TPF Users Group
Grapevine Texas

IBM Software Group

What You Can Do Today To Get Ready For Tomorrow

Assembler Programs

Application Subcommittee

Sue Zee Wolfsie - IBM TPF Development

October 2004

AIM Core and Enterprise Solutions

IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Single Source Macros & Tools Provided in 4.1

- Provide ability to maintain a single copy of source application programs
  - Can build the same source code for either 4.1 or z/TPF
- 4.1 enhancements available for source compatibility, where z/TPF changes require application changes,
  - Loading the program base
  - Using the ECB register save area
  - Accessing the program name and nesting information
  - Handling moving code at execution time
  - Eliminated $ from file names and macro names
  - Defining transfer vectors and data programs
  - Calling C/C++ programs from assembler programs
  - C / C++ library routines written in BAL
  - Code relocation
  - Large block sizes for tape
- Can start making changes today to facilitate your z/TPF migration

Single Source Macros & Tools Provided in 4.1

- General macro interfaces will not break in z/TPF. This includes the size of the macro expansions not being larger in z/TPF than they were in 4.1.
  - There are some exceptions where this was not possible. Where ever changes to existing 4.1 interfaces had to be made in z/TPF, single source apars or tools are provided.
  - Once you start to exploit z/Architecture and z/TPF you will need to address maintaining different code in 4.1.
- Apply single source apars to your TPF 4.1 system
  - Single source apars do not require any changes to applications - they do not break any existing interfaces
  - Single source apars enable you to change applications to make them compatable with z/TPF. The changes do not have to made at one time.
- Assess your code to identify where changes need to be made
  - For most cases scans will identify where changes can be made
  - Not all single source apars will require changes to your applications. The impact for some of the changes may be minimal to none.

Load the program base

- 4.1 applications may reload program base from ECB register save area
- CE1SVP obsoleted in z/TPF
  - z/TPF changes to program packaging and the ability to execute baseless programs
- Identify 4.1 code that uses CE1SVP and replace with LBASEC
- LBASEC (PJ29218)
  - Loads the base of the issuing ECB's currently executing program
  - Code generated in 4.1:

```
        LBASEC REG=R8
+       L     R8,CE1SVP
+       NOPR  R1
```

  - Note: The length of the in-line code generated is the same on 4.1 and z/TPF. The NOPR is to reserve space used by the z/TPF expansion. This is to ensure a base register is not exceeded due to assembling for one system or the other. The code generated on z/TPF is:

```
        LBASEC REG=R8
+       LARL      R8,START_CBCD              the program executing is CBCD
```

Using the ECB register save area

- 4.1 applications may use the ECB register save area to restore registers
- The ECB register save area has been expanded and moved in z/TPF
  - z/TPF changes to accommodate 8 byte registers
  - Moved from page 1 to page 3
- Identify applications that restore registers from the ECB register save area starting with CE1SVR, and replace with LREGSC
- LREGSC (PJ29218)
  - Loads the contents of the ECB's register save area into the requested register(s).
  - Code generated in 4.1:

```
        LREGSC LOREG=R0,HIREG=R8
+       LM    R0,R8,CE1SVR(R9)
+       NOP   0
+       NOP   0
+       NOPR  R1
```

Using the ECB register save area

- 4.1 applications may use the ECB register save area to save registers
- The ECB register save area has been expanded and moved in z/TPF
  - z/TPF changes to accommodate 8 byte registers
  - Moved from page 1 to page 3
- Identify applications that save registers in the ECB register save area starting with CE1SVR, and replace with SREGSC
- SREGSC (PJ29969)
  - Stores the contents of the ECB's register save area into the requested register(s).
  - Note: TPF System Services use the ECB register save area.
  - Code generated in 4.1:

```
        SREGSC LOREG=R0,HIREG=R8
+       STM   R0,R8,CE1SVR(R9)
+       NOP   0
+       NOP   0
+       NOPR  R1
```

