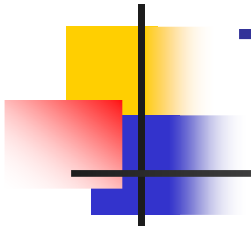
Heap Damage, Get into the zone!

Thomas Petrolino
IBM Poughkeepsie
tapetro@us.ibm.com



Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

Language Environment

z/OS

CICS

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
UNIX is a registered trademark of The Open Group in the United States and other countries.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

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.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

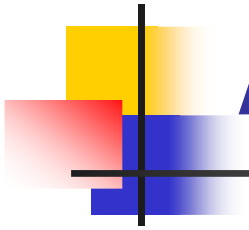
All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations 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.

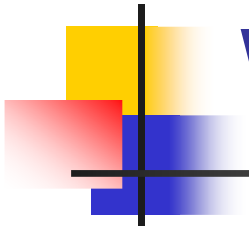
Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products 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.



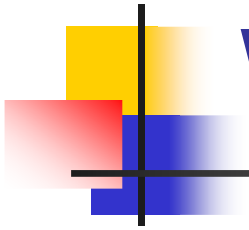
Agenda

- What do heaps look like?
- What kind of damage might you see?
- Who caused the damage?
- Getting closer to the zone!
- The HEAPZONE!
- Sources of Additional Information



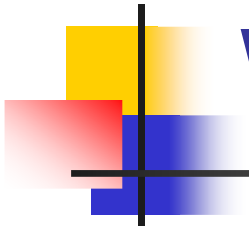
What do heaps look like?

- Language Environment Storage Management
  - Heaps
    - Completely random access
    - Allows storage to be dynamically allocated at runtime
  - Enclave level control structures
    - Each 'main' has its own enclave and therefore heap
    - Each 'link' create a new enclave and therefore heap
    - pthreads share a single heap for all threads



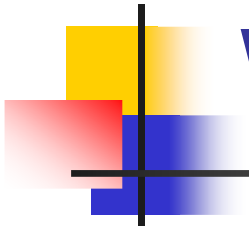
What do heaps look like?

- Language Environment Storage Management
  - Heaps
  - Four independently maintained sets of heap segments all with similar layouts:
    - User Heap
      - COBOL WORKING-STORAGE
      - C/C++ (malloc() or operator new)
      - PL/I dynamic storage (allocate)
    - LE Anywhere Heap
      - COBOL and LE above the line CBs
    - LE Below Heap
      - COBOL and LE below the line CBs
    - Additional Heap
      - Defined by the user



What do heaps look like?

- Heap Layout
  - Based on algorithm from IBM's Watson Research Center
    - FAST 1ST!!!
    - Storage Efficiency 2nd
    - Error checks 3rd
  - Elements of user requested length
    - User responsible for requesting storage be freed.

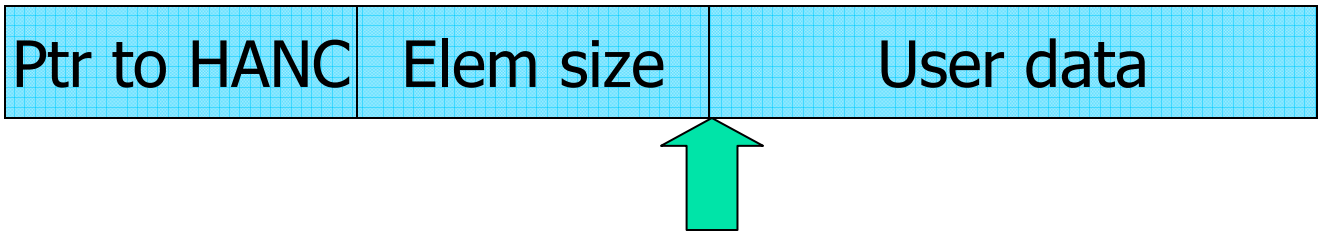


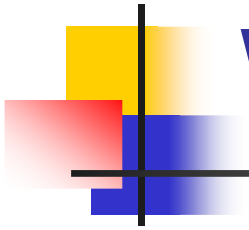
What do heaps look like?

- Heap Layout continued...
  - Header
    - x'20' byte header (8 fullwords of data)
      - Eyecatcher 'HANC'
      - Pointer to next Heap segment or HPCB
      - Pointer to previous Heap segment or HPCB
      - Heapid (user heap = 0)
      - Pointer to beginning of segment
      - Root address (largest free element in segment)
      - Heap Segment Length
      - Root element length (size of largest free element)

What do heaps look like?

- Heap Layout continued...
  - Allocated element
    - 8 byte header
      - Pointer to beginning of heap segment (HANC)
      - Size of element including header
    - User portion
      - Address returned to user



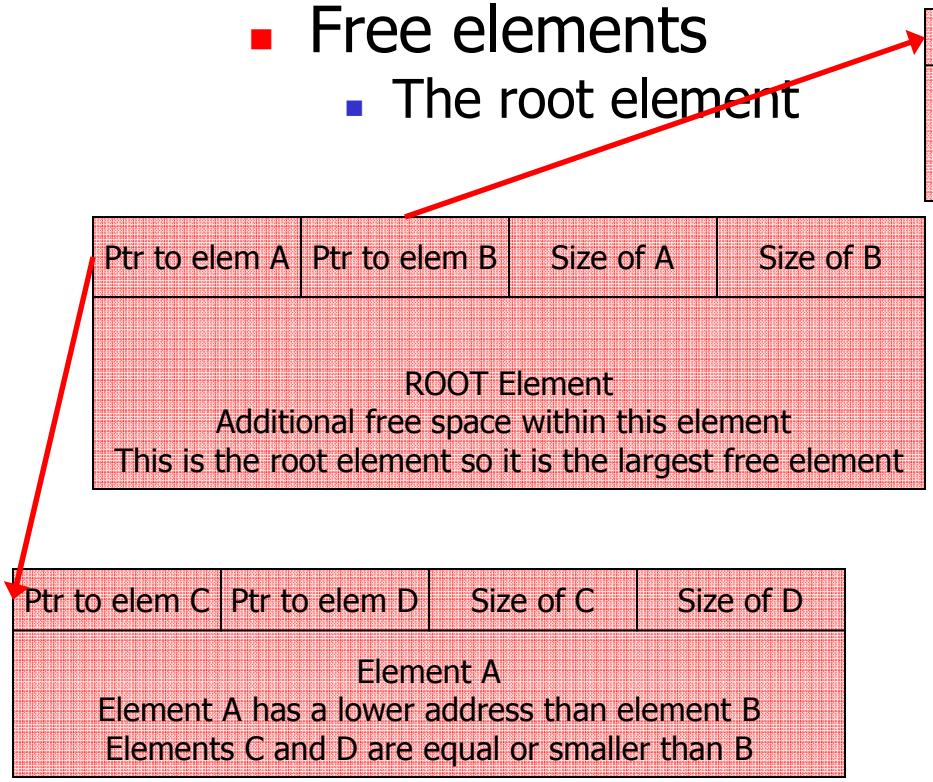


What do heaps look like?

- Heap Layout continued...
  - Free elements
    - Maintained in a Cartesian Tree
      - Larger elements toward the root
      - Smaller elements toward the leaves
      - Lower addresses to the left
      - Higher addresses to the right
  - Each free area contains x'10' bytes of information about OTHER free elements.
    - Left node address
    - Right node address
    - Left node size
    - Right node size

What do heaps look like?

