Mastering EOS

Release 1.0

Sean Fisk and Ira Woodring

October 20, 2014

Mastering EOS is an initiative to produce better documentation for the Grand Valley State University School of Computer and Information Systems Exploratory Operating System Labs (GVSU CIS EOS Labs). It is intended to augment the original EOS documentation located on the School of CIS website.

Mastering EOS is written by:

- Sean Fisk, BS and MS in Computer Science from GVSU
- Ira Woodring, GVSU EOS Lab system administrator

Contents:

CHAPTER

ONE

INTRODUCTION

The Exploratory Operating Systems Labs (EOS Labs) are a collection of computer labs maintained by the GVSU School of Computing and Information Systems. Some are for general CIS student use, and others are specific to certain courses, labs, or applications. The labs within the EOS Lab umbrella include:

Name	Location	Purpose
Exploratory Operating Systems Lab (EOS Lab)	MAK A-1-171	studying operating systems; general CIS computing use
Architecture Lab (Arch Lab)	MAK A-1-101	studying computer architectures; general CIS computing use
Data Communications Lab (Datacom Lab)	MAK 1-1-167	studying networking
Hardware Lab	MAK 1-1-105	studying hardware; multi-purpose

When addressing this collections of labs, we often refer to it simply as *EOS*. Though the EOS Lab is only one of the CIS department's computing labs, it was the original lab and other labs are largely based upon it. It is also the one most often used by the greatest variety of students.

1.1 Physical Access (Keycards)

The EOS Labs contain equipment specific to the CIS majors and are therefore closed to the general public. To access the labs, you are required to obtain a keycard. To be eligible for a keycard, you must be currently enrolled in a course which utiltizes the lab you'd like to access. After registration, visit the computing office in MAK C-2-100 to receive your card. A $25 deposit paid from your student account is charged upon receipt of the card. When the card is returned in acceptable condition, the deposit will be refunded. Please note that once the $25 deposit has been posted to your student account you must pay the balance, otherwise you will be charged a late fee. The late fee will not be refunded once you return the keycard.

To then gain entry to a lab to which you have been granted access, simply swipe the card at the reader next to the door to unlock it. In addition to room access, the card grants 24-hour access to Mackinac Hall. The courtyard door which is closest to the Architecture Lab possesses a reader which will open when the key is swiped. No other building doors are equipped with readers.

It is important that you do not allow anyone else to use your card. In addition, do not keep the card in your wallet, as the cards contain tiny hair-like wires that break easily when the card is flexed.

1.2 Computer Access (Credentials)

The EOS system uses separate user accounts from the general university computing infrastructure. EOS uses the same network ID given by GVSU's IT department, but authenticates against the CIS department's own LDAP server. The

accounts are not interchangeable and administrators for one account cannot reset passwords for the other.

If you are registered for one or more eligible CIS courses [#eligible-cis-courses], you qualify for an EOS account. On the Friday before the semester starts, an account will be automatically created for you and a temporary password sent to your GVSU email address. To activate your account, you need to log in to the system once, either at a physical machine or *remotely using SSH*, and follow these steps:

- Enter your username.
- Enter the temporary password you have been given.
- If the temporary was entered correctly, you will be asked for the Current Password again. Enter the temporary password again.
- If the two temporary passwords match, you will be asked to create a new password.

Note the following rules when creating a new password:

- Your password must be at least 7 characters.
- Your password must not be based on a dictionary word.
- Your password should not be all numbers. The system will accept such a password, but it is incredibly insecure. Additionally, the system will often prevent login with such a password.

Please take the time to memorize your password! Password resets are available by contacting Ira Woodring, but it often takes a day to get to it. Professors are also able to reset passwords via an SSH reset mechanism, though some are unaware of this mechanism.

CHAPTER

TWO

RULES AND PROCEDURES

It is important that some basic rules and procedures be established to help maintain the lab and aid in its shared use. To those ends, please be aware of the following guidelines:

2.1 Disk Space

You are given 8 gigabytes (GB) of disk space. Once you exceed this limit, you can no longer write new data to the filesystem. This often leads to being unable to login via a graphical session, as the desktop manager must be able to write to disk. You must then log in via a text-based console and delete files to make space.

The amount of disk space you have currently used may be checked with the **quota** command:

```
$ quota --human-readable
Disk quotas for user smithj (uid 1234):
     Filesystem   space   quota   limit   grace   files   quota   limit   grace
148.61.162.101:/home
                  9728M*  8192M  10240M            303k       0       0
```

The output of this command is two sets of four columns. The first four refer to physical disk space, while the second four refer to the number of files. The columns have the following meanings:

Column	Meaning
space	The amount of disk space you are currently using.
files	The number of files in your home directory.
quota	The soft limit for space or files. You may exceed this for the time listed in the grace period.
limit	The hard limit for space or files. You may never exceed this limit.
grace	The amount of time for which you can exceed the soft limit.

For the previous output, the user has a soft limit of 8192 MiB / 1024 MiB/GiB = 8 GiB and a hard limit of 10240 MiB / 1024 MiB/GiB = 10 GiB. The user is currently using 9728 MiB / 1024 MiB/GiB = 9.5 GiB of disk space, and is over quota (indicated by the $\star$). The output is expressed in mebibytes, which might be different than the megabytes to which you are accustomed. There is no grace period set, so the user is able to write files until reaching the hard limit. There is also no limit on the number of files the user can create, only a limit on the amount of space consumed.

2.2 Copyrighted Material

Any files may be stored in your home directory. This includes games, movies, and music. However, allowing others to transfer copyrighted material may constitute a copyright infringement.

These incidents are usually caused by unintentional permissions issues or deliberate misuse of the system. Whatever the cause, copyright infringement is a violation of school policy.

These cases are taken very seriously. Your user account will be terminated immediatly upon discovering the infringement. Additionally, this offense is against our school's honor code, so you may be expelled or even face criminal proceedings.

Suffice to say, please be very careful when dealing with copyrighted materials within the EOS system. Be familiar with how permissions work, and take the time to set them correctly.

2.3 Food and Drink

You are allowed to eat and drink in the labs — these labs were made for you and we want them to be as comfortable and useful as possible. Many restaraunts in the area deliver to the labs. However, it is your responsibility to clean up after yourself. If the lab becomes too messy, policies to limit food and drinks will be instituted.

2.4 Overnight Parking

Even if you have a parking pass, it is still neccessary to obtain a special permit to park overnight. These permits are granted by Campus Safety. Parking overnight without one of these permits can result in a ticket or towing.

2.5 Living in the Lab

It should go without saying (but hasn't in the past) that you cannot live in the lab. People have been found living in the lab for brief periods of time between leases, etc. We must note that this is not only a huge safety violation but is illegal. If caught living in the lab, you will be removed and Campus Safety notified.

2.6 Malicious Activity

The infrastructure provided by the EOS Labs includes very powerful tools that can be used to secure applications and network infrastructure. Unfortunately, these tools may also be used for malicious purposes. We provide these tools for you to learn to defend future systems it may be your job to secure. Under no circumstances should these tools be used to attack other students, machines, or entities. We do not provide these resources without reasonable oversight as to their use, and those using them illegally will be noticed and face strong consequences — possibly including removal from the university and criminal charges.

2.7 Games

We allow playing games in the lab. However, if you are playing games and all machines are currently in use, please be polite and yield your machine to students needing to complete coursework. Failure to do so may result in suspension of your account, or all games being removed from the system.

CHAPTER

THREE

COMMAND LINE BASICS

Please visit EOS Basics for a tutorial on how to use the command line. This is a great place to learn, but be aware that some things may be outdated due to updates in the system.

CHAPTER

FOUR

REMOTE ACCESS (SSH/VNC)

When not sitting at a physical machine in the EOS or Arch lab, EOS can be accessed from home using a protocol called Secure Shell (SSH). SSH gives you a prompt open to *Bash*, the default shell on EOS. From this shell, you can run commands as you would inside a normal terminal emulator in an EOS desktop session. The commands will be run on the EOS machine to which you have connected.

SSH is a command-line-only technology. However, graphical remote access is available through a protocol called Virtual Network Computing (VNC). VNC allows access to a graphical desktop as if sitting at an EOS machine. Because the VNC protocol has no security of its own, our lab setup requires tunnelling VNC traffic through the SSH protocol. Each respective guide describes how to do this, but remember that you will first need to successfully set up SSH before attempting to use VNC.

In addition to command-line and graphical access, you will likely need to transfer files between your local machine and EOS. This can be accomplished using the protocols Secure Copy (SCP) and Secure File Transfer Protocol (SFTP). Using these protocols, files can be transferred to and from EOS as well as synced between EOS and your local machine. In addition, the use of software FUSE and SSHFS allows you to treat files on EOS as if they were located on your local machine.

The hostnames for the EOS machines are organized as follows: `eos`*`XX`*`.cis.gvsu.edu` where *XX* is 01 through 32 and `arch`*`XX`*`.cis.gvsu.edu` where *XX* is 01 through 10. Use these names to connect to a specific EOS machine.

Your SSH and VNC clients of choice depend on your machine's operating system.

4.1 Microsoft Windows

4.1.1 SSH

The most popular SSH client for Windows is called PuTTY. It can be installed by visiting the PuTTY download page. We recommend installing via the Windows installer, labeled *A Windows installer for everything except PuTTYtel*.

Logging In

The first step we will take is to create a saved session for our EOS connection configuration. This will save time for future logins.

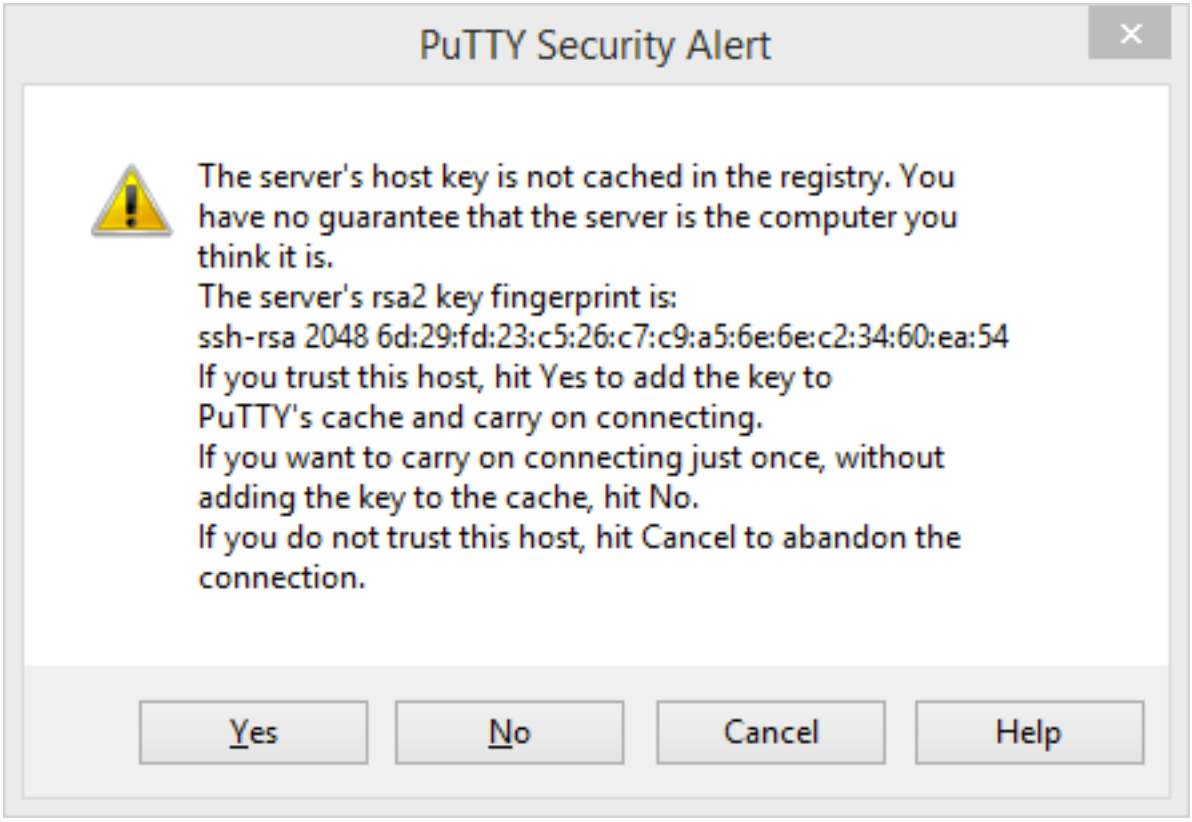
Open PuTTY and enter your username and the hostname of the EOS machine into the *Host Name* field. This has the form *user@host*, for example, `smithj@eos`*`XX`*`.cis.gvsu.edu`, where `XX` is the number of the chosen machine.

In the field under *Saved Sessions*, type `EOS` (this name is not strictly required, but is assumed in the next section). Click *Save*.

To log in to EOS, click *Open*. For future logins, simply select the session you created and click *Open* to connect.

Checking Host Fingerprints

When logging in to an EOS machine for the first time, you will see a dialog like this:



This is your SSH client requesting you to validate the identity of the machine to which you are connecting.

Each EOS machine has a so-called fingerprint, a series of characters which is used to verify its identity. To ensure that an attacker between your client and the actual EOS machine is not pretending to be an EOS machine, you must check that the machine's fingerprint matches the table below. Please report any mismatches to Ira Woodring immediately.

Host	Fingerprint
arch01	41:a3:8e:22:65:e7:cb:35:0a:51:9c:0f:cc:83:9c:20
arch02	41:e0:df:d1:e4:bd:b3:7f:c0:d4:15:dc:d1:dc:39:bf
arch03	64:fb:e5:5f:94:e4:d4:56:9d:8d:27:a1:a8:d8:e4:02
arch04	88:45:6e:3e:e5:20:7b:90:7d:ac:fe:f4:8f:46:3b:4e
arch05	f3:a7:a0:14:92:8d:29:0c:ab:d9:4d:d2:73:01:1e:f0
arch06	fe:75:66:3f:65:fc:63:3a:ce:c5:cd:82:ec:f9:21:fd
arch07	c6:a3:27:17:16:8a:7f:0c:8e:9f:07:77:86:b1:7b:20
arch08	b5:f9:51:c8:81:45:92:70:5a:33:06:ee:f8:7b:52:52
arch09	9b:8e:0c:48:ea:73:f6:16:b6:eb:7a:66:d4:4c:43:bc
arch10	9e:0c:a3:90:c1:12:79:72:92:c6:16:cd:06:77:78:2b
eos01	6d:31:03:c3:d4:76:ff:cc:89:dc:39:90:07:09:18:0d
eos02	6b:aa:3f:31:0f:4a:a8:c4:f8:48:db:30:8f:7e:87:db
eos03	58:21:fa:89:fc:5d:ca:76:99:b3:28:53:fa:5c:73:cf
eos04	a2:cf:9f:f3:6a:ed:df:5b:79:22:d8:00:db:cc:7b:bc
eos05	b5:c2:d4:40:23:d0:0c:af:32:28:36:b0:92:cc:86:da
eos06	down for maintenance
eos07	86:c9:3f:a4:0c:33:42:db:66:a9:4d:88:43:6a:bc:98

