Scripting Perforce

Setting the Perforce Environment

In the scenario for this set of exercises, your system administrator has already set up your local environment variables so you can connect to your Perforce server and has assigned you the super user name "bruno" with a password "brunopass" and a client workspace spec named "bruno_ws." You have a Perforce server running at security level 1 on your local machine listening on port 1666.

Your objectives for this exercise:

- Familiarize yourself with your programming and Perforce environment.
- Create a Perforce user name for your scripts.
- Add your script's user name to a group with a long session timeout.
- Grant appropriate permissions in the protections table to your group.
- Set a password for your script's user name and log in.
- Create a client workspace specification for your script's user name.

1. Run "perl -v" to view the version of Perl installed.

   perl -v

   This is perl, v5.8.8 built for MSWin32-x86-multi-thread (with 18 registered patches. Use perl -V for more detailed version info).

2. Run "p4 info" to find your client workspace root folder.

   p4 info

   ```
   C:\ p4 info
   User name: bruno
   Client name: bruno_ws
   Client root: c:\bruno_ws
   Server address: localhost:1666
   Server root: C:\Perforce
   ```
   ...etc.

3. Log into Perforce.

   p4 login

   ```
   Enter password: brunopass
   User bruno logged in.
   ```

4. As super user bruno, create a Perforce user name "scriptuser" to be used by your scripts.

   Hint:

   > Use the administrative command *p4 user -f* to create the new user specification.

   p4 user -f scriptuser

5. Include your script's user name "scriptuser" in a "scriptsonly" group and set appropriate limits for data queries and log in time:

```
MaxResults:    10000
MaxScanRows:   50000
MaxLockTime:   10000  (10 seconds)
Timeout:       172800 (48 hours)
Owners:        bruno
Users:        scriptuser
```

p4 group scriptsonly

Add the field values listed above.

6. One of the scripts in your database submits files to Perforce. Grant write permission to the "scriptsonly" group to the //Sim/... depot folder. You'll also need the scripts in the //depot/... depot folder. Grant at least read permission to the "scriptsonly" group to //depot/... Optional: If you wish, edit the first line in the protections table and change the "write user * * //..." line to be something like "write group perfusers * //..." and include the other Perforce users in the group. This can easily be accomplished using the Administration tool within P4V.

p4 protect

Add these lines to the protections table:

```
Protections:
      write group scriptsonly * //Sim/...
      read  group scriptsonly * //depot/...
```

7. Create a client workspace root folder, "C:\scripts," and change to that folder.

**cd **
mkdir C:\scripts
cd C:\scripts

8. Create a file called "config.txt" in your "C:\scripts" folder using Notepad and add these lines to the file:

P4USER=scriptuser
P4PORT=localhost:1666

Notepad config.txt

Add these lines, save the file and exit Notepad:

P4USER=scriptuser
P4PORT=localhost:1666

9. Set the P4CONFIG environment variable to point to "config.txt."

p4 set P4CONFIG=config.txt

10. Your script's user name "scriptuser" will need a strong password at security level 1. Set the password for "scriptuser" to "Scriptpass" and login.

p4 passwd
Type the new password, **Scriptpass,** *twice.*

p4 login
Type the password **Scriptpass** *to log in.*

11. Create a client workspace spec for your script's user named "script_ws." Use the "p4 set" command to set your P4CLIENT environment variable to "script_ws."

    p4 client script_ws

    Edit the workspace root:

    ```
    Root:  C:\scripts
    ```

    Save your client workspace specification.

    p4 set P4CLIENT=script_ws

12. Sync all the files in your server to your client workspace.

    p4 sync

13. Go to the "C:\scripts\depot\Misc\perl" folder.

    cd C:\scripts\depot\Misc\perl

A Submit Script

A Perl script, submit.pl is in your "perl" folder. It performs a submit without invoking an editor.

Your objectives for this exercise:

- Script submitting files without invoking an editor, using the command line program, p4.
- Add an error check to be sure the data you report to the user is valid.
- Time permitting, write a routine for submitting files using a pre-2006.2 server.