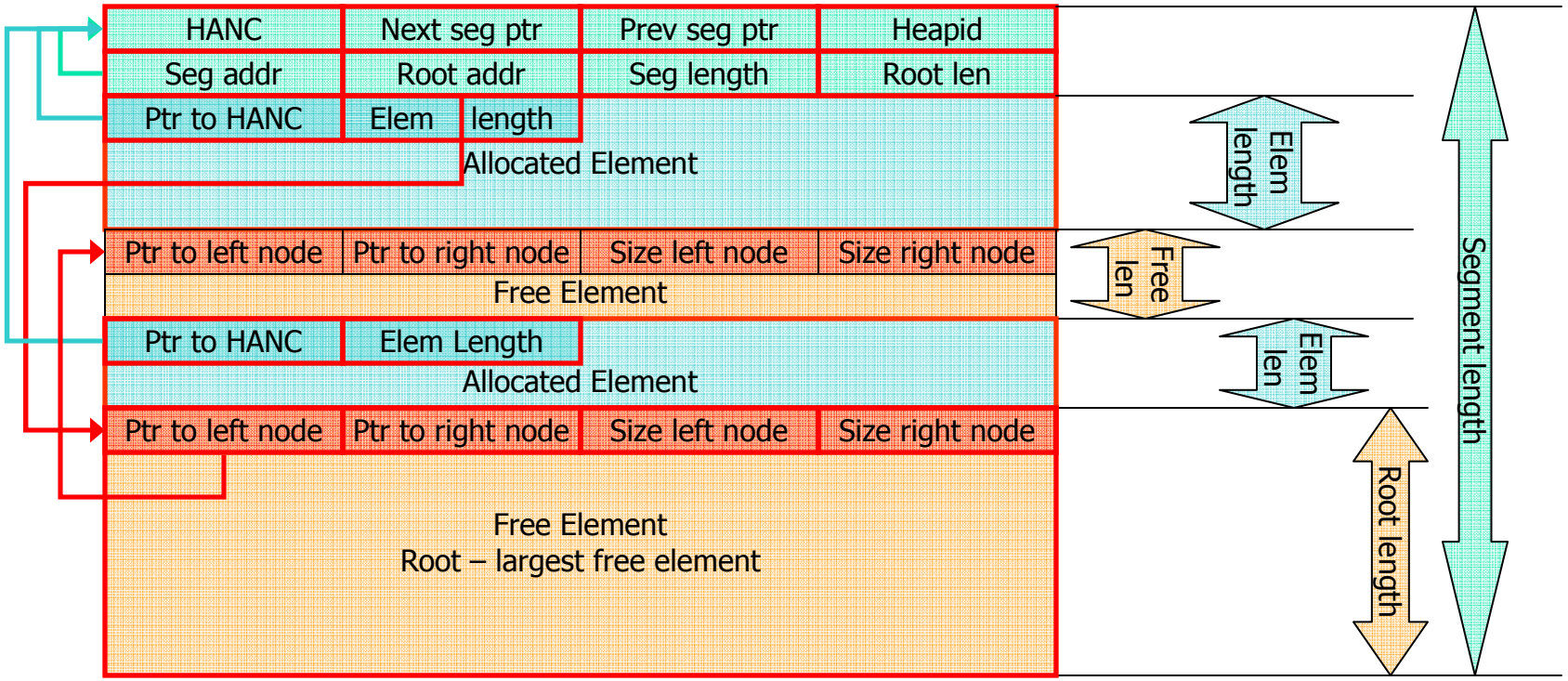
- Heap Layout continued...
  - Free elements
    - The root element

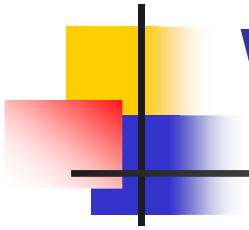


- Free elements pointing to 0
  - End that branch of the tree
- Free elements pointing to element with size of 8
  - The eight byte element begins a "twig"
    - May be multiple 8 byte elements

What do heaps look like?

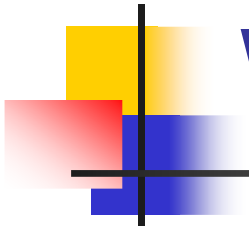
- Heap Layout continued... A "simple" example





What do heaps look like?

- Processing
  - Allocation
    - The Free Element tree of the latest heap segment is searched starting at the root for the smallest element which will satisfy the request.
    - If no free element found in that segment earlier segments are searched.
    - If no free element is large enough to satisfy the request, an additional heap segment is allocated via GETMAIN.
    - The needed storage from the free element is then allocated (8 byte header filled out).
      - If additional storage is left over in the element it is added to the free tree as a new free element.
    - A pointer to the user storage (after the 8 byte header) is returned to the user.



What do heaps look like?

- Processing
  - Free
    - Size of element is determined from the 8 byte header.
    - Element (including header) is returned to the free tree
      - If free element is adjacent to an existing free element the elements are combined into a larger free element
      - The free tree will be restructured as necessary.
    - If the entire heap is now free the segment will be FREEMAINed if FREE is specified.

Let's cause some damage

- Sample program to cause heap damage

```
000100       IDENTIFICATION DIVISION.
000200       PROGRAM-ID. SHAREHP.
000300
000400       DATA DIVISION.
000500       WORKING-STORAGE SECTION.
000600       01  WS-EYE                PIC X(4) VALUE "WS-B".
000700       01  HEAPID                PIC S9(9) BINARY VALUE 0.
000800       01  STORAGE-SIZE-24         PIC S9(9) BINARY VALUE 24.
000900       01  STORAGE-SIZE-16         PIC S9(9) BINARY VALUE 16.
001000       01  STORAGE-SIZE-8        PIC S9(9) BINARY VALUE 8.
001100       01  ADDRESS-0             POINTER.
001200       01  ADDRESS-1             POINTER.
001300       01  ADDRESS-2             POINTER.
001400       01  WS-DATA.
001500         03 WS-DATA1             PIC X(16) VALUE "1234567890123456".
001600         03 WS-DATA2             PIC S9(9) BINARY VALUE 0.
...
003100       LINKAGE SECTION.
003200       01 HEAP-DATA-AREA       PIC X(17).
003300       PROCEDURE DIVISION.
```

Let's cause some damage

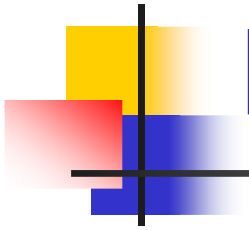
- Sample program to cause heap damage…

```
003400       MAIN-PROG.
003500           DISPLAY "STARTING SHAREHP..."
003600           CALL "CEEGTST" USING HEAPID, STORAGE-SIZE-16,
003700                                ADDRESS-0, FC.
003900           CALL "CEEGTST" USING HEAPID, STORAGE-SIZE-16,
004000                                ADDRESS-1, FC.
004200           CALL "CEEGTST" USING HEAPID, STORAGE-SIZE-16,
004300                                 ADDRESS-2, FC.
004400
004500           CALL "CEEFRST" USING ADDRESS-1, FC.
004600
004700           SET ADDRESS OF HEAP-DATA-AREA TO ADDRESS-2.
004800           MOVE WS-DATA TO HEAP-DATA-AREA.
004900
005000           CALL "CEEGTST" USING HEAPID, STORAGE-SIZE-8,
005100                                  ADDRESS-1, FC.
005200
005300           CALL "CEEFRST" USING ADDRESS-0, FC.
005400           CALL "CEEFRST" USING ADDRESS-1, FC.
005500           CALL "CEEFRST" USING ADDRESS-2, FC.
005600           GOBACK.
```

