PROGRAMMING NOTES

WINDOWS ASPECT : A Scripting Language A Tutorial - Part One
For PROCOMM for WINDOWS v.2

GHOST BBS v.3.20 © 1994 *by Gregg Hommel*

Some time back, about six months after I first installed my copy of Procomm Plus for Windows v.1, I took a serious look at the Host mode supplied by Datastorm. Earlier I had written a freeware script to manage the widely accepted *QWK* mail packet format on a PCBoard BBS running a Qmail mail door. I called the script PCBMail (still in use by a lot of people to this day), and decided to develop a shareware version of it, to be called PCB Freedom. PCB Freedom would extend beyond the rather restrictive limits of PCBMail into a more generic mail management utility.

I thought that perhaps the Datastorm supplied Host mode would offer an additional method of supporting the users of both scripts. It took only a fairly quick look for me to realize that, although Host mode was sufficient to handle the task at hand, it certainly was nothing more than barely adequate, as far as I was concerned. Not to disparage the Datastorm Host script, but I found it "ugly" in appearance, and somewhat in need of a few other modifications in other areas.

Thus began a saga which led to GHOST BBS that continues to this day. It led me to discover things about the Windows Aspect (Wasp) programming language of Procomm Plus for Windows, that I did not dream were possible when I first read their manuals. All of this led me into writing and posting on various BBS networks a twelve part tutorial in Wasp programming.

Although some information in that tutorial is now out of date, due to the release of Procomm Plus for Windows 2.0 with its new version of the Wasp language, much of it still applies. It is my intention to update that tutorial as necessary, and to post it in this ezine as a semi-regular column whenever the editors will let me stick one in an issue. So far they seem agreeable to the idea, but editors have been known to change their minds before. Red flag and charging bull mode *ON* ! *Talk about a public challenge! YOU are publicly hired at your current stipend for as many tutorials as you can come up with. Wise guy!* This column is the first in that (semi-?)regular series of columns.

When I first began looking at modifying the Host script for my needs, I admit it... I was a neophyte, with a whole truck load of naiveté trailing behind me. I had already written and posted PCBMail and had a beta version of PCB Freedom in

the works, but on a comparative basis, those scripts were not remotely close to what I had gotten myself into with what eventually became known as GHOST BBS.

For those of you who already know what GHOST BBS is, please skip to the next paragraph. For those that don't.... GHOST BBS is a shareware replacement for the Datastorm provided Host mode which ships with Procomm Plus for Windows. It features additional security levels, multiple log-on bulletins, multiple download libraries, support for ANSI and non-ANSI menus and displays support for multiple languages, editable prompts and displays, and many other enhancements over Host.

Logging onto a BBS, getting a mail packet, and uploading a reply packet is a relatively simple procedure. You let the communications application do most of the work. All the script really has to do is to watch what is happening on the BBS and send the appropriate responses that will perform the desired actions. The script doesn't have to read and then handle everything coming in the port. It only needs concern itself with that incoming data which requires a response or action. The rest of it can be ignored.

Not so with "host" scripts. They have to watch and interpret *everything* coming in, and expect the unexpected. The hardest thing to adjust to in writing GHOST BBS was the fact that my script was no longer the passive object, acting only upon a limited set of selected items from an external source. It was now the active object, responding and initiating to all information being sent by an external source.

There is much more to it than that, of course. An example, -- when you log onto a BBS, effectively Procomm manages what appears in the terminal window while your script looks at that information and, when certain key words or phrases are present, responds to them. With a host mode, the script manages what appears both in your terminal window, and on the remote end, based upon characters, etc. sent to it by the remote, which it must trap, interpret, and act upon. Not quite as simple as a log-on script, even one as complicated as PCB Freedom became.

This need to have the script do everything also caused a second problem, at least, under Procomm Plus for Windows 1.0x. The Wasp 1.0 compiler has certain very restrictive memory limits made upon a script at compile time. These limits involve a combination of factors, including the total number of global and local variables in use and the total number of "nested" code segments in the script. PCB Freedom was fairly complex, but nothing compared to the code required to make GHOST do what I, and my users, wanted it to do.

Indeed, GHOST was so complex a collection of code that when I released

GHOST BBS 3.00, the first shareware version of GHOST, I thought that it would actually be the last version of GHOST released, period !! I was able to successfully compile the script, but only by using the PCP for Windows v.1.01 of the Wasp compiler. It had, if memory serves, something like 149 *bytes* of extra memory available for use at compile time, versus the compiler in PCP for Windows v. 1.02. I didn't think that it would be possible to add any additional features or enhancements, unless or until Datastorm delivered a compiler with more available compile time memory.

But, in the end, I did. GHOST BBS 3.10/3.20 is currently in final beta test, and has a lot of new features. How I did this, and what it taught me about programming in Wasp is the point of all this rambling.

