News from Nan

COMPUTER HISTORY

MY HIGH-TECH ADVENTURE: CHAPTER 09, CODE SAMPLES

OCTOBER 10, 2018 | DICK

Dick wrote this post, which was part of a book, My high-tech adventure, *to record his memories of working in the computer industry, starting in about 1960. It was first published in 2012.*

In this chapter I have included fragments of code I wrote myself on various projects. The client names and programming languages are included in the titles.

Programming languages

During all the time I worked in computing, the main thing I was working to produce was code in a computer language. Some of languages are still in use, and some are now close to being dead languages. To succeed in the computer business, you have to be learning new things constantly!

Languages I have learned and used include:

- Apple II BASIC/BASICA
- Apple 6502 Assembler
- C
- DITA XML
- Fortran II
- Fortran IV
- IBM 1401 Autocoder
- IBM 7094 Assembler
- IBM PLAS
- IBM System 360/370 Assembler
- Intel 8088 Assembler
- JAVA
- PHP
- Python

- PL/I
- UNIX Shell
- VM EXEC/EXEC2
- VM REXX
- WYLBUR EXEC
- XHTML
- XSLT

I spent the most time during my career coding in IBM 360/370 Assembler and later C. These days my language of choice is Python.

ADSM device driver (IBM, C)

```
899     /*
900      * Call the table lookup routine to find the proper recovery action
901      * for the sense bytes from the error.
902      *
903      * Note:  Occasionally, it is necessary to use the last executed SCSI
904      * command in addition to the sense bytes when determining which
905      * action to perform for an error.  This would be the case for devices
906      * which require that different actions be taken for the same sense
907      * code combination when received in response to different SCSI
908      * commands.  For this reason, the operation code of the last
909      * executed SCSI command will be passed to erpfind.
910      */
911
912     recovery_action = erpfind(sense_bytes, recov_table_p, table_width,
913                               scsi_command);
914
915
916     /*
917      * Check that the table lookup found a recovery action.
918      * (If one is not found, there is something wrong with the table!)
919      */
920
921     if (recovery_action == NO_ACTION) {
922
923       erp_rc              = ERP_LOGIC_ERROR;
924       scsiout_p->compcode = DD_DEVICE_DRIVER_FAILURE;
925
926     } else {
927
928       /*
929        * Call the error recovery action processor to carry out the
930        * recovery action called for in the previous table lookup.
931        */
932
933       erp_rc = erpact(recovery_action, scsi_device_type, scsi_command,
934                       dev_p,       helper,           scsiout_p);
935     }
936
937     /*
938      * Return to caller with proper return code set.
939      */
940
941     DD_EXIT_SRC("erp_main: ", erp_rc);
942     return erp_rc;
943
944   } /* end erp_main() */
945
```

