z/TPF V1.1

Migrating C and C++ Programs Above The Bar

Edwin van de Grift

TPF Services & Education

edwinvandegrift@us.ibm.com

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

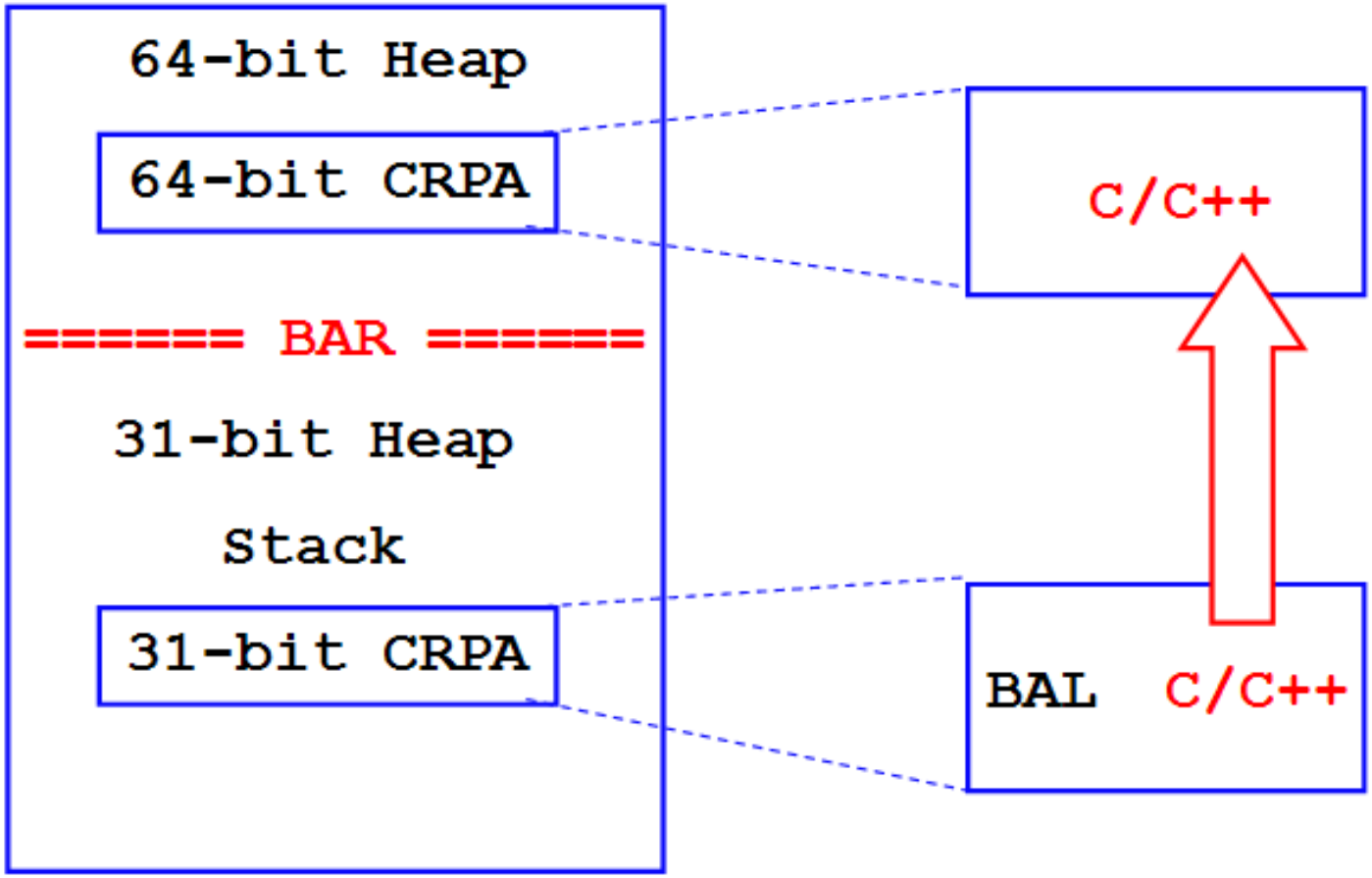
Introduction

- **The bar in this presentation refers to the 2GB bar.**
- **After migrating from TPF41 to z/TPF, your C and C++ programs likely reside BELOW the bar**
  - In the 31-bit Core Resident Program Area (CRPA)
- **This presentation discusses aspects of moving these C and C++ programs ABOVE the bar**
  - To the 64-bit Core Resident Program Area

Program Allocation

- **Assembler (BAL) programs – unless rewritten as part of the z/TPF effort – MUST reside below the bar**
  - 31-bit Assembler can only execute in the 31-bit Core Resident Program Area
- **C and C++ programs can reside either below or above the bar**
  - 64-bit Assembler can execute in BOTH Core Resident Program Areas
  - Most z/TPF customers have forced C and C++ programs to reside below the bar, as a safety precaution

