Assignment 3

Condition Code Handling and JCL Overrides

Objective

1. Understand how condition codes are used to control job step execution.
2. Learn how to modify procedure parameters at runtime using JCL overrides.

Theory

This assignment covers two fundamental techniques in Job Control Language (JCL) that are essential for creating flexible and intelligent batch jobs.

1. Condition Code Handling (Simulating IF-THEN-ELSE)

In JCL, we cannot write a direct if-else statement like in other programming languages. Instead, we control the execution of job steps based on the outcome of previous steps. This is achieved using **Return Codes (RC)** and the **COND** parameter.

- **Return Code (RC):** After a program in a job step finishes, it returns a number called a return code (or condition code) to the system. This code indicates whether the program was successful.
    - **0000:** Successful execution.
    - **0004:** Warning; the step was successful but there might be a minor issue.
    - **0008:** Error; the step ran but encountered an error.
    - **0012 or higher:** Severe error; the step failed significantly.
- **The COND Parameter:** This parameter is added to an EXEC statement to test the return codes from previous steps. Its logic is often described as "reverse logic" because **the step is BYPASSED if the condition is TRUE.**

  The syntax is: COND=(code, operator, stepname)
    - **code:** The number you want to test against.
    - **operator:** The comparison to perform (e.g., EQ, GT, LT).
    - **stepname:** The name of the previous step whose return code you want to check.

  Common Operators:
    - **EQ:** Equal
    - **NE:** Not Equal
    - **GT:** Greater Than

- **LT:** Less Than
- **GE:** Greater Than or Equal
- **LE:** Less Than or Equal

By using two steps with opposing COND parameters, we can create an IF-THEN-ELSE structure. For example, if STEP1 is our condition:

THEN Block: EXEC PGM=..., COND=(0,NE,STEP1) (Runs only if STEP1 RC is 0).

ELSE Block: EXEC PGM=..., COND=(0,EQ,STEP1) (Runs only if STEP1 RC is NOT 0).

2. JCL Procedure Overriding

A JCL Procedure (PROC) is a pre-written, reusable piece of JCL code that can be stored and called by multiple jobs. This prevents you from having to write the same JCL statements over and over again. To make PROCs flexible, we use **symbolic parameters**.

- **Symbolic Parameters:** These are like variables within a PROC. Their names begin with an ampersand (&), such as &DSN or &MEMBER. You can set a default value for these parameters in the PROC definition.

  Example PROC definition with a default value: //CREATEFL PROC DSN='Z76894.DEFAULT.FILE'

- **Overriding:** When you execute the PROC using an EXEC statement, you have the option to provide a new value for any of its symbolic parameters. This new value **overrides** the default value for that specific run only.

  Example EXEC statement that overrides the default: //RUNOVRDE EXEC PROC=CREATEFL,DSN='Z76894.NEW.FILE'

  In this example, even though the PROC has a default DSN, the job will use 'Z76894.NEW.FILE' for this execution. This technique makes PROCs extremely powerful and reusable for many different scenarios without needing to change the underlying procedure code.

Part A – Condition Code Handling

Step 1: Create JCL

In your dataset `Z76893.JCL(CONDJCL)`:

zosmf > Z76893.JCL > JCL ASSI2.jcl > ...

```
1   //CONDJCL  JOB (ACCT),'SNEHAL',CLASS=A,MSGCLASS=X
2   //STEP1    EXEC PGM=IEFBR14
3   //STEP2    EXEC PGM=IEFBR14,COND=(0,NE,STEP1)
4   //DD1      DD   DUMMY
5
```

Step 2: Submit Job

```
(base) PS D:\mainframe> zowe jobs submit data-set "Z76893.JCL(ASSI2)" > cond_log.txt
```

```
(base) PS D:\mainframe> zowe jobs view job-status-by-jobid JOB00348
jobid:   JOB00348
retcode: CC 0000
jobname: CONDJCL
status:  OUTPUT
(base) PS D:\mainframe>
```

```
(base) PS D:\mainframe> cat cond_log.txt
jobid:   JOB00348
retcode: null
jobname: CONDJCL
status:  INPUT
(base) PS D:\mainframe>
```

