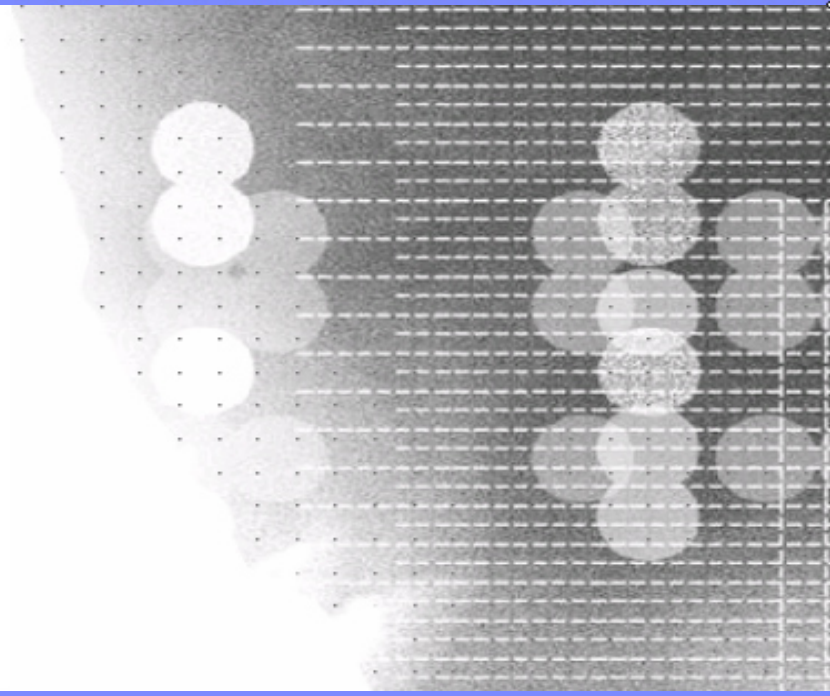
TPF Users Group Fall 2007

Main Title: TPF Lab Process

Sub-title: Code Flow and Versioning

Name: Brian Laferriere

Venue: Hot Topics



AIM Enterprise Platform Software

IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

z/TPF SCM - Source File Names

- Source files are named with a relative pathname, not an absolute pathname. For example:
  - Relative: base/cp/ccnucl.asm
  - Absolute: /ztpf/cur/base/cp/ccnucl.asm
- Source file names do not include a version code in the pathname or base name.

z/TPF SCM

- All source files are managed in a single release for the life of the product.
- Each change (fix/project) is contained in a unique branch, based off the main trunk.
- Each branch is named according to the associated APAR number.
- Only source files (.c, .asm, .mak, etc.) are managed in the SCM -- not listings or binaries.

z/TPF SCM

- Changes to the same file can occur in more than one branch at the same time.
- All branches must stem from the main trunk.
- Close order defines merge order; SCM identifies collisions.
- All branches are merged back into the main trunk at completion.
- There is no fall-back! To undo a change, a new branch must be opened.

z/TPF SCM

- Two baselines are maintained:
  - CURRENT:
    - Moves with each APAR.
    - Includes all closed APARs.
  - COMMIT:
    - Moves once a month.
    - Includes all closed APARs at that time.

z/TPF SCM

- Developers identify what code to change and control the content of the branch.
- Developers cannot affect the baselines; the build team joins the branch back into the main trunk as part of the APAR release process.

z/TPF Directories on Linux

- One directory for each SCM baseline:
  - /ztpf/cur
    - Updated with each closed APAR.
  - /ztpf/commit:
    - Updated once a month.
    - Includes all closed APARs at that time.
  - Both include all binaries, listings, etc. for the built programs.

z/TPF Systems

- Three configurations: BAS, BSS, WP
- One development system for each configuration and baseline:
  - CUR (Images: tpf01, ztpfa/ztpfb)
    - Each APAR is loaded as it closes.
  - COMMIT (Images: tpf01, ztpfa/ztpfb)
    - Full build/load once a month with all closed APARs at that time.
    - Includes images for previous 2 PUTs (PUTn-1, PUTn-2).
- One system verification test system (SVT) for each configuration:
  - SVT (Images: tpf01, zsvtc, zsvtp)
    - COMMIT build also loaded here once a month.
    - SVT project build loaded weekly to zsvtc.
    - Image from previous week moved weekly to zsvtp.