Continued on next page

Table 4.1 – continued from previous page

Host	Fingerprint
eos08	48:1a:76:fc:ea:89:42:fb:01:9d:9c:94:87:2e:18:9d
eos09	47:0e:ab:d3:9d:56:09:60:93:3f:7f:e1:75:da:4a:a9
eos10	47:23:74:5a:b4:f5:72:a9:29:41:02:cf:65:ef:c7:6e
eos11	2b:2c:b6:e1:0e:23:62:d0:32:9a:60:64:98:c8:6e:1b
eos12	c2:ed:49:40:72:49:6c:57:cf:f3:17:9b:04:18:f1:e1
eos13	62:b0:46:b3:9d:77:f4:6d:17:ad:53:b6:36:6b:72:24
eos14	cb:39:d3:5a:b2:84:a8:63:0d:cf:7c:40:ff:b6:1a:4d
eos15	10:fd:78:c3:37:79:a5:eb:ed:71:bd:d2:ee:3e:b7:ab
eos16	bb:8a:b7:f7:d5:64:d7:6a:21:d9:e6:0a:f6:9e:3c:09
eos17	e2:be:c4:1f:fc:b5:8a:ab:3d:b8:31:6a:f8:4a:4c:ae
eos18	df:d4:13:e5:74:71:77:0b:f6:5c:58:a5:b4:00:c4:c2
eos19	e2:2c:1e:20:a0:00:c9:38:c7:85:58:f2:8c:d5:71:bc
eos20	b0:63:3d:46:cc:a6:75:47:ea:0b:92:cf:26:9f:c6:54
eos21	df:ac:2b:cf:b1:1f:65:1c:c4:23:ff:b1:89:e0:08:a5
eos22	6e:1c:31:0b:37:12:56:32:e4:0d:7c:52:9f:3f:3d:ef
eos23	03:d0:ec:be:74:75:c7:b9:e6:b3:bc:b1:b2:db:10:cf
eos24	4c:b4:c5:36:ee:f2:5c:87:55:4f:a6:28:7b:80:c2:af
eos25	86:7a:af:f0:a6:ea:70:e4:69:6c:13:62:ac:59:2a:28
eos26	47:07:b9:d6:c5:a6:48:f7:7c:b5:3c:9a:48:d8:a0:c6
eos27	89:7a:06:61:87:b5:8e:df:9e:93:d2:26:a0:a4:b5:19
eos28	74:e2:00:99:ce:b1:ca:df:70:b5:6d:64:99:e4:1f:eb
eos29	07:07:e7:ff:c9:1c:31:11:2a:ad:80:69:d6:90:ee:cf
eos30	2d:d1:63:05:69:39:32:77:49:bf:d7:f4:60:93:62:6a
eos31	down for maintenance
eos32	c0:2a:f1:6c:41:52:f8:49:5f:5c:c7:bb:a6:f2:85:29

Password-less Logins (SSH keys)

It is often handy to be able to SSH into a host without having to type a password, for instance as part of a script. First, we need to generate your public/private key pair. Open PuTTYgen from the PuTTY distribution to begin the generation process. Click *Generate* and do the mouse nonsense to generate your keys.

Copy the text from the field labelled *Public key for pasting into OpenSSH authorized_keys file*. Open Notepad and paste this text. Save it to the desktop as `id_rsa.pub`. Now open Windows PowerShell from the Start Menu and run the following command. If your EOS saved session is named something other than "EOS", you will need to change it in the command below.

```
$puttySessionName = 'EOS'; Get-Content "$env:USERPROFILE\Desktop\id_rsa.pub" | & "$env:SYSTEMDRIVE\Pr
```

Your public key has now been uploaded to EOS. The file `id_rsa.pub` may be deleted now.

However, we still need to be able to tell PuTTY to use your private key to log in to EOS. Back in PuTTYgen, click *Save private key* and save the resulting PPK file to a location of your choosing. We recommend your home directory. Answer *Yes* when you are warned about saving the private key without a passphrase.

Note: If you would like to use a passphrase for your key, see the PuTTY Guide to Pageant after completing this guide. Setting up an SSH agent is out of the scope of this guide.

Now start up PuTTY, select your saved session, then click *Load*. This loads our previously configured session for editing. In the configuration tree to the left, expand *Connection* → *SSH* and click on *Auth*. Click *Browse...* to the right of the field labelled *Private key file for authentication*. Select the PPK file you saved earlier.

Go back to *Session* and click *Save*. PuTTY is now configured to use this private key to connect to EOS. Click *Open* to log in, which you should be able to do without a password.

Important: When you make changes to your configuration, make sure to go back to *Session* and click *Save*. If you click *Open* after making changes, those changes will be applied to the current session but will not be saved for the next time you open PuTTY.

Hint: You can also save a modified configuration under another name by editing the session name in the text box under *Saved Sessions* and clicking *Save*.

As is obvious from these instructions, SSH key management is not a simple process. We recommend reading the PuTTY Guide to SSH Keys, which is the source for much of this information. If you would like to use a passphrase with your key, please see the PuTTY Guide to Pageant, PuTTY's SSH agent.

Tunnelling / Port Forwarding

The SSH protocol possesses a special feature which allows it to tunnel other protocols within itself. This is called tunnelling or port forwarding. SSH can forward local ports (allowing the local machine access to resources on the remote machine) and remote ports (allowing the remote machine access to resources on the local machine).

Local port forwarding is the more used feature, and is explained in the following sections. Remote port forwarding is similar but is outside the scope of this guide.

Fortunately, port forwarding with PuTTY is quite easy. Fire up PuTTY and select your saved session, then click the *Load*. In the configuration tree to the left, expand *Connection* → *SSH* and click on *Tunnels*.

In the *Source port* field, we will enter the port to which traffic should arrive on our local machine. In the destination field, we will enter the host and port from which the traffic should be forwarded in the form `host:port`. The host will usually match the EOS machine to which you are connecting using SSH, although this is not required. The radio buttons should be left at *Local* for the forwarding type and *Auto* for the Internet protocol.

For example, where `eosXX.cis.gvsu.edu` is the remote EOS machine, to access a web server running on port 8000 on the EOS machine from your machine on port 5555, enter the following:

Source port	5555
Destination	`eosXX.cis.gvsu.edu:8000`

Click *Add* to add this as a forwarded port, then click *Open* (we will not save this configuration).

You can test the forwarding by running this in the SSH prompt:

```
python -m SimpleHTTPServer
```

and opening http://localhost:5555/ in your local web browser. You should see a web listing of your home directory! Press `Control-C` to kill the web server.

The remote host which is hosting the resource need not be the EOS machine to which you are connecting with SSH. Let's access the CIS web server through the SSH tunnel.

Restart PuTTY, load your session, and navigate back to the *Tunnels* screen. Enter the following information:

Source port	5678
Destination	`cis.gvsu.edu:80`

Click *Add* and *Open*, then visit http://localhost:5678/ in your local web browser. The CIS home page should appear!

4.1.2 VNC

First, we need to create a tunnel in order to forward VNC through our SSH connection. The remote port to which we must connect depends on the desired resolution of the remote desktop. Select a desired resolution from the following table, and note the port to which it corresponds.

Display	Port	Geometry
0	5900	1280x1024
1	5901	1024x768
2	5902	800x600
3	5903	640x480
4	5904	1440x900
5	5905	1280x800
6	5906	1152x864
7	5907	1680x1050
8	5908	1920x1200
9	5909	1400x1050
10	5910	1440x1000
11	5911	1024x600
12	5912	1600x900
13	5913	1920x1080
14	5914	1360x768

In the following instructions, replace `REMOTE_PORT` with the port that you have selected.

Restart PuTTY, load your session, and navigate back to the *Tunnels* screen. Enter the following information:

Source port	5900
Destination	`eos`*`XX`*`.cis.gvsu.edu:`*`REMOTE_PORT`*

Go back to *Session* and click *Save*. You are now ready to tunnel your VNC session. Click *Open* to start the tunnel.

Hint: If you clone a session for an EOS machine (using *Load* and *Save*), don't forget to change the tunnel to forward ports to that machine.

The recommended VNC client for Windows is TightVNC. Download it, install, then open. In the field labelled *Remote Host*, type `localhost`. Click *Connect* to start the connection.

For future connections, simply start TightVNC and click *Connect*. Alternatively, during the session, you can save the configuration to a file by clicking the *Save* button, shown as a diskette. After saving the configuration to a `*.vnc` file, double click the file to start the connection.

4.1.3 File Transfer

Graphical

The recommend graphical client for file transfer on Windows is WinSCP, which can be found on the WinSCP downloads page. We recommend downloading the latest stable installer, labeled *Installation package*. It should be near the top.

After downloading, run the installer. The *Typical installation* is usually fine, but feel free to customize the installation options. You can choose either the *Commander* or *Explorer* interface, but keep in mind that most people use the *Commander* interface. Don't forget to disable the sneaky Google Chrome installer included with this installer.

After the installer copies its files, it may detect your sessions from PuTTY. If so, click *OK* to import them. Select the sessions you'd like to import and click *OK* again. This is the single easiest way to start quickly with WinSCP.

At the end of the installer, leave the box labeled *Launch WinSCP* checked. You can choose to open the *Getting started* page as well, although there is really no need to do so.

If you didn't choose to import your sites from PuTTY in the installer, you can also import them from the WinSCP Login screen by clicking *Tools* → *Import Sites...*, selecting the sites, and clicking *OK*.

There is really no reason not to import your sites from PuTTY if you already have them configured (you should). However, if you'd like to create a custom site, click *New Site*. Choose SFTP as the protocol, and enter in the EOS machine for *Host name* as well as your username. For authentication, you can use a password or SSH keys. To select a key, click *Advanced...*, then *SSH* → *Authentication* → *Authentication parameters* → *Private key file* to select the private key file. Click *Save* to save your site.

Tip: You can create a desktop shortcut for your site by right-clicking your site in the WinSCP Login screen, then clicking *Desktop Icon*. This allows you to open your site directly without visiting the WinSCP Login screen. Creating a 'Send To' shortcut for Windows Explorer is similarly useful.

Automatic Synchronization

Automatic synchronization of local to remote directories is a very useful advanced feature of WinSCP. It is especially useful when developing a website on EOS. This partially makes up for the lack of a maintained free Windows SSHFS implementation.

To start using it, click *Commands* → *Keep Remote Directory up to Date...*. You can get more information about this task and its use in the WinSCP Keep Remote Directory up to Date documentation.

For more information on using WinSCP, consult the excellent WinSCP Documentation.

Command Line (SCP)

Files can be transferred on the command line using a utility called SCP, implemented in PuTTY through a command called `pscp`. Because `pscp` uses PuTTY for authentication, if you have set up *Password-less Logins (SSH keys)*, you will not have to type any passwords. SCP stands for *Secure Copy* and works very similar to the GNU/Linux `cp` command, except that it can also transfer files across the network. Make sure you are familiar with the operation of `cp` before using SCP.

PuTTY's commands are not added to the Windows Path by default. To add them to the `Path`, open Windows PowerShell from the Start Menu and run the following command. If you installed PuTTY to a non-default location, you will need to change it in the command below.

```
$puttyInstallPath = 'C:\Program Files (x86)\PuTTY'; [Environment]::SetEnvironmentVariable('Path', [Er
```

Restart PowerShell or your terminal emulator after running this command to allow your updates to the `Path` to take effect. The `pscp` utility may now be used from PowerShell by simply typing `pscp`.

The following examples showcase the typical use of `pscp`. Each file can be prefixed with a PuTTY session name or user/host, which tells `pscp` where it is or should be located. The session name "EOS" is used in these examples; change it to match your PuTTY session name if it is different. Files with no prefix are assumed to be on the local machine. Paths on the remote machine start at your home directory, so there is typically no need to include `/home/smithj` in the path. Here are some examples of use of `pscp`:

```
# Typical upload
pscp classes\cis162\hw1.txt EOS:classes/cis162
# Typical download
pscp EOS:classes/cis162/hw2.txt classes\cis162
# Upload a directory
pscp -r projects EOS:classes/cis163
```

```
# User/host instead of EOS session name (will likely require password)
pscp smithj@eos01.cis.gvsu.edu:classes/cis162/hw3.txt classes/cis162
```

Note: Windows uses `\` as a path separator by default, while GNU/Linux uses `/`. While Windows is generally forgiving and will accept `/` as well, GNU/Linux is not. *Always use `/` as a path separator when specifying GNU/Linux paths.*

`Path`
: The Windows search path for executable files.

4.1.4 Alternative Clients

Though PuTTY is the recommended SSH client for Windows, OpenSSH is also available. The recommended way of running OpenSSH on Windows is through Cygwin. Cygwin is not simple to use and configure, but depending on your needs, it may provide a better SSH experience. OpenSSH is well-known as the best SSH client out there, and EOS uses OpenSSH as an SSH server as well.

There are a plethora of alternate VNC viewers available for Windows, many based on the same original RealVNC code.

UltraVNC and TigerVNC offer relatively simple user interfaces with an appropriate amount of configuration options. If you don't like or are having trouble with TightVNC, give UltraVNC a try.

RealVNC Viewer and Viewer Plus are freeware viewers, but require registration. RealVNC also offers RealVNC Viewer for Google Chrome, a free Google Chrome extension which does not require registration.

Cyberduck is also available for Windows. Cyberduck has a more attractive and intuitive interface than WinSCP. However, unlike WinSCP, Cyberduck does not support automatic synchronization. This is important because high-quality free versions of SSHFS and rsync are not available for Windows.

MobaXterm is an all-in-one solution for SSH, SCP, VNC, RDP, and more. Since it is a unified product, it provides a smoother experience than a collection of standalone applications. However, because it includes much more capability, it can be difficult to configure. It is worth a try if your time spent on EOS warrants it.

Danger: Another popular client for SFTP is FileZilla. However, because FileZilla stores its passwords insecurely, we cannot recommend it. Please avoid its use to keep your password secure.

4.2 Mac OS X

4.2.1 SSH

Mac OS X comes preinstalled with OpenSSH, the most popular implementation of the SSH protocol. The client can be run from the command-line and is simply called `ssh`.

Logging in

First, open your terminal emulator to start a shell. To connect to a specific machine, pass your username and hostname to the command-line client in the form *user@host*. For example, where *XX* is the number of the chosen machine, run:

```
ssh smithj@eosXX.cis.gvsu.edu
```

Checking Host Fingerprints

When logging in to an EOS machine for the first time, you will see a message like this:

```
The authenticity of host 'eos01.cis.gvsu.edu (148.61.162.101)' can't be established.
RSA key fingerprint is 6d:29:fd:23:c5:26:c7:c9:a5:6e:6e:c2:34:60:ea:54.
Are you sure you want to continue connecting (yes/no)?
```

This is your SSH client requesting you to validate the identity of the machine to which you are connecting.

Each EOS machine has a so-called fingerprint, a series of characters which is used to verify its identity. To ensure that an attacker between your client and the actual EOS machine is not pretending to be an EOS machine, you must check that the machine's fingerprint matches the table below. Please report any mismatches to Ira Woodring immediately.