Storage Layout



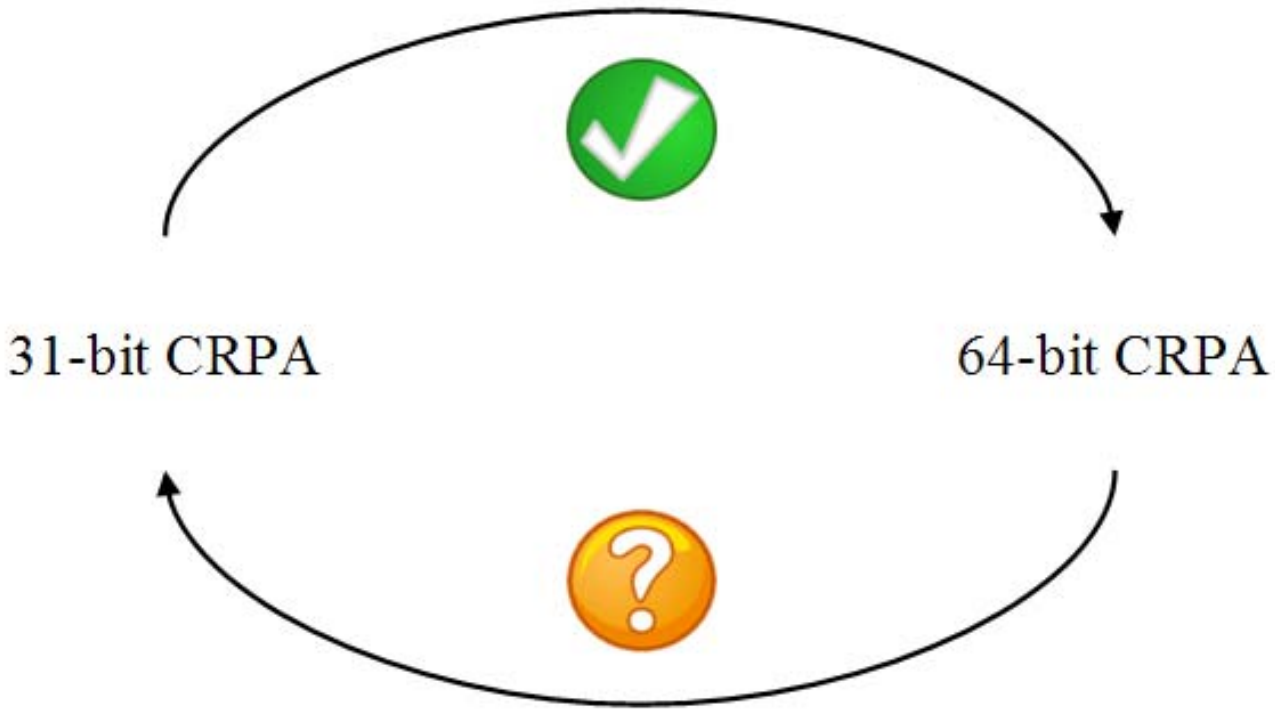
Loading a C/C++ Program Above the Bar

- **Remove the following linker option for the C/C++ program(s):**
  - -defsym CGCC_31BIT=0
    - Set in MakeTPF as follows:
    - LDFLAGS_$(APP) := -Xlinker --defsym -Xlinker CGCC_31BIT=0
- **Consideration: Make sure the size of the 64-bit Core Resident Program Areas is large enough**

Core Resident Program Areas: ZDCRP

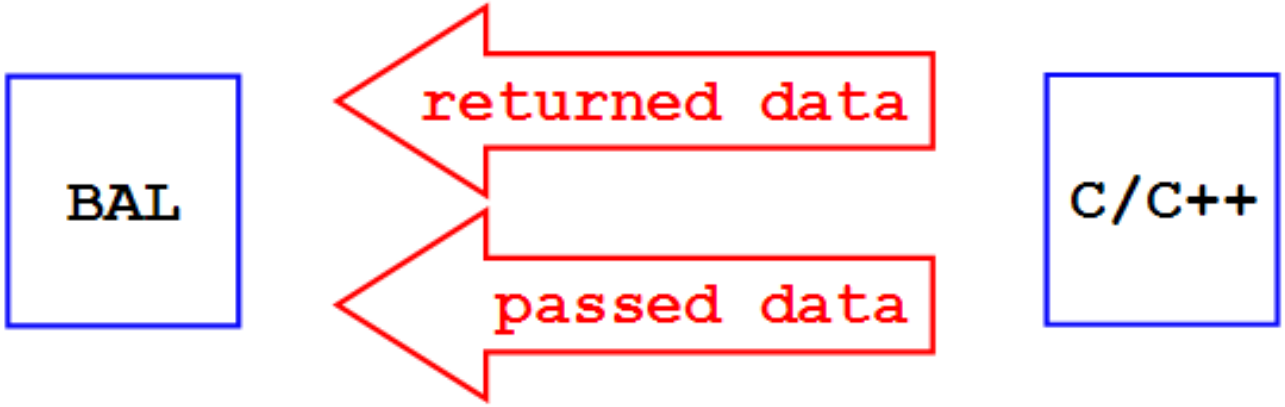
```
DCRP0004I 11.07.44 CORE RESIDENT PROGRAM AREA STATUS
                        31-BIT CRPA 31-BIT C-O-W CRPA
TOTAL ALLOCATION         41 943 040        10 485 760
CURRENTLY BACKED         11 534 336         5 242 880
PROGRAM USAGE            11 038 720         4 132 864

                        64-BIT CRPA 64-BIT C-O-W CRPA
TOTAL ALLOCATION         14 680 064        83 886 080
CURRENTLY BACKED         14 680 064        36 700 160
PROGRAM USAGE             4 214 784        35 999 744
HIGHWATER USAGE          13 344 768        35 999 744
```

