Basic System Problem Analysis

Bill Cadier
Hewlett-Packard Co.
TCSD MPE/iX Lab
25 April 2003

troduction

- commonly used DAT macros
- some OS structures and their types
- PA-RISC Registers
- short vs. long pointers
- overview of the procedure calling convention
- case studies

ommonly Used Macros

- sys_abort
- m_ptree
- m_family
- m_errors
- m_fpib
- i_showjob
- i_cihistory
- i_showvar
- fs_open_files
- fs_file
- fs_find_gufd_entry
- lcx
- io_ios_diag_log
- rm_format_sirs
- rm_semaphore
- rm_sem_deadlock
- process_dispatcher
- process_wait
- vs_page_info
- mm_page_info
- mm_active_io
- mm_completed_io
- tbl_info

rocess Management Structures

PIB: process information block, type “pib_type”

PIBX: process information block extension, type “pibx_type”

PCB: process control block (CM), type “pcb_type”

PCBX: process control block extension (CM), type “pcbx_type”

ob/Session Management Structures

JMAT: job master table, type “jmat_entry_type”

JIT: job information table, type “jit_entry_type”

JDT: job directory table, type “jdt_header_type”

le System Structures

PLFD: process local file descriptor (file handle), type "plfd_type"

GDPD: global data pointer descriptor (file pointers), type "gdpd_t"

GUFD: global unique file descriptor, type "gufd_t"

FLAB: file label, type "flab_t"

rtual Space Management Structures

/SOD: virtual space object descriptor, type "vs_od_type"

Cache entry: type "cache_entry_type"

B-tree's:
- extent b-tree, type "b_tree_root_type"
- extent A/R (variable access rights) b-tree, same type

e mory Management Structures

HPDIR: hashed page directory, type “hpdir_rec”

IPDIR: indexed page directory, known system object (KSO) 3, type “ipdir_rec”

MIB: memory management information block, type “mib_type”

Memory Management Globals, known system object (KSO) 4, type “mm_global_info_rec”

Dispatcher Structures

- dispatcher globals, KSD 127, type “disp_globals_type”
- TCB, task control block, type “tcb_type”

able Management

used extensively in the OS, table header type “tbl_hdr

characteristic of a “table management” table is that the
irst two words of the table point to itself

various types of tables are used, FIFO, LIFO,
monotonic etc.

TBL_INFO macro is the easiest way to view a table
management header

ystem Globals

ystem globals is ALWAYS found at address $a.c0000000

he type is “SYSTEM_GLOBALS_TYPE”

he macro SYSGLOB will return a field within the object.

hort pointer can be created with ZDEPI 3,1,2,rx

A-RISC General Registers

-RISC uses 32 general registers. The procedure calling convention defines:

- R30 is “SP” or the stack pointer
- R27 is “DP” or the data pointer (global variables)
- R2 is “RP” or the procedure return pointer
- R28 and R29 are function return (ret0 and ret1)
- R26, R25, R24 and R23 can contain the first four arguments passed to procedures (arg0..arg3)
- R31 is used as the “millicode RP”
- R0 is a read only register whose value is zero

A-RISC Space Registers

- There are 8 space registers, SR0 to SR7
- SR0 saves space ID for external branches
- SR1 to SR3 loaded by software as needed
- SR4, SR5, SR6 and SR7 are defined by the calling convention
  - SR4 is code, typically the space ID of your program
  - SR5 is data, the space ID of a process STACK
  - SR6 is always $b (#11), OS structures and short mapped files
  - SR7 is always $a (#10), OS structures, NL.PUB.SYS and short mapped files

hort vs. Long Pointers

oad and Store instructions that specify a space register of
ero intend that the hardware will derive the space register
y using the first 2 bits of the offset portion of the address
nd add 4 to that giving the SR number to use.

```
OW -296(0,30),22
```

R30 contains 418432f0 the '4' is 0100 in binary. The first 2
ts, are 01 + 4 = 5. So SR5 will be used to complete the
inter.

hort vs. Long Pointers

short pointer addressing the high order 2 bits of an offset a
ed to denote the space register therefore they are NOT us
part of the address. This means that using short pointers
its addressability to $2^{(30)}$-1 or 1GB.

ong pointer reference would specify a space register from
7, for example:

```
W  272(sr1,r19),r21
```

rocedure Calling Convention

Stack Frames are built by non-leaf procedures so that
when they call other procedures registers can be spill
nto the frame and restored from there on return.