Host	Fingerprint
arch01	41:a3:8e:22:65:e7:cb:35:0a:51:9c:0f:cc:83:9c:20
arch02	41:e0:df:d1:e4:bd:b3:7f:c0:d4:15:dc:d1:dc:39:bf
arch03	64:fb:e5:5f:94:e4:d4:56:9d:8d:27:a1:a8:d8:e4:02
arch04	88:45:6e:3e:e5:20:7b:90:7d:ac:fe:f4:8f:46:3b:4e
arch05	f3:a7:a0:14:92:8d:29:0c:ab:d9:4d:d2:73:01:1e:f0
arch06	fe:75:66:3f:65:fc:63:3a:ce:c5:cd:82:ec:f9:21:fd
arch07	c6:a3:27:17:16:8a:7f:0c:8e:9f:07:77:86:b1:7b:20
arch08	b5:f9:51:c8:81:45:92:70:5a:33:06:ee:f8:7b:52:52
arch09	9b:8e:0c:48:ea:73:f6:16:b6:eb:7a:66:d4:4c:43:bc
arch10	9e:0c:a3:90:c1:12:79:72:92:c6:16:cd:06:77:78:2b
eos01	6d:31:03:c3:d4:76:ff:cc:89:dc:39:90:07:09:18:0d
eos02	6b:aa:3f:31:0f:4a:a8:c4:f8:48:db:30:8f:7e:87:db
eos03	58:21:fa:89:fc:5d:ca:76:99:b3:28:53:fa:5c:73:cf
eos04	a2:cf:9f:f3:6a:ed:df:5b:79:22:d8:00:db:cc:7b:bc
eos05	b5:c2:d4:40:23:d0:0c:af:32:28:36:b0:92:cc:86:da
eos06	down for maintenance
eos07	86:c9:3f:a4:0c:33:42:db:66:a9:4d:88:43:6a:bc:98
eos08	48:1a:76:fc:ea:89:42:fb:01:9d:9c:94:87:2e:18:9d
eos09	47:0e:ab:d3:9d:56:09:60:93:3f:7f:e1:75:da:4a:a9
eos10	47:23:74:5a:b4:f5:72:a9:29:41:02:cf:65:ef:c7:6e
eos11	2b:2c:b6:e1:0e:23:62:d0:32:9a:60:64:98:c8:6e:1b
eos12	c2:ed:49:40:72:49:6c:57:cf:f3:17:9b:04:18:f1:e1
eos13	62:b0:46:b3:9d:77:f4:6d:17:ad:53:b6:36:6b:72:24
eos14	cb:39:d3:5a:b2:84:a8:63:0d:cf:7c:40:ff:b6:1a:4d
eos15	10:fd:78:c3:37:79:a5:eb:ed:71:bd:d2:ee:3e:b7:ab
eos16	bb:8a:b7:f7:d5:64:d7:6a:21:d9:e6:0a:f6:9e:3c:09
eos17	e2:be:c4:1f:fc:b5:8a:ab:3d:b8:31:6a:f8:4a:4c:ae
eos18	df:d4:13:e5:74:71:77:0b:f6:5c:58:a5:b4:00:c4:c2
eos19	e2:2c:1e:20:a0:00:c9:38:c7:85:58:f2:8c:d5:71:bc
eos20	b0:63:3d:46:cc:a6:75:47:ea:0b:92:cf:26:9f:c6:54
eos21	df:ac:2b:cf:b1:1f:65:1c:c4:23:ff:b1:89:e0:08:a5
eos22	6e:1c:31:0b:37:12:56:32:e4:0d:7c:52:9f:3f:3d:ef
eos23	03:d0:ec:be:74:75:c7:b9:e6:b3:bc:b1:b2:db:10:cf
eos24	4c:b4:c5:36:ee:f2:5c:87:55:4f:a6:28:7b:80:c2:af
eos25	86:7a:af:f0:a6:ea:70:e4:69:6c:13:62:ac:59:2a:28
eos26	47:07:b9:d6:c5:a6:48:f7:7c:b5:3c:9a:48:d8:a0:c6
eos27	89:7a:06:61:87:b5:8e:df:9e:93:d2:26:a0:a4:b5:19
eos28	74:e2:00:99:ce:b1:ca:df:70:b5:6d:64:99:e4:1f:eb
eos29	07:07:e7:ff:c9:1c:31:11:2a:ad:80:69:d6:90:ee:cf

Continued on next page

Table 4.2 – continued from previous page

Host	Fingerprint
eos30	2d:d1:63:05:69:39:32:77:49:bf:d7:f4:60:93:62:6a
eos31	down for maintenance
eos32	c0:2a:f1:6c:41:52:f8:49:5f:5c:c7:bb:a6:f2:85:29

Password-less Logins (SSH keys)

It is often handy to be able to SSH into a host without having to type a password, for instance as part of a script. First, generate your public/private key pair with:

```
ssh-keygen
```

Accept the default values by pressing `Enter` at each prompt unless you know what you are doing. Once the keys have been generated, you can copy the public key over to the remote system by entering:

```
ssh smithj@eos01.cis.gvsu.edu 'umask u=rwx,go= && mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys' <
```

When you SSH into EOS now, you should be able to do so without having to provide a password:

```
ssh smithj@eos01.cis.gvsu.edu
```

Note: In this setup, we created our public/private key pair without a passphrase, which is less secure. If you would like to use a passphrase, please see Mark Hershberger's excellent guide to ssh-agent and Github's guide to SSH passphrases.

Tunnelling / Port Forwarding

The SSH protocol possesses a special feature which allows it to tunnel other protocols within itself. This is called tunnelling or port forwarding. SSH can forward local ports (allowing the local machine access to resources on the remote machine) and remote ports (allowing the remote machine access to resources on the local machine).

Local port forwarding is the more used feature, and is explained in the following sections. Remote port forwarding is similar but is outside the scope of this guide.

Port forwarding can be accomplished with OpenSSH by passing arguments to the command-line client or by editing the client configuration file.

Forwarding on the Command Line

Forwarding local ports on the command-line can be accomplished using the following syntax:

```
ssh -L local_port:remote_host:remote_port user@host
```

For example, to access a web server running on port 8000 on `eos01.cis.gvsu.edu` from your machine on port 5555, use the following command line:

```
ssh -L 5555:eos01.cis.gvsu.edu:8000 smithj@eos01.cis.gvsu.edu
```

You can test the forwarding by running this in the SSH prompt:

```
python -m SimpleHTTPServer
```

and opening http://localhost:5555/ in your local web browser. You should see a web listing of your home directory! Press `Control-C` to kill the web server.

The remote host which is hosting the resource need not be the EOS machine to which you are connecting with SSH. For example, to access the CIS web server through your SSH tunnel, you can run:

```
ssh -L 5678:cis.gvsu.edu:80 smithj@eos01.cis.gvsu.edu
```

and visit http://localhost:5678/ in your local web browser. The CIS home page should appear!

Forwarding in the Config File

The command-line works well for one-off tunnels, but for frequently established tunnels, it pays to alter the OpenSSH client configuration file. The OpenSSH client configuration resides on your local machine in the file `~/.ssh/config`. This is a file inside a hidden directory inside your home directory. The easiest way to open this file, creating it if it doesn't exist, is to run:

```
umask u=rwx,go= && mkdir -p ~/.ssh && touch ~/.ssh/config && open -t ~/.ssh/config
```

To establish the CIS web server forwarding shown in the last section, one could use the following configuration:

```
Host eoscisweb
HostName eos01.cis.gvsu.edu
User smithj
LocalForward 5678 cis.gvsu.edu:80
```

To use this host from the command line, simply type:

```
ssh eoscisweb
```

4.2.2 VNC

First, we need to create a tunnel in order to forward VNC through our SSH connection. The remote port to which we must connect depends on the desired resolution of the remote desktop. Select a desired resolution from the following table, and note the port to which it corresponds.

Display	Port	Geometry
0	5900	1280x1024
1	5901	1024x768
2	5902	800x600
3	5903	640x480
4	5904	1440x900
5	5905	1280x800
6	5906	1152x864
7	5907	1680x1050
8	5908	1920x1200
9	5909	1400x1050
10	5910	1440x1000
11	5911	1024x600
12	5912	1600x900
13	5913	1920x1080
14	5914	1360x768

In the following instructions, replace `REMOTE_PORT` with the port that you have selected.

To create the tunnel, use the following command line:

```
ssh -L 5900:eosXX.cis.gvsu.edu:REMOTE_PORT smithj@eosXX.cis.gvsu.edu
```

Or the following configuration file:

```
Host eosvnc
HostName eosXX.cis.gvsu.edu
User smithj
LocalForward 5900 eosXX.cis.gvsu.edu:REMOTE_PORT
```

If you used the configuration file, run the following to create the tunnel:

```
ssh eosvnc
```

You are now ready to tunnel your VNC session.

The recommended VNC client for OS X is Chicken, which is free and open-source software. Visit the website to download, install as you would any other Mac OS X application, and open.

You will be prompted to create a new server. If not prompted, click *Connection → Open Connection...* from the menu bar. Double click "New Server" in the list on the left and rename it to "EOS", or create a new session with the + button if none exist. Leave the *Host* field as `localhost` or fill it in if missing. Leave the *Display or port* field at 0 or fill it in if missing.

Chicken has had some problems with the ZRLE encoding with our server. As this can cause a premature end to your session, our recommendation is to manually disable this encoding. To do this, first click the drop-down menu next to *Profile*, and click *Edit Connection Profiles...*. The *Profile Manager* configuration window will open. In the bottom left, enter "EOS" into the field and click the + button. Now click the checkbox next to the ZRLE encoding to disable it for EOS sessions. Close the *Profile Manager* window.

Click *Connect* to begin your VNC session with EOS. To connect in the future, select *Connection → Open Connection...* from the menu, select your EOS configuration, and click *Connect*.

Note: Although Chicken offers an option to tunnel directly through SSH, we have not had luck using this option with our setup. We recommend sticking with the traditional SSH tunnel, as it is tested and works well.

4.2.3 File Transfer

Graphical

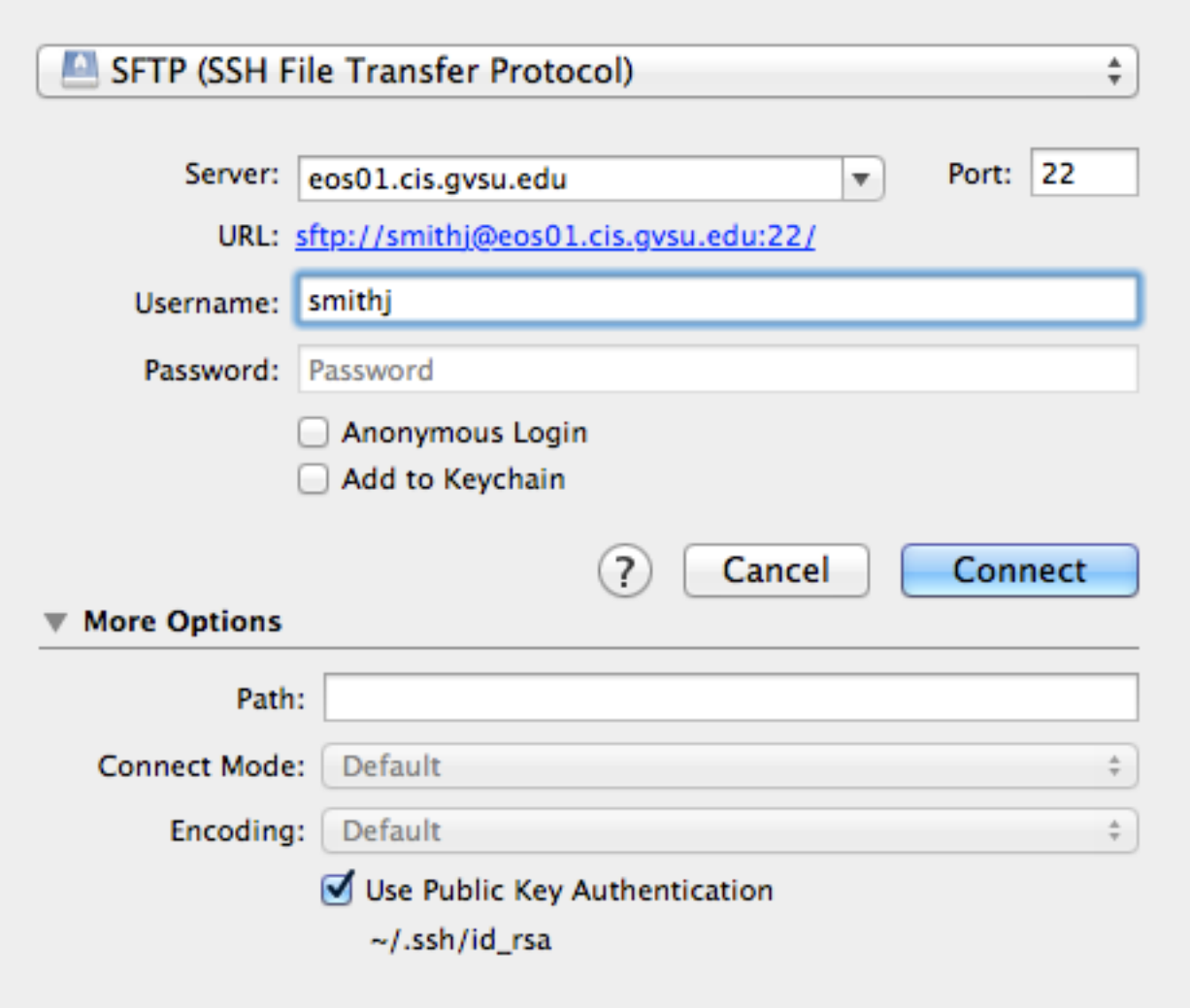
The recommend graphical client for file transfer on Mac OS X is Cyberduck. It is quite easy to use. Download the application, install as normal, and start Cyberduck.

Click *Open Connection*. From the drop-down box, select *SFTP (SSH File Transfer Protocol)*. Type in the EOS machine to which you'd like to connect in the *Server* field and fill in your username.

If you have set up *Password-less Logins (SSH keys)*, click the *Use Public Key Authentication* checkbox and select the file `id_rsa`. This is the identity file that you use to log in to EOS. If you have not set up your keys, you can still use password authentication, although this is not recommended.

If you have created a config file with hostname aliases, you may also notice Cyberduck auto-detect your configuration and fill in some information.

Your connection information should look something like this:



After connecting, we recommend creating a bookmark so that you can easily return. Click *Bookmark → New Bookmark* to create one. You can change the nickname if you like. When you start Cyberduck again, simply click your bookmark to connect.

Most of Cyberduck's action are available through the *File* menu or the right-click context menu. In addition, Cyberduck has great support for dragging between it and Finder.

Command Line (SCP)

Files can be transferred on the command line using a utility called SCP. Because SCP uses SSH for authentication, if you have set up *Password-less Logins (SSH keys)*, you will not have to type any passwords. SCP stands for *Secure Copy* and works very similar to the `cp` command, except that it can also transfer files across the network. Make sure you are familiar with the operation of `cp` before using SCP.

Each file can be prefixed with a machine name, which tells SCP where it is or should be located. Files with no prefix are assumed to be on the local machine. Paths on the remote machine start at your home directory, so there is typically no need to include `/home/smithj` in the path. Here are some examples of use of SCP:

```
# Typical upload
scp classes/cis162/hw1.txt eos01.cis.gvsu.edu:classes/cis162
# Typical download
```

```
scp eos01.cis.gvsu.edu:classes/cis162/hw2.txt classes/cis162
# Upload a directory
scp -r projects eos01.cis.gvsu.edu:classes/cis163
# Include username as well
scp smithj@eos01.cis.gvsu.edu:classes/cis162/hw3.txt classes/cis162
# Hostname aliases make this easier
scp eos01:classes/cis162/hw4.txt classes/cis162
```

4.2.4 Advanced OpenSSH

OpenSSH can do much more than simply allow the user to establish connections with remote servers. If you use OpenSSH, there are a great many neat tricks available to you.

This section is based in large part on the Smylers SSH Productivity Tips blog post. Please visit this post for *even more SSH awesomeness!*

Hostname Aliases

It's useful not to have to type out the entire full-qualified domain names to EOS machines. What you might normally type would be something like this:

```
ssh smithj@eos02.cis.gvsu.edu
# or
ssh smithj@arch04.cis.gvsu.edu
```

By adding a section to the config file, this becomes easier. Add this to your `~/.ssh/config` as mentioned earlier:

```
# EOS
# Match all eos01, eos11, arch08, etc.
Host eos?? arch??
HostName %h.cis.gvsu.edu
User smithj
```

With this, now you need only type:

```
ssh eos02
# or
ssh arch04
```

Shared Connections

There is a about a minute timeout between allowed successive connnections to each individual EOS machine. This can prove very annoying when establishing multiple SSH connections to use **scp** to copy files or opening multiple terminals (but see terminal multiplexing). One way to mitigate this annoyance is by using GVSU's VPN. Another way is to used SSH shared connections. These solutions are also not mutually exclusive.

