interskill learning

Batch Systems

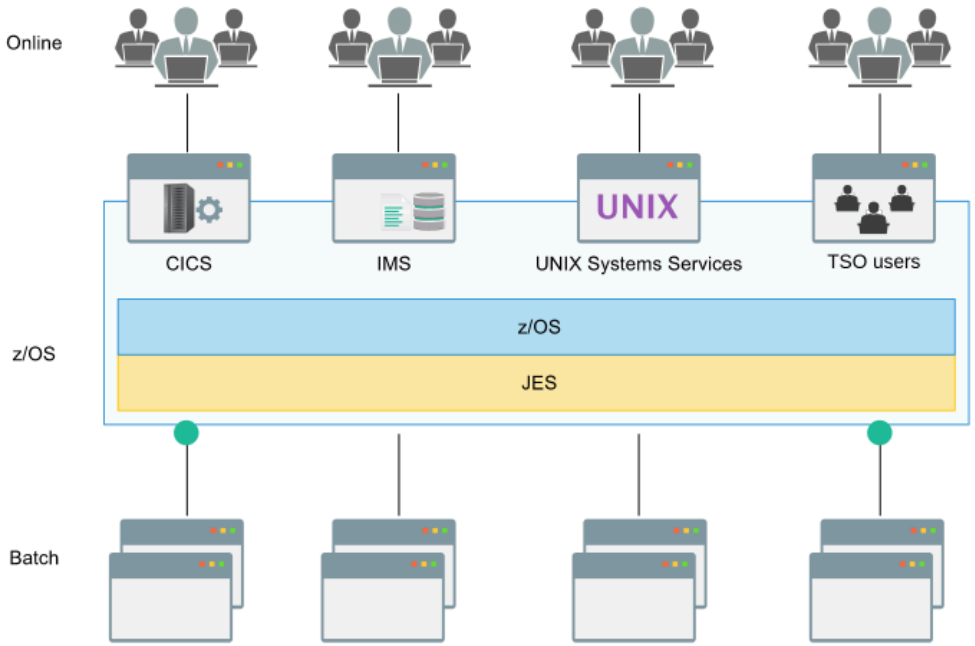
Objectives

Batch Systems

In this module, you will discover how the IBM enterprise environment is built to enable maximum batch processing in major corporations.

After completing this module, you will be able to:

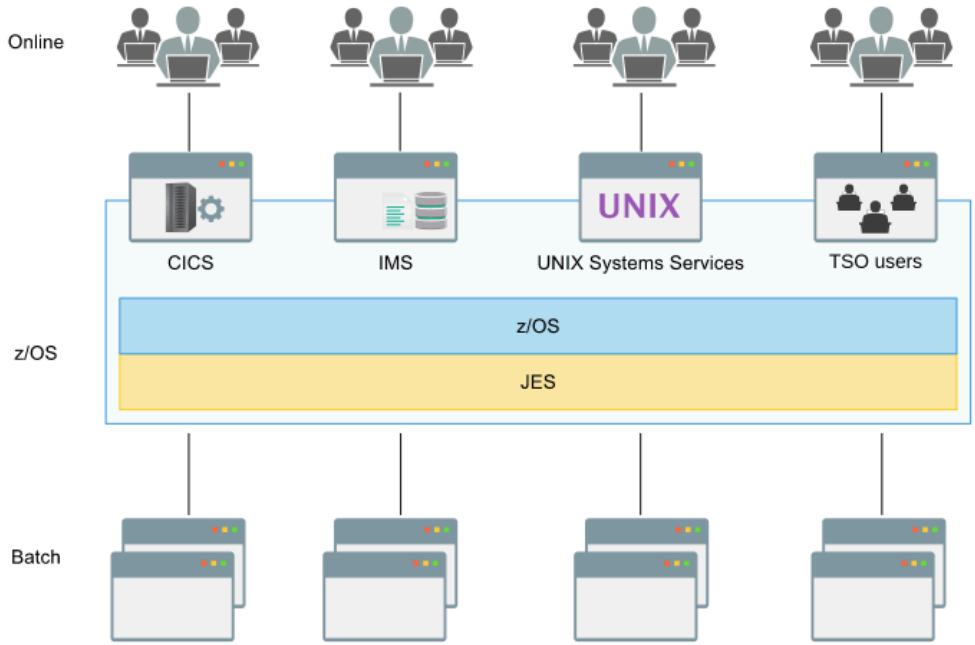
- Identify the Function of Job Entry Subsystem (JES)
- Identify How Job Control Language (JCL) Controls the Running of a Program
- Define Batch Processing
- Define System Display and Search Facility (SDSF)



Online systems like IMS and CICS have given us interactive user interfaces that have been extended to the Internet.

Although online systems are the most visible components, another efficient use of computer resources is to perform processing in the background. These processes are called batch jobs and they have no visible user interface.

