Appendix 1

WORKING WITH OTHER OPERATING SYSTEMS

This first appendix provides details on how to get set up if you have a computer with Microsoft Windows or Linux, rather than Mac OS X. As described in *Before You Begin*, Linux and OS X are both based on Unix. They are much more similar to each other than to Windows. Because of this, it will be easier to follow along with this book if you are using Linux than if you are using Windows. The main difficulties for using this book with Windows will be related to the command line, introduced in Chapters 4–6. There are work-arounds for that section, and it should not be much of a problem to work through the rest of the book.

Most of the important tools used in this book are in fact available for all three major operating systems—OS X, Linux, and Windows. Some of these tools, such as regular expressions, are entirely platform independent, and can be used within many programs and languages. Certain others, such as specific brands of text editors, do depend on the operating system, but are available for each OS with more or less similar features. Still other tools, however, have only approximate alternatives for the different operating systems.

The instructions in this appendix can't anticipate all of the possible variations of systems, so if you get stuck, check the content at `practicalcomputing.org`.

Microsoft Windows

Should I work in Windows or install Linux?

If you have a computer running Microsoft Windows, you have two options. The first is to install programs, languages, and environments within Windows that add most of the functionality needed to develop and use the skills we present—for example, jEdit, Python, and Cygwin. The second and more comprehensive option is to install Linux on your computer alongside Windows, giving you the chance to learn your way around an operating system based on Unix. Robust and freely available virtual machine software lets you do this without needing to reformat your hard drive or reboot your computer to switch between operating systems.

We'll cover both approaches in this appendix, starting with the first—installing programs within Windows for added functionality.

Text editors for text editing and regular expressions in Chapters 1–3

Using jEdit At this time, our two recommendations for Windows text editors are Notepad++, described below, and jEdit. jEdit is a full-featured open-source text editor that is available for all major operating systems. It has many of the same features and advantages as TextWrangler, the editor for OS X that is used throughout the main text of this book, and like TextWrangler, supports regular expressions. You can obtain jEdit from `www.jEdit.org`. Click the download link on the Web site and select the Windows Installer for the stable version. Follow the instructions when you launch the installer. You will need to have Java Runtime Environment (JRE) 1.4 or greater installed, which you can find at `www.java.com/en/download/`. Once you've installed the editor, select Find... from the Search menu and make sure that Regular expressions is checked.

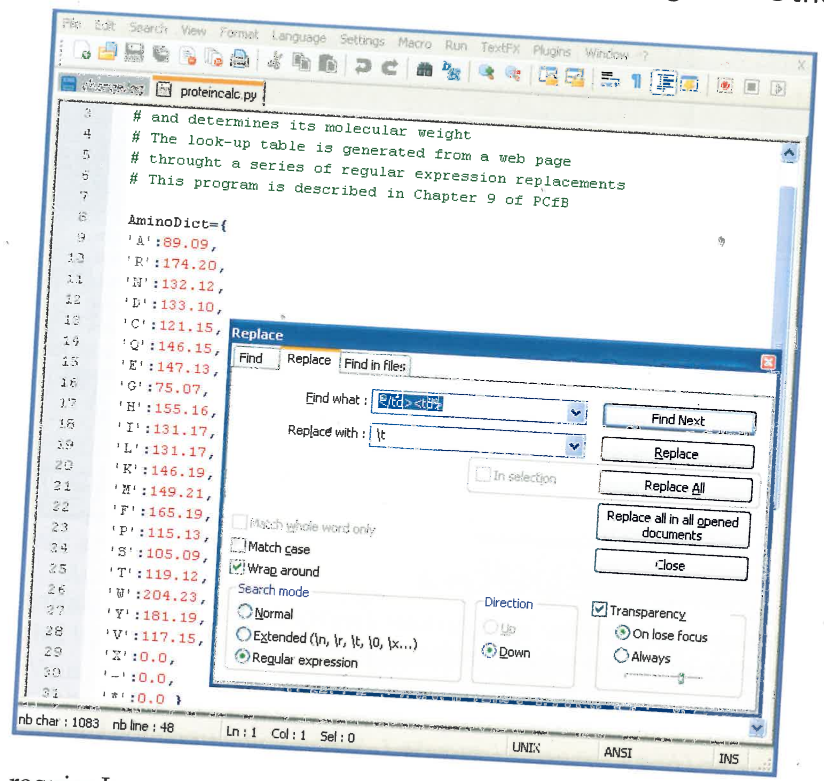
There are some differences from TextWrangler in the way jEdit handles regular expressions. One is that when you capture text with parentheses, the text is still held in numbered variables, but these are preceded by a `$` rather than a `\`. For instance, the replacement term `\1\t\2\t` in TextWrangler is written as `$1\t$2\t` in jEdit. Note that not all backslashes are replaced with dollar signs, only those used to designate that a number represents a bit of captured text.

Another difference is that in regular expression searches, jEdit uses `\n` as the universal end-of-line character; this is different from TextWrangler. To change this character in an open document (whether to carriage return, line feed, or both), click the Utilities menu and select Buffer Options... to open a dialog box. One of the options in this dialog is to modify the Line separator, which is just another term for line ending.

Using Notepad++ One of the most popular text and script editors for Windows is Notepad++, which supports many features for programmers (Figure A1.1). It can be downloaded from `www.sourceforge.net/projects/notepad-plus/`. With this editor, in Chapters 2 and 3 you will use a Replace dialog box (ctrl H) rather than a Find dialog, and you will need to make sure that regular expression is checked at the bottom of the dialog box.