Shared connections are established by creating a socket which multiplexes multiple connections. This socket is controlled by the `ControlMaster` and `ControlPath` keywords. The first connection is called the "master" and creates the socket. Subsequent connections use the already-created socket. This behavior can be automated by setting `ControlMaster` to the value `auto`. `ControlPath` specifies the path to the socket, with variables substituted as necessary. The following config amend the previous config to add connection sharing to EOS machines.

```
# EOS
# Match all eos01, eos11, arch08, etc.
Host eos?? arch??
```

```
HostName %h.cis.gvsu.edu
User smithj
ControlMaster auto
#                         Host
#                           |Port
#                           |   | Username
#                           V   V   V
ControlPath /tmp/ssh_mux_%h_%p_%r
```

Connection sharing may be useful to enable for most hosts. However, it needs to be done with care because it typically conflicts with X forwarding and port forwarding.

Persistent Connections

It is often useful to keep connections open in the background even after the terminal has actually been closed. This is useful as it allows OpenSSH to reconnect to the server without re-establishing a connection. Turning this behavior on is trivially simple. Add the following line under the host for which you would like connections to persist:

```
# Persist connections for 2 hours.
ControlPersist 2h
```

For GitHub users, this is especially useful when using Git over SSH. Within this period, OpenSSH does not need to re-establish a connection to the Git server, which makes pushes and pulls much faster.

Multi-Hop Connections

Oftentimes a machine is only available when SSH'ing into another machine. For example, this is the case with the DEN's Okami server, used in CIS 677 High-Performance Computing. In addition, Okami's SSH server is only available on a non-standard port. This typically results in the user going through this process:

```
smithj@local$ ssh smithj@eos01.cis.gvsu.edu
smithj@eos01$ ssh -p 43022 okami
smithj@okami$ # Finally here!
```

This is annoying and unnecessary. By using the `ProxyCommand` keyword in our config file, we can automate this process:

```
# DEN Okami
Host okami
User smithj
Port 43022
ProxyCommand ssh eos01 -W %h:%p
```

The `-W` flag allows us to hop through the first host to the host and port specified by the variables (`okami:43022`). Note that the use of `eos01` here requires presence of the aliases set up in Hostname Aliases.

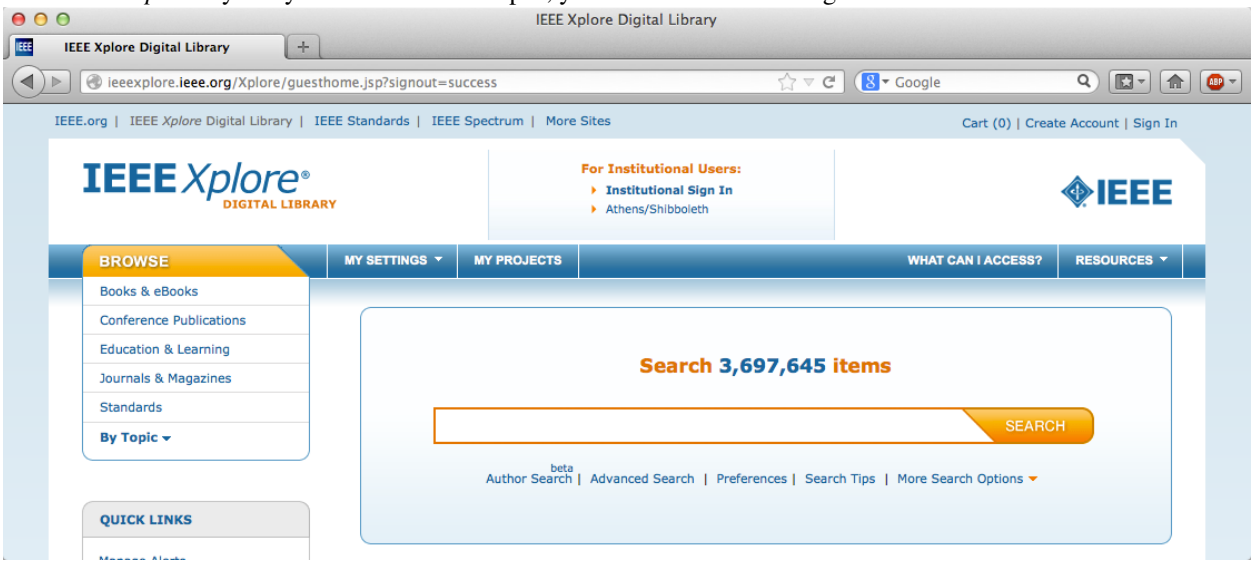
The process has now been simplified to:

```
smithj@local$ ssh okami
smithj@okami$ # Yay! Easy!
```

Using SSH as a Proxy

It is also possible to use SSH as a proxy for all network traffic. This can be useful if there are resources available from the SSH server that are not available from the local machine.

An example of such a resource is the IEEE Xplore Digital Library, which contains technical articles targeted at computer scientists and engineers. GVSU subscribes to this library, but access to the subscription is only available while *on campus*. If you try to access it off campus, you will see the following:



By using a proxy through the EOS machines, we can transparently access the IEEE library as if we were on campus.

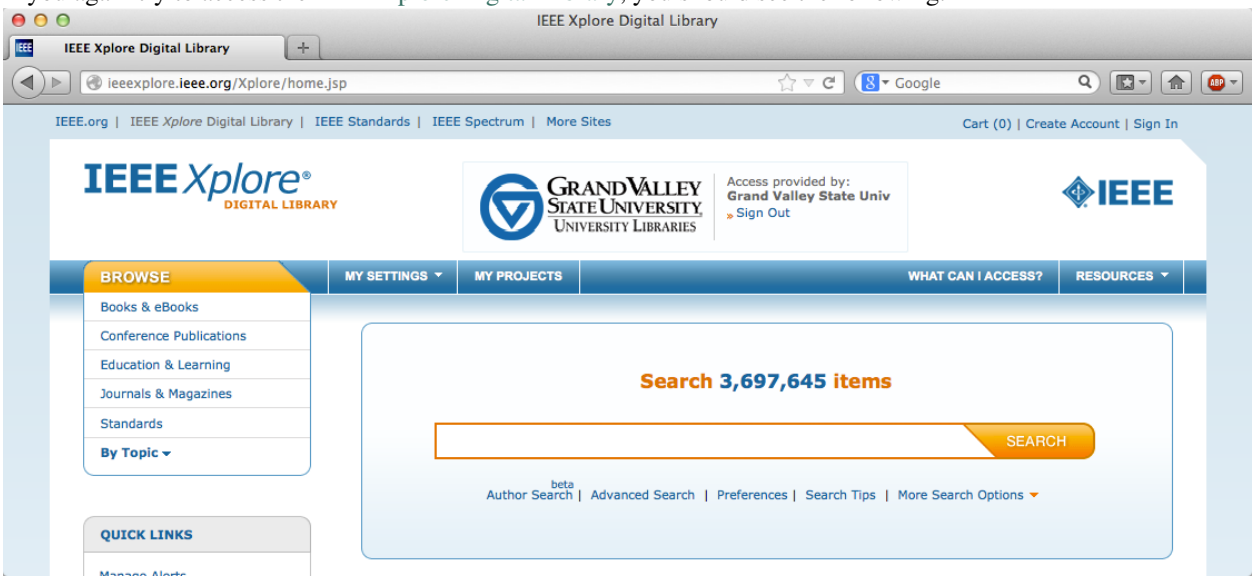
OpenSSH support the SOCKS protocol for proxying. Activating the SOCKS feature is accomplished with the `-D` flag like so:

```
ssh -D 5555 eos01
```

This establishes a SOCKS proxy with EOS01 served up on the local machine on port 5555. Now we must configure our operating system or browser to use this proxy.

On Mac OS X, configuring your system to use our SOCKS proxy is quite simple. First, open *System Preferences*. From here, choose *Network* → *Advanced...* → *Proxies* → *SOCKS Proxy*. Under the label *SOCKS Proxy Server*, enter `localhost` and `5555` to match the port passed to the `-D` flag. This can be any port as long as these numbers match. Check the box next to *SOCKS Proxy*, then click *OK* and *Apply* to turn on the proxy.

If you again try to access the IEEE Xplore Digital Library, you should see the following:



You have now successfully used OpenSSH to establish a SOCKS proxy!

Warning: By using a SOCKS proxy, *all* your network traffic is sent through the proxy. This has two implications:

- Network access will likely be slower.
- GVSU will be able to monitor your traffic as they do when you are on campus.

Please keep this in mind when using the proxy feature.

Example

For an example OpenSSH configuration file, see Sean's SSH config.

4.2.5 Alternative Clients

Chicken is not the only VNC viewer available for Mac OS X. Some alternatives are:

- TigerVNC is a capable free and open source VNC viewer. Its interface is not as Mac-friendly as Chicken, but it works well. If you are having problems with Chicken, try TigerVNC.
- RealVNC Viewer is a freeware viewer, but requires registration. RealVNC also offers RealVNC Viewer for Google Chrome, a free Google Chrome extension which does not require registration.
- JollysFastVNC is a full-featured VNC client with trial and paid versions available.
- Chicken of the VNC is an older version of Chicken and is not recommended.

Danger: Another popular client for SFTP is FileZilla. However, because FileZilla stores its passwords insecurely, we cannot recommend it. Please avoid its use to keep your password secure.

4.3 GNU/Linux

4.3.1 SSH

The most popular implementation of the SSH protocol on GNU/Linux is OpenSSH. The SSH client can be run from the command-line and is simply called **ssh**.

Many GNU/Linux distributions come with OpenSSH pre-installed. If your GNU/Linux distribution does not have it installed by default, please install it with your package manager. You should not install this software from scratch or on your own.

On Debian-based systems (Ubuntu, Linux Mint, and friends), run the following command:

```
sudo apt-get install openssh-client
```

On Red Hat-based systems (Fedora, CentOS, RHEL, and friends), run the following command:

```
sudo yum install openssh-clients
```

With other distributions (Arch, etc.), you are on your own.

Logging in

First, open your terminal emulator to start a shell. To connect to a specific machine, pass your username and hostname to the command-line client in the form *user@host*. For example, where *XX* is the number of the chosen machine, run:

```
ssh smithj@eosXX.cis.gvsu.edu
```

Checking Host Fingerprints

When logging in to an EOS machine for the first time, you will see a message like this:

```
The authenticity of host 'eos01.cis.gvsu.edu (148.61.162.101)' can't be established.
RSA key fingerprint is 6d:29:fd:23:c5:26:c7:c9:a5:6e:6e:c2:34:60:ea:54.
Are you sure you want to continue connecting (yes/no)?
```

This is your SSH client requesting you to validate the identity of the machine to which you are connecting.

Each EOS machine has a so-called fingerprint, a series of characters which is used to verify its identity. To ensure that an attacker between your client and the actual EOS machine is not pretending to be an EOS machine, you must check that the machine's fingerprint matches the table below. Please report any mismatches to Ira Woodring immediately.

Host	Fingerprint
arch01	41:a3:8e:22:65:e7:cb:35:0a:51:9c:0f:cc:83:9c:20
arch02	41:e0:df:d1:e4:bd:b3:7f:c0:d4:15:dc:d1:dc:39:bf
arch03	64:fb:e5:5f:94:e4:d4:56:9d:8d:27:a1:a8:d8:e4:02
arch04	88:45:6e:3e:e5:20:7b:90:7d:ac:fe:f4:8f:46:3b:4e
arch05	f3:a7:a0:14:92:8d:29:0c:ab:d9:4d:d2:73:01:1e:f0
arch06	fe:75:66:3f:65:fc:63:3a:ce:c5:cd:82:ec:f9:21:fd
arch07	c6:a3:27:17:16:8a:7f:0c:8e:9f:07:77:86:b1:7b:20
arch08	b5:f9:51:c8:81:45:92:70:5a:33:06:ee:f8:7b:52:52
arch09	9b:8e:0c:48:ea:73:f6:16:b6:eb:7a:66:d4:4c:43:bc
arch10	9e:0c:a3:90:c1:12:79:72:92:c6:16:cd:06:77:78:2b
eos01	6d:31:03:c3:d4:76:ff:cc:89:dc:39:90:07:09:18:0d
eos02	6b:aa:3f:31:0f:4a:a8:c4:f8:48:db:30:8f:7e:87:db
eos03	58:21:fa:89:fc:5d:ca:76:99:b3:28:53:fa:5c:73:cf
eos04	a2:cf:9f:f3:6a:ed:df:5b:79:22:d8:00:db:cc:7b:bc
eos05	b5:c2:d4:40:23:d0:0c:af:32:28:36:b0:92:cc:86:da
eos06	down for maintenance
eos07	86:c9:3f:a4:0c:33:42:db:66:a9:4d:88:43:6a:bc:98
eos08	48:1a:76:fc:ea:89:42:fb:01:9d:9c:94:87:2e:18:9d
eos09	47:0e:ab:d3:9d:56:09:60:93:3f:7f:e1:75:da:4a:a9
eos10	47:23:74:5a:b4:f5:72:a9:29:41:02:cf:65:ef:c7:6e
eos11	2b:2c:b6:e1:0e:23:62:d0:32:9a:60:64:98:c8:6e:1b
eos12	c2:ed:49:40:72:49:6c:57:cf:f3:17:9b:04:18:f1:e1
eos13	62:b0:46:b3:9d:77:f4:6d:17:ad:53:b6:36:6b:72:24
eos14	cb:39:d3:5a:b2:84:a8:63:0d:cf:7c:40:ff:b6:1a:4d
eos15	10:fd:78:c3:37:79:a5:eb:ed:71:bd:d2:ee:3e:b7:ab
eos16	bb:8a:b7:f7:d5:64:d7:6a:21:d9:e6:0a:f6:9e:3c:09
eos17	e2:be:c4:1f:fc:b5:8a:ab:3d:b8:31:6a:f8:4a:4c:ae
eos18	df:d4:13:e5:74:71:77:0b:f6:5c:58:a5:b4:00:c4:c2
eos19	e2:2c:1e:20:a0:00:c9:38:c7:85:58:f2:8c:d5:71:bc
eos20	b0:63:3d:46:cc:a6:75:47:ea:0b:92:cf:26:9f:c6:54
eos21	df:ac:2b:cf:b1:1f:65:1c:c4:23:ff:b1:89:e0:08:a5
eos22	6e:1c:31:0b:37:12:56:32:e4:0d:7c:52:9f:3f:3d:ef

Continued on next page

Table 4.3 – continued from previous page

Host	Fingerprint
eos23	03:d0:ec:be:74:75:c7:b9:e6:b3:bc:b1:b2:db:10:cf
eos24	4c:b4:c5:36:ee:f2:5c:87:55:4f:a6:28:7b:80:c2:af
eos25	86:7a:af:f0:a6:ea:70:e4:69:6c:13:62:ac:59:2a:28
eos26	47:07:b9:d6:c5:a6:48:f7:7c:b5:3c:9a:48:d8:a0:c6
eos27	89:7a:06:61:87:b5:8e:df:9e:93:d2:26:a0:a4:b5:19
eos28	74:e2:00:99:ce:b1:ca:df:70:b5:6d:64:99:e4:1f:eb
eos29	07:07:e7:ff:c9:1c:31:11:2a:ad:80:69:d6:90:ee:cf
eos30	2d:d1:63:05:69:39:32:77:49:bf:d7:f4:60:93:62:6a
eos31	down for maintenance
eos32	c0:2a:f1:6c:41:52:f8:49:5f:5c:c7:bb:a6:f2:85:29

Password-less Logins (SSH keys)

It is often handy to be able to SSH into a host without having to type a password, for instance as part of a script. First, generate your public/private key pair with:

```
ssh-keygen
```

Accept the default values by pressing `Enter` at each prompt unless you know what you are doing. Once the keys have been generated, you can copy the public key over to the remote system by entering:

```
ssh-copy-id smithj@eos01.cis.gvsu.edu
```

When you SSH into EOS now, you should be able to do so without having to provide a password:

```
ssh smithj@eos01.cis.gvsu.edu
```

Note: In this setup, we created our public/private key pair without a passphrase, which is less secure. If you would like to use a passphrase, please see Mark Hershberger's excellent guide to ssh-agent and Github's guide to SSH passphrases.

