**Руководство пользователя ЦОД ПИК**

1. **Терминальный доступ**

Терминальный доступ к ЦОД ПИК производится по протоколу SSH на адрес

**login.pnpi.nw.ru**

Для соединения можно использовать SSH-клиенты, поддерживающие протокол SSH версии 2. Подойдут, например:

* [PuTTY](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) (Windows);
* [OpenSSH](http://www.openssh.org/) (UNIX, входит в комплект практически всех современных UNIX-подобных операционных систем).

Для копирования файлов можно использовать следующие утилиты:

* [WinSCP](http://winscp.net/) (Windows, графический интерфейс наподобие Explorer);
* scp (UNIX, входит в OpenSSH);
* pscp.exe/psftp.exe (Windows, текстовой интерфейс, входит в PuTTY).

1. **Создание ключей SSH**

Для аутентификации на ЦОД ПИК в настоящий момент используются SSH-ключи. Вкратце, SSH-ключи – это пара файлов. Один из них называется закрытым (приватным) ключом, обычно защищен паролем и не показывается никому. Второй файл называется открытым (публичным) ключом и может быть показан кому угодно, более того, его необходимо зарегистрировать на нашем сервере, чтобы он начал вас пускать. Открытый и закрытый ключи всегда образуют пару, но восстановить один из другого за обозримое время невозможно, так что в случае утери любого из них, пару придётся создавать заново.

Имея только открытый ключ, сервер может проверить, есть ли у вас соответствующий ему закрытый ключ. А поскольку предполагается, что закрытый ключ есть только у вас (именно поэтому его так важно никому не показывать), то сервер может утверждать, что если проверка прошла успешно, то владелец закрытого ключа – именно вы.

* 1. **Создание ключей SSH на Unix-системах**

Предполагается, что у вас установлен пакет OpenSSH, в котором есть необходимая утилита ssh-keygen. Запускаем её:

**$ ssh-keygen -t rsa -b 2048**

*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 public key has been saved in /home/user/.ssh/id\_rsa.pub.*

*The key fingerprint is:*

*22:f7:7b:96:c5:4b:b2:0d:2d:9f:5f:67:57:48:46:37 user*[AT]*host.net*

Соответственно, если вы хотите сохранить ваш закрытый ключ не в /home/user/.ssh/id\_rsa, а куда-то еще, вы можете ввести полный путь в ответ на приглашение «Enter file in which to save the key». Если вы не уверены, нужно ли это, то лучше оставить все как есть.

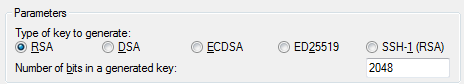
После завершения работы ssh-keygen закрытый ключ будет находиться в файле /home/user/.ssh/id\_rsa, а открытый в /home/user/.ssh/id\_rsa.pub. Именно последний файл с расширением .pub нужно отослать нам по электронной почте. Пожалуйста, не перепутайте!

Если вы используете нестандартное имя файла с закрытым ключом, то вам либо придется указывать его расположение каждый раз с помощью ключа -i, либо нужно добавить в файл ~/.ssh/config следующие строчки:

Host login.pnpi.nw.ru

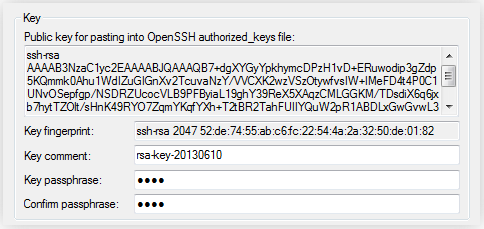
IdentityFile ~/.ssh/id\_rsa\_pik

* 1. **Создание ключей SSH на Windows-системах**

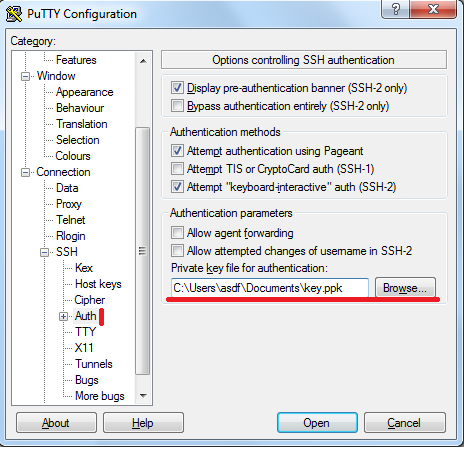
Для создания SSH-ключа вам нужно использовать программу PuTTYgen из комплекта PuTTY. Запускаем программу и выбираем тип ключа «RSA» с длиной 2048 бит:

Нажимаем на кнопку «Generate».