Let's cause some damage

- Output gives no clue to what happened

```
08.51.19 JOB25721 ---- WEDNESDAY, 16 FEB 2005 ----
08.51.20 JOB25721  $HASP373 JMONTIGO STARTED - WLM INIT  - SRVCLASS WLMLONG  - SYS AQFT
08.51.20 JOB25721  IEF403I JMONTIGO - STARTED - TIME=08.51.20
08.51.23 JOB25721  IEA995I SYMPTOM DUMP OUTPUT  345
   345               USER COMPLETION CODE=4039 REASON CODE=00000000
   345              TIME=08.51.21  SEQ=09748  CPU=0000  ASID=00BA
   345              PSW AT TIME OF ERROR  078D1000   A01ECE96  ILC 2  INTC 0D
   345                ACTIVE LOAD MODULE           ADDRESS=201256C0  OFFSET=000C77D6
   345                NAME=CEEPLPKA
   345                DATA AT PSW  201ECE90 - 00181610  0A0D58D0  D00498EC
   345                AR/GR 0: 80AB5B3E/84000000   1: 00000000/84000FC7
   345                      2: 00000000/20381F88   3: 00000000/00000002
   …
   345                      E: 00000000/A01E0DDE   F: 00000000/00000000
   345              END OF SYMPTOM DUMP
08.51.23 JOB25721  IEA993I SYSMDUMP TAKEN TO POSIX.JMONTI.SHARE.HEAP.SYSMDUMP
08.51.23 JOB25721  IEF450I JMONTIGO GO - ABEND=S000 U4038 REASON=00000001  347
   347                     TIME=08.51.23
```



Let's cause some damage

- Messages speak of damage but who?

```
STARTING SHAREHP...
CEE0802C Heap storage control information was damaged.
         From compile unit SHAREHP at entry point SHAREHP at statement 896 at
compile unit offset +000005C2 at entry offset +000005C2 at address 201011B2.
```

- Messages can only:
  - Describe that there is damage
  - Tell where the program was when it first saw the damage
  - But CANNOT tell who was at fault!

Let's cause some damage

- More information from CEEDUMP

```
CEE3DMP V1 R6.0: Condition processing resulted in the unhandled condition.        02/16/05 8:51:23 AM                Page:  1

Information for enclave SHAREHP

  Information for thread 8000000000000000

  Traceback:
    DSA Addr  Program Unit  PU Addr   PU Offset  Entry        E Addr    E  Offset   Statement  Load Mod  Service  Status
    20381648  CEEHDSP       201DD170  +00003F58  CEEHDSP      201DD170  +00003F58              CEEPLPKA  UQ91316  Call
    203814B0  CEEHSGLT      201EF1B8  +0000005C  CEEHSGLT     201EF1B8  +0000005C              CEEPLPKA  HLE7709  Exception
    20381398  CEEV#GTS      202BBFF8  +00000698  CEEV#GTS     202BBFF8  +00000698              CEEPLPKA  HLE7709  Call
    203812E8  CEEVGTST      202CC720  +00000072  CEEVGTST     202CC720  +00000072              CEEPLPKA  HLE7709  Call
    20381100  IGZCFCC       2034AB78  +000002CA  IGZCFCC      2034AB78  +000002CA              IGZCPAC            Call
    20381030  SHAREHP       20100BF0  +000005C2  SHAREHP      20100BF0  +000005C2         896  SHAREHP            Call
```

  - Has details about what led up to the DETECTION of the damage
    - CEEV#GTS recognized there was damage
  - We are told who "detected" the error, not who "caused" the error.

