175th Parallel Programming Workshop with Trial Account
Supercomputing for Beginners

22nd Apr, 2022

This is for participants who use

OpenSSH or Cygwin on Windows, Mac OS, or Linux.

東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

20th Apr. 2022 v1.3

Agenda

13:00 – 14:00

Introduction, How to use Zoom for this tutorial

What are supercomputers ? (lecture)

Login tutorial (lecture + exercise)

14:15 – 15:45

Write, compile, and run programs (lecture + exercise)

Compile and run parallel programs (lecture + exercise)

16:00 – 17:00

How to run machine learning on supercomputers
(lecture + exercise)

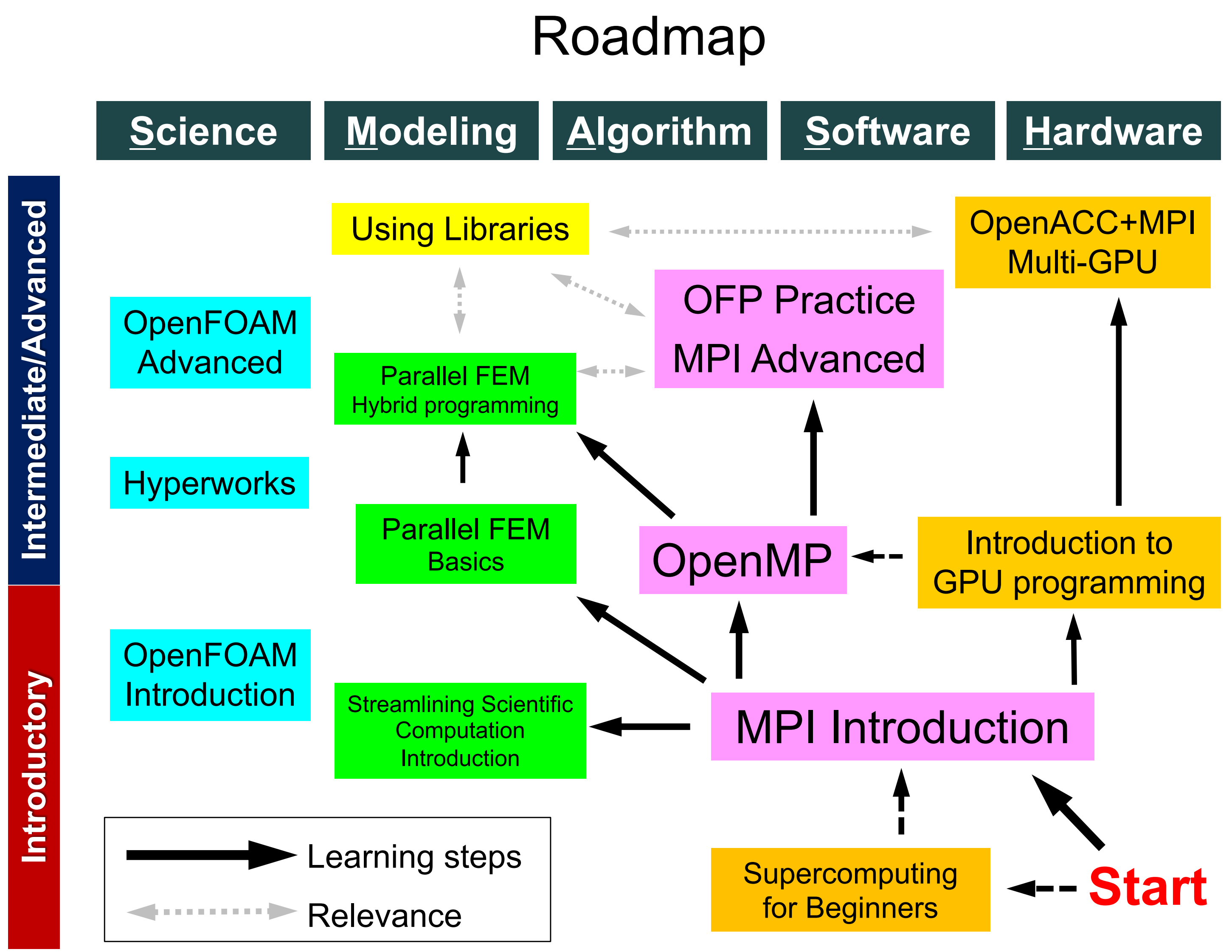
For advanced usage（lecture）

Important notices

- Your account will be valid for one month (from today).
  - Comply with "Terms of use" (https://www.cc.u-tokyo.ac.jp/en/guide/application/rules.php)
    - ✓ You may use supercomptuers only for contributing to academic research, education, and society.
  - Your account and your files on the supercomputer will be deleted afterwards.
  - The account will be disabled if you do not keep attending to this lecture till its end.
- For participants from industries: please finish one of the series of these workshops@ITC, if you want to apply to "trial usage".
- If you have any questions, do not ask ITC official e-mail. Please write instead to the lecturer in charge:

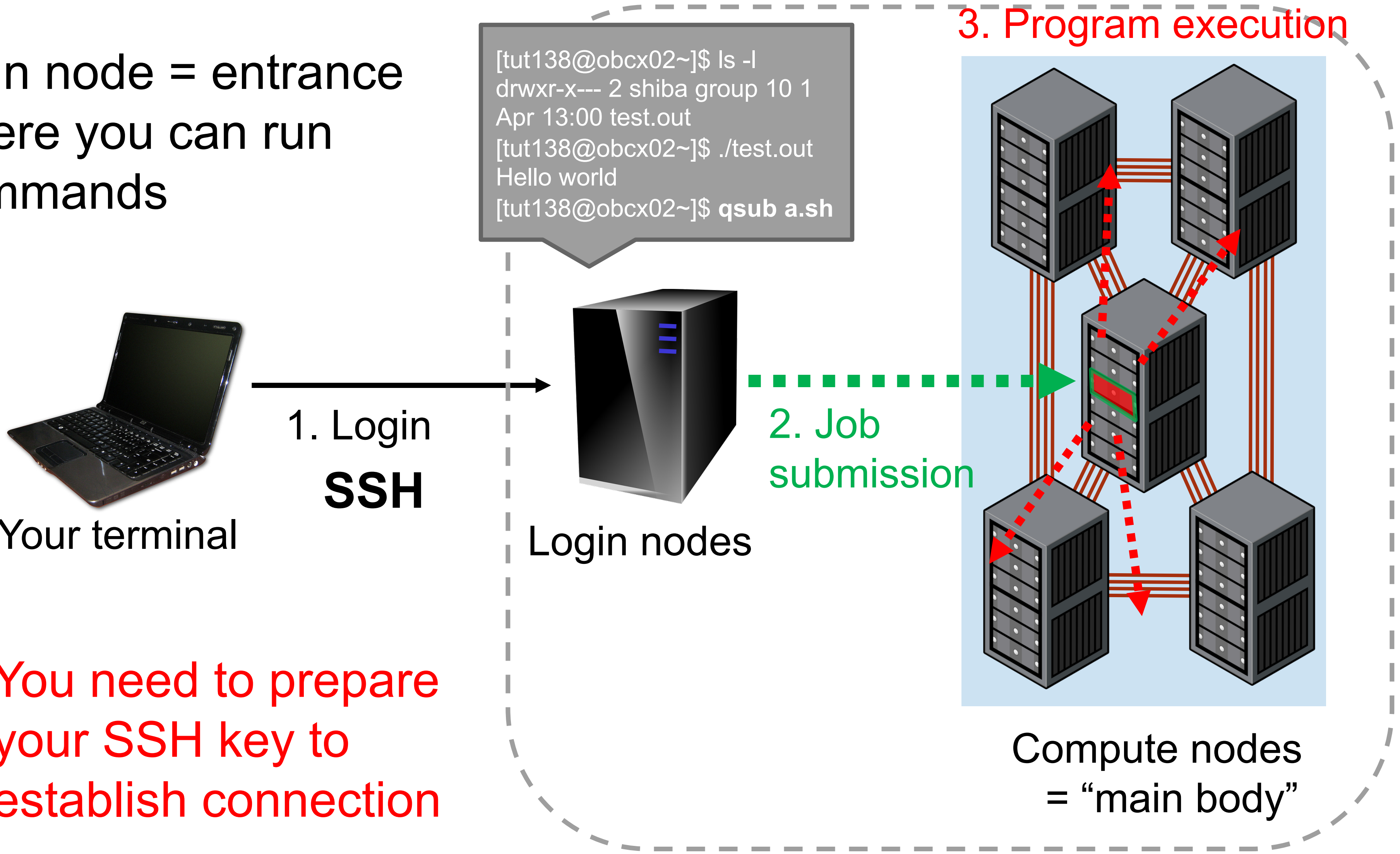
  shiba [at] cc.u-tokyo.ac.jp

Supercomputer workshop (tutorials) at ITC, U-Tokyo



How to use supercomputer

login node = entrance where you can run commands

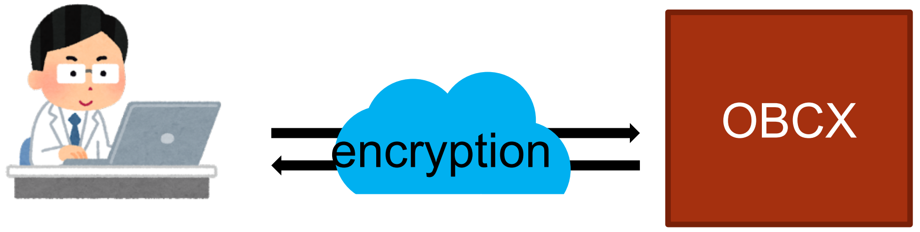


```
[tut138@obcx02~]$ ls -l
drwxr-x--- 2 shiba group 10 1
Apr 13:00 test.out
[tut138@obcx02~]$ ./test.out
Hello world
[tut138@obcx02~]$ qsub a.sh
```

You need to prepare your SSH key to establish connection

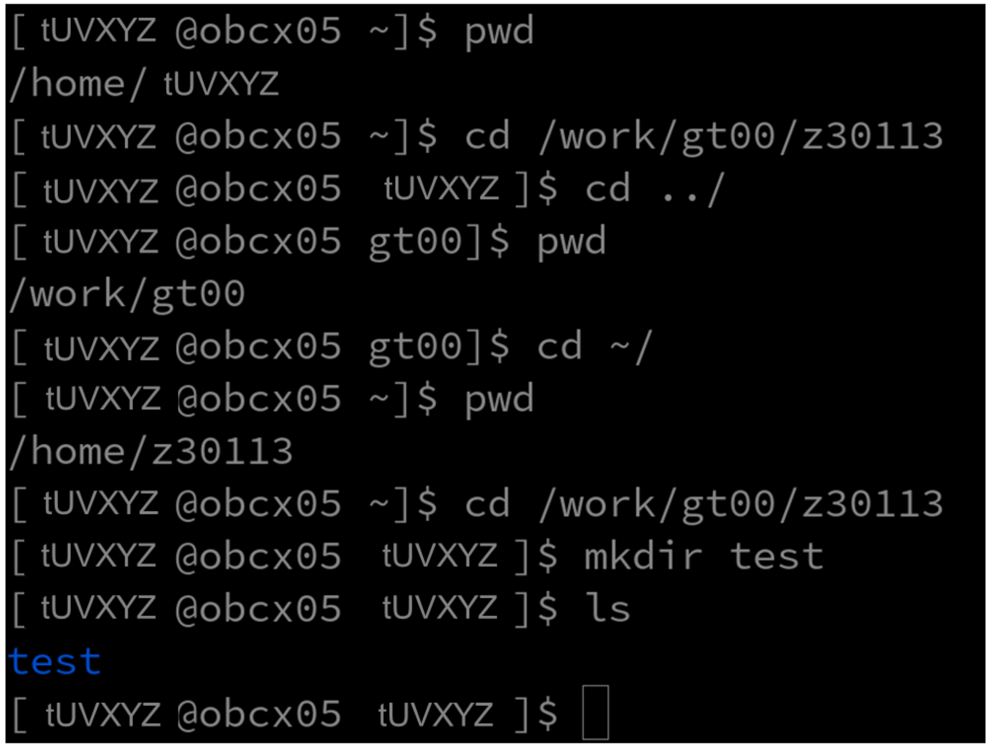
Secure Shell protocol

- Shell = command base software, connecting the user and the OS
- SSH = encrypted connection that enables remote connection



Via the SSH connection, you can
- Copy files
- Forward the GUI
- Tunneling
- Mount to your directory

Shell command example after SSH login



```
[ tUVXYZ @obcx05 ~]$ pwd
/home/ tUVXYZ
[ tUVXYZ @obcx05 ~]$ cd /work/gt00/z30113
[ tUVXYZ @obcx05  tUVXYZ ]$ cd ../
[ tUVXYZ @obcx05 gt00]$ pwd
/work/gt00
[ tUVXYZ @obcx05 gt00]$ cd ~/
[ tUVXYZ @obcx05 ~]$ pwd
/home/z30113
[ tUVXYZ @obcx05 ~]$ cd /work/gt00/z30113
[ tUVXYZ @obcx05  tUVXYZ ]$ mkdir test
[ tUVXYZ @obcx05  tUVXYZ ]$ ls
test
[ tUVXYZ @obcx05  tUVXYZ ]$
```