Далее PuTTYgen создает ключ и предлагает ввести для него пароль:

  
Пароль нужно выбирать такой, чтобы злоумышленник его не подобрал, а вы – не забыли. Также можно ввести комментарий (любой).

Сохраняем private key (закрытый ключ) в файл mykey.ppk, а public key (открытый ключ) в файл id\_rsa.pub.

id\_rsa.pub нужно послать нам, а путь к закрытому ключу нужно настроить в PuTTY, чтоб он знал, откуда его брать:

* 1. **Перенос ключей SSH между системами**

При переносе SSH-ключей между различными операционными системами иногда возникает путаница, при этом основные различия имеются между форматами закрытых ключей. Ниже будет показано, как отличить файлы с ключами различных форматов.

Вот как должен выглядеть закрытый ключ в формате OpenSSH:

-----BEGIN RSA PRIVATE KEY-----

Proc-Type: 4,ENCRYPTED

DEK-Info: DES-EDE3-CBC,F45627B370E24DB1

XD4u/JN7snja3Wl1Dj7eoIGKiAx8IOALvLbmUTik4liCMxk3OPW1eOTQxAV7CJom

PMfH5kOMAber45D4yiQzGYGc/mxStfNjtm8V7j6N4acb6E4BKRUGmv7ZkLrw9csr

PpZzSQ5wTYiXRzJE55r5Q+CWd2V5qVJypAW2Q8pUMGF6A+DSxno1LSc8TwQ43o8G

FTGw8bCEgnfhXmGGLm6lirbXOQIJRCqbszBlPGZ4uNNaLAkbs7O85dqjKFsTMXLv

XCloDn+8wFr8eg7mZAkAjs0jz9l0PBDAYxivzrw97GRmemHynqsDSaE4F6pxaed9

lhxLwtK0VZH1ofEZUdT64lHJUX4oow9t2r7ajrItlnGBo7redY4yTaLKArxSOdVG

AKOO0PyAv4J2Mm0F8bNwYecUskpu+efr4n2343jJKmXONc+KxfZjdDBFsVFkmpv3

p4AyryKisiT/BbgAdLYXomVKeMSBQe6eDUSQtIabirOBsk63aiJAy5j5yzqzudJ8

IzmCuNB7p4SkmqiPoqW7vPR5G2PlK4G790vszcYr221w5QE3keVB/iRszyJrQHl8

k9LzX3oejumGknLubihJqelWBDP54tGlL88YB9uPQ4jSUFOv8+M68ebBONVh1eB9

uUe0C3FwxUHm53i7v8DnDFki8onnk6mUxa8pgIJJBUMdDnie0CGh1QAT6NiLE2tT

mnBIG7pHxU3HbnUdA+yDbvvfvkoP6Lc5XWCQsn8SNjDv2gPkNrYRbxaRjgOykTiQ

ZHEEU+JYwzyOtw6tbuOOUfcH4BqKhVc0YwZrDDKHOXAo0PgGcuxoE8EAKZptV3lz

XyFZjtfw5KvulRmwjuMydu0Alg5qo8cpyCSFCyxRQlWjyiPwOI3w7ixLpZFpQGMQ

bmVrgLKB/XdgPnmXj7K/6KD2YU2FxZjCFdbGdDUY6E/cBAHD/7sHjGV6CXJ72ZkO

oGWhLgkRk/Dy9doysm6DwCiLS7K/cddUkZcFKvxzBdmOaTt+jlB2tXKDvRAJIwrm

GiEx2LlKjbbgoTrV+rjuFFgVhsHualxP52NsvujQZVpeFtomZ/amk3ceOMTFTkab

QJcb/zOWjG+PrtiQ6BR/Te0kl44S2L3AR5AOCVD13k1nEOZ1yHyCtti04xM7JavP

Jy+RSUmIt7hSD9A0e4nHHXEhPZnGgG8ekVrR6FEQ+0FbvYLpv05Ir+igQSMftZwA

YdBjA16KeJL4jKAOWzVe5tdA0BQcJvPjzPK97N6HkCrbcmSy7mQAsXCZ7BcInwwa

UD8zOeH8Ii8atzy+YM+bQRoRfQLkzpJ3pVe68ZwqKFHl+YQNlh0sCJ3slihLKKVR

mUXutpa2c385Yb9djLifKaSdPYG3rutmdx7HY1JzYvvkIau+ixiO1H6dETI8tLZC

4l1FTSisGt4LZyH3WgPpQzhiWs0KX9yIQ/0lqRhgEL8zqqDm3jo/jIQdFMVfHBlI

YSKjoySUN3GXk8MfBsxxgbJRNwfcdiuB5qcsAxYkNVJgczHScEoM5NoatlHVlbyV

Ymu8j7BqNZ5a1W/YYadEV2prdQeAUOTX8yGVq14MU/5X6uTZyCj9fQ==

-----END RSA PRIVATE KEY-----

Детали могут отличаться, но первая и последняя строчки почти всегда имеют вид

-----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----

либо

-----BEGIN DSA PRIVATE KEY-----

-----END DSA PRIVATE KEY-----

Если вы хотите использовать такой ключ в PuTTY, то вам необходимо его преобразовать в другой формат[. Без преобразования формата, Pu](http://computing.kiae.ru/user-support/user-manual.html#ssh-keys.unix-to-windows)TTY не будет правильно распознавать ваш ключ.

Вот как выглядит ключ в формате PuTTY:

PuTTY-User-Key-File-2: ssh-rsa

Encryption: aes256-cbc

Comment: rsa-key-20130610

Public-Lines: 6

AAAAB3NzaC1yc2EAAAABJQAAAQB7+dgXYGyYpkhymcDPzH1vD+ERuwodip3gZdp5

KQmmk0Ahu1WdIZuGIGnXv2TcuvaNzY/VVCXK2wzVSzOtywfvsIW+IMeFD4t4P0C1

UNvOSepfgp/NSDRZUcocVLB9PFByiaL19ghY39ReX5XAqzCMLGGKM/TDsdiX6q6j

xb7hytTZOlt/sHnK49RYO7ZqmYKqfYXh+T2tBR2TahFUIIYQuW2pR1ABDLxGwGvw

L3T2laIT5Lt3//6ioCT0jlIkBuVii1hlm2quiyqogDs1U1ihOGL0wKNvhpds4Hfd

wyRCjU1WJkZGGrLkdtvlg4AQhM7J8FVrYhTNZe90GJZXjnJP

Private-Lines: 14

Xp4pdSAaE8Ftj9Va68OgwyTtO1V75m7cLmSHJSxzVaQCpw8zzn2gPeFnl/7fuusN

Q1cZ8uDdpzI5E2+iEtM5hSn4SoOfH5eyhSa+FYObxj2IN+ylwS4vIbPmj7pY4Mom

BGvxPdgZzKSITblxiE6YSFCNq4CmTIjtg/OXU5/slw7oifdEPcvPDudaGtztmv4a

zSMPO9tvgrk1C5hzY5FfDoKpZHK94mgyjypHt7NEAbEls7GQCn7qTsng2Au2xeu/

l5ZDmb52zu/iLYb2JALnz+h5qyE2XFHsVjwyY+rzxstEWgkgCZdCrbOexHrgIlyc

sKr7H3c3pJ7ZsVsXFvoWLVRcFhq/8Sa7rTr7ONkaUBPUiF00fdp2UiC3bI5U9FlQ

MRhXKEfogEdJDZdWvGibHyX2on8PkJzDZCuQaG/K6pm92VHhv8AYMs4ADfrUVg6k

Y7A5q+sDtXM1QAiL/Qct1Hr8uAp8sLfKYsnBSc1teTHdIjugEjfOn8zMR96vgFEi

bXWcA8o4WROMJ42f9Dy3hTsg7kf4ZcGmY8ua3deYABu3KuAttUNTshAF5qfPGjEk

gR53PoPgLzA1IeYIgX9w9INHQgfolsWguuTb5i1UdXKnk7SwXxmyl1o9F3DSRGu9

pl7bKLQGjrDcaLyxavUyZoHb1zwiDoGBJ4pefC1e1LKJ6OUiWXCWcUsTIeuj9q+L

eEr4RWr59A/11jGvbrTi2Qjxp71aqHmQjsiXonT/bl6Xydb3zS8e4AmJJifxu179

EgfXBPePYarjX6PPivt6uURnRDyTVU6jXJl4ddIudtpUpY14M4BekA4MZ9m2PMsx