ADSM device driver

VM/CMS internal trace (SLAC, IBM/370 Assembler)

```
*
* MAIN ENTRY POINT WHEN WE ARE INVOKED VIA SVC 202.
* WE COULD BE RUNNING AS A MODULE OR AS A NUCLEUS
* EXTENSION.
*
BASEADDR LR    R12,R15            ADDRESSING STUFF
         LA    R13,2048(,R12)
         LA    R13,2048(,R13)
         LA    R11,2048(,R13)
         LA    R11,2048(,R11)
         USING BASEADDR,R12,R13,R11
         USING NUCON,R0
         ST    R14,REG14          SAVE RETURN
         LA    R1,8(,R1)          ANY OPERANDS?
         CLC   0(8,R1),=CL8'XYZZY' GOT MAGIC WORD?
         BE    GOFORIT            PROCEED IF SO
         CLC   0(8,R1),=CL8'RESET' NUCX GOODBYE CALL?
         BE    NUCXDROP           PROCEED IF SO
IGNORE   LA    R15,100            IGNORE ANYTHING ELSE
         BR    R14
         EJECT
*
*        NUCXDROP IS KILLING US, UNHOOK AND GO AWAY.
*
NUCXDROP DS    0H
*
         ENABLE INTTYPE=NONE      DISABLE INT
         MVC   SVCNPSW(8),OLDSPSW PUT BACK PSW
         MVC   EXTNPSW(8),OLDEPSW PUT BACK PSW
         MVC   IONPSW(8),OLDIPSW PUT BACK PSW
         MVC   ADMSITI(4),IONPSW+4 DMSITE CHECKS THIS FOR BLIP
         MVC   PGMNPSW(8),OLDPPSW PUT BACK PSW
         CLI   BALRFLAG,X'00'     TRACING BALRS?
         BNE   NUCXDRP2           NO, SKIP NEXT STUFF
*
         MVC   ARDBUF(4),OLDRDBUF PUT BACK POINTER TO RDBUF
         MVC   AWRBUF(4),OLDWRBUF PUT BACK POINTER TO WRBUF
         MVC   ARDTK(4),OLDDIOR   PUT BACK DIOR POINTER
         MVC   AWRTK(4),OLDDIOW   PUT BACK DIOW POINTER
         MVC   AUPDISK(4),OLDAUD  PUT BACK AUD POINTER
         MVC   AESTATE(4),OLDSTATE PUT BACK STATE POINTER
         MVC   ADMSFREB(4),OLDFREB PUT BACK FREB POINTER
*
         L     R15,NUCCSLBK       POINT TO DMSCSLBK IN NUCON
         USING CSLBK,R15          MAP IT
         MVC   CSLACCR(4),OLDCSL  PUT BACK CSL POINTER
         DROP  R15
*
NUCXDRP2 ENABLE INTTYPE=ALL       REENABLE EVERYTHING
         SR    R4,R4              ZERO WORK AREA POINTER
         ST    R4,USERLVL         USED BY CTRACE
         CLI   SPOOLFLG,X'00'     SPOOLING ACTIVE?
         BNE   NUCXDRP3           NO, SKIP NEXT COMMANDS
         L     R4,NCCW            ANYTHING IN BUFFER STILL?
         LTR   R4,R4              ...
         BZ    NUCXCPRT           NO, JUST CLOSE SPOOL FILE
* EMPTY OUT SPOOL BUFFER BEFORE CLOSING SPOOL FILE
         S     R4,=F'1'           CCW COUNT-1
         MH    R4,=H'8'           TIME LENGTH OF A CCW
         LA    R5,PRTCCW          ADDRESS OF START OF CHAIN
         AR    R5,R4              ==>FLAG IN LAST CCW
         MVI   4(R5),X'20'        TURN OFF CHAINING
*
```

VM/CMS internal trace

Multi-pathing device driver (IBM, C)

```
135         /* We have multiple paths to chose from, use appropriate algorithm
136            to pick one. */
137
138         /* cannot used load sensitive algorithm if no counters exist */
139         path_algorithm = dev_p->policy;
140         if ( (path_algorithm == DD_POLICY_OPTIMIZED) &&
141              !counter_flag )
142           path_algorithm = DD_POLICY_ROUND_ROBIN;
143
144         switch ( path_algorithm )
145         {
146           case DD_POLICY_ROUND_ROBIN:
147
148           path_alive      = DD_PATH_NONE;
149           path_alive_same = DD_PATH_NONE;
150           nnl = 0;
151
152           /*
153            * Scan the paths looking for various types of paths for this device:
154            *
155            *  1. A path other than the last one used for this device.
156            *  2. A path for this device that is usable.
157            *
158            */
159
160           for(i=0 ; i<dev_p->path_count; i++)
161           {
162             /* optional detailed trace for grouping */
163             if( dev_p->controller_flag )
164             {
165               DU_DEBUG_S4("path %d defcntrl %d defcnt %d pathcntrl %d",
166                 i, dev_p->default_controller, dev_p->default_path_count,
167                 dev_p->path[i].controller_number);
168             }
169             /* never touch closed path */
170             if(( dev_p->path[i].state == DD_PATH_CLOSED ) ||
171                ( dev_p->path[i].state == DD_PATH_INVALID ))
172               continue;
173
174             if( path_alive == DD_PATH_NONE )
175             {
176               /* In case ALL paths are currently being bypassed, we will
177                  use this one anyway. It is at least not dead yet.   */
178               if( dev_p->path[i].state == DD_PATH_OPEN )
179               {
180                 /* possible path if no suitable path found later */
181                 path_alive = i;
```

