Introduction to Perforce for Users

FILE OPERATIONS

You may complete these exercises using the command line or P4V client program. Exercises that require using the command line only are noted as such. The answers use command line syntax. Notepad, the default text editor for Perforce forms on a Windows platform, will be used throughout these exercises for editing both Perforce forms, such as changelists, and for text files. The steps used to answer the questions in all exercise sets are more important than the results you get within this demo database, as an aid to adapting what you learn in class to your work situation at home.

In the scenario for these exercise sets, you are a new Perforce user, starting to work on files in an existing Perforce database. Your system administrator has already set up your local environment variables so you can connect to your Perforce server and has assigned you the user name "bruno," and the client workspace spec name "bruno_ws". You have a Perforce server running on your local machine, and you will be primarily working on the 'Jam' project data files.

Your objectives for this exercise:

- To perform fundamental operations required for manipulating files in your local client workspace.
- To submit your completed work to your Perforce depot.

1. What is your connection information, including:
   - the port address of your current Perforce server?
   - your effective Perforce user name?
   - the name of your current Perforce client workspace spec?
   - your client workspace root directory?

   p4 info

2. Fill your client workspace with files from the depot. What command did you use?

   p4 sync

3. In the c:\bruno_ws\Jam\MAIN\src directory, open jam.c and jam.h for editing.

   cd c:\bruno_ws\Jam\MAIN\src
   p4 edit jam.c jam.h

4. Open variable.c and variable.h for deletion.

   p4 delete variable.c variable.h

5. Use notepad to create a new file, save it in your workspace and open the new file for adding to Perforce. (Recall notepad adds ".txt" to your file name when you save the file).

   notepad myfile.txt
   p4 add myfile.txt

6. What command lists the files you have open?

   p4 opened

7. What command would remove jam.c from the list, and return it to its original state?

   p4 revert jam.c
   or:
   p4 revert c:\bruno_ws\Jam\MAIN\src\jam.c
   or:

p4 revert //depot/Jam/MAIN/src/jam.c

8. Edit jam.h using notepad, then submit your changelist. What is the changelist number?
   notepad jam.h
   Add a few lines of text to the file and save, then exit notepad.
   p4 submit
   Edit the description line in the form and save, then exit notepad.
   The changelist number is displayed after you save & exit the editor.

9. Open filent.c for editing, and make a small change.
   p4 edit filent.c
   notepad filent.c
   Add a few lines of text to the file and save, then exit notepad.

10. Rename (move) filent.c to file7.c. How does file7.c differ from the depot revision of filent.c which you last synced? Submit your change.
    p4 move filent.c file7.c
    p4 diff file7.c
    p4 submit
    Edit the description line in the form and save, then exit notepad.

FILE INFORMATION

Your objectives for this exercise:

- To be able to retrieve information about file revisions in the Perforce depot
- To determine the current state of files
- To find out what and when changes were made to files.

1. Change //depot/Jam/MAIN/src/Build.mpw so that when it is synced to any client workspace it is locked for exclusive open and its original modtime is preserved.
   p4 edit -t text+lm //depot/Jam/MAIN/src/Build.mpw
   or
   p4 edit -t +lm //depot/Jam/MAIN/src/Build.mpw
   p4 submit

2. List depot files under the //depot/Talkhouse/main-dev/com folder that have the extension ".txt."
   p4 files //depot/Talkhouse/main-dev/com/….txt

3. Sync all the revisions of the files in //depot/Jam/MAIN/src to your client workspace in the state they were in as of changelist 165.
   p4 sync //depot/Jam/MAIN/src/…@165

4. What files were affected by all changelists from changelist 1 to changelist 172 inclusive? What files were affected by changelist 172?
   p4 files @172
   and
   p4 files @172,172

5. How does the March 24, 2002 version of jambase.c differ from the April 1, 2002 version? (You will need "p4 diff2" for this).
   p4 diff2 jambase.c@2002/3/24 jambase.c@2002/4/1

6. In the //depot/Jam/MAIN/src directory, how are versions #19 and #20 of make.c different?
   p4 diff2 make.c#19 make.c#20

7. What files in //depot/Jam/MAIN/src had content changes during March 2002?
   p4 diff2 -q //depot/Jam/MAIN/src/...@2002/3/1 //depot/Jam/MAIN/src/...@2002/4/1