S5syWlWFvfPIoHMLpdDWV/D2kimvamcEbbCxK0uI9VlVHNmTotI49OYt0XaHrZGS

Private-MAC: 464d517a6c7b61184a4f7f1ec2ad4f33c1b15294

Характерными особенностями данного формата является первая строка вида

PuTTY-User-Key-File-2: ssh-rsa

вслед за которой довольно часто следуют строки вида

Encryption: aes256-cbc

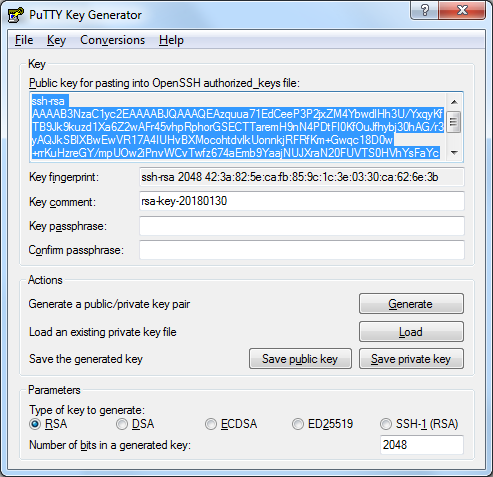
Comment: rsa-key-20130610

Если вы хотите использовать такой ключ в OpenSSH, то вам необходимо его преобразовать в другой формат[.](http://computing.kiae.ru/user-support/user-manual.html#ssh-keys.unix-to-windows) [Без преобразования формата, OpenSSH не будет правильно распознавать ваш ключ.](http://computing.kiae.ru/user-support/user-manual.html#ssh-keys.windows-to-unix)

* + 1. **Преобразование ключей из формата PuTTY в OpenSSH**

Для преобразования ключа вам нужно использовать программу PuTTYgen.

**Шаг 1**: вы загружаете существующий закрытый ключ в PuTTYgen (меню «File», пункт «Load private key»). Получается такое окно:

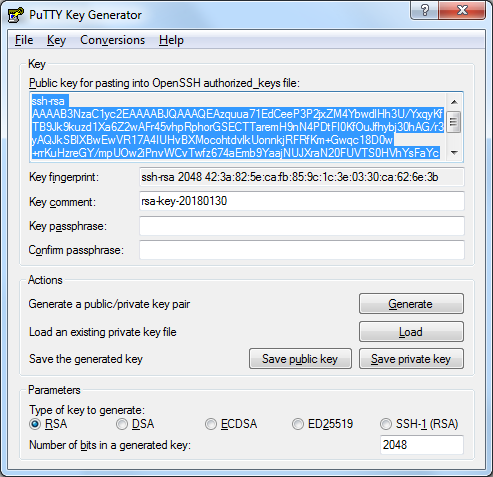
  
**Шаг 2**: находим в меню «Conversions» пункт «Export OpenSSH key», вводим имя файла и выбираем каталог, в который будет сохранен закрытый ключ. Пароль сохранённого закрытого ключа будет таким же, как и пароль исходного ключа.

**Шаг 3**: переносим закрытый ключ на Unix-машину. Крайне желательно делать это с использованием физического носителя, а не через сеть или электронную почту.