GR3 through GR18 are "callee save" registers, they ar
spilled, if necessary by the procedure that is being
called.

GR19 through GR22 are "caller save" registers, saved
by the procedure making the call.

rocedure Calling Convention: Registers

```
n = FREAD ( MPE_fd, &buf, -32767 );

 =00000000 40100480 013dc097 41845630 R4 =d66e8018 d66ea018 00000000 00000000
 =00000000 00000000 00000000 00000000 R12=00000000 00000000 00000000 00000000
6=00000000 00000000 00000000 00000000 R20=0000000a 013dc08c c01075a0 000002d6
4=41845abc d44b1400 0000000a c0202008 R28=00000020 00000000 4184db30 0000008b

 ($2c5) nmdebug > dv sp-60,10
RT $2d6.4184dad0 $ 00000000 4184568c 0000004d 4184567c
RT $2d6.4184dae0 $ 00000029 00000000 00000000 00000000
RT $2d6.4184daf0 $ 00000000 00000000 00000000 ffff8001 <- sp-34
RT $2d6.4184db00 $ 40bbee00 00000000 d44b1400 4164671c
```

le number is $a or 10. The 'buffer' parameter to FREAD is a long pointer. As sult it must be aligned in registers and R23 and R24 will contain the value 6.41845abc and R25 is skipped. We only use R26..R23 so the "length" arameter is saved in the stack at SP-$34.

rocedure Calling Convention: Stack rame

```
3dc0c8   FREAD          6bc23fd9   STW      r2,-20(sr0,r30)
3dc0cc   FREAD+$4       6fc30200   STWM     r3,256(sr0,r30)
3dc0d0   FREAD+$8       6bc43e09   STW      r4,-252(sr0,r30)
3dc0d4   FREAD+$c       6bc53e11   STW      r5,-248(sr0,r30)
3dc0d8   FREAD+$10      6bd73da1   STW      r23,-304(sr0,r30
3dc0dc   FREAD+$14      6bd83da9   STW      r24,-300(sr0,r30
3dc0e0   FREAD+$18      08000240   OR       r0,r0,r0
. .
3dc160   FREAD+$98      d35a1ff0   EXTRS    r26,31,16,r26
```

rocedure Calling Convention: SP & PSP

nce the STWM (or LDO) instruction is executed to build a
ew stack frame, all references to a procedure's parameters
ecome "PSP" (previous stack pointer) relative.

R26 to GR23 **may** be spilled to PSP-$24 to PSP-$30
spectively.

ou cannot count on that occurring! There may be no need
save a register to memory.

ase Study: SA663

```
=a.0019fe78 system_abort
 0) SP=418562e0 RP=a.00a51bc8 sm_quarantine_gufd+$1fc
 1) SP=418562e0 RP=a.00ee5a5c tm_close_common.tm_unlink_plfd_and_gdpd+$18
 2) SP=418558e0 RP=a.00ee75cc tm_close_common+$1a98
 3) SP=41855860 RP=a.0158a8e4 tm_ord_fix_buf_disc+$1e4
 4) SP=418548a0 RP=a.01164370 fclose_nm+$5d4
 5) SP=418547e0 RP=a.01163d68 ?fclose_nm+$8
      export stub: a.013d22a8 FCLOSE+$b8
 6) SP=41854560 RP=a.013d21bc ?FCLOSE+$8
      export stub: 298.00279b68 cr_fclose+$1c
 7) SP=418544a0 RP=298.00272350 COB_CLOSE+$17c
 8) SP=41854468 RP=298.0026e804 ?COB_CLOSE+$8
      export stub: 97c.0000e1d4
 9) SP=418543f0 RP=97c.00000000
  (end of NM stack)
```

ase Study: SA663

```
a ($70) nmdat > lev 5

b ($70) nmdat > dv psp-60,10
T $866.41854500 $ d6ef0a94 41854418 05650003 4f4bbec1
T $866.41854510 $ ca12a970 0300000a 84000000 013d21bc
T $866.41854520 $ 06020000 00000000 00000003 00000000
T $866.41854530 $ 01030000 00000000 41850000 4185000d
```

lose_nm has spilled the file number $d to the stack.
e member that the file number is a 16 bit value (see the
CLOSE intrinsic definition).

