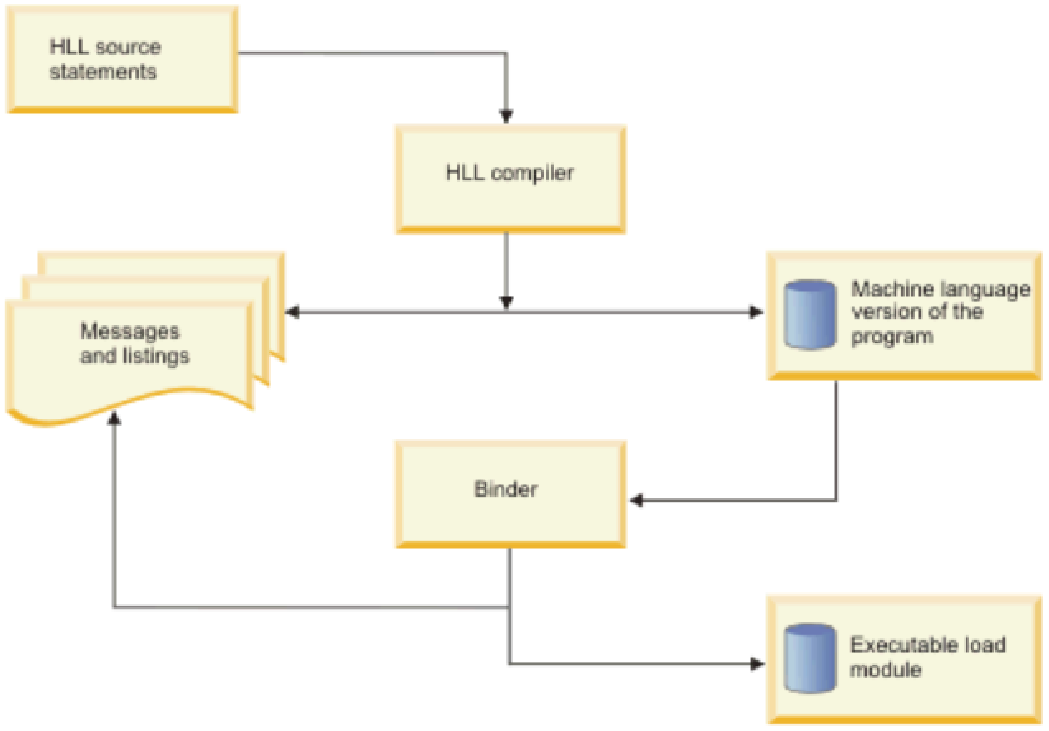
INFO3105 Week 2 Part 1

Review

- Chapter 17
- COBOL PDS (Partitioned Data Set) creation
- Lab2 PDS member creation
- Lab2INP data sequential file creation

Next – Compile, Link and Execute the Lab2 COBOL Program



We'll now go through the process of compiling, linking and executing our program on the Marist host. The diagram above shows the process we have to go through (there is a similar diagram on page 49 of the text). Our COBOL source code is the first rectangle and we need to pass it to a compiler to turn it into machine code (object code) or notify us of any compiler errors which will show up in a separate listing file. To compile our code we need to look at new topic of JCL or Job Control Language.

Job Control Language (JCL)

Job Control Language (JCL) tells the system what program to execute and provides a description of program inputs and outputs.

There are three basic JCL statements:

- **JOB -** Provides a name to the system for this batch workload. It can optionally include accounting information and a few job-wide parameters.

- **EXEC** Provides the name of a program to execute. There can be multiple EXEC statements in a job. Each EXEC statement within the same job is a *job step*.

- **DD** The Data Definition provides inputs and outputs to the execution program on the EXEC statement. This statement links a data set or other I/O device or function to a DDNAME coded in the program. DD statements are associated with a particular job step.

