(https://peeterjoot.com)Utilisation du débogueur pour comprendre la sémantique COBOL niveau 88.

10 août 2020 COBOL (https://peeterjoot.com/category/mainframe/cobol/)
COBOL (https://peeterjoot.com/tag/cobol/) , NIVEAU 88 (https://peeterjoot.com/tag/level-88/) , lldb (https://peeterjoot.com/tag/lldb/)

COBOL dispose d'un mécanisme d'énumération appelé variable LEVEL-88. J'ai trouvé certains aspects de ce mécanisme contre-intuitifs, comme je l'ai mentionné dans quelques articles précédents. Avec l'aide du débogueur, je pense maintenant avoir enfin compris ce qui se passe. Voici un exemple de programme qui utilise quelques variables LEVEL-88 :

TEST.cob

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. TEST.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
       DATA DIVISION.
       FILE SECTION.
       WORKING-STORAGE SECTION.
       01  FEEDBACK.
           02  CONDITION-TOKEN-VALUE.
           88 MY88-VAR-1  VALUE X"0000000141C33A3B".
           88 MY88-VAR-2  VALUE X"0000000241C33A3B".
               03  CASE-1-CONDITION-ID.
                   04  SEVERITY     PIC S9(4) BINARY.
                   04  MSG-NO       PIC S9(4) BINARY.
               03  CASE-SEV-CTL-0.
                   04  CASE-SEV-CTL    PIC X.
                   04  FACILITY-ID     PIC XXX.
           02  I-S-INFO            PIC S9(9) BINARY.
       PROCEDURE DIVISION.
           MOVE SPACES TO FEEDBACK

           SET MY88-VAR-1 TO True

           IF MY88-VAR-2 OF feedback THEN
              PERFORM IMPOSSIBLE
           END-IF

           IF MY88-VAR-1 THEN
              PERFORM EXPECTED
           END-IF

           SET MY88-VAR-2 TO TRUE

           IF MY88-VAR-1 OF FeedBack THEN
              PERFORM IMPOSSIBLE
           END-IF

           IF MY88-VAR-2 THEN
              PERFORM EXPECTED
           END-IF

           STOP RUN
           .

       Impossible.
           DISPLAY 'Should not get here!'

           STOP RUN
           .

       Expected.
           DISPLAY 'This condition was expected.'
           .
```

(https://peeterjoot.com/wp-content/uploads/2020/08/Screen-Shot-2020-08-10-at-11.28.50-AM.png)

Nous pouvons utiliser un débogueur pour découvrir la signification du code COBOL. Commençons par parcourir pas à pas la première instruction MOVE jusqu'à juste avant le SET MY88-VAR-1 :

```
(lldb) n
Process 2462 stopped
* thread #13, name = 'LZ000550', stop reason = step over
    frame #0: 0x00007ffff7fdf7b8 COBRC.NATIVE.LZ000550(LZ000550).03e2f998`TEST at TEST.cob:21:1
   18                    04  FACILITY-ID     PIC XXX.
   19               02  I-S-INFO            PIC S9(9) BINARY.
   20           PROCEDURE DIVISION.
-> 21    _           MOVE SPACES TO FEEDBACK
   22
   23               SET MY88-VAR-1 TO True
   24
(lldb)
Process 2462 stopped
* thread #13, name = 'LZ000550', stop reason = step over
    frame #0: 0x00007ffff7fdf81d COBRC.NATIVE.LZ000550(LZ000550).03e2f998`TEST at TEST.cob:23:1
   20           PROCEDURE DIVISION.
   21               MOVE SPACES TO FEEDBACK
   22
-> 23    _          SET MY88-VAR-1 TO True
   24
   25               IF MY88-VAR-2 OF feedback THEN
   26                  PERFORM IMPOSSIBLE
```

(https://peeterjoot.com/wp-content/uploads/2020/08/Screen-Shot-2020-08-10-at-11.45.38-AM.png)

Ici, j'exécute le programme LZ000550 à partir d'un PDS nommé COBRC.NATIVE.LZ000550. Nous nous attendons à des espaces EBCDIC ('\x40') dans la structure FEEDBACK à ce stade et c'est ce que nous voyons :