Notepad++ doesn't support all of the regular expressions presented in this book. Because of this, when working through Chapters 2 and 3, you might want to use jEdit instead. See Appendix 2 for more on regular expressions and compatibility. Notepad++ does not support the modifier `?` to change the greediness of quantifiers. It uses the same convention as TextWrangler, `\1`, for using captured text in substitutions. Support for end-of-line characters in searches may require selecting Extended mode instead of Regular expression.

For programming, however, Notepad++ will probably give you a better user experience. One minor advantage Notepad++ has over jEdit is that it does not



```
# and determines its molecular weight
# The look-up table is generated from a web page
# throught a series of regular expression replacements
# This program is described in Chapter 9 of PCfB

AminoDict={
'A':89.09,
'R':174.20,
'N':132.12,
'D':133.10,
'C':121.15,
'Q':146.15,
'E':147.13,
'G':75.07,
'H':155.16,
'I':131.17,
'L':131.17,
'K':146.19,
'M':149.21,
'F':165.19,
'P':115.13,
'S':105.09,
'T':119.12,
'W':204.23,
'Y':181.19,
'V':117.15,
'X':0.0,
'-':0.0,
'*':0.0 }
```

Replace

Find what : </td><td>

Replace with : \t

[ ] In selection
[ ] Match whole word only
[ ] Match case
[✓] Wrap around

Search mode: ( ) Normal ( ) Extended (\n, \r, \t, \0, \x...) [✓ Regular expression]

Direction: ( ) Up [✓ Down]

[✓] Transparency: [✓ On lose focus] ( ) Always

nb char : 1083 nb line : 48 Ln : 1 Col : 1 Sel : 0 UNIX ANSI INS

File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
proteincalc.py
Find
Replace
Find in files
Find Next
Replace
Replace All
Replace all in all opened documents
Close

FIGURE A1.1 The Notepad++ editor, showing syntax coloring and the regular expression search box

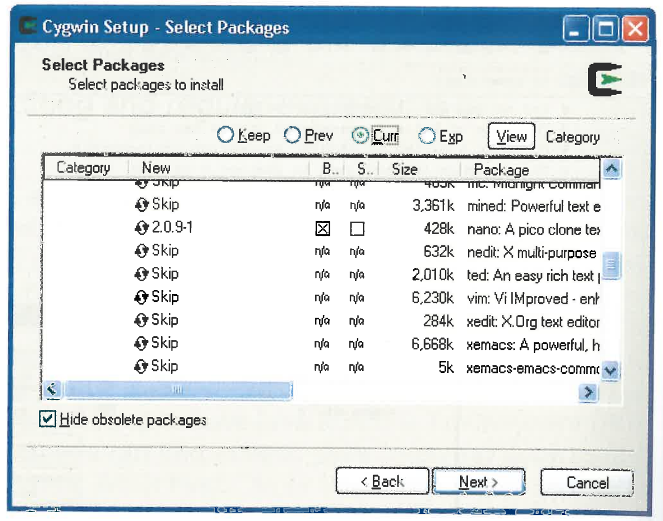
require Java or other ancillary programs, so file opening and saving dialog boxes have a more familiar look and feel.

Cygwin for emulating Unix shell operations in Chapters 4–6

The DOS and PowerShell command-line interfaces available in Windows are very different from the Unix command line, and they cannot be used in place of it when working through the examples in this book, nor in real life. To get a proper Unix command line, you can install a full Linux system, via the instructions described later in this appendix. However, it is also possible to partially emulate a Unix command line with the Cygwin package. Cygwin can be downloaded from `cygwin.com`, which provides installation instructions as well.

Many of the programs described in the shell chapters, such as `cd`, `ls`, `mv`, and `cp`, are installed by default with Cygwin. Other useful programs which aren't installed by default can be added from within the `setup.exe` installer, ideally at the time you originally install Cygwin. Programs you should be sure to add in this manner include:

- Editors: `nano`
- Interpreters: `python`
- Net: `curl`



```
Cygwin Setup - Select Packages

Select Packages
    Select packages to install

          ○ Keep  ○ Prev  ◉ Curr  ○ Exp   [View] Category

Category   New            B..  S..   Size    Package
           Skip           n/a  n/a   3,361k  mined: Powerful text e
           2.0.9-1        ☒    ☐     428k    nano: A pico clone te
           Skip           n/a  n/a   632k    nedit: X multi-purpose
           Skip           n/a  n/a   2,010k  ted: An easy rich text
           Skip           n/a  n/a   6,230k  vim: Vi IMproved - enh
           Skip           n/a  n/a   284k    xedit: X.Org text editor
           Skip           n/a  n/a   6,668k  xemacs: A powerful, h
           Skip           n/a  n/a   5k      xemacs-emacs-comm

☑ Hide obsolete packages

                            < Back   Next >   Cancel
```

FIGURE A1.2 The installer for Cygwin, a Unix subsystem for Windows Be sure to install the optional packages as indicated in the text.

