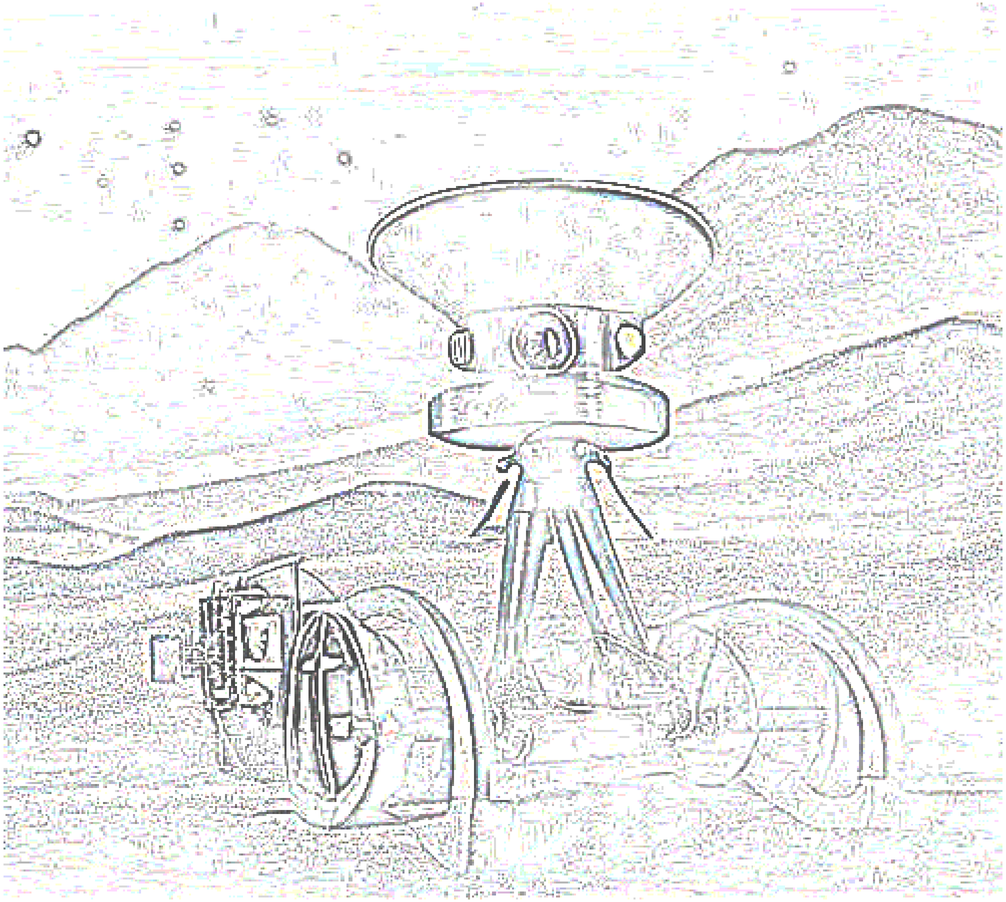
FlexBerry



AI art via nightcafe.studio for “Spectrograph Telegraph Wonder”

Preamble

The cover art was generated at https://images.nightcafe.studio/ using the keywords “Spectrograph Telegraph Wonder”. It was further processed in a way to reduce toner use with Gimp. Presented here is the AI Generated first cover art I’ve ever used for my notes!

FlexSpec1 Raspberry Pi Startup Guide

FS1 Team

Friday 30th August, 2024

Abstract

Acquire the latest image of Ubuntu 22.04 from Canonical for the Raspberry Pi. Burn the ISO image onto a high quality SD Card. Start the system and 'take the quiz': language, keyboard, geographic location, user name and password, machine name etc.

Follow the FollowMe.sh script to do a full installation.

In this case, the user is fred, the machine runs DHCP and has the name of pier15. See Section A

Instructions for installing the IRAF Community 2.17 and Pyraf3 are included as well.