IBM Data Path Optimizer device driver code fragment

AIX/ESA disk device driver (IBM, C)

```
        /* Also, we only allow I/O in multiples of disk block size */
        if (bp->b_bcount & (DEV_BSIZE - 1)) {
                debug(CKDDBG_B,("ckdstrategy: bcount not multiple of DEV_BSIZE.\n"));
                bp->b_error = EINVAL;
                bp->b_flags |= B_ERROR;
                bp->b_resid = bp->b_bcount;
                goto unlock_error_exit;
        }
        /* Make sure the 1st block is inside the partition */
        if (bp->b_blkno < 0 || bp->b_blkno >= pp->p_size )
                {

                /*
                 *  bio readahead always tries to read one extra block
                 */
                if ((bp->b_blkno >= pp->p_size)
                                && ((bp->b_flags & B_READ) == B_READ)) {
                        debug(CKDDBG_B, ("ckdstrategy: blk out of range(2)\n"));
                        bp->b_resid = bp->b_bcount;
                        goto unlock_error_exit;
                }
                debug(CKDDBG_B, ("ckdstrategy: blk out of range(1)\n"));
                bp->b_resid = bp->b_bcount;
                bp->b_error = EINVAL;
                bp->b_flags |= B_ERROR;
                goto unlock_error_exit;
        }
        /* If the ending block extends beyond the end of the partition,
           truncate back to the end of the partition. */
        if (bp->b_blkno + nblks > pp->p_size) {
                debug(CKDDBG_B, ("ckdstrategy: bcount truncated.\n"));
                bp->b_bcount = (pp->p_size - bp->b_blkno)<<DEV_BSHIFT;
        }

        /* Set b_cylin (b_resid) for disksort */
        bp->b_cylin = bp->b_blkoff / fi->fi_disklabel.d_secpercyl;

        /* queue lock fast path */
         fi->fi_stats.fi_qlock_strategy++; /* count fast path */
         disksort(&fi->fi_tab,bp); /* sort the requests  */
         ufcnt++;
```

IBM AIX/ESA DASD driver code fragment

PC DOS trace (SLAC, Intel 8088 Assembler)

```
CODE_START:
; BASIC INITIALIZATION STUFF
          MOV     AX,CS         ; FIX UP DS
          MOV     DS,AX         ; ...
          MOV     SS,AX         ; AND SS ALSO
          LEA     AX,BEGSTACK   ; SET SP
          MOV     SP,AX         ; ...
;
          MOV     AL,OUTINT     ; TRACING ALREADY ACTIVE?
          MOV     AH,35H        ; GET CURRENT VECTOR
          INT     DOSINT
          MOV     AX,ES         ; POINT ANYWHERE?
          CMP     AX,0
          JE      TSTART
          CMP     BX,0
          JE      TSTART
;
          LEA     DX,SIGNOFF    ; TYPE ABORT MESSAGE
          MOV     AH,9
          INT     DOSINT
          MOV     AL,16         ; SET NON-ZERO RC
          MOV     AH,4CH        ; TERMINATE PROCESS
          INT     DOSINT
;
TSTART:   LEA     AX,BEGTRACE   ; INIT TRACE TABLE POINTER
          MOV     POINT,AX
          MOV     AL,YES        ; TURN ON TRACING
          MOV     TFLAG,AL
          MOV     AL,NO         ; NOT WRAPPED YET
          MOV     WRAP,AL
;
          MOV     AX,CS         ; CONVERT LOAD ADDR FOR PRINTING
          LEA     SI,LOADCS
          CALL    HEXPRT
          MOV     AX,100H
          LEA     SI,LOADIP
          CALL    HEXPRT
;
          LEA     DX,SIGNON     ; TYPE SIGNON MESSAGE
          MOV     AH,9
          INT     DOSINT
;
; SET INT FOR PRODUCING OUTPUT
;
          LEA     DX,OUTPUT     ; ADDRESS OF ROUTINE
          MOV     AL,OUTINT     ; INT NUMBER
          MOV     AH,25H        ; SET INTERRUPT VECTOR
          INT     DOSINT        ; HAVE DOS PLACE IT
;
; Set INT intercepts
```

