EXECUTIVE OVERVIEW

Jol is a fully programmable, English-like high level command language that increases your performance while reducing the costs of computing resources.

It is a high level language that enables Users to detail the true characteristics of their jobs in simple, easily learnt statements.

Jol allows the commands required by the Operating System to be expressed concisely, logically and in an extremely simplified manner.

Jol is a proven alternative to IBM's complex Job Control Language - JCL - and offers vastly increased flexibility while substantially simplifying the whole command procedure.

Jol allows all data set attributes to be stored in a data base - when Jol requires information about data sets, it automatically uses the details from the data base.

The Jol System includes a compiler and execution monitor that can run your work under TSO, or in Background, with or without JCL. It also includes a Scheduling and Networking Facility that can be used with your proven JCL procedures.

Jol complements or can replace IBM's OS JCL, TSO CLISTs and ISPF Panels.

Jol offers many advantages over your current mode of operation whilst offering tremendous cost saving, viz:

- Increased programmer productivity
- Greater machine throughput
- Complete management supervision and control
- Independence from Operating System and Hardware

AREAS OF USE

- End User
- Programming Maintenance and Development
- Production (i.e., operations, scheduling)
- Management Control

MAJOR FUNCTIONS OF JOL

- Saves hours of programmer time
- Saves hours of machine time
- Schedules and Networks jobs
- Enforces installation standards
- Enhances utilization of existing software
- Reduces operating costs associated with data processing
- Provides many new facilities

PRACTICAL ATTRIBUTES OF JOL

- Easy to learn
- Easy to use
- Easy to change
- Flexible
- Powerful
- Legible
- Concise

With Jol the complexities of OS JCL disappear.

Jol Command Language procedures can be written in a logically structured manner using simple problem oriented commands. Jol statements have a free format and a simple grammatical structure.

Should an error occur, clear and intelligible diagnostics are issued.

CONVERSION TO OS

Jol can assist substantially in reducing costs and time requirements during conversion to OS. Jol minimizes the traumas that you normally experience with this conversion and provides a dramatic reduction in overall project costs.

Convertors are available to convert OS/JCL, Fujitsu X8 JCL and DOS JCL to Jol.

SYSTEM OVERVIEW

Jol is an English-like high level command language which uses a free form command structure to communicate with OS and effectively control programs and events.

MAJOR SYSTEM BENEFITS OF JOL

- Provides programming with a dynamic and procedural high level command language.
- Provides extensive logic to allow programs to be event driven.
- Aids and maximizes use of existing software products.
- Reduces time and effort required to create an OS jobstream or equivalent.
- Handles OS syntax requirements automatically.
- Enforces User defined standards with no hooks to the operating system.
- Supports all OS JCL and TSO facilities.
- Allows the creation of clean correct jobstreams by persons with little OS expertise.
- Provides programmers with a scheduling tool.
- Improves machine efficiency in processing jobs.
- Provides a means to centralize program and data set definitions, thus removing repetitiveness and reducing code.
- Provides thorough error detection and job validation.
- Reduces operating system and hardware dependence.

OS JOBSTREAMS

The execution of jobs in background or foreground is an ongoing function. To create Jol command language procedures that execute either in background or foreground, the User need only gather the variable information pertinent to that job, such as, job name, program name, and file names. The variable input information can be simplified through the use of formatted screens provided with Jol or built by the User with the Jol **PANEL** instruction. The formatted screen allows data values and variable input to be specified without concern for OS syntax or keyword requirements.

DATA BASE OF DATA SET ATTRIBUTES

Version 5 of Jol adds a data base of data sets to Jol. With this facility, Mainframe jobs become as easy as running programs on MS/DOS or UNIX.

For example, to execute a program called **UPDATE** you can code:

```
Exec `Update Payroll.Master(0),
Trans.Action(0),
Payroll.Master(+1);

If Update = 0
then do;
        Catalog Payroll.Master(+1);
        Submit Job2;
end;
else Stop 'Error in Job';
```

The addition of the Jol data set data base facility:

- Greatly reduces the training required for new JCL Programmers and increases their productivity even further.

- Further reduces the amount of coding required to run a job - most JCL jobs are reduced by an average of 75% or more. Executing a program now only requires writing a line or two of code for each program.

- Provides a set of commands that will function in a similar way on Mainframes, Personal Computers and Unix systems.

- Assists in creating an environment whereby the Data Manager can have more (and separate) control over data set placement and other attributes.