Contents

Overview 1

1.1 Cast of Characters 3

2 The FlexBerry in action 3

2.1 The Quiz 4

3 Scenario 4

4 Nginx Install 4

4.1 Nginx Administration 5

A Names and Places 5

B Components of FlexBerry 5

C Certificates of Authority 7

D CertMe.sh 7

E Filesharing 7

E.1 NFS 10

E.2 Flex Nginx Configuration 11

F PuTTY and Xming on Windows 12

References 16

Index 16

extcolorred Action Items: 16

List of Figures

1-1 Initial Commands. Bring install to current released content. 2
1-2 Add System User, get the FlexSpec1 git repository, run the script. 2
D-3 Nginx Files 8
D-4 nginx sample file 9
F-5 XMing screen snaps of the 4 configuration menu panels. 14
F-6 PuTTY screen snaps of the 4 configuration menu panels. 15

List of Tables

Overview

This FlexBerry implementation uses Ubuntu 22.04 and NOT the Raspbian image based on Ubuntu programs. The prescription for making this system is contained in the GitHub repo TheSMTSci/FlexBerry repo. Simply visit: `https://github.com/The-SMTSci/FlexBerry` and follow the instructions in the ReadeMe section of the first page.

The process of making an `iso` file is tedious, onerous and cabable of damaging a system's file system in a very permenant way. The approach here is to "build" the complete system directly then locally copy the image to make backups.

The basic steps:

1. Get the Raspberry Imager if you don't have it.
2. Download the Ubuntu 22.04.01 image.
3. Create the SD card.
4. Insert the SD into a Raspberry Pi 4B with attached Monitor/keyboard/mouse. This is needed to do a few very early steps.
5. Powerup/Boot the Raspberry Pi.
6. Take the usual Ubuntu quiz:
   username and password; machine name; country details etc.
7. The machine grind away will reboot.
8. After the reboot, open a terminal and start opening the new system up.
9. Visit `https://github.com/The-SMTSci/FlexBerry`, Follow the Readme section to grab the RPi/FollowMe.sh script. Follow the instructions on the website.
10. After the scipt finishes:
    sudo raspi-config
    and enable the serial ports.
11. Reboot.

Reboot with sudo restart command.

The machine will be 'headless' – meaning no keyboard/mouse/terminal on the telescope. In order to communicate with the device, you will want to start a "ssh terminal session" from the PC. This is standard with Linux desktops, but needs additional packages for Win10/11 machines.

```
cd $HOME
wget https://raw.githubusercontent.com/The-SMTSci/FlexBerry/main/RPi/FollowMe.sh
sudo ./FollowMe.sh pier15 ishmael  # Hostname -> pier15, username -> ishmael
sudo raspi-config
```

```
sudo reboot
```

Figure 1-1: Initial Commands. Bring install to current released content.

```
sudo useradd -m -d /home/flex -G dialout -p "$(openssl passwd -1 'happy startrails')" flex
cd /home/flex
mkdir -p git
cd /home/flex/git
```

```
git clone https://github.com/The-SMTSci/FlexSpec1.git
cd /home/flexgit/FlexSpec1/FlexBerry
sudo bash FollowMe.sh  # this will take a while.
```

Figure 1-2: Add System User, get the FlexSpec1 git repository, run the script.

One of the best is PuTTY.

On your Windows machine, go to the Microsoft store and acquire PuTTY. Microsoft perports this to be the safest way to install things on your machine.

1.1 Cast of Characters

Review the Scenario in Section **??** below.

For the local machine on a local lan, assume:

1. the Raspberry Pi is called pier15
2. We assume the default user's name is fred. (Note this is unique!)
3. It is found at http://pier15.local
4. It may be accessed with ssh:
   ssh fred@pier15.local
5. In addition to your login on the machine, we install a special user called `flex`.
   (a) The `flex` user holds all of our materials in its home directory.
   (b) These are critical to operation of FlexBerry.

Set Ubuntu's opinion of the username for this machine. Here we use pier15. Add the Country locale information, keyboard type, and we recommend you think about boot without login.

At the console you want to add a few initial packages. The goal is to use the git facility to download a copy of `https://github.com/The-SMTSci/FlexSpec1` repository with the comprehensive script of all the packages to load and configurations to be made.

2 The FlexBerry in action

The FlexBerry has two main services running: the bokeh server for the GIU, and a flexdispatch socket service to attach to the Arduino. These are started and maintained by the OS and should not require any thought. The FlexBerry runs the nginx web-server. This listens on port 80 – per usual – and offers access to documentation etc on each of the FlexBerry Raspberry Pi machines.

A special user `flex` will be added with a default password.