```
(lldb) fr v -format x -L FeedBack
0x00007ffba6846908: (FEEDBACK) FEEDBACK = {
0x00007ffba6846908:   CONDITION-TOKEN-VALUE = {
0x00007ffba6846908:     CASE-1-CONDITION-ID = {
0x00007ffba6846908:       SEVERITY = 0x4040
0x00007ffba684690a:       MSG-NO = 0x4040
    }
0x00007ffba684690c:     CASE-SEV-CTL-0 = {
0x00007ffba684690c:       CASE-SEV-CTL = {
0x00007ffba684690c:         [0] = 0x40
      }
0x00007ffba684690d:       FACILITY-ID = {
0x00007ffba684690d:         [0] = 0x40
0x00007ffba684690e:         [1] = 0x40
0x00007ffba684690f:         [2] = 0x40
      }
    }
  }
0x00007ffba6846910:   I-S-INFO = 0x40404040
}
```

(https://peeterjoot.com/wp-content/uploads/2020/08/Screen-Shot-2020-08-10-at-11.48.59-AM.png)

En passant par l'instruction SET, notre structure de mémoire ressemble maintenant à ceci :

```
(lldb) n
Process 2462 stopped
* thread #13, name = 'LZ000550', stop reason = step over
    frame #0: 0x00007ffff7fdf879 COBRC.NATIVE.LZ000550(LZ000550).03e2f998`TEST at TEST.cob:25:1
   22
   23              SET MY88-VAR-1 TO True
   24
-> 25   _          IF MY88-VAR-2 OF feedback THEN
   26                 PERFORM IMPOSSIBLE
   27              END-IF
   28
(lldb) fr v -format x -L FEEDBACK
0x00007ffba6846908: (FEEDBACK) FEEDBACK = {
0x00007ffba6846908:   CONDITION-TOKEN-VALUE = {
0x00007ffba6846908:     CASE-1-CONDITION-ID = {
0x00007ffba6846908:       SEVERITY = 0x0000
0x00007ffba684690a:       MSG-NO = 0x0001
    }
0x00007ffba684690c:     CASE-SEV-CTL-0 = {
0x00007ffba684690c:       CASE-SEV-CTL = {
0x00007ffba684690c:         [0] = 0x41
      }
0x00007ffba684690d:       FACILITY-ID = {
0x00007ffba684690d:         [0] = 0xc3
0x00007ffba684690e:         [1] = 0x3a
0x00007ffba684690f:         [2] = 0x3b
      }
    }
  }
0x00007ffba6846910:   I-S-INFO = 0x40404040
}
```

(https://peeterjoot.com/wp-content/uploads/2020/08/Screen-Shot-2020-08-10-at-11.51.30-AM.png)

En définissant la variable LEVEL-88 sur TRUE, toute la mémoire commençant à l'adresse de feedback est désormais écrasée par la valeur numérique 0x0000000141C33A3BL. Si nous continuons le programme, le SYSOUT finit par ressembler à ceci :

```
</>
```

```
Cette condition était attendue.
Cette condition était attendue.
```

Le premier « IF MY88-VAR-1 THEN » se déclenche, et après le « SET MY88-VAR-2 TO TRUE » suivant, le deuxième « IF MY88-VAR-2 THEN » se déclenche. Le SET écrase la mémoire de structure au point où la variable 88 a été déclarée, et une vérification IF de ce même nom de variable vérifie si la mémoire à cet emplacement contient la valeur de la variable 88. La disposition spécifique de la structure à ce point n'a pas d'importance. Nous voyons qu'une vérification IF de la variable de niveau 88 teste simplement si la valeur à cette adresse a ou non le modèle spécifié dans la variable. Dans ce cas, nous n'avons qu'une seule variable de niveau 88 avec le nom donné dans le programme, donc le « IF MY88-VAR-2 OF feedback » qui a été utilisé était redondant et aurait pu être codé simplement comme « IF MY88-VAR-2 », ou aurait pu être codé comme « IF MY88-VAR-2 OF CONDITION-TOKEN-VALUE of Feedback »

Nous pouvons en déduire que le WORKING-STORAGE du code COBOL a la disposition C++ équivalente suivante :

