Cygwin Installation for Developers

This tutorial briefly discusses the use of the Cygwin environment by the RFPK Software Team and then describes a procedure for obtaining and installing it.

Table of Contents

Introduction

Cygwin is a Unix-like environment that runs within a Microsoft Windows system. It is free, open source software. Cygwin consists primarily of shell commands and applications from the GNU/Linux environment. Because it supports the complete GNU suite of software development tools, including compilers for C, C++ and Fortran, many applications for the Linux environment can simply be transferred to Cygwin in source code form, compiled, linked and run. Because Cygwin supports **sh**, **perl**, **python** and the **cron** batch scheduling system, it is an excellent environment for batch production on a windows system. Most applications developed to run in the Cygwin environment can be transferred to Linux or Unix with almost no changes.

The first major application for the Cygwin environment by the RFPK Software Team is a platform for testing SPK. Key portions of this application are written in the **perl** and **sh** scripting languages. This tutorial describes the process of installing Cygwin on a Windows machine for developing or running this or similar applications.

Downloading and Installation

The software can be downloaded for free over the Internet, from the *Cygwin* [1] web site.

1. On the Cygwin home page, press the ***Install Now*** icon.

   A window will appear, asking you to provide the path to a directory into which to download `setup.exe`, the Cygwin downloading and installation wizard. The path of a temporary directory is recommended.

2. Execute the Cygwin **setup** program. You can use the Windows filesystem explorer to find the file you just downloaded, then double-click its icon.
3. The wizard will ask you to provide some information.
   a. Choose a download source. Select **Install from Internet** .
   b. Select Root Install Directory. Provide the name of a directory that will be large enough to contain your Cygwin environment. You may need quite a bit of space, depending on what optional components you select in the steps that follow. For example, the Cygwin installation on the author's workstation occupies about one gigabyte of space, and is contained in the directory `f:\cygwin` .
   c. Select Local Package Directory. Provide the path of a directory with temporary storage for the downloaded files used in the installation process.

d. Select Your Internet Connection. The choice **Direct Connection** seems to work fine at RFPK.

e. Choose A Download Site. A list of mirror sites is provided. Pick one that seems like it would be relatively close and might have good bandwidth. Universities are a good choice.

f. Select Packages. A list of all available packages is displayed in a scrollable window as a multi-level hierarchy. The top level category is named **All** . This top category name, and all of the subcategory names, are followed by a directive which, initially, is **Default** . By clicking repeatedly on a directive, you can change it from **Default** to **Install** to **Reinstall** and finally to **Uninstall** before the sequence repeats itself.

   If you want everything, all you have to do is change the directive for **All** to **Install** and then press the **Next** button to start the download.

   If you want less than everything, you will need to go to the individual categories and specify what you want in addition to the defaults. In general, a software developer will want at least the following:

Devel

cvs

expat

expat

gcc

gdb

make

rcs

Doc

cygwin-doc

expat

perl-manpages

Editors

vim

Interpreters

perl

python

Net

openssh

> Note that in the above list, the **Editors** category does not include **emacs** . This tutorial recommends downloading the gui version of **emacs** directly from Gnu rather than using the version available from Cygwin.

4. Proceed with the downloading. If, for some reason, the connection fails in the middle of this process, you can restart it with the same mirror or a different mirror and the process is supposed to resume where it left off.
5. Follow the wizard to completion. Allow it to create shortcuts on the desktop and in the **Start** menu.

Post Installation

After the installation completes, start up a terminal window by double-clicking the icon on the desktop. You should be started in your Cygwin home directory. To verify this, enter the following command after the shell prompt:

```
pwd
```

The response should be

```
/home/your-user-name
```

The default shell for Cygwin is **sh**, the Bash shell, which is a version of the original Bourne shell enhanced in several ways, including the addition of command-line history and editing. For portability, it is important to always write shell scripts for **sh** rather than for **csh**, **tcsh**, **ksh**, or any other shell. It is recommended that RFPK developers use **sh** for all editing. including command-line editing, in order to eliminate confusion and because the current version has incorporated nearly all of the useful features that were pioneered by one or another of the above-named alternatives.

The Bash shell uses several configuration files, that it will read when it starts up, if they are located in your home directory. A brief description of these files follows.

.bash_profile

Bash reads this file when you "log in." With Cygwin, you are, in effect, logging in when you double-click the *Cygwin* icon on the desktop or in the **Start** menu.

The primary use of this file is to define environment variables. For example, the `PAGER` environment variable tells programs which paging command to use. The default is usually the **more** command. Most people find, however, that the **less** command is superior. You can instruct **sh** and other programs that query the `PAGER` variable to use **less** by placing the following commands in your `.bash_profile`

```
PAGER=less
export PAGER
```

Note that you should not depend on your `.bash_profile` being read prior to execution of a shell script, because the shell script might not be run from a log-in shell. For example, there is no login shell for scripts run by **cron**.

.inputrc

This file is used to specify that you would like to have the command-line editing feature enabled and which mode of editing, either **vi** or **emacs** you prefer. For example, the following entry will provide command-line editing that behaves like **vi**:

```
set editing-mode vi
```

.bashrc

Bash reads this file whenever it is started interactively. This occurs when you log in, as explained above, or when an additional bash shell is spawned by a shell that is already running interactively.

The primary use of this file is to define aliases. For example, the printer interface command, **lpr**, may need a number of arguments to specify exactly which printer to use. If you always use the same printer, you can redefine **lpr** with the following alias:

```
alias lpr='lpr -P'\"Xerox DocuPrint N32 PS'\"'
```

Note that you should not depend on your `.bashrc` file being read prior to execution of a shell script, because the shell script might not be run from an interactive shell. For example, there is no interactive shell for scripts run by **cron**.

Notes

1. http://www.cygwin.com/