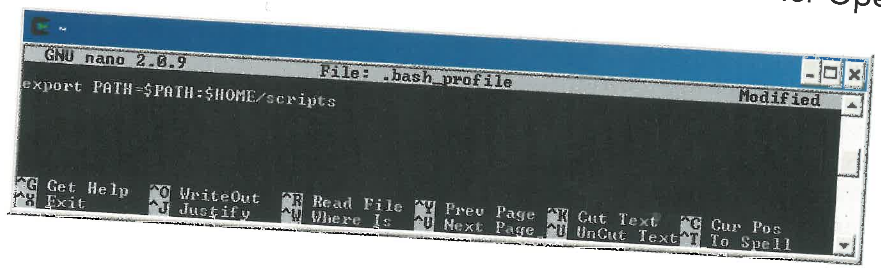
To select these packages, click on the little circular arrows under the New column heading in the installer, so that a version number appears in place of the word Skip (Figure A1.2).

More information on the installation and use of Cygwin can be found at `www.physionet.org/physiotools/cygwin/`. When you launch the Cygwin environment, you will be presented with a window showing the `bash` shell. You will be able to operate using Unix commands as described in the text, somewhat equivalent to running a terminal program in a true Unix system.

The root directory within Cygwin (the directory specified by `/` as described in Chapters 4–6) is not the Windows root directory, `c:\`, but corresponds instead to the Windows directory `c:\cygwin`. This means, for instance, that the file `/home/Administrator/.bash_profile` in the Cygwin environment is the same file as `c:\cygwin\home\Administrator\.bash_profile` in Windows. Conversely, the Windows root directory `c:\` is available from within the Cygwin environment as the directory `/cygdrive/c`. This allows you to access files from Windows without having to copy them within `c:\cygwin`.

When you launch the Cygwin environment it puts you within the default home directory. Depending on your system and installation, this can be `/home/Administrator` or, particularly confusing, `/cygdrive/c/Documents and Settings/lucy`. This directory corresponds to the example home directory we use throughout the book, `/Users/lucy`, and at the command line you can always use the `~/` shortcut within Cygwin to refer to your home directory, rather than typing out the whole thing.

If you are using Cygwin, you should create your scripts and examples folders as subdirectories within whatever directory you are in upon first launching Cygwin. You can find this initial directory by typing either `pwd` at the initial prompt,



```
GNU nano 2.0.9              File: .bash_profile                    Modified
export PATH=$PATH:$HOME/scripts




^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text    ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text  ^T To Spell
```

FIGURE A1.3 Using the editor `nano` in Cygwin to modify the `PATH` variable in the `.bash_profile` settings file

or `echo $HOME`. The instructions for editing the `$PATH` variable in your `.bash_profile`, as described in Chapter 6, should apply here as well; this is because the `$HOME` variable is used to make a reference which is portable between systems (Figure A1.3).

One drawback of Cygwin is that you cannot copy and paste from within the terminal window by default. There is a convoluted way to enable this by right-clicking on the Cygwin tab in the Windows taskbar and in the preferences enabling QuickEdit. Now you can select text, copy it by pressing return, and paste into another terminal window.

Another catch is that ctrl C, the usual Windows shortcut for copy, is also a widely used Unix and Cygwin shortcut for interrupting a process, such as when you have a runaway program. Similarly, ctrl Z, which typically means "undo" in Windows, is the Unix and Cygwin command to stop (suspend the operation of) a running process. You will have to retrain your fingers to use the right shortcuts and avoid the wrong ones for each environment you work in.

Python on Windows for Chapters 8–12

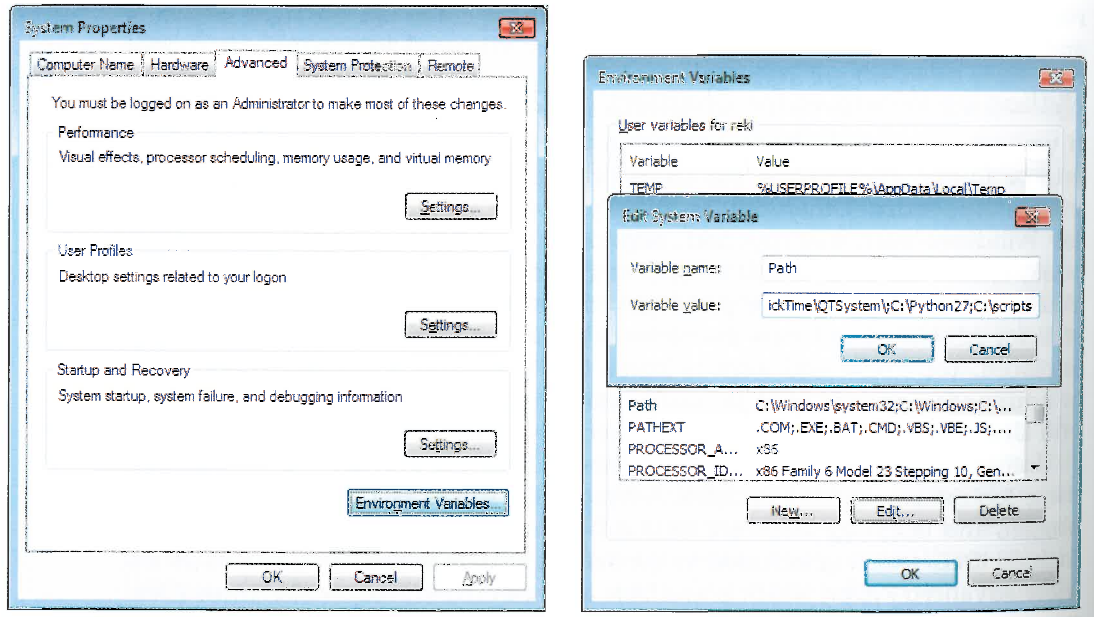
If you are going to use Windows with a virtual Unix environment—either Cygwin or Linux—then follow the Linux or Cygwin instructions for Python later in this Appendix. Otherwise, to run Python in Windows, download Python for Windows from `www.python.org/download`. For compatibility with this book, choose the most recent version of Python 2.x, not any of the 3.x versions. You will need administrator privileges to complete the installation process. Install for All Users, and note the name of the folder name where the installation is placed. You might also want to install the Python Win32 Extensions, available at `sourceforge.net/projects/pywin32`. Among other things, this will give you Unicode support for use of extended characters.

