Capturing the POSIX File System

by René Bruijns - Atraxis AG

Since I think it's not fair to always rely on Mr. Taylor (my former boss) for contributions from the old continent, I've decided to write a little article about a capture system for the POSIX file system which I'm developing at Atraxis.

POSIX Capture

Quite recently a TO2 Recoup timeout on one of our test systems caused all the Collection Support pool files to be released. Although the timeout problem has been fixed by letting Recoup abort, I realised that we needed to have some sort of capture for TPFCS and the POSIX file system.

According to the Database Reference manual it is left to the user to correct data corruption in TPFCS, so I made an attempt to capture files of the POSIX file system. As each POSIX file is actually a BLOB collection embedded in an inode fixed file, it's possible to use the TO2_capture() and TO2_restore() functions, which are part of TPFCS.

Since these functions require other TPFCS functions to be called - to set up an archive interface and an environment block -, the best thing was to 'cut and paste' these from Bookmanager (and change the variable names a little so it looks like I made it all up myself) and use the Visual Age Debugger to see what comes out of it.

I created one segment that processes the following entries:

```
>>— ZUPSX CAPTURE———————————————————————— <<
```

```
>>— ZUPSX RESTORE—— i ———————————————————— <<
                    + path +     + -n +
```

i
  inode file ordinal

path
  path name of file or directory

-n
  number of captures backwards

ZUPSX CAPTURE is processed by function CaptureAllFiles(), which loops through the inode fixed file range to get to the Persistent Identifier (PID) of each embedded collection.

This PID is input to TO2_capture(), but to be able to do a TO2_restore() more than once for the same POSIX file, it is copied to a separate (unode) fixed file whichs corresponds to the inode fixed file.

TO2_capture() will request an archive tape to be mounted and returns a pointer to a buffer, that contains 11 location structures with data of where the collection is stored (volume serial number, offset on the tape, etc).

This location data is also saved on the unode fixed file. The unode fixed file contains 7 * 11 location structures and 7 capture PID structures to allow for data from 7 captures.

POSIX Restore
ZUPSX RESTORE is processed by function RestoreFile().

Input to RestoreFile() is the inode fixed file ordinal or the full path name, which is converted to the ordinal by function lstat().

RestoreFile() opens the inode file and the unode fixed file - which contains the location data - and calls TO2_restore() that will request the relevant tape to be mounted.

The new PID that is returned by TO2_restore() is stored in the inode fixed file and the old collection is deleted.

Any internal error can be retrieved by TO2_getErrorCode() and the error message can be displayed using TO2_getErrorText().

The new PID will automatically be picked up by Recoup 'FC2A' processing.

Examples:

```
CSMP0099I 14.37.12 010000-A ZUPSX CAPTURE
COSK0079A 14.37.13 *CP* BSS0   MOUNT ARA TAPE FOR OUTPUT
CSMP0099I 14.37.34 010000-A ZTMNT ARA 479
COTM0310I 14.37.34 TMNT BSS0   TAPE ARA MOUNTED ON DEVICE 479
              VSN A00033 G0010 S0001 F38K2    SL  NOBLK  COMP
ZPSX0098I 14.37.50 TPF FILES SUCCESSFULLY CAPTURED

CSMP0099I 14.38.27 010000-A ZUPSX RESTORE 18
TPXD0001A 14.38.27 LOAD TAPE VOLUME A00033 INTO DRIVE 0479 MOUNTED AS ARB
COSK0079A 14.38.27 *CP* BSS0   MOUNT ARB TAPE FOR INPUT
CSMP0099I 14.40.48 010000-A ZTMNT ARB 479 AI
COTM0310I 14.40.48 TMNT BSS0   TAPE ARB MOUNTED ON DEVICE 479
              VSN A00033 G0011 S0001 F38K2    SL  NOBLK
ZPSX0099I 14.40.49 TPF FILE SUCCESSFULLY RESTORED

CSMP0099I 14.45.31 010000-A ZUPSX RESTORE 1000
ZPSX0009E 14.45.31 TPF FILE SYSTEM RESTORE - NO FILE
ZPSX0013E 14.45.31 TPF FILE SYSTEM RESTORE ABENDED
```