CERN - DATA HANDLING DIVISION
DD/73/28
R.D. Russell,
P. Sparrman,
M. Krieger*
August, 1973

O R I O N - THE OMEGA REMOTE INTERACTIVE ON-LINE SYSTEM

(to be presented at the International Computing Symposium 1973, Davos, Switzerland, 4-7 September 1973)

(* MPS Division, CERN)

DD-jh

O R I O N - THE OMEGA REMOTE INTERACTIVE ON-LINE SYSTEM

R.D. Russell*, P. Sparrman*, M. Krieger**

* Data Handling Division, CERN, Geneva
** Machine Proton Synchrotron Division, CERN, Geneva

ORION is a system which permits the manipulation of files, records and characters, remote job submittal and retrieval of output files including the direct loading of remote on-line computers. The system uses the computer hardware of the OMEGA project at CERN, and is designed to assist researchers in development and debugging of their programs.

INTRODUCTION

ORION is designed to give the users of mini-computers that are on-line to physics experiments in the CERN OMEGA project remote access to the file storage, program development, and batch processing capabilities of the medium-sized central OMEGA computer, as well as program loading into the remote machine. ORION consists of a program in the remote computer that interacts with one or more users at teletype or display terminals, and a program in the central machine that communicates with the remote machines via data links in order to manage files on the large central disk, to submit jobs, and to make status inquiries on behalf of the on-line user. The commands include a complete set of text-editing and file manipulation facilities in a format that is easily read and understood.

This paper describes the design philosophy and implementation of the ORION programs. The OMEGA project computing system, which consists of a CII 10070, an EMR 6130 and four 16k PDP-11/20's, is described elsewhere (1), as is the data link software (2) and hardware(3).

USER DESIGN CONSIDERATIONS

The design of ORION will be discussed from two points of view: that of the users and that of the implementers. The primary emphasis at the user level was on what we call 'human engineering'. Much thought and effort went into designing a system that could be used and understood easily by inexperienced users, yet would be powerful enough to satisfy more sophisticated users. The design is based primarily on the authors' experience with WYLBUR (4,5), although other editors were also considered (6,7).

The first step toward our goal was to choose meaningful English words for the commands typed by the user. The basic commands are a small set of action verbs (after all, they are commands) that convey to an inexperienced user some idea of their function. Computer jargon and cryptic abbreviations were carefully avoided, and each command performs a single function - complicated option strings that cause a wide range of different actions for the same command word is a classic example of bad human engineering. We also felt that there should be a 'good resolution' between commands so that one command would not be confused with another because the two words looked or sounded alike.

The fact that every command is an English word makes it very easy to explain the command set to a novice and to give him confidence that he is really using the command he means to use. For experienced users who find it burdensome to type long words for familiar operations, ORION will accept any partial spelling of a command word that is unambiguous. For most commands the minimum spelling is simply the first letter, due to the 'resolution' between command words. For example, 'E', 'ED', 'EDI', 'EDIT' are all accepted as the EDIT command, but the REPLACE command requires at least 'REP' to distinguish it from RENUMBER, 'REN'. Neither 'R' nor 'RE' are acceptable abbreviations. It is important to note that all the letters typed by the user must be correct or the command is rejected. For example, 'REPR' is not interpreted as 'REPLACE', since the 'R' does not match the 'L' (even though 'REP' is unambiguous). If the user were a novice who had intended 'REPR' to mean 'REPRODUCE' (a non-existent command), he would be most confused to find that ORION had REPLACED what he had meant to copy.

This example illustrates another design principle - that the syntax should be such that an acceptable input should have the same meaning for both the user and the computer. The notation should not have several interpretations as far as a novice is concerned - it should really represent what it seems to represent. The converse proposition is also essential - that all strings acceptable to the computer should look meaningful to the user. We felt that the horrible hieroglyphics of many text editors were another example of bad human engineering. When being shown other text editors, we were always impressed if the experienced demonstrator was unable to decipher the meaning of a legal input string that was either given in the manual, typed by someone else, or typed by himself earlier in the session. ORION avoids the hieroglyphics problem by the use of free-field input with optional punctuation characters (blank, comma, equal sign), and keywords such as 'IN', 'FROM', 'TO', 'BY', 'FOR', 'COLUMN', which precede each option, as in the examples:

```
SEARCH 10/25 FOR 'SUBROUTINE'
CHANGE FROM 'INTEGER' TO 'REAL' IN 1/9 COLUMNS 7/9
RENUMBER ALL TO 10.2 BY 5.52
```

These keywords allow the options to appear in any sequence, and are much easier to use and remember than special characters or positional dependencies. They also result in an easily read typescript of the editing session.