8. Which revisions sync'd (#have) to your workspace differ to the revisions tagged in the label 'jam-1.1.0'?
   p4 diff2 -q …#have …@jam-1.1.0
   or
   p4 diff2 -q //depot/Jam/MAIN/src/...#have //depot/Jam/MAIN/src/...@jam-1.1.0

9. Without deleting any files from the depot (do **not** use "p4 delete"), remove from your client workspace all the files under //depot/Jam/MAIN/src.
   p4 sync //depot/Jam/MAIN/src/…#none
   or
   p4 sync //depot/Jam/MAIN/src/…#0

10. How can you find out what a sync would do without actually syncing the files?
    p4 sync -n

11. Bring your client workspace up to date with the head revisions of the files in your client workspace spec view.
    p4 sync

12. Which change by earl FINALLY fixed the jam "include" bug in //depot/Jam/MAIN/src/compile.c?
    p4 changes -l -u earl //depot/Jam/MAIN/src/compile.c
    or
    p4 filelog -l //depot/Jam/MAIN/src/compile.c
    Examine the descriptions to find the answer.

13. Go to the c:\bruno_ws\Jam\MAIN\src directory. What version of make.c is in your client workspace?
    cd c:\bruno_ws\Jam\MAIN\src
    p4 have make.c

14. In what revision of //depot/Jam//MAIN/src/command.c was the “command too long\n” line deleted? What is the range of changelists in which the line was included?
    Hint: This may be easier in P4V.
    p4 annotate -a command.c
    and
    p4 annotate -a -c command.c

CHANGELIST MANAGEMENT

Your objectives for this exercise:

- To retrieve selected submitted changelists in order to report information about files in your depot
- To create a pending changelist so you can efficiently organize your work in your client workspace.
- Shelve your pending changelist and revert the files you had opened for edit.

1. What files were affected by changelist 388?
   p4 describe 388
   or
   p4 describe -s 388 (to suppress the diffs)

2. Sync your workspace to #head. cd to "C:\bruno_ws\Jam\MAIN\src" and open the file "command.h" for edit.
   p4 sync
   cd c:\bruno_ws\Jam\MAIN\src
   p4 edit command.h

3. Create a new changelist. Give it the description "Fixing header file bugs."
   p4 change
   Enter the description in the changelist form.

4. Open the file "compile.h" for edit within the new changelist you created.
   p4 edit -c <changelist#> compile.h

5. Open the file "hash.h" for edit in the default changelist. Reopen file "hash.h" within the changelist you just created.
   p4 edit hash.h
   p4 reopen -c <changelist#> hash.h

6. Make some simple changes to the files you've opened (or don't) and shelve the changelist.
   p4 shelve -c <changelist#>

7. Revert the opened files in the changelist you just shelved.
   p4 revert -c <changelist#> //...

8. Diff the file in your workspace against the content shelved in your workspace.
   p4 diff -f //...@=<changelist#>

HANDLING FILE CONFLICTS

Your objectives for this exercise:

- To create a situation where you are editing files that are out of date (as you would be when someone else edits the same file you are working on and submits their work before you)
- To resolve your files before you submit your work to your depot.
- To back out your change after you submit your completed work, because the project lead has rejected your work.

1. For the following exercises, start by syncing your "bruno_ws" client workspace with changelist 356. Some files will now be out of date.
   p4 sync @356

2. Open //depot/Jam/MAIN/src/RELNOTES and //depot/Jam/MAIN/src/patchlevel.h for editing. Ignore any warning messages.
   Edit RELNOTES and add some text at the beginning of the file (You can use the 'notepad' program to edit the files.)
   Edit patchlevel.h and add some text at the top. Also, to create a conflict in the file, change the first instance of "PATCHLEVEL 1.21" to "PATCHLEVEL 3.0".)
   cd c:\bruno_ws\Jam\MAIN\src
   p4 edit RELNOTES patchlevel.h
   Make the suggested changes in Notepad and save the files.

