Going Reentrant

What is a reentrant program?

When a single copy of a program can safely be executed concurrently, the program is said to be *reentrant*. This implies that an invocation of the program could be interrupted, a second process executing the same code started, and the first program restarted afterward without causing any issues with the two processes′ shared variables. The term *reentrant* derives from the fact that two processes can safely "reenter" the same code concurrently.

If a program doesn′t modify memory, it is by default, reentrant. As you might expect, almost all useful programs modify storage. (An exception occurs in JCL, which invokes an assembler program that simply does a return. Research utility program IEFBR14 for details.) Most interesting programs modify memory. If that is the case, how then can the same code execute concurrently without one instance of the program corrupting memory for the other? The answer is that all the memory a reentrant program modifies must be dynamically acquired. Each process that executes concurrently allocates the storage it needs dynamically.

Assembler program reentrancy

We consider the aspects of assembler programs that affect a program being reentrant.

1) Dynamically Allocating and Releasing All Modifiable Memory – Each reentrant process must acquire and release modifiable storage dynamically. (Storage that is initialized at compile-time and never modified, is excluded. We call this static storage.) IBM provides a reentrant-friendly macro called STORAGE that can obtain and release storage. Here are two examples:

In the first example, DYNSIZE contains the number of required bytes when STORAGE OBTAIN is executed. After execution, the address of the allocated storage is stored in register 11.

```
STORAGE OBTAIN,LENGTH=DYNSIZE,ADDR=(R11),LOC=(24,31)
```

The storage acquired by this macro should be described by a DSECT, which uses the specified register for addressing. The LOC parameter can be omitted, but I included it to emphasize that storage for the DCBs must reside "below the line." The "line" is equal to 2^{24} = 16 megabytes, the limit for 24-bit addresses. Open macros, being very old like the author, assume 24-bit addresses. The sample program uses AMODE 31 and RMODE 24, meaning the program uses 31-bit addresses and resides below the line. Central storage for the program resides below 2 gigabytes.

When the reentrant program is finished using the acquired storage, it is important to release that storage, otherwise, memory leaks can occur. STORAGE RELEASE will return the storage to the subpool from which it was acquired. LENGTH specifies the number of bytes to return, and ADDR specifies the storage address. Variations of these invocations are described in the appropriate Assembler Services Reference.

```
STORAGE RELEASE,LENGTH=DYNSIZE,ADDR=(R11)
```

This invocation releases DYNSIZE bytes starting at the address in register **11**.

2) Adding a DSECT Describing Dynamic Storage – All the modifiable storage needed by the program should be described in a DSECT and a USING coded to establish addressability. This will include storage areas, list-form macros, DCBs (more later), and save areas. Static storage does not need to be included. Also, registers do not require special handling because they are saved by the operating system when switching between processes.

3) Coding Macros – Assembler macros like OPEN, CLOSE, GETMAIN, FREEMAIN, and DCB generate storage areas that are modified as a program executes. We use different techniques for handling these macros:

a) In the cases of GETMAIN and FREEMAIN, we can substitute reentrant-friendly macros STORAGE OBTAIN and STORAGE RELEASE. These macros don't modify local storage areas.

b) Many older macros, like OPEN and CLOSE, provide List and Execute versions, which split each macro into two parts. The List portion of a macro contains the macro storage which is modified during execution. This storage can be included within the storage the program dynamically acquires. This macro will be contained in the DSECT describing dynamic storage. The execute portion of each macro can be included in the executable version of the program. Here is an example of the OPEN macro:

```
            LA    R2,FILEIN
            OPEN  ((2),INPUT),MF=(E,OPENLSTI)
            …
OPENLSTI    OPEN  (,INPUT),MF=L
```

The first OPEN above is executable and is coded in the executable part of the program. This is indicated by the E parameter in the macro format description (MF). The E is followed by a reference to the data management parameter list created by a list form of the macro (OPENLSTI). This part of the macro is not executable and only generates a collection of fields needed for the OPEN. It is coded in a DSECT which describes the dynamically created storage areas needed by the program.