ase Study: SA663

```
193 ($70) nmdat > fs_file(,d)
ilename: TESTFILE.PUB.AP

ative Mode file
ccess options:          APPEND,NOMR,LOCK,SHR,BUF,NOMULTI,WAIT,NOCOPY
ccess method:           $0
ast error number:       $0
 . .
ile options:            SYS,BINARY,FORMAL,F,NOCCTL,DEQ,STD,NOLABEL
ile code:               $9c5
ecord size:             $100
lock size:              $100
ecord limit:            $fffeff00

dev:                    $76
```

ase Study: SA663

```
9d ($70) nmdat > lev 2

9e ($70) nmdat > dc pc
S $a.ee5a5c
ee5a5c  tm_close_common.tm_unlin*+$184    2000008f  ** Stmt  143

9f ($70) nmdat > dcx pc-184,188/4
```

ase Study: SA663

```
. .
_close_common.tm_unlin*+$118    4bda3eb1  LDW      -168(sr0,r30),r26
_close_common.tm_unlin*+$11c    287fefff  ADDIL    $fffff000,r3,1
_close_common.tm_unlin*+$120    343900c8  LDO      100(r1),r25
_close_common.tm_unlin*+$124    4bd83ea9  LDW      -172(sr0,r30),r24
_close_common.tm_unlin*+$128    23ed0012  LDIL     $91a000,r31
_close_common.tm_unlin*+$12c    e7e02430  BLE      536(sr4,r31)
.ee5a04          *Call To:  tm_unlink_gdpd
_close_common.tm_unlin*+$130    081f0242  OR       r31,r0,r2
_close_common.tm_unlin*+$134    2000008d  ** Stmt  141
_close_common.tm_unlin*+$138    4bd63ea9  LDW      -172(sr0,r30),r22
_close_common.tm_unlin*+$13c    4ac10000  LDW      0(sr0,r22),r1
_close_common.tm_unlin*+$140    84202132  COMIBT,=,N 0,r1,tm_unlink
lfd_and_gdpd+$1e0
. .
```

ase Study: SA663

```
11 ($70) nmdat > dv sp-#172
RT $866.41855834 $ 418545a8

12 ($70) nmdat > dv [sp-#172]
RT $866.418545a8 $ fc0e008f

13 ($70) nmdat > wl errmsg(S16(fc0e), 8f)
e manager; unable to unlink the GDPD.
```

ase Study: SA663

```
a6 ($70) nmdat > fv fs_gufd(fs_plfd(,d)) 'gufd_t'

CORD
. .
ILE_VIR_ADDR                    : 2e4.0
 GDPD_PTR                         : 0
. .
UARANTINE_REASON                :
       ALL : fc0e008f
 QUARANTINE_TIME                  : 3b167877d4453
 EOF_OFFSET                       : 94dd300
. .
TORE_ACTIVE              : 1
```

angs

hangs are usually difficult to diagnose.

determine the scope of the hang, what is affected

gather as much information as possible BEFORE deciding to get a memory dump.

f you have to reboot the system to clear a hang you may as well get a memory dump too, time permitting

memory dumps of hangs can be MUCH larger than system abort memory dumps

Memory Dump is a Static Image

this aircraft…

taking off or landing?
f landing, where has it been?
f taking off, where is it going?
what is its speed?
what is its current heading?

efore a Memory Dump

repeating the SHOW PROC command can tell if
processes are using CPU time or not

SHOW JOB will tell what is presently running

SHOW Q/SHOW WG will show the present queue &
workgroup settings

are disks active or idle

use debug to trace suspect processes
- macros such as pm_semaphore, rm_semaphore can hel

ase Study: Hang Memory Dump

```
0 ($0) nmdat > process_wait

                ===== DISPATCHER INFORMATION FOR A PROCESS =====

PIN #  State          Wait Event      Pri    Class  Blocked Reason
----   -----          ----------      ---    -----  --------------
1      LONG_WAIT      IPC             $7918  AS     Known Port ffffffed
                                                    Progen Global Port
2      LONG_WAIT      IPC             $38ff  BS     JUNK_WAIT
  .
886    LONG_WAIT      Control Block   $33ff  CS     CNTL_BLOCK_WAIT
887    LONG_WAIT      IPC             $33ff  CS     TERMINAL_READ_WAIT
888    LONG_WAIT      IPC             $33ff  BS     Jsmain Port ffff7fb0
889    LONG_WAIT      IPC             $33ff  CS     CHILD_WAIT
88a    LONG_WAIT      Control Block   $1bff  CS     CNTL_BLOCK_WAIT
```