The following is the JCL we will use today:

```
1.  //LAB2 JOB LAB2,NOTIFY=&SYSUID
2.  //*********************************************
3.  //* JCL TO COMPILE COBOL SOURCE CODE AND LINK MODULE
4.  //*********************************************
5.  //S1 EXEC IGYWCL
6.  //COBOL.SYSIN DD DSN=&SYSUID..COBOL(LAB2),DISP=SHR
7.  //LKED.SYSLMOD DD DSN=&SYSUID..LOAD(LAB2),DISP=SHR
8.  //*********************************
9.  //* JCL TO EXECUTE COBOL PROGRAM MODULE
10. //*********************************
11. //EXECUTE EXEC PGM=LAB2
12. //STEPLIB DD DSN=&SYSUID..LOAD,DISP=SHR
13. //PRNT DD SYSOUT=*
14. //INPUT DD DSN=KC03xxx.LAB2INP,DISP=SHR
```

Line 1 – The job card
- Forward slashes in columns 1-2
- Next 1-8 characters are the Job Name, in this case the job is called LAB2
- Notify is used to indicate to a particular TSO system user that the job is submitted, in our case it will be your K03... account (The &SYSUID parameter will be auto filled in with your KC03... ID value when the job executes)

Line 2 –4
- Comments - the asterisk in col 3 indicates a comment

Line 5 – Execute procedure

- Forward slashes in cols 1-2
- Job Name for procedure (S1 here stands for Step 1)
- Name of real procedure (IGYWCL stands for Cobol compile and link)

Line 6 – DD for the Cobol compile
- Forward slashes cols 1-2
- COBOL.SYSIN name for input file
    - COBOL is a name in the PROC IGYWCL. This line in the proc that actually executes the compiler
    - SYSIN is a name that the compiler is looking for input from
- &SYSUID..COBOL(LAB2) this is the fully qualified name of the file,
    - &SYSUID is the account you have signed into TSO with (KC03…) this &SYSUID parameter gets replaced with your id when the job runs
    - COBOL(LAB2) is the PDS and member in this case the source for our COBOL program
- DISP=SHR means the file can be read by others while we are reading it

Line 7 - Forward slashes cols 1-2
- LKED.SYSLMOD name for input file
    - LKED is a name in the PROC IGYWCL. This line in the proc that actually executes the linker (linker transforms obj code into executable code)
    - SYSLMOD is a name that the system compiler is looking to output to
- &SYSUID..LOAD(LAB2) this is the fully qualified name of the load file,
    - &SYSUID is the account you have signed into TSO with (KC03…)
    - LOAD(LAB2) is the PDS and member in this case the output for our Linkage step (we haven't created this PDS yet but will later)
    - Note : This KC03xxx.LOAD(LAB2) only gets updated if the compile is successful – if there are compile errors the executable is not created/updated
- DISP=SHR read by others while we are using it

Lines 8-10 - Comments

Line 11
- Forward slashes in cols 1-2
- Name of the Execute step of the JOB
- EXEC PGM=LAB2 is actually going to run the LOAD module called LAB2

LINE 12
- STEPLIB – name for the library of the load module
- &SYSUID..LOAD(LAB2) this is the fully qualified name of the file,
    - &SYSUID is the account you have signed into TSO with (KC03…)
    - LOAD(LAB2) is the PDS and member in this case the output for our Linkage step (we haven't created this PDS yet but will later)
- DISP=SHR read by others while we are reading it

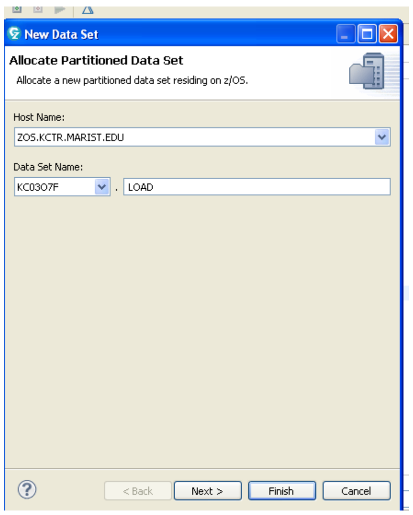
LINE 13
- PRNT is the name of our printer output file, this will match the logical name found in the FILE-CONTROL section of our COBOL program
- DD designates the line as a data definition statement
- SYSOUT=* tells the system to use a name SYSOUT which is just a file that mimics a printer