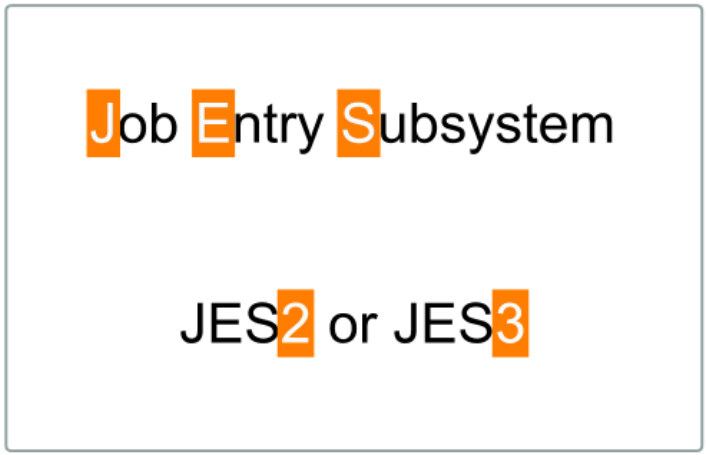
Batch jobs may act on data or triggers entered in an online system, or they may be periodic processes set up in a scheduling tool that occur once per interval, for example, hourly, daily, or monthly.



Through the use of different transaction processors and subsystems, the IBM enterprise environment enables many thousands of users to process concurrently with many background tasks.

Online users spend most of their time thinking or entering information. On entry, z/OS processes this information and waits for more. This leaves a lot of spare capacity for other uses, such as background or batch processes.

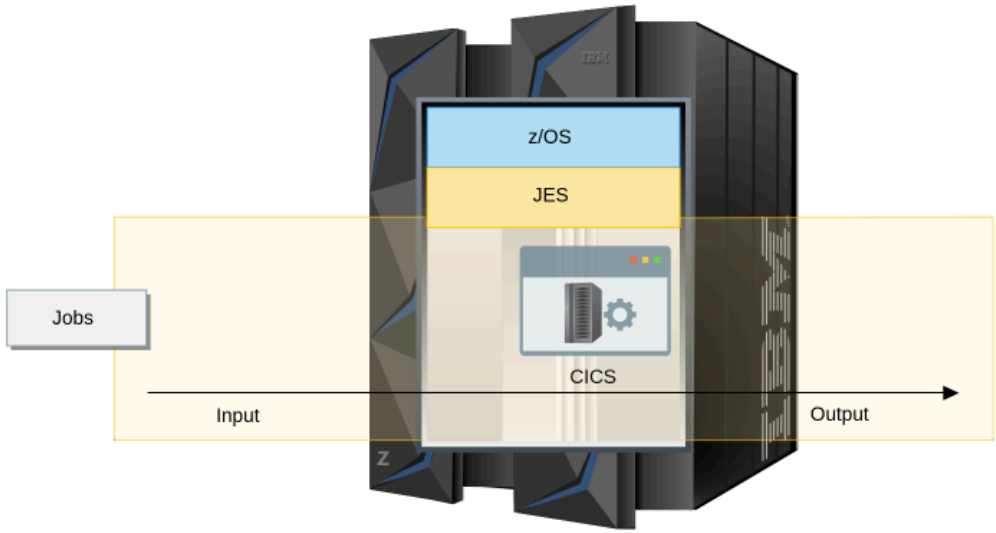
You will look at some of the facilities that enable this processing, and see how some specific processes are initiated.



z/OS provides two options for batch processing subsystems, JES2 and JES3. Both were developed in the 1960s from different predecessors and both provide scheduling of execution and spooling.

JES2 comes standard with z/OS and is very popular. JES3 is an optional feature of z/OS, which is used by a much smaller number of large organizations. Together, and where they provide similar functionality, they are referred to as JES.

In this overview, you will concentrate on generic JES and the specific features of JES2.

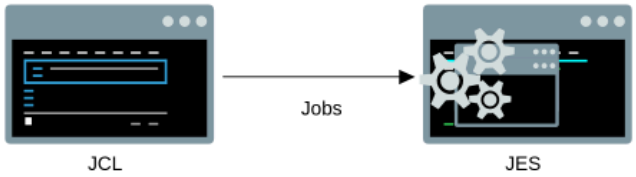


The JES subsystem runs as a started task within the z/OS system, controlling the scheduling, priority, input, and output of jobs.

These jobs include long-running processes that are commonly thought of as batch jobs, and subsystems that can run as jobs, including CICS, Db2, UNIX Systems Services, and other intrinsic processes.

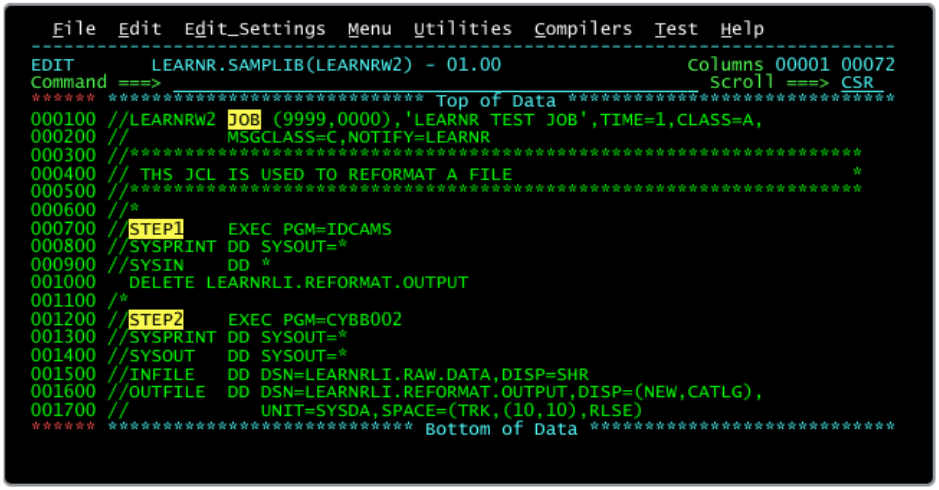
Jobs are submitted to JES which, depending on the job's parameters such as class and priority, and on system load, may put them in an input queue, execute them immediately, wait until resources are available, or reject them because of incorrect information.

