Transaction Processing Facility

Compiler Roll-out Plan
and
zOS 1.2 Changes

Colette A. Manoni

Compilers and System Levels

A new system level does not imply a new compiler

- OS/390 2.10 compiler is shipped with the following systems.
  - OS/390 V2R10
  - zOS 1.1
  - zOS 1.2 thru zOS 1.5

- zOS 1.2 compiler is shipped with the following systems.
  - zOS 1.2
  - zOS 1.3
  - zOS 1.4

TARGET Option with zOS Compiler

- The TARGET option allows you to specify the run-time environment for your program's object module.

- TPF 4.1 supports a run-time environment equivalent to OS/390 V2R10

  - TARGET(OSV2R10) must be specified when using the zOS 1.x compilers

Compiler Roll-out - Available

	TPF **PUT Level**	**MVS** **System Level**	**Compiler Level**
June 2002	PUT 16	zOS 1.2	◆ OSV2R10 (shipped w/ zOS 1.2) * ◆ zOS 1.1 ◆ OS/390 2.10
June 2003	PUT 17	zOS 1.2	◆ OSV2R10 (shipped w/ zOS 1.2) * ◆ zOS 1.1 ◆ OS/390 2.10
June 2004	PUT 18	zOS 1.3	◆ zOS 1.2 with Target(OSV2R10)* ◆ OSV2R10 (shipped w/ zOS 1.4) ◆ zOS 1.1 ◆ OS/390 2.10

If you use the zOS compiler

- Apply all compiler maintenance
  - Current Maintenance level (2/28/04)
    - UQ83267 UQ81416
  - APAR PQ72662
    - Erroneously exports functions __ReThrow and __Throw
- Apply TPF APARs
  - PJ29024
  - PJ29716
- Investigate Compile Options

Required Compiler Options

- LANGLVL(NOANSIFOR,DBCS,LIBEXT,OLDTEMPACC)
- LANGLVL(CS,LONGLONG,ANONSTRUCT)
- TARGET(LE,OSV2R10)
- KEYWORD(export)
- TMPLPARSE(WARNING)

zOS 1.2

- z/OS V1R2 C/C++ includes a C++ compiler which is fully compliant with the ISO C++ 1998 Standard.
  - Will require source code changes

- z/OS V1R2 C is a straight-forward follow-on to OS/390 V2R10
  - Most code compiles cleanly without changes

Common Problems - Anonymous Unions

CCN6608W An anonymous union should only define non-static data members.

```
        union {                                             /*PJ27393*/

              TPF_FA8         fa;      /* 8-byte file address         */

              struct {                                       /*PJ27393*/
CCN6608W An anonymous union should only define non-static data members.
                   unsigned char __rsv2[4];                  /*PJ27393*/
                   unsigned char __mchr[4];                  /*PJ27393*/
                } __fa_mchr;                                 /*PJ27393*/

           #ifdef __cplusplus                                /*PJ29160*/
              };                                             /*PJ29160*/
           #else                                             /*PJ29160*/
              }union_declarator;                             /*PJ29160*/
           #endif                                            /*PJ29160*/
```

Common Problems - Anonymous Unions

Corrected Code

```
union {                                             /*PJ27393*/

      TPF_FA8         fa;      /* 8-byte file address        */

      struct {                                       /*PJ27393*/
           unsigned char __rsv2[4];                  /*PJ27393*/
           unsigned char __mchr[4];                  /*PJ27393*/
        } __fa_mchr;                                 /*PJ27393*/

   ~~#ifdef __cplusplus~~                                /*PJ29160*/
      ~~};~~                                             /*PJ29160*/
   ~~#else~~                                             /*PJ29160*/
      }union_declarator;                             /*PJ29160*/
   ~~#endif~~                                            /*PJ29160*/
```

Common Problems - Member Name Same as Structure

CCN6125W The data member "char flowID" cannot have the same name as its containing class.

```
struct flowID                                                /*PJ26161*/
{  /* not currently in use */                                /*PJ26161*/
     struct mscb             *session;                       /*PJ26161*/
     struct flowID           *next;                          /*PJ26161*/
     char                     flowID;                        /*PJ26161*/
CCN6125W The data member "char flowID" cannot have the same name as its
containing class.
     unsigned int             reservedIBM : 24;              /*PJ26161*/
};
```

Common Problems - Member Name Same as Structure

Corrected Code

```
struct flowID                                                /*PJ26161*/
{  /* not currently in use */                                /*PJ26161*/
     struct mscb             *session;                       /*PJ26161*/
     struct flowID           *next;                          /*PJ26161*/
     char                     flowid;                        /*PJ26161*/
     unsigned int             reservedIBM : 24;              /*PJ26161*/
};
```

Common Problems - #pragma map

CCN6410W The function "__CREEC" specified in "#pragma map" is an overloaded function. The pragma is ignored.

- This message only appears on the console
- Is not displayed in the listing file

```
#pragma map(__CREEC, "@@CREEC")

void __CREEC(int length, const void *parm, const char *segname,  \
             enum t_lvl level, int priority);
void __CREEC(int length, void *parm, const char *segname,  \
             TPF_DECB *decb, int priority);
```

Common Problems - #pragma map

Corrected code:

```
#ifdef __cplusplus
    #pragma map(__CREEC(int, const void *, const char *,         \
                        enum t_lvl, int),"@@CREEC")
#else
     #pragma map(__CREEC,"@@CREEC")
#endif

void __CREEC(int length, const void *parm, const char *segname,  \
              enum t_lvl level, int priority);
 void __CREEC(int length, void *parm, const char *segname,  \
              TPF_DECB *decb, int priority);
```

Common Problems - Type Checking

CCN5256 (S) A parameter of type "void *" cannot be initialized with an expression of type "const char *".

```
const char *msg = "This is a constant string";
void  some_function(void* parm);

void foo (void) {
     ...
     some_function(msg);
     ..
}
```

Common Problems - Type Checking

Corrected Code

```
~~const~~ char *msg = "This is a constant string";
void  some_function(void* parm);

void foo (void) {
     ...
some_function(msg);
..
}
```

Other Common Problems

- Uninitialized variables
  - CCN6102W "x" might be used before it is set.

```
int  fun3(void)
{
  int x;
  ....
  return(x);
}
```

- Comments in comments.
  - CCN5804W The characters "/*" are detected in a comment.

```
 /*
    If you set up block comments
     //* an imbedded comment causes a warning
  */
```

Legal Disclaimer

Any references to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

IBM, z/OS, and OS/390 are trademarks of the IBM Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.