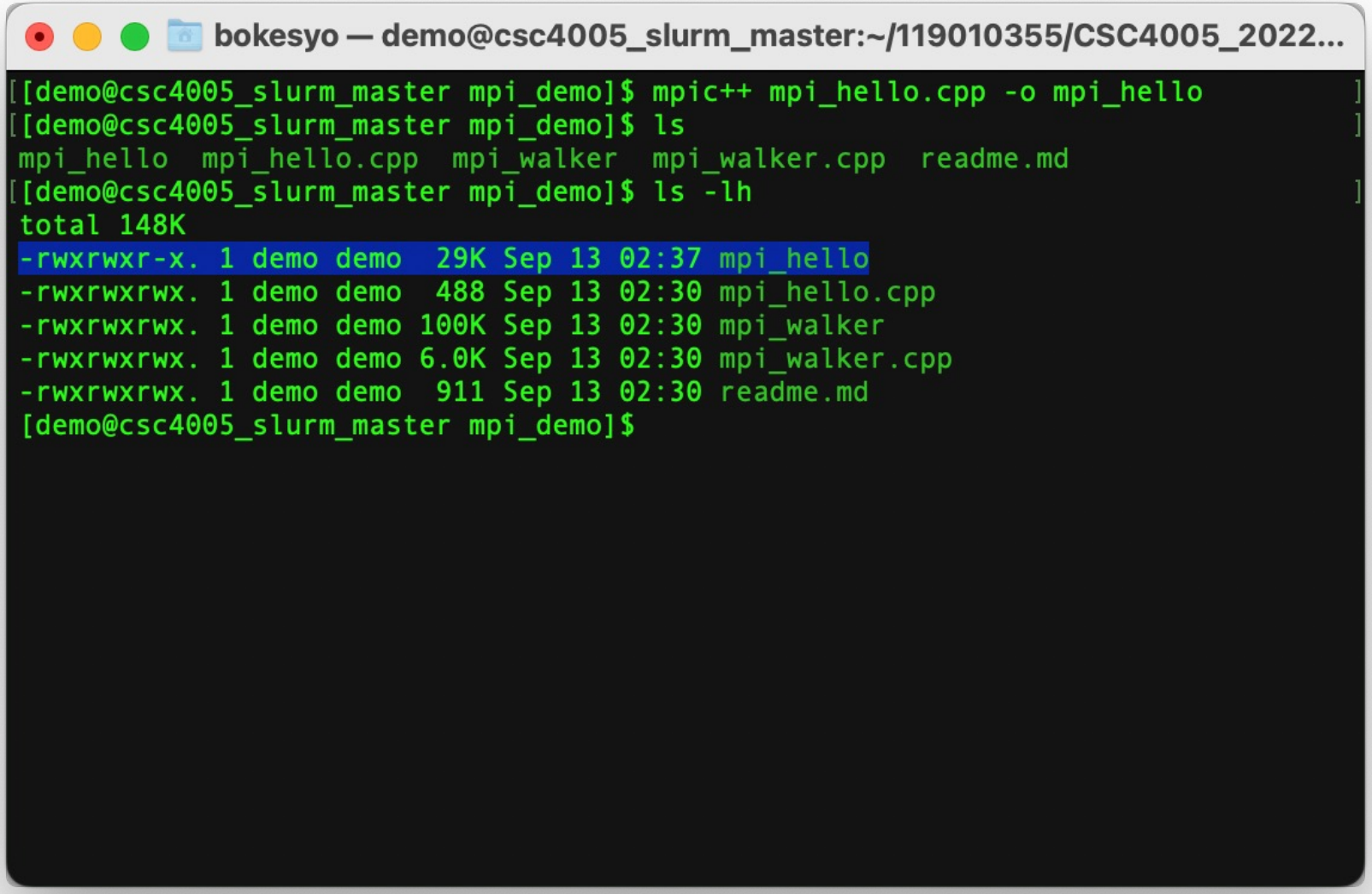

```
cd ~/Desktop/CSC4005_2022Fall_Demo/mpi_demo # switch to code folder
mpic++ mpi_hello.cpp -o mpi_hello # compile command
```