1. Acquire the latest image of Ubuntu 22.04 from Canonical for the Raspberry Pi.
2. Burn the ISO image onto a high quality SD Card.
3. Insert the SD Card into the Pi and power on.
4. Start the system and ’take the quiz’: language, keyboard, geographic location, user name and password, machine name etc.

2.1 The Quiz

sec:TheQuiz)

The Quiz consists of all the details a fresh Ubuntu system needs to know. In particular is the user name and initial password.

A special user has been added: "flex". We recommend you keep this user, as it retains some key "user level" data.

3 Scenario

The system is designed to run standalone without a router. It is also designed to be part of a world-wide collaboration of like minded sites; complete site with its domain name, subnets into observatories, each observatory with its own subnet for one or more piers, each pier with one or more OTAs; each OTA with one or more payloads.

The site’s domain name: example.com, with one observatory with one telescope called pier15.

This may be accessed at pier15.example.com in our scenario.

To make this happen DNS needs to be added to one Raspberry Pi within the pier15 subnet. Instructions for this are included.

4 Nginx Install

We will add https to Nginx. This requires adding to nginx.conf

Nginx lives at /etc/nginx.

The pages for nginx lives at /var/www/html deep.

The HTML manual for the FlexSpec1 is at http://pier15.local/flexhelp.

The Arduino GUI is at http://pier15.local/flexspec.

4.1 Nginx Administration

The /etc/nginx/sites-available has all the config information.

The /etc/nginx/sites-enabled has the subset of files we're using.

Please don't mess around with this area unless you understand it, and need to.

TBD

A Names and Places

The design is one of using a network-enabled device on one of many piers in an observatory at a site. There may be one or more sites within a collaboration. One or more collaborations within a federation[1].)

The observatory's Domain name will be referred to herein as the venerable `https://example.com`. (Try it!)

We assume multiple piers, each pier with one or more OTAs, each OTA with one or more instruments under control of one or more Raspberry Pi's or other hardware.

B Components of FlexBerry

The nginx web engine is installed to provide the gateway to the Bokeh code to communicate with the FlexSpec1's Arduino for motors, etc. It also serves as the gateway to the interactive blog for observing sessions. You may add additional pages to the /var/www/html section.

[1] OK Too much late night Star Trek©

A bind9 DNS package is installed, and is used to provide subnet names for the net behind the FlexBerry. There only should be one per subnet. The bind9 package may be simple but it is often manages multi-site corporate networks. We keep it simple.

In additon to nominal things installed as part of an Ubuntu 20.04 system, we add the `supervisord` and `gunicorn` daemons to manage flask/boheh web apps. This complexity permits access from outside the network.

ACTION: Other Pa
ages[1]

C Certificates of Authority

You can use a commercial Certifying Agency to manage Certificates of Authority for you site/subnets or self-sign certificates. Here we play with self-signed certificates. .

D CertMe.sh

This section covers issuing a 'self-signed-certificate' for nginx. It is designed for local network use. We will enable ssh certificate login AND keep PasswordAuthentication. PasswordAuthentication leaves the machine open to brute force attacks – we'll live with that for now.

In order to ssh into the FlexBerry, adding a certificate mechanism to the user's directory is not a bad or difficult thing to do. Here are the steps. .

The files:

```
cd ~/.ssh
ssh-keygen
ssh-copy-id  wayne@pier15
```

E Filesharing

The FlexSpec, running on a Raspberry Pi, takes science images and may store them locally in the Pi filesystem. It may also copy these files to remote machine elsewhere. To work with file processing/viewing utilities one has two choices:

1. Move the file to the Remote machine, and use sofware there.
2. Install the package on the RPi

Of the two, doint all processing/viewing on the remote machine makes the most sense. This shifts the CPU load away from the instrumentation, and allows programs native to a mix of operating systems to be used.

```
# snippets/certme1.txt
ls -LR /etc/nginx
.:
conf.d/         koi-win             nginx.conf       sites-enabled/
fastcgi.conf    mime.types          proxy_params     snippets/
fastcgi_params  modules-available/  scgi_params      uwsgi_params
koi-utf         modules-enabled/    sites-available/ win-utf

./conf.d:

./modules-available:

./modules-enabled:
50-mod-http-geoip2.conf        50-mod-mail.conf
50-mod-http-image-filter.conf  50-mod-stream.conf
```