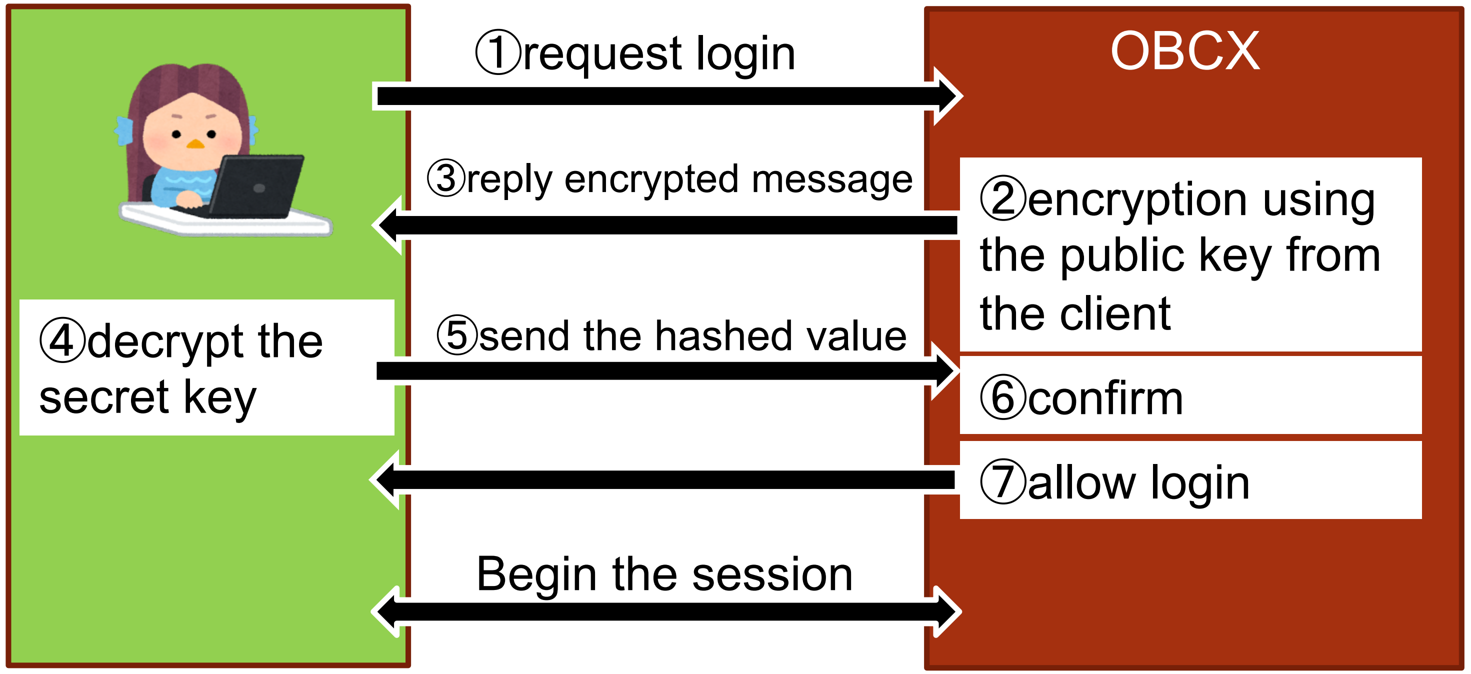
SSH key authentication

enables secure connection to supercomputers

Use “SSH key pairs” instead of (plain text) passwords.

Initial settings (for the first login)

- Generate key pairs
- Register the public keys on the login node



Cautions on SSH key authentication

Your secret key should be kept strictly confidential

✓ to prevent illegal login of other people.

● Do not copy your secret key to any other places

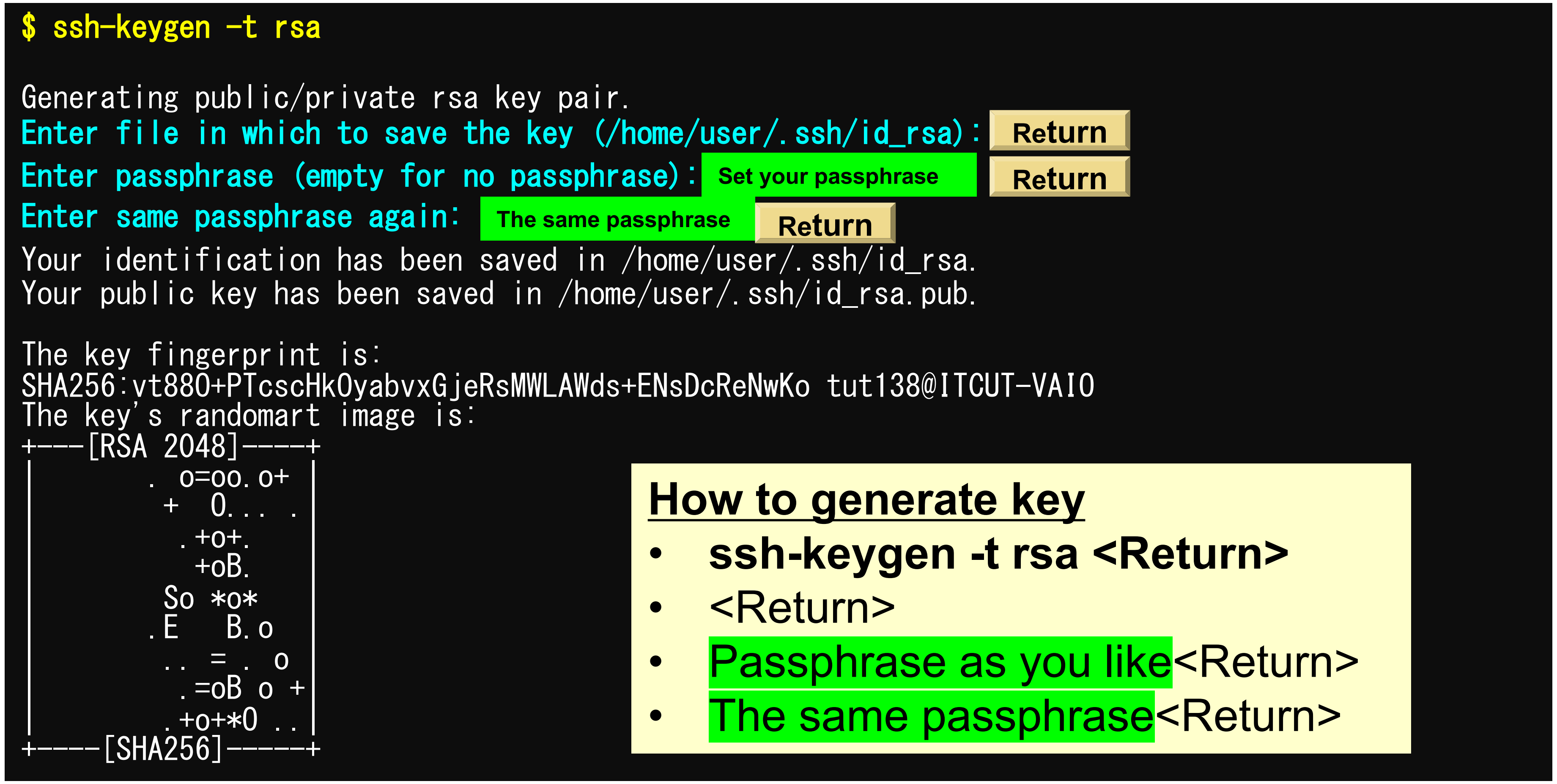
= The key pairs should not be recycled (on other machines)

- Set PASSPHRASE when generating your SSH key!
- The passphrase of your key should be different from other passwords, including those of OBCX user portal and user password on OBCX.

SSH key authentication

Generate your key pair on your local computer (1/3)

Open Cygwin or Terminal and begin the following operation



```
$ ssh-keygen -t rsa

Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.

The key fingerprint is:
SHA256:vt88O+PTcscHkOyabvxGjeRsMWLAWds+ENsDcReNwKo tut138@ITCUT-VAIO
The key's randomart image is:
+---[RSA 2048]----+
|       .  o=oo.o+|
|        +  O... .|
|         .+o+.   |
|          +oB.   |
|        So *o*   |
|       .E   B.o  |
|        ..  = .  o|
|         .=oB o +|
|        .+o+*O ..|
+----[SHA256]-----+
```

How to generate key

- **ssh-keygen -t rsa <Return>**
- <Return>
- Passphrase as you like<Return>
- The same passphrase<Return>

SSH key authentication

Confirm that there are both secret and public keys on your local computer（2/3）

If you do not understand the meaning, type letters and confirm a similar output

```
$ cd .ssh

$ ls

 id_rsa                              ⇒  Private Key
 id_rsa.pub                          ⇒  Public Key

$ cat id_rsa.pub

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDa6InmOYYaCrWjQDukjiNEfdW8veUwJyZtEI3oDuOA28eey6pOwbtI7JB
O9xnI17O7HG4yYvOM81+/nlAHy5tAfJlyOdsPzjTgdTBLdgi3cSf5pWEY6U96yaErOEi8Wge1HkXrhcwUjGDVTz
vTORefe6zLdRziL/KNmmesSQfR5lsZ/ihsjMgFxGaKsHHq/IErCtHIIIf9V/Ds2yj6vkAaWH6asBn+ZsRiRFvwH
PhkYAnp/j3LY6b8QfqgOp4WZRenh/HgySWTYIGi8x67VzMaUlm9qlKOQFMCaK2rivX1fmbwyWJ/vrWDqiek6YXo
xLDu+GPeQ4CPvxJcZnqF9gf3 tut138@ITCUT-VAIO
```

SSH key authentication

Copy public key（3/3）

Cut and paste your “id_rsa.pub” file

```
$ cd .ssh

$ ls

  id_rsa
  id_rsa.pub

$ cat id_rsa.pub

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDa6InmOYYaCrWjQDukjiNEfdW8veUwJyZtEI3oDuOA28eey6pOwbtI7JB
O9xnI17O7HG4yYvOM81+/nlAHy5tAfJlyOdsPzjTgdTBLdgi3cSf5pWEY6U96yaErOEi8Wge1HkXrhcwUjGDVTz
vTORefe6zLdRziL/KNmmesSQfR5lsZ/ihsjMgFxGaKsHHq/IErCtHIIIf9V/Ds2yj6vkAaWH6asBn+ZsRiRFvwH
PhkYAnp/j3LY6b8QfqgOp4WZRenh/HgySWTYIGi8x67VzMaUlm9qlKOQFMCaK2rivX1fmbwyWJ/vrWDqiek6YXo
xLDu+GPeQ4CPvxJcZnqF9gf3 tut138@ITCUT-VAIO
```

Procedure
- **cat id_rsa.pub <Return>**
- Drag and Drop from “ssh-rsa” to the filnal part(“tut138@ITCUT-VAIO” in this case)

SSH key authentication

id_rsa

- Private Key — keep it on your PC
- Keep it confidential, do not move it from where it is generated, do not send it to others

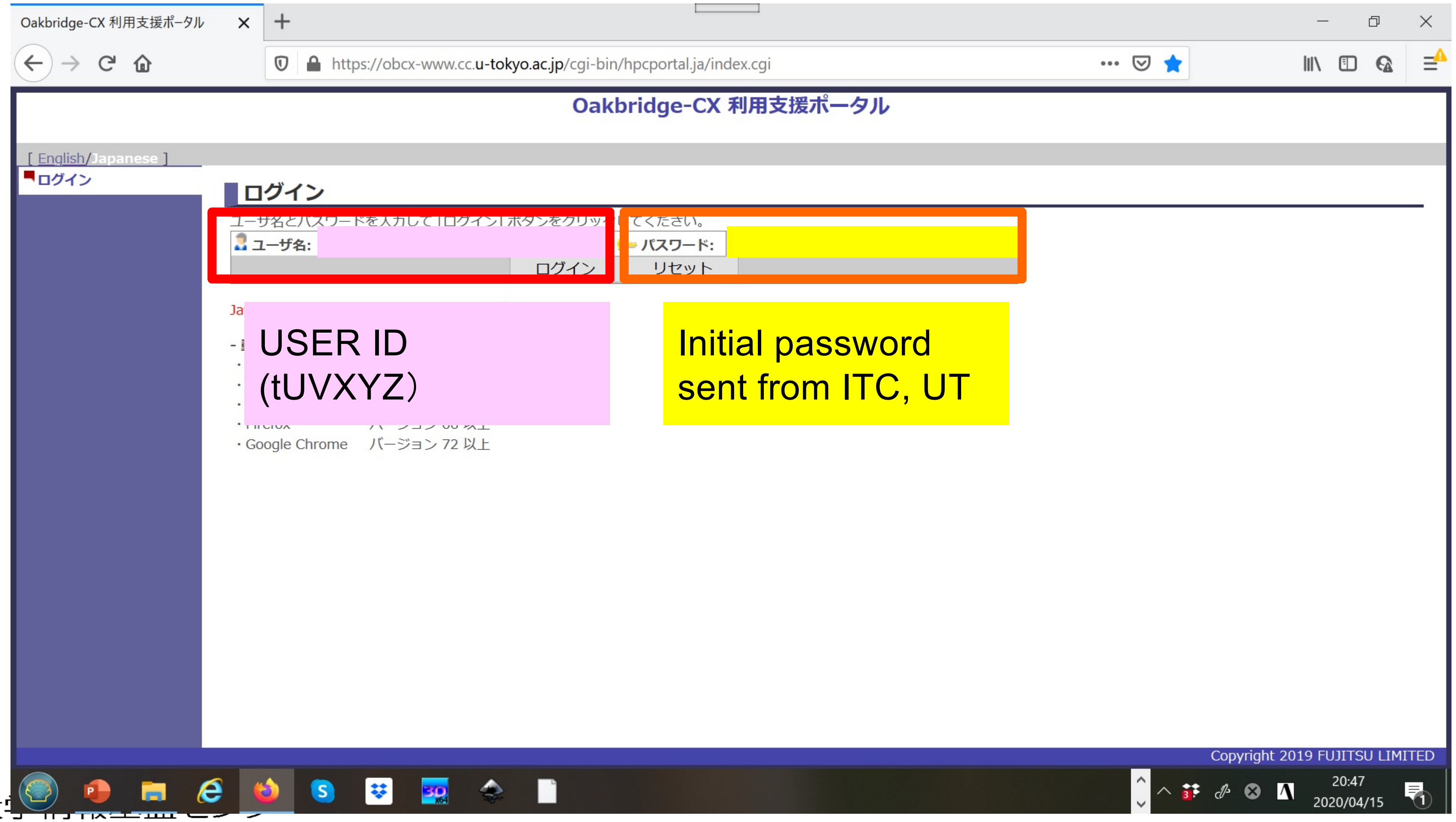
id_rsa.pub

- Public Key — put on supercomputer
- You may copy and send it to others by e-mail.

- If you want to log into a supercomputer from multiple local computers, then generate a pair of public and private keys on each local computer. You can register multiple public keys.

