Working from Home

Tools and Technologies for Improving Your Programming Environment

Daniel J. Hood
danielhood@umbc.edu
Overview

- Tools for MS Windows (and others OSs)
  - SSH
  - SCP
  - X Windows
  - Tunneling X Connections
  - XEmacs for Windows
  - AFS Clients
  - Cygwin

- Options for Running Linux (without giving up MS Windows)
  - Dual Booting
  - Virtualization
  - Bootable Linux Distributions

- Conclusion
Tools for MS Windows (and others OSs)

- This first group of tools is geared towards augmenting MS Windows
- The goal here is to work in MS Windows, not something else
Accessing Linux Servers Remotely

• Most people do not want to sit in the lab to work all of the time
• Want a mechanism to work remotely
• Connect to remote server to work
• Common mechanism to do this is by using an SSH client
SSH Overview

- SSH stands for Secure SHell
- Secure alternative to telnet
- Provides secure terminal or shell access to some remote computer
SSH Clients

• There are many SSH clients
  – PuTTY
    • My favorite
  – Tera Term SSH
    • What OIT distributes
  – SSH.com
  – OpenSSH
  – Others…
PuTTY In Action

UMBC Office of Information Technology  http://www.umbc.edu/oit
_______________________________________________________
If you have any questions or problems regarding these systems, please send mail to "systems@umbc.edu", or call the OIT Helpdesk at 410-455-3838.

ATTENTION STUDENTS:
Please DO NOT RUN, TEST, OR DEVELOP your shell or applications which utilize the fork() system call on the shared-access remote login systems (the GL servers.) Please limit this work to lab workstations or your personal systems.

*** ALSO ***

The public network is not a place to test or develop your broadcast or multicast-based applications. These should be developed in an isolated environment. Do not run these applications on the linux.gl servers.

This server has been upgraded to Red Hat Enterprise Linux 5. GSSAPI Authentication to SSH is currently disabled as we work out some issues. Please see the release notes for more information:


linux3[1]:
Accessing Files Remotely

- Additionally, many people may want to transfer files back and forth between GL and their home machines
- The common way to do this is by using an SCP client
SCP Overview

- SCP stands for Secure Copy
- Secure alternative to FTP
- Provides secure copy mechanism to/from some remote computer

Windows Client → Files copied to/from remote server → Linux Server

scp foo.c

file foo.c
Likewise, there are many SCP Clients

- **WinSCP**
  - This is what OIT distributes, my favorite
- **CuteFTP**
  - Can also do secure copying
- **SSH.com**
  - Also acts as an SCP client
WinSCP In Action
Frequently students want to access remote applications which are graphical, such as…

- XEmacs
- DDD
If we try to run XEmacs over SSH it does not usually work as desired

- Either defaults to text mode or errors out
X Windows Overview

• The X Window System (X for short) is the standard GUI on UNIX systems
• X provides framework for drawing, moving and interacting with windows
• X features network transparency
  – The machine where program (the client application) runs need not be the local machine (the display server)
SSH to linux.gl.umbc.edu, try to run xemacs, get the infamous…
  “Error: Can't open display:”
To get these commands to draw X windows we need to configure our SSH client to request them.

Most SSH clients have an “X Forwarding” section where you can usually just check a box.

Most command line SSH clients support this feature with the “-X” option.
Enabling X Forwarding in PuTTY
Tunneling X Connections

• Alright, the server is now returning X connections, but there’s still an error…
  – "X connection to linux1.gl.umbc.edu:16.0 broken (explicit kill or server shutdown)"
X Windows vs. Microsoft Windows

- Windows and X do not play nice
  - Windows has no clue how to go about interpreting the language that X uses
- Solution is a X server for Windows
  - Interprets the X server language, translates the commands into the Windows language, and acts upon them
There are many choices when it comes to X Servers for Windows, some of the more common ones…

- **Xming**
  - Lightweight

- **Cygwin/X**
  - Optionally part of Cygwin installation

- **Exceed**
  - Widely used (commercial)
Successful X Forwarding

• The full puzzle...

