Computer Writing and Research Lab

White Paper Series: #060928-1

Automated Customization in the CWRL PC Lab

Will Martin
wdmartin@mail.utexas.edu
University of Texas at Austin

28 September 2006

Keywords: customization, software

Abstract: CWRL computers are on a "mirrored" system. This allows for quick software updates and ease of control by system administrators. However, it also makes it difficult for individual users to customize the settings on CWRL computers. This whitepaper addresses the issue of customization by offering one way to customize CWRL PCs.

The Problem

The CWRL labs are configured to make it possible for individual instructors and students to install custom software. All users have essentially complete administrative powers, with rare exceptions (such as altering the system clock). Changes such as program installation or configuration are not persistent: when the user logs out, the computer removes any changes made during the session and restores itself to a pristine state for the next user. Only the network administrator can make permanent changes to the system, and this is rarely done during the term, since adding or removing programs can sometimes lead to instability.

In general, this system is good. It allows lab users a great deal of flexibility, it removes any viruses or other malware from the systems, and ensures the greatest possible amount of up-time. At the same time, however, the lack of persistence across sessions means that users who frequently want to use custom programs must install them every time they log into a CWRL computer. This can be onerous. Furthermore, instructors who wish to use a custom application as part of a lesson must coach their students through the program's installation process. Almost invariably, one or two students will encounter problems during installation, requiring the teacher to intervene.

The Solution

Fortunately, it is possible to automate tasks like program installation. Nor is it especially difficult, at least in the basic forms. The remainder of this paper will describe how to do this, using a real example the author created for his own use as a guide. Automating the Mac labs will not be addressed, since I have not taught in the CWRL Mac labs, and lack the necessary experience to describe the process adequately. Rather than attempting to describe the process in abstract terms, the paper will take the form of a tutorial, with commentary. For best effect, follow the instructions as you read the paper.

Basic Automation

Taming the Installer

I often need to use FTP, and strongly prefer an open-source FTP client called FileZilla (http://filezilla.sourceforge.net/). I wanted to install that program each time I logged onto a CWRL computer. The program has an installer of its own, offering a typical Windows-style installation process. The first step to automating that process is to observe a regular installation. You may want to do this on a second CWRL computer, in case you wish to log in or out during the tutorial.

1. Download the FileZilla installer: http://prdownloads.sourceforge.net/filezilla/FileZilla_2_2_27_setup.exe?...
2. Save it to the Desktop.
3. Once the download is complete, double-click the installer.

As the installer runs, observe what you have to do:

1. Select your language (defaults to English).
2. Click "okay."
3. Accept the license agreement (the GPLv2).
4. Select any custom options you want.
5. Click "next."
6. Select the installation location (defaults to C:\Program Files\FileZilla).
7. Choose what Start Menu folder to put the icons in, or optionally choose not to create any icons.
8. Click "next."

9. Choose whether the program should remember passwords, and how it stores its configuration settings.
10. Click "Install."
11. Wait for the installation to finish.
12. Click "Close."

So the installation program requires eleven different actions from the user in order to finish installing. Most of those go by pretty quickly; if you were only doing this once, it wouldn't be a burden. But having to do it every single time you start the computer is a pain, particularly since all of the default settings work perfectly well. Even the "secure mode" settings are fine: it doesn't matter if FileZilla stores your passwords, because as soon as you log out, the computer will delete FileZilla completely.
So the first step to automating this is simply to suppress the installer's need for your attention. Most installation programs can already do this. The key lies in using the file from a command prompt rather than through the GUI. Thus:

1. Click the Start button.
2. Select "Run."
3. In the prompt, type in "cmd" (no quotes) and press enter.
4. Type "cd Desktop" to change to the directory containing files on your desktop.
5. Type "dir" (no quotes) in order to show a list of the files in the current directory.
6. You should see the FileZilla installer program listed.
7. Type in the FileZilla installer's file name (as of this writing, that file name is "FileZilla_2_2_27_setup.exe").
8. Press Enter.

The FileZilla installer will start again. Go ahead and cancel it, and then:

1. Return to the command prompt.
2. Type in the FileZilla installer's file name.
3. Press space, and then type in "/S". The entire command should look like "FileZilla_2_2_27_setup.exe /S".
4. Press Enter.

This time, as I'm sure you noticed, the installer ran without requiring any user intervention. It didn't even create any visible windows. Which is exactly what we want. The "/S" is

called a "command line switch." It controls how the program runs. In this case, the "S" stands for "Silent" - meaning that the installation process doesn't open any windows, and doesn't ask for user input. Most (not all) installation programs have such an option built into them. The trick to triggering an unattended installation, then, consists mostly of figuring out what command-line switch will work with the particular program you want to install. Here are some common ones:

1. "/S" will cause a silent install in any installer written using the Nullsoft Scriptable Install System (http://nsis.sourceforge.net/). (FileZilla's installer was written using this.) These are easy to identify, because they usually have the words "Nullsoft Install System" written into them in grey text underneath the license agreement.
2. "/SILENT" will cause a silent install in Inno Setup installers (http://www.jrsoftware.org/isinfo.php). These typically have a plain "exe" file extension.
3. "/qn" will do it in Microsoft Installer files (typically having an "msi" file extension).
4. The installer for Firefox takes the switch "-ma" for some reason.