c) The DCB macro generates fields that are initialized at compile-time, but it also generates fields that are modified during execution. As a result, we make two copies of each DCB. One copy of the DCB occupies the static storage that is allocated to the program at compile-time. This DCB contains information about the file that is gleaned during compilation. The other identical DCB is defined in a DSECT, and its storage is dynamically acquired. During execution, the "static" DCB is copied to the "dynamic" DCB before the file is OPENed. The "dynamic" DCB is referenced during execution of the OPEN macro and filled with run-time information.

4) Checking for Reentrancy – The assembler will check for reentrancy if the first control section starts with the RSECT instruction instead of CSECT. This sets the RENT option for the assembler. (RENT is different for the assembler and linkage editor.) The assembler cannot check the logic of the program, so checking is not exhaustive. Here is an example of RSECT:

```
         REENTRA1 RSECT ,
```

REENTRA1 is the CSECT name.

5) Marking the Load Module – The binder and linkage editor can mark the load module as reentrant, allowing it to be called concurrently. Here is a JCJ example that marks the load module as reentrant by specifying the RENT option during the linkage step:

```
//STEP1    EXEC PROC=HLASMCLG,PARM.L='RENT'
```

This JCL adds the RENT parm to the link step, marking the load module as reentrant. Reentrant programs can only call other reentrant modules.