Tunnelling / Port Forwarding

The SSH protocol possesses a special feature which allows it to tunnel other protocols within itself. This is called tunnelling or port forwarding. SSH can forward local ports (allowing the local machine access to resources on the remote machine) and remote ports (allowing the remote machine access to resources on the local machine).

Local port forwarding is the more used feature, and is explained in the following sections. Remote port forwarding is similar but is outside the scope of this guide.

Port forwarding can be accomplished with OpenSSH by passing arguments to the command-line client or by editing the client configuration file.

Forwarding on the Command Line

Forwarding local ports on the command-line can be accomplished using the following syntax:

```
ssh -L local_port:remote_host:remote_port user@host
```

For example, to access a web server running on port 8000 on `eos01.cis.gvsu.edu` from your machine on port 5555, use the following command line:

```
ssh -L 5555:eos01.cis.gvsu.edu:8000 smithj@eos01.cis.gvsu.edu
```

You can test the forwarding by running this in the SSH prompt:

```
python -m SimpleHTTPServer
```

and opening http://localhost:5555/ in your local web browser. You should see a web listing of your home directory! Press `Control-C` to kill the web server.

The remote host which is hosting the resource need not be the EOS machine to which you are connecting with SSH. For example, to access the CIS web server through your SSH tunnel, you can run:

```
ssh -L 5678:cis.gvsu.edu:80 smithj@eos01.cis.gvsu.edu
```

and visit http://localhost:5678/ in your local web browser. The CIS home page should appear!

Forwarding in the Config File

The command-line works well for one-off tunnels, but for frequently established tunnels, it pays to alter the OpenSSH client configuration file. The OpenSSH client configuration resides on your local machine in the file `~/.ssh/config`. This is a file inside a hidden directory inside your home directory. The easiest way to open this file, creating it if it doesn't exist, is to run:

```
umask u=rwx,go= && mkdir -p ~/.ssh && touch ~/.ssh/config && gedit ~/.ssh/config
```

To establish the CIS web server forwarding shown in the last section, one could use the following configuration:

```
Host eoscisweb
HostName eos01.cis.gvsu.edu
User smithj
LocalForward 5678 cis.gvsu.edu:80
```

To use this host from the command line, simply type:

```
ssh eoscisweb
```

4.3.2 VNC

First, we need to create a tunnel in order to forward VNC through our SSH connection. The remote port to which we must connect depends on the desired resolution of the remote desktop. Select a desired resolution from the following table, and note the port to which it corresponds.

Display	Port	Geometry
0	5900	1280x1024
1	5901	1024x768
2	5902	800x600
3	5903	640x480
4	5904	1440x900
5	5905	1280x800
6	5906	1152x864
7	5907	1680x1050
8	5908	1920x1200
9	5909	1400x1050
10	5910	1440x1000
11	5911	1024x600
12	5912	1600x900
13	5913	1920x1080
14	5914	1360x768

In the following instructions, replace `REMOTE_PORT` with the port that you have selected.

To create the tunnel, use the following command line:

```
ssh -L 5900:eosXX.cis.gvsu.edu:REMOTE_PORT smithj@eosXX.cis.gvsu.edu
```

Or the following configuration file:

```
Host eosvnc
HostName eosXX.cis.gvsu.edu
User smithj
LocalForward 5900 eosXX.cis.gvsu.edu:REMOTE_PORT
```

If you used the configuration file, run the following to create the tunnel:

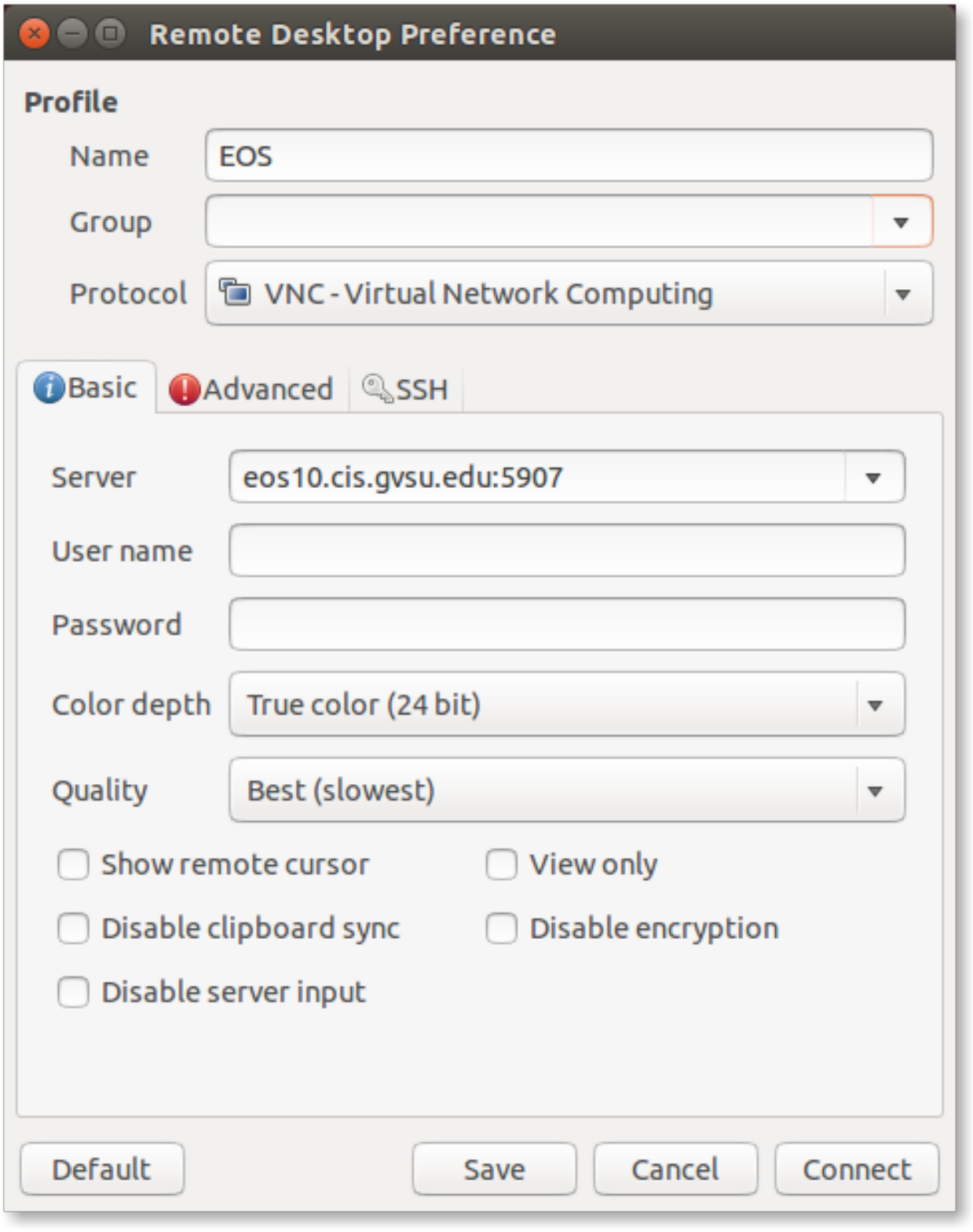
```
ssh eosvnc
```

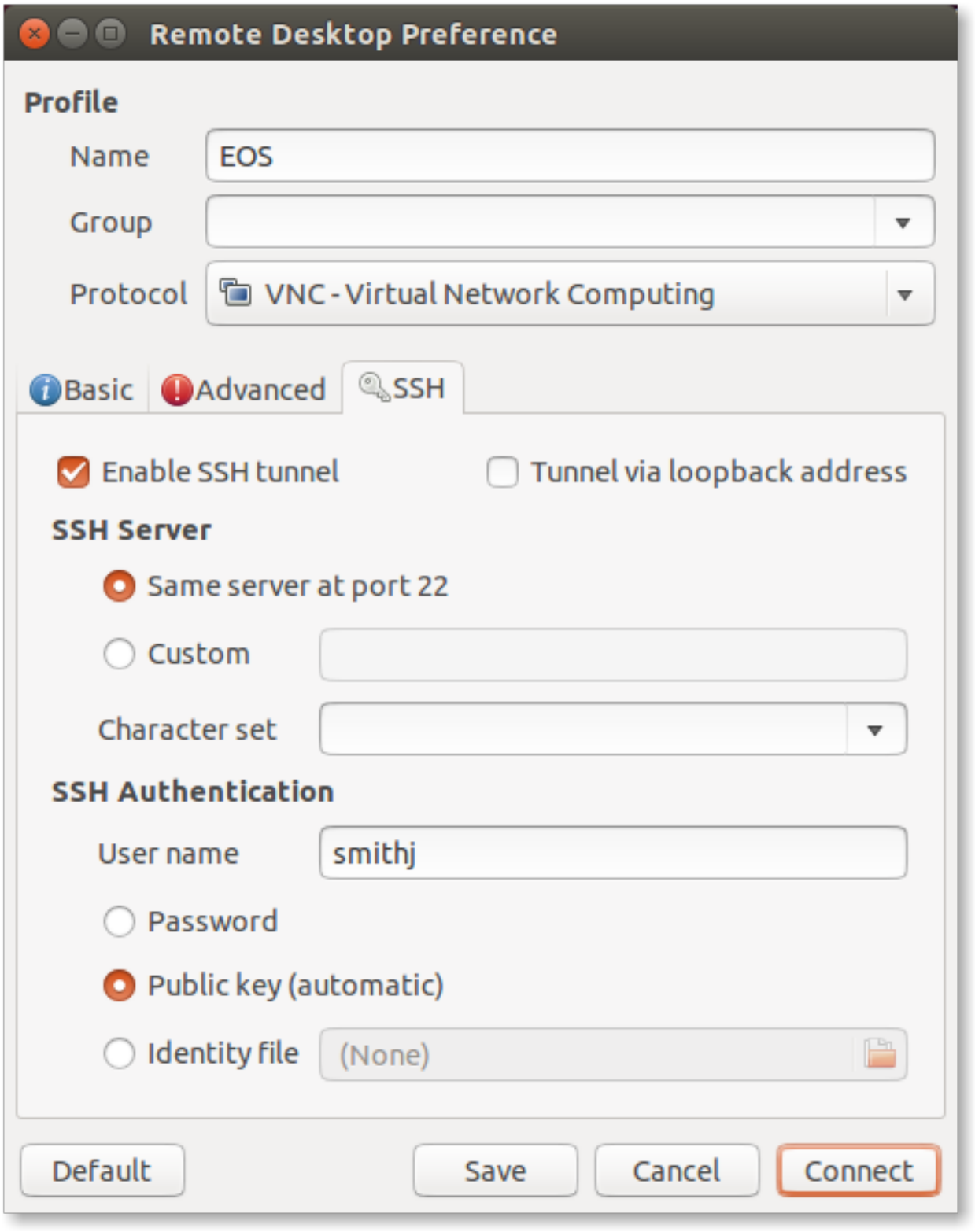
You are now ready to tunnel your VNC session.

There are a number of VNC clients for GNU/Linux, but the most capable and intuitive is Remmina (formerly tsclient). Remmina also supports RDP, so you can use it with *Winserv*. It installed by default in Ubuntu 14.04. If it is not installed in your distribution, you should install from your package manager.

In addition, Remmina supports automatic SSH tunneling. You do not need to establish a tunnel beforehand as shown in the previous section. However, if you need a shell or otherwise want to do it that way, there is nothing stopping you as it works just as well.

To configure Remmina for VNC with automatic SSH tunneling, open Remmina and select *Connection* → *New* to create a new connection. Under the *Basic* and *SSH* tabs, respectively, enter the following information. This configuration uses EOS10 and port 5907, but use the host of your choice and the port which matches your resolution from the previous section. Because *Public Key* is selected, if you have set up password-less login earlier, the login should be automatic.


Remote Desktop Preference
Profile
Name
EOS
Group
Protocol
VNC - Virtual Network Computing
Basic
Advanced
SSH
Server
eos10.cis.gvsu.edu:5907
User name
Password
Color depth
True color (24 bit)
Quality
Best (slowest)
Show remote cursor
View only
Disable clipboard sync
Disable encryption
Disable server input
Default
Save
Cancel
Connect


Remote Desktop Preference
Profile
Name
EOS
Group
Protocol
VNC - Virtual Network Computing
Basic
Advanced
SSH
Enable SSH tunnel
Tunnel via loopback address
SSH Server
Same server at port 22
Custom
Character set
SSH Authentication
User name
smithj
Password
Public key (automatic)
Identity file
(None)
Default
Save
Cancel
Connect

4.3.3 File Transfer

Command Line (SCP)

Files can be transferred on the command line using a utility called SCP. Because SCP uses SSH for authentication, if you have set up *Password-less Logins (SSH keys)*, you will not have to type any passwords. SCP stands for *Secure Copy* and works very similar to the `cp` command, except that it can also transfer files across the network. Make sure you are familiar with the operation of `cp` before using SCP.

Each file can be prefixed with a machine name, which tells SCP where it is or should be located. Files with no prefix are assumed to be on the local machine. Paths on the remote machine start at your home directory, so there is typically no need to include `/home/smithj` in the path. Here are some examples of use of SCP:

```
# Typical upload
scp classes/cis162/hw1.txt eos01.cis.gvsu.edu:classes/cis162
# Typical download
scp eos01.cis.gvsu.edu:classes/cis162/hw2.txt classes/cis162
# Upload a directory
scp -r projects eos01.cis.gvsu.edu:classes/cis163
# Include username as well
scp smithj@eos01.cis.gvsu.edu:classes/cis162/hw3.txt classes/cis162
# Hostname aliases make this easier
scp eos01:classes/cis162/hw4.txt classes/cis162
```

4.3.4 Advanced OpenSSH

OpenSSH can do much more than simply allow the user to establish connections with remote servers. If you use OpenSSH, there are a great many neat tricks available to you.

This section is based in large part on the Smylers SSH Productivity Tips blog post. Please visit this post for *even more SSH awesomeness!*

Hostname Aliases

It's useful not to have to type out the entire full-qualified domain names to EOS machines. What you might normally type would be something like this:

```
ssh smithj@eos02.cis.gvsu.edu
# or
ssh smithj@arch04.cis.gvsu.edu
```

By adding a section to the config file, this becomes easier. Add this to your `~/.ssh/config` as mentioned earlier:

```
# EOS
# Match all eos01, eos11, arch08, etc.
Host eos?? arch??
HostName %h.cis.gvsu.edu
User smithj
```

With this, now you need only type:

```
ssh eos02
# or
ssh arch04
```

Shared Connections

There is a about a minute timeout between allowed successive connnections to each individual EOS machine. This can prove very annoying when establishing multiple SSH connections to use **scp** to copy files or opening multiple terminals (but see terminal multiplexing). One way to mitigate this annoyance is by using GVSU's VPN. Another way is to used SSH shared connections. These solutions are also not mutually exclusive.

Shared connections are established by creating a socket which multiplexes multiple connections. This socket is controlled by the `ControlMaster` and `ControlPath` keywords. The first connection is called the "master" and creates the socket. Subsequent connections use the already-created socket. This behavior can be automated by setting `ControlMaster` to the value `auto`. `ControlPath` specifies the path to the socket, with variables substituted as necessary. The following config amend the previous config to add connection sharing to EOS machines.

```
# EOS
# Match all eos01, eos11, arch08, etc.
Host eos?? arch??
HostName %h.cis.gvsu.edu
User smithj
ControlMaster auto
#                          Host
#                            |Port
#                            |   | Username
#                            V   V   V
ControlPath /tmp/ssh_mux_%h_%p_%r
```

Connection sharing may be useful to enable for most hosts. However, it needs to be done with care because it typically conflicts with X forwarding and port forwarding.