SSH to Linux computer, requests X connections be tunneled back

SSH forwards X connections back

X Server

X Server listens for X connections and acts as the display for remote clients

Client
Xming & Tunneled X Connections In Action

```
#include <stdio.h>

int main(int argc, char** argv) {
    printf("Hello X-Tunneled World!\n");
    return 0;
}
```

```
linux3[1] 🞣 xemacs hello-world.c &
  [1] 12151
linux3[2] 🞣 xemacs: /usr/lib/libdb.so.3: no version information available (required by xemacs)

linux3[2] 🞣 gcc hello-world.c
linux3[3] 🞣 ./a.out
Hello X-Tunneled World!
linux3[4] 🔎
```
Downsides to X Forwarding

• Tunneling X connections does have its drawbacks…
  – Resource intensive for the server
  – This approach eats up a lot of bandwidth, as a lot of data is needed to draw the windows
    • If connection is not really fast, this can result in a very sluggish experience
    • Not really usable on anything less than broadband
Another approach would be to actually install some of the common apps you use on the server on your computer.

Can install a number of applications locally, including editors such as XEmacs.
• **XEmacs** is not just for UNIX
• Official ports are available for:
  – MS Windows, Mac OS X, etc…
  – Can install native version of XEmacs under MS Windows to edit files locally
**XEmacs Under MS Windows**

XEmacs 21.4 |patch 21| "Educational Television" (win32) of Sun Oct 07 2007 on V9\* SHELTON-PC2

`C-` means the control key, `M-` means the meta key

Information, on-line help:

XEmacs comes with plenty of documentation...

M-? F: read the XEmacs FAQ (a capital F!)
M-? t: read the XEmacs tutorial (also available through the Help menu)
M-? i: get help on using XEmacs (also available through the Help menu)
M-? I: read the on-line documentation

M-x describe-project: read about the GNU project
M-x about-xemacs: see who’s developing XEmacs

Raw-----XEmacs: *scratch* (Lisp Interaction)-----All----------------------
Accessing Remote Files Locally

• If we do install something like XEmacs locally, then we need a way to edit our files
  – Could SCP them back and forth
    • Tedious
    • Can get out of sync
  – Better if they “appeared” local
• AFS is the file system which is used at UMBC to house your files on GL
• What is AFS?
  – AFS is short for the Andrew File System
  – Basically AFS is distributed network file system
• Files are stored on remote servers and clients access the files using an AFS client
There are AFS clients available for many operating systems
  – Windows, Mac OS X, Linux, Solaris, IRIX, etc…

Much more convenient to access your Linux home directory like a local drive
Open AFS

- There are several AFS clients, though I would recommend **Open AFS**
  - It is what is used here in the labs to map your UNIX home directory to the Windows S:\ drive
Open AFS In Action
Open AFS, XEmacs & PuTTY in Action

```c
#include <stdio.h>

int main(int argc, char* argv) {
    printf("Hello AFS World!\n");
    return 0;
}
```

```
linux1[4] ls
Desktop Mail bin hello-world.c tmp www
linux1[2] gcc hello-world.c
linux1[3] a.out
Hello AFS World!
linux1[4]
```

```
\"\n```
Total Stand-Alone Solution

• Some might prefer to completely work at home and when completed SCP their work to GL for submission
• Might want something that does not rely on Internet connection
  – Good for something like dialup
  – Or, if you just want to operate in a disconnected mode
Cygwin Overview

- **Cygwin** is a Linux-like environment for Windows
- Provides a collection of Linux-like tools, including…
  - A Linux shell
  - Compiler
  - Editors
  - X-Server
  - Man pages
Many UNIX applications are bundled with Cygwin — listed is a very small subset of these applications

- Development
  - ddd, cvs, gcc/g++, gdb, indent, make, nasm, rcs,
- Editors
  - emacs, xemacs, vi, nano
- Interpreters
  - python, perl, ruby, clisp
- Shells
  - bash, tcsh
- Others
  - man, xterm, xorg X server, diff, grep, tar, gzip, scp, ssh
Cygwin Motivations

• Cygwin allows you to develop in an environment completely detached from UMBC
  – All work is being done on localhost
  – SCP work over when done
• Use UNIX utilities (ls, grep, etc…) under Windows
• Can write applications that run on Windows
  – You are now generating a.exe instead of a.out
Cygwin Cautions

• Almost all instructors require your project to run on linux.gl.umbc.edu
• There may be some subtle differences between Cygwin and the Linux servers
  – Such as random number generation
• Always check your progress periodically against linux.gl.umbc.edu, to be sure your program works as expected on GL
• Make sure you backup your work (SCP to UMBC periodically or copy via AFS), this way you do not lose your work if your computer breaks down
Cygwin Installation

• The Cygwin installation uses an installer that allows you to choose from a variety of packages to download and install from the Internet

• Packages of interest include
  – Development
  – Editors
  – Shells
  – X if you want a local X server
When you open a Cygwin shell, you be put in your Cygwin home directory:
   – /home/username/

This is actually mapped to some directory in the Windows file system where Cygwin stores your files

You have still have access to all of you drives under the Cygwin path:
   – /cygdrive/c/Document and Settings/
   – “c:\Documents and Settings\”
Cygwin In Action