The sample program

```
000100 //KC02486A JOB (KC024861),'WOOLBRIGHT',REGION=3M,CLASS=A,MSGCLASS=H,
000200 // NOTIFY=&SYSUID,MSGLEVEL=(1,1)
000300 //STEP1    EXEC PROC=HLASMCLG,PARM.L='RENT'
000400 //C.ASMAOPT DD DSN=KC02486.ASM.SRC(ASMAOPT),DISP=SHR
000500 //C.SYSIN   DD *
000600          PRINT ON,NODATA,NOGEN
000700 REENTRA1 AMODE 31
000800 REENTRA1 RMODE 24
000900 REENTRA1 RSECT ,
001000 ***********************************
001100          SAVE  (14,12)               SAVE THE CALLER'S REGS
001200          BASR  R12,0                 ESTABLISH ADDRESSABILITY
001300          USING *,R12                 FOR THE CSECT
001400          STORAGE OBTAIN,LENGTH=DYNSIZE,ADDR=(R11),LOC=(24,31)
001500          USING  DYNAREA,R11          ESTABLISH DYNAMIC AREA BASING
001600          LA    R2,SAVEAREA           POINT TO MY LOWER-LEVEL SA
001700          ST    R2,8(,R13)            FORWARD-CHAIN MINE FROM CALLER'S
001800          ST    R13,SAVEAREA+4        BACK-CHAIN CALLER'S FROM MINE
001900          LR    R13,R2                SET 13 FOR ANY SUBROUTINE CALLS
002000 **********************  BEGIN LOGIC  ********************************
002100          MVC   FILEIN(FILEINL),FILEINM     INIT EXECUTEABLE DCB
002200          MVC   FILEOUT(FILEOUTL),FILEOUTM  INIT EXECUTEABLE DCB
002300          LA    R2,FILEIN
002400          LA    R3,FILEOUT
002500 *
002600          OPEN  ((2),INPUT),MF=(E,OPENLSTI)
002700          OPEN  ((3),OUTPUT),MF=(E,OPENLSTO)
002800          MVC   BUFFER,=CL80'=========='
002900          PUT   (R3),BUFFER
003000          GET   (R2),BUFFER
003100 LOOP     EQU   *
003200          PUT   (R3),BUFFER
003300          MVC   BUFFER,=CL80'=========='
003400          PUT   (R3),BUFFER
003500          GET   (R2),BUFFER
```

```
003600          J     LOOP
003700 EXIT     EQU   *
003800          CLOSE ((2)),MF=(E,CLOSELI)
003900          CLOSE ((3)),MF=(E,CLOSELO)
004000 **********************  END LOGIC    ********************************
004100 RETURN   EQU   *                      BRANCH TO HERE FOR NORMAL RETURN
004200          L     R13,SAVEAREA+4         POINT TO CALLER'S SAVE AREA
004300          STORAGE RELEASE,LENGTH=DYNSIZE,ADDR=(R11)
004400          RETURN (14,12),RC=0          RESTORE CALLER'S REGS & RETURN
004500 FILEINM  DCB   DSORG=PS,              MODEL INPUT DCB                 X
004600                MACRF=(GM),                                           X
004700                DEVD=DA,                                              X
004800                DDNAME=FILEIN,                                        X
004900                EODAD=EXIT,                                           X
005000                RECFM=FB,                                             X
005100                LRECL=80
005200          OPEN  (,INPUT),MF=L          LIST TO OPEN DCB
005300          CLOSE (),MF=L                LIST TO CLOSE DCB
005400 FILEINML       EQU       *-FILEINM    MODEL DCB LENGTH
005500 FILEOUTM DCB   DSORG=PS,              MODEL OUTPUT DCB                X
005600                MACRF=(PM),                                           X
005700                DEVD=DA,                                              X
005800                DDNAME=FILEOUT,                                       X
005900                RECFM=FB,                                             X
006000                LRECL=80
006100          OPEN  (,OUTPUT),MF=L         LIST TO OPEN DCB
006200          CLOSE (),MF=L                LIST TO CLOSE DCB
006300 FILEOUTML      EQU      *-FILEOUTM    MODEL DCB LENGTH
006400 ***************  STATIC DATA AREAS   ********************************
006500 STATXMP  DC   F'7'                    STATIC DATA EXAMPLE
006600          LTORG
006700 *************** DYNAMIC DATA AREAS   ********************************
006800 DYNAREA  DSECT                        DSECT FOR THE DYNAMIC AREA
006900 DYNSIZE  EQU  DYNEND-*                CALCULATE AMOUNT OF SPACE NEEDED
007000 SAVEAREA DS    18F                   SAVEAREA SPACE IS DYNAMIC!
007100 BUFFER   DS    CL80                  INPUT/OUTPUT BUFFER
007200 FILEIN   DCB   DSORG=PS,              INPUT DCB                       X
007300                MACRF=(GM),                                           X
007400                DEVD=DA,                                              X
007500                DDNAME=FILEIN,                                        X
007600                EODAD=EXIT,                                           X
007700                RECFM=FB,                                             X
007800                LRECL=80
007900 OPENLSTI OPEN  (,INPUT),MF=L          LIST TO OPEN DCB
008000 CLOSELI  CLOSE (),MF=L                LIST TO CLOSE DCB
008100 FILEINL  EQU   *-FILEIN               DCB MODEL LENGTH
008200 FILEOUT  DCB   DSORG=PS,              OUTPUT DCB                      X
008300                MACRF=(PM),                                           X
008400                DEVD=DA,                                              X
008500                DDNAME=FILEOUT,                                       X
008600                RECFM=FB,                                             X
```

```
008700                 LRECL=80
008800 OPENLSTO OPEN   (,OUTPUT),MF=L          LIST TO OPEN DCB
008900 CLOSELO  CLOSE  (),MF=L                 LIST TO CLOSE DCB
009000 FILEOUTL EQU    *-FILEOUT               DCB MODEL LENGTH
009100 DYNEND   EQU  *                         EQUATE FOR LENGTH CALCULATION
009200          YREGS ,                        R0, R1, ETC. EQUATES
009300          END REENTRA1                   END OF SECTION
009400 /*
009500 //G.FILEIN   DD  *
009600 THIS IS RECORD 1
009700 THIS IS RECORD 2
009800 THIS IS RECORD 3
009900 THIS IS RECORD 4
010000 THIS IS RECORD 5
010100 /*
010200 //G.FILEOUT  DD  SYSOUT=*
010300 //G.SYSUDUMP DD  SYSOUT=*
010400 //
****** **************************** Bottom of Data ****************************
```

Program Notes

000300 //STEP1   EXEC PROC=HLASMCLG,PARM.L='RENT'

The JCL informs the linkage editor to mark the load module as reentrant.