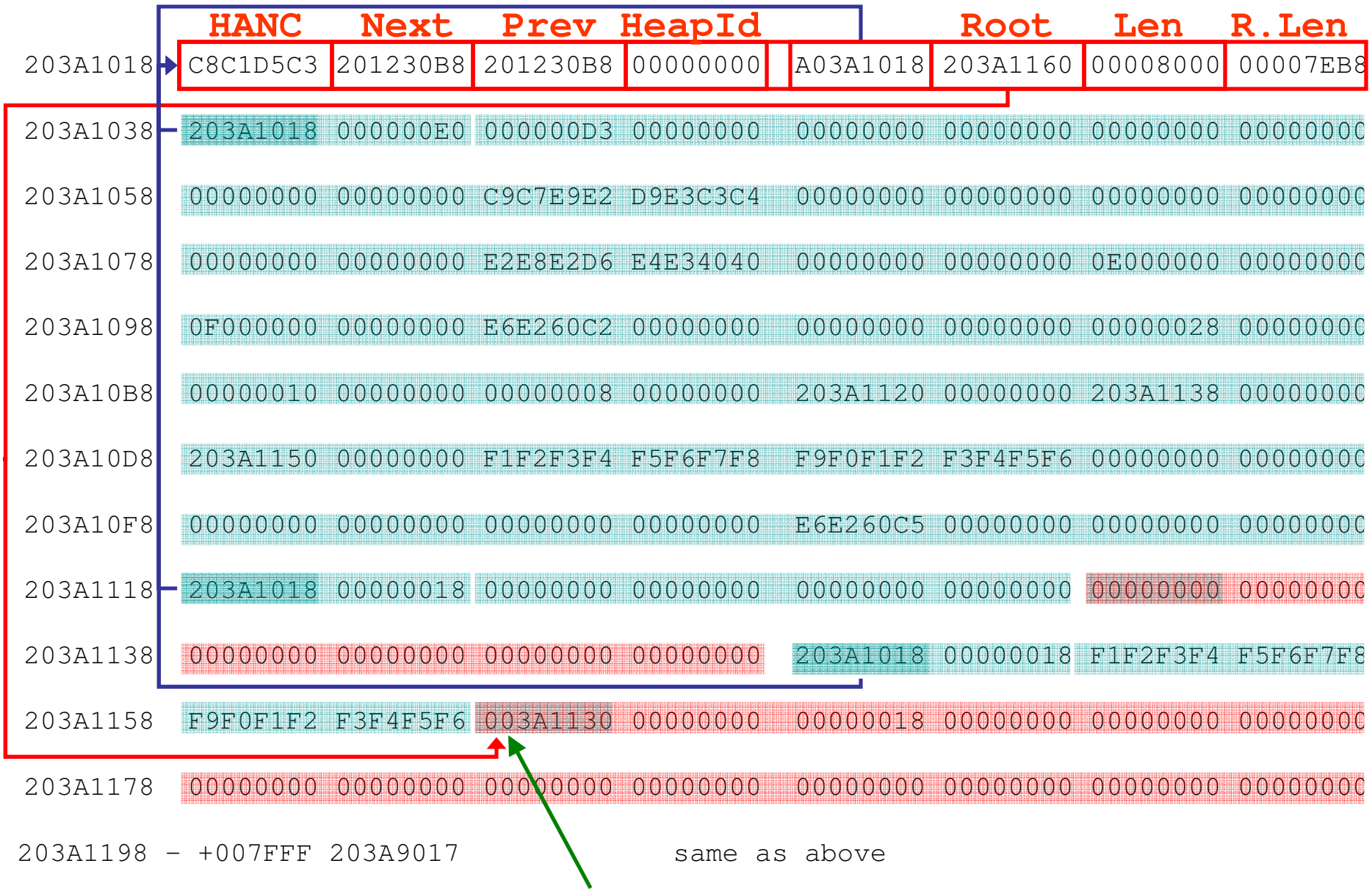
Let's cause some damage

- Looking at the damage more closely

```
Initial (User) Heap                                    : 203A1018
  +000000 203A1018  C8C1D5C3 201230B8 201230B8 00000000  A03A1018 203A1160 00008000 00007EB8 |HANC.....................-......=.|
  +000020 203A1038  203A1018 000000E0 000000D3 00000000  00000000 00000000 00000000 00000000 |...........L....................|
  +000040 203A1058  00000000 00000000 C9C7E9E2 D9E3C3C4  00000000 00000000 00000000 00000000 |........IGZSRTCD................|
  +000060 203A1078  00000000 00000000 E2E8E2D6 E4E34040  00000000 00000000 0E000000 00000000 |........SYSOUT  ................|
  +000080 203A1098  0F000000 00000000 E6E260C2 00000000  00000000 00000000 00000028 00000000 |........WS-B....................|
  +0000A0 203A10B8  00000010 00000000 00000008 00000000  203A1120 00000000 203A1138 00000000 |................................|
  +0000C0 203A10D8  203A1150 00000000 F1F2F3F4 F5F6F7F8  F9F0F1F2 F3F4F5F6 00000000 00000000 |...&....1234567890123456........|
  +0000E0 203A10F8  00000000 00000000 00000000 00000000  E6E260C5 00000000 00000000 00000000 |................WS-E............|
  +000100 203A1118  203A1018 00000018 00000000 00000000  00000000 00000000 00000000 00000000 |................................|
  +000120 203A1138  00000000 00000000 00000000 00000000  203A1018 00000018 F1F2F3F4 F5F6F7F8 |........................12345678|
  +000140 203A1158  F9F0F1F2 F3F4F5F6 003A1130 00000000  00000018 00000000 00000000 00000000 |90123456........................|
  +000160 203A1178  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000 |................................|
  +000180 203A1198 - +007FFF 203A9017            same as above
```

- Damage normally occurs in User Heap so check there first

```
203A1018  C8C1D5C3 201230B8 201230B8 00000000  A03A1018 203A1160 00008000 00007EB8
203A1038  203A1018 000000E0 000000D3 00000000  00000000 00000000 00000000 00000000
203A1058  00000000 00000000 C9C7E9E2 D9E3C3C4  00000000 00000000 00000000 00000000
203A1078  00000000 00000000 E2E8E2D6 E4E34040  00000000 00000000 0E000000 00000000
203A1098  0F000000 00000000 E6E260C2 00000000  00000000 00000000 00000028 00000000
203A10B8  00000010 00000000 00000008 00000000  203A1120 00000000 203A1138 00000000
203A10D8  203A1150 00000000 F1F2F3F4 F5F6F7F8  F9F0F1F2 F3F4F5F6 00000000 00000000
203A10F8  00000000 00000000 00000000 00000000  E6E260C5 00000000 00000000 00000000
203A1118  203A1018 00000018 00000000 00000000  00000000 00000000 00000000 00000000
203A1138  00000000 00000000 00000000 00000000  203A1018 00000018 F1F2F3F4 F5F6F7F8
203A1158  F9F0F1F2 F3F4F5F6 003A1130 00000000  00000018 00000000 00000000 00000000
203A1178  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
203A1198 - +007FFF 203A9017              same as above
```



Let's cause some damage

- Sample program CEEDUMP

```
Initial (User) Heap                                    : 203A1018
  +000000 203A1018  C8C1D5C3 201230B8 201230B8 00000000  A03A1018 203A1160 00008000 00007EB8  |HANC.....................-......=.|
  +000020 203A1038  203A1018 000000E0 000000D3 00000000  00000000 00000000 00000000 00000000  |...........L....................|
  +000040 203A1058  00000000 00000000 C9C7E9E2 D9E3C3C4  00000000 00000000 00000000 00000000  |........IGZSRTCD................|
  +000060 203A1078  00000000 00000000 E2E8E2D6 E4E34040  00000000 00000000 0E000000 00000000  |........SYSOUT  ................|
  +000080 203A1098  0F000000 00000000 E6E260C2 00000000  00000000 00000000 00000028 00000000  |........WS-B....................|
  +0000A0 203A10B8  00000010 00000000 00000008 00000000  203A1120 00000000 203A1138 00000000  |................................|
  +0000C0 203A10D8  203A1150 00000000 F1F2F3F4 F5F6F7F8  F9F0F1F2 F3F4F5F6 00000000 00000000  |...&....1234567890123456........|
  +0000E0 203A10F8  00000000 00000000 00000000 00000000  E6E260C5 00000000 00000000 00000000  |................WS-E............|
  +000100 203A1118  203A1018 00000018 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
  +000120 203A1138  00000000 00000000 00000000 00000000  203A1018 00000018 F1F2F3F4 F5F6F7F8  |........................12345678|
  +000140 203A1158  F9F0F1F2 F3F4F5F6 003A1130 00000000  00000018 00000000 00000000 00000000  |90123456.......................|
  +000160 203A1178  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  |................................|
  +000180 203A1198 - +007FFF 203A9017             same as above
```

- The crash resulted in the CEE0802C error.
- This is a simple case with a single small heap.
- Large heaps and/or multiple heaps are difficult to diagnose by hand.
- Problem 1!!!!

