How to Solve Problems in C Using Dumps and Debuggers

Share Session 8130

Don Poitras

Sas Institute, Inc.

Methodology

Use the tools provided that solve 90 percent of bugs first. If that doesn't work, then start digging.

- Traceback
- SYSLOG, SYSTERM and SYSOUT
- Runtime options
  - =btrace
  - =warning
  - =fillmem
  - =version
  - =quit
- PSW and registers
- Linkedit listing
- Program listing (OMD)
- SYSUDUMP
- SYSMDUMP (IPCS)
- printf(), btrace(), wto()
- Debuggers

Traceback

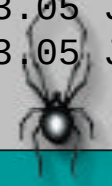
```
LSCX056 SAS/C library release  6.50C (resident), release  6.50C (transient).
LSCX041 **** ERROR ****
        ABEND occurred in line    56 of    FUNC2(SHR2DYN),offset 0000D4
        Program terminated by operating system. ABEND code = S0C4
Calling trace:
    Function         Line    Offset    Context
   FUNC2(SHR2DYN)      56    0000D4
    MAIN(SHR2MAN)      43    000158
```

```
Line#    Block                    Source

  51         1     printf("func2() was successfully dynamically called!!!:n")
  52         1     count = 5;
  53         1     printf("Local count from SHR2DYN=%i:n", count);
  54         1     printf("global count from SHR2DYN=%i:n", pGlobal->count);
  55         1     pGlobal->count = 8;
  56         1     result = (*pGlobal->callback) ();
  57         1     log("from func2:n");
  58         1     printf("Local count from SHR2DYN=%i:n", count);
  59         1     printf("global count from SHR2DYN=%i:n", pGlobal->count);
  60         1  // abort();
```

SYSLOG, SYSTERM and SYSPRINT

```
19.53.02 JOB00284 ---- SUNDAY,    27 FEB 2000 ----
19.53.02 JOB00284  IRR010I  USERID SASDTP2  IS ASSIGNED TO THIS JOB.
19.53.03 JOB00284  ICH70001I SASDTP2  LAST ACCESS AT 19:52:33 ON SUNDAY, FEBRUAR
19.53.03 JOB00284  $HASP373 SHR2MAN  STARTED - INIT 9    - CLASS T - SYS PROD
19.53.03 JOB00284  IEF403I SHR2MAN - STARTED
19.53.03 JOB00284  +LSCX064 ABEND diagnostic messages directed to ddname SYSTERM
19.53.04 JOB00284  $HASP375 SHR2MAN  ESTIMATED  LINES EXCEEDED
19.53.05 JOB00284  IEA995I SYMPTOM DUMP OUTPUT
                   SYSTEM COMPLETION CODE=0C4  REASON CODE=00000004
                    TIME=19.53.03  SEQ=61853  CPU=0000  ASID=002F
                    PSW AT TIME OF ERROR  078D3000   8F00F260  ILC 0  INTC 04
                      NO ACTIVE MODULE FOUND
                      NAME=UNKNOWN
                      DATA AT PSW  0F00F25A - 00080F00  F2600F00  0A180F00
                      GPR  0-3  0F037FA0  8F02E600  00000008  0001DFF8
                      GPR  4-7  0F033470  0F033218  0F00F260  0F00F258
                      GPR  8-11 0F0334D8  0F002548  00019FC8  00017BF8
                      GPR 12-15 80006030  0001D0B8  8F0332EE  0F00F260
                    END OF SYMPTOM DUMP
19.53.05 JOB00284  IEF450I SHR2MAN TEST1 - ABEND=S0C4 U0000 REASON=00000004
19.53.05 JOB00284  IEFACTRT005I:  STEP TEST1             ENDED ----------------
19.53.05 JOB00284  IEF404I SHR2MAN - ENDED
19.53.05 JOB00284  $HASP395 SHR2MAN  ENDED
```