JES will then ensure that the job is run, monitor it while it is running, and handle its output when it finishes.



To define jobs to JES, z/OS provides a language called Job Control Language (JCL), which is made up of highly structured statements that:

- Identify the job, its accounting information, priority, and scheduling information
- Define the program or procedure to be executed
- Define resources to be allocated for use by the executing program, such as data sets
- Define conditions that can control job flow



JCL is unique to the IBM mainframe environment. The example above is JCL that is stored within a member of a partitioned data set, and contains:

- A JOB statement that describes the characteristics of the job to the system
- A step called STEP1 that invokes the IDCAMS program
- A step called STEP2 that invokes the CYBB002 program

Statements are run sequentially, meaning that STEP2 cannot run before STEP1 has completed.

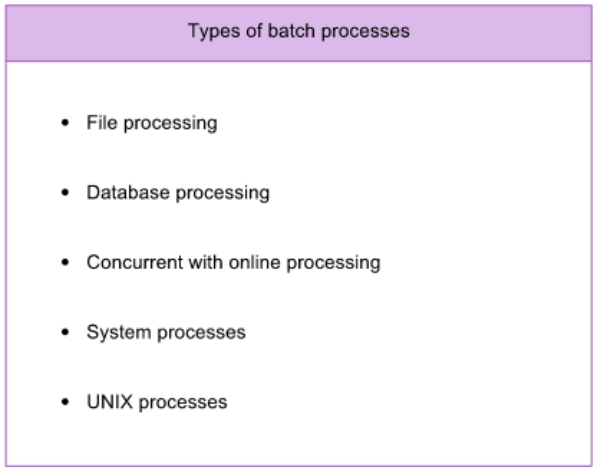
```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       LEARNR.SAMPLIB(LEARNRW2) - 01.00               Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100 //LEARNRW2 JOB (9999,0000),'LEARNR TEST JOB',TIME=1,CLASS=A,
000200 //          MSGCLASS=C,NOTIFY=LEARNR
000300 //*********************************************************************
000400 // THS JCL IS USED TO REFORMAT A FILE                                 *
000500 //*********************************************************************
000600 //*
000700 //STEP1    EXEC PGM=IDCAMS
000800 //SYSPRINT DD SYSOUT=*
000900 //SYSIN    DD *
001000   DELETE LEARNRLI.REFORMAT.OUTPUT
001100 /*
001200 //STEP2    EXEC PGM=CYBB002
001300 //SYSPRINT DD SYSOUT=*
001400 //SYSOUT   DD SYSOUT=*
001500 //INFILE   DD DSN=LEARNRLI.RAW.DATA,DISP=SHR
001600 //OUTFILE  DD DSN=LEARNRLI.REFORMAT.OUTPUT,DISP=(NEW,CATLG),
001700 //            UNIT=SYSDA,SPACE=(TRK,(10,10),RLSE)
****** ************************** Bottom of Data ***************************
```

JCL is also used to identify the resources required by the programs you want to execute. For example, in step STEP2 the input data is indicated by the INFILE DD statement and the output data will be stored in a file indicated by the OUTFILE DD statement. The SYSPRINT DD statement contains output describing the success of the utility functions that have been performed within the program, while the SYSOUT DD statement is output in the form of system processes performed within the step.

```
HEADER
  STEP 1 - RUN PROGRAM
    RESOURCES
  STEP 2 - RUN PROGRAM
    RESOURCES
  STEP 3 - RUN PROGRAM
    RESOURCES
```

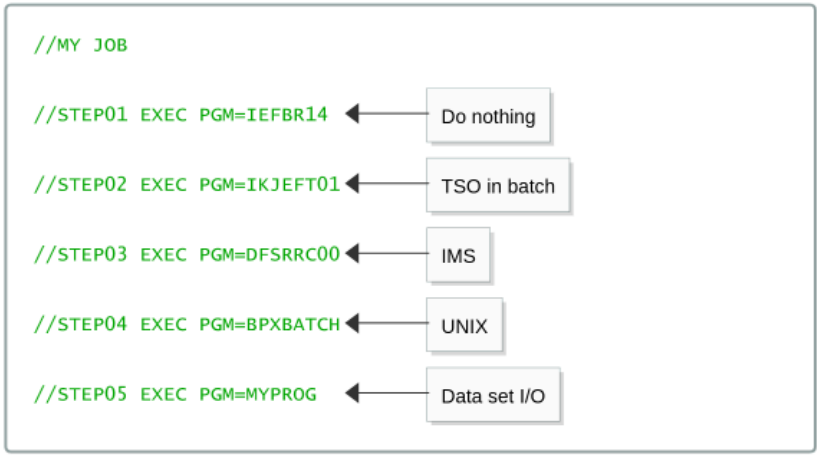
JCL can be complicated to read and write because conditional coding and multiple levels of substitution may be used.