Try these; if none of them work, take a look at AppDeploy.com's article on Triggering Unattended Installations (http://www.appdeploy.com/articles/commandline.asp). It details not only a number of common command-line switches, but also discusses how you go about discovering the right one for your program. (Basically: experimentation, Google, and luck.) Once you've got your command-line switch, half the battle is won.

Avoiding the Command Line

"But wait," you may say. "How is this any easier than just clicking my way through the installer? I don't want to have to type all those commands in every time I log in!"
Naturally not. Fortunately, this too can be avoided. The commands that you type in at a command prompt can also be stored in a file, called a batch file, which will execute them each in turn when you double-click the file. Let's make one now:

1. Right-click your desktop.
2. Move the mouse cursor down to "New."
3. Click "Text Document" in the sub-menu that appears.

4. A new text document will appear on your desktop. Press escape to accept the default file name.
5. Press enter to open the file in Notepad.
6. Type in the command from the last section ("FileZilla_2_2_27_setup.exe /S").
7. Save the file and close Notepad.
8. Rename the text file to "setup.bat".
9. Double-click setup.bat.

As you can see, a command prompt opens up, the command is run, the installer does its thing, and then the command prompt closes itself. At this point, you have all the basics in place - you can copy the FileZilla installer and setup.bat to your teacher folder, so that in future all you have to do to install FileZilla is copy those two files back to the desktop from your teacher folder, and then double-click setup.bat. However, there are still some final touches we should add.

1. Right-click setup.bat and select "Edit" to open it in Notepad.
2. Put the cursor at the end of the existing command and press enter.
3. Type in "del FileZilla_2_2_27_setup.exe" and then press enter.
4. Type in "del setup.bat" and then press enter.

The "del" commands will delete the specified files. In future, when you run setup.bat, it will cause FileZilla to install itself (silently), and then both the installer and setup.bat itself will be deleted so that they're not cluttering up the desktop.

You can run as many installers as you'd like from one setup.bat file. Mine runs three:

```
"Firefox Setup 1.5.0.7.exe" -ma
FileZilla_2_2_27_setup.exe /S
Combobulator.exe /SILENT
del "Firefox Setup 1.5.0.7.exe"
del FileZilla_2_2_27_setup.exe
del Combobulator.exe
del setup.bat
```

Note that one of those file names has spaces in it - be sure to put file names like that "in quote marks" so that they'll run properly. The command-line switch goes outside the quote marks.

Those are the basics; you can put together a decent automated customization routine with nothing more than what we've covered so far. If all you need to do is install a bunch of programs without causing yourself a headache every time you log in, then you can stop reading right now.

The Final Hurdle: Program Configuration

Dealing with Settings Data

But often, the default settings aren't exactly what you need. You may need to customize your program's behavior - set preferences, perhaps even install add-on extensions to the program's capability. What good is simply installing FileZilla if I have to manually enter the FTP sites every time I want to use it?

It is sometimes possible to restore your program's custom settings - but it's trickier than our earlier efforts. Furthermore, the exact approach you'll need to take will vary widely depending on the program you're configuring. I can't offer you a precise prescription for how to do this part. However, I can comment on the general process you'll need to go through. Basically, the process consists of four steps:

1. Configure your program the way you want it, taking careful note of the things that you change.
2. Figure out how your program keeps track of settings, and where it stores that data.
3. Collect the settings data.
4. Put together some way of restoring that data to its proper place each time you install the program.

FileZilla makes the first three steps easy on us. Adding FTP site information is a fairly straightforward process, done using the program's site manager (which I won't describe in detail here). The FileZilla installer itself told us where it keeps its settings data: remember the installer asking us where to keep the configuration data? The default setting, which we accepted, was "use XML file." And sure enough, a quick browse through the program's directory (C:\Program Files\FileZilla) reveals a file called "FileZilla.xml". There are no other XML files in the directory, so that's got to be the one. With that file in hand, all we have to do is make a copy of it, and add a command to our .bat file copying it back into the program directory after each installation.

However, FileZilla is an unusually simple case. A single XML file is pretty straightforward. Most programs have much more elaborate systems for keeping track of user settings. Recall FileZilla had another option, too. We could have chosen, during the installation process, to have FileZilla store its configuration settings in the Windows registry (a shared database of program data maintained by Windows) instead of an XML file. Other programs store their user preferences in multiple files. For example, my Firefox profile consists of 47 unique files scattered across fourteen directories.