One of the advantages of any terminal system such as ORION is that the system can interact with the user so that he learns while using it. The ORION HELP command is essentially a self-tutorial in the use of ORION. This command, which includes instructions on how to use it, gives the user on-line access to the entire ORION users manual in a hierarchical manner from general command descriptions to detailed examples with full use of all options. For example, at any time a user can type:

```
HELP ADD
```

and have printed at his terminal an explanation of the general use of the ADD command plus instructions on how to obtain more specific details. The user should therefore never be at a loss as to what to do next just because no one is around or because he left his manual at home. The system is designed to be self-instructing.

An interactive system by definition should always react to the user, and ORION is designed so that the terminal never 'goes dead' during the processing of a user's command. The syntax analysis of each input command, which is done in the PDP-11 and hence is virtually instantaneous, results in either an error message or a semicolon response indicating 'syntax acceptable, processing begun'. A message is also typed at the termination of all file-handling, remote job submittal, and status commands. If the processing of a particular command takes a long time without teletype output (for example, an unsuccessful SEARCH of a large file), an 'idle-sync' pulse is issued to indicate that the system is still 'alive'.

The response produced by ORION to any user action is designed so that the typescript of the session will be a complete log showing clearly the sequence of actions and responses that occurred. Input and output lines that are cancelled by the user are indicated by three trailing dots. Whenever a command is aborted by the user before it has finished, three up-arrows are typed.

Foolproof error detection consumed most of the coding effort that went into ORION. All errors result in a meaningful message to the user, not just a cryptic error code that has to be looked up in a manual. There is never a loss of files or previous editing due to a detected error, since in general ORION recovers to the state it was in before it started the processing that led to the error. As a further safety measure, all additions, deletions, and editing are done on a workfile which must be explicitly saved to make it permanent. This reduces the possibility of catastrophic errors, since permanent files are never modified, and are only replaced or erased by file commands that explicitly name the file.

All files used by ORION consist of a set of numbered records, one record per printed line of text. Commands specify a specific line or group of consecutive lines on which they are to operate. Any line in the file can be referred to at any time - there is no need to process a file sequentially from beginning to end, as in FOCUS for example. There is also no need for the user to remember 'where he is' in the file, as is true in editors without line numbers. The line numbers are in the form of 'WYLBUR numbers' (4,5) - decimal numbers in the range .001 to 99999.999 with at most three digits to the right of the decimal point. The fractional digits, which are printed only if they are non-zero, give a very obvious indication of inserted lines, since files are usually created with integral line numbers. Any part of the file or the entire file can be renumbered at any time, but only by explicit command from the user. (There are editors that automatically renumber as lines are inserted and deleted, which implies that the numbers are useless, since the user is never quite sure which number goes with which line.)

TECHNICAL DESIGN CONSIDERATIONS

We decided from the beginning to use existing software whenever possible. On the CII 10070 ORION runs as a user job under the normal SIRIS 7 operating system (8). It can therefore coexist with all normal operations, an important consideration for smooth, rapid development. ORION also uses only the standard file formats and access methods available in SIRIS 7. This means complete compatibility with all other processors in the system. This yields enormous advantages - no extra effort to develop file formats, no costly overhead to convert between formats, no confusion for users, and full use of all system facilities even during debugging. SIRIS 7 includes a standard compressed format that can be read directly by the language processors (FORTRAN, PL-70, assembler), and this can also be read and written by ORION if desired in order to reduce the disk space required by large files.

On the PDP-11 ORION utilizes a multi-tasking monitor that had already been developed at CERN to drive display terminals (9). This permits the same PDP-11 to be used simultaneously for driving both graphics displays and ORION terminals. The syntax analysis of commands is done using the table-driven parsing routines that were written as part of the PL-11 compiler development. Communication with the CII is done with link packages that are standard on both machines.

ORION is written largely in high-level languages. Most CII routines are written in FORTRAN, although some Assembly language coding was necessitated by the lack of good I/O and error handling facilities in FORTRAN. All the PDP-11 coding is written in PL-11, an intermediate-level language designed and implemented at CERN (10). The advantages of this decision are obvious - faster, more accurate coding, easier debugging and modification, and self-documentation of the programs.

We assumed from the start that neither ORION nor the operating systems would ever be completely free of crashes, and hence recovery procedures were designed in as a fundamental part of ORION. The tables describing the status of all terminals and files in use are kept on the CII disk, and are rewritten to disk each time the status changes. Reading these tables from disk during restart painlessly restores the 'memory' of ORION - nothing is lost.

Of primary concern to the user is the survival of his workfile through a system crash. This is done using standard system facilities in a straight-forward manner. All workfiles are kept as permanent files on an account to which only ORION has access. The operating system does not touch permanent files during the restart of either ORION or SIRIS 7, and the recovery procedure in ORION simply reopens these files based on information in the status tables that are also recovered from disk. This technique required very little effort for the implementers, yet ensures with high probability that all work in progress will be recovered. The most that is ever lost is the current command of one user. Use of standard files also made debugging faster and easier. The fact that the workfile account occupies a fixed-head disk on a channel separate from that used by the disk packs containing permanent files produced high performance as an added fallout.