Passing Data Between Assembler and C/C++



What do we need to care about?

- **When calling C/C++ from Assembler (BAL)**
  - The returned data
- **When calling Assembler from C & C++**
  - The passed data



C/C++ interacting with C/C++

- **Calling C/C++ from C/C++ is NEVER a problem**

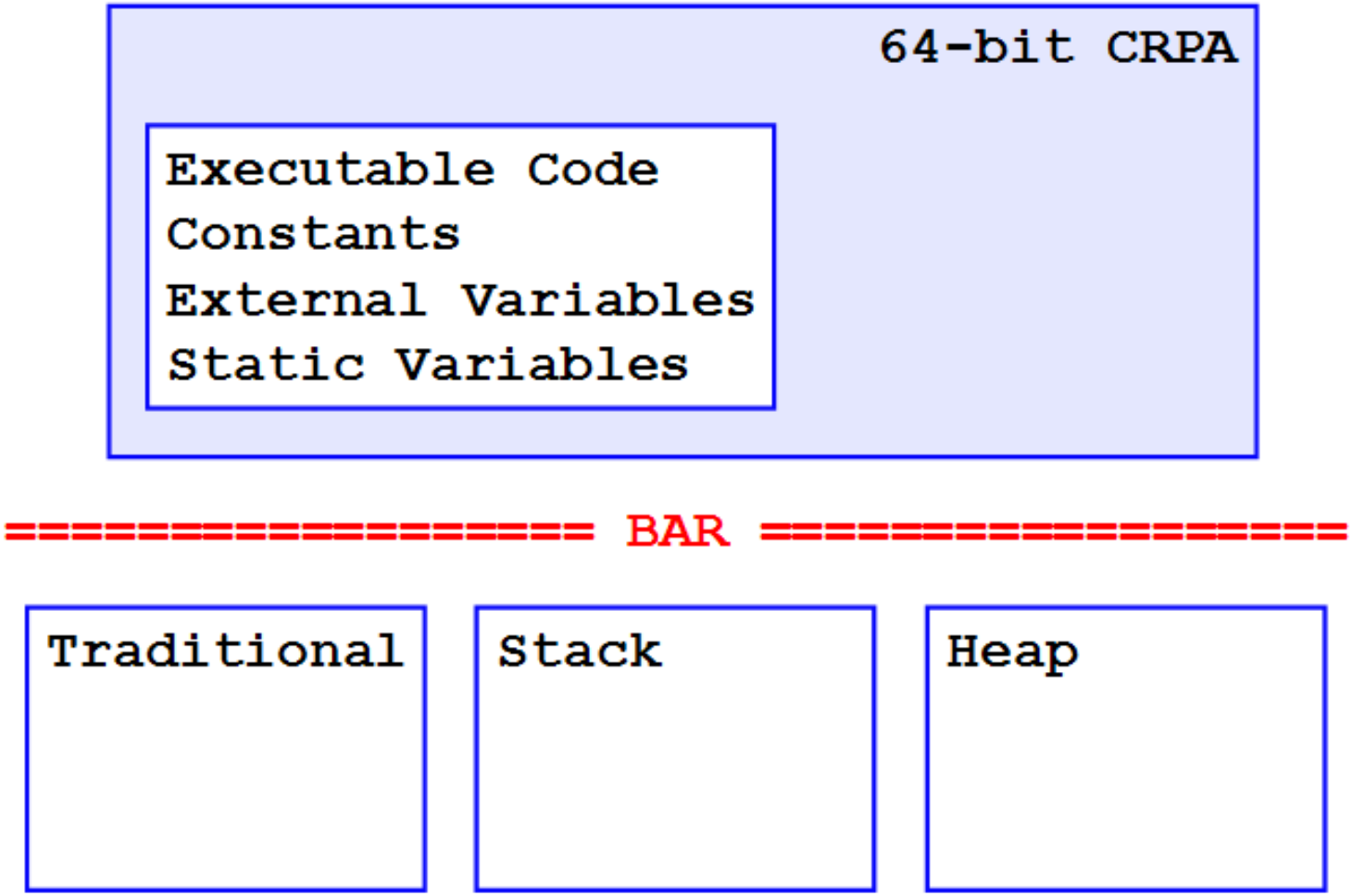
Storage Types

- **Traditional storage**
- **Stack storage**
- **Heap storage**
- **Constants, static & external variables**

Storage Types

```
int i1 = 42;
static int i2 = 43;
char* m1 = "this is external data";
static char* m2 = "this is static data";

void* myFunction(void)
{
    int i3 = 43;
    static int i4 = 45;
    char* m3 = "this is constant data";
    const char* m4 = "this is constant data";
    static char* m5 = "this is static data";
    struct mi0mi* input = ecbptr()->ce1cr0;
    return(malloc(3));
}
```

Storage Locations



Traditional Storage

- **Traditional storage resides below the bar**
- **No problem addressing traditional storage**

Stack Storage

- **Stack storage resides below the bar**
- **No problem addressing stack storage**

Heap Storage

- **Heap storage resides below the bar**
- **No problem addressing heap storage**
  - To use 64-bit heap storage, the C/C++ source must be changed
    - Requires different APIs, like malloc64(), realloc64(), et cetera

Constants, Static & External Variables

- **All of these reside in the program, and thus above the bar**
- **These CANNOT be addressed from 31-bit software**
- **Oftentimes use of static and external variables are the result of lazy programming rather than necessity**
- **You may not have to change anything**
  - “It depends.”

Summarizing

- **No problem:**
  - Traditional storage
  - Stack storage
  - Heap storage
- **Possible Problem:**
  - Storage in C/C++ programs
    - Constants
    - Static data
    - External data

No Problem or Possible Problem ?

```
int myFunction1();
char myFunction2();
long myFunction3();

char* myFunction4();
struct ab00cd* myFunction5();
```

Don’t Panic!

- **Keep in mind that pointers are ONLY a problem if they point to:**
  - Constants
  - Static variables
  - External variables

- **There still is NO problem if they point to:**
  - Traditional storage
  - Stack storage
  - Heap storage

Two Basic Approaches

- **Change the Assembler software**
  - So that the software can access 64-bit data