Step 3: Check Job Output

```
(base) PS D:\mainframe> cat cond_log.txt
jobid:   JOB00348
retcode: null
jobname: CONDJCL
status:  INPUT
(base) PS D:\mainframe> zowe jobs view all-spool-content JOB00348 > cond_output.txt
(base) PS D:\mainframe> cat cond_output.txt
Spool file: JESMSGLG (ID #2, Step: JES2)
1                J E S 2  J O B  L O G  --  S Y S T E M  S 0 W 1  --  N O D E  S V S C J E S 2
0
 00.15.41 JOB00348 ---- WEDNESDAY, 10 SEP 2025 ----
 00.15.41 JOB00348  IRR010I  USERID Z76893   IS ASSIGNED TO THIS JOB.
 00.15.41 JOB00348  ICH70001I Z76893   LAST ACCESS AT 00:09:24 ON WEDNESDAY, SEPTEMBER 10, 2025
 00.15.41 JOB00348  $HASP373 CONDJCL  STARTED - INIT 1    - CLASS A        - SYS S0W1
 00.15.41 JOB00348  -                                    -----TIMINGS (MINS.)------                          -----PAGING COUNTS
----
 00.15.41 JOB00348  -STEPNAME PROCSTEP    RC   EXCP   CONN       TCB       SRB  CLOCK          SERV  WORKLOAD  PAGE  SWAP   VIO S
WAPS
 00.15.41 JOB00348  -STEP1               00      1      0       .00       .00     .0               BATCH        0     0     0
   0
 00.15.41 JOB00348  -STEP2               00      2      0       .00       .00     .0               BATCH        0     0     0
```

Part B – JCL Overrides

Step 4: Create a PROC

In your JCL dataset (Z76893.JCL(PROCJOB)):

```
zosmf > Z76893.JCL > JCL COND.jcl > ...
1   //PROCJOB  JOB (ACCT),'SNEHAL',CLASS=A,MSGCLASS=X
2   //MYPROC   PROC INFILE=SYS1.INPUT,OUTFILE=SYS1.OUTPUT
3   //STEP1    EXEC PGM=IEFBR14
4   //DD1      DD   DSN=&INFILE,DISP=SHR
5   //DD2      DD   DSN=&OUTFILE,DISP=OLD
6   //         PEND
7   //*-------------------------------------------
8   //STEPX    EXEC MYPROC,INFILE=Z76893.MY.INPUT,OUTFILE=Z76893.MY.OUTPUT
9
```

Step 4: Submit Job

```
(base) PS D:\mainframe> zowe jobs submit data-set "Z76893.JCL(COND)"
jobid:   JOB00381
retcode: null
jobname: PROCJOB
status:  INPUT
```

```
(base) PS D:\mainframe> zowe jobs view all-spool-content JOB00381 > proc_output.txt
(base) PS D:\mainframe>
(base) PS D:\mainframe> cat proc_output.txt
Spool file: JESMSGLG (ID #2, Step: JES2)
1                J E S 2  J O B  L O G  --  S Y S T E M  S 0 W 1  --  N O D E  S V S C J E S 2
0
 00.23.44 JOB00381 ---- WEDNESDAY, 10 SEP 2025 ----
 00.23.44 JOB00381  IRR010I  USERID Z76893   IS ASSIGNED TO THIS JOB.
 00.23.44 JOB00381  ICH70001I Z76893   LAST ACCESS AT 00:15:41 ON WEDNESDAY, SEPTEMBER 10, 2025
 00.23.44 JOB00381  $HASP373 PROCJOB  STARTED - INIT 1    - CLASS A        - SYS S0W1
 00.23.44 JOB00381  -                                    -----TIMINGS (MINS.)------                          -----PAGING COUNTS
----
 00.23.44 JOB00381  -STEPNAME PROCSTEP    RC   EXCP   CONN       TCB       SRB  CLOCK          SERV  WORKLOAD  PAGE  SWAP   VIO S
WAPS
 00.23.44 JOB00381  -STEPX    STEP1    FLUSH      0      0       .00       .00     .0               BATCH        0     0     0
   0
 00.23.44 JOB00381  IEF453I PROCJOB - JOB FAILED - JCL ERROR
 00.23.44 JOB00381  -PROCJOB  ENDED.  NAME-SNEHAL              TOTAL TCB CPU TIME=      .00 TOTAL ELAPSED TIME=    .0
 00.23.44 JOB00381  $HASP395 PROCJOB  ENDED
0------ JES2 JOB STATISTICS ------
-  10 SEP 2025 JOB EXECUTION DATE
-           8 CARDS READ
-          50 SYSOUT PRINT RECORDS
-           0 SYSOUT PUNCH RECORDS
-           3 SYSOUT SPOOL KBYTES
-        0.00 MINUTES EXECUTION TIME
```

Conclusion

- **Condition Codes** control execution of dependent job steps, ensuring efficiency and preventing unnecessary execution.
- **JCL Overrides** can be used even without a PROCLIB by coding the PROC inside the same JCL and then overriding its parameters.
- This makes JCL flexible and easy to adapt without modifying the base procedure.