Integrated Development Environments, or IDEs, are available for Python, and these too can run inside Windows. A typical IDE serves as a text editor, code debugger, and language reference source all in one. Popular free IDEs for Python include IDLE, which is included in the default Python installation, as well as the more advanced PyDev, which is available at `pydev.org`. From within these IDEs, you can edit and run the Python programs described in this book. In Windows, the commands for reading and writing files will vary slightly from our examples; this is because you will be using paths such as `'C:\scripts'` rather than Unix or OS X paths written as `'~/Documents/scripts'`. Even so the underlying idea is the same: make a new folder called scripts on your main drive, for example `C:\scripts`, and use this as your storage location for all working scripts.

In Windows, running Python scripts from the Command Prompt is more complicated than running them from within an IDE such as IDLE. You will have to inform the Windows command-line environment of where to find both the Python program and your scripts. There is a chance that the installer has made the first modification for you already, but proceed with these modifications to make sure that your scripts folder can also be found. First you must determine where `python.exe` was installed on your computer—typically this will be something like `C:\Python27`. At the same time, you can also let the operating system know about the scripts directory you created above. In the Windows Control Panel, click on the System icon, then select Advanced system settings. In the dialog that appears, click the Environment Variables button near the bottom (Figure A1.4).

Scroll down among the list of Environment Variables until you see the Path variable, and double-click to edit this. At the end of the existing string value, type a semicolon, followed by the name of the folder that contains `python.exe` and the location of your scripts folder. For example, you might type something like this: `;C:\Python27; C:\scripts;`.

Click OK to save your new path. Thanks to adding `python.exe` to your Path variable, you will now be able to launch the Python interactive prompt just by



You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your logon

System startup, system failure, and debugging information

User variables for reki

Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp

Variable name: Path

Variable value: ickTime\QTSystem\;C:\Python27;C:\scripts

Variable	Value
Path	C:\Windows\system32;C:\Windows;C:\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86
PROCESSOR_ID...	x86 Family 6 Model 23 Stepping 10, Gen...

FIGURE A1.4 Setting the Path environment variable in Windows, so that the Python executable and the `scripts` folder can be found

typing `python` at the Command Prompt. In Windows you use ctrl Z instead of ctrl D to exit the Python environment. Note that because the Python installation process will have already associated `.py` files with Python, you do not strictly need to use a shebang line in your scripts as you must on OS X and Linux. You should still include it, though, so that your programs can work on these other platforms as well.

To execute scripts stored in the scripts directory, just type their name at the Command Prompt. You can get the prompt window by clicking the Start button in the Task bar and selecting Programs ▸ Accessories ▸ Command Prompt. Remember that the Command Prompt is not the same as the Unix command-line environment, and will therefore not have the same commands available. You can also run a script by double-clicking its icon on the desktop or in an Explorer window, but this is only useful if the program doesn't require command-line parameters, and also doesn't need to read or write files based on the working directory, rather than just the folder in which the icon happens to be located.

Python under Cygwin If you install Cygwin as described above, then during the installation process you should also install Python as one of the optional packages. You can use the Unix instructions in the text with little modification. Editing your `.bash_profile` for setting the scripts path will use your Cygwin home directory instead of `/Users/lucy`, as set by the `$HOME` variable. In your programs, for indicating paths to files you can actually use normal slashes, for example `c:/Users/lucy`, to avoid backslashes being misinterpreted in a special way. Also, open files using the mode `'rU'` instead of simply `'r'`, to avoid problems with end-of-line characters.

Working with MySQL on Windows for Chapter 15

In Windows, if you haven't configured MySQL to autostart during the installation, first get a Command Prompt, then type `cd c:\mysql\bin`. In that directory, start the database server with the command `mysqld`. At this point, you can launch `mysql` and follow along as described in the chapter. File paths will need to be specified using the `c:\` notation instead of the Unix-style notation of paths like `/Users/lucy`.

To configure the server to autostart, see the detailed instructions (and helpful user comments) at `dev.mysql.com/doc/refman/5.1/en/windows-start-service.html`.

Working with vector and pixel art in Windows for Chapters 17–19

In Windows, you can use the free vector and pixel editors Inkscape, GIMP, and ImageJ, or the commercial Adobe Illustrator and Photoshop. The experience will be the same as described in these graphics chapters except that some key combinations will be a bit different. Specifically, the alt key is used in place of option, and ctrl is used where Command (⌘) would be used in OS X.