General Run-Time Options

Specified on the Command Line		Specified in the Program Source	
option	negation	int _options	int _negopts
=abdump	=noabdump	_ABDUMP	_NOABDUMP
=btrace	=nobtrace	_BTRACE	_NOBTRACE
=debug	=nodebug	_DEBUG	_NODEBUG
=fillmem	=nofillmem	_FILLMEM	_NOFILLMEM
=hcsig	=nohcsig	_HCSIG	_NOHCSIG
=htsig	=nohtsig	_HTSIG	_NOHTSIG
=multitask	=nomultitask	_MULTITASK	_NOMULTITASK
=quit	=noquit	_QUIT	_NOQUIT
=storage	=nostorage	_STORAGE	_NOSTORAGE
=usage	=nousage	_USAGE	_NOUSAGE
=version	=noversion	_VERSION	_NOVERSION
=warning	=nowarning	_WARNING	_NOWARNING
=zeromem	=nozeromem	_ZEROMEM	_NOZEROMEM

http://www.sas.com/software/sas_c/document/doc/clug/zntimeop.htm

Runtime Options

- =w =q =v
  - print all warnings and quit on the first one. Often the first thing Tech. Support will ask for since this gives a dump at the first sign of trouble.
  - =v will print version information to diagnose a problem using an incompatible resident/transient library combination. (and to determine if there are known library bugs at this release.)
- =nohtsig =nohcsig
  - Useful when trying to get a dump that isn't cluttered with recovery processing. ESPIE and ESTAE routines use the same stack as the user program and info about called routines will be overlayed.
  - You won't get a traceback in this case because it's the recovery process that prints the traceback.
- =btrace
  - Print a traceback at every warning.
- =fillmem
  - fill uninitialized data with 0xFC. Also forces =fdump linkage.

Linkedit Listing

```
 *** M O D U L E  M A P ***

---------------
CLASS  B_TEXT            LENGTH =     3EF8  ATTRIBUTES = CAT,   LOAD, RMODE=ANY
---------------

 SECTION    CLASS                                                 ------- SOURCE --------
  OFFSET   OFFSET  NAME                  TYPE    LENGTH  DDNAME   SEQ  MEMBER

               0  SHR2DYN@              CSECT       368  SYSLIN    01  **NULL**
       0       0     @ENTRY               LABEL
       0       0     @DYNAMN              LABEL

             368  SHR2DYN:              CSECT        58  SYSLIN    01  **NULL**

             3C0  @@DYNAMN              CSECT         C  SYSLIN    01  **NULL**

             3D0  SHR2DYN$              CSECT       1C3  SYSLIN    01  **NULL**

             598  SHR2DYN?              CSECT        C5  SYSLIN    01  **NULL**

             660  SHR2DYN+              CSECT        54  SYSLIN    01  **NULL**

             6B8  SHR2DYN>              CSECT         6  SYSLIN    01  **NULL**
```

If a linkedit listing can not be obtained, running AMBLIST against the load module will produce the same info. Some info can also be produced by using the PDS program and the “map” command.

```
                        ---------------    ,
                        CLASS  B_PRV             LENGTH =      E01  ATTRIBUTES = MRG, NOLOAD
                        ---------------

SHR2MAN* ,XD   0004 000007 000050
GLOBAL    XD   0005 000007 000008
SHR2MAN&  XD   0006 000007 000008
COUNT     XD   0007 000007 000004
TABLE     XD   0008 000007 000004
##IO      XD   0009 000007 000004
#SCANTAB  XD   000A 000007 000100
```

CLASS OFFSET	NAME	TYPE	LENGTH	SECTION
0	SHR2MAN*	PART	50	
50	##12	PART	4	
58	COUNT	PART	4	
60	GLOBAL	PART	8	
68	SHR2MAN&	PART	8	
70	TABLE	PART	4	
78	##IO	PART	4	
80	#SCANTAB	PART	100	
180	L$UXBSIV	PART	4	
188	L$UXBSOC	PART	4	
190	L$CXLAMB	PART	40	
1D0	L$CXMINF	PART	30	
200	#STKABV	PART	4	
208	#STKRELS	PART	4	
210	#EPOCH	PART	8	
218	#ENVIRON	PART	4	
220	#NIO	PART	4	
228	#DIBPTR	PART	4	
230	##MG	PART	4	
238	##EN	PART	4	
240	L$CXSGEP	PART	4	
248	L$CXGIOT	PART	480	

In a reetrant program the DXDs created by the compiler will be combined by the linkage editor into a single map that the runtime library will use to allocate a PRV. R12 + 0xC will point to this area. There is one per load module. Externs are given a unique qcon, while statics are lumped together under the name *sname**.