LINE 14

- INPUT is the name of our source data file, this will match the logical name found in the FILE-CONTROL section of our COBOL program
- DD designates the line as a data definition statement
- DSN=KC03JA4.LAB2INP is the name of a sequential file we created last class (note not a PDS) file that contains the input data for our COBOL program

Next we need to create a PDS to store/house the output of our compile and link process when the compile and link are successfuls, this is known as an **object module** (see page 513 of the text). We'll call this new PDS **LOAD.**

So we create this PDS with the following characteristics:



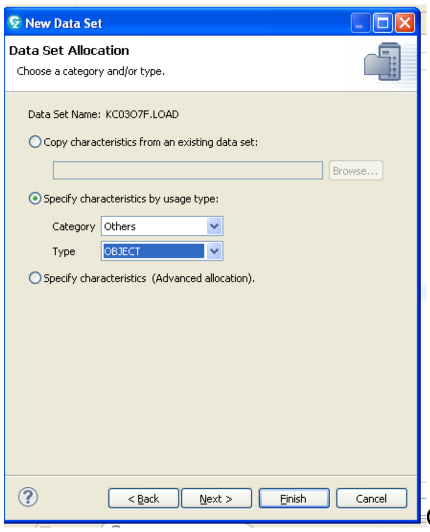
New Data Set

Allocate Partitioned Data Set

Allocate a new partitioned data set residing on z/OS.

Host Name: ZOS.KCTR.MARIST.EDU

Data Set Name: KC03O7F . LOAD



New Data Set

Data Set Allocation

Choose a category and/or type.

Data Set Name: KC03O7F.LOAD

Copy characteristics from an existing data set: [ ]

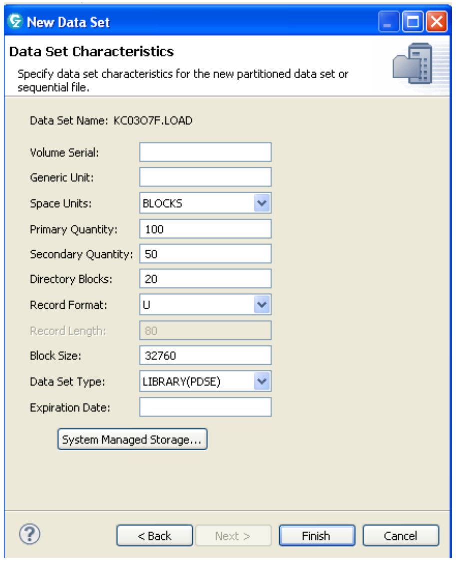
[✓ Specify characteristics by usage type:]

Category: Others

Type: OBJECT

Specify characteristics (Advanced allocation). [ ]

Category: **Others**, Type: **Object**



New Data Set

Data Set Characteristics

Specify data set characteristics for the new partitioned data set or sequential file.