(1) Login to OBCX user portal

https://obcx-www.cc.u-tokyo.ac.jp/cgi-bin/hpcportal.en/index.cgi

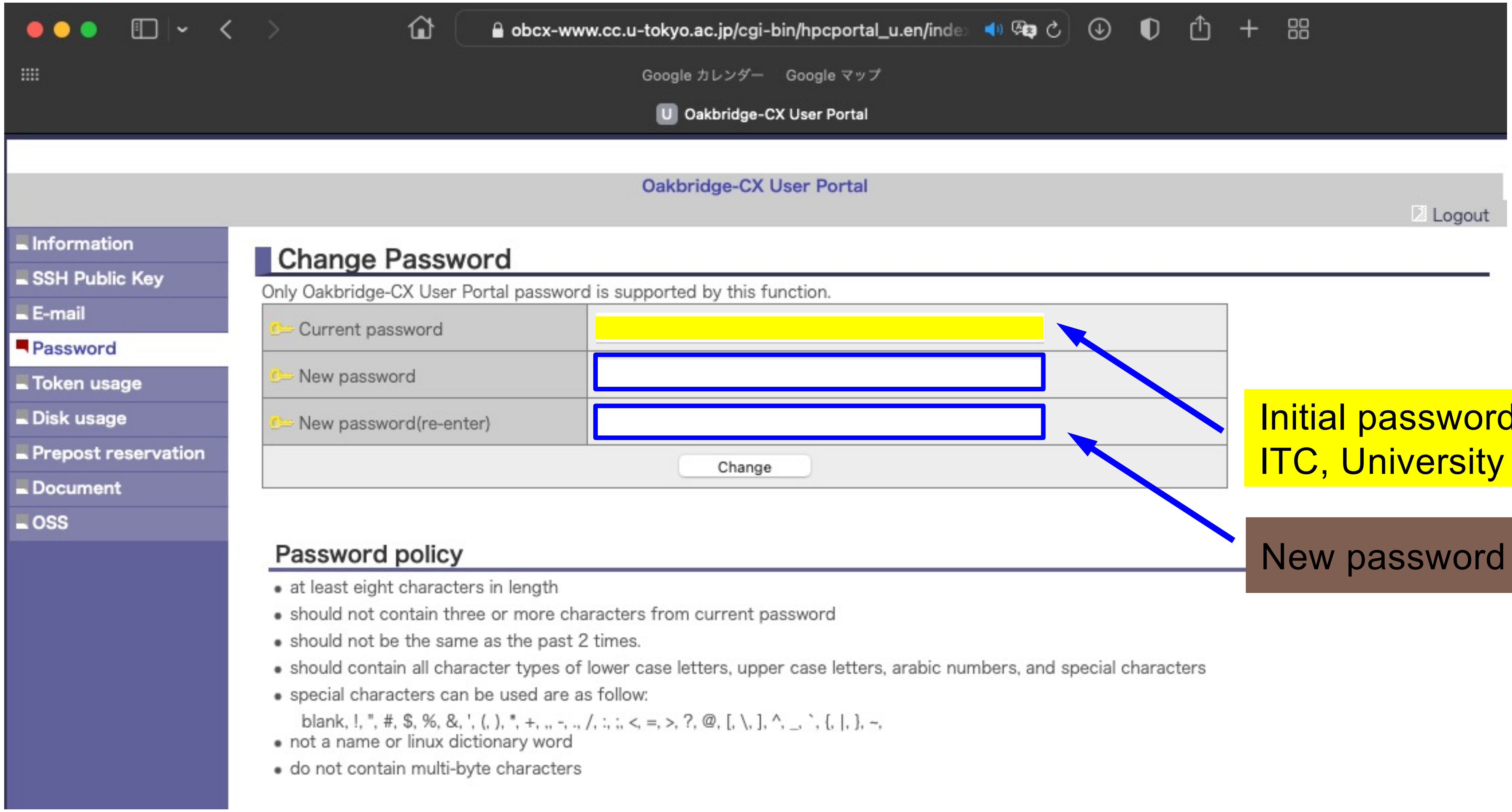


Oakbridge-CX 利用支援ポータル
https://obcx-www.cc.u-tokyo.ac.jp/cgi-bin/hpcportal.ja/index.cgi
Oakbridge-CX 利用支援ポータル
[English/Japanese]
ログイン
ログイン
ユーザ名とパスワードを入力して「ログイン」ボタンをクリックしてください。
ユーザ名:
パスワード:
ログイン
リセット
・Google Chrome バージョン 72 以上
Copyright 2019 FUJITSU LIMITED
20:47
2020/04/15

USER ID (tUVXYZ)

Initial password sent from ITC, UT

(2) Change your portal password



Change Password

Only Oakbridge-CX User Portal password is supported by this function.

Current password	
New password	
New password(re-enter)	

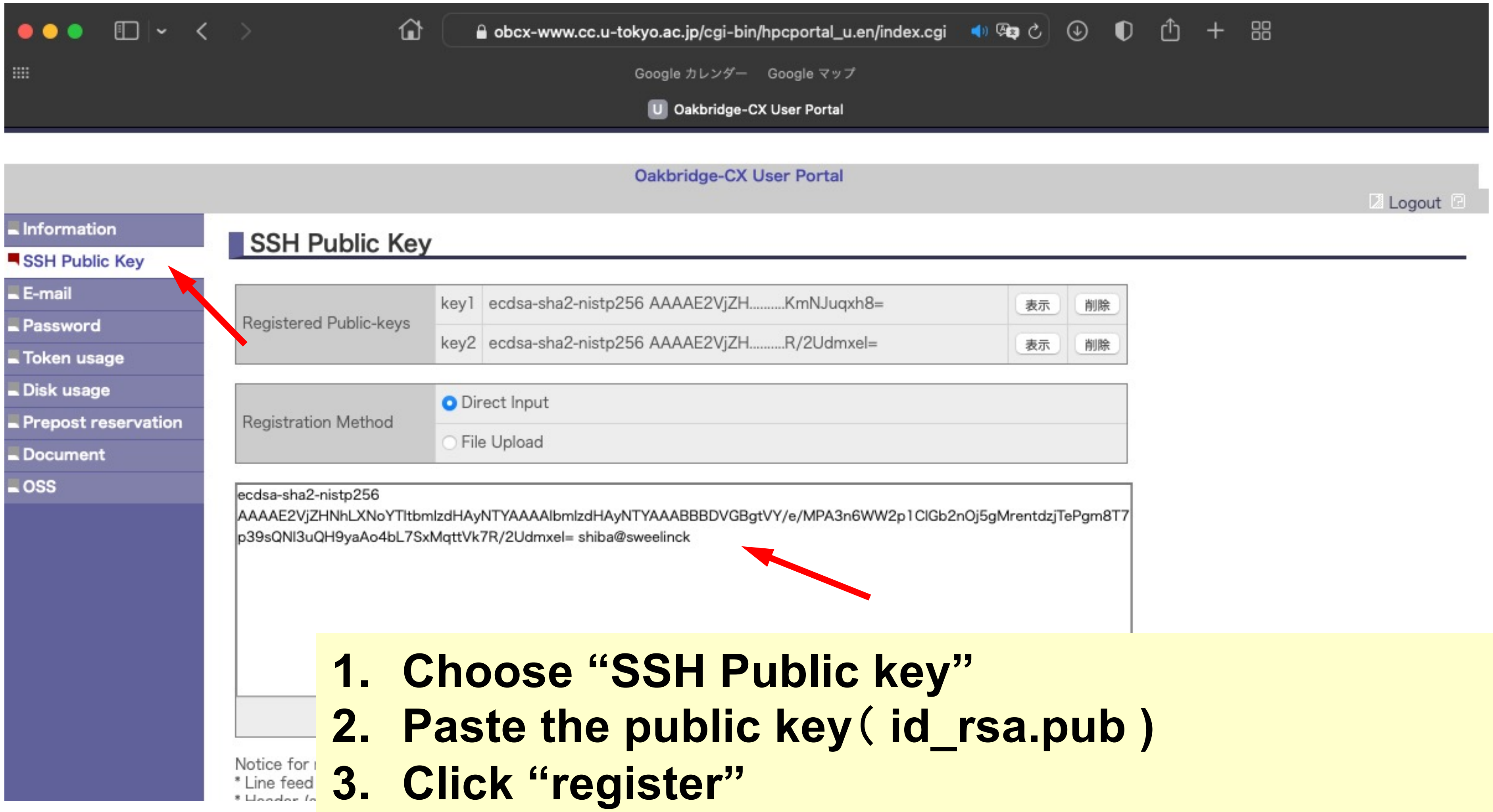
Password policy

- at least eight characters in length
- should not contain three or more characters from current password
- should not be the same as the past 2 times.
- should contain all character types of lower case letters, upper case letters, arabic numbers, and special characters
- special characters can be used are as follow:
  blank, !, ", #, $, %, &, ', (,), *, +, ,, -, ., /, :, ;, <, =, >, ?, @, [, \,], ^, _, `, {, |, }, ~,
- not a name or linux dictionary word
- do not contain multi-byte characters

Initial password sent from ITC, University of Tokyo

New password (twice)

(3) public key registration （id_rsa.pub）



SSH Public Key

Registered Public-keys	key1	ecdsa-sha2-nistp256 AAAAE2VjZH.........KmNJuqxh8=	表示	削除
	key2	ecdsa-sha2-nistp256 AAAAE2VjZH.........R/2Udmxel=	表示	削除

Registration Method: [✓ Direct Input] File Upload

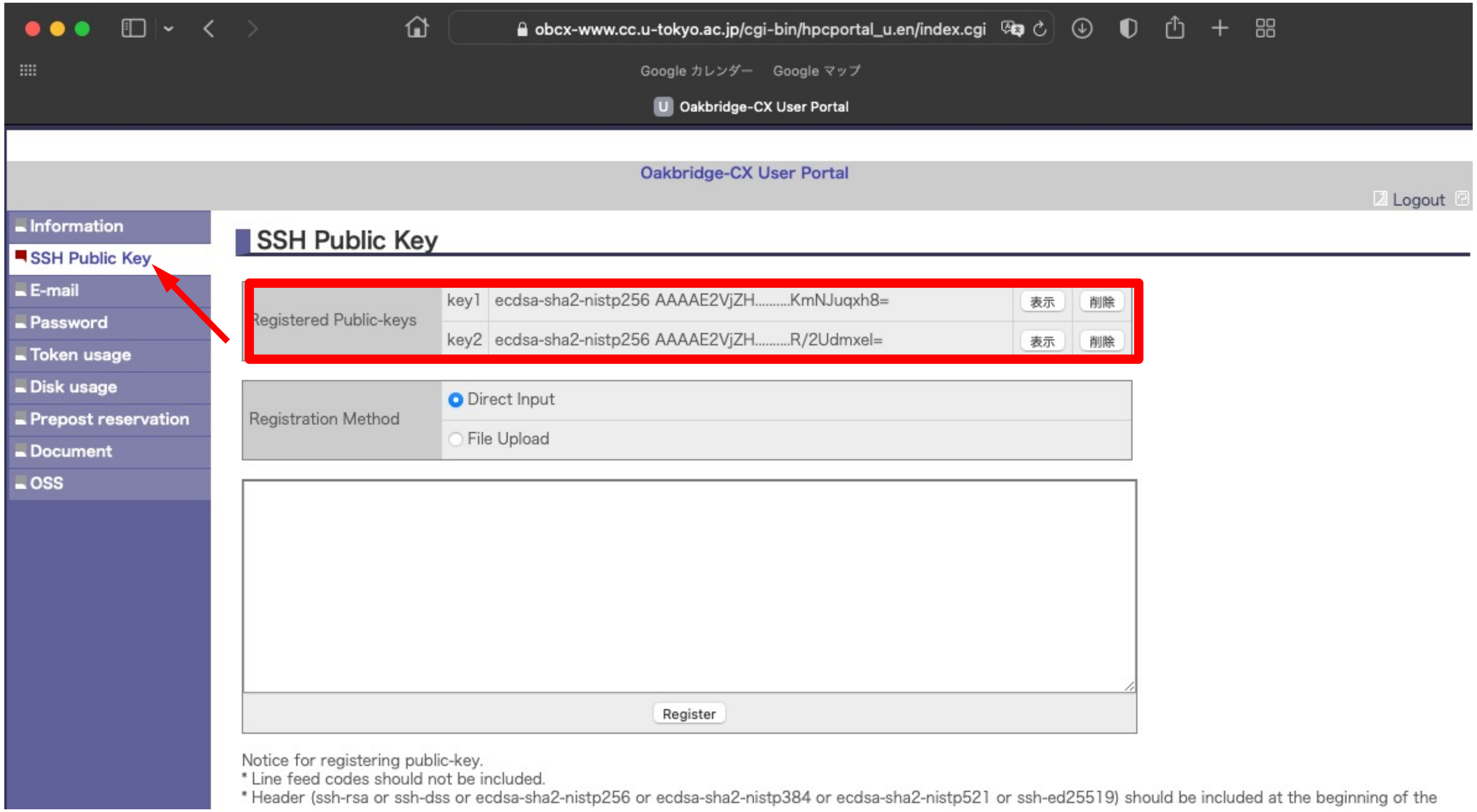
```
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBDVGBgtVY/e/MPA3n6WW2p1ClGb2nOj5gMrentdzjTePgm8T7
p39sQNl3uQH9yaAo4bL7SxMqttVk7R/2Udmxel= shiba@sweelinck
```

Notice for
* Line feed
* Header (

1. **Choose “SSH Public key”**
2. **Paste the public key(id_rsa.pub)**
3. **Click “register”**

