Remote Access

To run programs on a CS machine when you're not sitting in front of it, connect over the network using ssh. Many computers are available for remote logins.

Who

Anyone with a CS account.

Where

The new official shell host for the Department of Computer Science is linux.cs.uchicago.edu. Please use it as your first choice to ssh into.

linux.cs.uchicago.edu

This is actually uses DNS round robin to point to multiple servers:

- linux1.cs.uchicago.edu
- linux2.cs.uchicago.edu
- linux3.cs.uchicago.edu

Specs

- 16 Cores (2x 8core 3.1GHz Processors), 16 threads
- 64gb RAM
- 2x 500gb SATA 7200RPM in RAID1

Unix

Most Unix-like systems (including Mac OS X) include the ssh command line utility. To run command line utilities in Mac OS X, first launch Terminal, found in Applications * Utilities * Terminal. Linux distributions such as Ubuntu also include Terminal or xterm, where you can type commands like ssh.

For example:

```
$ ssh username@linux.cs.uchicago.edu
Last login: Fri Apr 16 17:18:33 2009 from belmont.cs.uchicago.edu
$
```

Windows

Remote Terminal

We suggest using PuTTY, but there are other choices as well.

File Transfer

Unix

To move files between your local machine and a CS login machine, we recommend using scp (which uses the ssh protocol underneath). Unix

Most Unix-like systems (this includes Mac OS X) include the command line utility scp. To run command line utilities in Mac OS X, first launch Terminal, found in Applications * Utilities * Terminal.

For example:

```
  $ scp username@linux.cs.uchicago.edu:~/file.txt ./
  file.txt               100% |*****************************|  1088
00:00
```

Windows

We suggest WinSCP, which presents a familiar drag-and-drop interface, with two sections: one shows your local filesystem, and the other shows the remote filesystem. Problems?

First, check that you are running the software we recommend above. If you continue to have issues with PuTTY, WinSCP, or the Unix command line utilities, please contact us at techstaff@cs.uchicago.edu. The more details you give us, the faster we will be able to diagnose your problem.

The command line Unix programs have documentation available via the man command.

You can get diagnostics by running the command line Unix ssh program in verbose mode. For example:

```
  $ ssh -v user@linux.cs.uchicago.edu
  OpenSSH_4.7p1 Debian-12, OpenSSL 0.9.8k 25 Mar 2009
  debug1: Reading configuration data /etc/ssh/ssh_config
  debug1: Applying options for *
  debug1: Connecting to linux.cs.uchicago.edu [128.135.11.76] port 22.
  debug1: Connection established.
  ...more diagnostic info omitted...
```

Please include the entire diagnostic output in your problem report.

Advanced Usage

Using GUI applications remotely

Things you will need:

1. Windows 10 (https://uchicago.onthehub.com/WebStore/OfferingDetails.aspx?o=c46f1734-397a-e811-8106-000d3af41938)

2. Ubuntu Windows Subsystem for Linux (https://docs.microsoft.com/en-us/windows/wsl/install-win10) (note: only Ubuntu installation from the Windows store has been tested)

3. Xming (https://sourceforge.net/projects/xming/)

Steps: 1. Once you open the Ubuntu shell, enter the following command:

```
export DISPLAY=localhost:0.0
```

2. Make sure Xming is running on your computer.

3. SSH into linux.cs.uchicago.edu as you normally would, but add the -Y flag:

```
ssh -Y <cnet>@linux.cs.uchicago.edu
```

4. Now, to test, try opening SublimeText:

```
subl .
```

Passwordless Authentication

It is possible to use public-key cryptography so that passwords are not neccessary to login to CS machines. To do this, you need to create a key pair:

```
$ ssh-keygen -t rsa -N '' -f ~/.ssh/id_rsa
...
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
...
$ chmod 0600 ~/.ssh/authorized_keys
```

You should now be able to ssh between CS machines without a password.

What this does is change the burden of authentication from something you know (a password) to something you possess (the private key in the key pair generated above). Any entity that possesses the private key will be trusted as you by the system, so it makes sense to protect the private key. The above example uses the -N '' option to protect the private key with an empty passphrase (that is, no protection at all). You are trusting the strength of Unix file permissions to prevent a third party from accessing your private key, which is an unwise thing to do for a variety of reasons.

You can use a keypair more securely by encrypting the private key. Thus, if someone gets access to the raw private key file, they still need to decrypt it for authentication use. You can create a passphrase-protected key pair exactly as above, but leaving out the -N '' option.

However, if you do this, you have recreated the original problem you were trying to solve! You have just replaced typing an account password with typing a key passphrase every time you wish to authenticate. There is a solution though, that will make everyone happy, using programs called ssh-agent and keychain. Using them is the subject of the next section.

SSH configuration

Remembering all the options you need for SSH or scrolling in history to find the command might be cumbersome. Instead you can specify the type off connection in you SSH config file. This is located in ~/.ssh/config. Each entry contains a Host and a hostname. Additional arguments can be specified. Example for linux.cs.uchicago.edu:

```
Host uchicago
    Hostname linux.cs.uchicago.edu
    User <CNet ID>
    IdentityFile ~/.ssh/id_rsa

Host desktop
    Hostname <desktop hostname>   # Do not include .uchicago.edu
    User <CNet ID>
    IdentityFile ~/.ssh/id_rsa
    ProxyJump uchicago
```

Passphrase-protected SSH Keys using ssh-agent and keychain

ssh-agent is a program that runs in the background and keeps a decrypted copy of your private key in memory. Thus, you reduce the number of times you need to type in your passphrase from once per authentication event, to once every time the machine gets rebooted.

Rather than trusting someone that has access to your private key file (as above), or knowledge of your CS account password (as if you had not created a key pair at all), ssh-agent moves the trust to processes running with your identity.

ssh-agent uses environment variables to locate the connection (socket) to a running ssh-agent. keychain manages starting an ssh-agent if necessary, unlocking a key for the agent to hold, and export the appropriate environment variables so that other processes can use the ssh agent.

If using a Bourne-style shell (such as bash) add the following lines to a shell init file (.profile, .bash_profile, or whatever is appropriate to your shell of choice):

```
  keychain id_rsa
  . ~/.keychain/`uname -n`-sh
```

keychain, by default, is quite verbose. Once you are sure that it is working the way you intend, add a -q option onto the keychain invocation to silence it.