Data Set Name: KC03O7F.LOAD

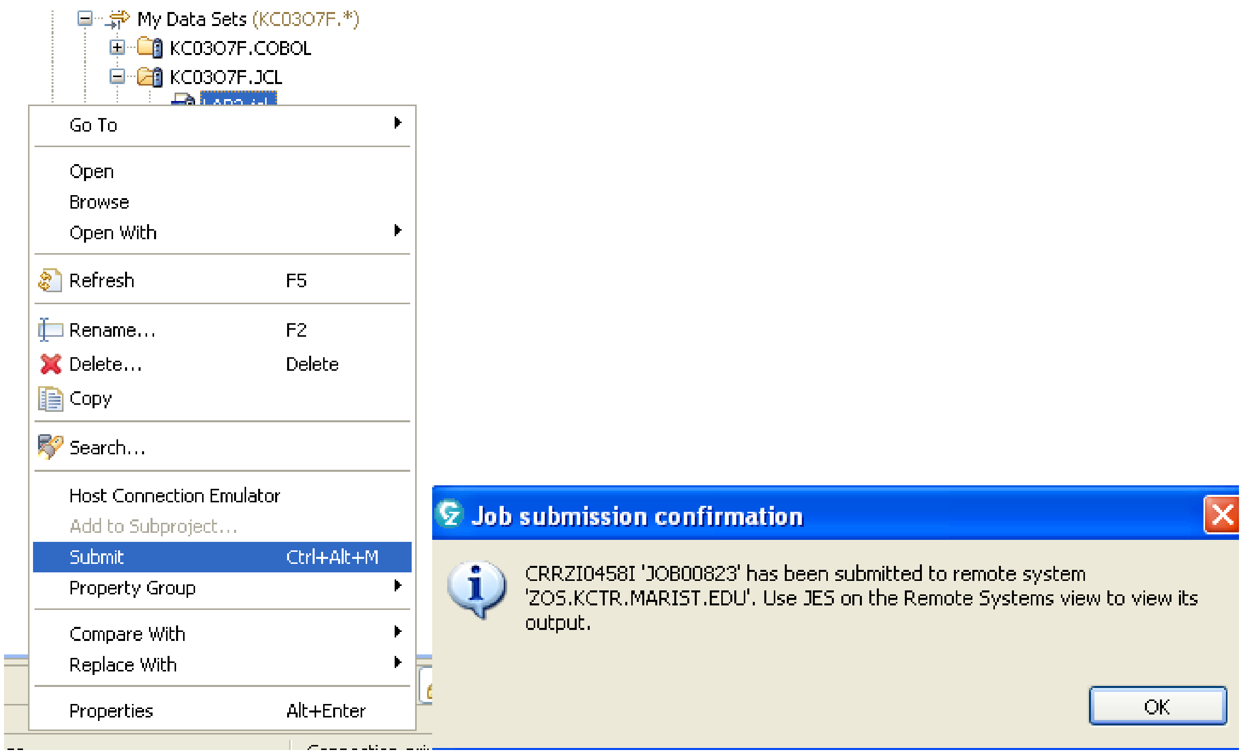
Volume Serial:	
Generic Unit:	
Space Units:	BLOCKS
Primary Quantity:	100
Secondary Quantity:	50
Directory Blocks:	20
Record Format:	U
Record Length:	80
Block Size:	32760
Data Set Type:	LIBRARY(PDSE)
Expiration Date:	

Primary 100, Secondary 50, **Record Format U, Block Size 32760**

We now go through the exact same process as last class and create one more source PDS called **JCL**. This PDS has the same characteristics as our COBOL. To the new PDS add a new member called **LAB2**. The result will be the creation of obviously KC03___**.JCL(LAB2).** Edit this member and add the following lines (xxx is your number – remember to save the changes before you Submit this job):