Ask the staffs if you have troubles

More than one public keys can be registered on OBCX



Oakbridge-CX User Portal

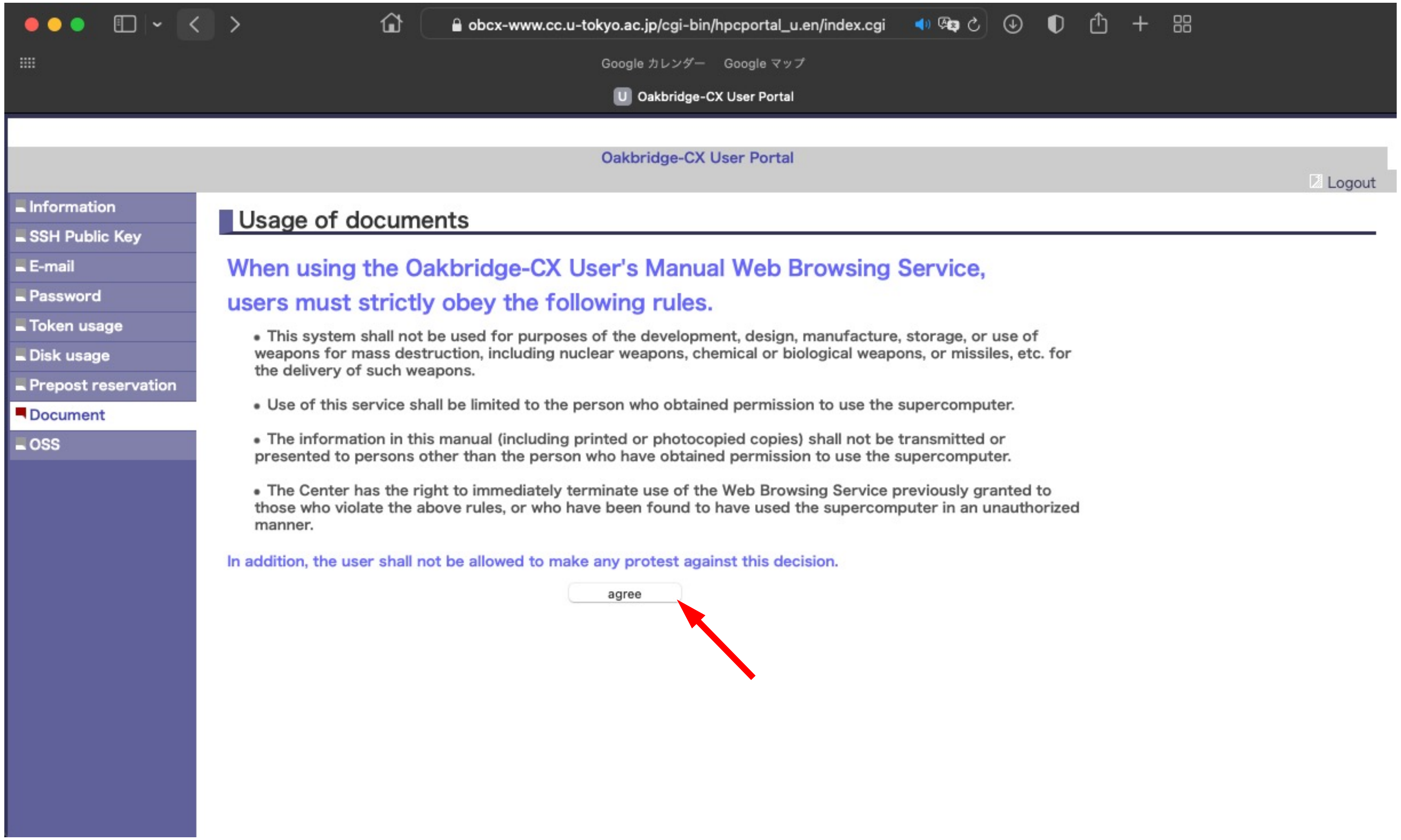
SSH Public Key

Registered Public-keys	key1	ecdsa-sha2-nistp256 AAAAE2VjZH.........KmNJuqxh8=	表示	削除
	key2	ecdsa-sha2-nistp256 AAAAE2VjZH.........R/2Udmxel=	表示	削除

Registration Method	[✓ Direct Input]
	File Upload

Notice for registering public-key.
* Line feed codes should not be included.
* Header (ssh-rsa or ssh-dss or ecdsa-sha2-nistp256 or ecdsa-sha2-nistp384 or ecdsa-sha2-nistp521 or ssh-ed25519) should be included at the beginning of the

Get Manual PDF files on OBCX portal （1/2）



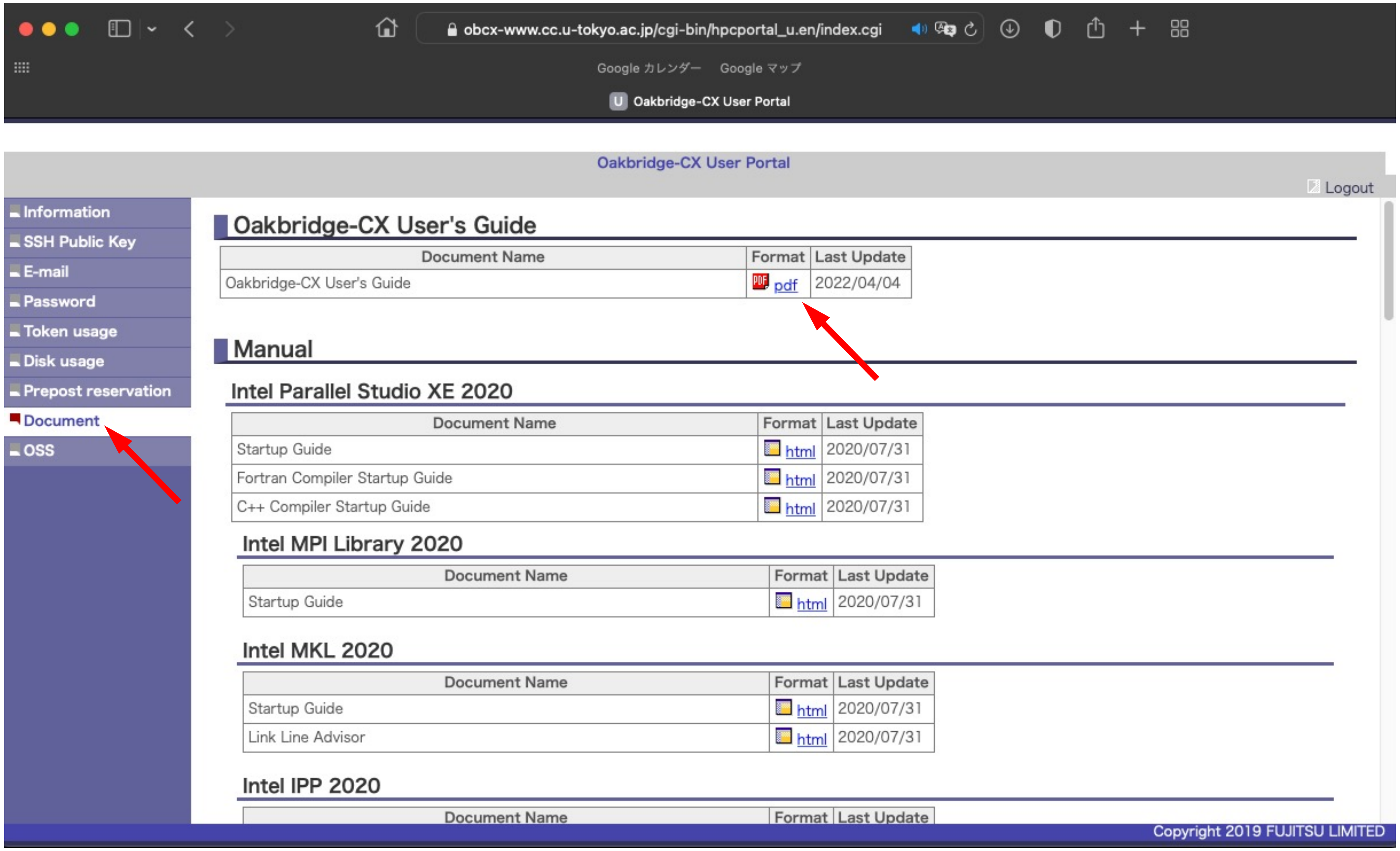
Usage of documents

When using the Oakbridge-CX User's Manual Web Browsing Service, users must strictly obey the following rules.

- This system shall not be used for purposes of the development, design, manufacture, storage, or use of weapons for mass destruction, including nuclear weapons, chemical or biological weapons, or missiles, etc. for the delivery of such weapons.
- Use of this service shall be limited to the person who obtained permission to use the supercomputer.
- The information in this manual (including printed or photocopied copies) shall not be transmitted or presented to persons other than the person who have obtained permission to use the supercomputer.
- The Center has the right to immediately terminate use of the Web Browsing Service previously granted to those who violate the above rules, or who have been found to have used the supercomputer in an unauthorized manner.

In addition, the user shall not be allowed to make any protest against this decision.

Get Manual PDF files on OBCX portal （2/2）



obcx-www.cc.u-tokyo.ac.jp/cgi-bin/hpcportal_u.en/index.cgi

Oakbridge-CX User Portal

Logout

- Information
- SSH Public Key
- E-mail
- Password
- Token usage
- Disk usage
- Prepost reservation
- Document
- OSS

Oakbridge-CX User's Guide

Document Name	Format	Last Update
Oakbridge-CX User's Guide	pdf	2022/04/04

Manual

Intel Parallel Studio XE 2020

Document Name	Format	Last Update
Startup Guide	html	2020/07/31
Fortran Compiler Startup Guide	html	2020/07/31
C++ Compiler Startup Guide	html	2020/07/31

Intel MPI Library 2020

Document Name	Format	Last Update
Startup Guide	html	2020/07/31

Intel MKL 2020

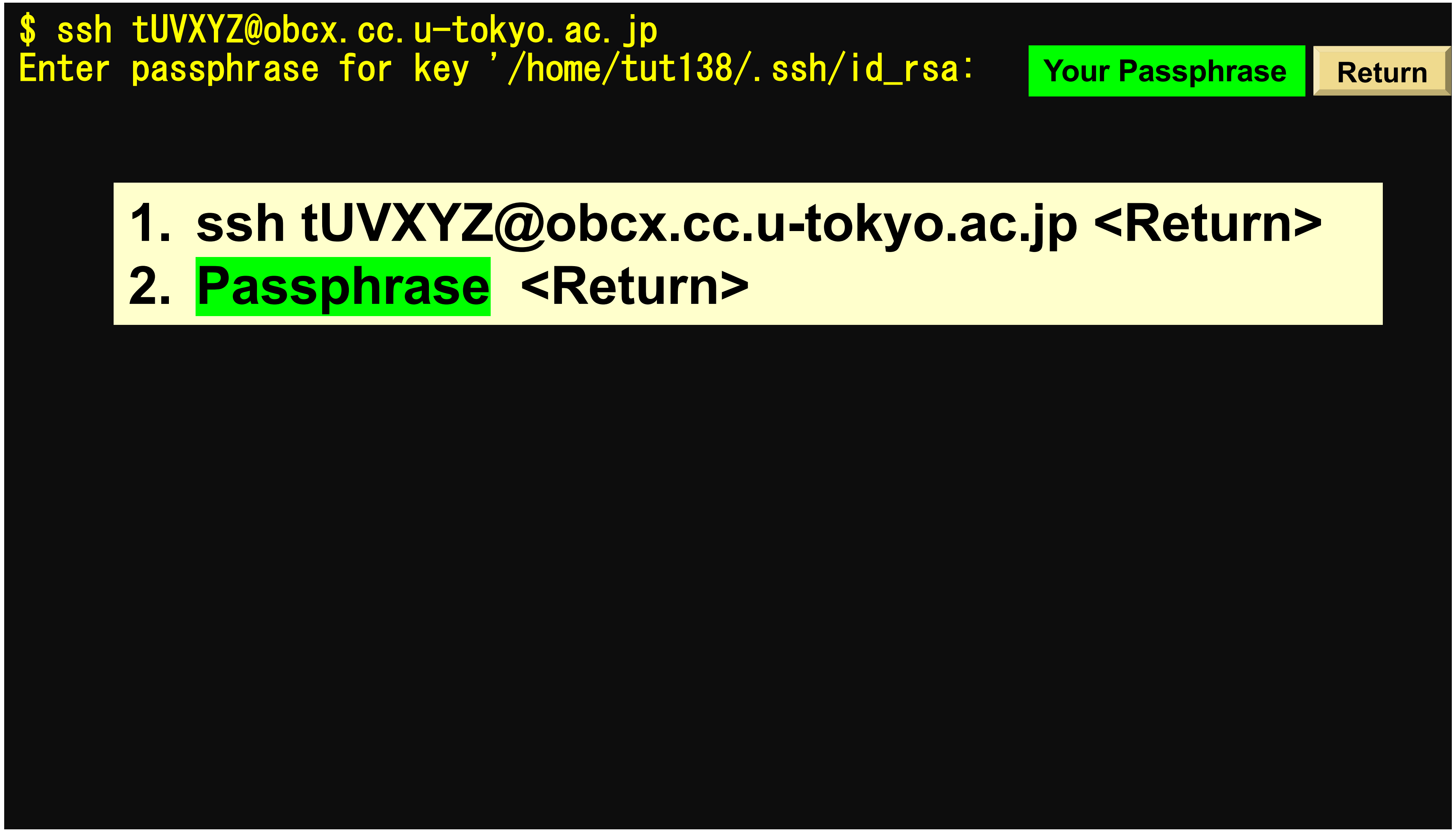
Document Name	Format	Last Update
Startup Guide	html	2020/07/31
Link Line Advisor	html	2020/07/31