Defining transfer vectors and data programs

- 4.1 program definitions
  - Transfer vectors defined in the allocator
  - Data programs not explicitly defined
- z/TPF program definitions
  - Transfer vectors must be defined in the program using the BEGIN macro
  - Programs used as data records define themselves using the BEGIN macro
- Identify application transfer vectors by going through the allocator. Identify programs used as data records. Update BEGIN macro in appropriate programs.
- BEGIN (PJ29218)
  - New keywords in 4.1 do nothing. They allow for compatability with z/TPF source
  - Define transfer vectors

```
BEGIN NAME=UBL0,IBM=YES,
    TV=(UBL1,UBL2,UBL3,UBL4,UBL5,UBL6)
```

  - Define program as a data record

```
BEGIN NAME=BKZC,IBM=YES,TPFISOC=NO,TYPE=DATA
```

Accessing the program name and nesting information

- 4.1 applications may interrogate the program nesting levels or Program Allocation Table
- Program packaging, linkages and program nesting have changed in z/TPF
- Investigate applications that call macros IDSPNL and IDSPAT and replace appropriate code with the following equivalent functions using PNAMC
- PNAMC (PJ29691 & PJ29969)
  - Determine whether a program is in the nesting chain

```
        PNAMC FINDNAME,FIELD=PGMNAME,FOUND=OK, NOTFOUND=ERROR
          .
          .
PGMNAME  DC     C'QPN0'
```

  - Determine whether there are any nested programs.

```
        PNAMC SAVENAME,FIELD=EBW000,NAMETYPE=CALLER,FOUND=RET
        EXITC
RET     BACKC
```

Accessing programs and program information

- 4.1 applications may
  - Bring a program into main storage
  - Lock a program in main storage
  - Verify whether a program is defined in the PAT
- Program packaging, program residency and defining program attributes have changed in z/TPF
  - Support >4K programs
  - All programs reside in multiple records
  - All programs core resident
  - Change how transfer vectors and data programs are defined
  - PAT does not contain entries for transfer vectors, so no longer can retrieve information for a transfer vector (Still can for parent program)
  - Can still use GETPC to bring a program into main storage, if defined as on demand

Accessing programs and program information

- Need to identify and investigate usage of GETPC and PROGC.
  - If feasable, change data programs to records
    - FILE parameter on GETPC obsoleted in z/TPF
  - If using CORE parameter on GETPC for a data program, no change is necessary, as both 4.1 and z/TPF return the address of the program starting at the 8 byte program header.
  - If using CORE parameter on GETPC for an executable program, investigate why and code an alternative, as z/TPF will return the address of the ELF header.
  - If using the GETPC to validate a program name, can replace with PROGC.
- GETPC / PROGC changes (PJ29964)
  - FILE parameter on GETPC flagged with MNOTE

Calling C / C++ programs from assembler programs

- 4.1 applications use enter type macros to call C / C++ programs
  - If passing parameters, set up R1 to point to parameter list
- Program packaging and linkages have changed in z/TPF.
  - Enters to C / C++ programs still work if no parameters are passed.
  - R1 no longer used for address of parameter list.
  - CPROC and CALLC used to call C / C++ programs when passing parameters
  - CPROC and CALLC will also work to call C / C++ programs without parameters.
- Identify BAL programs that call C / C++ programs and replace enters with:
- CPROC and CALLC (PJ29692)
  - CPROC macro defines the C language data type of the parameters
  - CALLC macro generates the code needed to enter the C/C++ program.

```
CPROC RETURN=void,BJ04,(i)          Set up linkage
CALLC BJ04(R14)                      Enter program BJ04

CPROC RETURN=i,CFUN,(i)             Set up linkage
CALLC CFUN(R2),PARMS=R4             Enter CFUN with parms
```

C / C++ library routines

- 4.1 C / C++ library routines written in BAL use TMSPC and TMSEC for the prolog and epilog
- TMSPC and TMSEC obsoleted in z/TPF. New macros provided in z/TPF (PJ29640)
  - PRLGC - generates prolog
  - EPLGC - generates epilog
  - CSTKC - get address of current C stack frame
  - PBASC - gets address of calling program's base
- Change BAL code that is C / C++ library routines to use new macros

Moving object code to execute from another location

- 4.1 programs, e.g. library routines, may move code to the ECB or stack to be executed.
- z/TPF changes to macro code generated in-line may include changing branches to branch relatives
  - Moving code generated from a macro expansion that includes branch relatives will not work.
  - DEFBC macro in z/TPF changes the global variables that are used to direct the code generation process for macro calls.
- Identify where application code moves code generated from a macro and update to wrap the code with the DEFBC macro.