csc4005-v6
Applications
Places
Terminal
Tue 02:39
Trash
Home
CSC400
Visual Studio Code
Terminal
csc4005@localhost:~/Desktop/CSC4005_2022Fall_Demo/mpi_demo
File Edit View Search Terminal Help
[csc4005@localhost ~]$ cd ~/Desktop/CSC4005_2022Fall_Demo/mpi_demo
[csc4005@localhost mpi_demo]$
[csc4005@localhost mpi_demo]$ mpic++ mpi_hello.cpp -o mpi_hello
[csc4005@localhost mpi_demo]$
[csc4005@localhost mpi_demo]$ ls -lh
total 148K
-rwxrwxr-x. 1 csc4005 csc4005  29K Sep 13 02:39 mpi_hello
-rwxrwxrwx. 1 csc4005 csc4005  488 Sep 12 05:47 mpi_hello.cpp
-rwxrwxrwx. 1 csc4005 csc4005 100K Sep 12 05:47 mpi_walker
-rwxrwxrwx. 1 csc4005 csc4005 6.0K Sep 12 05:47 mpi_walker.cpp
-rwxrwxrwx. 1 csc4005 csc4005  911 Sep 12 05:47 readme.md
[csc4005@localhost mpi_demo]$
7
CENTOS
csc4005@localhost:~/Desktop/CSC...