Let's cause some damage

- Problem 2 – modify the program slightly

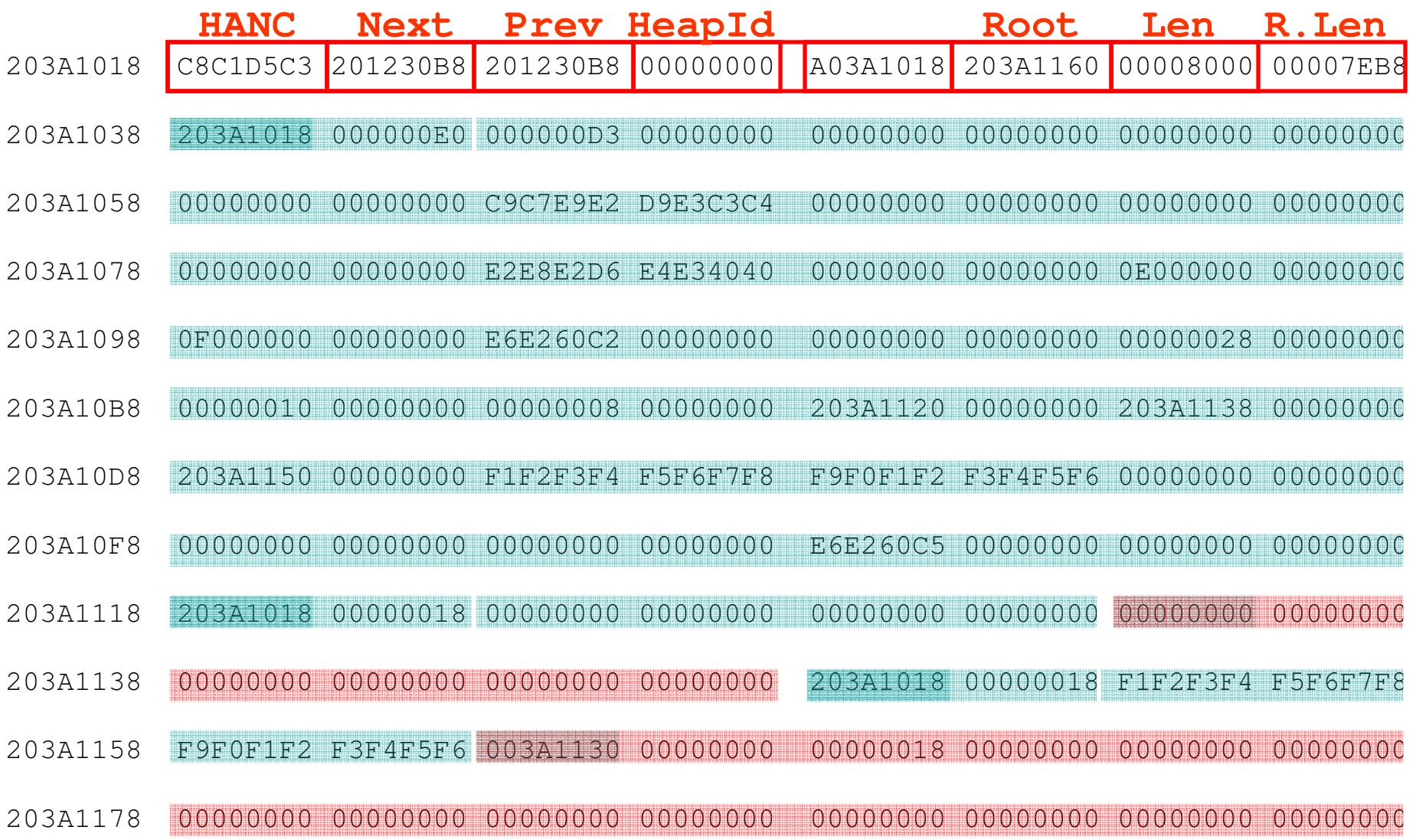
```
<snip>
004100
004200           CALL "CEEGTST" USING HEAPID, STORAGE-SIZE-16,
004300                                ADDRESS-2, FC.
004400
004500           CALL "CEEFRST" USING ADDRESS-1, FC.
004600
004700           SET ADDRESS OF HEAP-DATA-AREA TO ADDRESS-2.
004800           MOVE WS-DATA TO HEAP-DATA-AREA.
004900
004910           CALL "CEEGTST" USING HEAPID, STORAGE-SIZE-24,
004920                                ADDRESS-1, FC.
004930
005000           CALL "CEEGTST" USING HEAPID, STORAGE-SIZE-8,
005100                                ADDRESS-1, FC.
005300           CALL "CEEFRST" USING ADDRESS-0, FC.
005400           CALL "CEEFRST" USING ADDRESS-1, FC.
005500           CALL "CEEFRST" USING ADDRESS-2, FC.
005600           GOBACK.
```

Let's cause some damage

- Problem 2 – Hit and Run!

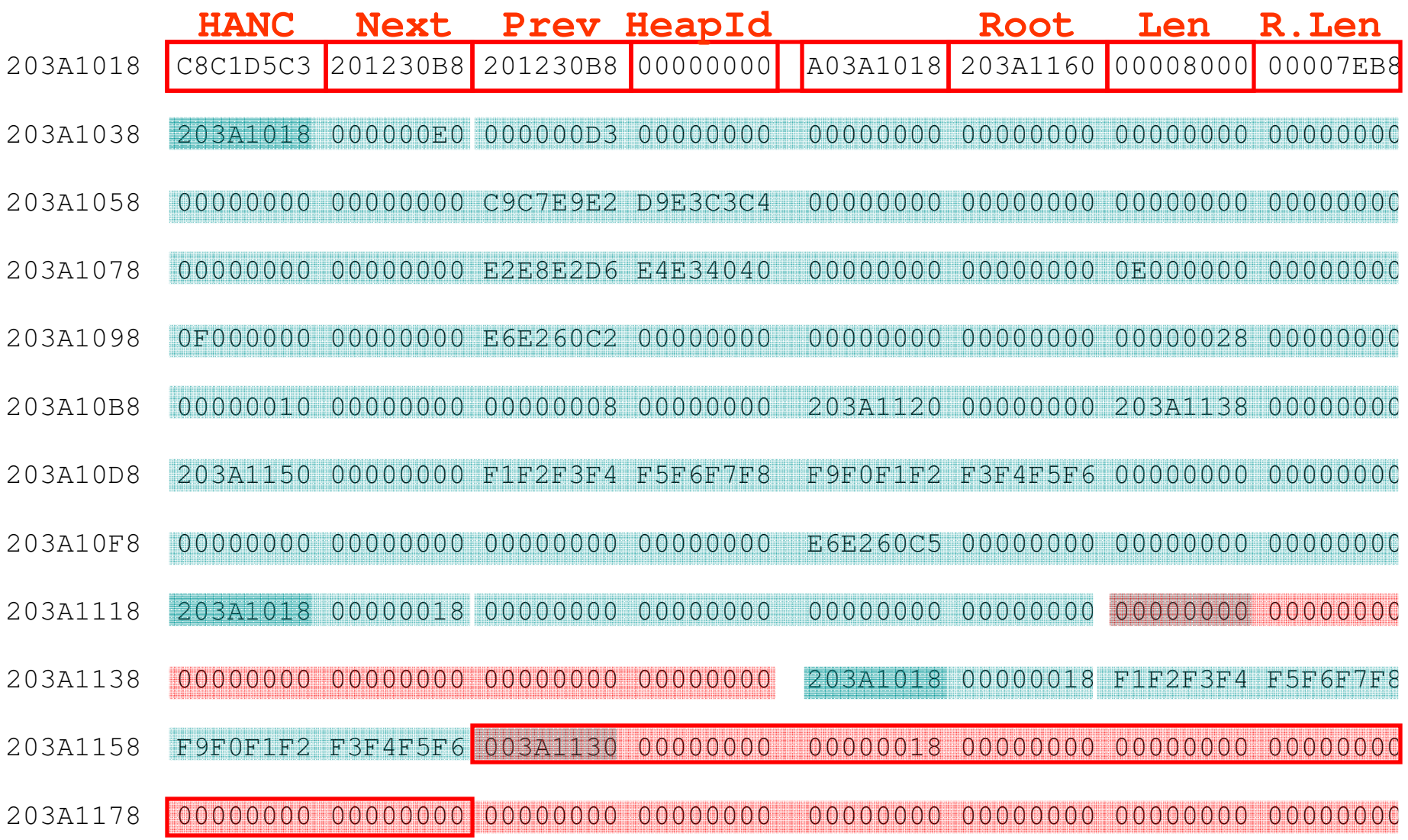
```
Initial (User) Heap                                                 : 203A1018
  +000000 203A1018  C8C1D5C3 1FF230B8 1FF230B8 00000000  A03A1018 203A1180 00008000 00007E98
  +000020 203A1038  203A1018 000000E0 000000D3 00000000  00000000 00000000 00000000 00000000
  +000040 203A1058  00000000 00000000 C9C7E9E2 D9E3C3C4  00000000 00000000 00000000 00000000
  +000060 203A1078  00000000 00000000 E2E8E2D6 E4E34040  00000000 00000000 0E000000 00000000
  +000080 203A1098  0F000000 00000000 E6E260C2 00000000  00000000 00000000 00000018 00000000
  +0000A0 203A10B8  00000010 00000000 00000008 00000000  203A1120 00000000 203A1168 00000000
  +0000C0 203A10D8  203A1150 00000000 F1F2F3F4 F5F6F7F8  F9F0F1F2 F3F4F5F6 00000000 00000000
  +0000E0 203A10F8  00000000 00000000 00000000 00000000  E6E260C5 00000000 00000000 00000000
  +000100 203A1118  203A1018 00000018 00000000 00000000  00000000 00000000 00000000 00000000
  +000120 203A1138  00000000 00000000 00000000 00000000  203A1018 00000018 F1F2F3F4 F5F6F7F8
  +000140 203A1158  F9F0F1F2 F3F4F5F6 203A1018 00000020  00000018 00000000 00000000 00000000
  +000160 203A1178  00000000 00000000 003A1130 00000000  00000018 00000000 00000000 00000000
  +000180 203A1198  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
  +0001A0 203A11B8 - +007FFF 201A9017              same as above
```