Intel IPP 2020

Document Name	Format	Last Update

Copyright 2019 FUJITSU LIMITED

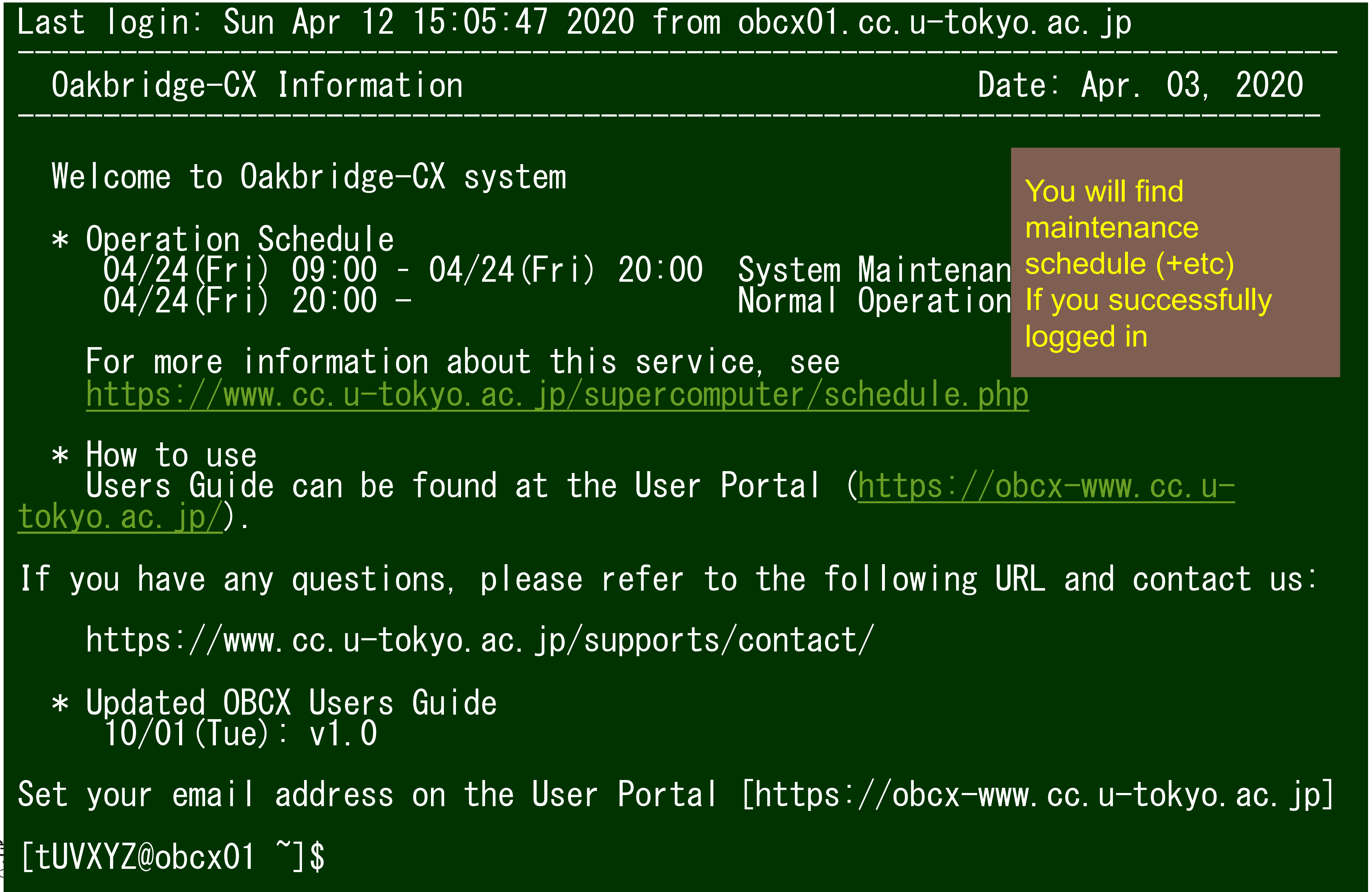
Login to OBCX supercomputer



```
$ ssh tUVXYZ@obcx.cc.u-tokyo.ac.jp
Enter passphrase for key '/home/tut138/.ssh/id_rsa:
```

1. ssh tUVXYZ@obcx.cc.u-tokyo.ac.jp <Return>
2. Passphrase <Return>

Login to OBCX



```
Last login: Sun Apr 12 15:05:47 2020 from obcx01.cc.u-tokyo.ac.jp
-----------------------------------------------------------------------------
  Oakbridge-CX Information                                  Date: Apr. 03, 2020
-----------------------------------------------------------------------------

  Welcome to Oakbridge-CX system

  * Operation Schedule
     04/24(Fri) 09:00 - 04/24(Fri) 20:00   System Maintenan
     04/24(Fri) 20:00 -                    Normal Operation

    For more information about this service, see
    https://www.cc.u-tokyo.ac.jp/supercomputer/schedule.php

  * How to use
    Users Guide can be found at the User Portal (https://obcx-www.cc.u-
tokyo.ac.jp/).

If you have any questions, please refer to the following URL and contact us:

    https://www.cc.u-tokyo.ac.jp/supports/contact/

  * Updated OBCX Users Guide
     10/01(Tue): v1.0

Set your email address on the User Portal [https://obcx-www.cc.u-tokyo.ac.jp]

[tUVXYZ@obcx01 ~]$
```

You will find maintenance schedule (+etc)
If you successfully logged in

After login

You will see the linux shell → type commands

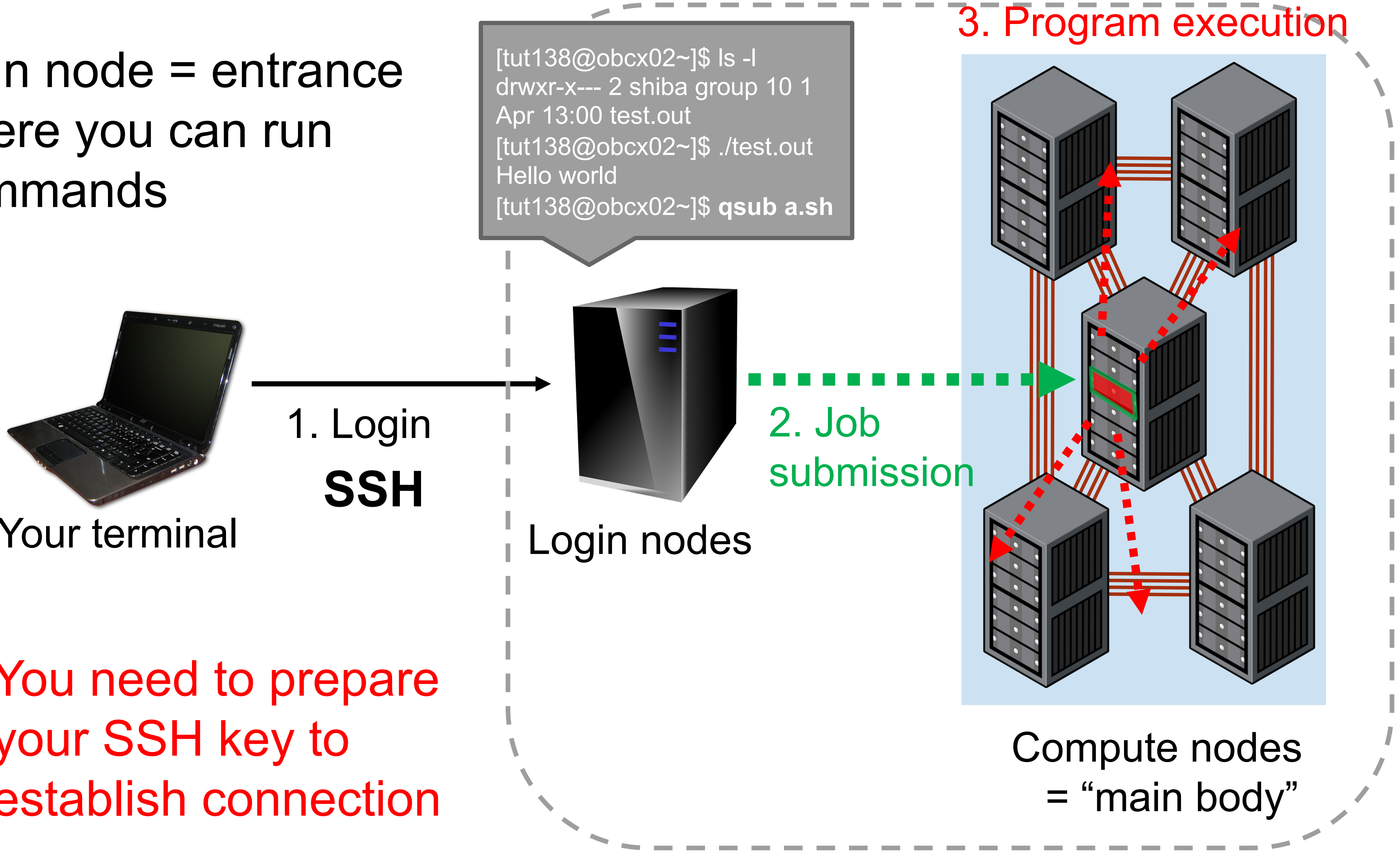
```
[tUVXYZ@obcx01 ~]$ pwd Return

/home/tUVXYZ
```

pwd ― shows the present directory

Break

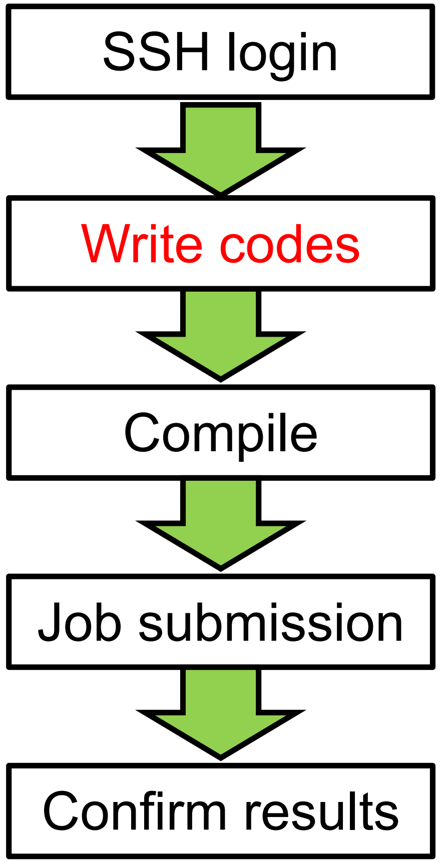
How to use supercomputer

login node = entrance where you can run commands


```
[tut138@obcx02~]$ ls -l
drwxr-x--- 2 shiba group 10 1
Apr 13:00 test.out
[tut138@obcx02~]$ ./test.out
Hello world
[tut138@obcx02~]$ qsub a.sh
```

You need to prepare your SSH key to establish connection

How to use supercomputers ?