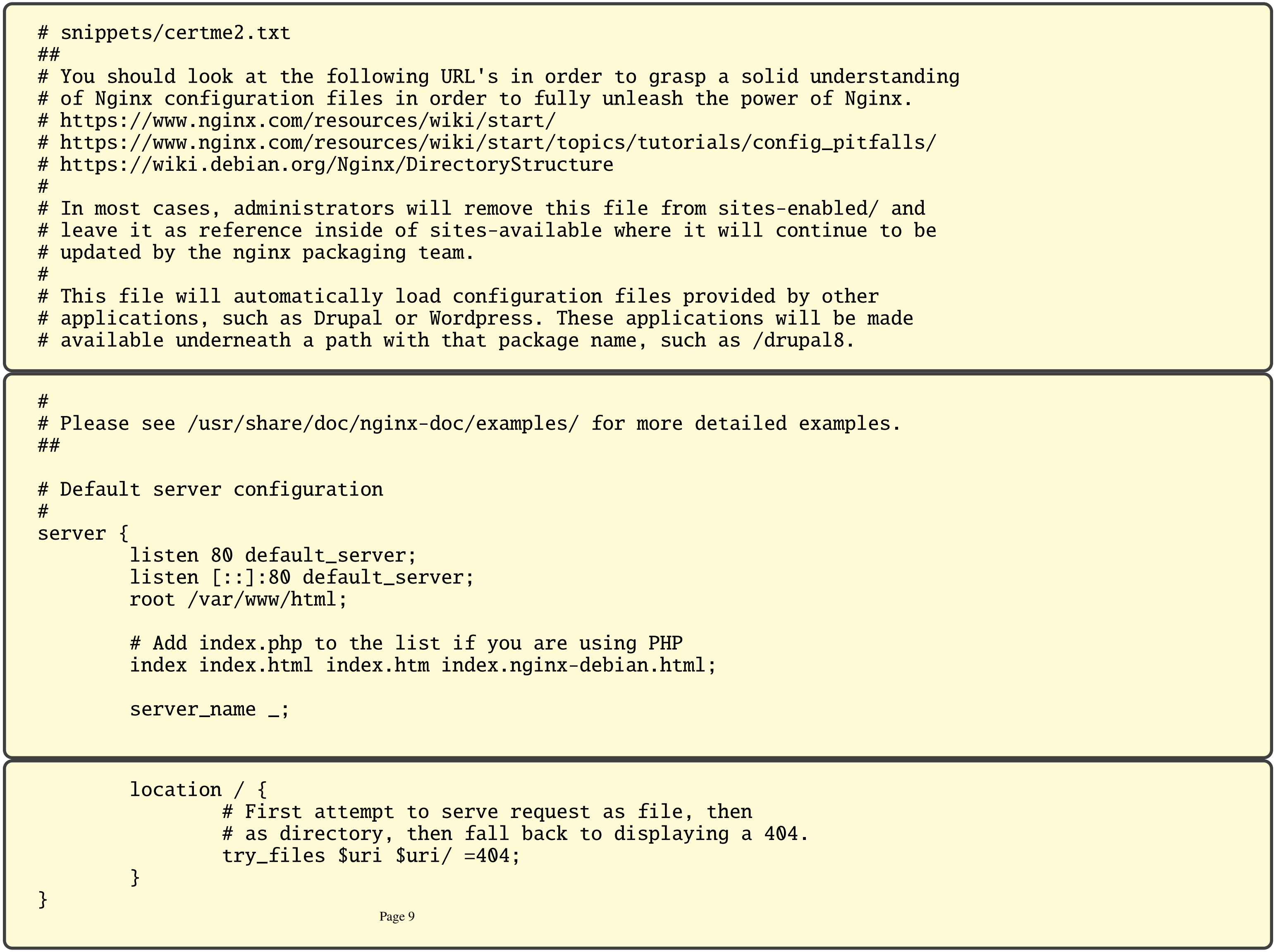
```
50-mod-http-xslt-filter.conf   70-mod-stream-geoip2.conf

./sites-available:
default

./sites-enabled:
default

./snippets:
fastcgi-php.conf   snakeoil.conf
```

Figure D-3: Nginx Files



```
# snippets/certme2.txt
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;
        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }
}
```

Page 9

Figure D-4: nginx sample file

E.1 NFS

```
sudo apt install nfs-common
/mnt/share 10.0.0.0/24(rw,sync,no_subtree_check)
/export/flex 192.168.0.0/24(rw,async,no_subtree_check,anonuid=1000,anongid=1000)
sudo exportfs -a
sudo systemctl restart nfs-kernel-server
sudo ufw allow from 10.0.2.15/24 to any port nfs
sudo ufw enable
# sudo ufw status # look for 2049
```