However, when the JCL is submitted to and interpreted by JES, the result is a series of steps, each consisting of a program to be run and the resources that it needs.



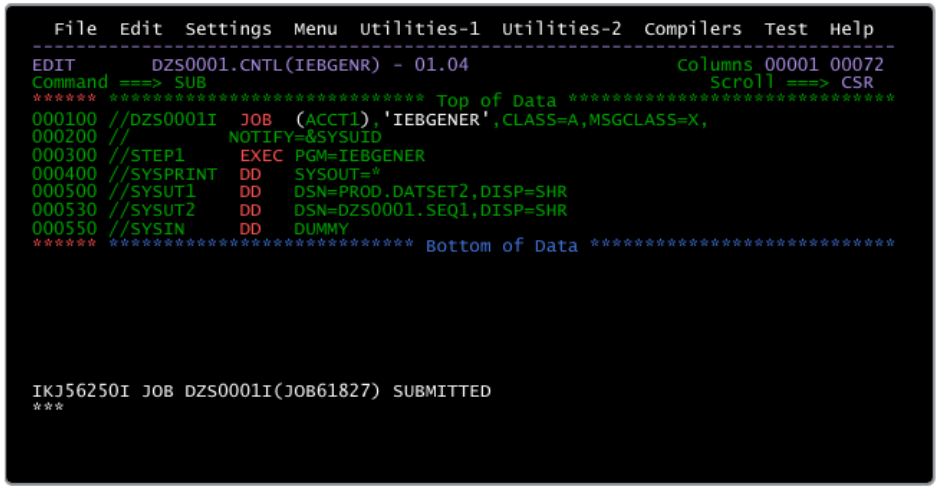
A job is made up of one or more job steps, executed in order. Each step runs a program. This program can be any type of process, depending on the application requirements.

There is even a program called IEFBR14 that does nothing. It is often used as a place-holder to enable JCL DD statements to act on data sets.



Although a single job can contain a mixture of process types, it is good practice to group processes in a manner that:

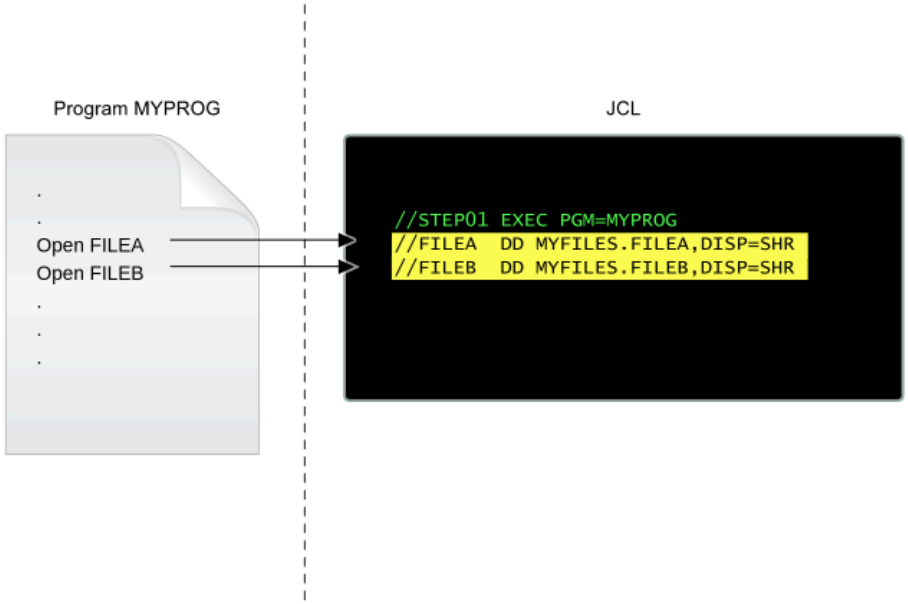
- Minimizes contention
- Reduces exposure to failure in other systems
- Enables efficient recovery



There are many ways to start, or submit, a batch job. For example, a user could use the TSO SUBMIT command to submit JCL stored in a partitioned data set.

A batch job could be submitted from within another batch job, or even from a CICS or IMS program. Batch control software can also submit batch jobs.

However, most users will use the ISPF SUBMIT, or SUB for short, command when editing the JCL for a batch job in ISPF.