PC DOS internal trace code fragment

VM/CMS profile exec (SLAC, IBM REXX)

```
/*
  PROFILE - DJOHNSON PROFILE EXEC.

  REXX version, 10/01/84
                                  */
trace 'o';Address command;yes='Y';no='N'

/*============== Query   Environment ==================*/

'DISKADRM S'  /* IBMCMS? */
pull . . slabel .
 if slabel='IBMCMS' then 'EXEC SLACINIT'

'CPSTACK QUERY CPUID'   /* Display where I am running */
pull . . cpuid .;machine=substr(cpuid,9,4)

say "You are on the" machine

batch='NO';'CMSINFO BATCH';if rc>0 then batch='YES'
if batch='YES' then say "PROFILE running in batch."

'CPSTACK QUERY CPLEVEL' /* Display time/date of last IPL */
pull .;pull .;pull ipldata
say ipldata

'QUERY CMSLEVEL(STACK LIFO' /*what level of CMS?*/
pull . . cmslevel ',' .
say "Running in SP"cmslevel

'CP .VAR Q LOGON'  /* Is this an IPL or a LOGON */
if rc=0 then logged=yes; else logged=no

if logged=no then do /* set my logon variable for next time */
                  'CP .VAR SET LOGON Logon:' date() 'at' time()
                  end

'QCONSOLE';pull . con . /* get console state */

/*============== Set up  Environment ==================*/

'CP .SET ACNT OFF';'CP .SET EMSG ON'

'EXEC MYDISK' /* set disk config */

'DISKADRM P'  /* got SYSTEMS? */
if rc=0 then pull junk
 else do /* put it there */
      'SET CMSTYPE HT'
      'EXEC GIME SYSTEMS (QUIET REP'
      'SET CMSTYPE RT'
      end
```

IBM REXX code fragment

VM/CMS XEDIT macro (SLAC, IBM EXEC2)

```
 1  &TRACE OFF
 2  &BUFFER *
 3  &FOUNDIN = &STRING OF Found in:
 4  PRESERVE
 5  SET WRAP OFF
 6  &CASE M
 7  &TARG = &ARGSTRING
 8  SET MSGMODE OFF
 9  &FOUND = 0
10  TRANSFER SIZE LINE
11  &READ VARS &LASTL &WASAT
12  &IF &N LT 1 &GOTO -TELL
13  &IF .&1 EQ .? &GOTO -TELL
14  TOP
15   &LOOP -LOOP *
16  CL /&TARG
17  &IF &RC NE 0 &GOTO -DONE
18  &FOUND = &FOUND + 1
19  TRANSFER LINE
20  &READ VARS &L
21  &FOUNDIN2 = &CONCAT OF &FOUNDIN &L &BLANK
22  &LENFOUND = &LENGTH OF &FOUNDIN2
23  &IF &LENFOUND LT 80 &FOUNDIN = &FOUNDIN2
24  &WL = &CONCAT OF &L .
25  &WL = &CONCAT OF 00000 &WL
26  &WL = &RIGHT OF &WL 6
27  STACK
28  &READ STRING &DATA
29  &TYPE &WL &DATA
30  CL *
31   -LOOP
32  -DONE
33  &IF &FOUND GT 0 &GOTO -WASFOUND
34  &ESTR = &CONCAT OF ' &TARG '
35  SET MSGMODE ON
36  EMSG &ESTR not found in file.
37  &IF &WASAT = 0 TOP
38  &IF &WASAT > 0 :&WASAT
39  -WASFOUND
40  RESTORE
41  &IF &FOUND GT 0 EMSG &FOUNDIN
42  &EXIT
43  -TELL
44  SET MSGMODE ON
45  &TYPE Format: DJOTYPE string
46  &TYPE
47  &TYPE Type all lines containing "string"
48  &TYPE
49  &EXIT 100
```