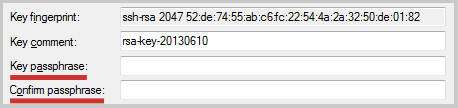
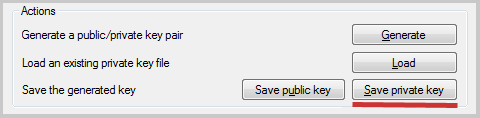
* + 1. **Преобразование ключей из формата OpenSSH в PuTTY**

Для преобразования ключа вам снова нужно использовать программу PuTTYgen.

**Шаг 1**: находим в меню «Conversions» пункт «Import key», выбираем файл с ключом в формате OpenSSH, вводим в открывшемся окне пароль для этого закрытого ключа и получаем такое окно:

****

**Шаг 2**: нажимаем в этом окне кнопку «Save private key»,

  
после чего выбираем имя файла и каталог, куда будет сохранен файл закрытого ключа. Пароль сохранённого закрытого ключа будет совпадать с паролем исходного ключа. Если вы хотите его изменить, то перед сохранением ключа в формат PuTTY необходимо ввести новый пароль в поля «Key passphrase» и «Confirm passphrase»:

1. **Программное окружение пользователя**

На ресурсах ЦОД ПИК используется высокопроизводительная параллельная файловая система, доступная на всех вычислительных узлах. Домашний каталог пользователя расположен в каталоге /home/$USER. Общее прикладное программное обеспечение и данные исследовательских групп, при необходимости, разворачиваются в каталогах /share/sw и /share/data соответственно.

### Компиляторы

* GCC 14.2.0 – модуль sw/gcc-14.2
* GCC 13.3.0 – модуль sw/gcc-13.3
* GCC 12.4.0 – модуль sw/gcc-12.4
* GCC 11.5.0 – модуль sw/gcc-11.5
* GCC 10.5.0 – модуль sw/gcc-10.5
* GCC 4.8.5 – системный, не требует загрузки модулей
* Intel Compiler – модуль intel/compiler

### Библиотеки MPI

* Open MPI 4.1.7 – модуль mpi/ompi-4.1
* Open MPI 4.0.7 – модуль mpi/ompi-4.0
* Open MPI 3.1.6 – модуль mpi/ompi-3.1
* Open MPI 3.0.6 – модуль mpi/ompi-3.0
* Open MPI 2.1.6 – модуль mpi/ompi-2.1
* Platform MPI 9.1 – модуль mpi/pmpi
* Intel MPI – модуль intel/mpi

Все версии библиотек Open MPI собраны компилятором GCC 14.2 и при загрузке автоматически подгружают его модуль.

### Вычислительные библиотеки

* FFTW 3.3.10 – модуль sw/fftw-3.3 (подгружает модуль mpi/ompi-4.1)
* Intel MKL – модуль intel/mkl
* Intel DNNL – модуль intel/dnnl

**Настройка программного окружения**

В связи с тем, что пользователи ЦОД ПИК могут использовать различные комбинации установленного программного обеспечения, у нас используется система настройки программного окружения, называемая environment-modules. Для того, чтобы настроить рабочее окружение, выполните команду module load <модуль>.

Чтобы воспользоваться компиляторами Intel, выполните команду:

**$ module load intel/compilers**

Чтобы скомпилировать параллельную задачу с использованием библиотеки Platform MPI, выполните:

**$ module load mpi/pmpi**

Всё это можно сделать и одной командой:

**$ module load intel/compilers mpi/pmpi**

Следующая команда позволяет узнать какие модули имеются в системе:

**$ module avail**

Посмотреть список уже используемых в данном сеансе модулей можно следующей командой:

**$ module list**

Очистить все используемые в данном сеансе модули можно одной командой:

**$ module purge**

Более подробную информацию о системе environment-modules можно узнать, запустив команду man module. Для того, чтобы не приходилось настраивать окружение каждый раз при начале сеанса, необходимые команды можно поместить в файл .bash\_profile в вашем домашнем каталоге.

1. **Запуск пакетных заданий**

Запуск пакетных заданий осуществляется командой sbatch. Подробную документацию можно посмотреть командами man sbatch и man srun, здесь же будет рассмотрена только базовая функциональность.

* 1. **Запуск сценария одиночной задачи**

**Важно:** система управления пакетными заданиями умеет запускать только сценарии, поэтому не пытайтесь вместо сценариев запустить исполняемый файл – из этого, скорее всего, ничего не выйдет.