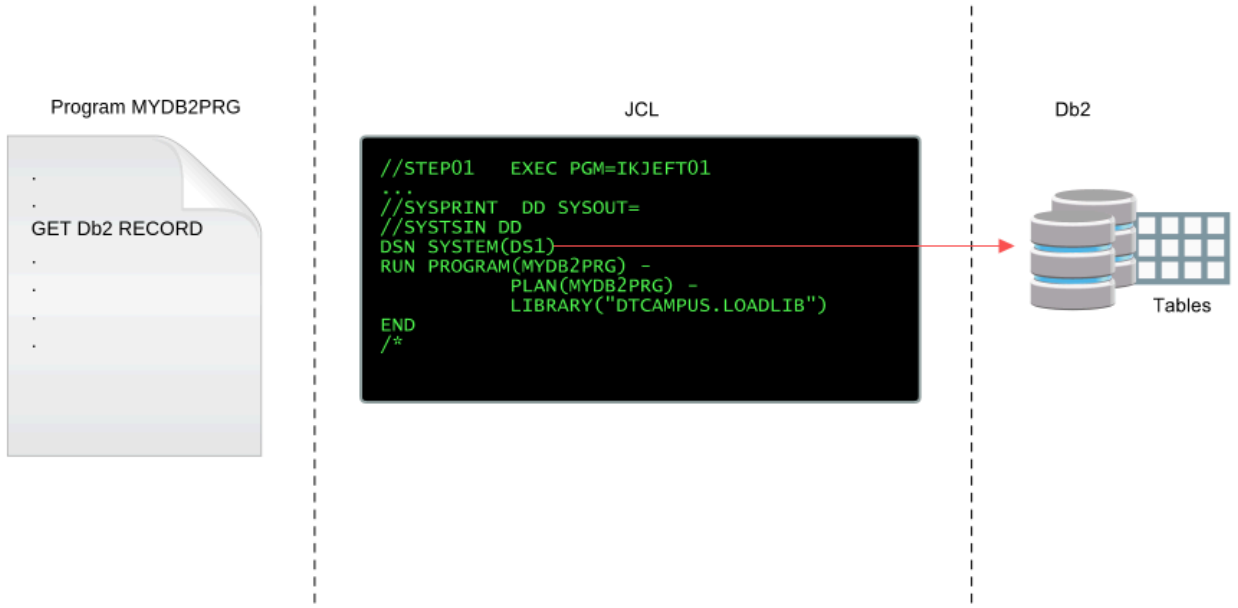
The most straightforward batch processes are those that run programs which fully control their own input and output. These programs are run explicitly and will access data sets that are defined within the JCL.

Such data sets are known to the program by a DDNAME that is assigned to a data set or resource within the JCL by a DD statement.

Database systems manage the resources and data sets under their control so when a batch application is designed to access data contained in a database, it is usually run under the control of the database manager.

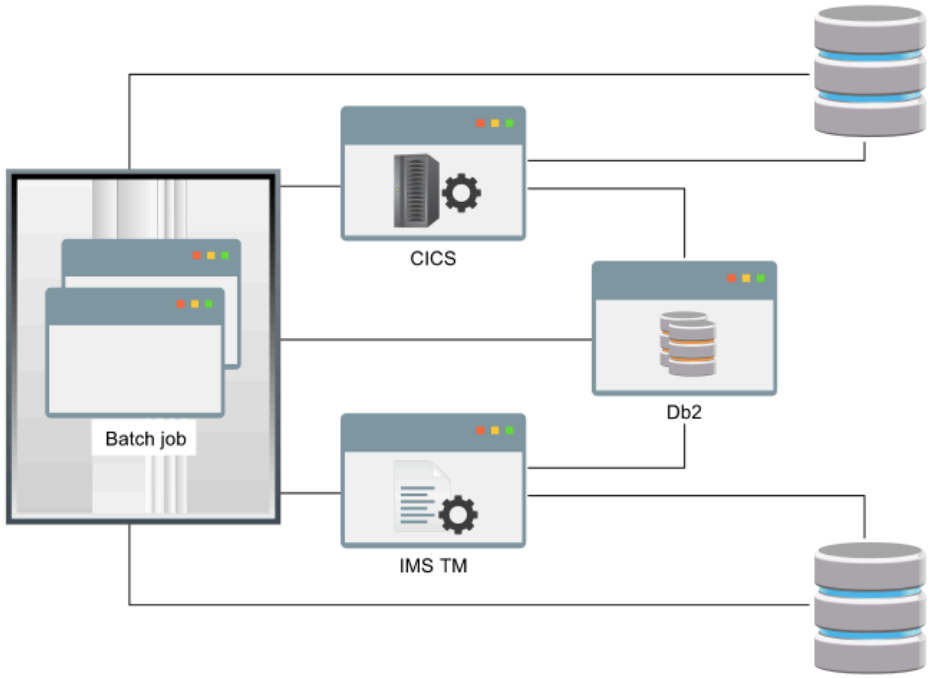
With an IMS batch program, the JCL executes the batch IMS interface and the application program is passed as a parameter.

IMS then controls the execution of the program and manages the resource access.



Similarly, with a Db2 batch program, programs are run under TSO in batch mode and connect to a Db2 system by the DSN command.

Db2 then runs and controls the requested program; therefore, the database management system controls the database data sets, including all aspects of sharing and concurrency.