3. Confirm that the only files you have open are patchlevel.h and RELNOTES. If you have any others opened, revert them.
   p4 opened
   If any other files are opened:
   p4 revert <other files>

4. Try submitting your changelist.
   p4 submit

5. What files need to be resolved before you can submit your changelist?
   p4 resolve -n

6. What revisions of patchlevel.h and RELNOTES were you working on before you did your submit?
   p4 have patchlevel.h RELNOTES

7. What revisions of patchlevel.h and RELNOTES are now opened in your client workspace?
   p4 opened patchlevel.h RELNOTES

8. What can you do to prevent someone else from submitting new versions of patchlevel.h and RELNOTES before you submit yours?
   p4 lock patchlevel.h RELNOTES

9. How does your RELNOTES differ from the RELNOTES in the depot?
   p4 diff RELNOTES

10. Run p4 resolve to start up the interactive dialog. In patchlevel.h, what is your change? What is their change? What is the default merge resolution? Instead of selecting the default, select "edit" and add some more text. Then select the new default resolve action to resolve this file.
    'dy' shows your change, 'dt' shows their change.

The default merge resolution is 'am' (accept merged).
'e' allows you to edit the file in a text editor.
After the file has been edited, the default becomes 'ae'.

To accept the default, type 'ae' + 'Enter', or just press 'Enter'.

11. In RELNOTES, show their change and your change. Then, select "edit' and look for the diff conflict markers in the file. Remove the markers and save/exit the file. What is the default merge resolution?
    Resolve the merge by accepting the edited version.
    The default merge resolution after editing is 'ae'

    Diff conflict markers look like:
    >>>> ORIGINAL
    ==== THEIRS
    ==== YOURS
    <<<<

12. Submit your changelist.
    p4 opened (to see what the changelist number is)
    p4 submit -c 631 (your changelist number may be different)

13. What revision of RELNOTES has your added text in it?
    p4 annotate RELNOTES | more
    Find the revision number associated with your added text.

14. Uh-oh – the Jam project lead didn't like your change and wants you to back it out. What sequence of steps do you take to restore the depot files to the state they were in prior to your changelist? When you have completed your work, verify that your workspace files are identical to the previous versions of each file.
    Assuming your bad changelist was 631, first sync to depot state before your changelist:
    p4 sync ...@630

    p4 edit RELNOTES patchlevel.h
    p4 sync ...
    p4 resolve -ay
    p4 submit

    Verify that your workspace files are now identical to the revisions you synced in the first step.
    p4 diff -f RELNOTES@630
    p4 diff -f patchlevel.h@630
    (Your changelist number may be different than 630.)

CLIENT WORKSPACE MANAGEMENT, PART 1

Up until now you have been working on Perforce depot files using a client workspace spec created for you by your Perforce Administrator.

Your objectives for this exercise:

- To make changes to your client workspace spec.
- To create a new client workspace spec for a different project.

1. Update your workspace ('bruno_ws') to only map the files in '//depot/Jam/MAIN/src/' and all of its subdirectories.

   p4 client

   Edit the View field:
   View:
   //depot/Jam/MAIN/src/... //bruno_ws/Jam/MAIN/src/...

2. Change the 'Options' field "normdir" to "rmdir", so empty folders are deleted when sync'd. Set 'SubmitOptions' to 'revertunchanged' to automatically revert unchanged files on submit.

   p4 client

   Edit the Options field:
   Options: noallwrite noclobber nocompress unlocked nomodtime rmdir
   Edit the SubmitOptions field:
   SubmitOptions: revertunchanged

3. Sync your client workspace with the //depot/Jam/MAIN/src files.

   p4 sync

4. Where is the file //depot/Jam/MAIN/src/command.c mapped to?

   p4 where //depot/Jam/MAIN/src/command.c

5. Modify your workspace and change 'nomodtime' to 'modtime'. Test this by syncing '#0' of '//depot/Jam/MAIN/src/timestamp.c' and then syncing to the head revision.
   Hint: Use the MS-DOS "dir" command to view the timestamp on the file.

   p4 client
   Change the option nomodtime to modtime:
   Options: modtime
   Save your client workspace.

   p4 sync //depot/Jam/MAIN/src/timestamp.c#0
   p4 sync //depot/Jam/MAIN/src/timestamp.c
   dir
   Find timestamp.c and check the date listed for the file.