Moving object code to execute from another location

- DEFBC (PJ29691)
  - In 4.1 does nothing

```
                DEFBC RELATIVE=NO,PUSH
MOVECODE        FINWC D0,ERRRTN
CODELEN          EQU     *-MOVECODE
                DEFBC POP=RELATIVE
                 .
                 .
                 .
                MVC     EBW000(CODELEN),MOVECODE
```

Eliminated $ from file names and macro names

- 4.1 $LOCKC / $UNLKC
- $ is a special character in Unix and Linux and requires special handling. Better to eliminate it from file names.
- Identify usage of $LOCKC / $UNLKC and replace with LOCKC / UNLKC
- LOCKC / UNLKC (PJ29218)
  - Same as $LOCKC and $UNLKC without the $

Change to behavior of code relocation

- 4.1 does not support relocatable adcons
  - Value of relocatable adcons in BAL loaded object code contain a relative address
  - 4.1 code assumes an offset and will not work properly with an absolute address
    - uses the displacement generated in the adcon
    - or the code must self relocate, i.e. add program base to adcon value
- z/TPF supports adcon relocation
  - Relocatable adcons in BAL code contain the absolute address after being fetched into main storage

Change to behavior of code relocation

- Run tool and investigate programs flagged. Update code accordingly.
  - Update programs. Options include
    - Replace adcons with LA instruction.
    - Replace variable in adcon with a non relocatable adcon
  - Tool provided in 4.1 to flag relocation entries in BAL programs (PJ29948)
    - Option on TLDR
    - "ADCNWARN" parameter designates adcon checking
    - Identifies programs that contain relocatable adcons
    - Reports number of relocatable adcons per program
    - Condition code 4 if any relocatable adcons found
    - Can dummy TLDR output if only want TLDR report
    - sample output:

```
TPFL0006W CHU7BM CONTAINS 0001 ADCONS
TPFL0006W CHZABM CONTAINS 0004 ADCONS
```

Support tape block sizes > 32760

- 4.1 supports maximum tape block size of 32760
  - RTL and RTA could be either blocked or unblocked
- z/TPF added support of large tape block sizes
  - 32K to 128K
  - RTL and RTA must be blocked
- Can change RTL and RTA to blocked (ZTLBL)
- Identify off-line programs that process TPF tapes.
  - Avoid defining block size in JCL for tapes created on TPF
  - If written in a high level language and it processes the RTL
    - either write off-line BAL routines to access tape or
    - modify application to write to another tape

Support tape block sizes > 32760

- If off-line program written in BAL, must code a DCBE to request LBI. This works for both LBI and 32760.

```
AFADPT   DCB   DDNAME=SYS000,                    +
               DSORG=PS,                        +
               MACRF=(GM),                      +
               EODAD=AFAEOF,                    +
               SYNAD=AFAERR,                    +
               DEVD=TA,                         +
               DCBE=AFADPTE
AFADPTE  DCBE  BLKSIZE=0
```

Summary

- Single source macros & tools provided in 4.1, as well as implementing coding recommendations
  - Makes migrating to z/TPF easier
  - Allows you to start today
  - Provide the capability to maintain a single copy of source application programs for 4.1 and z/TPF
- Single source apars for assembler programs
  - PJ29218: new macros BASEC & REGSC, updates to BEGIN, LOCKC & UNLKC
  - PJ29969: new macro SREGSC, PNAMC enhancements
  - PJ29691: new macro DEFBC, updates to PNAMC & ENTRC
  - PJ29692: new macros CALLC & CPROC
  - PJ29948: updates to TLDR
  - PJ29964: updates to GETPC & PROGC
  - PJ29640: new macros CSTKC, EPLGC, PBASC, & PRLGC
- Apply these 4.1 apars, assess your code and update accordingly to ensure a smooth and easy migration to z/TPF

Legal

IBM is a trademark of International Business Machines Corporation in the United States, other countries, or both.