1. Open submit.pl and have a quick look at the script.

   Notepad submit.pl

2. Open one or all files under the "//Sim/Prod/MAIN/src/..." folder for edit and submit the files using the submit.pl script.

   p4 edit //Sim/Prod/MAIN/src/…
   submit.pl

3. Look for the "Exercise Hint 1" and "Exercise Hint 2" sections in submit.pl and modify these sections to skip the "*p4 describe*" section of submit.pl if a submit fails.

   See **//depot/Misc/perl/answers/submitclean.pl** *for a suggested solution.*

   How does the suggested solution differ from yours?

4. Optional: If your production server is older than 2006.2, rewrite the submit operation to print an edited "p4 change -o" array to a file and then run "p4 submit -i" using your file name. The solution includes an error checking section as in the previous exercise.

   See **//depot/Misc/perl/answers/submitold.pl** *for a suggested solution.*

   How does the suggested solution differ from yours?

A Release Notes Script

P4Perl Notes: *A Perl script, p4template.pl, is included in your "perl" folder. It implements the generic methods discussed in the lecture, for initializing P4Perl, running a command, printing errors, and printing the results of your command. You may wish to run p4template.pl to see the difference between running p4 info with tagged data output on or off.*

A Perl script, relnotes.pl is in your "perl" folder. It prints a set list of changelist numbers for a specific path.

Your objectives for this exercise:

- Create a text file that contains a specified number of changelist descriptions that characterize a "release."
- Add job descriptions to your release notes file.

1. Run "relnotes.pl" Add to main a call to the writedata subroutine using the hints supplied under "MAIN PROGRAM" and "WRITE TO A FILE" sections of relnotes.pl. Your goal is to store the description of each changelist to a file.

   See **//depot/Misc/perl/answers/relnotes1.pl** *for a suggested solution.*

   How does the suggested solution differ from yours?

2. Add job descriptions for each fix associated with your selected changelists.

   See **//depot/Misc/perl/answers/relnotes2.pl** *for a suggested solution.*

   How does the suggested solution differ from yours?

A Review Daemon

A Perl script, p4review.pl is in your "perl" folder. It writes email messages for subscribed users.

Your objectives for this exercise:

- Update the protections table so your "scriptuser" user name can update the review counter.
- Modify some user specifications so the users will receive email for submits to selected paths.
- Run the p4review.pl script to process all submitted changelists in your Perforce server.
- Examine the "emails" file to see how email messages are formatted.
- Add subroutines to the p4review.pl script to send email to subscribed users when Perforce jobs are modified.

1. As super user "bruno" add review privilege for the "scriptsonly" group to the protections table for the //Sim/... depot.

   p4 -u bruno protect

   Add this line to the protections table

   ```
   View:
         review group scriptsonly * //Sim/…
   ```

2. As super user "bruno" edit a couple of user specifications in your server and add a path or two to the `Reviews:` field. Add //depot/jobs to at least one user's spec.

   p4 -u bruno user-f dai

   ```
   Reviews:
         //Sim/Prod/MAIN/…
         //depot/jobs
   ```

   ... etc.

3. Start the p4review.pl review daemon running. The review daemon will print "Finished polling" when it completes one poll. Use "ctrl C" to stop the daemon. It should have processed all of the submitted changelists in your server. Open the "jobemails" file to see how each email message is formatted.

   p4review.pl

   Wait for the message "Finished polling." then use
   Ctrl-c *to stop the script.*

   Notepad jobemails

4. Add appropriate subroutines to the review daemon so it will send email to users with the `Reviews:` field set to "//depot/jobs" when jobs are edited. Hints are included in the p4review.pl script.

   See **//depot/Misc/perl/answers/p4review1.pl** *for a suggested solution.*

   How does the suggested solution differ from yours?

   Notepad jobemails

5. Start your review daemon again and edit a few jobs and submit a few changelists to be sure your daemon is working properly.

   Check your "jobemails" file to be sure data is being stored properly.