  - Same overlay occurs with element at 203A1160
  - However damage now moved to 203A1180 – HIT AND RUN!!!!
  - Why? Bigger problem!

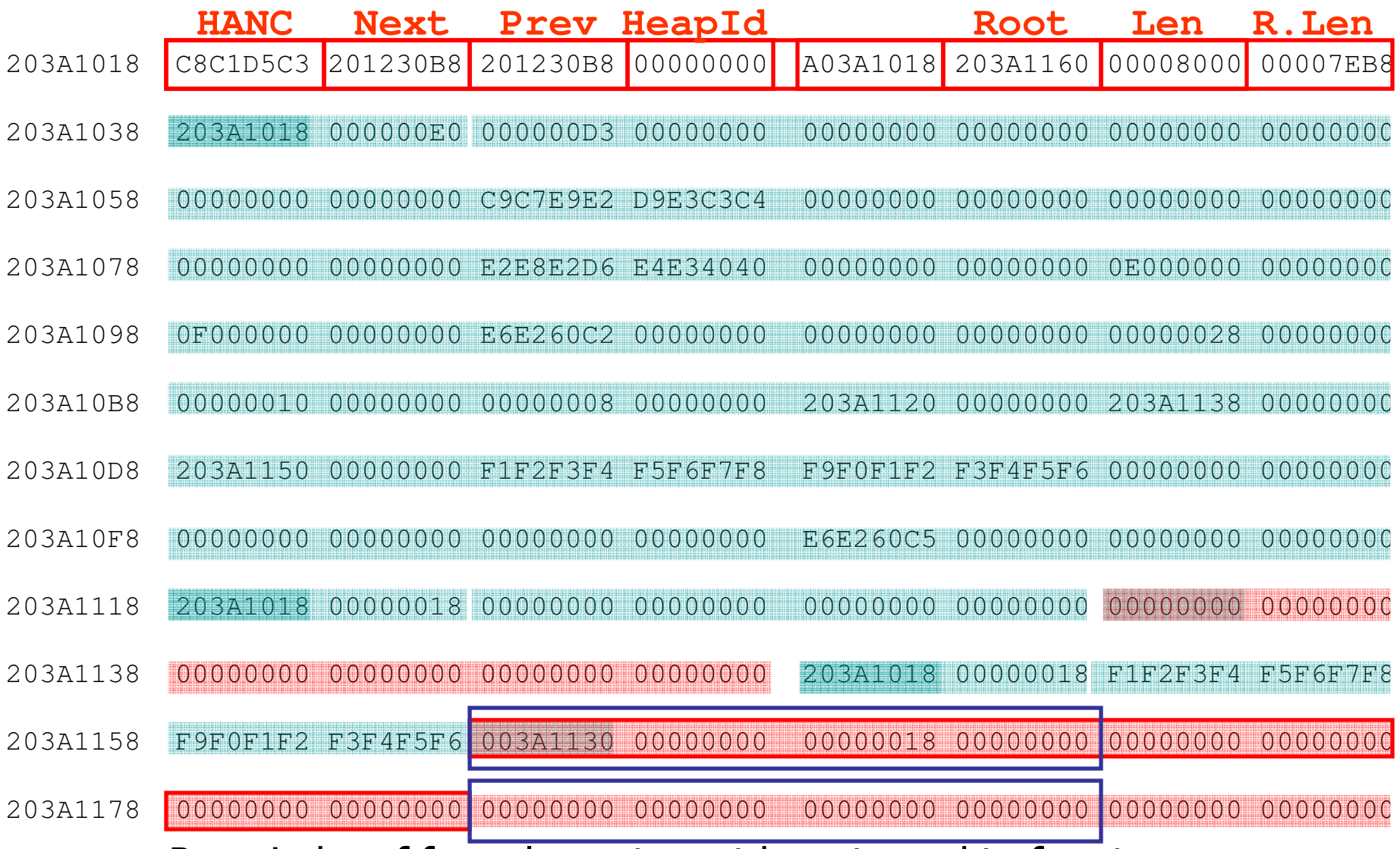


Request for 32 (x'20') bytes comes in – 24 + 8 byte header

Follow root to 203A1160 – non zero pointer – but size only x'18 – no room

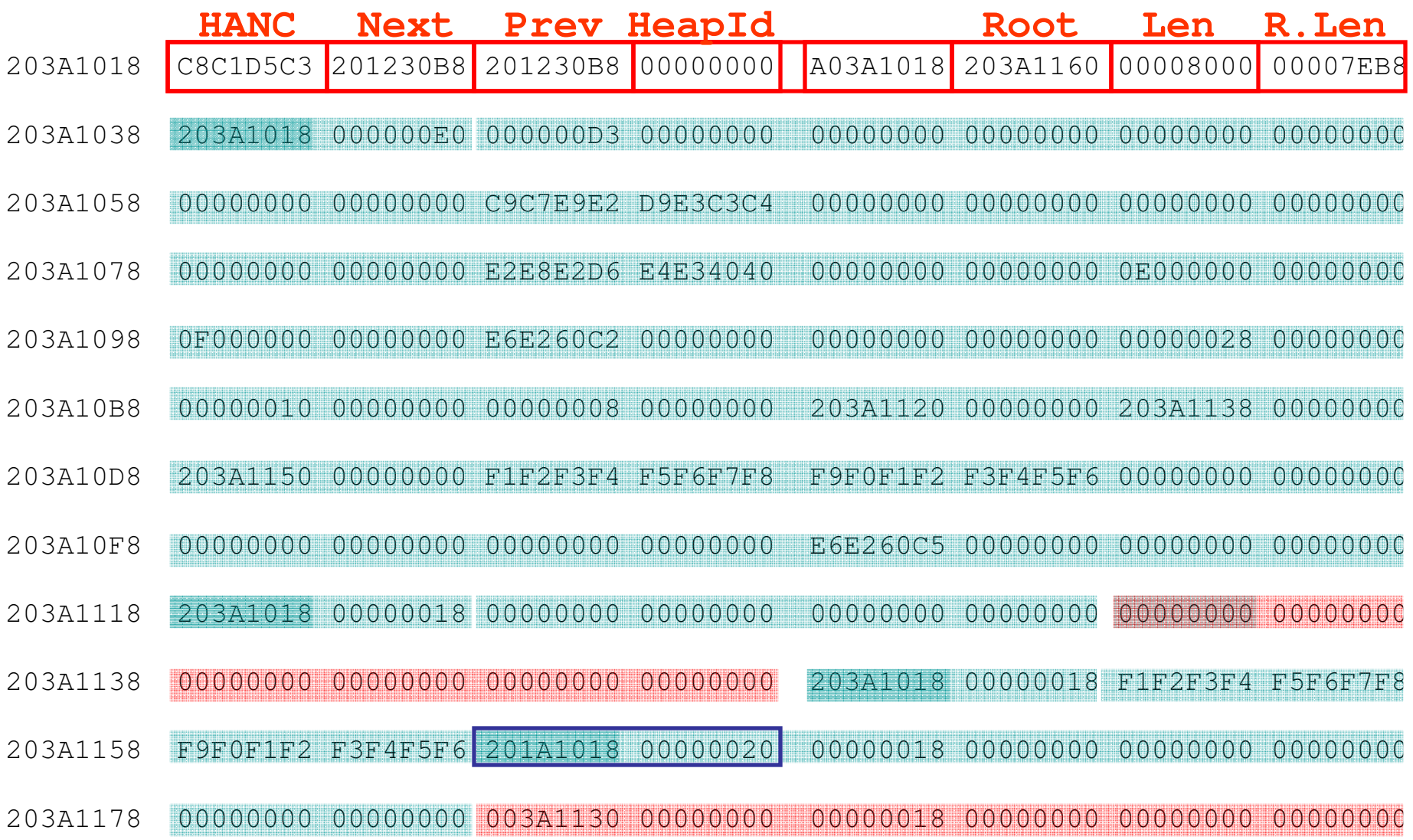


So use the first x'20' bytes for a new allocated element.



Remainder of free element must be returned to free tree.
So copy the information from existing free element.

Remainder of free element must be returned to free tree.
So copy the information from existing free element.



Now create the allocated element
Problem 2! Damage has moved

Getting closer to the zone!

- SYSMDUMP – IPCS
  - Use VERBX LEDATA 'HEAP'
    - First ENSM – Enclave Storage Management CB

```
Language Environment Product 04 V01 R06.00

Heap Storage Control Blocks

     ENSM:  201230A0
   +000000  EYE_CATCHER:ENSM  ST_HEAP_ALLOC_FLAG:00000000
   +000008  ST_HEAP_ALLOC_VAL:00000000    ST_HEAP_FREE_FLAG:00000000
   +000010  ST_HEAP_FREE_VAL:00000000     REPORT_STORAGE:00000000
   +000018  UHEAP:C8D7C3C2 203A1018 203A1018 00008000 00008000 00002000 00001000 00000000 00
   +000048  AHEAP:C8D7C3C2 2037D000 2037D000 00004000 00002000 00002000 00001000 00000000 00
   +000078  BHEAP:C8D7C3C2 20123118 20123118 00002000 00001000 00002000 00001000 80000000 00
   +0000A8  ENSM_ADDL_HEAPS:F0F00000
```

Getting closer to the zone!

- SYSMDUMP – IPCS
  - Then HPCB and header for user heap

```
User Heap Control Blocks

     HPCB:  201230B8
   +000000  EYE_CATCHER:HPCB  FIRST:203A1018    LAST:203A1018

     HANC:  203A1018
   +000000  EYE_CATCHER:HANC  NEXT:201230B8     PREV:201230B8
   +00000C  HEAPID:00000000   SEG_ADDR:A03A1018       ROOT_ADDR:203A1160
   +000018  SEG_LEN:00008000        ROOT_LEN:00007EB8

   This is the last heap segment in the current heap.
```

Getting closer to the zone!

- SYSMDUMP – IPCS
  - Next is the free tree information
    - NOTE: *ERROR* indicates an error was found

```
Free Storage Tree for Heap Segment 203A1018

          Node       Node      Parent      Left       Right      Left      Right
 Depth  Address     Length      Node       Node       Node     Length    Length
     0  203A1160  00007EB8  00000000  003A1130  00000000  00000018  00000000
*ERROR*  The left node address does not fall within the current heap segment
```

Getting closer to the zone!

- SYSMDUMP – IPCS
  - Next map all elements in heap segment

```
 Map of Heap Segment 203A1018
 To display entire segment: IP LIST 203A1018 LEN(X'00008000') ASID(X'0181')

 203A1038: Allocated storage element, length=000000E0. To display: IP LIST