Program Listing (OMD)

see the CPROLOG macro for these flags

DSA size

line number table

constant CSECT. Lib will set R4 to point here.

offset of static in PRV

```
------ /*-------------------------------------------------------------------+
------ |                                                                    |
------ | FUNCTION NAME:    func2                                            |
------ |                                                                    |
------ | DESCRIPTIVE NAME: Dynamically loaded function                      |
------ |                                                                    |
------ +-------------------------------------------------------------------*/

                                    ++++  static function
000110 47F0F028                       61    B     40(0,15)
000114 11
000115 404040C6E4D5C3F2                        FUNC2
00011D 4DE2C8D9F2C4E8D55D                      (SHR2DYN)
000126 00120000 0090
00012C 00000047                                A(SHR2DYN?+X'47')
000130 00000000                                A(SHR2DYN:)
000134 00000000
000138 90E8D00C                       62    STM   14,8,12(13)
00013C 185F                           63    LR    5,15
00013E 58E0C004                       64    L     14,4(0,12)
000142 47F0E000                       65    B     0(0,14)
000146 18D1                           66    LR    13,1
000148 98E8D00C                       67    LM    14,8,12(13)
00014C 07FE                           68    BR    14
 *** line 49 ***
```

debugger call

Nonreentrant code. If moved into LPA, this would have caused ABEND.

```
 *** line 55 ***
------    pGlobal->count = 8;

0001BE 051C                          100    BALR  1,12
0001C0 41200008                      101    LA    2,8(0,0)
0001C4 5870400C                      102    L     7,12(0,4)
0001C8 58607058                      103    L     6,88(0,7)
0001CC 50206000                      104    ST    2,0(0,6)
 *** line 56 ***
------    result = (*pGlobal->callback) ();

0001D0 051C                          105    BALR  1,12
0001D2 5880400C                      106    L     8,12(0,4)
0001D6 58708058                      107    L     7,88(0,8)
0001DA 58607004                      108    L     6,4(0,7)
0001DE 5060D078                      109    ST    6,120(0,13)
0001E2 18F6                          110    LR    15,6
0001E4 05EF                          111    BALR  14,15
0001E6 50F0D080                      112    ST    15,128(0,13)
```

rent

```
****,constant,section ****
000000 F64BF5F0 C6404040   6.50F
000008 00000000            A(SHR2MAN+)
00000C 00000000            Q(SHR2MAN*)
000010 00000000            Q(GLOBAL)
000014 00000000            Q(SHR2MAN&)
000018 000002E0            A(SHR2MAN@+X'2E0')
00001C 000001E4            A(SHR2MAN@+X'1E4')
000020 0000056C            A(SHR2MAN@+X'56C')
000024 00000000            Q(COUNT)
000028 00000000            V(SHR2MAN$)
00002C 00000000            V(PRINTF)
000030 00000000            Q(TABLE)
000034 00000000            V(UNLOADM)
000038 00000000            V(TIME)
00003C 00000000            V(#LOCALTM)
000040 00000000            V(OSBDCB)
000044 00000000            V(OSBOPEN)
000048 00000000            V(OSREAD)
00004C 00000000            V(OSCHECK)
000050 00000000            V(OSBCLOSE)
000054 00000004
000058 00000000            V(ADDSRCH)
00005C 00000000            V(LOADM)
000060 00000000            Q(##IO)
000064 00000000            Q(#SCANTAB)
```

norent

```
****,constant,section ****
000000 F64BF5F0 C6404040   6.50F
000008 00000000            A(SHR2DYN+)
00000C 00000000            V(SHR2DYN$)
000010 00000000            V(PRINTF)
000014 0000026C            A(SHR2DYN@+X'26C')
000018 00000000            V(TIME)
00001C 00000000            V(#LOCALTM)
000020 00000000            Q(##IO)
000024 00000000            Q(#SCANTAB)
000028 4028402A 402C402E
000030 40304032 40344036
000038 4038403A 403C403E
000040 00000000            V(L$CFNAD)
000044 80000040            A(SHR2DYN:+X'80000040')
000048 00000064            A(SHR2DYN@+X'64')
00004C 00000000            V(L$CFNAD)
000050 8000004C            A(SHR2DYN:+X'8000004C')
000054 00000110            A(SHR2DYN@+X'110')
000058 00000000            V(L$CFNAD)
00005C 80000058            A(@@DYNAMN+X'80000000')
000060 00000000            V(@DYNAMN)
```

```
SYMBOL    TYPE  ID    ADDR   LENGTH  LDID
SHR2DYN@   SD   0001 000000 0002BC
#DYNAMNR   ER   0002
@ENTRY     LD        000000          0001
@DYNAMN    LD        000000          0001
SHR2DYN?   WX   0003
SHR2DYN:   WX   0004
SHR2DYN:   SD   0001 000000 000058
SHR2DYN$   WX   0002
SHR2DYN+   WX   0003
##IO       XD   0004 000007 000004
```

In general, compiler-generated external names are created by appending one or more special characters to the section name. Each type of data object has a unique special character associated with it. If the section name is less than seven characters long, then all of the created names are suffixed by an @ , followed by the special character for the data object type (unless that is a second @). If the section name is exactly seven characters long, then only the special character is used as the suffix.