Implementing Triggers

*A Perl script, checksubmit.pl, is in your "perl" folder. It examines the file set being submitted to be sure that when " *.c " files are edited a RELNOTES file is included in the changelist. Also located in your "perl" folder is, a Perl script, checkworkspace.pl that checks for malformed client workspace specifications. These types of scripts can be used to enforce your site policies.*

Your objectives for this exercise:

- Specify the checksubmit.pl script as a change-submit trigger and try submitting changelists with various combinations of files to make sure it operates properly.
- Specify checkworkspace.pl as a form-save trigger and test the trigger by creating and saving a new client workspace specification without editing the form.

1. As super user "bruno" install checksubmit.pl as a change-submit trigger. Pass the trigger the variables %changelist% and %client% . Test your trigger by opening various combinations of files and attempting to submit your changelist.

   p4 -u bruno triggers

   ```
   Triggers:
      ckrel change-submit //... "perl C:\scripts\depot\Misc\perl\checksubmit.pl %changelist% %client%"
   ```

   p4 edit //Sim/Prod/MAIN/src/sim.c //Sim/Prod/MAIN/src/RELNOTES
   p4 submit

   Your trigger should pass.

2. As super user "bruno" edit the triggers table and add the checkworkspace.pl script as a form-save trigger. Pass the %formfile% variable to the trigger Create and save a new client workspace specification without editing the form to test the trigger.

   p4 -u bruno triggers

   ```
   Triggers:
      cclient form-save client "perl C:\scripts\depot\Misc\perl\checkworkspace.pl %formfile%"
   ```

   p4 client testworkspace

   Save the form without editing and the trigger should fail.

3. Time permitting, think about how you would make the first trigger, checksubmit.pl, more specific for making sure that when the file RELNOTES is submitted with " *.c" files they are both under the same folder in your server.

   Hint: *The file path is available in your trigger as $1 in the regular expression.*

Creating a Temporary Client Spec

A Perl script, "p4client.pl" is in your "perl" folder. It edits a default client specification to change and save parameters in the `Options:` *and* `View:` *fields. This type of script may be useful for such activities as creating a new client specification on a machine for syncing a specific set of files such as files tagged by a label.*

Your objectives for this exercise:

- Set the P4CLIENT environment variable to auto-client and run the p4client.pl script.
- Check the saved client specification.
- Edit the p4client.pl script and add your own client specification modifications.

1. Unset P4CONFIG and set the P4CLIENT environment variable to "auto-client" so the p4client.pl script will read the new client workspace specification.

   p4 set P4CONFIG=
   p4 set P4CLIENT=auto-client
   p4 info

   Client name: *should be* **auto-client**

2. Run p4client.pl and examine the stored client workspace specification to be sure it has been stored correctly.

   p4client.pl

   This script produces no standard output.

   p4 client

   Examine the `Options:` *and* `View:` *fields for the changes made by the script.*

3. Modify the p4client.pl script and make some additional changes to customize the client specification form.

   Suggested modifications:
   - Add a description
   - Change the submit options
   - Add more paths to the view

Detecting Client Workspace Abuse

A Perl script, "clientspecabuse.pl" is in your "perl" folder. It reports workspaces with an empty `Host:` *field and users who have configured more than a set number of client workspaces.*

Your objectives for this exercise:

- Run the clientspecabuse.pl script and see who is violating your client workspace configuration policy.
- Add a test of your choice to help enforce your site's policy.

1. Run the clientspecabuse.pl script to see who is abusing client workspace specifications on your server.

   clientspecabuse.pl

   The output of the script is a list of client workspace names that match the script parameters, thus violate the stated policies.

2. Add a test of your choice to examine the client workspace or user specifications on your server.

   Suggested tests:
   - List workspaces with an "open" view mapping, e.g. //Sim/... //bruno_ws/...
   - Check that the "rmdir" option is always set.
   - Does each user specification have a valid email address?

Congratulations! You have completed the Perforce Training Course. We hope this course material has prepared you to use Perforce with confidence and ease.