000700 REENTRA1 AMODE 31
000800 REENTRA1 RMODE 24

The program will use 31 bit addresses and reside below the line.

000900 REENTRA1 RSECT ,

The control section is marked as reentrant for the assembler.

001400   STORAGE OBTAIN,LENGTH=DYNSIZE,ADDR=(R11),LOC=(24,31)
001500   USING DYNAREA,R11   ESTABLISH DYNAMIC AREA BASING

DYNSIZE bytes are acquired from subpool 0. The address of that storage is copied to R11. The storage is acquired below the line and central storage is below 3 gigabytes. The USING associated DSECT DYNARE with R11, establishing addressability to the dynamic storage area.

```
002100           MVC   FILEIN(FILEINL),FILEINM     INIT EXECUTEABLE DCB
002200           MVC   FILEOUT(FILEOUTL),FILEOUTM  INIT EXECUTEABLE DCB
```

The static versions of the input and output DCBs (including the list formats of OPEN and CLOSE, are copied to the dynamic versions of those macros. When opened, the dynamic versions will contain all the needed information about the files gleaned from compilation and execution of OPEN and CLOSE.

```
002300           LA    R2,FILEIN
002400           LA    R3,FILEOUT
002500 *
002600           OPEN  ((2),INPUT),MF=(E,OPENLSTI)
002700           OPEN  ((3),OUTPUT),MF=(E,OPENLSTO)
```

R2 and R3 are loaded with the runtime addresses of FILEIN and FILEOUT (the DCBs in dynamic storage). The files are opened using the executable versions of these macros. The executable versions reference OPENLSTI and OPENSTO, the list versions of these macros, which contain fields modified during OPEN.

```
002800           MVC   BUFFER,=CL80'=========='
002900           PUT   (R3),BUFFER
003000           GET   (R2),BUFFER
003100 LOOP      EQU   *
003200           PUT   (R3),BUFFER
003300           MVC   BUFFER,=CL80'=========='
003400           PUT   (R3),BUFFER
003500           GET   (R2),BUFFER
003600           J     LOOP
```

The part of the program which reads an input file and writes an output file.

```
003800           CLOSE ((2)),MF=(E,CLOSELI)
003900           CLOSE ((3)),MF=(E,CLOSELO)
```

The executable CLOSE macro is called twice to close both files. The executable macro references CLOSELI and CLOSELO, the list forms of CLOSE.

```
004100 RETURN    EQU   *                     BRANCH TO HERE FOR NORMAL RETURN
004200           L     R13,SAVEAREA+4        POINT TO CALLER'S SAVE AREA
004300           STORAGE RELEASE,LENGTH=DYNSIZE,ADDR=(R11)
004400           RETURN (14,12),RC=0         RESTORE CALLER'S REGS & RETURN
```

The program prepares to return but first releases the dynamic storage it acquired.

```
004500 FILEINM   DCB   DSORG=PS,             MODEL INPUT DCB                 X
004600                 MACRF=(GM),                                           X
004700                 DEVD=DA,                                              X
004800                 DDNAME=FILEIN,                                        X
004900                 EODAD=EXIT,                                           X
005000                 RECFM=FB,                                             X
005100                 LRECL=80
```

```
005200          OPEN  (,INPUT),MF=L          LIST TO OPEN DCB
005300          CLOSE (),MF=L                LIST TO CLOSE DCB
005400 FILEINML       EQU       *-FILEINM    MODEL DCB LENGTH
005500 FILEOUTM DCB   DSORG=PS,              MODEL OUTPUT DCB                X
005600                MACRF=(PM),                                           X
005700                DEVD=DA,                                              X
005800                DDNAME=FILEOUT,                                       X
005900                RECFM=FB,                                             X
006000                LRECL=80
006100          OPEN  (,OUTPUT),MF=L         LIST TO OPEN DCB
006200          CLOSE (),MF=L                LIST TO CLOSE DCB
006300 FILEOUTML      EQU      *-FILEOUTM    MODEL DCB LENGTH
```