SD = CSECT
WX = WXTRN
XD = DXD
ER = EXTERN or VCON

Control Section Suffixes

Suffix	Type of Data	Compiler Options
@	executable code	(any)
:	constants	(any)
$	string literals	(any)
$	static data	norent , rentext
=	initialization data	rent , rentext
?	line number/offset table	lineno , debug
+	run-time constants	(always generated)
>	extended function names	extname
<	extended identifiers other than function names	extname

http://www.s390.ibm.com:80/bookmgr-cgi/bookmgr.cmd/BOOKS/ASMR1002/3%2e2%2e7?SHELF=ASMSH009#FIGCDS

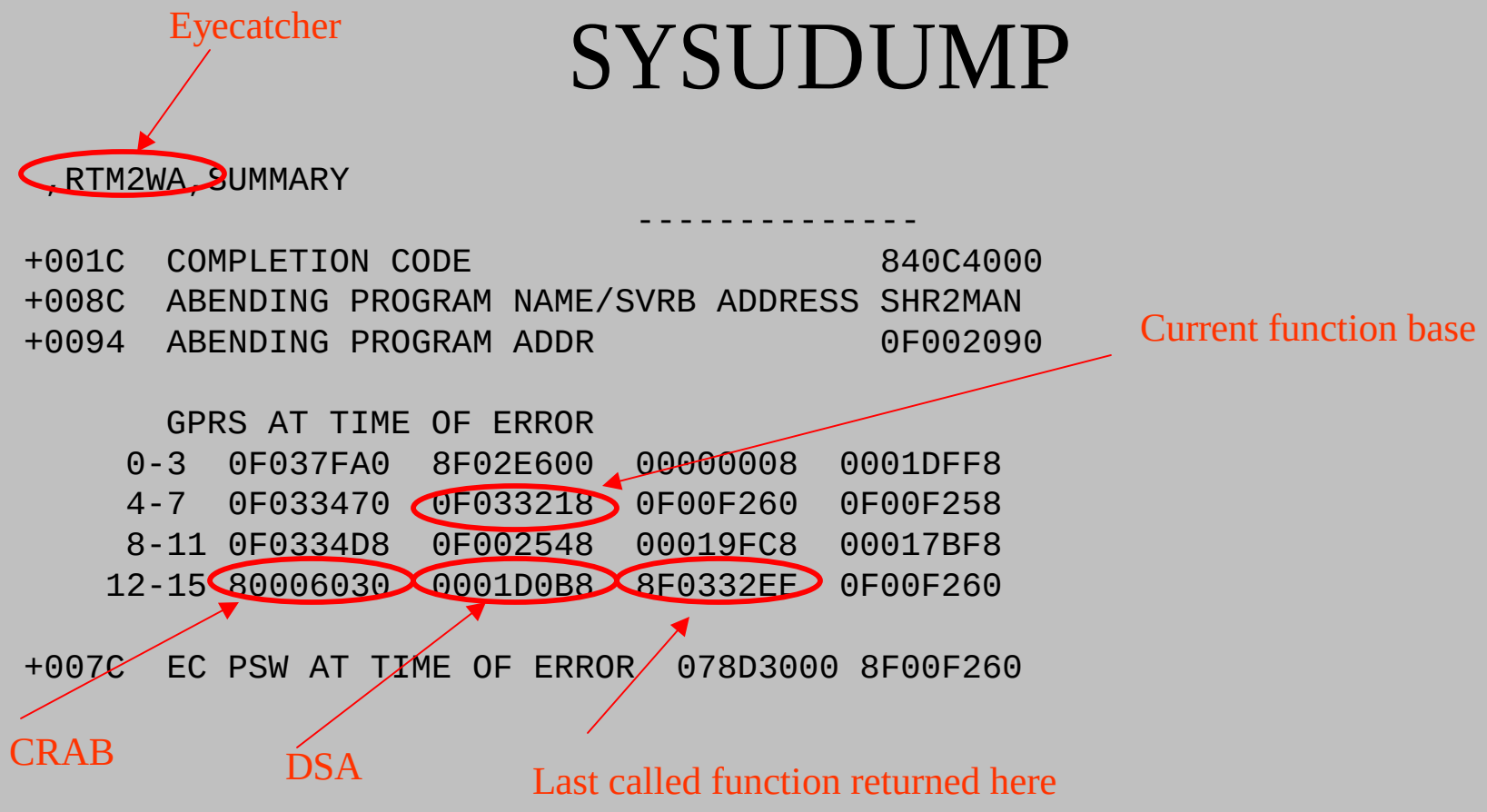
Diagnostic Code

- assert()
  - If compiled with ‘DEBUG’ option, assert() will generate code to produce a dump if the argument evaluates to 0 (false). A message with the filename and filenumber will be written and abort() will be called. To disable asserts and still compile debug, **`#undefine`** the NDEBUG symbol.
- printf()
  - The easiest way to check code.
- fprintf(stderr, …)
  - if you don’t want to clutter your normal output. Make sure SYSTERM DD is in the JCL
- wto()
  - WTP() is the macro form and will write messages to the SYSLOG or operator console if programmer messages are being displayed. Good for TSO or STC programs where adding DD’s can be problematic.

Dumps

- SYSUDUMP
  - easier to produce, just add a sysout DD
  - can be transmitted in ascii format for viewing on a PC or workstation
  - good for program producing multiple dumps
- SYSMDUMP
  - IPCS to view - very powerful
  - allows clist processing
  - at most sites gives more complete info

SYSUDUMP


Eyecatcher
,RTM2WA,SUMMARY

+001C COMPLETION CODE 840C4000