```c
#include <stdio.h>

int main(int argc, char** argv) {
    printf("Hello Cygwin World\n");
    return 0;
}
```

```
$ cd /cygdrive/c/Documents and Settings/Dan/Desktop/
$ gcc hello-world.c
$ .a.exe
Hello Cygwin World
```
Running Linux (without giving up MS Windows)

- This next set of options focuses more on actually running and working in Linux, rather than MS Windows
- Goal on all of these options is for both MS Windows and Linux to co-exist
  - You do not need to abandon MS Windows
Dual Booting

- The first option is to dual boot your machine like in the labs
- When you power on the computer you can choose between MS Windows or Linux
Dual Booting Installation

• If you have un-partitioned disk space you’re good to go…
  – Download, burn a Linux distribution, boot from CD/DVD and install
  – Most installers have an option to just use this un-partitioned space and leave MS Windows alone
  – Will install a boot menu which allows you to choose which OS to boot when you power on your computer
  – As always, good idea to backup first in case something goes awry
Dual Booting Installation

• If you do not have un-partitioned space, but you do have free disk space in Windows…
  – Can use a free application like GParted to take that space away from MS Windows and make it un-partitioned again
  – Again, good idea to backup first in case something goes awry
GNU GRUB version 0.97 (638K lower / 260832K upper memory)

Fedora (2.6.21-1.3194.fc7)
Other

Use the ↑ and ↓ keys to select which entry is highlighted. Press enter to boot the selected OS, 'e' to edit the commands before booting, 'a' to modify the kernel arguments before booting, or 'c' for a command-line.
If you do not have free disk space another option may be a bootable Linux distribution

- Does not install anything what-so-ever
- Uses the hardware in your computer, leaves hard drives untouched
- Makes for a nice way to experiment with Linux – no risks, just reboot
Bootable Linux – Live CDs/DVDs

• Many of these bootable distributions of Linux are run from an optical drive
  – Download and burn CD/DVD
  – Reboot computer, selecting optical drive as boot option
  – Many will recognize hard drive, but do not do anything to it

• Since CD/DVD is read-only, where do we save?
  – Some support writing back to the HDD
  – Frequently a flash drive can be used
Bootable Linux – Flash Drive

• There are also versions which can be booted and run from a flash drive
• Since this media is read/write, you can have a Linux OS and all your work on a small flash drive
• If you go this route, make sure your computer can boot from a USB drive
  – Look in your BIOS
Fedora Developer Spin Live DVD

```
#include <stdio.h>

int main(int argc, char** argv) {
    printf("Hello Live CD/DVD World\n");
    return 0;
}
```

```
[fedora@localhost ~]$ emacs hello-world.c &
[1] 3151
[fedora@localhost ~]$ gcc hello-world.c
[fedora@localhost ~]$ ./a.out
Hello Live CD/DVD World!
[fedora@localhost ~]$`

```
Virtualization

• Another path would be to run Linux in a virtual environment
• There are software applications which mimic a real computer that allow you to run an OS inside of them
• Common virtualization software
  – VMware Player, VMware Server
  – Qemu
VMware Player & Ubuntu In Action

```c
#include <stdio.h>

int main(int argc, char** argv) {
    printf("Hello Virtualized World!\n");
    return 0;
}
```

```
jars@jars-desktop:~$ gcc hello-world.c
jars@jars-desktop:~$ ./a.out
Hello Virtualized World!
```

To direct input to this virtual machine, press Ctrl+G.
• You've just seen a lot of techniques, which is right for you?
• Depends on a lot of things, what are your constraints?
Slow Internet Connection

• Goal would be to minimize or eliminate need for network connectivity

• Reasonable options:
  – Cygwin + XEmacs
  – Dual boot Linux/MS Windows
  – Bootable Linux distribution
  – Run Linux in a virtualized environment
Low on Disk Space

• Goal would be to keep installation footprint to a minimum
• Reasonable options:
  – Bootable Linux distribution
  – SSH + X Server
  – XEmacs + Open AFS + SSH
Not a Fast CPU / Not a Lot of RAM

• Goal is to minimize resources (primarily CPU and RAM) being used

• Reasonable options:
  – Dual boot Linux/MS Windows
  – Bootable Linux distribution
  – SSH + X Server
  – XEmacs + Open AFS + SSH
  – Cygwin + XEmacs
Questions