6. Create a new workspace named "bruno-projb" that maps "//depot/www/dev/" to a directory under "c:\bruno-projb". For this workspace, we won't need the folder "//depot/www/dev/images, so you'll need to exclude that from the client workspace view.

   p4 client -t bruno_ws bruno-projb
   Set the Root and View fields :
   Root: c:\bruno-projb
   View:
   //depot/www/dev/... //bruno-projb/www/dev/...
   -//depot/www/dev/images/... //bruno-projb/www/dev/images/...

CLIENT WORKSPACE MANAGEMENT, PART 2

Your objectives for this exercise:

- To set up the environment for your client workspaces.
- To find out information about other users' client workspaces.

1. Create a 'c:\bruno-projb' directory. Cd to that directory and create a file named 'p4config.txt' (without quotes) which sets your client workspace spec to 'bruno-projb'. Set the P4CONFIG environment/registry variable to point to files named 'p4config.txt'.

   mkdir c:\bruno-projb
   cd c:\bruno-projb
   notepad p4config

   In the "p4config.txt" file, type the one line:
   P4CLIENT=bruno-projb

   Save the file and on the command line type:
   p4 set P4CONFIG=p4config.txt

2. Test the configuration then sync files into your bruno-projb client workspace.

   p4 info or p4 set
   p4 sync

3. In the workspace 'bruno-projb' you will find the file 'index.html'. Where is this file in the depot?

   cd c:\bruno-projb\www\dev
   p4 where index.html

4. Change back to your bruno_ws client workspace directory, and check to make sure your Perforce configuration is set to the correct client workspace spec name.

   cd c:\bruno_ws
   p4 info* or *p4 set

5. There is a client workspace spec "raj-nt-spruce". What command would you use to show where the //depot/Jam/MAIN/src/jam.c file appears in the "raj-nt-spruce" client's workspace, without resetting your P4CLIENT environment variable?

   p4 -c raj-nt-spruce where //depot/Jam/MAIN/src/jam.c,
   or
   p4 -c raj-nt-spruce client

6. What files are out of date on client workspace raj-fir?

   p4 -c raj-fir sync -n

7. Change bruno's password to a strong one, "Brunopass," and login with the new password.

   p4 passwd
   Enter new password: Brunopass
   Re-enter new password: Brunopass
   Password updated.

   p4 login
   Enter password: Brunopass

BRANCHING & INTEGRATION, PART 1

Your objectives for this exercise:

- To create a release branch, since the development work on the 'Jam' project has reached a point where it is appropriate.
- To fix some bugs reported in your new branch.

Perforce works on files. If you are specifying a directory in the following exercises, be sure to add "/..." at the end of the file pattern to specify all the files in that directory and its subdirectories, e.g., //depot/Jam/MAIN/....

1. Make a branch specification that defines a branch view mapping the //depot/Jam/MAIN files to a new branch //depot/Jam/REL2.3. Call this branch specification "JamMAINtoREL2.3".
   p4 branch JamMAINtoREL2.3

   Add this line to the View field:
   View:
   //depot/Jam/MAIN/... //depot/Jam/REL2.3/...

2. Modify your client workspace spec view so that all //depot/Jam/MAIN/... and //depot/Jam/REL2.3/... files are mapped.
   p4 client
   The View field in your client workspace spec should have at least the following two lines:
   View:
   //depot/Jam/MAIN/... //bruno_ws/Jam/MAIN/...
   //depot/Jam/REL2.3/... //bruno_ws/Jam/REL2.3/...

3. Using **p4 integ** and the branch specification "JamMAINtoREL2.3", branch '//depot/Jam/MAIN/' to '//depot/Jam/REL2.3/'. Don't submit the changelist yet.
   p4 integ -b JamMAINtoREL2.3

4. Which files are opened in your client workspace?
   p4 opened | more

5. Submit your changelist.
   p4 submit

6. 'cd' to 'c:\bruno_ws\Jam\REL2.3\src\' directory. Open 'jam.c' and 'jam.h' for editing. Modify them and submit your changelist.
   cd c:\bruno_ws\Jam\REL2.3\src
   p4 edit jam.c jam.h
   Edit the files and save your changes.
   p4 submit

7. Show the revision history of //depot/Jam/REL2.3/src/jam.c.
   p4 filelog //depot/Jam/REL2.3/src/jam.c

8. Show the revision history of //depot/Jam/REL2.3/src/jam.c, including integration file history.
   p4 filelog -i //depot/Jam/REL2.3/src/jam.c

BRANCHING & INTEGRATION, PART 2

The following exercises will build on the branch created in the previous exercise. Having created a release branch, testing has revealed a number of bugs that need to be fixed. Normally, these fixes would happen in the release branch (where the problem was found) and integrated back to main, if appropriate.