Online systems like CICS and IMS TM have allocated and controlled data sets. If it is necessary for a batch job to access the same data sets, the applications and JCL must be constructed to work with the online system. There are many ways to provide for this, including batch interfaces to CICS and IMS, use of common connectors such as MQ, the use of third-party sharing systems, the use of Db2 which treats online systems and batch processors as equals, or the building of applications that allocate and release resources to enable sharing.

With forethought, it is possible to create application systems that provide concurrent 24 hour, seven days a week online access and batch processing of data.

CICS

```
//CICSA   JOB MSGLEVEL=1
//.....
//CICS       EXEC PGM=DFHSIP,REGION=&REG,TIME=1440,
//             COND=(1,NE,CICSCNTL),
//             PARM='START=&START,SYSIN'
//....
```

TCP/IP

```
//TCPIP  JOB MSGLEVEL=1
//STARTING    EXEC TCPIP
```

RACF (started task)

```
//
//.....
//RACF    PROC
//RACF    EXEC PGM=IRRSSM00,REGION=0M
```

UNIX Systems Services (start task)

```
//
//.....
//OMVS    PROC
//OMVS    EXEC PGM=BPXINIT,REGION=0K
```

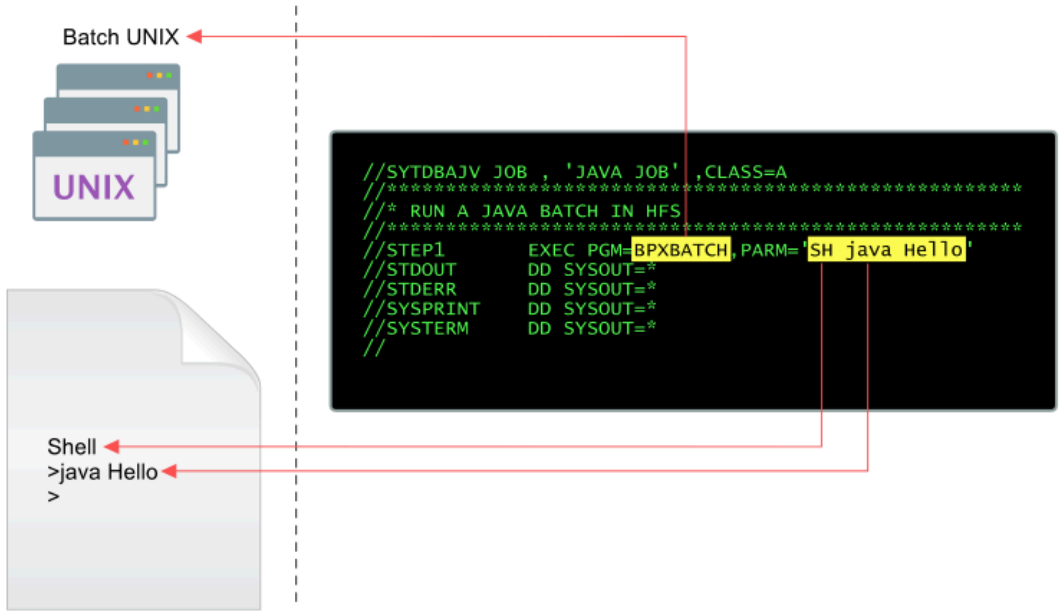
Db2

```
//DSN1MSTR JOB MASGLEVEL=1
//STARTING    EXEC DSN1MSTR
//.....
```

```
//DSN1DBM1 JOB MSGLEVEL=1
//STARTING    EXEC DSN1DBM1
//.....
```

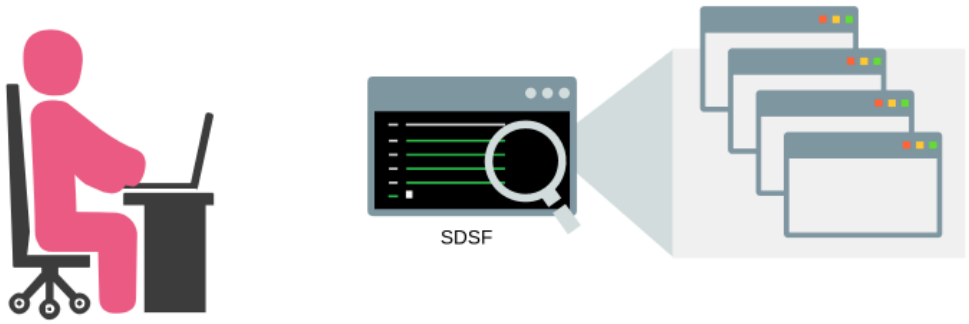
We have been referring to system processes like CICS and Db2 as if they are essentially different from batch jobs. In fact, although they may provide user interfaces and interfaces to other systems, from the system's point of view they are just long-running batch jobs whose identity and resources are also defined by JCL.

These processes may be submitted differently at the system's start up or from the console and they may be called started tasks, but in many other aspects, they can be treated as batch jobs.



UNIX System Services enables the initiation of background processes in the same manner as most UNIX systems, but we often need to run batch UNIX processes, such as Java programs or even WebSphere from z/OS.