Fix Process - Developer

1. An APAR is received via RETAIN. For example: PJ32332
2. A branch is created in the SCM, named with the APAR number.
3. The affected files are checked-out through the branch, from the CURRENT baseline, to the developer's working directory:
   /home/user/PJ32232
4. The developer optionally sets internal and/or external version codes in maketpf.cfg:
   TPFOBJPP_COMMENT := PJ32232
   USER_VERSION_CODE := BL or VERSION_CFTP := BL
5. The files are updated and then built against the CURRENT directory:
   TPF_ROOT := /home/user/PJ32232
   TPF_ROOT += /ztpf/cur
   Only the affected files are compiled; additional objects for unchanged files are picked up from the CURRENT directory for the links.
6. The programs are loaded/tested using VPARS on the CURRENT image.
7. A check is made for any baseline changes of checked-out files, resulting from APARs closed since the original checkout; merges are performed and steps 4-6 are repeated as necessary.
8. The APAR is reviewed and then added to the APAR to the build queue.

Fix Process – Build Team

1. The next APAR on the build queue is selected.
2. A check is made for any baseline changes of files in the APAR, resulting from APARs closed since the APAR was added to the build queue. If changes are found, the APAR is returned to the developer to resolve.
3. The APAR files are extracted to a build directory, named with the APAR number:
   /ztpf/aparbld/PJ32232
4. The internal build version is set to the APAR number in maketpf.cfg:
   TPFOBJPP_COMMENT := PJ32232
   The files are built against the CURRENT directory:
   TPF_ROOT := /ztpf/aparbld/PJ32232
   TPF_ROOT += /ztpf/cur
   Only the affected files are compiled; additional objects for unchanged files are picked up from the CURRENT directory for the links.
5. The programs are loaded on CURRENT image.
6. The APAR branch is merged back into the main trunk, setting a new CURRENT baseline.
7. The files extracted to and generated in the build (/ztpf/aparbld/PJ32232) directory are copied into the /ztpf/cur directory.

New Project Process - Developer

1. A branch is created in the SCM, named for the project, for example: curl.
2. The affected files are checked-out/created through the branch, from the COMMIT baseline, to the developer's working directory:
   /home/user/newprj
3. The developer optionally sets internal and/or external version codes in maketpf.cfg:
   TPFOBJPP_COMMENT := newprj
   USER_VERSION_CODE := BL or VERSION_CFTP := BL
4. The files are updated and then built against the COMMIT directory:
   TPF_ROOT := /home/user/newprj
   TPF_ROOT += /ztpf/commit
   Only the affected files are compiled; additional objects for unchanged files are picked up from the COMMIT directory for the links.
5. The programs are loaded/tested using VPARS on the COMMIT image.
6. With each monthly COMMIT baseline change, a check is made for any baseline changes of checked-out files (collisions), resulting from APARs closed since the original checkout; merges are performed and steps 3-5 are repeated as necessary.
7. When the project development is complete, an internal APAR is created in RETAIN and the project is renamed using the APAR number.
8. The APAR is reviewed and then added to the APAR to the build queue.

Monthly Baseline – Build Team

1. APAR processing for the CURRENT baseline is temporarily suspended.
2. The files changed since the last COMMIT baseline are extracted into a build directory:
   /ztpf/bld
3. The internal build version is set to the monthly build iteration (or the PUT level if the final build for the PUT) in maketpf.cfg:
   TPFOBJPP_COMMENT := PUT3
4. The files are built against the COMMIT directory:
   TPF_ROOT := /ztpf/bld
   TPF_ROOT += /ztpf/commit
   A full force build is performed.
5. The programs are loaded on the COMMIT image.
6. The new baseline is set in the SCM; open projects must check for collisions.
7. The files in the build (/ztpf/bld) directory are copied into the /ztpf/commit directory.
8. If the final build for a PUT, the files in the build directory are also copied into the /ztpf/cur directory.