- Adds full VSAM support.

SYSTEM FEATURES

The system features that contribute to increased program productivity, enhanced data center operations, and efficient machine usage are outlined below.

INCREASED PRODUCTIVITY

Programmer time spent on maintenance and development is a necessary cost in running an OS data center. However, a significant portion of this time is spent on performing non programming related functions, thereby increasing costs. Jol provides the following features which substantially reduce the time and training required to carry out these functions.

- **Ease of Use**: Jol's English-like free form and procedural format makes it easy for programmers and non-DP personnel to use.
- **Scheduling and Networking Capability:** Jol automatically submits jobs on particular days. Networks can easily be created allowing jobs to run in parallel, or concurrently. Jol's Scheduling and Networking Facility works equally well with Jol and JCL.
- **Dependent Jobs Facility**: Jol enables Users to submit subsequent jobs at any point within the job currently executing. This process provides a facility to effectively schedule dependent jobstreams. The dependent job is not submitted or placed on the system job queue until the current job has reached a point where it is desirable or safe to run the dependent job. By using this technique, a linkage can be created which controls the sequence of running a number of jobs.
- **Allocation, Reading and Writing Data Sets:** Jol has **ALLOCATE, READ, WRITE** and **CALL** instructions similar to TSO. A job can read a data set, examine it, and create Jol code based on the information in the data set. The **CALL** instruction also allows any program to be executed, and the return code tested during the Jol Compile. Thus, job streams can be created based on a program return code.
- **Relational Capability**: Jol's powerful macro facility is a set of instructions stored in a macro library that creates Jol instructions tailored to the Users specifications. This facility provides a transparent error free interface, for the programmer and User, to existing software products. Additionally, Jol macro instructions, such as, **SORT, COPY, PRINT, COMPILE** and **COMPRESS** are available to reduce the redundant labor efforts normally associated with these common routines.
- **Change Facilities**: Jol's comprehensive command structures reduce the labor intensive practices associated with OS jobstreams. With Jol, all program and data set definitions are defined once. This feature dramatically reduces the time required to implement changes.
- **Registration Capability**: Jol enables Users to record all attributes of a program once (i.e. name, language type, source location, function, load library name, file names, compiler options, link options, autocall libraries, and copy libraries). This facility provides standard enforcement and swift accessibility to the attributes of the program.
- **Library Facility**: Jol provides an include capability similar to the COBOL copy verb which allows common or static code to be stored in the **INCLUDE** library. This library can contain program, data set, and data card definitions which can optionally be defined in a single member or as individual members. The current values of these definitions can be defined with symbolic variables, thus, allowing temporary changes to the procedures and enabling multiple Users to share the member containing these definitions.
- **Logic Capability**: Jol contains **IF, THEN, ELSE, AND/OR** and **DO** logic which provides the programmer with a powerful tool to manipulate programs in jobstreams based on variable conditions, calendar information or ABEND situations.

ENHANCED DATA CENTER OPERATIONS

An efficient production environment is vital to the smooth operation of a data center. Jol provides the following powerful features to achieve this end.

- **On-line Data Entry For Job Submission**: Jol enables the Users to create '**PANELS**' or pre-formatted screens to allow data to be input to Jol. Effective use of this feature can insulate the User from most of the tiresome details of entering information pertinent to a job(s), and at the same time, provide an efficient low overhead method of doing so. Facilities are provided to display text; display text and allow replies to be entered; and to display text with default replies.

- **Calendar Facility**: Jol has an in-built calendar which can be used to select jobs or parts of jobs on particular days, dates, months, or years.

- **Error Detection And Job Validation**: Jol applies a comprehensive set of validation checks to all jobs. In addition to the normal syntax checks which are applied to each Jol statement, the internal consistency of the complete job procedure is examined. If inconsistencies are detected, then Jol either corrects the procedure or does not submit the job for execution.

    - The validation procedures used by Jol are oriented toward the total job rather than toward individual steps. This eliminates the situation in which machine time used in earlier job steps is wasted due to a control language error in a later step.

    - The comprehensive validation and error detection procedures of Jol are supported by clear and informative diagnostic messages. This ensures that if errors do occur, then corrective action can be taken quickly with minimal effort.

- **Rerun/Restart Facility**: Jol, by using symbolic variables, provides simple instructions to allow total flexibility in organizing restarts. This facility allows the job to commence execution at a point other than the beginning of a job. Additionally, execution of a job can be stopped at any point other than at the end of a job.