Although it is quite possible that you will never write any script as complex as GHOST BBS, the things that I learned, or was forced to learn, in the process of writing GHOST expanded my knowledge of Wasp and how to programme using it at a fairly rapid pace. I don't know if that is enough to qualify me to offer a series of columns on Wasp programming, but it will certainly help and the editors of *WindoWatch* seem to think that it will work, so we'll give it a shot.

This column then, on an *irregular* (sheesh!) basis, over the next few months at least, will offer tutorials, based upon my experiences as a Wasp programmer using that script language and writing usable scripts with it, starting with this column....

At the risk of losing some of you, we are going to start at a *very* basic level, with the question "What is a script?".

In Wasp, a script is a series of commands which Procomm Plus for Windows will read when told to, and execute as specified. These commands are written using a particular form, following a particular syntax, to instruct Procomm Plus for Windows to perform various tasks at specified times, and in a specified order.

A script can be a complex thing telling Procomm Plus for Windows how to do almost everything. It can leave very little for Procomm Plus for Windows to do on it's own without instructions from, for example, GHOST BBS. It can also be a simple little set of commands to automate a given procedure or task.

One of the most common, and simplest scripts many Procomm Plus for Windows v.1.0x users were interested in writing had to do with the barest attempt at automating their log-on to the bulletin boards they used. What they wanted to do was to create meta keys in Procomm Plus for Windows which would send their user name or password out the comm port when they were asked for the information by the BBS.

At this point, we shall "create" an imaginary, neophyte Procomm Plus for Windows user named George. He wants to use Procomm Plus for Windows to access several different bulletin boards, and like all good BBS'ers, uses different passwords on each system. But, he also wants to automate sending his name and password, because that is easier than trying to remember the correct information for each system when he logs on to it.

To accomplish this, George might start out by assigning scripts to be run when each of two meta keys are selected. This is how a lot of Procomm Plus for Windows 1.0x users get their first taste of script writing. This process is not a concern under Procomm Plus for Windows 2.0 that it was under 1.0x. In version 2 there are icons which perform these functions which can be added to your action bar and don't require the attachment of a script. However a lot of Procomm Plus for Windows 2.0 users still use these meta keys for that job.

Let's look at the two scripts that he could write to be attached to those meta keys...

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;SENDNAME.WAS - send the UserID for this system when meta key used

proc main
  transmit $USERID
  transmit "^M"
endproc

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
;SENDPASS.WAS - send the password for this system when meta key used

proc main
  transmit $PASSWORD
  transmit "^M"
endproc

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -- - - - - - - - - - - - - - - - - - -
```

Although not fancy scripts... in truth quite simple, they do the job, and automate the task of having to remember the name and password George uses on each BBS he logs onto *and* they introduced George to the concept of a script and what it does, along with a couple of basic principles of script writing.

1) *ALL* scripts must contain at least one procedure (proc) called MAIN.

2) *ALL* procedures in a script must begin with a "proc name" statement and mark their end with the "endproc" statement.

3) Variables in Wasp which begin with a "$" are pre-defined, system variables which contain values obtained from within Procomm Plus for Windows and its setup.

All of that from such a simple script ! Of course, for many users, this turned out to be just a start... once they discovered that it was a fairly simple task to automate a portion of their log on and they wanted more. So, we'll help George get a little further along, by writing a simple log on script to be used.

Let's assume, for now, that every system George uses, right after connection, asks for his name, and then his password, using the prompts "What is your name?" and "What is your password?". George might then write a script to automate this procedure that might look something like...

```
proc main
  waitfor "name?"
  transmit $USERID
  transmit "^M"
  waitfor "password?"
  transmit $PASSWORD
  transmit "^M"
endproc
```

A script such as this would be "attached" to the dialing directory entry for each BBS, to be run after a connection is established - the default for Procomm Plus for Windows. In truth it is really not much more than the two scripts George first had, but combined into one script and with two "waitfor" commands added. The WAITFOR command in Wasp is fairly straight forward in concept, and easy to understand, even for a George, which is why we used it here.

We now have our imaginary, neophyte Procomm Plus for Windows user writing his first script. But George is an incurable optimist, and *knows* that he can do better, especially so, now that he is getting more sophisticated. He has added a couple of new systems to his list... systems which do not follow the simple log on pattern of his earlier systems. Now he wants to automate those log-ons which is where we will leave George for now in the hopes that the editors want me to write a second one of these.....

Gregg Hommel has been a Data Storm beta tester and is presently a consultant for Delrina. He is an active participant on many nets and is presently Co-Host of the RIME Windows conference.
Your questions and comments to gregg.hommel@canrem.com

Editorial Note: Gregg and I tease but we both know he is always welcome here. Do let us know what you think about these tutorials - either to Gregg directly or to lois.laulicht@channel1.com

NEXT