+008C ABENDING PROGRAM NAME/SVRB ADDRESS SHR2MAN
+0094 ABENDING PROGRAM ADDR 0F002090
GPRS AT TIME OF ERROR
0-3 0F037FA0 8F02E600 00000008 0001DFF8
4-7 0F033470 0F033218 0F00F260 0F00F258
8-11 0F0334D8 0F002548 00019FC8 00017BF8
12-15 80006030 0001D0B8 8F0332EE 0F00F260
+007C EC PSW AT TIME OF ERROR 078D3000 8F00F260
Current function base
CRAB
DSA
Last called function returned here

Save area trace

```
SAVE AREA TRACE
PROCEEDING FORWARD FROM TCBFSA
NAME=SHR2MAN
WAS ENTERED VIA LINK         AT EP L$CMAIN
JOB SHR2MAN          STEP TEST1           TIME 154710   DATE 00062    ID = 000

SA   00005FA0  WD1 00000000   HSA 00000000   LSA 00007028   RET 80FD0590   EPA 8F002090   R0  FD00000A
               R1  00005FF0   R2  00000040   R3  007C8D9C   R4  007C8D78   R5  007EE620   R6  007B6FF8
               R7  FD000000   R8  007EE388   R9  007F18A0   R10 00000000   R11 007EE620   R12 00DBC612
NAME=SHR2MAN
WAS ENTERED VIA CALL         AT EP     MAIN(SHR2MAN)
SA   00007028  WD1 C3E2C190   HSA 00005FA0   LSA 0F000738   RET 8F002CCA   EPA 0F000738   R0  0F000738
               R1  00007140   R2  00005FA0   R3  00007182   R4  00000002   R5  007EE620   R6  00000000
               R7  00000000   R8  00000000   R9  0F002548   R10 00019FC8   R11 00017BF8   R12 80006030
INVALID BACK CHAIN
SA   0F000738  WD1 47F0F028   HSA 11404040   LSA 40D4C1C9   RET D54DE2C8   EPA D9F2D4C1   R0  D55D0012
               R1  00000090   R2  0F001120   R3  0F000DC0   R4  00000000   R5  90E8D00C   R6  185F58E0
               R7  C00447F0   R8  E00018D1   R9  98E8D00C   R10 07FE58E0   R11 C00C5EE0   R12 400C50E0
INTERRUPT AT 8F00F260
```