Persistent Connections

It is often useful to keep connections open in the background even after the terminal has actually been closed. This is useful as it allows OpenSSH to reconnect to the server without re-establishing a connection. Turning this behavior on is trivially simple. Add the following line under the host for which you would like connections to persist:

```
# Persist connections for 2 hours.
ControlPersist 2h
```

For GitHub users, this is especially useful when using Git over SSH. Within this period, OpenSSH does not need to re-establish a connection to the Git server, which makes pushes and pulls much faster.

Multi-Hop Connections

Oftentimes a machine is only available when SSH'ing into another machine. For example, this is the case with the DEN's Okami server, used in CIS 677 High-Performance Computing. In addition, Okami's SSH server is only available on a non-standard port. This typically results in the user going through this process:

```
smithj@local$ ssh smithj@eos01.cis.gvsu.edu
smithj@eos01$ ssh -p 43022 okami
smithj@okami$ # Finally here!
```

This is annoying and unnecessary. By using the `ProxyCommand` keyword in our config file, we can automate this process:

```
# DEN Okami
Host okami
User smithj
```

```
Port 43022
ProxyCommand ssh eos01 -W %h:%p
```

The `-W` flag allows us to hop through the first host to the host and port specified by the variables (`okami:43022`). Note that the use of `eos01` here requires presence of the aliases set up in Hostname Aliases.

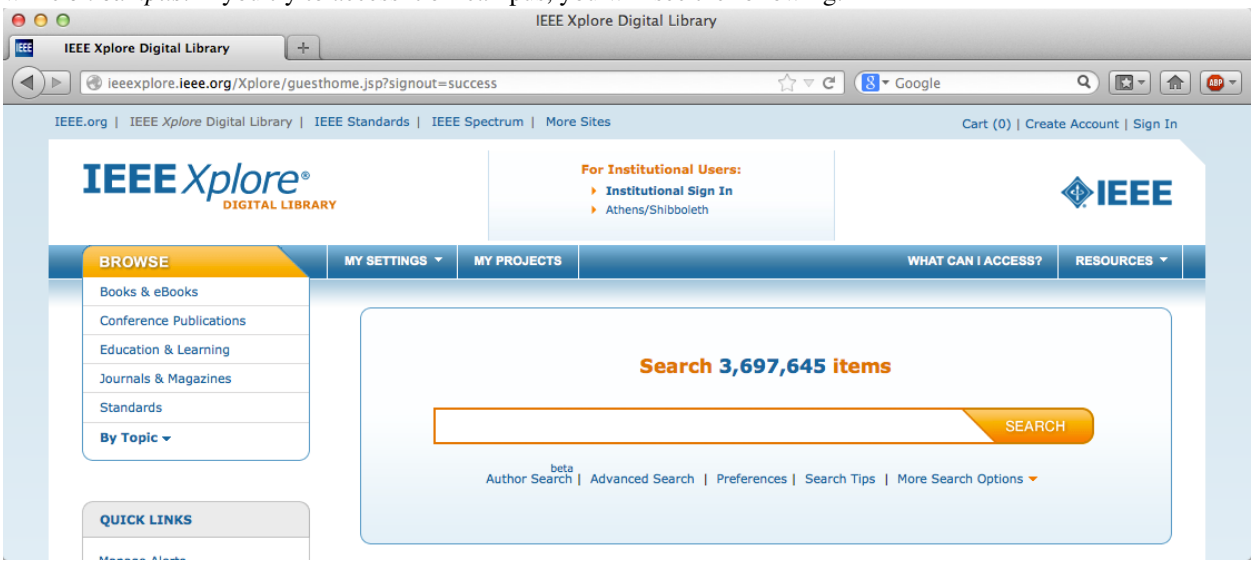
The process has now been simplified to:

```
smithj@local$ ssh okami
smithj@okami$ # Yay! Easy!
```

Using SSH as a Proxy

It is also possible to use SSH as a proxy for all network traffic. This can be useful if there are resources available from the SSH server that are not available from the local machine.

An example of such a resource is the IEEE Xplore Digital Library, which contains technical articles targeted at computer scientists and engineers. GVSU subscribes to this library, but access to the subscription is only available while *on campus*. If you try to access it off campus, you will see the following:



By using a proxy through the EOS machines, we can transparently access the IEEE library as if we were on campus.

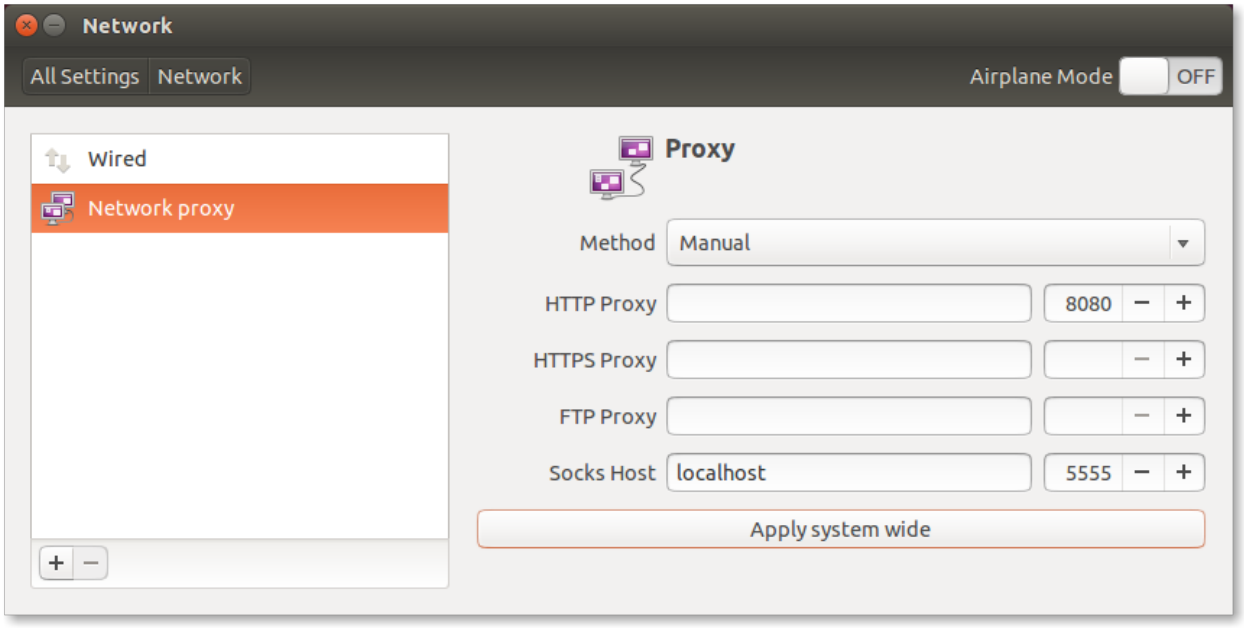
OpenSSH support the SOCKS protocol for proxying. Activating the SOCKS feature is accomplished with the `-D` flag like so:

```
ssh -D 5555 eos01
```

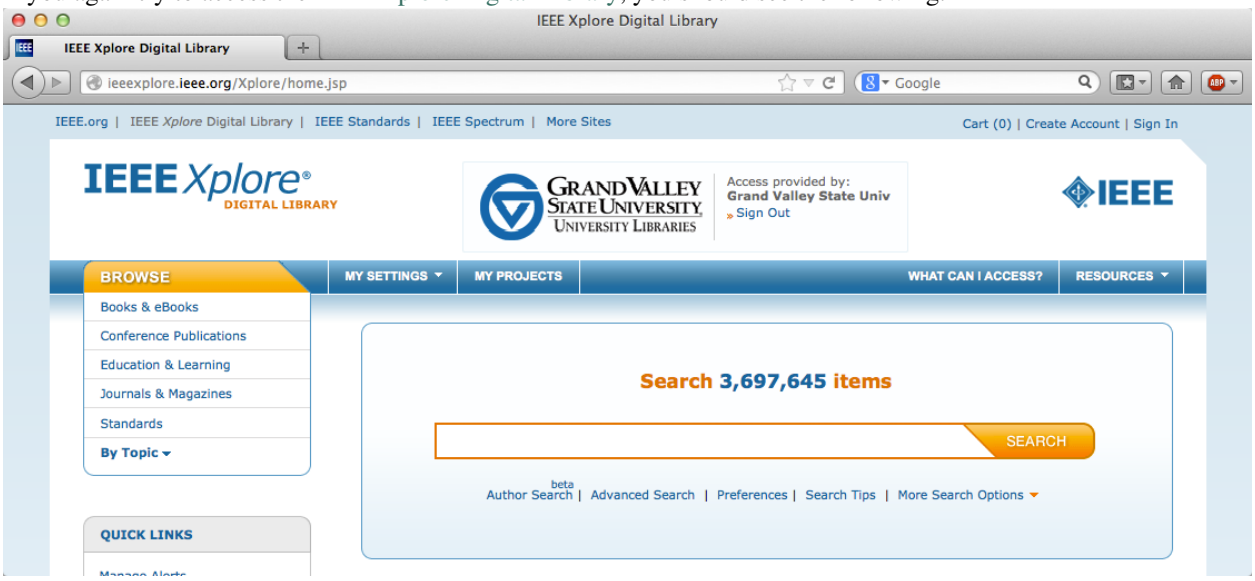
This establishes a SOCKS proxy with EOS01 served up on the local machine on port 5555. Now we must configure our operating system or browser to use this proxy.

Proxy configuration varies from distro to distro. These steps show how to configure a system-wide SOCKS proxy on Ubuntu 14.04, but other distros may be similar.

First, from the menu, select *System Settings* → *Network* → *Network Proxy*. For *Method*, select *Manual*, and under *Socks Host* enter `localhost` and `5555`. Then click *Apply system wide* and enter your password to turn on the proxy. To turn off the proxy, switch *Method* back to *None* and click *Apply system wide*.



If you again try to access the IEEE Xplore Digital Library, you should see the following:



You have now successfully used OpenSSH to establish a SOCKS proxy!

Warning: By using a SOCKS proxy, *all* your network traffic is sent through the proxy. This has two implications:

- Network access will likely be slower.
- GVSU will be able to monitor your traffic as they do when you are on campus.

Please keep this in mind when using the proxy feature.

Example

For an example OpenSSH configuration file, see Sean's SSH config.

4.3.5 Alternative Clients

We have tried various VNC clients, but found Remmina to be the easiest to use. However, other VNC clients for GNU/Linux exist and include:

- KRDC — free and open-source, part of KDE
- Vinagre — free and open-source, part of GNOME
- TigerVNC — command-line based, free and open-source
- RealVNC Viewer — free and paid versions available
- RealVNC Viewer for Google Chrome — free Google Chrome extension

Operation of each of these applications is similar. For the host, enter in the hostname of the EOS machine to which you have SSH'ed. If a display is requested, enter `0`; if a port is requested, enter `5900` (these mean the same thing). If the viewer offers support for multiple protocols, make sure you select "VNC".

Danger: Another popular client for SFTP is FileZilla. However, because FileZilla stores its passwords insecurely, we cannot recommend it. Please avoid its use to keep your password secure.

CHAPTER

FIVE

SHELLS

A shell is a textual user interface that allows you to control the operating system and run programs. The EOS Labs have a variety of different shell options from which you can choose. The default is Bash, but you may use any of the following provided shells.

5.1 Bash

GNU Bash (Bourne Again SHell) is setup for you automatically. It is the default shell for the GNU Project and is widely the standard shell for most distributions of GNU/Linux.

5.2 Tcsh

Tcsh (TENEX C SHell) is a C style shell that adds a few extra features such as command line completion.

5.3 Kornshell

Kornshell is a shell developed at AT&T Bell Laboratories. It has features of both the Bash and Korn shells, as well as additional features.

CHAPTER

SIX

USER-LEVEL SOFTWARE INSTALLATION

The EOS labs already have many packages installed to augment the base system. However, it is entirely possible that your work requires software which is not already installed. In this case, one option is to request the installation of this software from Ira Woodring. This is a good option if you feel the software would also be useful to others. If the software is primarily for personal use, however, you are advised to first attempt a user-level installation of the software.

Begin by reading about the *The Standard Hierarchy*, and then try your hand at *Manual Installation*. Once you have succeeded at installing the examples manually, give *Linuxbrew* a try.

6.1 The Standard Hierarchy

Before installing software on your own, it is important to understand the concept of the standard hierarchy. The standard hierarchy is a way of organizing files on the filesystem such that it may be used cohesively by multiple programs. If you are a Windows or Mac OS X user, you may be used to programs having their own subdirectories of `C:Program Files` or `/Applications`, respectively. However, UNIX-like machines do not typically work in this way, electing instead to separate installed files by type or purpose.

The root hierarchy, `/`, is the directory which contains all other files. Files which are absolutely essential to the system's operation may be installed here. The primary hierarchy for programs used by a typical user is the `/usr` hierarchy. Both these hierarchies can contain each of the following directories:

Name	Purpose
`bin`	Program executable files; *bin* stands for *binary* which is another name for *executable*
`include`	Include files (headers) for the C programming language
`lib`	Shared libraries, which frequently correspond to headers in the `include` directory
`etc`	Configuration files [2]
`src`	Source code for programs installed to said hierarchy
`man`	Manual pages for programs installed to said hierarchy

Although there are more directories which can be present in each hierarchy, these directories are the most important ones with which to be familiar.

Files in the root and `/usr` filesystems are usually readable but not writable by ordinary users. Ordinary users typically only have one directory to where persistent data can be written: their home directory. This is typically `/home/`*`username`*. As such, this is the place where a user installs their own programs. Although the system hierarchies cannot be written by a standard user, the structure of these system hierarchies is typically mirrored by hierarchies created in a user's home directory.

[1] Unlike most hierarchy directories which contain files related to other files in their hierarchy, configuration files in `/etc` are usually used to configure programs in many different hierarchies. Software configuration is a complex beast — consult each specific piece of software's documentation for the exact files used for configuration.

[2] Unlike most hierarchy directories which contain files related to other files in their hierarchy, configuration files in `/etc` are usually used to configure programs in many different hierarchies. Software configuration is a complex beast — consult each specific piece of software's documentation for the exact files used for configuration.

Programs typically look in the system hierarchies for programs, headers, libraries, configuration files, and other data files which they may use or need. Special considerations usually need to be applied in order to make programs compile and run correctly from a user's home directory.

For more information on the standard hierarchy, please see the very well-written Filesystem Hierarchy Standard, which is the source of most of this information.

6.2 Manual Installation

System package managers like Apt and Yum usually install pre-compiled software, binaries which are typically compiled on a build server infrastructure like Launchpad. Mac OS X and Windows users are also used to pre-compiled software, as most downloadable programs are distributed as application bundles or ready-to-run EXEs, respectively.

However, due to the myriad of different GNU/Linux distributions, software for GNU/Linux is often distributed in source code form only. This requires a potential user of the software to build the software from its source code, colloquially known as *compiling from source*. In addition, due to the lack of an accepted structure for installing user-level software, almost all software that is installed in a per-user fashion will need to be compiled from source.

6.2.1 Ready the Prefix

The hierarchy to which a program is installed is typically called its *prefix*. Most build systems will install by default to `/usr/local`, the system hierarchy typically used for custom-installed programs or programs compiled from source. However, because a standard user cannot write to this prefix, we are required to change our installion prefix to a directory within our home directory.

The prefix that we recommend for manually-installed user-level programs is `~/.local`, which is the hidden `.local` directory immediately within your home directory. This directory has some precedent, being used by Python and in the Freedesktop specifications. Setting this prefix is the primary step to successfully installing user-level software.

