IBM Education Assistance for z/OS V2R2

Item: CSV EXITS
Element/Component: BCP Contents Supervisor (CSV)

Material current as of May 2015

Agenda

Trademarks

Presentation Objectives

- Understand the CSVFETCH exit

Overview

- Problem Statement / Need Addressed
  - Monitoring programs want to be able to see what is being fetched.
- Solution
  - Provide the CSVFETCH (dynamic) exit
- Benefit / Value
  - Less front-ending of z/OS services

Usage & Invocation – CSVFETCH exit

- You may add your exit routine to the CSVFETCH exit via normal dynamic exits functions (SET PROG command with PROGxx parmlib member, SETPROG command, CSVDYNEX macro)
- You may indicate for which event types you want the exit routine to get control

Usage & Invocation – CSVFETCH exit points

- Fetch with Get Storage
- Fetch, found on Job Pack Queue
- z/OS Unix Fetch, with Get Storage
- z/OS Unix Fetch, found on JPQ
- Fetch from LPA
- Fetch, load with addr (directed load)
- z/OS Unix Fetch, directed load
- Fork

Usage & Invocation – CSVFETCH exit points

- Unfetch, and free storage
- Unfetch, and no freeing of storage (users remain)
- Unfetch from LPA

Usage & Invocation – CSVFETCH exit points

- Fetch event exit calls are done after the module has been fetched
- Unfetch event exit calls are done before the module is truly unfetched (before module storage has been freed)

Usage & Invocation – CSVFTCHX macro

- Maps the data provided to the exit routine

```
DCL 1 FTCHX Type

      ,3 FTCHX_ServiceID Char(8)      /* ServiceID provided by exit caller
                                         See equates beginning
                                         FTCHX_ServiceID_                 */

      ,3 FTCHX_EpName Char(8)         /* The entry point name when not a

                                         path name. Otherwise *PATHNAM    */

      ,3 FTCHX_EpAddr64 Ptr(64)       /* The 64-bit entry-point address   */

      ,3 FTCHX_CdeAddr Ptr(31)        /* From CDE can locate major CDE and
                                         from major CDE can locate XTLST  */

      ,3 FTCHX_Flags Bit(32)

        ,5 FTCHX_UnFetch Bit(1)       /* Off for "fetch", on for "unfetch" */

        ,5 FTCHX_ByPathName Bit(1)    /* When fetch by path name          */

        ,5 FTCHX_ByDCB Bit(1)         /* When fetch with DCB              */
```

Usage & Invocation – CSVFTCHX macro

```
  ,5 FTCHX_GlobalNotFixed Bit(1) /* When LOAD GLOBAL=YES             */
   ,5 FTCHX_GlobalFixed Bit(1)    /* When LOAD GLOBAL=(YES,FIXED)      */
   ,5 FTCHX_LoadWithAddr Bit(1)   /* When LOAD with ADDR or ADDR64     */
 ,3 FTCHX_PathnameAddr Ptr(31)    /* When FTCHX_ByPathName             */
 ,3 FTCHX_UCBADDR PTR(31)         /* Address of UCB associated with DS.
                                     0 if no CDX                       */
 ,3 FTCHX_CCHH CHAR(4)            /* CCHH of DS on volume. 0 if no CDX */
 ,3 FTCHX_DCBADDR PTR(31)         /* When FTCHX_ByDCB                  */
 ,3 FTCHX_XTLST64 CHAR(264)       /* For fetch (not unfetch) event,
                                     8-byte header (bytes 4-7 indicate the
                                     number of extents that follow), 1-16
                                     16-byte extents each of which has
                                     8-byte address and 8-byte length */
```

Usage & Invocation – CSVFTCHX macro

```
Dcl FTCHX_ServiceID_Fetch_GetStore Char(8)
         Constant('0000000100000000'x);  /* For this fetch, the
                      module was not on the JPQ so a new copy
                      was gotten. If this is an alias, the
                      storage is associated with the major name   */
Dcl FTCHX_ServiceID_Fetch_JPQ Char(8)
         Constant('0000000200000000'x);  /* For this fetch, the
                      module was already on the JPQ and the
                      existing copy was used. If this is an alias,
                      the storage is associated with the major
                      name.                                       */
```

Usage & Invocation – CSVFTCHX macro

```
Dcl FTCHX_ServiceID_Unix_GetStore Char(8)
          Constant('0000000400000000'x);  /* For this fetch, the
                        module was not on the JPQ so a new copy
                        was gotten. If this is an alias, the
                        storage is associated with the major name   */
Dcl FTCHX_ServiceID_Unix_JPQ Char(8)
          Constant('0000000800000000'x);  /* For this fetch, the
                        module was already on the JPQ and the
                        existing copy was used. If this is an alias,
                        the storage is associated with the major
                        name.                                       */
```

Usage & Invocation – CSVFTCHX macro

```
Dcl FTCHX_ServiceID_Fetch_LPA Char(8)      Constant('00000010 00000000'x);
Dcl FTCHX_ServiceID_Fetch_Dirload Char(8)
          Constant('0000000200000000'x);  /* For this fetch, directed
                          load (LOAD with ADDR or ADDR64) was used.
                          The requestor provided the storage.         */
Dcl FTCHX_ServiceID_Unix_Dirload Char(8)
          Constant('0000000400000000'x);  /* For this UNIX fetch,
                          directed load was used.
                          The requestor provided the storage.         */
Dcl FTCHX_ServiceID_Fork Char(8)           Constant('0000008000000000'x);
```

Usage & Invocation – CSVFTCHX macro

```
Dcl FTCHX_ServiceID_UnFetch_FreeStore Char(8)
          Constant('0000000000000001'x);  /* For this unfetch, there
                        are no remaining users, so the module
                        storage is freed.                          */
Dcl FTCHX_ServiceID_UnFetch_NoFree Char(8)
          Constant('0000000000000002'x);  /* For this unfetch, there
                        are remaining users, so the module storage
                        is not freed.                              */
Dcl FTCHX_ServiceID_UnFetch_LPA Char(8)    Constant('00000000 00000010'x);
```

Usage & Invocation – CSVFETCH exit environment

- Key 0, supervisor state
- Task mode
- Primary ASC mode
- AMODE 31
- Local lock held (the exit routine must **not** release the local lock)
- Primary = Home = Secondary
- Enabled for I/O and External Interrupts

Usage & Invocation – SETPROG and PROGxx

- Exit routine may use ServiceMask to identify for which events it is to get control
- EXIT ADD SERVICEMASK=sm
  - Analog of CSVDYNEX
  - "sm" is "x1" or "(x1)" or "(x1,x2)"
  - "x1" represents bytes 0-3 of the mask
  - "x2" is bytes 4-7
  - Specified in hex. E.g., SERVICEMASK=7F would produce 0000007F_00000000

Presentation Summary

- CSVFETCH exit is provided

Appendix

- Publications:
  - Authorized Assembler Services Reference
  - System Commands
  - Init & Tuning Reference