Identifying how your particular program stores its configuration data will require research. Google is your friend; no matter what program you're using, chances are excellent that you're not the first person who ever wanted to figure out where it keeps its configuration data. You may find yourself using "regedit" to read through the Windows registry; or you may wind up tracking down hidden system files buried in a sub-sub-sub-sub-sub-sub-sub-sub-sub-directory. (Be sure your copy of Windows' file explorer is set to show hidden files if you do this!) Furthermore, you have to identify not just the files, but also their proper locations on the disk.
And then you have to figure out how to put it all back in place. Dealing with such complicated situations is going to take considerably more than a .bat file.

Using Inno Setup
In fact, what you need is a custom installer of your own, something that can put files in place, enter data into the registry, alter INI files, and do so automatically. For this task, I recommend Inno Setup, a program for writing installer programs. Recall the "Combobulator.exe" file in the example .bat file? It's a setup program written using Inno Setup.
Inno Setup can become complicated in a hurry. Here's an example file:

```
[Setup]
AppName=Combobulator
AppVerName=Combobulator 3
DefaultDirName={pf}\wscite
DisableDirPage=yes
DefaultGroupName=SciTE
DisableProgramGroupPage=yes
OutputBaseFilename=Combobulator
Compression=lzma
SolidCompression=yes

[Files]
; FileZilla Prefs (note: this
line is a "comment" that
Inno Setup ignores).
Source: "FileZilla.xml";
DestDir: "C:\Program
Files\FileZilla";
Flags: ignoreversion
```

All the stuff in the [Setup] section defines the behavior of the installer that Inno Setup will create when this script is compiled into a finished file. The single line in the [Files] section does the actual work: when Inno Setup compiles the installer, this line tells it to take a file called "FileZilla.xml", located in the same directory as the Inno Setup script itself, and compress it into the output file. Later, when the generated file (Combobulator.exe) is run, the FileZilla.xml file will be copied into its destination directory (DestDir). The "ignoreversion" flag tells the installer to ignore any existing copy of FileZilla.xml, and overwrite it if it's there.

All this may seem like an amazingly complicated way of doing the same thing you could accomplish with a .bat file. But remember my Firefox profile, with the 47 files and 14 directories? In the full version of this script, it puts all of those into place for me. All those files get reduced to a single file, making it easier to copy everything back from the teacher folder. Inno Setup can also manipulate data in the Windows registry, and in INI files. The full source code for Combobulator currently consists of 168 lines of code. That's fairly small as computer programs go, but in those few lines it undertakes and completes some fairly complicated tasks. Between them, my setup.bat and Combobulator.exe install three programs and configure each of them with all my preferred settings, automatically.

A full guide to writing Inno Setup scripts is well beyond the scope of this paper. However, Inno Setup includes extensive documentation. The help files are excellent. Everything I know about Inno Setup, I learned from reading its help files. And you don't even need to read those if all you want to do is copy a bunch of files to known destinations - Inno Setup includes a "script wizard" that can generate basic installers for you, based on your answers to a set of questions.

Final Remarks

This paper has been more an introduction than an exhaustive guide. If you only need to install programs repeatedly, and the default settings are acceptable, then you should now know all you need to in order to begin creating your own automatic installation scripts for use in the CWRL. If, however, you need or want to configure your programs after they've been installed, then this is only a basic introduction. Either way, I hope you find this information helpful in your CWRL teaching endeavors.

Addendum: Microsoft Word

Microsoft Word's configuration settings can be preserved without much difficulty.

1. Open MS Word with a BLANK DOCUMENT.
2. Set it up exactly the way you want (eg, type your own name into the user info section, turn off grammar checking, add keyboard shortcuts, whatever).
3. Save the document as a template. (File -> Save, select "Document Template" from the drop-down list, select a place to save it and a file name, eg my-template.dot.)
4. Close Word.
5. Go to Start -> Programs -> Microsoft Office -> Microsoft Office Tools -> Microsoft Office 2003 Save My Settings Wizard.
6. Go through the wizard (it will save your settings to a file, eg "my-settings.ops").
7. In your .bat file, the command "C:\Program Files\Microsoft Office\OFFICE11\PROFLWIZ.EXE /u /q /all /r C:\my-settings.ops" will restore your settings silently, assuming that your settings file is called "my-settings.ops" and it's stored in C:\.
8. Put the document template in "C:\Documents and Settings\YOUR-USER-NAME-HERE\Application Data\Microsoft\Word\STARTUP" and any

customizations to the toolbars or command shortcuts will become available the next time you start Word.

Putting the my-settings.ops and my-template.dot files in their respective places can be done easily using Inno Setup; the command to restore the Word settings from the OPS file can also be run from Inno Setup, using the [Run] section (see Inno Setup help file for details). For further info, check Microsoft's article on the MS Office Profile Wizard Command Line Options (http://support.microsoft.com/default.aspx?scid=kb;en-us;q281928), and also Stephanie Krieger's strangely-formatted blog entry Go Global! Create a Global Word Template to Customize Word (http://www.arouet.net/archives/2005/06/what_can_you_do.php).