# C++ parallel program compilation

- For those students unable to run a VM, ssh to HPC clusters.
  - **ssh 119010355@10.26.200.21** Password: csc4005
  - *This is only a test account,* ***we will distribute your own account next week.***

```
mkdir 119010355 # change to your student id
cd 119010355 # change to your student id
git clone https://github.com/bokesyo/CSC4005_2022Fall_Demo.git # download our demo script
chmod -R 777 CSC4005_2022Fall_Demo # make those files executable
cd CSC4005_2022Fall_Demo/mpi_demo # switch to code folder
mpic++ mpi_hello.cpp -o mpi_hello # compile command
```




Do not kill this terminal!

# C++ parallel program compilation

- If no error occurred at compilation step, let’s continue.

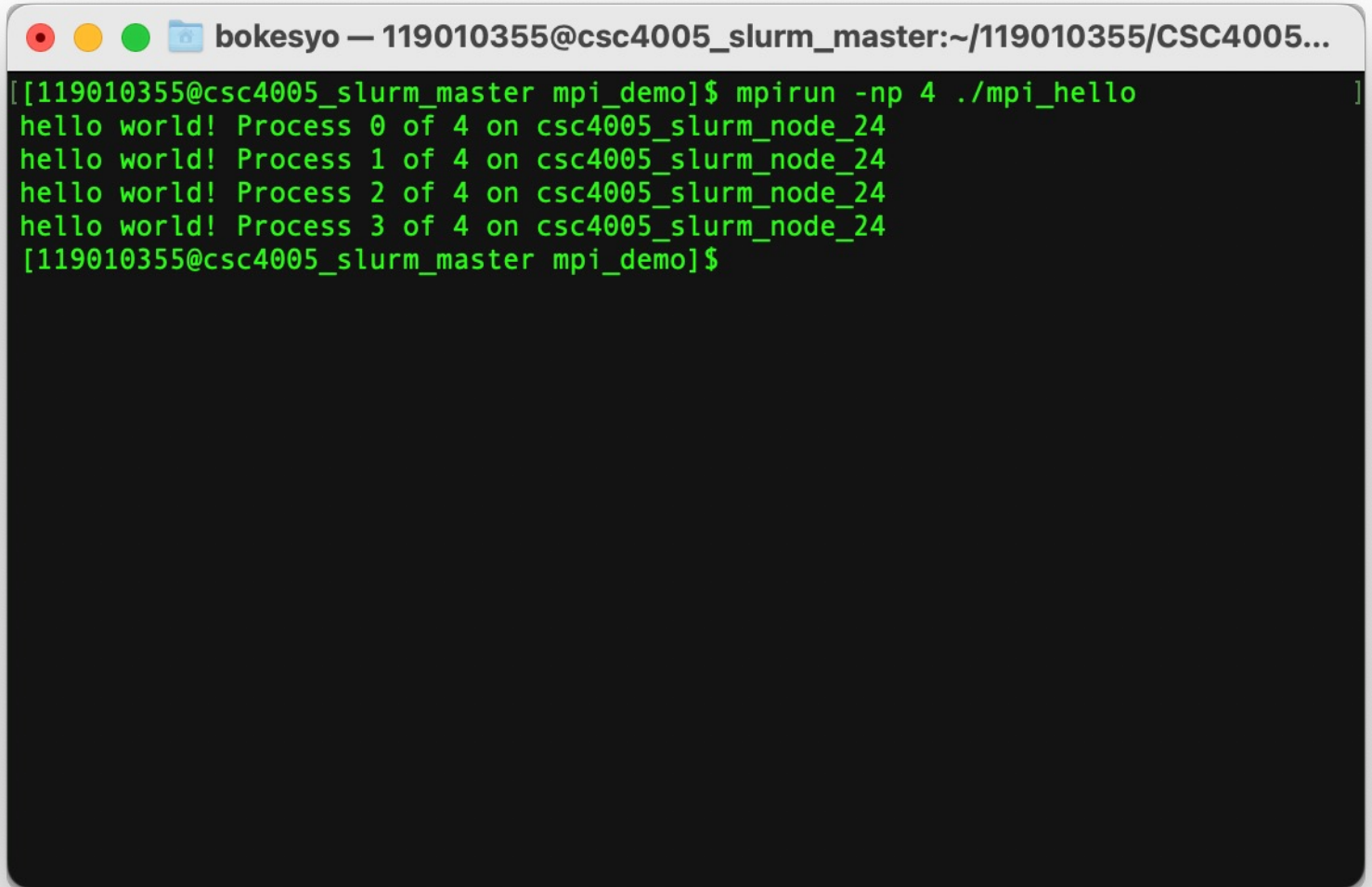
# Run your C++ parallel programming on VM or HPC cluster