The program defines two “static” macros and the executable OPENs and CLOSEs they will use. Memory for these macros is in central storage along with the executable instructions of the program. FILEINML is the length of the input DCB and its executable OPEN and Close. FILEOUTML is the length of the output DCB and its executable OPEN and Close.

```
006400 ***************  STATIC DATA AREAS   *********************************
006500 STATXMP  DC   F'7'                    STATIC DATA EXAMPLE
006600          LTORG
```

Static variables like STATXMP can reside in central storage. The literal pool denoted by LTORG can also reside in central storage.

```
006700 *************** DYNAMIC DATA AREAS   *********************************
006800 DYNAREA  DSECT                        DSECT FOR THE DYNAMIC AREA
006900 DYNSIZE  EQU  DYNEND-*                CALCULATE AMOUNT OF SPACE NEEDED
007000 SAVEAREA DS    18F                   SAVEAREA SPACE IS DYNAMIC!
007100 BUFFER   DS    CL80                  INPUT/OUTPUT BUFFER
007200 FILEIN   DCB   DSORG=PS,              INPUT DCB                       X
007300                MACRF=(GM),                                           X
007400                DEVD=DA,                                              X
007500                DDNAME=FILEIN,                                        X
007600                EODAD=EXIT,                                           X
007700                RECFM=FB,                                             X
007800                LRECL=80
007900 OPENLSTI OPEN  (,INPUT),MF=L          LIST TO OPEN DCB
008000 CLOSELI  CLOSE (),MF=L                LIST TO CLOSE DCB
008100 FILEINL  EQU   *-FILEIN               DCB MODEL LENGTH
008200 FILEOUT  DCB   DSORG=PS,              OUTPUT DCB                      X
008300                MACRF=(PM),                                           X
008400                DEVD=DA,                                              X
008500                DDNAME=FILEOUT,                                       X
008600                RECFM=FB,                                             X
008700                LRECL=80
008800 OPENLSTO OPEN  (,OUTPUT),MF=L         LIST TO OPEN DCB
008900 CLOSELO  CLOSE (),MF=L                LIST TO CLOSE DCB
```

```
009000 FILEOUTL EQU   *-FILEOUT              DCB MODEL LENGTH
009100 DYNEND   EQU  *                       EQUATE FOR LENGTH CALCULATION
```

The DSECT for the dynamic storage area begins here. This DSECT contains the SAVEAREA since since each concurrently running process will need its own save area. BUFFER is modified during execution, so it goes into the DSECT for dynamic storage. Copies of the input and output DCBs (and each Open and CLOSE) are required in dynamic storage. We will copy the static versions of these macros to the dynammic versions during execution.

```
009200          YREGS ,                      R0, R1, ETC. EQUATES
009300          END REENTRA1                 END OF SECTION
```

Register equate names are established and the RSECT is ended.

```
009400 /*
009500 //G.FILEIN   DD  *
009600 THIS IS RECORD 1
009700 THIS IS RECORD 2
009800 THIS IS RECORD 3
009900 THIS IS RECORD 4
010000 THIS IS RECORD 5
010100 /*
010200 //G.FILEOUT  DD  SYSOUT=*
010300 //G.SYSUDUMP DD  SYSOUT=*
010400 //
```

The input file is defined “instream”. FILEOUT and a dump file are defined.
