Application cleanup after a system error (PJ47122)

Application Development

Barry Goldberg

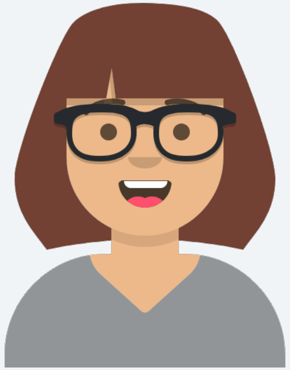
2024 TPF Users Group Conference
May 5-8 2024

IBM Z

Problem Statement

If a system error causes an ECB to exit abnormally, the application that was running at the time of the error might need to do cleanup processing

Users



Anna
Application architect

Anna worries about an application being unable to clean up after an unexpected error, which causes a loss of resources that can only be resolved with an IPL

Pain Points

- The standard atexit function is called when a process or thread (ECB) exits, but architecturally there is no equivalent function for abnormal termination (called on a system error with exit)
- There is no method for cleanup if a system error caused the ECB to exit.

As-Is User story

- An application gets system heap for a new table that will be swapped with another table in system heap.
- Before the ECB completes, the ECB takes a dump with exit
- As a result of this, the system heap used for the new table is not released and is lost until the next IPL.

To-Be user story

- An application gets system heap for a new table that will be swapped with another table in system heap.
- The application registers a function by using tpf_systemerror_atexit
- Before the ECB completes, the ECB takes a dump with exit
- The registered function releases the system heap before the ECB exits, which prevents its loss

Technical details

tpf_systemerror_atexit

```
#include <tpf/tpfapi.h>
 int tpf_systemerror_atexit(*func)

    func
      the name of a user-defined function to be called when an ECB
      exits with a system error.  The user defined function must
      contain a pointer to the idssye structure which describes the
      dump. For example, if you register a user-defined function name
      valid4, the valid4 function must have the following prototype:
            void valid4(struct idssye *ptr)
```

a.

Technical details

tpf_systemerror_atexit function

- You can pass data to the registered function using tagged ECB heap, ECB workarea, named DECB, data blocks, or system heap with a unique token
- Function will be called in the same subsystem in which it is registered in
- Information about the dump will be passed in the c_idssye structure

Technical details

tpf_systemerror_atexit function

- The tpf_systemerorr_atexit specifies the address of a function that will be called when a system error (serrc or snapc) with exit is issued.
  - The function that will be called must not take a dump with exit.
  - The function that will be called must return.
- This function will not be activated if ECB heap has been corrupted
- This function will not be activated if critical ECB fields have been corrupted
- The ECB stack at the time of error will be set to the top of the stack when the function is called. The user should not try to traverse the stack to get data from the time of error
- This function is thread unique
- A maximum of 32 functions can be registered for an ECB
- The activated function cannot issue a dump with exit or just exit the ECB itself

Technical details

tpf_systemerror_atexit

- The functions registered by tpf_systemerror_atexit will be called in a Last in first out basis (LIFO)
  - Example
    - Function A issues a tpf_systemerror_atexit function to call Function C
    - Function B issues a tpf_systemerror_atexit function to call Function D
    - A dump with exit happens in function B
      - Function D gets called first followed by Function C and then the ECB exits through CPSA

Technical details

tpf_systemerror_remove_atexit

```
#include <tpf/tpfapi.h>
 int tpf_systemerror_remove_atexit();
```

This function cancels the registration of the function that was most recently registered by the tpf_systemerror_atexit function

Technical details

tpf_systemerror_remove_atexit

- Use when there is no further need to do cleanup if a system error with exit were to occur.
- This must be done when leaving the function that requires the cleanup
- No need to issue this function when exiting the ECB, as its storage will be released during EXITC processing.
- Removes the last registered tpf_systemerror_atexit function added

Sample code application

```
void  handle_system_error(struct idssye *);
#define TOKEN “IBMTEMP1”

void QZZ9(void) { // this code will get system heap buffer for a new table
  int i,frm=2;
  struct count {int count1;} *k;

  // In case of an error go to handle_system_error
  i = tpf_systemerror_atexit(handle_system_error);

  // Create context for handle_system_error
  k = (struct count *)calloc(1,10);  // Get ECB heap buffer to hold count of frames
  k->count1= frm;                    // Set the count of system heap frames
  tpf_eheap_tag(k,"TEST1");          // Tag the ECB heap buffer

  tpf_gsysc(frm,TOKEN,"TEST OWNER",GSYSC_UNIQUE+GSYSC_64BIT+GSYSC_1MB);
  // Add data into the system heap buffer and put address in a CINFC tag (code not shown)

  // Clean up
  free(k);                          // Release the ECB heap buffer that was used for context
  tpf_systemerror_remove_atexit();  // Unregister the remove_atexit function
  return;
}
```

Sample code-registered function

```
void handle_system_error(struct idssye *i){
  int j;
  struct count { int count1;} *k;

  // Locate the ECB heap buffer that contains the count of system heap frames
  k = (stuct count *)tpf_eheap_locate("TEST1");

  if (k !=NULL)
     j=tpf_rsysc(NULL,k->count1,TOKEN); // release the system heap

  // also clear out the cinfc tag

 return;
}
```

Value Statement

With the tpf_systemerror_atexit function the application can clean up after a system error caused the ECB to exit abnormally.

Conclusion

APAR PJ47122 provides the support and was shipped in November 2023

Thank you