```
</>
```

```
struct CONDITION_TOKEN_VALUE
{
   short SEVERITY;
   short MSG_NO;
   char CASE_SEV_CTL;
   char FACILITY_ID[3];
};

enum my88_vars
{
   MY88_VAR_1 = 0x0000000141C33A3BL,
   MY88_VAR_2 = 0x0000000241C33A3BL
};

struct feedback
{
   union {
      CONDITION_TOKEN_VALUE c;
      my88_vars e;
   } u;
   int I_S_INFO;
};
```

et que le flux de contrôle du programme peut être modélisé comme suit :

</>

```
//...
feedback f;

int main()
{
   memset( &f, 0x40, sizeof(f) );
   f.u.e = MY88_VAR_1;
   if ( f.u.e == MY88_VAR_2 )
   {
      impossible();
   }
   if ( f.u.e == MY88_VAR_1 )
   {
      expected();
   }

   f.u.e = MY88_VAR_2;
   if ( f.u.e == MY88_VAR_1 )
   {
      impossible();
   }
   if ( f.u.e == MY88_VAR_2 )
```

Les choses deviennent encore plus confuses si la variable LEVEL-88 spécifie moins d'octets que la structure dans laquelle elle est intégrée. Dans ce cas, SET de la variable complète la structure avec des espaces et une vérification de la variable recherche également ces espaces de fin supplémentaires :

```
      *     88 MY88-VAR-3  VALUE X"01020304".
      * ...
            SET MY88-VAR-3 TO TRUE
            IF MY88-VAR-1 OF FeedBack THEN
               PERFORM IMPOSSIBLE
            END-IF

            IF MY88-VAR-2 THEN
               PERFORM IMPOSSIBLE
            END-IF

            IF MY88-VAR-3 THEN
               PERFORM EXPECTED
            END-IF
```

(https://peeterjoot.com/wp-content/uploads/2020/08/Screen-Shot-2020-08-10-at-12.34.27-PM.png)

L'objet CONDITION-TOKEN-VALUE utilise un total de 8 octets. Nous pouvons voir les espaces dans l'affichage de la structure FEEDBACK si nous regardons dans le débogueur :