Linux

Installing Linux

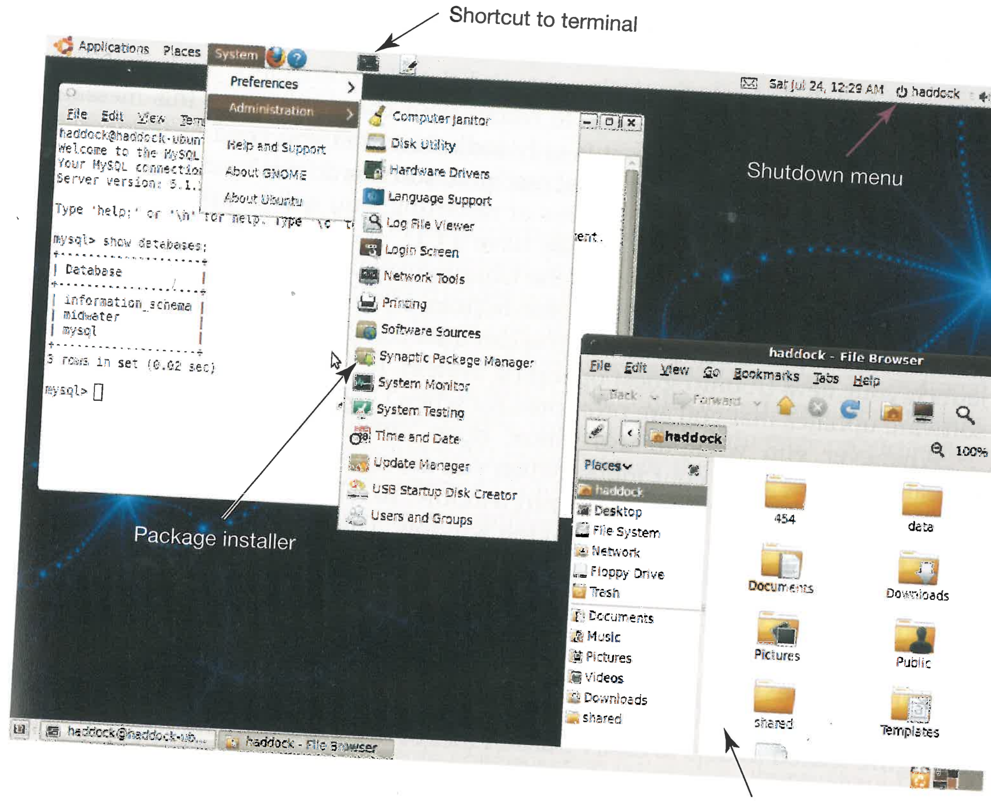
There are many options for installing Linux. The first decision you need to make is which of the many Linux distributions to select. We recommend starting with Ubuntu (`www.ubuntu.com`) for the simple reason that it works well with most consumer computers. It also has a very wide user base, in part because it is so accessible to beginners. The instructions here were written for Ubuntu 10.04, and some details may change with future versions. Consult `www.ubuntu.com` for updated installers or `practicalcomputing.org` for additional instructions.

If you are running Windows, we recommend that you install Ubuntu within a virtual machine, as described below. This is the simplest way to get Linux and Windows going on the same computer, and to easily switch between them as you work. A virtual machine is a program that runs within a **host** operating system and simulates an entire computer running a **guest** operating system. In this case the host operating system is Windows, and the guest operating system is Ubuntu Linux. You can run your virtual machine right alongside other Windows programs, without any need to restart your computer to switch between the host and guest operating systems. You can even suspend the guest computer if you want to close the virtual machine program, and then restart it later right where you left off.

If you would like to erase Windows and install Linux alone on your computer, or install Linux on a separate disk partition that you can boot Linux from when you restart the computer, you can follow the standard installations instructions on the Ubuntu web site. A dual-boot system that can start up as either Windows or Linux can be quickly configured using Wubi, found at `wubi-installer.org`. These standard installations, though, don't give the benefits of using Windows programs when you need them, alongside the Linux system.

There are virtual machine programs that work with just about any combination of host and guest operating systems, except that OS X cannot be installed as a guest operating system on any virtual machine. We recommend the free, full-featured virtual machine VirtualBox by Oracle. It works within many host operating systems, including Windows, OS X, and Linux. To get up and running, you will need to install VirtualBox, create a virtual computer within it, install Ubuntu on the virtual computer, and then configure the system (Figure A1.5).

Installing VirtualBox If you have a standard PC running Windows, download and install the x86 version of VirtualBox for Windows hosts. It is available at `www.virtualbox.org/wiki/Downloads`. The default installer settings should be fine. Once VirtualBox is installed, launch it and click the New button to create a new virtual computer that you will later install the guest operating system on. The setup wizard will guide through this process. You can give the virtual machine any name you like—Ubuntu would do just fine. Select Linux for the operating system, Ubuntu for the version, and select a minimum of 512 MB base memory. You will then need to create a virtual hard disk. In the Virtual Hard Disk window, click New... and select the option to create a dynamically expanding storage disk.



```
haddock@haddock-ubun
Welcome to the MySQL
Your MySQL connectio
Server version: 5.1.

Type 'help;' or '\h' for help. Type '\c' t

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| midwater           |
| mysql              |
+--------------------+
3 rows in set (0.02 sec)

mysql> 
```

FIGURE A1.5 The Ubuntu Linux environment running in a VirtualBox window Note the convenient terminal shortcut in the VirtualBox window bar.