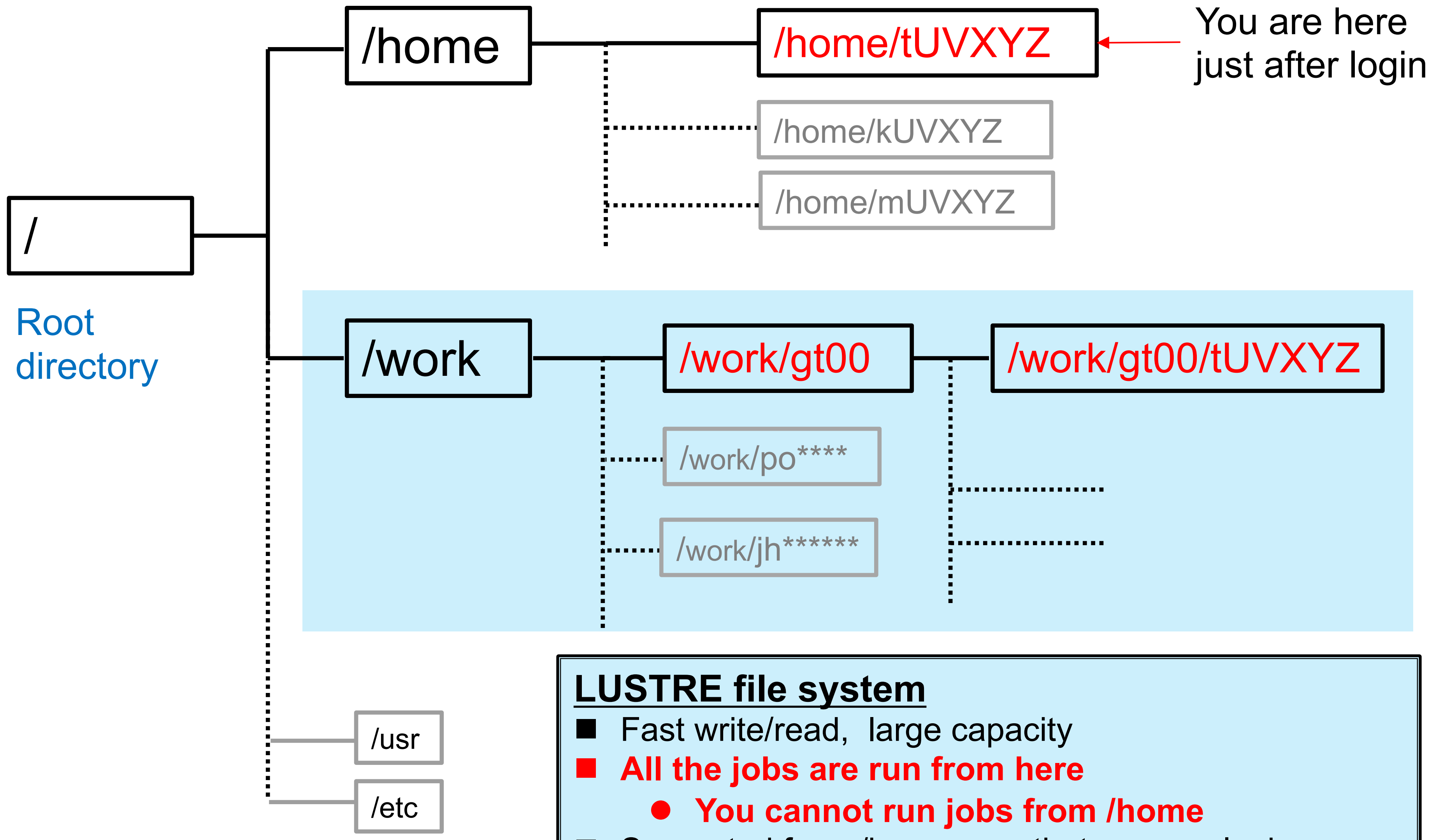
Putty、Cygwin、SSH、key authentication

How to use editors (emacs, vi, nano), Linux basics

Environment module, compiler options

Writing job scripts, how to manage jobs

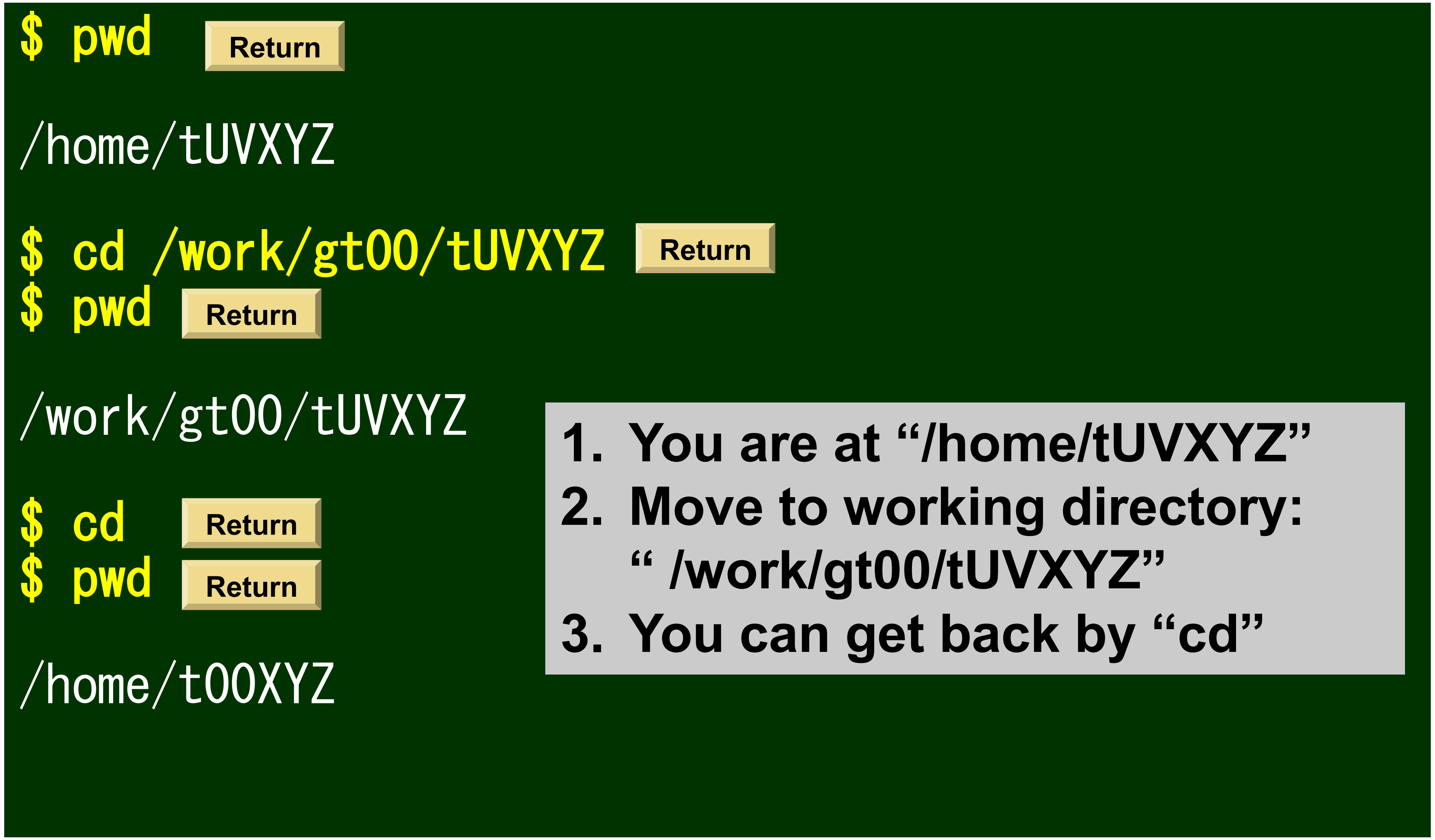
Linux Directories on OBCX Linux



LUSTRE file system

- Fast write/read, large capacity
- **All the jobs are run from here**
  - **You cannot run jobs from /home**
- Separated from /home , so that you can login even when /work is in trouble

After you logged in…



```
$ pwd   Return

/home/tUVXYZ

$ cd /work/gt00/tUVXYZ  Return
$ pwd  Return

/work/gt00/tUVXYZ

$ cd   Return
$ pwd  Return

/home/t00XYZ
```

1. You are at "/home/tUVXYZ"
2. Move to working directory: " /work/gt00/tUVXYZ"
3. You can get back by "cd"

Exercise: computing Fibonacci sequence

defined as a recurrence relation

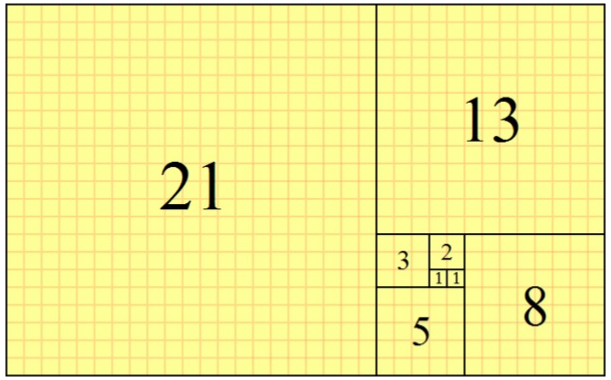
$$F_0 = 1$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2} \quad (n \geq 2)$$

1, 1, 2, 3, 5, 8, 13, 21 ….

Iterative computation is a starting point for using supercomputers



From Wikipedia
https://ja.wikipedia.org/wiki/フィボナッチ数

Fibonacci sequence, up to 92nd

1	75025	7778742049	806515533049393
1	121393	12586269025	1304969544928657
2	196418	20365011074	2111485077978050
3	317811	32951280099	3416454622906707
5	514229	53316291173	5527939700884757
8	832040	86267571272	8944394323791464
13	1346269	139583862445	14472334024676221
21	2178309	225851433717	23416728348467685
34	3524578	365435296162	37889062373143906
55	5702887	591286729879	61305790721611591
89	9227465	956722026041	99194853094755497
144	14930352	1548008755920	160500643816367088
233	24157817	2504730781961	259695496911122585
377	39088169	4052739537881	420196140727489673
610	63245986	6557470319842	679891637638612258
987	102334155	10610209857723	1100087778366101931
1597	165580141	17167680177565	1779979416004714189
2584	267914296	27777890035288	2880067194370816120
4181	433494437	44945570212853	4660046610375530309
6765	701408733	72723460248141	7540113804746346429
10946	1134903170	117669030460994	
17711	1836311903	190392490709135	
28657	2971215073	308061521170129	
46368	4807526976	498454011879264	

Task:
Compute this sequence, and find "747031" in the result

Compose a program using text editors

First, make a directory where you work on.

```
[tUVXYZ@obcx01 ~]$ cd Return
[tUVXYZ@obcx01 ~]$ mkdir fibonacci Return
[tUVXYZ@obcx01 ~]$ cd fibonacci Return
```

Please use text editors such as Emacs, vim, and nano.

例1： Compose a Fortran code using Emacs

```
[tUVXYZ@obcx01 fibonacci]$ emacs fibonacci.f90 Return
```

例2： Compose a Python script using vim

```
[tUVXYZ@obcx01 fibonacci]$ vim fibonacci.py Return
```

例3： Compose a C code using nano

```
[tUVXYZ@obcx01 fibonacci]$ nano fibonacci.c Return
```

Exercise: computing Fibonacci sequence

C

Edit the code

```
[tUVXYZ@obcx01 fibonacci]$ emacs fibonacci.c
```

Return

```
#include <stdio.h>

int main(void) {
    int i;
    long a, b, tmp;

    a=1;
    b=1;
    printf("%ld\n", a);

    for (i=2; i<=92; i++) {
        tmp = b;
        b = a + b;
        a = tmp;
        printf("%ld\n", a);
    }
}
```

Exercise: computing Fibonacci sequence

Fortran

Edit a code

```
[tUVXYZ@obcx01 fibonacci]$ emacs fibonacci.f90
```

Return

```
program main

    implicit none
    integer(kind=4)  i
    integer(kind=8)  a,b,tmp

    a = 1
    b = 1

    do i=2, 92
       tmp = b
       b= a + b
       a = tmp
       print *, a
    end do

end program main
```

Exercise: computing Fibonacci sequence

Python

Edit a code

```
[tUVXYZ@obcx01 fibonacci]$ emacs fibonacci.py
```

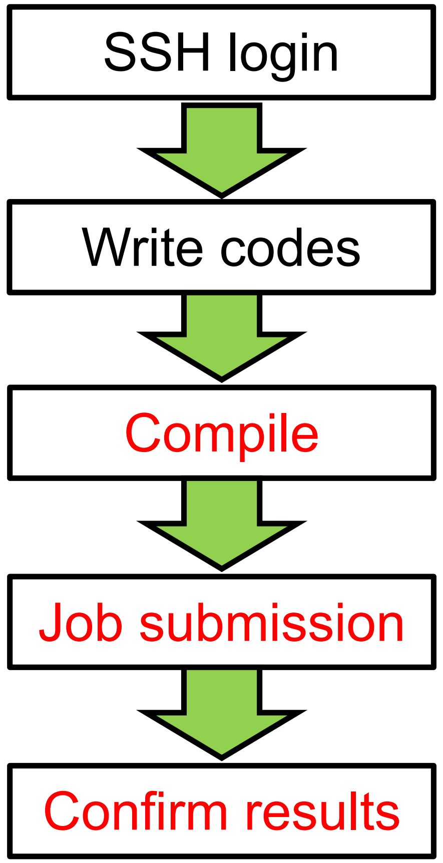
Return

```
a, b = 1, 1
print(a)

for i in range(1,92):
    a, b = b, a+b
    print(a)
```

In python…

- Data type is auto detected (e.g. 64 bit integrer)
- Multiple variables can be updated at once.
- Compilation not necessary.

How to use supercomputers ?



Putty、Cygwin、SSH、key authentication

How to use editors (emacs, vi, nano), Linux basics

Environment module, compiler options

Writing job scripts, how to manage jobs

Environment modules (1)

We can choose compilers (etc)

gnu compiler, intel compiler, python interpreters, …

We can choose libraries (for accelerating computations)

Fast Fourier Transform, linear algebra/simultaneous linear equations, and etc…

Show modules which is in use

```
[tUVXYZ@obcx01 ~]$ module list
Currently Loaded Modulefiles:
  1) impi/2019.9.304    2) intel/2020.4.304
```

Return

Intel MPI library   Intel compiler

Clear up current settings (we want to switch to others)

```
[tUVXYZ@obcx01 ~]$ module purge
[tUVXYZ@obcx01 ~]$ module list
No Modulefiles Currently Loaded.
```

Return

Return

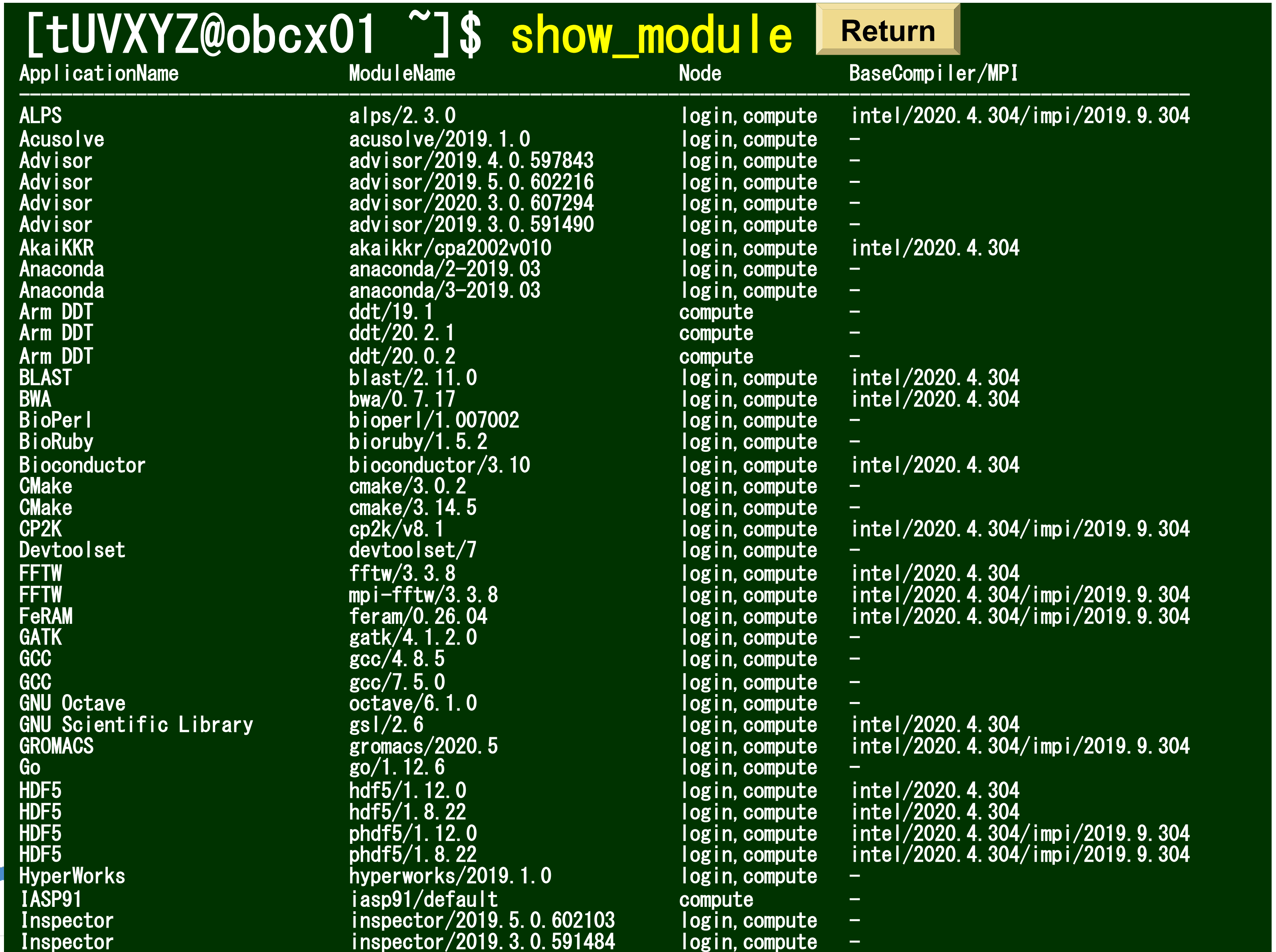
Displaying available environment modules

```
[tUVXYZ@obcx01 ~]$ module avail
```