Weekly System Test – Build Team

1. Projects to be included are given a close order by the build team.
2. The build team identifies conflicts between projects and the project teams must resolve -- outside the scope of the SCM.
3. The SVT build from the previous week is moved from “current” to “previous”:
   1. The directories is moved: /ztpf/svtcur -> /ztpf/svtprev
   2. The system images are moved: zsvtc -> zsvtp
4. The projects are extracted, in close order, into a the SVT current build directory:
   /ztpf/svtcur
5. The internal build version is set to an SVT label:
   TPFOBJPP_COMMENT := SVT1008
6. The files are built against COMMIT directory:
   TPF_ROOT := /ztpf/svtcur
   TPF_ROOT += /ztpf/commit
   Only the affected source is compiled and linked.
7. The programs are loaded on the zsvtc image.
8. The SCM is not affected.

Note: an independent full SVT build is run weekly, to check for compile/link errors outside the scope of the affected source. That build is not loaded.

Internal Versioning

1. The TPFOBJPP_COMMENT inserted in every object gives the last compiled version.
2. A maketpf object automatically inserted into every linked shared object giving last linked version.
3. Versions can be queried on line with ZDMAP.
4. Versions can be queried on linux with tpfzdmap (utility available for download).

Versioning Example

PJ32232

- Updated source: base/rt/cftp1.c
- Compiled object: base/obj/cftp1.o
- Linked shared object: base/load/CFTP.so
- CFTP contains:
  - base/rt/cftp1.c
  - base/rt/cftp3.c
  - base/rt/cglobb.c
  - base/rt/cftp2.y

ZDMAP cftp i-lpath

```
zdmap cftp i-lpath
CSMP0097I 07.31.16 CPU-B SS-BSS  SSU-HPN  IS-01
DMAP0003I 07.31.16 LINK MAP DATA DISPLAY
                   CFTP   ACTIVE IN LOADSET BASE IN SUBSYSTEM BSS
                   PROGRAM ADDRESS                          0000000390E02D60
                   PROGRAM SIZE                                     0000A45C

/ztpf/aparbld/PJ32232/base/obj/cftp1.o      - OBJ FILE AT ADDR 0000000390E02D60
  OBJECT FILE SIZE                                            000058A4
  COMPILER INFO -  GCC: (GNU) 4.1.2 20060724 (prerelease) (GNUPro 06r1)  _
  COMPILED ON 2007/09/19 AT 10.42.31
  COMMENT - PJ32232

/ztpf/bld/base/obj/cftp3.o                  - OBJ FILE AT ADDR 0000000390E08604
  OBJECT FILE SIZE                                            00000610
  COMPILER INFO -  GCC: (GNU) 4.1.2 20060724 (prerelease) (GNUPro 06r1)
  COMPILED ON 2007/02/05 AT 13.26.15 _
  COMMENT – PUT3

  NAME - /ztpf/aparbld/PJ32232/base/obj/bld-PJ32232.o _
  OBJECT FILE SIZE                                            00000000
  COMPILED ON 2007/09/19 AT 10.42.42
  COMMENT - PJ32232
```

tpfzdmap base/obj/cftp1.o

```
Hex dump of section '.tpfzdmap':
  0x00000000 00000000 0000144f 00000002 00000001 .......|........
  0x00000010 0007d7d1 f3f2f2f3 f2000200 2661a9a3 ..PJ32232..../zt
  0x00000020 97866181 97819982 938461d7 d1f3f2f2 pf/aparbld/PJ322
  0x00000030 f3f26182 81a28561 96829161 8386a397 32/base/obj/cftp
  0x00000040 f14b9600 03003640 c7c3c37a 404dc7d5 1.o.... GCC: (GN
```