You will then need to select a size for the virtual hard disk. 8 GB is sufficient for most purposes.

Installing Ubuntu in VirtualBox Once you have installed VirtualBox and created a virtual computer, as described above, you'll need to install Ubuntu on the virtual computer. Follow the links for downloading Ubuntu from `www.ubuntu.com` and download the 32-bit Desktop edition. This is a large file, and may take several hours if you have a slow Internet connection. Start the virtual computer you created in VirtualBox by double-clicking it in the left side of the VirtualBox window. This will launch the First Run Wizard. Select CD/DVD-ROM Device for the Media type, click on the folder icon under Media Source, click the Add button at the top of the window that pops up, and then browse to and select the Ubuntu file you downloaded. Once you have finished the First Run Wizard, your virtual computer will start up and Ubuntu will guide you through the installation process. If you have any questions or difficulties with the Ubuntu installation process, consult Ubuntu's Web site for help.

When the Ubuntu installer asks to prepare disk space, it is only looking at the virtual disk and not your entire hard drive. You should follow the default settings, which will dedicate the entire virtual disk to Ubuntu. Once the installation is complete you will get a message to restart your computer. Since this message is within your virtual machine, it is only indicating that you need to restart that virtual computer and not the physical machine. You can do this by clicking Restart in the message window. In the process of restarting, you will be asked to remove the installation CD. You don't actually have a CD, of course, but you do need to disconnect the virtual machine from the Ubuntu installation file you downloaded. To do this, click on the CD icon at the bottom of the virtual machine window, and select Unmount CD/DVD ROM. You can then complete the restart, which will launch your new Ubuntu guest operating system. It is a good idea to install any software updates that you are prompted for when Ubuntu starts.

Whenever you want to use Linux, open VirtualBox and launch Ubuntu. VirtualBox will capture your mouse when you are using Ubuntu, which will keep you from moving it outside the Ubuntu window. To release the mouse, hit the key indicated at the lower right corner of the window. There are many books and online resources for learning your way around Ubuntu and around Linux in general. We encourage you to seek out these resources if you are interested in becoming more familiar with the specifics of this computing environment.

Installing Guest Additions After the basic installation is working, you will probably want to do two things: first, increase the screen resolution, and second, configure shared folders to easily transfer files between the host operating system, Windows, and the guest operating system, Linux. To accomplish these tasks, you will install Guest Additions, which are enhancements to the guest operating system that help it interact more fluidly with the host operating system. Follow these instructions:

1. From the VirtualBox menu bar on your host operating systems, go to the Devices menu and choose Install Guest Additions. This won't actually install the Guest Additions, it just makes the installation files available from within the guest operating system.
2. At the top of the Ubuntu screen, click on the Places menu and select Computer. This will open a window with several icons, including one for a CD the name of which will start with VBOXADDITIONS. Double-click on the CD icon to see the Guest Additions installation files.
3. You will need to do the installation via the command line, which requires opening a terminal window. Click on the Applications menu at the top of the Ubuntu screen, select Accessories, and then click Terminal.
4. Type the word `sudo` with a space after it at the command line. Then drag the file `VBoxLunuxAdditions-x86.run` from the installation file window you opened earlier to the terminal window. This just inserts the full name of the file with its path.

5. Press return. `sudo` will ask you for your Ubuntu password, and then launch the installation.
6. When the installation is done, restart Ubuntu.

Now that the Guest Additions are installed and are ready to configure shared folders, you can resize the Ubuntu Desktop just by resizing the VirtualBox window.

Setting up shared folders Configuring shared folders will allow you to easily transfer files between operating systems, but still requires a few more steps. Unfortunately, configuring the shared folder is a bit complicated and isn't particularly well documented, even though this is one of the first things that most people installing VirtualBox will want to do. If you don't successfully complete the optional shared folder instructions below you can still transfer files to and from Ubuntu over the web or with disks. If you have trouble, consult the Ubuntu page on VirtualBox at `help.ubuntu.com/community/VirtualBox`, or the VirtualBox documentation at `www.virtualbox.org/manual/ch04.html#sharedfolders`.

Setting up a shared folder requires multiple steps. You must create a folder on the host (Windows) operating system, create a folder on the guest (Ubuntu) operating system as well, configure VirtualBox to share the host folder with the guest, and then configure the guest operating system to connect the two folders each time it boots up.

1. On the Windows desktop (or somewhere else on the Windows filesystem, if you prefer), create the folder you will share with the guest operating system. For the sake of this example configuration let's call it UbuntuShared and put it on the desktop.
2. Open a terminal window in Ubuntu, and create a folder on the Ubuntu side. We'll call it `hostshare`:

```
mkdir ~/hostshare
```

From the VirtualBox menu bar on your host operating systems, go to Devices menu and select Shared folders... Click the folder icon with a plus to add a shared folder, and browse to the UbuntuShared folder you created above. Make sure that the Make Permanent option is selected, and click OK twice to exit.

3. Restart Ubuntu using the menu at the right side of the upper menu bar.

At this point you have created the needed folders and configured the VirtualBox program to share the host folder with the guest operating system. You still need to configure the guest operating system to mount the host folder, which is the most complicated part. First, you'll gather the information you need mount the folder. Then you'll edit a configuration file to mount the folder each time Ubuntu starts. All of the following commands are issued in the Ubuntu terminal window.