IBM VM EXEC2 code fragment

DITA/XML (VR Communications, XML)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN" "../dtd/concept.dtd">
<concept xml:lang="en-us" id="code_ditaxml">
    <title>VM/CMS ditaxml macro (SLAC, IBM EXEC2)</title>
    <shortdesc id="shortdesc"/>
    <prolog>
        <author type="creator">Richard Johnson</author>
        <author type="contributor">Anna van Raaphorst</author>
        <publisher>VR Communications, Inc.</publisher>
        <copyright>
            <copyryear year="2003"/>
            <copyrholder>VR Communications, Inc.</copyrholder>
        </copyright>
        <critdates>
            <created date="2012-Mar-26"/>
            <revised modified="2012-Mar-26"/>
        </critdates>
        <metadata>
            <keywords>
                <keyword>code_sample</keyword>
                <keyword>DITA XML</keyword>
                <indexterm>code samples
                <indexterm>DITA XML</indexterm>
                </indexterm>
                <indexterm>DITA XML code sample</indexterm>
            </keywords>
        </metadata>
    </prolog>
    <conbody>
        <section>
            <p>
                <image href="../images/code_ditaxml.jpg"/>
            </p>
        </section>
    </conbody>
</concept>
```

DITA XML code fragment

Python code (VR Communications – Pearson, Python)

new_password_advanced.py

```
for i in range(tries):
    #
    # get the user's input
    #
    # prompt for the new password
    password = str(input("Enter new password: "))

    #
    # test the conditions and report errors
    #
    # test the password length
    if len(password) >= minlength:
        # test for all alphanumeric
        if password.isalnum():
            # make sure  there is at least one letter
            if not password.isdigit():
                # make sure there is at least one number
                if not password.isalpha():
                    # password passes all tests, mark it valid
                    valid = True
                    # if the password is valid after 1 or 2 tries
                    # break out of this section
                    # and proceed to the "cleanup" section
                    break
                else:
                    # there must be at least one digit in the password
                    print("Error, password does not contain a numeric digit.")
            else:
                # there must be at least one letter in the password
                print("Error, password does not contain a letter.")
        else:
            # password contains an invalid character
            print("Error, password is not alphanumeric.")
    else:
        # password was not long enough
        print("Error, password is less than",minlength,"characters.")
```

Python code fragment

2018 ADSM DEVICE DRIVER ADSTAR DISTRIBUTED STORAGE MANAGER (ADSM) AIX/ESA DISK DEVICE DRIVER C PROGRAMMING LANGUAGE DITA/XML IBM EXEC2 PROGRAMMING LANGUAGE IBM REXX PROGRAMMING LANGUAGE IBM/370 ASSEMBLER PROGRAMMING LANGUAGE INTEL 8088 ASSEMBLER PROGRAMMING LANGUAGE MULTI-PATHING DEVICE DRIVER MYHITECH PC DOC TRACE PYTHON PROGRAMMING LANGUAGE VM/CMS INTERNAL TRACE VM/CMS PROFILE EXEC VM/CMS XEDIT MACRO XML PROGRAMMING LANGUAGE