/subsectionSMB Windows Filesharing

A SMB server is added to promote pushing image files to remote servers during observing.

```
sudo apt install -y samba samba-tools smbclient cifs-utils
sudo systemctl enable --now smbd              # retister for all reboots
sudo ufw allow samba
sudo usermod -aG sambashare flex
sudo smbpasswd -a "flex%time has come"
sudo mkdir -p /samba/{$USER,flex}             # make shares for the two main users
sudo chgrp -R sambashare /samba
sudo usermod -aG sambashare $USER             # add these users to group
sudo usermod -aG sambashare flex
#smb://winhost/shared-folder-name
# TODO mod /etc/samba/smb.conf
sudo systemctl restart smbd
sudo systemctl restart nmbd
```

Gnome's file manager has built in SMB support.

https://wiki.samba.org/index.php/User_and_Group_management has decent examples.

/etc/samba/smb.conf:

server role = standalone server interfaces = 127.0.0.0/8 eth0 bind interfaces only = yes

[flex] path = /samba/flex browseable = no read only = no force create mode = 0660 force directory mode = 2770 valid users = flex @sadmin

E.2 Flex Nginx Configuration

This is a rambling collection of notes about the Nginx install. This is initial setup:

```
FILE 50-mod-http-geoip2.conf
load_module modules/ngx_http_geoip2_module.so;
FILE 50-mod-http-image-filter.conf
load_module modules/ngx_http_image_filter_module.so;
FILE 50-mod-http-xslt-filter.conf
load_module modules/ngx_http_xslt_filter_module.so;
FILE 50-mod-mail.conf
load_module modules/ngx_mail_module.so;
FILE 50-mod-stream.conf
load_module modules/ngx_stream_module.so;
FILE 70-mod-stream-geoip2.conf
load_module modules/ngx_stream_geoip2_module.so;
```

F PuTTY and Xming on Windows

The PuTTY program provides a X11/ssh connection to remote machines. Here the machine's hostname is pier15.local, the port is 22.

XMing is a X11 "server" (meaning is backwards from data client/server sense) that "serves" the graphics from a remote programming doing all the heavy computing.

Grab Xming and install on a Win machine.

First some X11 magic:

```
the files /etc/X11
/etc/X11/xinit/xserverrc
~/.xinitrc
~/.xsession  -- start programs when the login is complete
~/.profile
```

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

```
xmodmap -pke > ~/.Xmodmap # make a .Xmodmap to hack upon
```

This varies with new modern keyboards, there is massive confusion over what a key is called.

xev program will show keycodes when a key changes state, like shift down, then shift up etc... When using the "caps" key:

```
keycode  37 = Control_L NoSymbol Control_L
```