tpfzdmap base/load/CFTP.so

```
Hex dump of section '.tpfzdmap':
  0x00000010 0007d7d1 f3f2f2f3 f2000200 2661a9a3 ..PJ32232..../zt    < cftp1.o … last compiled at PJ32232
  0x00000020 97866181 97819982 938461d7 d1f3f2f2 pf/aparbld/PJ322
  0x00000030 f3f26182 81a28561 96829161 8386a397 32/base/obj/cftp
  0x00000040 f14b9600 03003640 c7c3c37a 404dc7d5 1.o.... GCC: (GN

  0x00001460 06d7e4e3 f44bf300 02001a61 a9a39786 .PUT3....../ztpf    < cftp3.o … last compiled at PUT 3
  0x00001470 61829384 618281a2 85619682 91618386 /bld/base/obj/cf
  0x00001480 a397f34b 96000300 3640c7c3 c37a404d tp3.o.... GCC: (

  0x00001640 000e8200 00000200 00000100 06d7e4e3 ..b.........PUT    < cglobb.o … last compiled at PUT 3
  0x00001650 f44bf300 02001b61 a9a39786 61829384 3....../ztpf/bld
  0x00001660 618281a2 85619682 91618387 93968282 /base/obj/cglobb
  0x00001670 4b960003 003640c7 c3c37a40 4dc7d5e4 .o.... GCC: (GNU

  0x000024c0 00000001 3b000000 02000000 010006d7 ............P    < crt0.o … last compiled at PUT 3
  0x000024d0 e4e3f44b f3000200 1961a9a3 97866182 UT3....../ztpf/b    < automatically added because CFTP is a main
  0x000024e0 93846182 81a28561 96829161 8399a3f0 ld/base/obj/crt0
  0x000024f0 4b960003 003640c7 c3c37a40 4dc7d5e4 .o.... GCC: (GNU

  0x00002600 00000002 00000001 0006d7e4 e3f44bf3 .........PUT3..    < cftp2.o … last compiled at PUT 3
  0x00002610 0002001a 61a9a397 86618293 84618281 ..../ztpf/bld/ba
  0x00002620 a2856196 82916183 86a397f2 4b960003 se/obj/cftp2.o..

  0x000029c0 00000001 0007d7d1 f3f2f2f3 f2000200 ......PJ32232...    < build object, bld-PJ32232
  0x000029d0 2c61a9a3 97866181 97819982 938461d7 ./ztpf/aparbld/P    < automatically added, gives link date/time
  0x000029e0 d1f3f2f2 f3f26182 81a28561 96829161 J32232/base/obj/
  0x000029f0 82938460 d7d1f3f2 f2f3f24b 96000400 bld-PJ32232.o...
  0x00002a00 0ef2f0f0 f7f0f9f1 f9f1f0f4 f2f4f200 .20070919104242.
```

Notes

- Pathnames in zdmap output reflect the build directory. This may not be the same as the promote/shadow directory. For example, PJ32232 was compiled and linked in /ztpf/aparbld, but when closed, it is promoted to /ztpf/cur. Therefore, the cftp1.o file will always show /ztpf/aparbld/PJ32232 as the directory name; it will not change to /ztpf/cur when the promotion occurs.
- When monthly COMMIT builds occur, all objects lose the last APAR comment and directory info. It is replaced by the PUT x.y comment and the /ztpf/bld pathname.
- Objects changed during any given PUT keep their APAR-level version in the comment, in the /ztpf/cur directory, for the duration of the PUT. At the end of the PUT, all object comments are replaced by the “PUT n” version.

Discussion

- Is a convention needed for defining a version string in the COMMENT data, for example:
  TPFVER=PJ32232
- Does the source file name need to be included in the COMMENT data?
  TPFSRC=/ztpf/aparbld/PJ32232/base/rt/cftp1.c
- Where does the COMMENT information need to be displayed?
  - ZDPGM?
  - Dumps?
- How does the COMMENT information need to be accessible?
  - Programmatically via APIs? (GETCIC can be used)
  - On Linux through scripts?
    - Does tpfzdmap output need to be modified?
- Other?

Trademarks

- IBM is a trademarks of International Business Machines Corporation in the United States, other countries, or both.
- Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.

Notes

- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
- This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
- All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
- Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
- Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
- This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.