Pascal scripting

Michaël Van Canneyt

July 7, 2011

Abstract

System administrators often write small scripts to handle routine tasks. Bash and Perl are often used for this purpose. A small tool included with Free Pascal allows to use pascal for this task. The tool is called `instantfpc` and will be discussed here.

1 Introduction

System administration often consists of automating tedious tasks: cleaning up directories, taking backups, compressing log files. Simple tasks are often performed with shell scripts, which are interpreted and executed by one of the many unix system shells, such as bash. For more complicated tasks - for example tasks that require processing some text files - perl is routinely used instead of the shell.

The nice thing about shell scripting is that there is no compile step involved. On a system where the C language, with its cumbersome string handling and tedious compilation step, is the main development language, it is no surprise that system administrators preferred a simple scripting language to automate tasks: scripts are more easily developed. Since execution speed is not an issue, this makes perfect sense.

The downside is that the scripts are interpreted and some errors - as in all interpreted languages - pop up only by repeatedly running and rewriting the script. This is where pascal scripting can ease the life of the system administrator.

2 instantfpc

The recent addition of instantfpc to Free Pascal allows the system administrator to write a small script in pascal and execute it as he would execute a shell script. What is required for this is a recent snapshot of Free Pascal: the tool has not yet been distributed. Writing a pascal script is done in exactly the same way as one would write a shell script, as shown in the following lines:

```
#!/usr/local/bin/instantfpc
begin
  Writeln('Hello world from instantfpc');
end.
```

Note the first line (the shebang line).It contains a reference to the instantfpc tool. Save this in the file hello and make it executable:

```
chmod 755 hello
```

Now the script can be run:

```
home: >./hello
Hello world from instantfpc
```

Note that instantfpc - as a safety precaution - does not accept all extensions. Only the usual free pascal and Lazarus extensions are accepted, as well as no extension or an extension .cgi.

3 Architecture

How does it work ? The instantfpc program is a small wrapper around the free pascal compiler. When given the `-h` command-line option, it will display a list of available options which explain how it can be used. To use it for a script, the technique of the shebang line can be used, meaning the first line of the script should be

```
#!/usr/local/bin/instantfpc
```

It can even be used to specify command-line options to the compiler. The following instructs the compiler to enable object pascal extensions:

```
#!/usr/local/bin/instantfpc -S2h
```

What will happen is that instantfpc will strip the first line of the script and compile the resulting script with the Free Pascal compiler. If compilation is successful, the resulting binary will be executed and will be passed the command-line options that were given to the script:

```
#!/usr/local/bin/instantfpc -S2
begin
  writeln(cmdline);
end.
```

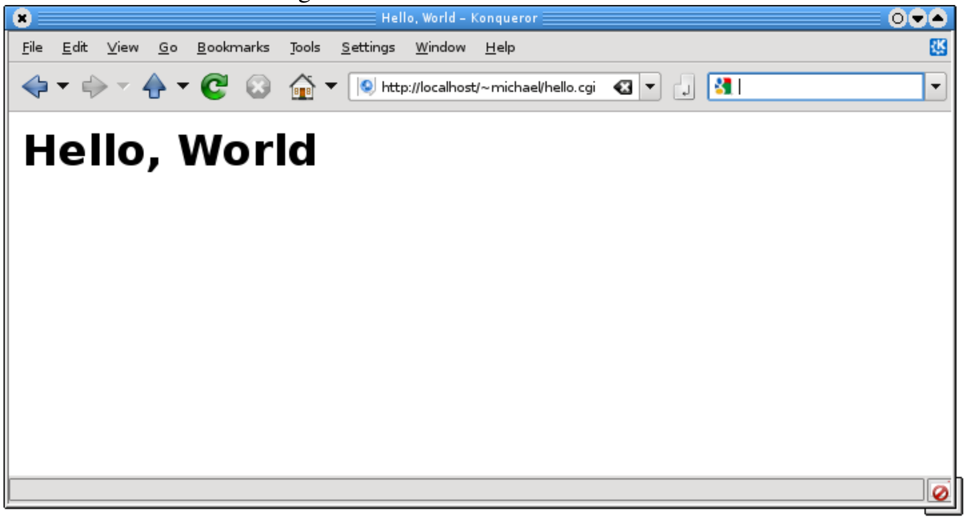
will output:

```
home: >./echo
./echo
home: >./echo with some options
./echo with some options
```

Naturally, it would be not very efficient to compile the script each time it is run. To remedy this, instantfpc keeps a cache of scripts it has compiled and the corresponding executable. Each time it is asked to run a script, it compares the script to the one in the cache. If the script has not changed, the cached compiled executable is immediatly executed. The cache directory can be displayed with the command-line option `-get-cache`. By default, it is in a subdirectory `.cache/instantfpc` of the user's home directory (it will be created if it does not yet exist).

When writing scripts, all units in the Free Pascal distribution are at the disposal of the script writer. If this is not enough, the user would be well advised to creat a unit with commonly encountered tasks in a script - such as get a list of all filesnames in a directory. Properly installed, this unit can then be used in all scripts on the system.

Figure 1: Hello world from instantFPC


Hello, World - Konqueror
File Edit View Go Bookmarks Tools Settings Window Help
http://localhost/~michael/hello.cgi

Hello, World

4 Web scripting

An interesting possibility is writing some simple CGI scripts. instantfpc is picky about the extensions it accepts in a script, but .cgi is an accepted extension. Since the webserver uses a reduced environment when calling a CGI script, no home directory or path may be available, and instantfpc needs to be told where to find the compiler and where it can cache the compiled scripts. This can be done with the `-compiler` and `-set-cache` command-line options. Thus, the following script will produce a nice greeting in your browser:

```
#!/usr/bin/instantfpc --compiler=/usr/bin/fpc --set-cache=/tmp
begin
  Writeln('Content-Type: text/html');
  Writeln();
  Writeln('<HTML><TITLE>Hello, World</TITLE>');
  Writeln('<BODY><H1>Hello, World</H1></BODY></HTML>');
end.
```

If it is saved in the user's public HTML directory with the name `hello.cgi`, then the following screenshot shows the result: Changing the script and refreshing the page in the browser will automatically recompile the script. In a live environment, it is wise to choose another cache directory, but make sure it is writable by the user that the webserver runs as.

5 Conclusion

People that like pascal can now also use it as a scripting tool. The instantfpc tool automates and largely hides the task of compilation, making it an easy replacement for regular scripting. The strict syntax and typing of pascal make the scripts less error prone, which is an advantage for more advanced scripts. The additional caching makes it also usable as a quick web scripting tool.