To ready your prefix for downloading and compiling source code, run the following command:

```
mkdir -p ~/.local/src
```

This command creates a `src` directory inside of your `~/.local` prefix. We will use this directory later for housing downloaded program source code.

6.2.2 Download and Extract

To compile and install software, you must first obtain the source code. The first step is to visit the project's web site and ascertain the location of its source code. The source code is typically distributed in a tar or zip archive, so look for files ending in `.tar.gz`, `.tar.bz2`, or `.zip`. After downloading, the files should be extracted to the source code directory, `~/.local/src`.

Example: GNU Bash and tar archives

Let's download the source code for GNU Bash, the default shell on the EOS system. First, begin by switching to the directory containing our source code:

```
cd ~/.local/src
```

The latest version of Bash at this time of writing is 4.3, so that is what we will download. Start by visiting the Bash home page. Under *Downloading Bash*, click the HTTP link. You will be taken to a directory index which contains a list of downloadable files. Scroll down to find a file named `bash-4.3.tar.gz`.

Although you can download this file directly using your browser, it is often easier to copy the URL and download on the command line. This is especially true if accessing EOS using SSH. Copy the URL by clicking *Copy Link Location* or similar in your browser, then download the source code using **wget**:

```
wget http://ftp.gnu.org/gnu/bash/bash-4.3.tar.gz
```

This should create a `bash-4.3.tar.gz` file in the current working directory. This file is a tar archive which has been compressed with the GNU zip compression algorithm. A file of this type typically has a `.tar.gz` or `.tgz` file extension and is colloquially known as a *tarball* [*tarball* refers to tar archives of any or no compression scheme].

The contents of this archive can be extracted using the **tar** program as follows:

```
tar -xf bash-4.3.tar.gz
```

Hint: **tar** can automatically detect the compression format by the extension, so passing the compression algorithm as you may see elsewhere is usually unnecessary.

Warning: Most source code tarballs are "well-behaved", meaning that they create one directory which matches the name of the tarball. This is a convention, but is by no means required. Make sure you trust the source of the tarball before extracting the files. You can view the contents of a tarball with the **less** built-in tar viewer:

```
less bash-4.3.tar.gz
```

This command should have created a `bash-4.3` directory in the current working directory. Make that directory your working directory:

```
cd bash-4.3
```

You are now in the root of the GNU Bash source distribution.

Hint: If you do not care about saving the original source tarball, you can download and extract simultaneously with:

```
wget http://ftp.gnu.org/gnu/bash/bash-4.3.tar.gz -O - | tar -xz
```

Note that you must pass the compression algorithm to **tar** because it is not able to detect the type by file extension when input is given through a pipe.

Example: libgit2 and zip archives

For our zip example, we will download the source code for libgit2, a linkable library providing Git functionality. First switch to the directory containing our source code:

```
cd ~/.local/src
```

The latest version of libgit2 at this time of writing is 0.21.1, so that is what we will download. Visit the download page for libgit2 0.21.1 and scroll down to copy the link for the source code zip archive. GitHub also offers tarball downloads, but we will use the zip for the purposes of this example.

Download the file with **wget** as shown in the earlier example:

```
wget --content-disposition https://github.com/libgit2/libgit2/archive/v0.21.1.zip
```

We use the `--content-disposition` flag here to tell Wget to honor the filename suggested by GitHub. This should create a `libgit2-0.21.1.zip` file in the current working directory. This file is a zip archive just like those you may have seen on your desktop operating system. This file can be extracted using the InfoZip **unzip** utility:

```
unzip libgit2-0.21.1.zip
```

> **Warning:** Unlike source tarballs, zip files sometimes have all files in one directory or sometimes have all files immediately in the root directory. Again, however, this is convention — make sure you trust the source of the archive before extracting the files. You can view the contents of a zip archive with **less** built-in zip viewer:
>
> ```
> less libgit2-0.21.1.zip
> ```

This command should have created a `libgit2-0.21.1` directory in the current working directory. Make that directory your working directory:

```
cd libgit2-0.21.1
```

You are now in the root of the libgit2 source distribution.

6.2.3 Build the Software

Almost all professional-grade software projects use a build system for compilation and installation. A build system automates the tedious task of constructing compiler commands and installing files to the proper places. Using a build system should not be viewed as running a program which automagically produces another program, but rather as a practical solution to a real problem.

There are several build systems used by typical software on GNU/Linux. Read the following sections to learn about the different build system and how to identify and use them.

Autotools

Autotools, also known as the GNU Build System, is the build system currently used by most programs on GNU/Linux. You can usually identify a program using Autotools by the presence of a `configure` script in the root of the source distribution.

The software which makes up Autotools itself is usually not necessary to build a program using Autotools as a build system. Instead, the functionality is embedded into the `configure` script itself. Autotools build systems typically only require the presence of Make.

Example: GNU Bash

An example of a piece of software that uses Autotools is GNU Bash, the subject of our earlier example. We will compile the version of GNU Bash that we extracted earlier. Start by switching to the source code root directory if not already there:

```
cd ~/.local/src/bash-4.3
```

The next step is to create the build directory, which we'll create inside the source directory for convenience:

```
mkdir build
cd build
```

Now, we must configure the software by running the `configure` script. It is to the `configure` script that we must also pass the all-important `--prefix` option. Run the following:

```
../configure --prefix ~/.local
```

You will see many lines printed to the terminal, which is the script doing various checks on the system and compiler and adjusting the build to our specific system.

`configure` scripts typically also accept a myriad of other options, which can be viewed with:

```
../configure --help | less
```

Passing other options is typically unnecessary unless you would like to customize the build. Piping to **less** is recommended due to the usual length of the output.

> **Warning:** Note that:
>
> ```
> ../configure --prefix=~/.local
> ```
>
> will *not* work, as Bash will not expand the tilde properly unless the path is its own argument.

Important: Many build systems (including Autotools) support both *in-source* and *out-of-source* builds. In-source builds take place when the `configure` script is run in the same directory as the source code, that is:

```
./configure
```

Running the `configure` script in any other directory is referred to as an out-of-source build. Out-of-source builds are generally preferred because they allow separation of build artifacts from the source code. However, not all build systems or projects support out-of-source builds. The build illustrated in this example is an out-of-source build.

After configuring the software, it is time to build. This can be accomplished with:

```
make
```

Running this command typically produces an avalanche of output. The lines that you see printed are primarily compiler commands, which are printed as they are being run.

After running this command, you should have a workable version of the Bash shell. Test this out by running:

```
$ ./bash --version
GNU bash, version 4.3.0(1)-release (x86_64-unknown-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

The final step is to install the files generated by the build. Do this with:

```
make install
```

GNU Bash has now been installed to your home directory! Continue reading to find out how to make your locally-installed version override the system version.

CMake

CMake is a popular alternative to Autotools as a build system on GNU/Linux. You can usually identify a CMake build system by the presence of a `CMakeLists.txt` file in the root of the source distribution.

The **cmake** program needs to be installed in order to build projects using CMake as a build system. It is installed by default on EOS.

Example: libgit2

An example of a project that uses CMake as a build system is libgit2, the subject of our earlier example. We will compile the version of libgit2 that we extracted earlier. Start by switching to the source code root directory if not already there:

```
cd ~/.local/src/libgit2-0.21.1
```

The next step is to create the build directory, which we'll create inside the source directory for convenience:

```
mkdir build
cd build
```

Now, we must configure the software by running CMake. Similar to the `configure` script, we tell CMake the install prefix at this stage. Run the following:

```
cmake -DCMAKE_INSTALL_PREFIX="$HOME/.local" ..
```

You will see various checks on the system and compiler printed to the terminal as with Autotools.

After configuring the software, it is time to build. This can be accomplished with:

```
cmake --build .
```

During the build, CMake will display which file is currently being built along with a percentage of files built on the left.

After running this command, you should have a built version of the libgit2 shared library. Because libgit2 is a library, there is really no executable to run, but we can verify the file type with:

```
$ file libgit2.so.0.21.0
libgit2.so.0.21.0: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, Buildl
```

The final step is to install the files generated by the build. Do this with:

```
cmake --build . --target install
```

libgit2 has now been installed to your home directory, and can be used in your C programs if you so desire.

Other Build Systems

The majority of C and C++ software that you may want to install to your EOS account likely uses Autotools or CMake as its build system. For those that don't, we recommend consulting the project's `README` or `INSTALL` file or the project's documentation or website for compilation instructions.

6.2.4 Adjusting the Environment

Executable Path

You can always use executables installed to your home directory by typing the full path to the executable, for example:

```
$ ~/.local/bin/bash --version
GNU bash, version 4.3.0(1)-release (x86_64-unknown-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

For obvious reasons, typing the full path can get tedious if you are using the executable frequently. In addition, other utilities may assume that the executable in question is available on the `PATH` and not in a custom prefix.

If you are installing an executable that is already present on the system, there is another consideration — you may want to override the system version with the version that you installed to your home directory. This is typically useful if you would like to use a newer version of a program than one installed to a system hierarchy.

To illustrate this, note that when typing:

```
$ which bash
/usr/bin/bash
```

the shell will still default to using the system Bash, which happens to be `/usr/bin/bash`.

To resolve both of these issues, we can add the executable's parent directory to executable search path, stored in the environment variable `PATH`. Open your `~/.bash_profile` in an editor and add the following line to the end:

```
export PATH=~/.local/bin:$PATH
```

This line prepends the path of your locally-installed executables to the executable search path. Your executable will now not only be accesible without typing the full path, but it will also override any executables of the same name in system hierarchies.

Restart your shell to effect the changes to your `~/.bash_profile` *by logging out and logging back in.* After logging back in, the following should yield:

```
$ which bash
~/.local/bin/bash
```

Now you should be able to simply type:

```
bash
```

to start the GNU Bash installed to your home directory!

Man and Info Paths

Although you are now able to run your new Bash without typing the full path, the commands:

```
man bash
info bash
```

still show the Bash documentation for the system Bash. Although this may not seem like a big deal, small changes between versions of the same program can be the difference between an working and non-working script. To allow **man** and **info** to find locally-installed documentation, add the following lines to your `~/.bash_profile`:

```
export MANPATH=~/.local/share/man:~/.local/man:$MANPATH
export INFOPATH=~/.local/share/info:$INFOPATH
```

There is unfortunately some inconsistency with the location of installed man pages, which why we added both directories to the `MANPATH`. `INFOPATH` does not have these problems.

After restarting your shell, the commands at the beginning of this section should bring up the correct documentation.

6.2.5 Library Dependencies

Bash and libgit2 both compile without issue on EOS. However, programs frequently have compile-time dependencies: libraries which need to be installed before compiling the program.

As with the project itself, one option is to ask the Ira Woodring to install the library for you. If you would like to compile and install the dependency on your own, it is possible, but is currently out of the scope of this guide. Here are some hints:

- When compiling the program, you may need to set the `CPPFLAGS` and `LDFLAGS` environment variables to allow the compiler to locate headers and libraries, respectively. See the Autoconf manual on Preset Output Variables for descriptions of each of these variables. Some build systems are able to locate headers and libraries automatically in the specified install prefix.
- If you installed the libraries to your home directory, the operating system will not know to search for them there when running a program (even if that program is in your home directory). To allow the program to find its shared library dependencies at runtime, you must either set its rpath (recommended) or use the `LD_LIBRARY_PATH` environment variable (not recommended). See the following links for hints on this topic:
  - Russ Allbery's notes on Shared Library Search Paths
  - The Autoconf manual on Preset Output Variables
  - The Wikipedia entry on rpath

  You can see the default paths in which the system looks for libraries by running:

  ```
  ldconfig -v | less
  ```

6.2.6 Conclusion

As you can see, manual installation of programs is a complex but predictable process. This is where package managers like *Linuxbrew* become useful.

6.3 Linuxbrew

Linuxbrew is a package manager for GNU/Linux systems. The main advantage of Linuxbrew over system package managers like Apt and Yum is that it allows installation of software on a per-user basis. Linuxbrew is a Linux port of Homebrew, the popular package manager for Mac OS X. As such, some of its packages still contain Mac-specific code or do not yet build on GNU/Linux. Your mileage may vary, but in general Linuxbrew works quite well.

Important: Before using Linuxbrew, please make sure that you are comfortable with compiling and installing software manually. Although Linuxbrew generally makes installing user-level software much easier, there is no magic — it performs the same steps as you would during a manual install. When Linuxbrew does not behave as intended, you will need knowledge of manual installation to fix the problem. What we are saying is this: do not come to Ira Woodring with Linuxbrew package installation issues unless you have first tried to compile the software on your own.

The Linuxbrew dependencies *should* already be satisfied, so you will be able to install without issue. If they are not, please talk to Ira Woodring. To install, then, please follow the installation instructions on the homepage.

After installation, run the following to uncover possible issues that you may have when installing packages:

```
brew doctor
```

Before moving forward, do your best to correct any issues reported by this command.

Installing packages with Linuxbrew is quite easy. For example,

```
brew install tmux
```

installs the latest version of tmux, the terminal multiplexer.

In this one command, Linuxbrew does a lot for you. It first installs tmux's dependency, libevent. Then it configures, builds, and installs tmux, setting the prefix to the correct location automatically. Furthermore, it sets the `tmux` executable's rpath (see *Library Dependencies*), meaning that the executable will automagically find the necessary libraries within your Linuxbrew prefix:

```
$ patchelf --print-rpath ~/.linuxbrew/bin/tmux
/home/smithj/.linuxbrew/lib
```

This command uses patchelf, which can also be installed using Linuxbrew ;)

Enjoy installing packages using Linuxbrew!

CHAPTER

SEVEN

WINSERV

Part of the department infrastructure is a Windows Server installation, called Winserv. Accounts for this machine are given as necessary — for example, when taking a course that involves projects which require the Windows platform. This server is named `winserv.cis.gvsu` and can be accessed via the Remote Desktop Protocol (RDP). The following are methods for accessing Winserv from various RDP clients.

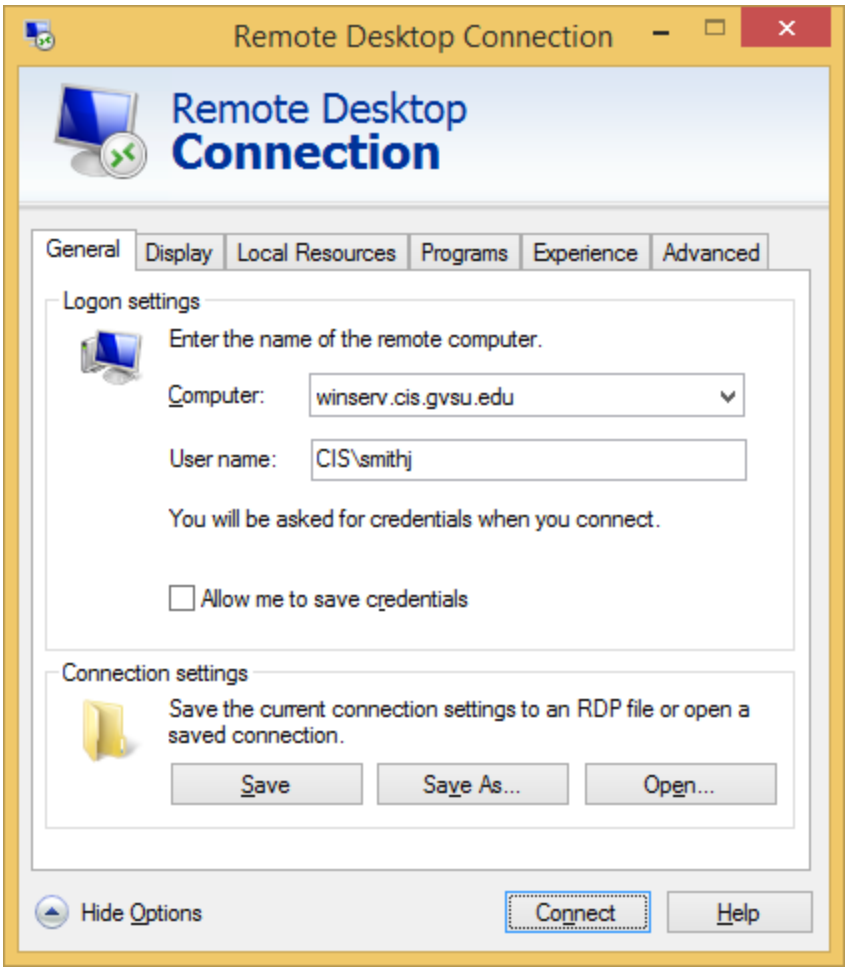
7.1 Common Settings

Regardless of the RDP client or platform you use, please remember a few things:

- The machine's IP address may change; use the DNS name instead.
- If outside of the EOS network, you will need to use the fully qualified domain name, `winserv.cis.gvsu.edu`. While on the EOS network, you can simply use `winserv`.
- You must login to the CIS domain. For instance, if your username is `smithj`, your login would be `CIS\smithj`. These may be specified together with the backslash or separately depending on your client.
- Our certificate is self-signed. You may want to instruct your client to save this information, or you will have to accept a security warning each time you login.