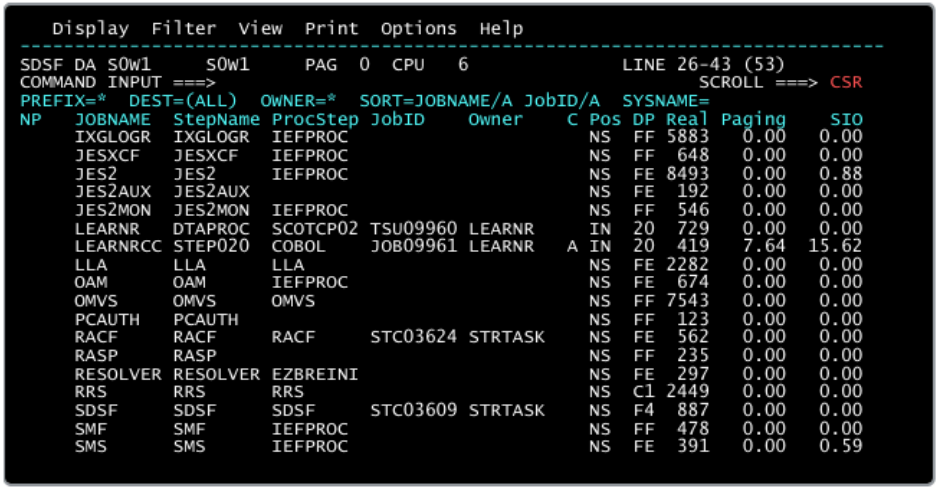
This is accomplished by running the UNIX shell in batch and then running UNIX processes, as in this example of a Java process.



An ability that is just as important to any batch system as being able to submit and run jobs is the capacity to control and manage jobs.

There are many interfaces available, ranging from ISPF Outlist to the GUI interfaces supplied by enterprise management systems like CA Workload Automation CA 7 Edition and IBM Tivoli Workload Scheduler.

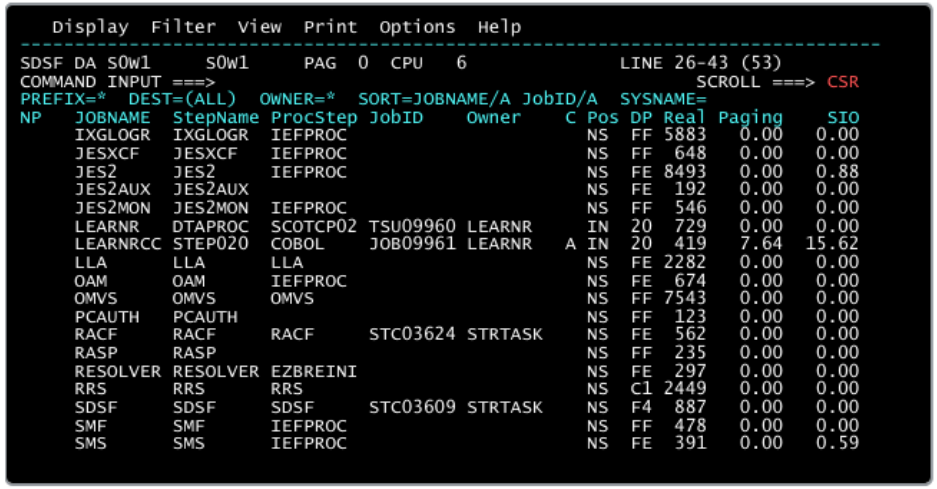
One of the most commonly used is called System Display and Search Facility (SDSF). SDSF is an optional feature of z/OS, but is used by most z/OS sites.