ase Study: Hang Memory Dump

```
51 ($0) nmdat > pin 86

52 ($86) nmdat > pm_semaphores

DRESS OF SEMAPHORE WAITED ON: $b.88ae19b0

54 ($86) nmdat > rm_semaphore b.88ae19b0

st of pins waiting on semaphore at $b.88ae19b0
0 $6e $76 $7e $86 $8e $92 $96 $9a $9e $a2 $a6 $aa $ae
n $35 has an exclusive lock on shareable semaphore at $b.88ae19b
```

ase Study: Hang Memory Dump

```
56 ($86) nmdat > pin 35

57 ($35) nmdat > tr,d,i
     PC=a.0017099c enable_int+$2c
* 0) SP=41853ef0 RP=a.00786004 notify_dispatcher.block_current_process+$3
  1) SP=41853ef0 RP=a.00787e44 notify_dispatcher+$268
  2) SP=41853e70 RP=a.001b6034 sem_block.wait_for_resource+$1bc
  3) SP=41853d70 RP=a.001b6428 sem_block+$358
  4) SP=41853cb0 RP=a.00757ce8 cb_shr_lock+$240
  5) SP=41853bb0 RP=a.00757a94 ?cb_shr_lock+$8
       export stub: fb.011380f8 lock'set'exclusive_345+$230
  6) SP=41853a70 RP=fb.0113bc78 nmdbunlock+$16f4
  7) SP=41853a30 RP=fb.0109ae70 dblock+$10c
  8) SP=418522b0 RP=fb.0109ad38 ?dblock+$8
       export stub: 48f.000060a0
```

ase Study: Hang Memory Dump

```
58 ($35) nmdat >pm_semaphores

DRESS OF SEMAPHORE WAITED ON: $b.88ae18d0

59 ($35) nmdat > rm_semaphore b.88ae18d0

st of pins waiting on semaphore at $b.88ae18d0
5 $72 $7a $82 $8a
n $60 has an exclusive lock on shareable semaphore at $b.88ae18d
```

ase Study: Hang Memory Dump

```
5a ($35) nmdat > pin 60

5b ($60) nmdat > tr,d,i
     PC=a.0017099c enable_int+$2c
* 0) SP=41853ef0 RP=a.00786004 notify_dispatcher.block_current_process+$3
  1) SP=41853ef0 RP=a.00787e44 notify_dispatcher+$268
  2) SP=41853e70 RP=a.001b6034 sem_block.wait_for_resource+$1bc
  3) SP=41853d70 RP=a.001b6428 sem_block+$358
  4) SP=41853cb0 RP=a.00757ce8 cb_shr_lock+$240
  5) SP=41853bb0 RP=a.00757a94 ?cb_shr_lock+$8
       export stub: fb.011380f8 lock'set'exclusive_345+$230
  6) SP=41853a70 RP=fb.0113bc78 nmdbunlock+$16f4
  7) SP=41853a30 RP=fb.0109ae70 dblock+$10c
  8) SP=418522b0 RP=fb.0109ad38 ?dblock+$8
       export stub: 309.000060a0
```

ase Study: Hang Memory Dump

```
5c ($60) nmdat > pm_semaphores

DRESS OF SEMAPHORE WAITED ON: $b.88ae19b0

5d ($60) nmdat > rm_semaphore b.88ae19b0

st of pins waiting on semaphore at $b.88ae19b0
0 $6e $76 $7e $86 $8e $92 $96 $9a $9e $a2 $a6 $aa $ae
n $35 has an exclusive lock on shareable semaphore at $b.88ae19b
```

Haven’t we been here before?

ase Study: Hang Conclusion

he hang is due to a database locking problem

he memory dump probably was not necessary,
OBUTIL "SHOW LOCKS" would probably have helped
determine what the problem was

he TELESUP utility "UNDEDLOCK" might even have
been able to correct it

ne End!

THANK YOU

invent