7.2 Microsoft Windows

All versions of Windows have the built-in Microsoft Terminal Services Client. You can find this program by searching for **mstsc** or *Remote Desktop Connection* in the Start Menu. Here is a sample configuration for Winserv:

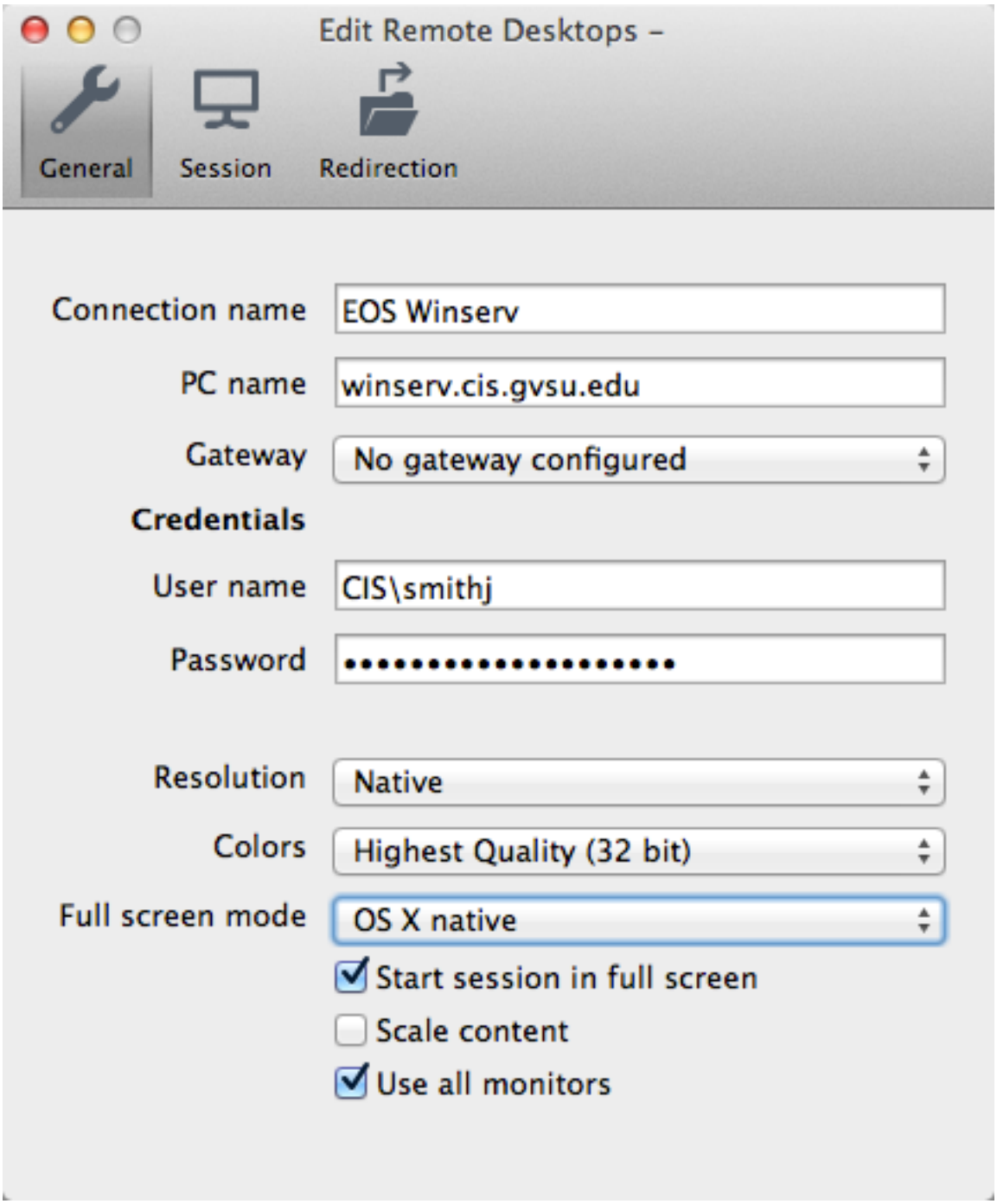


7.3 Mac OS X

Multiple options exist for Mac OS X, including the official Microsoft client and an open-source client called CoRD. Both work well and provide a similar set of features, so it is up to you which one you'd like to use. See here for a comparison of three different options including the two just mentioned.

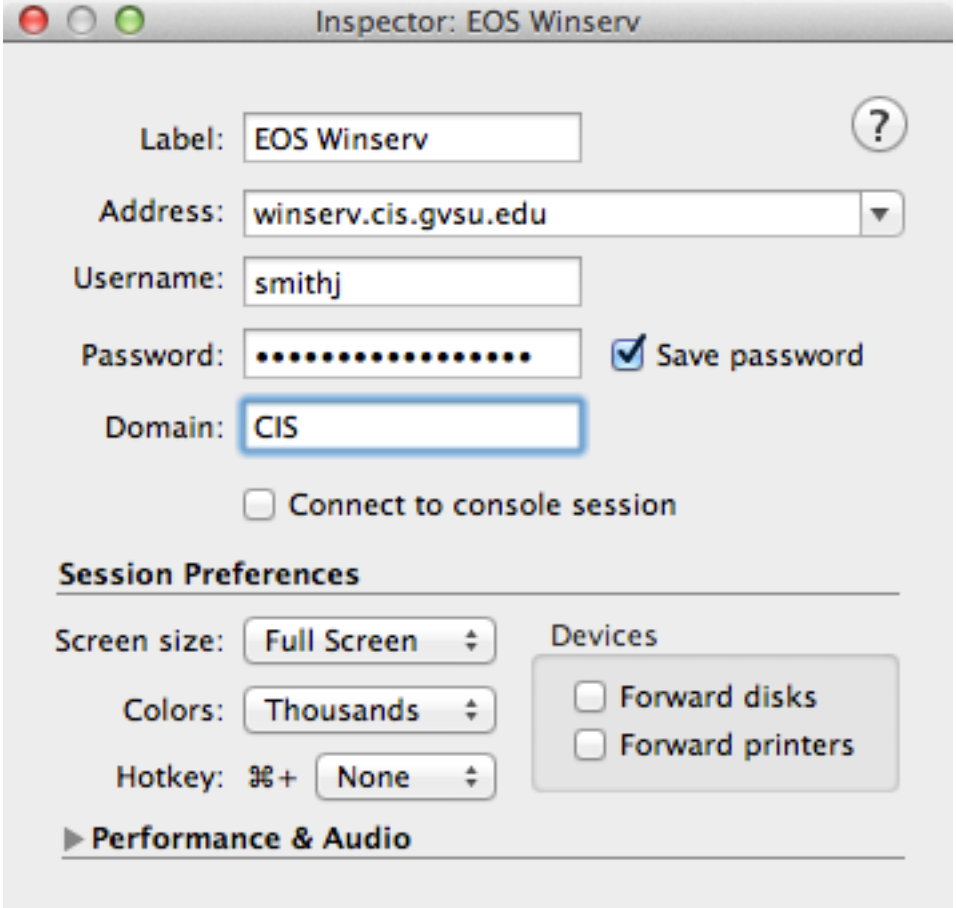
7.3.1 Microsoft Remote Desktop

This application is available as Microsoft Remote Desktop in the Mac App Store. *Do not* attempt to download Microsoft Remote Desktop Connection Client for Mac 2.1.1, as it does not work for more recent versions of Mac OS X. You do not need to configure a gateway in order to use this client with Winserv. Here is a sample configuration:



7.3.2 CoRD

CoRD is an open-source rdesktop-based RDP implementation for Mac OS X. Although their website states that CoRD "doesn't seem to work on 10.9 Mavericks", we have had no issues with the latest version. Here is a sample configuration for Winserv:



7.4 GNU/Linux

GNU/Linux systems usually have the rdesktop command-line tool in their repositories. If not, it can most likely be built from source. rdesktop provides a geometry flag that accepts both screen percentages or resolutions from the command-line to help adjust the client to an appropriate size. For instance, to allow the client to take up 90% of your screen:

```
rdekstop winserv.cis.gvsu.edu -g 90%
```

Alternatively, to force a resolution of 1024x768 pixels:

```
rdekstop winserv.cis.gvsu.edu -g 1024x768
```

For a graphical RDP client that can also be used for *VNC*, check out Remmina.

CHAPTER

EIGHT

DATABASES

Learning to setup and maintain a database is essential to any Computer Science curriculum. The EOS infrastructure supports multiple database variants depending on the your needs. To save space and setup time database accounts are only given to those who need them for a course or a specific project. Contact your professor and Ira Woodring if you need a database account for any reason.

8.1 MySQL

MySQL is a powerful open-source database. To access it via the command-line, login to an EOS/Arch machine and enter:

```
mysql -u smithj -p -h cis.gvsu.edu
```

This attempts to log you in with the provided username, using a password, to the host cis.gvsu.edu (our MySQL server).

8.2 Oracle

Oracle is a very powerful and complex enterprise quality system. Once you have been granted access, you can access it with the command:

```
sqlplus smithj@orcl
```

Please note that when you change your password for Oracle that you must not use the `@` character. Oracle will accept this but you will be unable to login.

8.3 Oracle APEX

Oracle also provides the APEX system for web based database development. An APEX account is separate from the normal Oracle account; a password for one will not work for the other. You may login to APEX once you have been granted access by opening your web browser to the URL

http://dbserv.cis.gvsu.edu:5560/apex

You will need to provide a workspace name, username, and password. If your username is `smithj`, a sample login would be

Workspace smithj_ws

Username smithj

Password **********

Do not attempt to use the *Reset Password* feature on the APEX homepage; it has never worked properly. If you attempt to use it you will be unable to login until a system administrator can delete and recreate your account.

8.4 MSSQL

Microsoft also provides an enterprise quality database server that we provide. Microsoft's database is called MSSQL. We host MSSQL on the Winserv machine, and accounts are granted when needed.

8.5 SQLite

The SQLite system is a relational database that can exist within your home directory. SQLite is different from the above mentioned databases in that it does not operate as a client/server set of processes, but instead can be linked to the application being programmed. As many databases as need be created (within storage limits) can be created by you, as each database is merely a separate file on the filesystem.

Outside of a programming context, SQLite can be accessed from the commandline with:

```
sqlite3
```

This will provide you with a command-line interface from which you can work using SQL statements.

8.6 Remote Database Connections

It is often advantageous for programs to connect their programs to databases to do work. There are a variety of ways to accomplish this task, and many are language specific. It is of note though that our databases are not accessible from outside of our network due to firewall restrictions. However, programs running from within the EOS infrastructure can make connections to databases.

CHAPTER

NINE

INTEGRATED DEVELOPMENT ENVIRONMENTS

Programming is often aided by the use of an integrated development environment (IDE). The EOS system has several IDEs to assist in programming tasks.

9.1 BlueJ

BlueJ is an IDE designed with learning in mind. While it is not the best choice if you are an advanced user or have a large project, it is one of the best IDEs for learning to program in Java. BlueJ's object inspector makes it quite easy for you to examine objects as they progress through a software instance's life cycle. BlueJ may be accessed by typing **bluej** from the command line.

9.2 Eclipse

Eclipse is a powerful, extensible, free and open-source IDE that can be used with a large number of languages. By default, Eclipse on EOS supports Java, PHP, Python, C, C++, and Android. Support for other languages or projects can be easily added if needed. Eclipse may be accessed by typing **eclipse** from the command line.

9.3 IntelliJ

For those that don't like Eclipse, we also provide the IntelliJ Java IDE. IntelliJ is a professional quality IDE and rather expensive to purchase. IntelliJ may be accessed by typing **idea.sh** from the command line.

CHAPTER

TEN

ENVIRONMENT VARIABLES

This page contains a list of all environment variables referred to in the manual.

`PATH`
: The GNU/Linux search path for executable files.

`MANPATH`
: The search path for manual pages readable by **man**.

`INFOPATH`
: The search path for manual pages readable by **info**.

`CPPFLAGS`
: Flags given to the C pre-processor during compilation. See the Autoconf manual on Preset Output Variables.

`LDFLAGS`
: Flags given to the linker during compilation. See the Autoconf manual on Preset Output Variables.

`LD_LIBRARY_PATH`
: Additional paths in which the dynamic linker should search for shared libraries. See Russ Allbery's notes on Shared Library Search Paths, the Autoconf manual on Preset Output Variables, and the Wikipedia entry on rpath.

CHAPTER

ELEVEN

CONTRIBUTING TO MASTERING EOS

This manual is a living document. It is maintained by its authors, but students (you!) are encouraged to contribute to the guide. The manual is written by GVSU CIS students for GVSU CIS students.

The easiest way to to contribute is by reporting an issue or requesting a section with our issue tracker. One of the authors should respond to your issue and give feedback.

The Mastering EOS manual is written using Sphinx, an excellent multi-target documentation generator. The documentation is written in the markup language reStructuredText. The poster is written in LaTeX using Beamer and beamerposter, although contribution to that is not necessarily as useful for obvious reasons. All of our tooling is written in Python. Although these tools may seem overwhelming, intimate familiarity with them is not necessary for contribution.

The source code for Mastering EOS is hosted on GitHub. GitHub is a popular site for hosting code repositories using Git, a popular version control system. If you are familiar with the GitHub contribution process, contributing to Mastering EOS is exactly the same.

For those unfamiliar with Git and GitHub, contributing to Mastering EOS may seem rather involved. However, by following the steps outlined, it should be rather straightforward. We request that you first give it a try on your own, but failing that, please don't hesitate to contact the authors. We want your contributions!

To contribute to Mastering EOS, follow these steps:

- Read GitHub's guide to Contributing to a Project.
- Set Up Git if you have not already done so.
- Fork the Mastering EOS repository and make your changes.
- Create a pull request to the original repo containing your changes.

An author should respond to your pull request, and with any luck, your changes should go live! We look forward to seeing your contributions!

11.1 Writing Style

When writing technical documentation, it is important to follow a consistent writing style. Within this manual, we attempt to follow the OpenStack writing conventions. Their documentation presents a great summary of how to write coherent technical documentation.

Sphinx uses SmartyPants to transform quotes, dashes, and ellipses into typographically correct entities for HTML output. You can use straight quotes, straight apostrophes, and three dots — they will be transformed into the correct characters. For dashes, first read up on the three types of dash. The transformations are as follows: Hyphens stay as is, two hyphens will be transformed into an en dash, and three hyphens will be transformed into an em dash.

Please also see the following links for writing technical documentation:

- ACS Distance Education Guidelines for Technical Writing
- Novell Open Source Documentation Style Quick Start
- BlueBream Documentation Guidelines
- Description of Imperative Mood on Wikipedia