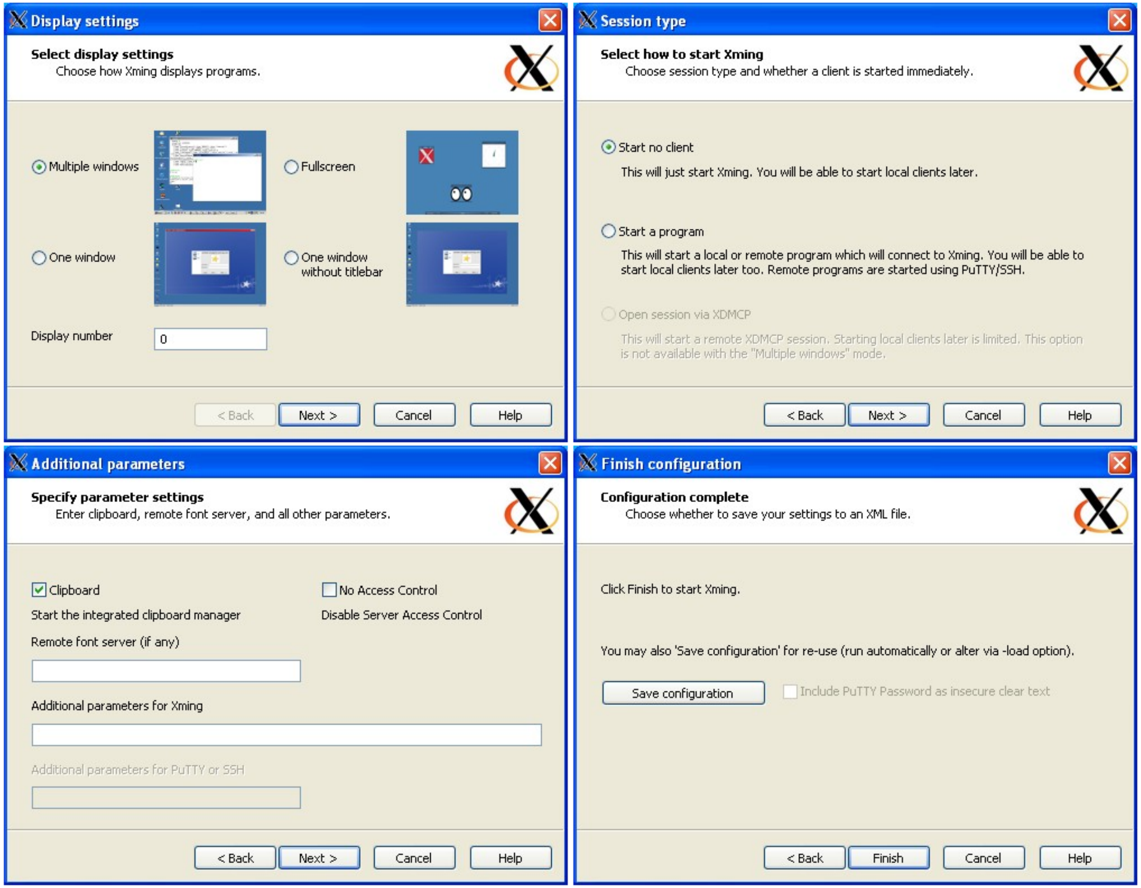
Run XLaunch verify settings:

1. multiple windows
2. start no client
3. clipboard checked
4. save configuration

Configure PuTTY:

1. Session:
   (a) hostname:pier15.local port 22
   (b) CTM_X
   (c) check Only on clean exit
2. Window
   (a) 120 50
   (b) scrollback 200
   (c) display scrollbar
   (d) Reset scrollback on display activity
   (e) push erased text into scrollback
3. Connection -> Data
   (a) login name
   (b) Terminal-type string xterm
   (c) Terminal Speed 38400, 38400
4. SSH->X11
   (a) Enable X11 forwarding
   (b) localhost:0.0
   (c) MIT-Magic-Cookie-1

On the Windows pane, enter a new configuration for this machine.



Display settings

Select display settings
Choose how Xming displays programs.

● Multiple windows
○ Fullscreen
○ One window
○ One window without titlebar

Display number: 0

Session type

Select how to start Xming
Choose session type and whether a client is started immediately.

● Start no client
This will just start Xming. You will be able to start local clients later.

○ Start a program
This will start a local or remote program which will connect to Xming. You will be able to start local clients later too. Remote programs are started using PuTTY/SSH.

○ Open session via XDMCP
This will start a remote XDMCP session. Starting local clients later is limited. This option is not available with the "Multiple windows" mode.

Additional parameters

Specify parameter settings
Enter clipboard, remote font server, and all other parameters.

☑ Clipboard
Start the integrated clipboard manager

□ No Access Control
Disable Server Access Control

Remote font server (if any)