Your objectives for this exercise:

- To propagate changes from a release branch back to main.
- To schedule content and action resolves as part of the change propagation.

1. In the release branch '//depot/Jam/REL2.3/src/', rename 'parse.c' and 'parse.h' to 'newparse.c' and 'newparse.h'. Submit your changes.
   p4 edit parse.*
   p4 move "parse.*" "newparse.*"
   or
   p4 move parse.c newparse.c
   p4 move parse.h newparse.h

2. Edit the files and save your changes.
   p4 submit

3. In 'c:\bruno_ws\Jam\REL2.3\src\', create a new file called 'release.log'. Edit the file, open it for add and submit your changes.
   cd c:\bruno_ws\Jam\REL2.3\src
   notepad release.log
   edit the file and save your changes
   p4 add release.log
   p4 submit

4. What changes in '//depot/Jam/REL2.3/' need to be integrated to '//depot/Jam/MAIN/'?
   p4 interchanges

5. Integrate from '//depot/Jam/REL2.3/' to '//depot/Jam/MAIN/' (using a branch or file spec), making sure to schedule a branch resolve as part of the integration.
   p4 integ –Rb -r –b JamMAINtoREL2.3
   or
   p4 integ –Rb //depot/Jam/REL2.3/... //depot/Jam/MAIN/...

6. What files were opened in your workspace?
   p4 opened

7. Resolve the integration in the following way:
   - If there are no conflicts, accept merged for all content resolves; otherwise resolve the file content interactively
   - Propagate the rename change to 'parse.c' and 'parse.h'
   - Ignore the branching of 'release.log.

   and submit your changes
   p4 resolve
   p4 submit

8. List all changelists that directly affect the files in '//depot/Jam/REL2.3/'.
   p4 changes //depot/Jam/REL2.3/…

9. List all changelists that affect the files in '//depot/Jam/REL2.3/, either directly or through integration. (Command line only.)
   p4 changes -i //depot/Jam/REL2.3/…

STREAMS

The following exercises will show how streams is an implementation of the 'Flow of Change'. During these exercises you will see how streams can help you manage the changes made to a codeline and its propagation to other codelines.

Your objectives for this exercise:

- Create a new development stream and populate it
- Work within this stream on a new feature, then switch to main to fix a bug
- Propagate changes between the streams.

1. List the current streams in '//jam/'
   p4 streams //jam/...

2. You are about to work on a new feature for the PB project, but you don't want to destabilize main with your changes. Create a new development stream called '//pb/dev.widgetX' that is a child of '//pb/main'.
   p4 stream -t development //pb/dev.widgetX
   or
   p4 stream -t development -P //pb/main //pb/dev.widgetX

3. Create a new workspace named "pb_ws" against the stream '//pb/dev.widgetX ', with a workspace root of 'c:\pb_ws'.
   Note: The view will be automatically generated for you, so any changes made to the workspace view will be discarded. As the next set of exercises use this workspace you may want to create a P4CONFIG file to store your settings.
   p4 client -S //pb/dev.widgetX pb_ws

4. Branch from '//pb/main' to '//pb/dev.widgetX' and submit your changes.
   p4 integ –S //pb/dev.widgetX –r
   or
   p4 merge –S //pb/dev.widgetX –r
   p4 submit

5. Sync your workspace to ensure that it's up-to-date
   p4 sync

6. Edit '//pb/dev.widgetX/src/parse.c' and '//pb/dev.widgetX/src/parse.h' and submit your changes
   p4 edit //pb/dev.widgetX/src/parse.*
   Edit the files
   p4 submit

7. A critical bug has been found in the mainline, that requires your immediate attention. Switch your current workspace and re-sync, so that it is now working against '//pb/main'. Edit '//pb/main/src/expand.c' to fix the problem and submit your change.
   p4 client -s -S //pb/main
   p4 sync //...
   p4 edit //pb/main/src/expand.c
   Edit the file
   p4 submit

8. Having fixed the bug, you want to copy your completed feature from your dev stream (//pb/dev.widgetX) to '//pb/main'. What happens if you attempt to copy your changes up?

   p4 copy -S //pb/dev.widgetX
   Stream //pb/dev.widgetX cannot 'copy' over outstanding 'merge' changes.