System Trace Table

```
02 02D7 007EE140   SVC    78 078D0000 8F00855E  00000170 00000499 00000000
02 02D7 007EE140   SVCR   78 078D0000 8F00855E  00000000 000004A0 0F037B60
02 02D7 007EE140   PGM   011 078D3000 8F00F260  00020011 0F038F00

02 02D7 0414AB38   SRB       070C0000 8118F2F0  000002D7 00F024D0 00F02460
                                                007EE140 A0
02 02D7 00000000   SSRV  12D           81049E9E  007EE140 000B0000 00000000
                                                00000000
02 02D7 00000000   SSRV  12D           81049082  007EE140 000C8000 FF3A0000
                                                00000000
02 02D7 00000000   SSRV  12D           8104909E  007EE140 000B8000 00000000
                                                00000000
02 02D7 00000000  *RCVY ABT           8118F640  840C4000 00000004 00000000
```

Subpool list

Use	Subpool Number	Subpool Name
Miscellaneous	1	L$CMISC
I/O	10	L$CIO
Debugger	11	L$CDEBUG
Stack	12	L$CSTACK
Heap	13	L$CHEAP
ILC	14	L$CILC

```
SUBPOOL 000  KEY 08  SHARED BY TCB 007EE620
    ADDRESS 00005000  LENGTH 00001000
        FREE AREA 00005000  LENGTH 00000650
        FREE AREA 00005938  LENGTH 00000188
        FREE AREA 00005D68  LENGTH 00000180
    ADDRESS 00013000  LENGTH 00001000
        FREE AREA 00013000  LENGTH 00000F68
    ADDRESS 00025000  LENGTH 00001000
        FREE AREA 00025000  LENGTH 000009A0

SUBPOOL 001  KEY 08  OWNED BY TCB 007EE140
    ADDRESS 00006000  LENGTH 00001000
        FREE AREA 00006000  LENGTH 00000028
    ADDRESS 00008000  LENGTH 00001000
        FREE AREA 00008000  LENGTH 00000F00
    ADDRESS 0F00F000  LENGTH 00001000
        FREE AREA 0F00F000  LENGTH 00000080
    ADDRESS 0F02A000  LENGTH 00004000
        FREE AREA 0F02A000  LENGTH 00000190
    ADDRESS 0F037000  LENGTH 00001000
        FREE AREA 0F037000  LENGTH 00000B60
```

IPCS and SYSMDUMP

Pre-allocate the dataset to hold the dump. Change the JCL to use SYSMDUMP rather than SYSUDUMP.

```
//SYSMDUMP DD  DSN=SASDTP.SYSMDUMP,DISP=SHR
//*SYSUDUMP DD  SYSOUT=*
```

```
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 4160
Block size  . . . . : 4160
1st extent cylinders: 54
Secondary cylinders : 25
```

Some useful IPCS commands

```
ip summ format -          Usually the first command I do.
ip eval psw -             Find the PSW, copy, locate and stack it.
l 13r? -                  Find the current save area.
l +4? -                   Find the previous save area, contains regs
                          at entry to failing routine. Stack it.
l 12r? -                  Find the CRAB. Stack it.
l 12r?+c? -               Current PRV.
l 5r? -                   Current C base.
l 4r? -                   Current constant CSECT.
ip verbx vsmdata -        Get subpool data.
ip where xxx -            Display what module address is in.
ip st -                   A quick problem display.
ip verbx mtrace -         Master console output (not usually present)
ip verbx logdata -        Logrec output. (not usually present)
ip systrace -             System trace.
ip lpamap -               Print LPA module addresses.

Outputs can be sent to the screen or to a dataset (ddname=IPCSPRNT)
```

Typical IPCS session

- Find where the abend occurred using the PSW, R14 or System Trace.
- Inspect parameters passed to the function on the stack. Stack also contains automatic variables. OMD will show the offset from R13 where these reside.
- Look for external or static variables.
- Walk the stack looking for problems with parms passed to calling functions.
- Set bookmarks all over the place so you can take up where you left off.

ASMIDF

- Not just for assembler.
- Use the PROFILE CLIST/REXX exec to customize the look and feel.
- Turn on the PATH option to enable going backwards with the HISTORY command. Warning! If you have PATH on and do TRACEALL and set a breakpoint, you may may retire before the breakpoint hits.
- Use MAP, TASK and WHERE to figure out where you are.
- Use BREAK, and DBREAK to stop and inspect storage.
- Use LAN LOAD “csect” to show assembler source.
- Assemble ADATA and run ASMLANGX.

Happy Debugging!

mailto:sasdtp@sas.com

ftp://ftp.sas.com/techsup/download/SASC/share8130/