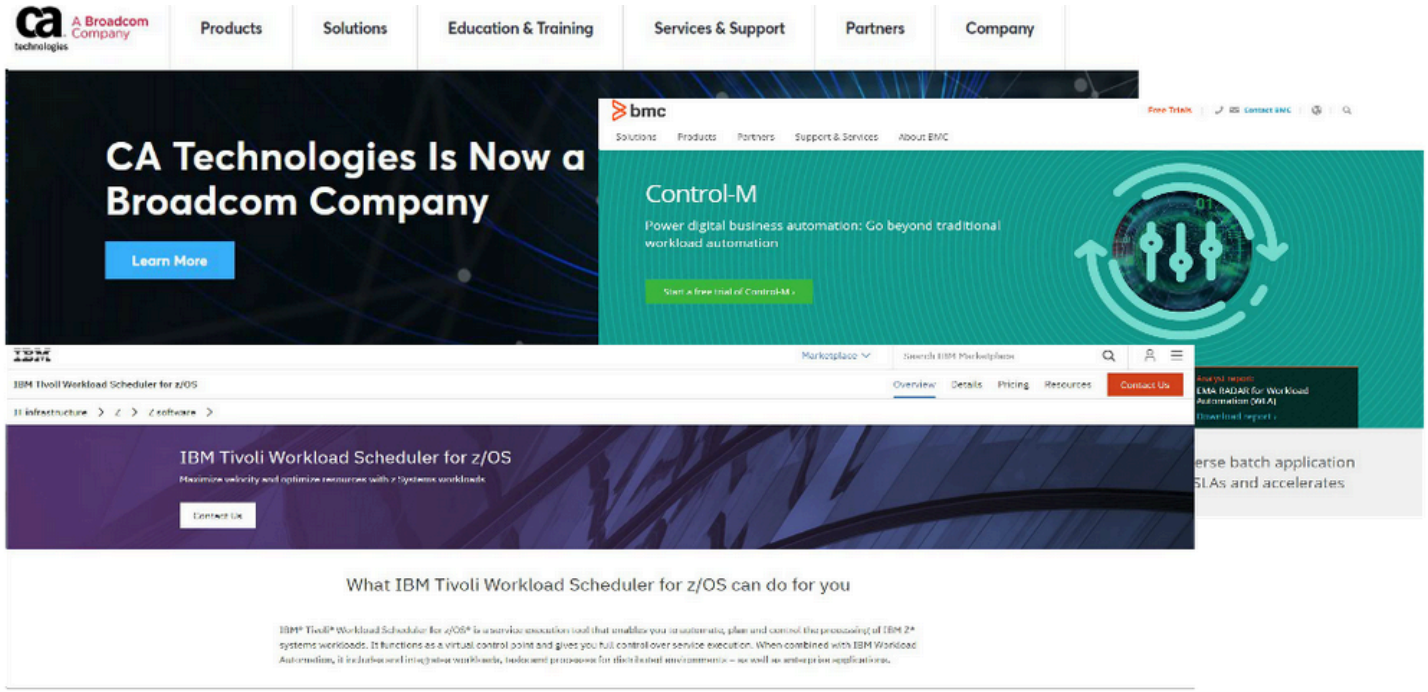
This is an example of the SDSF interface from an active system. It shows:

- Systems processes that are displayed in white
- An online TSO session (LEARNR) in the middle of the screen with a JOBID beginning with TSU
- A batch job (LEARNRCC) below the item above with a JOBID beginning with JOB

It also gives information such as statistics and priorities.

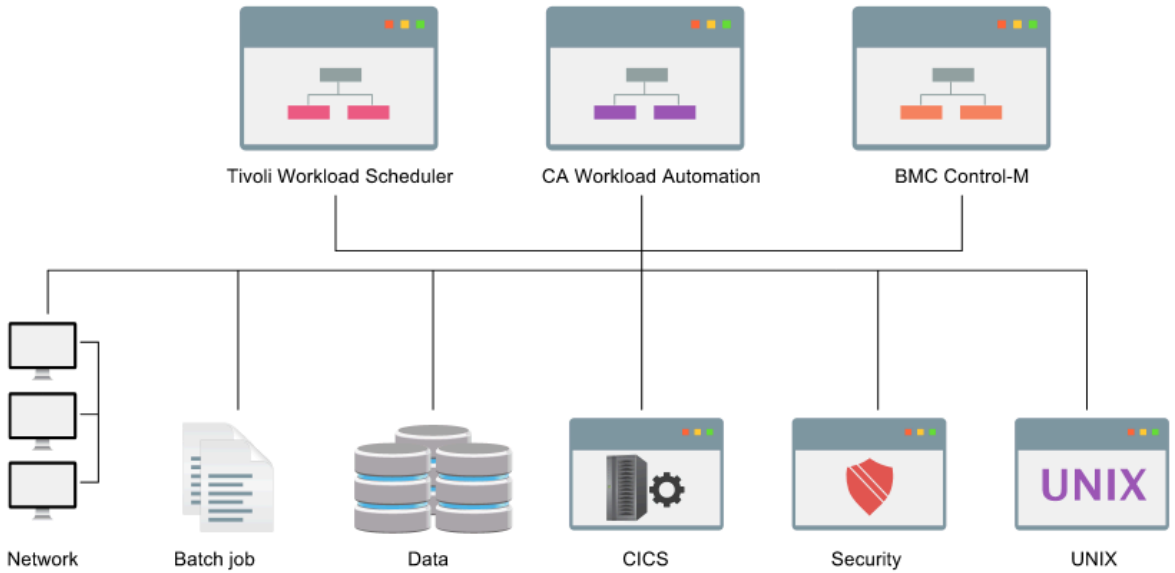


Even more importantly for users with the right security, this interface enables access to more detailed information about each job or process, and the ability to cancel jobs if need be.



SDSF and similar facilities enable you to control individual batch jobs, but as IBM enterprise systems are designed to handle thousands of processes, many enterprises have installed other batch and system management facilities.

CA Workload Automation CA7 Edition, BMC Control-M, and IBM Tivoli Workload Scheduler are examples of software products to automate and manage the submission, management, and monitoring of jobs.



In today's organizations, the IBM enterprise system is part of an integrated environment of networks, PCs, servers, and applications.

Enterprise management systems must therefore be expanded and developed to facilitate the control of these systems.

Summary

Batch Systems

In this module, you discovered how the IBM enterprise environment is built to enable maximum batch processing in major corporations.

You should now be able to:

- Identify the Function of Job Entry Subsystem (JES)
- Identify How Job Control Language (JCL) Controls the Running of a Program
- Define Batch Processing
- Define System Display and Search Facility (SDSF)