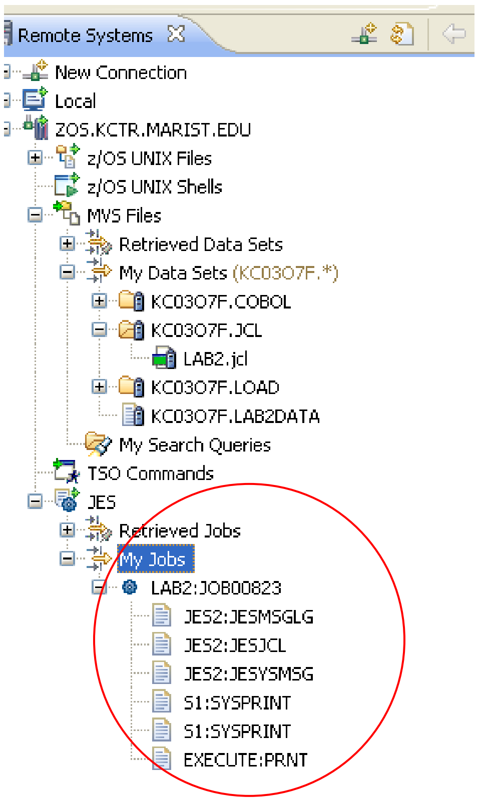
```
//LAB2 JOB LAB2,NOTIFY=&SYSUID
//*****************************************************
//* JCL TO COMPILE COBOL SOURCE CODE AND LINK MODULE
//*****************************************************
//S1 EXEC IGYWCL
//COBOL.SYSIN DD DSN=&SYSUID..COBOL(LAB2),DISP=SHR
//LKED.SYSLMOD DD DSN=&SYSUID..LOAD(LAB2),DISP=SHR
//*************************************
//* JCL TO EXECUTE COBOL PROGRAM MODULE
//*************************************
//EXECUTE EXEC PGM=LAB2
//STEPLIB DD DSN=&SYSUID..LOAD,DISP=SHR
//PRNT DD SYSOUT=*
//INPUT DD DSN=KC03xxx.LAB2INP,DISP=SHR
```

Once the JCL is entered we need to process it in order to compile, link and run our program. The way this OS works is everything runs in the context of a **job**. We'll send this JCL (job) to be processed with the **SUBMIT** command (right click the Lab2 member in the JCL PDS and choose submit or better yet use the Right-Click & Submit ZOS.KCTR.MARIST.EDU or the Ctrl-Alt-M command shortcut):

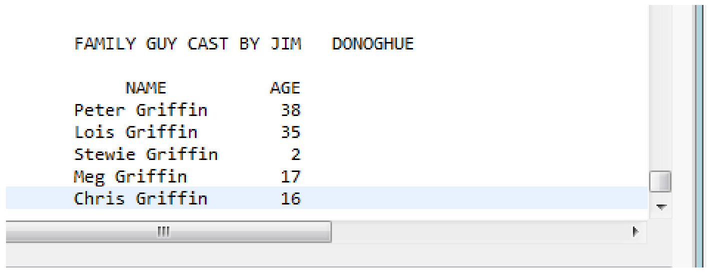


CRRZI0458I 'JOB00823' has been submitted to remote system 'ZOS.KCTR.MARIST.EDU'. Use JES on the Remote Systems view to view its output.

This will send the JCL to a queue for processing, you should see the information popup. Then locate the JES (Job Entry System folder) and locate the job in question, this example the Job is referenced as **JOB00823**.

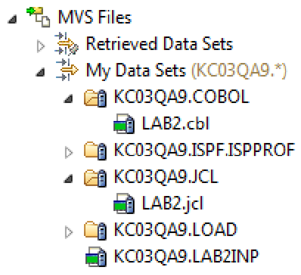


Lastly double click on the **EXECUTE:PRNT** to see the printer output from the program (if EXECUTE:PRNT is not in the list you have an error and need to look at the first S1:SYSPRINT entry to determine the errors).