Additional parameters for Xming

Additional parameters for PuTTY or SSH

Finish configuration

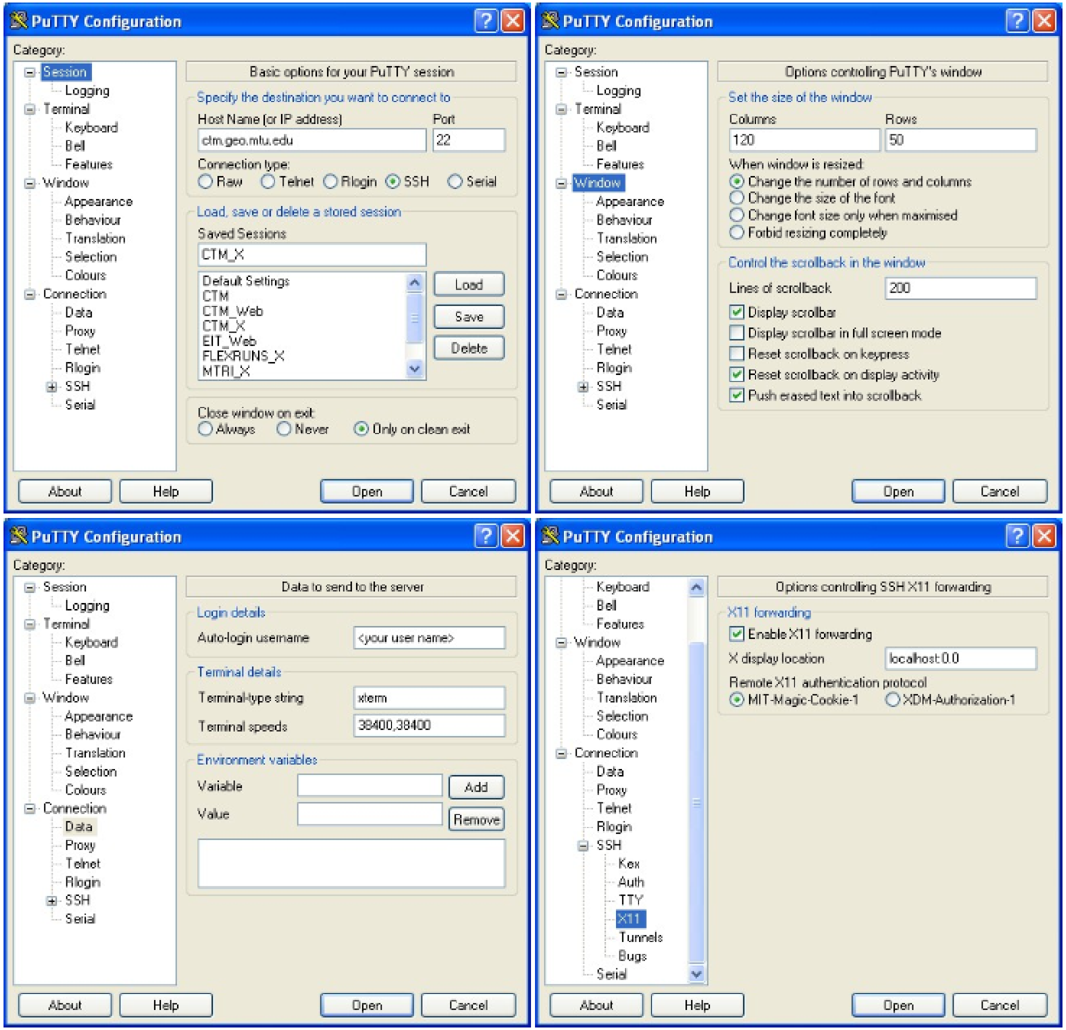
Configuration complete
Choose whether to save your settings to an XML file.

Click Finish to start Xming.

You may also 'Save configuration' for re-use (run automatically or alter via -load option).

□ Include PuTTY Password as insecure clear text

Figure F-5: XMing screen snaps of the 4 configuration menu panels.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): ctm.geo.mtu.edu   Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions: CTM_X

Default Settings
CTM
CTM_Web
CTM_X
EIT_Web
FLEXRUNS_X
MTRI_X

Close window on exit: ○ Always ○ Never ● Only on clean exit

Options controlling PuTTY's window

Set the size of the window

Columns: 120   Rows: 50

When window is resized:
● Change the number of rows and columns
○ Change the size of the font
○ Change font size only when maximised
○ Forbid resizing completely

Control the scrollback in the window

Lines of scrollback: 200

☑ Display scrollbar
□ Display scrollbar in full screen mode
□ Reset scrollback on keypress
☑ Reset scrollback on display activity
☑ Push erased text into scrollback

Data to send to the server

Login details

Auto-login username: <your user name>

Terminal details

Terminal-type string: xterm

Terminal speeds: 38400,38400

Environment variables

Variable:

Value:

Options controlling SSH X11 forwarding

X11 forwarding

☑ Enable X11 forwarding

X display location: localhost:0.0

Remote X11 authentication protocol: ● MIT-Magic-Cookie-1 ○ XDM-Authorization-1

Figure F-6: PuTTY screen snaps of the 4 configuration menu panels.

My Bibliography and References

Action Items:

[1] Explain other packages and subsytems.