Return

```
------ /home/opt/local/modulefiles/L/mpi/intel/2019.5.281/impi/2019.5.281 ------
alps/2.3.0(default)                   phdf5/1.10.5(default)
feram/0.26.04(default)                pnetcdf/1.11.2(default)
frontflow_blue/8.1(default)           ppohBEM/0.5.0(default)
frontistr/4.5(default)                ppohDEM_util/1.0.0(default)
modylas/1.0.4(default)                ppohFDM/0.3.1(default)
mpi-fftw/3.3.8(default)               ppohFEM/1.0.1(default)
netcdf-fortran-parallel/4.4.5(default) ppohFVM/0.3.0(default)
netcdf-parallel/4.7.0(default)        pt-scotch/6.0.6(default)
openmx/3.8(default)                   revocap_coupler/2.1(default)
parmetis/4.0.3(default)               superlu_dist/6.1.1(default)
petsc/3.11.2(default)                 xtapp/rc-150401(default)
phase/2019.01(default)

------------ /home/opt/local/modulefiles/L/compiler/intel/2019.5.281 ------------
R/3.6.0(default)               metis/5.1.0(default)
akaikkr/cpa2002v010(default)   mt-metis/0.6.0(default)
bioconductor/3.10(default)     netcdf/4.7.0(default)
blast/2.9.0(default)           netcdf-cxx/4.3.0(default)
bwa/0.7.17(default)            netcdf-fortran/4.4.5(default)
fftw/3.3.8(default)            paraview/5.6.1(default)
gsl/2.5(default)               povray/3.7.0.8(default)
hdf5/1.10.5(default)           ppohAT/1.0.0(default)
hdf5/1.8.21                    revocap_refiner/1.1.04(default)
impi/2019.5.281(default)       samtools/1.9(default)
intelpython/2.7                scotch/6.0.7(default)
intelpython/3.6(default)       superlu/5.2.1(default)
mesa/19.0.6(default)           superlu_mt/3.1(default)
metis/4.0.3                    xabclib/1.03(default)

---------------------- /home/opt/local/modulefiles/L/core ----------------------
acusolve/2019.1.0(default)          intel/2018.3.222
advisor/2019.3.0.591490             intel/2019.3.199
advisor/2019.4.0.597843             intel/2019.4.243
advisor/2019.5.0.602216(default)    intel/2019.5.281(default)
anaconda/2-2019.03                  intel/2020.1.217
anaconda/3-2019.03(default)         itac/2019.4.036
bioperl/1.007002(default)           itac/2019.5.041(default)
bioruby/1.5.2(default)              julia/1.4.0(default)
cmake/3.0.2                         llvm/7.1.0(default)
cmake/3.14.5(default)               massivethreads/0.97
```

To see at once complex module dependencies on OBCX



```
[tUVXYZ@obcx01 ~]$ show_module
ApplicationName               ModuleName                    Node            BaseCompiler/MPI
------------------------------------------------------------------------------------------------------------------
ALPS                          alps/2.3.0                    login,compute   intel/2020.4.304/impi/2019.9.304
Acusolve                      acusolve/2019.1.0             login,compute   -
Advisor                       advisor/2019.4.0.597843       login,compute   -
Advisor                       advisor/2019.5.0.602216       login,compute   -
Advisor                       advisor/2020.3.0.607294       login,compute   -
Advisor                       advisor/2019.3.0.591490       login,compute   -
AkaiKKR                       akaikkr/cpa2002v010           login,compute   intel/2020.4.304
Anaconda                      anaconda/2-2019.03            login,compute   -
Anaconda                      anaconda/3-2019.03            login,compute   -
Arm DDT                       ddt/19.1                      compute         -
Arm DDT                       ddt/20.2.1                    compute         -
Arm DDT                       ddt/20.0.2                    compute         -
BLAST                         blast/2.11.0                  login,compute   intel/2020.4.304
BWA                           bwa/0.7.17                    login,compute   intel/2020.4.304
BioPerl                       bioperl/1.007002              login,compute   -
BioRuby                       bioruby/1.5.2                 login,compute   -
Bioconductor                  bioconductor/3.10             login,compute   intel/2020.4.304
CMake                         cmake/3.0.2                   login,compute   -
CMake                         cmake/3.14.5                  login,compute   -
CP2K                          cp2k/v8.1                     login,compute   intel/2020.4.304/impi/2019.9.304
Devtoolset                    devtoolset/7                  login,compute   -
FFTW                          fftw/3.3.8                    login,compute   intel/2020.4.304
FFTW                          mpi-fftw/3.3.8                login,compute   intel/2020.4.304/impi/2019.9.304
FeRAM                         feram/0.26.04                 login,compute   intel/2020.4.304/impi/2019.9.304
GATK                          gatk/4.1.2.0                  login,compute   -
GCC                           gcc/4.8.5                     login,compute   -
GCC                           gcc/7.5.0                     login,compute   -
GNU Octave                    octave/6.1.0                  login,compute   -
GNU Scientific Library        gsl/2.6                       login,compute   intel/2020.4.304
GROMACS                       gromacs/2020.5                login,compute   intel/2020.4.304/impi/2019.9.304
Go                            go/1.12.6                     login,compute   -
HDF5                          hdf5/1.12.0                   login,compute   intel/2020.4.304
HDF5                          hdf5/1.8.22                   login,compute   intel/2020.4.304
HDF5                          phdf5/1.12.0                  login,compute   intel/2020.4.304/impi/2019.9.304
HDF5                          phdf5/1.8.22                  login,compute   intel/2020.4.304/impi/2019.9.304
HyperWorks                    hyperworks/2019.1.0           login,compute   -
IASP91                        iasp91/default                compute         -
Inspector                     inspector/2019.5.0.602103     login,compute   -
Inspector                     inspector/2019.3.0.591484     login,compute   -
```

BaseCompiler/MPI needs to be loaded before loading each module

Compilation & setting up your job

By environment modules

```
$ module [option] args
```

option	
avail	show available environment modules
list	show loaded environment modules
load	load specified environment module(s)
unload	Unload specified environment module(s)
switch	load and unload
purge	unload all the current modules

If you use Python3:

```
$ module load python/3.7.3
```

Environment modules (2)

Load a previous version of Intel compiler

```
[tUVXYZ@obcx01 ~]$ module load  intel/2020.1.217 Return
[tUVXYZ@obcx04 ~]$ module list Return
Currently Loaded Modulefiles:
  1) impi/2019.7.217    2) intel/2020.1.217
```

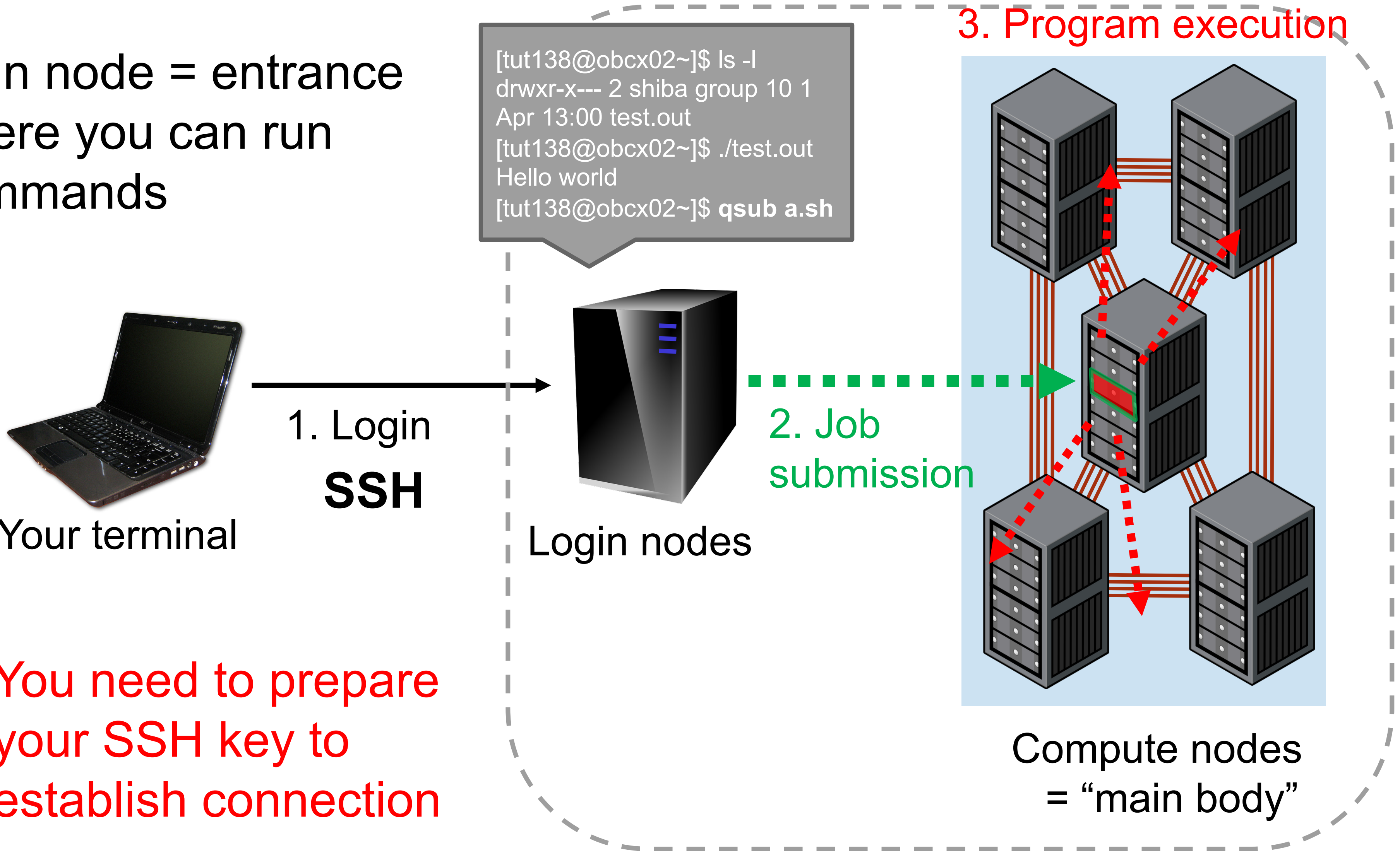
An appropriate Intel MPI library is also loaded

Load newer Python3 (system default is Python 2.7.5)

```
[tUVXYZ@obcx01 ~]$ module load python/3.7.3 Return
[tUVXYZ@obcx01 ~]$ module list Return
Currently Loaded Modulefiles:
  1) impi/2019.7.217    2) intel/2020.1.217   3) python/3.7.3
```

How to use supercomputer

login node = entrance where you can run commands



```
[tut138@obcx02~]$ ls -l
drwxr-x--- 2 shiba group 10 1
Apr 13:00 test.out
[tut138@obcx02~]$ ./test.out
Hello world
[tut138@obcx02~]$ qsub a.sh
```

You need to prepare your SSH key to establish connection

Job script = work instructions for supercomputers

Write a script to let your program run on compute nodes.
It is forbidden to run your program (binary) on the login node after compilation.
— login nodes are shared with other users, so should not be under load

C, Fortran (fibonacci.sh)

```
#!/bin/bash
#PJM -L rscgrp=tutorial     [resource group]
#PJM -L node=1              [number of nodes]
#PJM -L elapse=0:01:00      [ maximum time]
#PJM -g gt00                [group name]
#PJM -N fibonacci           [job name]
#PJM -o stdout.txt          [stdout file]
#PJM -j                     [merge error into stdout]

module purge
module load intel/2020.1.217
./fibonacci.out
```

Python (fibonacci.sh)

```
#!/bin/bash
#PJM -L rscgrp=tutorial
#PJM -L node=1
#PJM -L elapse=0:01:00
#PJM -g gt00
#PJM -N fibonacci
#PJM -o stdout.txt
#PJM -j

module load python/3.7.3
python ./fibonacci.py
```

We have already arranged your job scripts

Copy a job script we have prepared in advance.

```
[tUVXYZ@obcx01 ~]$ cd          Return
[tUVXYZ@obcx01 ~]$ cd fibonacci     Return
```

C, Fortran

```
[tUVXYZ@obcx01 fibonacci]$ cp /work/gt00/share/z30122/fibo_c/fibonacci.sh .
```

Return

Python

```
[tUVXYZ@obcx01 fibonacci]$ cp /work/gt00/share/z30122/fibo_python/fibonacci.sh .
```

Return

In case you could not edit the program in time

You may copy the one we have prepared in advance.

C

```
[tUVXYZ@obcx01 fibonacci]$ cp /work/gt00/share/z30122/fibo_c/fibonacci.c .
```

Return

Fortran

```
[tUVXYZ@obcx01 fibonacci]$ cp /work/gt00/share/z30122/fibo_fortran/fibonacci.f90 .
```

Return

Python

```
[tUVXYZ@obcx01 fibonacci]$ cp /work/gt00/share/z30122/fibo_python/fibonacci.py .
```

Return

Compile the program

If you use C or Fortran
→ Compile the code to generate an executable (binary) file.

C

```
[tUVXYZ@obcx01 fibonacci]$ icc fibonacci.c -o fibonacci.out
```

Return

$\underbrace{\texttt{fibonacci.c}}_{\text{program}}$ $\underbrace{\texttt{fibonacci.out}}_{\text{executable}}$

Fortran

```
[tUVXYZ@obcx01 fibonacci]$ ifort fibonacci.f90 -o fibonacci.out
```

Return

Compilers

Intel compiler / GNU compilers are available

Source code type		lang	command
Serial codes	Intel	C	icc
		Fortran	ifort
	GNU	C	gcc
		Fortran	gfortran
MPI parallel codes	Intel	C	mpiicc
		Fortran	mpiifort
	GNU	C	mpicc
		Fortran	mpif77 or mpif90

Ex 1) Compiling a C code by Intel compiler

```
$ icc [option]  “source.c” –o “exe.out”
```

Ex 2) Compiling a Fortran90/95 code by GNU compiler

```
$ gfortran [option] -free-line-length-none “source.f90” –o “exe.out”
```

-free-line-length-none : specifying Fortran free form

Compiler option examples

	Language	Intel option	GNU option	
parallelization	C/Fortran	-qopenmp	-fopenmp	Enable OpenMP
debugging	C/Fortran	-g		Embed debugging information into the executables
		-Wall		Display all warnings (potential bugs)
		-traceback	-fbacktrace	Trace where an error happens in running the executable
	Fortran	-check bounds	-fbounds-check	Detect whether a variable is within bounds
optimization	C/Fortran	-O0、-O1、-O2、-O3		Optimization (larger values are more aggressive)
		-xHost	-march=native	CPU-architechture-specific optimization

Running your jobs on OBCX

You cannot run your code from /home directory on OBCX

→ Copy your program into your /work directory

```
                                    Return
[tUVXYZ@obcx01 ~]$ cd               Return
[tUVXYZ@obcx01 ~]$ ls
fibonacci                                                           Return
[tUVXYZ@obcx01 ~]$ cp -r fibonacci  /work/gt00/tUVXYZ/
                                                                Return
[tUVXYZ@obcx01 ~]$ cd  /work/gt00/tUVXYZ/fibonacci
```

Submit the job script, then your program will be run on the compute nodes.

```
[tUVXYZ@obcx01 fibonacci]$ pjsub fibonacci.sh    Return
```

Check you job status & display the results

Confirm your jobs that are now running.
（Perhaps nothing will be displayed if they terminate at once).

```
[tUVXYZ@obcx01 fibonacci]$ pjstat
```

Display the results.

```
[tUVXYZ@obcx01 fibonacci]$ more result_fibo.txt
```

[**Return** will let you go downwards]

Search 747031 in the results.

```
[tUVXYZ@obcx01 fibonacci]$ grep 747031 result_fibo.txt
```

How to manage your job – scheduler commands

command	role	how to use
pjsub	submit a job	$pjsub “script.sh”
pjdel	delete a job	$pjdel “job ID”
pjstat	quaery job status	$pjstat

Caution) these commands are specific to Fujitsu Supercomputers

pjstat options

- -H : confirm jobs that have finished
- --rsc -b : query job “congestion” status in each resource group
- --rsc -x : query the number of nodes (resources) a user can request in a job
- --nodeuse : Check the current resource usage over the whole OBCX system

On your trial account for this workshop

You can use this account for one month.
"lecture" resource group is available after the workshop.
"tutorial" is available only today (13-17 pm).