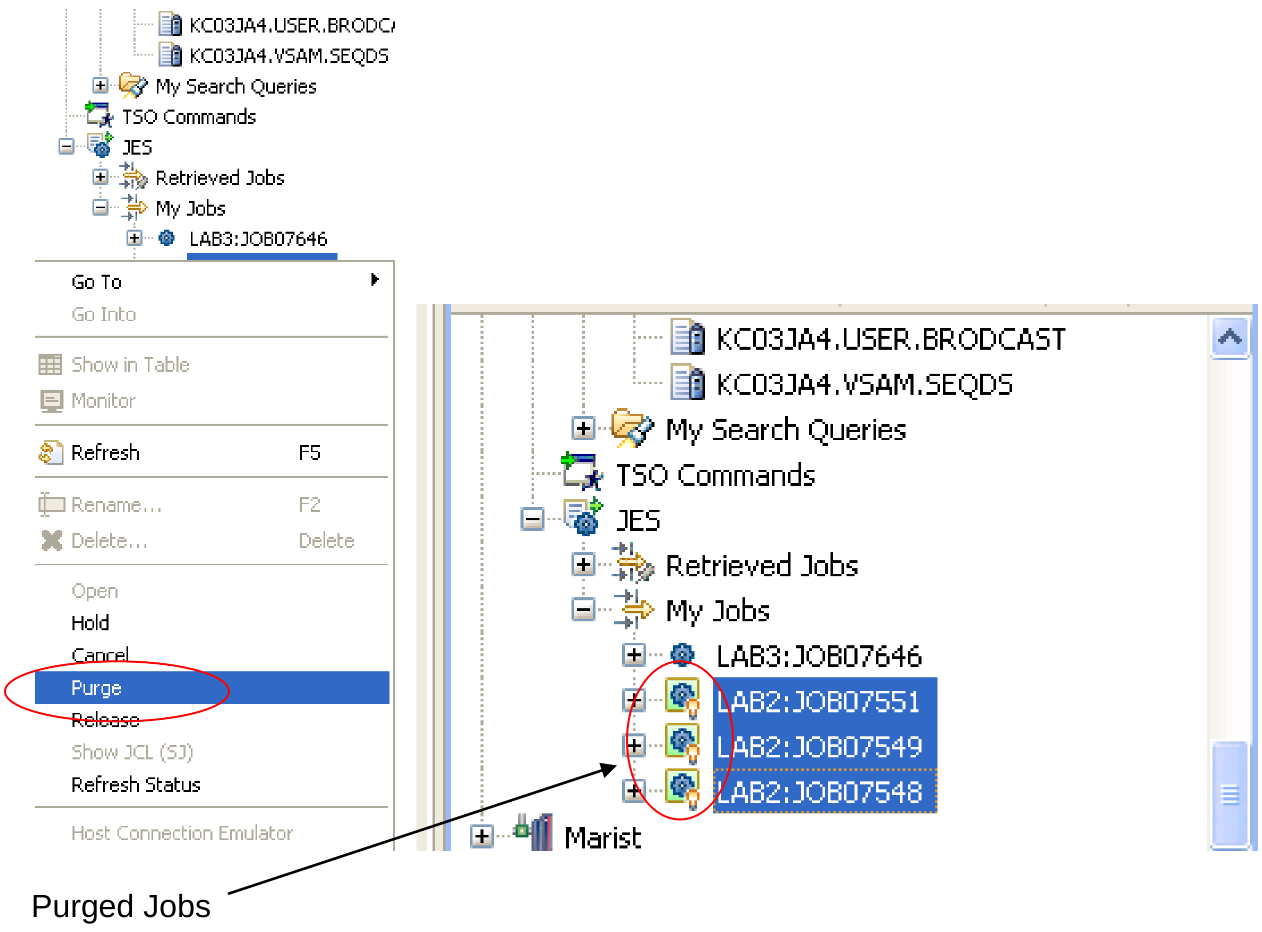


```
      FAMILY GUY CAST BY JIM    DONOGHUE

           NAME           AGE
      Peter Griffin        38
      Lois Griffin         35
      Stewie Griffin        2
      Meg Griffin          17
      Chris Griffin        16
```



You should also clean up your old listing by selecting a number of them, right clicking and choose **Purge**:



Purged Jobs

<u>Lab 3 - 2%</u>

- Using the RDZ environment,
  - Create a PDS called **LOAD** to house the compiled object code
    - Primary 100
    - Secondary 50
    - Record Format = U
    - Block Size = 32760
  - Create a PDS called **JCL** to house our compile link and execute jcl
  - Add a member called **LAB2** to the JCL PDS
  - Make the needed changes to the LAB2 cobol member to generate the new output (see pg 31 of the text for zero suppression).
  - Submit a screen shot of the new printer output (make **sure I can see your name and the tab heading that the output is in**).
- Read Chapter 4 of the text