4. Open a terminal window and type in the following commands shown in bold:

```
lucy@lucy-ubuntu:~$ cd ~/hostshare
lucy@lucy-ubuntu:~/hostshare$ pwd
/home/lucy/hostshare
```

5. The output of the above commands, e.g., `/home/lucy/hostshare`, is the path of the Ubuntu shared folder. The middle part of the path is your username. In this example the username is `lucy`, though yours will probably be something else. You will need this path and username in later steps.
6. The following steps will automatically mount the shared file each time Ubuntu starts up. Edit your system configuration file `fstab` in the terminal window by typing:

```
sudo nano /etc/fstab
```

7. Add the line below to the `fstab` file, right below the line that starts with: `# <file system> <mount point>...`:

```
UbuntuShare /home/lucy/hostshare vboxsf uid=lucy 0 2
```

   where `/home/lucy/hostshare` is the path you got with the pwd command in step 4 and `lucy` is your username (the middle part of the path). Once you have finished, type ctrl X, then follow the prompts to save changes and exit the nano program.
8. Restart Ubuntu. Try transferring some files to make sure that the shared folders are working properly.

If you cannot get it to work, try this additional step, just to see whether the shared folders can be mounted at all.

Use the following command in the terminal window to mount the folder. UbuntuShare is the name of the folder on the host operating system and `/home/lucy/hostshare` is the path to the folder on the guest operating system that you got in the previous step. (This path will be something other than the example used here). This command will require your Ubuntu password:

```
sudo mount -t vboxsf -o uid=1000,gid=1000 UbuntuShare /home/lucy/hostshare
```

Without restarting, you should now be able to share files between the operating systems by moving them to the shared folder. Test that the sharing worked

by moving some files in and out of the `hostshare` folders in Ubuntu and the UbuntuShare folder in Windows. If you get error messages or can't transfer files, check to make sure that the folder names are correct and retry the command. If you have further problems consult the Web sites mentioned in the introduction to sharing. If it does work, you are almost there. Recheck the `fstab` line and make sure that your directory names correspond. The `mount` command typed above is only a temporary fix that lasts until that terminal session is ended.

You can learn more about using VirtualBox from the user manual and from the forums at `practicalcomputing.org`. The manual is bundled with the VirtualBox installer and is also available at `http://download.virtualbox.org/virtualbox/UserManual.pdf`.

Installing software in Linux An important advantage of Linux is that it comes with a wider range or pre-installed software than other operating systems, and all of it is free. Right out of the box it will have many of the tools, from Python modules to full-functioned office productivity software, you will need on a daily basis. Chances are, though, that you will soon want to install additional software, including some of the more specialized tools described in this book.

There are several ways to install software on Linux. While you can download software packages from the Web and install them yourself, Ubuntu Linux comes with a few tools that streamline the process. If the software you want is available via these tools, it is a good idea to use them, since they simplify the installation process and ensure that everything is configured properly for your system. At the command line in Ubuntu, use `apt-get` for downloading and installing software. There are also a couple of GUI software installer interfaces. If you know the program you are looking for, use the Synaptic Package Manager (found in the System ▸ Administration menu). If you know what type of program you want but aren't sure which particular software packages are available, Ubuntu has a curated list of programs arranged by categories. This makes it simple to browse and install new software. This tool is available at Applications ▸ Ubuntu Software Center. Other versions of Linux have their own package managers, including the command-line tools `yum` and `rpm`.

Text editing and regular expressions with jEdit for Chapters 1–3

Ubuntu comes with several text editors pre-installed, and makes additional editors available via its Synaptic Package Manager. We suggest that you install jEdit. The Synaptic Package Manager makes installing not only jEdit but other programs a simple process. Select the System menu at the top of the Ubuntu screen, then the Administration submenu, and then click on the Synaptic Package Manager. Search for jEdit in the Quick search box. Click the checkbox next to jEdit in the Applications list, select Mark for Installation from the pop-up menu, and click Apply at the top of the window. Once the installation is complete, jEdit can be launched by selecting the Programming submenu from the Applications menu, then clicking the jEdit icon.

Basic instructions for using jEdit with regular expressions and for changing the end-of-line character can be found in the Windows jEdit section earlier in this appendix. If you prefer a native program rather than a Java application, try gedit, which is installed by default on Ubuntu. It is found in the Accessories submenu of the main Applications menu. You should also install the developer plug-ins (`launchpad.net/gdp`) to get regular expressions and other useful features. There are some reports that gedit cannot save directly to shared folders.

Using the Linux shell for shell operations in Chapters 4–6

To start using the command line in Ubuntu, launch the Terminal program from within the Accessories submenu of Applications. Since you will be using Terminal frequently, you might as well also drag its icon from Applications to the status bar at the top of the Ubuntu screen (see Figure A1.5).

Both OS X and Ubuntu use the `bash` shell by default, and nearly all the command-line programs used in this book that come with OS X also come with Ubuntu (install `curl` separately). This makes it simple to apply the lessons from the main text of the book directly to the Ubuntu command line. There are a few minor differences that could be confusing without some explanation, though.

The primary difference between OS X and Ubuntu at the command line is that the filesystems are arranged slightly differently. Home directories in Ubuntu are located in the `/home` folder, not the `/Users` folder employed by OS X. `/Users/lucy` on OS X would therefore correspond to `/home/lucy` on Ubuntu. On either type of system, `~/` is still the relative path to the home directory of the current user, even though those directories have different absolute paths. If you are using Cygwin on Windows, `~/` will also work, but there will be some additional path elements added before the home directory name.