```
(lldb) b 40
c
Breakpoint 8: where = COBRC.NATIVE.LZ000550(LZ000550).3275741d`TEST + 2249 at TEST.cob:40:1, address = 0x00007ffff7fdeb89
(lldb) c
Process 25590 resuming
Process 25590 stopped
* thread #13, name = 'LZ000550', stop reason = breakpoint 8.1
    frame #0: 0x00007ffff7fdeb89 COBRC.NATIVE.LZ000550(LZ000550).3275741d`TEST at TEST.cob:40:1
   37
   38          *     88 MY88-VAR-3  VALUE X"01020304".
   39          * ...
-> 40    _           SET MY88-VAR-3 TO TRUE
   41                IF MY88-VAR-1 OF FeedBack THEN
   42                   PERFORM IMPOSSIBLE
   43                END-IF
(lldb) n
Process 25590 stopped
* thread #13, name = 'LZ000550', stop reason = step over
    frame #0: 0x00007ffff7fdebeb COBRC.NATIVE.LZ000550(LZ000550).3275741d`TEST at TEST.cob:41:1
   38          *     88 MY88-VAR-3  VALUE X"01020304".
   39          * ...
   40                SET MY88-VAR-3 TO TRUE
-> 41    _           IF MY88-VAR-1 OF FeedBack THEN
   42                   PERFORM IMPOSSIBLE
   43                END-IF
   44
(lldb) fr v -format x feedback
(FEEDBACK) FEEDBACK = {
  CONDITION-TOKEN-VALUE = {
    CASE-1-CONDITION-ID = (SEVERITY = 0x0102, MSG-NO = 0x0304)
    CASE-SEV-CTL-0 = (CASE-SEV-CTL = " ", FACILITY-ID = "   ")
  }
  I-S-INFO = 0x40404040
}
(lldb) x/8xb &feedback
0x7ffba6846908: 0x01 0x02 0x03 0x04 0x40 0x40 0x40 0x40
```

(https://peeterjoot.com/wp-content/uploads/2020/08/Screen-Shot-2020-08-10-at-12.36.07-PM.png)

Voir les quatre espaces 0x40 de fin ici.

Il peut être difficile de déterminer les besoins totaux en stockage d'une structure COBOL en regardant simplement le code, car les correspondances entre les chiffres et le stockage dépendent des clauses d'utilisation. Si la structure utilise également des clauses REDEFINES (unions intégrées), comme c'était le cas dans le programme que j'étudiais à l'origine, la sortie de débogage est également très utile pour comprendre la taille des différents champs et leur emplacement.

Voici quelques-unes des leçons apprises :

- Vous pouvez voir une vérification du type « IF MY88-VAR-1 THEN », mais rien dans le programme ne définit explicitement MY88-VAR-1. Il s'agit en fait d'une valeur de variable globale définie comme un effet

secondaire d'un autre appel (dans le programme réel que j'examinais, ce qui modifiait cette « variable » était en fait un appel à CEEFMDA, un service système LE). Nous avons un passage par référence dans les appels de fonction, ce qui peut être une tâche d'ingénierie inverse pour lire n'importe quel programme et déterminer comment un champ donné peut avoir été modifié, et cela ne devient pas plus facile en introduisant des variables LEVEL-88 dans le mélange.

- Ce mécanisme d'énumération efficace n'est en aucun cas typé. L'utilisation correcte d'un LEVEL-88 repose sur le fait que les variables qui le suivent doivent avoir les types appropriés. Dans ce cas, « IF MY88-VAR-1 THEN » est essentiellement un raccourci pour :

```
IF SEVERITY = 0 AND MSG-NO = 1 AND CASE-SEV-CTL = X"41" AND
   FACILITY-ID = X"C33A3B" THEN
   PERFORM EXPECTED
END-IF
```

(https://peeterjoot.com/wp-content/uploads/2020/08/Screen-Shot-2020-08-10-at-12.22.37-PM.png)

- Il existe une déconnexion entre les variables modifiées ou vérifiées par une référence de variable LEVEL-88 donnée qui doit être déduite par le contexte.
- Une vérification IF d'une variable LEVEL-88 peut inclure une vérification implicite de la partie finale de la structure avec des espaces EBCDIC, si les champs qui suivent la variable 88 occupent plus d'espace que la valeur de la variable. De même, définir une telle variable sur TRUE peut effectivement définir un sous-ensemble final de la structure avec des espaces EBCDIC.
- Ce qui est modifié par une variable 88 donnée dépend du contexte. Par exemple, si les variables de niveau 88 se trouvaient dans un cahier de copie et que j'avais une seconde structure ayant la même disposition que FEEDBACK, les deux structures incluant ce cahier de copie, alors j'aurais deux instances de cette « énumération » et j'aurais besoin d'un ensemble de clauses de type « OF foo OF BAR » pour lever l'ambiguïté. Les variables de niveau 88 ne sont pas comme un ensemble de définitions C. Leur signification dépend du contexte, même si elles se font passer pour des constantes.

↑Retour en haut