203A1038 LEN(X'000000E0') ASID(X'0181')
 203A1040: 000000D3 00000000 00000000 00000000  00000000 00000000 00000000
00000000 |...L.............................|

 203A1118: Allocated storage element, length=00000018. To display: IP LIST
203A1118 LEN(X'00000018') ASID(X'0181')
 203A1120: 00000000 00000000 00000000 00000000
|................                |
```

Getting closer to the zone!

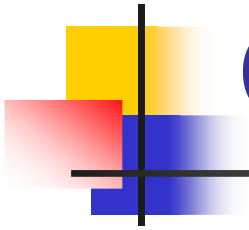
- SYSMDUMP – IPCS
  - Showing error as we go…

```
 203A1130: Allocated storage element, length=00000000. To display: IP LIST
203A1130 LEN(X'00000008') ASID(X'0181')
 203A1138: 00000000 00000000
|........                          |
*ERROR*  The heap segment address in the allocated storage element header is not
valid
WARNING  This storage element may be a free storage node not found during free
storage tree validation
*ERROR*  The length of this storage element is zero
WARNING  Attempting to identify a resume location after encountering a storage
element validation error
```

Getting closer to the zone!

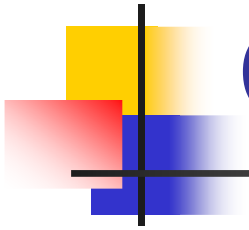
- SYSMDUMP – IPCS
  - Finally Summary information
    - Amount of allocated and free storage
    - Number of elements
    - Identifies if there were errors in this heap segment
    - Problem 1 solved!

```
Summary of analysis for Heap Segment 203A1018:
  Amounts of identified storage:  Free:00007EB8  Allocated:00000110  Total:00007FC8
  Number of identified areas   :  Free:       1  Allocated:       4  Total:       5
  00000018 bytes of storage were not accounted for.
  Errors were found while processing this heap segment.
  This is the last heap segment in the current heap.
```



Getting closer to the zone!

- HEAPCHK runtime option
  - Runtime debug tool to help diagnose a heap damage problem
  - In many cases any heap damage may not be noticed until significant time has passed – A hit and run!
  - The HEAPCHK runtime option forces all the heap segments to be validated on a regular basis.
  - Gets a dump closer to the the one at fault.
  - Generates a U4042 ABEND
    - Use System dump, if needed, to debug.



Getting closer to the zone!

- HEAPCHK runtime option ...
  - HEAPCHK(ON|OFF,freq,delay,depth,pooldepth)
    - ON - turns HEAPCHK on (performance dog)
    - OFF - normal processing
  - freq
    - Defaults to 1, indicates every call to a heap routine (get or free) validates the heap
    - Other values for less frequent checks
  - delay
    - Allows some number of calls to occur prior to 'freq' being used.
  - depth
    - Depth of traceback for storage leak
  - pooldepth
    - Depth of heappools trace

Getting closer to the zone!