9. Looks like you need to merge down first, before you can copy your new feature up. Switch your client workspace to '//pb/dev.widgetX' and sync. Then merge the changes from '//pb/main' down and submit.

   p4 client -s -S //pb/dev.widgetX
   p4 sync
   p4 merge -S //pb/dev.widgetX –r
   p4 resolve –as
   p4 submit

10. Now that you've merged the changes from '//pb/main' switch your workspace back to '//pb/main' and copy up your new feature.

    p4 client -s -S //pb/main
    p4 sync //...
    p4 copy -S //pb/dev.widgetX
    p4 submit

LABELS

Your objectives for this exercise:

- Tag files using two different methods:
  a. Tag files to a label without syncing to a workspace
  b. Tag files to a label based on the revisions currently sync'd to a workspace.

1. Tag files under the //depot/Jam/MAIN/src directory with a label named "jam-NT-build" that were current as of changelist 392, regardless of which versions of the files are in your client workspace.
   p4 tag -l jam-NT-build //depot/Jam/MAIN/src/...@392

2. List the versions of the files you have just tagged with label jam-NT-build.
   p4 files @jam-NT-build

3. Sync your bruno_ws client workspace with the file versions that were current when changelist 395 was submitted.
   p4 sync @395

4. Create a label specification called jam-2.2.1. Restrict the label's view to //depot/Jam/MAIN/src files.
   p4 label jam-2.2.1
   Edit the View field:
   View:
   //depot/Jam/MAIN/src/...

5. Tag the files in your client workspace with the jam-2.2.1 label.
   p4 labelsync -l jam-2.2.1

6. What version of //depot/Jam/MAIN/src/jam.h is tagged with the "jam-2.2.1" label?
   p4 files //depot/Jam/MAIN/src/jam.h@jam-2.2.1

7. Modify the label 'jam-2.2.1' and exclude all '*.c' and '*.h' files from the labels view. Update the set files tagged by the label to reflect the change in the label specs view.
   p4 label jam-2.2.1
   Add these lines to the label's View:
   -//depot/Jam/MAIN/src/....c
   -//depot/Jam/MAIN/src/....h

   p4 labelsync -l jam-2.2.1 ***or***
   p4 labelsync -d -l jam-2.2.1 //depot/Jam/MAIN/src/....c //depot/Jam/MAIN/src/....h
   or
   p4 tag -d -l jam-2.2.1 //depot/Jam/MAIN/src/....c //depot/Jam/MAIN/src/....h

8. Sync your client workspace with the file versions in the "jam-2.2.1" label.
   p4 sync @jam-2.2.1

9. List all the labels that contain files tagged under the //depot/Jam/MAIN path.
   p4 labels //depot/Jam/MAIN/...

JOB TRACKING

Your objectives for this exercise:

- Report a bug you have discovered for a Jam port, by:
- Checking if a job already exists for your bug, before entering a new job
- Create a new job to report your bug.
- Link completed work to an existing job.

1. List the 10 most recently-entered jobs in your server.
   p4 jobs -r -m10

2. List the jobs which contain the words “Jam” and “port”.
   p4 jobs -e “Jam port”

3. Which changelists are fixes for job “job000004”?
   p4 fixes -j job000004

4. Did any fixes for job000004 involve *.h files?
   p4 fixes -j job000004 //depot/….h

5. Create a new job. Enter any description, and leave “open” as its status.
   p4 job

6. Create a new numbered changelist and include your job in it.
   p4 change
   Add a Jobs field:
   Jobs: n
   where ‘n’ is the name of your newly-created job.

7. Which job does this changelist fix?
   p4 fixes -c <changelist #>
   where <changelist #> is the number of your newly-created changelist.

8. Update your new job and modify its status field to suspended.
   p4 job n
   where n is the name of your newly-created job

9. Link changelist 394 to job “job000002.”
   p4 fix -c 394 job000002

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