Например, пусть мы хотим запустить на выполнение сценарий test.sh, находящийся в текущей директории и содержащий следующие строчки:

*#!/bin/sh*

*#SBATCH -o %j.out*

*#SBATCH -e %j.err*

*#SBATCH -t 01:00:00*

*#SBATCH -p compute*

*hostname*

*df*

*date*

*sleep 10*

*date*

Этот сценарий должен быть сделан исполняемым:

**$ chmod +x test.sh**

Запускаем сценарий на выполнение:

**$ sbatch test.sh**

*sbatch: Submitted batch job 19*

В ответ мы получили идентификатор нашей задачи в очереди — 19. Пользуясь этим идентификатором, мы можем узнать текущий статус задачи:

**$ squeue -j 19**

Удалить задачу из очереди можно следующей командой:

**$ scancel 19**

После успешного завершения нашей задачи в домашнем каталоге пользователя создадутся файлы 19.out и 19.err, в которых будет записано содержимое потоков стандартного вывода и стандартной ошибки.

Файлы будут называться 19.out и 19.err поскольку мы написали

*#SBATCH -o %j.out*

*#SBATCH -e %j.err*

В этих директивах на месте макроса «%j» подставляется идентификатор задачи.

Директива

*#SBATCH -t 01:00:00*

указывает на максимальное время выполнения задачи (в данном случае — один час). Если вы можете более-менее точно оценить верхнюю границу этого времени, пожалуйста, указывайте её: это позволит планировщику более разумно распределять ресурсы. Однако вы должны понимать, что ваша задача будет «убита», если время её выполнения превысит указанное, поэтому максимальное время выполнения лучше указывать с некоторым запасом. Если время не указать совсем, то для задачи будет действовать ограничение по умолчанию, равное 24 часам. Максимально возможное время выполнения одной задачи – 90 дней.

Директива

*#SBATCH -p compute*

указывает, что мы хотим запустить задачу в очередь «compute». Чтобы понять, какую очередь нужно использовать для ваших нужд, пожалуйста, обратитесь к разделу «Список очередей и их назначение».

В случае, если вы запускаете многопоточную программу, способную занимать несколько ядер процессора, то в сценарий необходимо добавить директивы, указывающие требуемое количество вычислительных ядер и оперативной памяти:

*#SBATCH -n 28*

*#SBATCH –mem=1G*

В данном случае мы указываем, что программе требуется 28 ядер – это максимальное количество, доступное в очереди «compute», и 1 гигабайт оперативной памяти. Если не указывать требуемое количество памяти, то программе будет выделена вся доступная на узле память, что не позволит запуститься на этом же узле другим программам пользователя и может привести к неоптимальному использованию ресурсов.

Все вышеприведённые директивы также могут быть переданы команде sbatch через параметры командной строки.

* 1. **Запуск сценария параллельной задачи**

В сценарии запуска задачи окружение должно настраиваться точно так же, как и как в файле .bash\_profile. Будем предполагать, что вы используете библиотеку Platform MPI, в таком случае ваш сценарий запуска должен начинаться с команды:

**module load mpi/pmpi**

Следующим отличием от сценария одиночной задачи является запрос для задачи отличного от единицы количества вычислительных узлов:

*#SBATCH -N 10*

Эта директива говорит о том, что задача будет запущена на 10 узлах одновременно.

Последней модификацией в сценарии является добавление слова “srun” в начало команды запуска параллельной программы.

Собирая всё вместе, получаем следующий сценарий запуска параллельной задачи:

*#!/bin/sh*

*#SBATCH -N 10*

*#SBATCH -o %j.out*

*#SBATCH -e %j.err*

*#SBATCH -t 01:00:00*

*#SBATCH -p compute*

*module load mpi/pmpi*

*srun ./mpi-program*

1. **Список очередей и их назначение**

На данный момент на ЦОД ПИК существуют следующие очереди:

* **compute** – очередь по умолчанию для большинства стандартных расчётов;
* **knl** – очередь для вычислений, использующих векторные процессоры Xeon Phi;
* **bigmem** и **hugemem** – очереди на узлах с большой оперативной памятью.