- **Standards Enforcement**: Jol provides the means for consistent enforcement of standards and for the level of enforcement required. User supplied routines can be incorporated into the Jol processor when it is generated. Jol makes available to these routines all the job control information pertinent to a particular job. These routines can examine this information and override any element if necessary.

- **Communication Ability**: Jol provides the facility to communicate with the Operator and User and to place messages on the system log of the job. This communication may be in the form of a simple instruction or may request a specific reply.

- **Easy Overriding**: Any Jol definition can be overridden. To override the job, program, data set, and symbolic definitions, another definition is simply coded before the existing definition. The first definition is given preference. In some cases no modification of the Jol text may be required at all.

- Similarly, generic changes resulting from alterations in the specifications of a public or shared program may optionally be accomplished by changing a catalog text in the Jol 'INCLUDE' library or by changing a macro prototype in the Jol 'MACRO' library.

- **Using Backup Computers:** The global changes necessary when processing is transferred to a back-up machine with a different hardware configuration can be performed automatically by the Jol processor.

EFFICIENT MACHINE USAGE

Efficient machine usage and maximum use of the operating system resources is essential in achieving a smooth running and low cost data center. The use of Jol provides more efficient utilization of machine resources.

- **Catalog Management**: Optionally, Catalog searching is performed at submission time and is minimized. Jol searches the catalog once regardless of the number of times a data set is used.

- **Tape Access Management**: Jol ensures that tape data sets are left positioned correctly in cases where several data sets are created or read on one tape volume.

- **Reduction In Job Steps**: When using the JCL Generate Option rather than Dynamic Allocation, Jol uses a transient Scheduler/Initiator so all CATALOG and SCRATCH statements can be performed in one OS step. Sometimes, several programs can be executed in the same OS job step, further reducing the OS Initiation/Termination overheads.

- **Reduced Overhead In Symbolic Translation**: Because all Jol definitions are defined once only symbolic usage is greatly reduced. Additionally, all symbolics are resolved prior to submission greatly reducing the OS Reader/Interpreter overheads when using the JCL Generate Option.

- **No Hooks To The Operating System**: Installation of Jol is simple and does not require an IPL. Additionally, User exits can be incorporated into Jol to enforce standards and provide User defined facilities without concern to the operating system. This ensures that an operating system upgrade does not necessitate reapplication of these exits.

SUMMARY

Jol's facilities and savings have an immediate, positive impact on all areas of the data processing environment. Management, development, operations, scheduling and software, and functional areas realize that immediate benefit. Jol does not require any overnight conversion as JCL and Jol jobs can co-exist. Installation is rapid, and thorough training only requires three days.

The Jol manuals demonstrate both the power and simplicity of Jol commands. The characteristics of the language and the reasoning behind its structure are also described.

FURTHER READING

Other Introductory documentation includes the Jol Answers to Questions, the Jol Concepts and Facilities Manual and the General Information Manual. The Concepts and Facilities Manual contains discussions on using Jol definitions, instructions and macros, and illustrates the way in which Jol allows the natural expression of commands in an English-like manner. Also exemplified is the highly effective use of Jol's concise, brief commands.

The Jol Concepts and Facilities Manual also introduces examples of Jol applied to practical job control functions. The Automatic Scheduling section discusses the adaptation of Jol in another practical manner to encompass an area of responsibility not usually considered within job control facilities. A section on additional facilities expands these ideas to show Jol's flexibility and capability in adapting to a variety of facilities previously not available. Some of these include: Language extension through Jol macro commands, compile time facilities, symbolic variables, generation data groups and automatic restart.

Further supplemental materials are available which document Jol's cost effectiveness. Case studies, evaluation reports and user documentation easily provide specific evidence of Jol's claims to 67% reduction in manpower requirements over conventional JCL. Other functions and capabilities, like automatic restart and programmed application schedules, are more difficult to evaluate in terms of time or dollar savings and are considered as having the greatest merit by many existing Jol Installations.

Jol, a command language with the facilities and concepts of tomorrow, is ready today; are you?

Jol IS EASY TO INSTALL

Jol DOES NOT REQUIRE "OVERNIGHT" CONVERSIONS

Jol OPERATES UNDER TSO, MVS, and F4(FACOM)

Jol JOBS AND JCL JOBS CAN CO-EXIST