- For those students who can use VM

```
mpirun -np 4 ./mpi_hello
```

- For those students unable to run a VM

```
salloc -n4 # request 4 cores on compute nodes to run your compiled program
mpirun -np 4 ./mpi_hello
```


bokesyo — 119010355@csc4005_slurm_master:~/119010355/CSC4005...
[[119010355@csc4005_slurm_master mpi_demo]$ mpirun -np 4 ./mpi_hello ]
hello world! Process 0 of 4 on csc4005_slurm_node_24
hello world! Process 1 of 4 on csc4005_slurm_node_24
hello world! Process 2 of 4 on csc4005_slurm_node_24
hello world! Process 3 of 4 on csc4005_slurm_node_24
[119010355@csc4005_slurm_master mpi_demo]$

# Run your C++ parallel programming on VM or HPC cluster

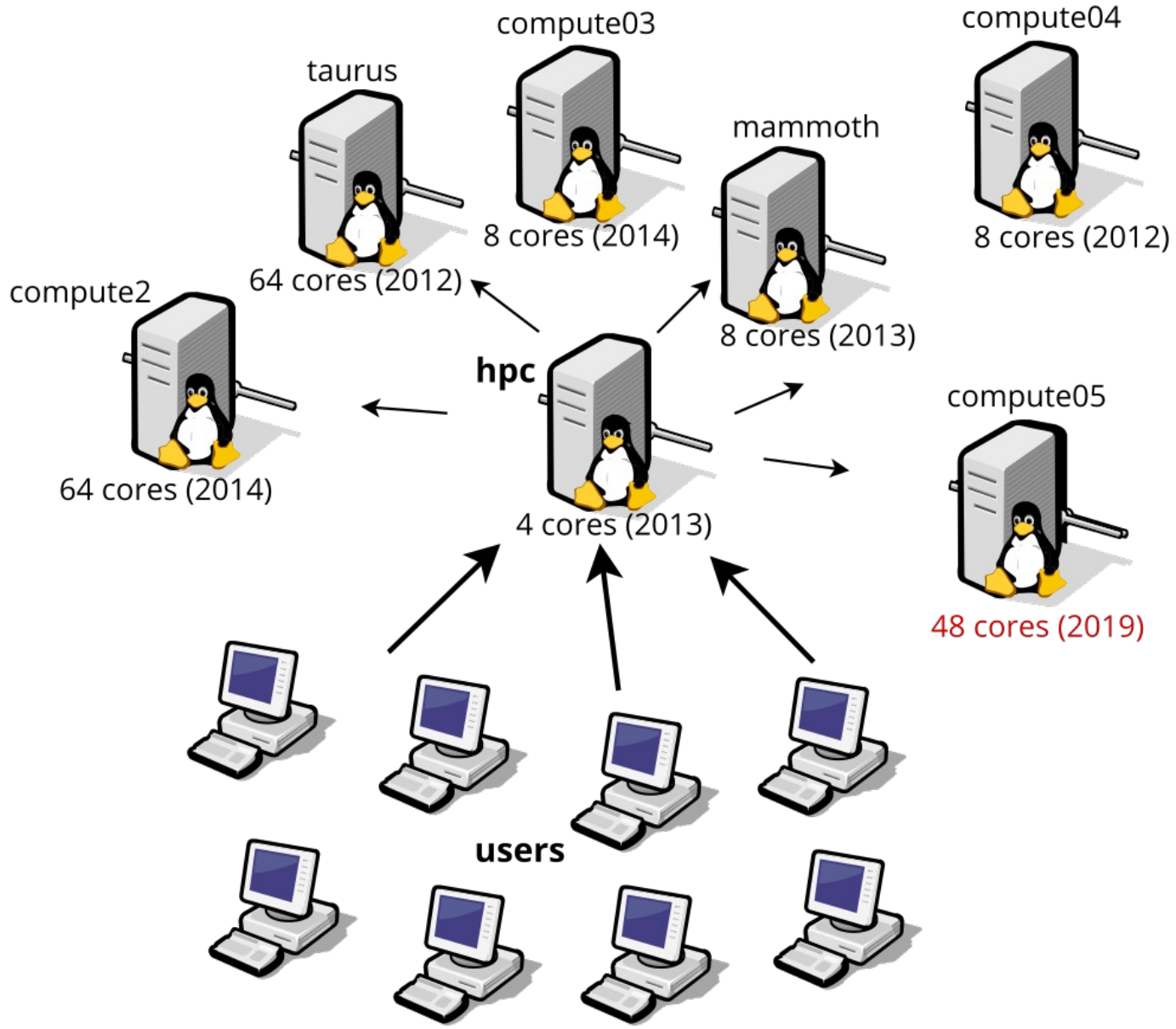
- If your output contains process 0,1,2,3, let’s continue.

# HPC cluster

- **Configuration**
- 1 login node
  - 20 Intel CPU cores (40 logic cores)
  - 100GB RAM
  - 1 Nvidia Quadro RTX 4000 GPU
- 30+ compute node
  - Each compute node has:
  - 20 Intel CPU cores (40 logic cores)
  - 100GB RAM
  - 1 Nvidia Quadro RTX 4000 GPU