  - May not be feasible, depending on:
    - Quantity of software that needs access to the data
- **Change the C/C++ software / Move the data**
  - So that 31-bit programs can access the data

Change the C/C++ software / Move the data

- **Two distinct scenarios, depending on data:**
  - Data is just that (data)
    - Typically true in traditional TPF transactions
  - Data contains pointers
    - No easy way out
    - Needs to be considered case by case

Example 1

```
something* myFunction(XXX);
```

```
something* myFunction(XXX) {
    ...
    something* p = (something*)malloc(sizeof(something));
    memcpy(p, something, sizeof(something));
    return(p);
}
```

Example 1 Considerations

- **Pro**
  - No change in the C or C++ interface
- **Con**
  - Heap storage ownership lies with the calling Assembler program(s)

Example 2 – Variation 1

```
something* myFunction(XXX);                    // Old prototype
void myFunction(something* p, XXX);            // New prototype
```

```
void myFunction(something* p, XXX) {
    ...
    memcpy(p, something, sizeof(something));
    return;
}
```

Example 2 – Variation 2

```
something* myFunction(XXX);                    // Old prototype
something* myFunction(something* p, XXX); // New prototype
```

```
something* myFunction(something* p, XXX) {
    ...
    memcpy(p, something, sizeof(something));
    return(p);
}
```

Example 2 Considerations

- **Pro**
  - No storage ownership issues
- **Con**
  - C or C++ interface change
  - May not be feasible if program called from many Assembler programs
  - Could consider writing a STUB to avoid having to update many Assembler programs

Summary

- **Research:**
  - C/C++ Interfaces
  - Assembler programs calling the C/C++ interfaces
  - How is data passed from C/C++ to Assembler?
  - How is data returned from C/C++ to Assembler?
  - What are the contents of the data passed?
- **Strategy:**
  - Change the C/C++ interface?
  - Leave the C/C++ interface as is?

The End

"Sometimes you eat the bar, and sometimes, well, he eats you."

- The Stranger

Trademarks

- IBM is a trademarks of International Business Machines Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Intel, Intel Inside (logos), MMX, Celeron, Intel Centrino, Intel Xeon, Itanium, Pentium and Pentium III Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- 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.