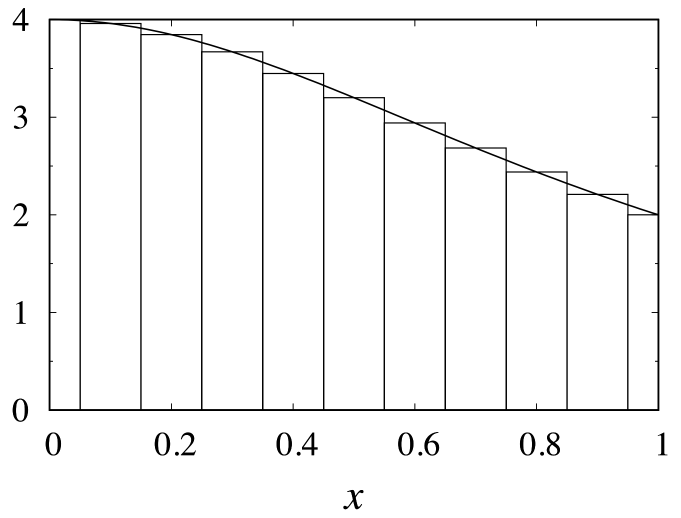
Job script
fibonacci.sh

```
#!/bin/bash
#PJM  -L  rscgrp=lecture        [resource group]
#PJM  -L  node=1
#PJM  -L elapse=0:01:00
#PJM  -g  gt00
#PJM  -N  fibonacci
#PJM  -o   stdout.txt
#PJM  -j

module purge
module load intel/2020.1.217
./fibonacci.out
```

Exercise: parallel computation of π

$$I = \int_0^1 \frac{4}{1+x^2}dx$$

$$= \pi$$



quadrature by parts

↓

sub-billion meshes

We will see parallelization speedup.

We prepared 3 sample programs: C, Fortran, Python

Exercise: measure execution time for various degree of parallelism

(understanding of MPI is not our focus now, just run !)

C source codes

pi.c [serial]

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

int main(void) {
  int i;
  int ndiv = 10000000;


  double width = 1.0 / (double)ndiv;
  printf("width = %.15f\n", width);
  double sum = 0.0;
  double x = 0.0;


  for (i=0; i<ndiv; i++) {
    x = (i+0.5)*width;
    sum += width * 4.0 / (1.0 + x*x);
  }


  printf("PI = %.15f\n", sum);
  return 0;
}
```

pi_mpi.c [parallel]

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

int main(int argc, char* argv[]) {
   int ndiv = 5600000000;
   int ierr,  myrank, nprocs;
   int ndiv_local, i;
   double x, width, sum, total_sum;
   double t1, t2;

   width = 1.0 / (double)ndiv;

   ierr = MPI_Init(&argc, &argv);
   ierr = MPI_Comm_rank(MPI_COMM_WORLD,
                                       &myrank);
  ierr = MPI_Comm_size(MPI_COMM_WORLD,
                                       &nprocs);

  t1 = MPI_Wtime();

  sum = 0.0;
  ndiv_local = ndiv / nprocs;

  for (i = myrank*ndiv_local;
         i < (myrank+1)*ndiv_local; i++) {
         x = (i + 0.5) * width;
     sum = sum + width * 4.0 / (1.0 + x*x);
  }

  MPI_Reduce(&sum, &total_sum, 1, MPI_DOUBLE,
                     MPI_SUM, 0, MPI_COMM_WORLD);

  t2 = MPI_Wtime();

  if (myrank == 0) {
    printf("PI(MPI) = %.18f\n", total_sum);
    printf("Number of cores utilized = %d\n", nprocs);
    printf("Execution time = %.8f (sec.)\n", t2 - t1);
  }

  ierr = MPI_Finalize();

  return 0;
}
```

compile

```
[tUVXYZ@obcx01 calc_pi_mpi]$ mpiicc pi_mpi.c -o pi_mpi.out
```

Fortran source codes

pi.f90 [serial]

```
program main

  implicit none

  integer i, ndiv
  real(kind=8) unit, width, sum, x

  ndiv = 560000000
  width = 1.0 / dble(ndiv)

  print *, "width"
  print '(F18.14)', width

  sum = 0.0d0
  x = 0.0d0

  do i = 1, ndiv
    x = ( dble(i-1) + 0.5) * width
    sum += width * 4.0 / (1.0 + x*x)
  end do

  print *, "sum"
  print '(F18.14)', sum

end program main
```

pi_mpi.f90 [parallel]

"_8" indicates 8-byte integer (larger # of digits)

```
program main

  Use mpi
  implicit none

  integer ndiv, ierr, myrank, nprocs
  integer ndiv_local, i;
  real(kind=8) unit, width, sum, x, total_sum
  real(kind=8) t1, t2

  ndiv = 5600000000_8
  width = 1.0 / dble(ndiv)

  call MPI_INIT(ierr)
  call MPI_COMM_RANK(MPI_COMM_WORLD,
                                         myrank, ierr)
  call MPI_COMM_SIZE(MPI_COMM_WORLD,
                                       nprocs, ierr)

  sum = 0.0d0
  ndiv_local = ndiv / nprocs

  t1 = MPI_Wtime()

  do i=myrank*ndiv_local+1, (myrank+1)*ndiv_local
    x = ( dble(i-1) + 0.5)*width
    sum = sum + width * 4.0 / (1.0 + x*x)
  end do

  call MPI_REDUCE(sum, total_sum, 1, MPI_REAL8,
MPI_SUM, 0, MPI_COMM_WORLD, ierr)

  t2 = MPI_Wtime()

  if (myrank .eq. 0) then
    print "('PI(MPI) = ', F18.16)", total_sum
    print "('Number of cores utilized = ', i0)", nprocs
    print "('Execution time = ', F12.8)", t2 - t1
  endif

  call MPI_FINALIZE(ierr)

end program main
```

Compile

```
[tUVXYZ@obcx01 calc_pi_fortran]$ mpiifort pi_mpi.f90 -o pi_mpi.out
```

Python source codes

pi.py [serial]

```
program main
  implicit none

  integer i, ndiv
  real(kind=8) unit, width, sum, x

  ndiv = 560000000
  width = 1.0 / dble(ndiv)

  print *, "width"
  print '(F18.14)', width

  sum = 0.0
  x = 0.0

  do i = 1, ndiv
    x = ( dble(i-1) + 0.5) * width
    sum = sum + width * 4.0 / (1.0 + x*x)
  end do

  print *, "sum"
  print '(F18.14)', sum

end program main
```

pi_mpi.py [parallel]

```
from mpi4py import MPI
import numpy

comm = MPI.COMM_WORLD
myrank = comm.Get_rank();
nprocs = comm.Get_size();

ndiv = 560000000
width = 1.0/ndiv

t1 = MPI.Wtime()

sum = numpy.zeros(1)
total_sum = numpy.zeros(1)
ndiv_local = ndiv // nprocs

for i in range (myrank*ndiv_local, (myrank+1)*ndiv_local):
    x = width * (i+0.5)
    sum[0] = sum[0] + width * 4.0 / (1.0 + x*x)

comm.Reduce([sum, MPI.DOUBLE], total_sum, op=MPI.SUM, root=0)

t2 = MPI.Wtime()

if comm.rank == 0:
    print("PI(MPI) = ", total_sum[0])
    print("Number of cores utilized = ", nprocs)
    print("Exectution time = ", t2 - t1, " (sec.)")
```

Copy the codes into your own “/work” directory

```
[tUVXYZ@obcx01 ~]$ cd /work/gt00/tUVXYZ
[tUVXYZ@obcx01 tUVXYZ]$ mkdir calc_pi_mpi
[tUVXYZ@obcx01 tUVXYZ]$ cd calc_pi_mpi
[tUVXYZ@obcx01 calc_pi_mpi]$ pwd
/work/gt00/tUVXYZ/calc_pi_mpi
```

Fortran

```
$ cp /work/gt00/share/z30122/pi_fortran_mpi/* .
```

asterisk = wild card (everything)

C

```
$ cp /work/gt00/share/z30122/pi_c_mpi/* .
```

Python

```
$ cp /work/gt00/share/z30122/pi_python_mpi/* .
```

Compile parallel programs

For C & Fortran, we need to compile the source code.

C

```
[tUVXYZ@obcx01 calc_pi_mpi]$ mpiicc pi_mpi.c -o pi_mpi.out
```

Fortran

```
[tUVXYZ@obcx01 calc_pi_mpi]$ mpiifort pi_mpi.f90 -o pi_mpi.out
```

For Python users

You need to set up in advance

```
[tUVXYZ@obcx01 ~]$ cd /work/gt00/tUVXYZ/calc_pi_mpi
[tUVXYZ@obcx01 calc_pi_mpi]$ emacs setenv.sh    (or vim setenv.sh)
```

Change the following:

```
export PYTHONUSERBASE=/work/gt00/tUVXYZ/.local
                                      ⇩
                                 your own user ID
```

Run setup.sh   (install numpy and mpi4py)

```
[tUVXYZ@obcx01 ~]$ ./setup.sh
```

Submit parallel jobs

We prepared job scripts with various degrees of parallelism,
to see performance increase.

```
[tUVXYZ@obcx01 calc_pi_mpi]$ pjsub run_n1c0004.sh
[tUVXYZ@obcx01 calc_pi_mpi]$ pjsub run_n1c0028.sh
[tUVXYZ@obcx01 calc_pi_mpi]$ pjsub run_n1c0056.sh
[tUVXYZ@obcx01 calc_pi_mpi]$ pjsub run_n2c0112.sh
[tUVXYZ@obcx01 calc_pi_mpi]$ pjsub run_n4c0224.sh
[tUVXYZ@obcx01 calc_pi_mpi]$ pjsub run_n8c0448.sh
```

- 1 node, 4 cores
- 1 node, 28 cores
- 1 node, 56 cores
- 2 nodes, 112 cores
- 4 nodes, 224 cores
- 8 nodes, 448 cores

Confirm the job status

Check your jobs and wait until all your jobs finish.

```
[tUVXYZ@obcx01 calc_pi_mpi]$ pjstat
```

You will get the results after the jobs finish.

```
[tUVXYZ@obcx01 calc_pi_mpi]$ ls
```

```
pi_mpi.c                result_n2c0112.txt    run_n1c0056.sh
pi_mpi.out              result_n4c0224.txt    run_n2c0112.sh
result_n1c0004.txt      result_n4c0448.txt    run_n4c0224.sh
result_n1c0028.txt      run_n1c0004.sh        run_n8c0448.sh
result_n1c0056.txt      run_n1c0028.sh
```

“result_n*c****.txt” are output files wherein the results are stored.

Confirm the results and compare execution times

You will find elapse times of your jobs in the output files.
Let's see the beginning part of each output file.

```
[tUVXYZ@obcx01 calc_pi_mpi]$ head result*.txt
```

We find that the execution time is shorter with smaller number of nodes.

```
==> result_n1c0004.txt <==
PI(MPI) = 3.141592653589913464
Number of cores utilized = 4
Execution time = **.******** (sec.)

==> result_n1c0028.txt <==
PI(MPI) = 3.141592653589770912
Number of cores utilized = 28
Execution time = **.******** (sec.)

==> result_n1c0056.txt <==
PI(MPI) = 3.141592653589800221
Number of cores utilized = 56
Execution time = **.******** (sec.)

==> result_n2c0112.txt <==
PI(MPI) = 3.141592653589794892
Number of cores utilized = 112
Execution time = **.******** (sec.)

==> result_n4c0224.txt <==
PI(MPI) = 3.141592653589791340
Number of cores utilized = 224
Execution time = **.******** (sec.)

==> result_n8c0448.txt <==
PI(MPI) = 3.141592653589797557
Number of cores utilized = 448
Execution time = **.******** (sec.)
```

If you want to “log in” to the computation node…

Interactive job

We can run a job on the compute nodes as if we were working in the bash shell on them.

```
[tUVXYZ@obcx04 tUVXYZ]$ pjsub --interact -g gt00 -L rscgrp=interactive,node=1
[INFO] PJM 0000 pjsub Job 517079 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 517079 started.
[tUVXYZ@cx0065 tUVXYZ]$
```

You are “virtually” logging into a compute node (cx0065) via the login node (node04).

Compute nodes are not shared with other users but are for your exclusive use. You may run any jobs you like in the interactive job.

Finally: transfer your simulation data to your laptop

Log out from supercomputer

```
[tUVXYZ@obcx01 ****]$ exit
Mac-mini:~ shiba$
```

Transfer a file using “sftp”

```
Mac-mini:~ shiba$ sftp tUVXYZ@obcx.cc.u-tokyo.ac.jp
sftp > cd /work/gt00/tUVXYZ/fibo
sftp > get fibonacci.txt
Fetching /work/00/gt00/tUVXYZ/fibonacci.txt fibonacci.txt
/work/00/gt00/tUVXYZ/fibonacci.txt      100%  171    11.9KB/s   00:00

sftp > exit
Mac-mini:~ shiba$ ls
fibonacci.txt
```