- Output with HEAPCHK(ON)

```
17.12.44 JOB22865 ---- WEDNESDAY, 16 FEB 2005 ----
17.12.45 JOB22865  $HASP373 JMONTIGO STARTED - WLM INIT  - SRVCLASS WLMLONG  - SYS AQFT
17.12.45 JOB22865  IEF403I JMONTIGO - STARTED - TIME=17.12.45
17.12.48 JOB22865  IEA995I SYMPTOM DUMP OUTPUT  937
   937              USER COMPLETION CODE=4042 REASON CODE=00000000
   937             TIME=17.12.45  SEQ=11443  CPU=0000  ASID=018E
   937             PSW AT TIME OF ERROR  078D1000   A01AB242  ILC 2  INTC 0D
   937               ACTIVE LOAD MODULE          ADDRESS=201256C0  OFFSET=00085B82
   937               NAME=CEEPLPKA
   937               DATA AT PSW  201AB23C - 00181610  0A0D47F0  B10A1811
   937               AR/GR 0: 80AB5B3E/84000000   1: 00000000/84000FCA
   937                     2: 00000000/00000000   3: 00000000/00000001
  ...
   937                     E: 00000000/00000000   F: 00000000/00000000
   937              END OF SYMPTOM DUMP
17.12.48 JOB22865  IEA993I SYSMDUMP TAKEN TO POSIX.JMONTI.SHARE.HEAP.SYSMDUMP
17.12.48 JOB22865  IEF450I JMONTIGO GO - ABEND=S000 U4042 REASON=00000000  939
   939                    TIME=17.12.48
```

Getting closer to the zone!

- Very good output with HEAPCHK(ON)
  - In this case CEE37xx messages are very meaningful
  - The damage has not yet moved
  - Full debug may still need to be done with SYSMDUMP and IPCS
  - Use a storage alteration (SA) SLIP if needed (non-CICS)
  - Problem 2 Solved! ?

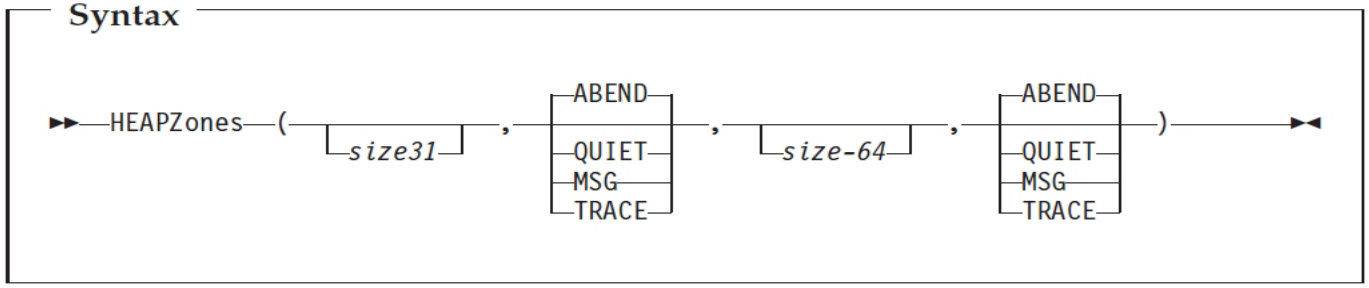
```
STARTING SHAREHP...
CEE3701W Heap damage found by HEAPCHK run-time option.
CEE3707I Left  pointer is bad in the free tree at 203A1160 in the heap
segment beginning at 203A1018.
  203A1140: 00000000 00000000 203A1018 00000018  F1F2F3F4 F5F6F7F8 F9F0F1F2
F3F4F5F6  |................1234567890123456|
  203A1160: 003A1130 00000000 00000018 00000000  00000000 00000000 00000000
00000000  |................................|
CEE3702S Program terminating due to heap damage.
```

The HEAPZONE!

- New for z/OS 2.1
  - HEAPZONES runtime option
  - Adds a check zone at the end of each allocated heap element.
  - Size of check zone is variable
  - Amount of error information is controllable
  - Zone contents validated when element is freed
    - Except for QUIET
  - Default is OFF
  - Significantly less performance overhead than HEAPCHK

The HEAPZONE!

- New for z/OS 2.1



• size31 – Indicates size of check zone. Value (0-1024) rounded up to a double word. 0 indicates HEAPZONES support is off.

• size64 – Indicates size of check zone for above the bar storage. Value (0-1024) rounded up to a double word. 0 indicates HEAPZONES support is off

• QUIET – indicates overlays are tolerated and not checked.

The HEAPZONE!

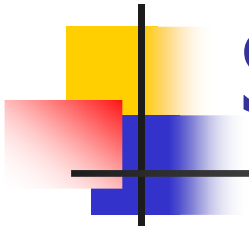
- Sample
  - Update the program with a free of the overlaid element.
  - HEAPZONES(8,ABEND)

```
- ================================================================================================
-                                    REGION        --- STEP TIMINGS ---                    ----
- STEPNAME PROCSTEP PGMNAME     CC     USED      CPU TIME  ELAPSED TIME    EXCP     SERV  PAGE
- STEP1    COBOL    IGYCRCTL    00    2048K    0:00:00.02    0:00:00.23     665     1444     0
- STEP1    LKED     HEWL        00      92K    0:00:00.01    0:00:00.10     215      726     0
+CEE3798I ATTEMPTING TO TAKE A DUMP FOR ABEND U4042 TO DATA SET: JMONTI.D220.T1744175.JMONTI@B
IGD101I SMS ALLOCATED TO DDNAME (SYS00001)  302
        DSN (JMONTI.D220.T1744175.JMONTI@B               )
        STORCLAS (STANDARD) MGMTCLAS (MIGONLY) DATACLAS (        )
        VOL SER NOS= SL101A
IGD104I JMONTI.D220.T1744175.JMONTI@B                RETAINED,  DDNAME=SYS00001
IEA822I COMPLETE TRANSACTION DUMP WRITTEN TO JMONTI.D220.T1744175.JMONTI@B
+CEE3797I LANGUAGE ENVIRONMENT HAS DYNAMICALLY CREATED A DUMP.
```

The HEAPZONE!

- Sample
  - Messages
    - Address and storage of element being freed
    - You can see the ’00’ overlay in the check zone
    - Offset indicates location of the “free” which should identify the owner of the storage

```
CEE3716I The heap check zone following the storage at address 214E0168 for length X'00000010' has been
overlayed at address 214E0178. Each byte in the zone from 214E0178 to 214E017C should contain the value
X'55'. From compile unit SHAREHP at entry point SHAREHP at statement 899 at compile unit offset +00000630 at
entry offset +00000630 at address 21200630.
  214E0168: F1F2F3F4 F5F6F7F8 F9F0F1F2 F3F4F5F6                    |1234567890123456|
  214E0178: 00555555 214E0178                                      |.....+..        |
```



Summary

- Summary for heaps
  - Heap used for dynamic storage
  - CEEDUMPs contain information on heap errors but they are difficult to find
  - SYSTEM DUMPs using LEDATA 'HEAP' make debug much simpler
  - Use HEAPCHK runtime option to debug
    - Big performance hit
  - Use HEAPZONES as a fast alternative!

Sources of Additional Info

- All Language Environment documentation available on the z/OS DVD collection and on the Language Environment website
  - Language Environment Debug Guide
  - Language Environment Runtime Messages
  - Language Environment Programming Reference
  - Language Environment Programming Guide
  - Language Environment Customization
  - Language Environment Migration Guide
  - Language Environment Writing ILC Applications
- Language Environment Web site
  - http://www-03.ibm.com/systems/z/os/zos/features/lang_environment/