# HPC cluster

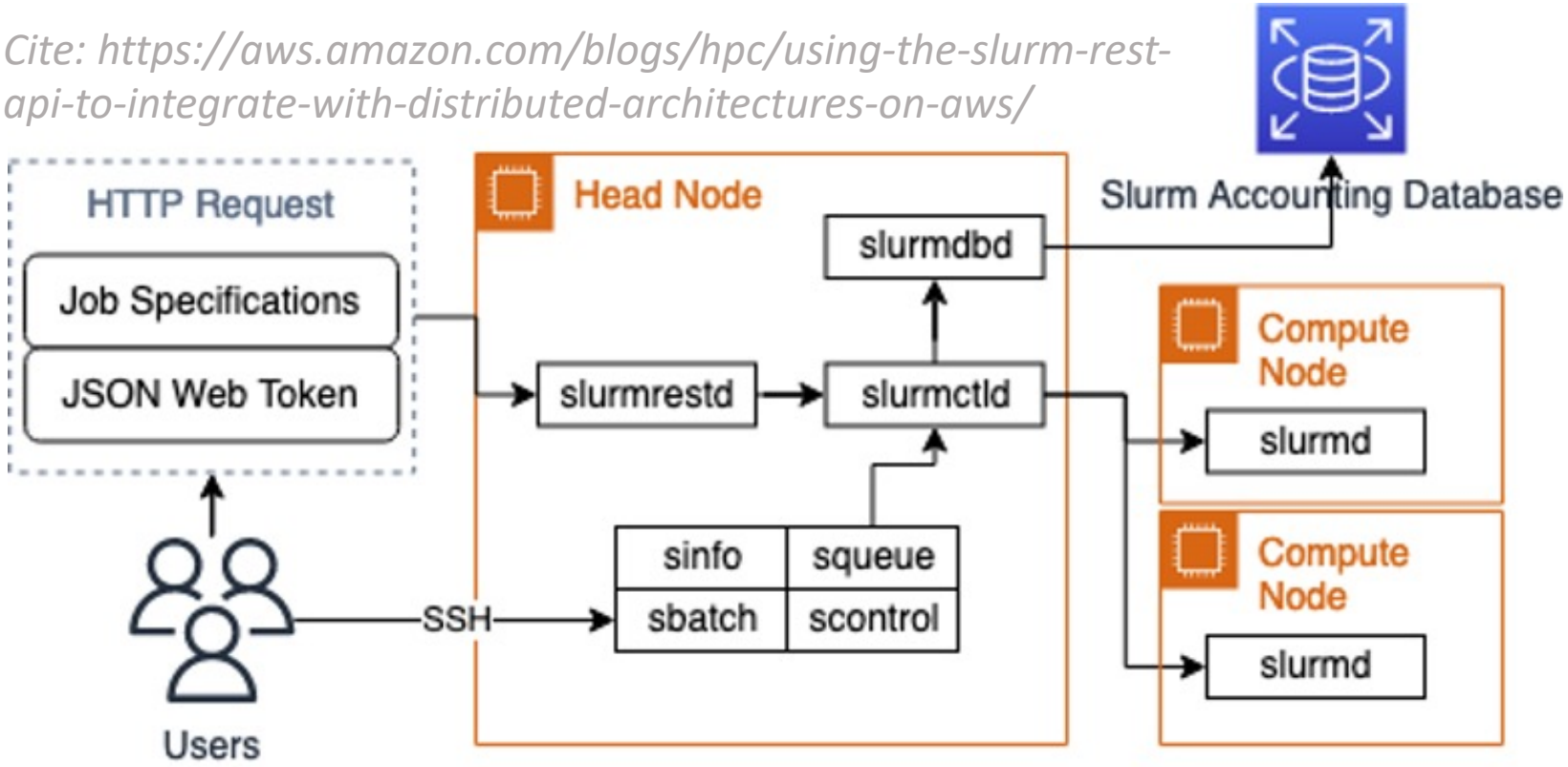
- **Topology**: one login node (controller) and 30+ compute node.




*Cite: https://hpc.ilri.cgiar.org/using-the-cluster*

# HPC cluster

*Cite: https://aws.amazon.com/blogs/hpc/using-the-slurm-rest-api-to-integrate-with-distributed-architectures-on-aws/*




- **Scheduler: slurm**
- **Basic command:**
- sbatch
- salloc
- more on Tutorial 2

- **Underlying process:**
- Slurm transfers your shell script to compute node (same environment has been configured on each compute node and file system is shared, so your shell script can run on compute node). The output will be stored in a log file. Slurm will reserve the resources you applied for your parallel programs.
- A parallel program can **create many processes** on the same machine (processes communicate within that machine) **or** on multiple machines (processes communicate by networks)

# HPC cluster

- **Login node is for you to:**
- Use file system (store your code and other files)
- Debug your code
- Compile your code
- **Apply for resources** and **create computing jobs** using slurm

- **Normal users cannot directly access compute node.**

# HPC cluster

- **Your personal directory on HPC cluster controller node**
- **/nfsmnt/[your student id]/**

- You can access your personal files, while others cannot.
- **/nfsmnt/ is shared among all nodes (both controller node and compute node) in cluster, when you submit your job, make sure that your program is inside /nfsmnt/[your student id]/. Then compute node can access your program.**

# About IDE

- 1) For Windows users, if you don’t want to waste time on setting up port forwarding/Xming X11 forwarding..., we suggest using CSC4005 Virtual Machine.
- 2) For Mac (Intel chip) users, if you don’t want to waste time on setting up port forwarding/XQuartz X11 forwarding..., we suggest using CSC4005 Virtual Machine.
- 3) For Mac (Apple Silicon) users, you can try if our VM can run on your PC, if not, please see special guidance for you on BB/Content/CSC4005 Development Platform.
- 4) For Linux Geeks, you can refer special guidance for you on BB/Content/CSC4005 Development Platform.
- 5) Other cases: You can DIY your development platform~