If you are using Linux, or if you are having problems getting the instructions from Chapter 6 to work, check to make sure that you have the `bash` shell running, rather than `tcsh` or some other shell variant. To do this, in a terminal window, type:

```
echo $SHELL
```

It will return the name of the program that currently runs within your terminal window, which should be `/bin/bash`. If your system reports another shell, such as `tcsh`, `ksh`, or `csh`, you have the option to change the default shell over to `bash` using the command `chsh` (short for **ch**ange **sh**ell).

First, determine the full path pointing to the `bash` program using the `which` command:

```
which bash
/bin/bash
```

Then run `chsh` with the `-s` option, followed by the location of `bash` returned by that command:

```
chsh -s /bin/bash
```

The system will probably ask you for your login password to confirm this operation.

Setting up your `.bash_profile` Your `bash` settings might be stored in a different location than described in Chapter 6. On Ubuntu and some other Linux variants, the settings for `bash` are stored in the file `~/.bashrc`, rather than the `~/.bash_profile` file used by OS X. The default Ubuntu installation already has a `~/.bashrc` file, so you won't need to create it, you can just modify it as indicated in the text. You can add additional lines after the end of existing lines in the file.

If you are using `tcsh` or `csh` shells, then the command to modify your path should go into the file `.cshrc`, and you should use this command instead:[1]

```
setenv PATH $PATH:$HOME/scripts
```

Setting up aliases in `tcsh` is also slightly different from `bash`. Instead of the equals sign, you just put spaces between the shortcut and its definition:

```
alias ll 'ls -l'
```

Other shells might use `.profile` as the settings file. If you are restricted to using a shell besides those above, do a search to find the location of the user login file on your system.

Python on Linux for Chapters 8–12

Python is included as part of the basic Ubuntu installation, so the instructions included in the main text of this book should also apply to your system. In some Linux installations, the path to the `python` folder or the `env` program used in the `#!` line of a script may be different. You can check this by typing which `env` in a terminal window, and using that location in place of `#!/usr/bin/env python` at the beginning of your scripts.

Many third-party Python modules, including NumPy, are installed by default in Ubuntu. Others, such as Biopython, can be easily installed through the Synaptic Package Manager or your Linux distribution's package installer.

Working with MySQL for Chapter 15

In Linux, the user experience for MySQL should be the same as described in the text. You will not have the same MySQL Preference Pane to control launching the MySQL database server each time your computer starts up.

[1]Many hidden shell settings files end with `rc`, so storing `csh` settings in a file called `.cshrc` is not as odd as it may sound.

You can install MySQL at the command line or using the GUI through System ▸ Administration ▸ Synaptic Package Manager. If you do install MySQL with `apt-get` at the command line, it is a good idea to update your installer information first with the command:

```
sudo apt-get update
```

Then type this command in a terminal window:

```
sudo apt-get install mysql-server
```

It will ask you a few times if you want to set the `root` password. You can leave it blank for the purposes of this demo and change it later if you wish to make your databases work over the Web.

Working with vector and pixel art in Linux for Chapters 17–19

The open source graphics programs discussed in this book (Inkscape, ImageJ, and GIMP) can be installed on Ubuntu through the Synaptic Package Manager. Select the System menu at the top of the Ubuntu screen, then the Administration submenu, and click on the Synaptic Package Manager. Search for the programs you want to install, mark them for installation, and click the apply button at the top of the screen.

Appendix 2

REGULAR EXPRESSION SEARCH TERMS

Regular expressions—ways to perform adaptive searches and replacements—are described in Chapters 2 and 3. Here we provide a quick reference to some of the more common regular expression terms. This table and the text of the book itself do not encompass the entire range of regular expressions. There are many other useful constructs, for example, embedding miniature scripts into your replacement terms, and searching for A or B in a string using the syntax `(sword|jelly)fish`. If you would like to delve deeper, there are many online references, and there is even an in-depth reference guide built into the Help menu of TextWrangler.

There is some variation in the terms supported from program to program and from language to language. The most widespread terms, which can be used almost anywhere that regular expressions are supported, are the POSIX Extended Regular Expressions. These include `.`, `*`, `+`, `{}`, `()`, `[]`, `[^]`, `^`, `$`, `?`, and `|`. While quite a bit can be accomplished with the POSIX terms, in many implementations the language has been supplemented with some nonstandard terms. Most of these nonstandard terms are based on Perl regular expressions. These include many of the character class wildcards listed in the tables below, such as `\d`, `\w`, and `\n`. These extra wildcards make it easier to write clear regular expressions. Lack of support for Perl-like regular expressions is one of the most common causes of confusion when moving to a new programming context.

If you are using regular expressions in a new context but find that they don't behave as expected, or that they generate errors, check to see which regular expressions are supported by the tool you are using. POSIX does define its own set of wildcards, but the syntax is different from the Perl-style `\w` format that we use in this book. These wildcards include `[:digit:]` in place of `\d` and `[:alpha:]` instead of `\w` that we use in this book (though not including the digits). These POSIX character classes can be used in some contexts where Perl classes aren't available, including SQL queries and the command-line tool `grep`. If you don't want to switch between wildcard types, a more universal solution is to replace character class wildcards with an explicit character range, such as `[0-9]` or `[A-Z]`.