The division of labor between the PDP and CII was decided largely on the basis of the recovery considerations mentioned above and the facilities available on the two machines. Good recovery dictated that all control and status information be kept in one central, recoverable place (the CII disk) rather than distributed throughout the system. Hence the PDP remembers very little about the history of the session, and must interrogate the CII whenever such information is required. Since the disks are on the CII, all file manipulation (USE, JOIN, SAVE, SCRATCH, RENUMBER) necessarily had to be done on that computer. This includes all error checking on the file level. The terminals are all connected to the PDP, so that all text handling, I/O formatting, and editing are best done in that machine, especially if rapid response at the level of single character input is to be achieved. The logical unit of transmission between the machines is a single line of text, which is equivalent to a single record in the file, although physical transmission usually involves a block of consecutive records for purposes of efficiency.

It is interesting to note that the CII disk is used for storing the text for the HELP command and for all error messages. The PDP-11 sends a 'GET-HELP' request containing the HELP keyword whenever the user enters a HELP command, and a 'GET-ERROR' request containing the error code whenever the user has made an error. The CII retrieves the appropriate lines of text from the disk file and sends it to the PDP for printing in exactly the same manner as text from an ordinary user file. This obviously allows messages of any length to be handled easily, and allows the files containing these messages to be edited by ORION (in effect, ORION is recursive). They can be updated at any time, with the updates becoming effective immediately. This technique also reduced the amount of work needed to implement these commands, and made their debugging extremely simple.

In order to ensure that parallel development of the different parts of ORION would be done in an orderly manner, two clearly-defined interfaces were designed in addition to the interfaces to the two operating systems. The most obvious is the one between the CII and the PDP - all transactions across this interface require an exchange of link messages. The PDP initiates the transaction by writing a buffer to the CII. For easy processing on both sides this buffer is in a fixed format. The CII processes the request by filling in empty slots in the buffer, usually with information retrieved from disk, and then sends the augmented buffer back to the PDP. Not until this has been received will the PDP reply to the user that the command has been processed. The buffer returned by the CII may contain several lines of text for printing and/or editing, or the text of an error message if an error was detected by either machine. Lines added or modified at the terminal require additional link exchanges in order to update the workfile on the CII disk. For debugging purposes, a switch in the CII can be turned on by command from the PDP in order to dump onto the CII line printer the input and output buffers and the internal tables for each transaction.

The second interface exists in the PDP between the control program and the syntax analyser. This interface is crossed only once for each command, when the control program passes an input string directly to the analyser for decoding by a bottom-up simple-precedence parser. The information gathered during this parse is placed into slots in a fixed-field buffer that is used to drive the actions of the control program in carrying out the command provided the analyser has found the input to be syntactically correct and consistent.

These two interfaces made the project very easy to manage, since the labor was clearly divided into three parts with one person on each, and easy to debug, since just dumping the buffer each time an interface was crossed quickly showed exactly who was responsible for the bug. During normal operation the PDP gives a response on the terminal each time an interface is crossed - after syntax processing a semi-colon or error message is typed, and while waiting for the CII an 'idle-sync' is sent at regular intervals. Although originally intended solely for response to the user, these outputs became one of our best debugging tools, since they clearly show which side of the interface a hangup or crash occurred, even after the debugging buffer dumps have been turned off.

A second beneficial 'fallout' arose from the design of the clean PDP-CII interface and the resulting design of the CII program as a general file manipulator. Since the CII program simply receives commands from a data link and sends the answer back, it is obvious that any program in any remote computer could easily send commands to the CII program - the interface design insured that the format of the link buffers was not dependent on the format of the teletype syntax. We therefore specified a set of subroutines called MOPEN, MCLOSE, MGET, MPUT, MADD, MDELETE, and MFIND that could be implemented identically on all three OMEGA computers and would give any user program anywhere identical access to all the ASAM and APAM file handling capabilities of the CII. These subroutines, which were written and tested for the PDP-11 and EMR 6130 in a matter of days, can be run by users having no knowledge whatever of the ORION terminal procedures (which do not even exist on the EMR).

A final interesting point is that ORION is designed to monitor its own activity on the CII by writing a record onto a special statistics file for each link transaction. This produces a complete log of all link traffic, and gives us a powerful tool for performance evaluation and fine-tuning of the system during normal operation.

USER LEVEL IMPLEMENTATION

A user of the ORION system has at his disposal a set of commands of which the major part is the 'file-editor'. In addition there are commands for remote job submittal and related functions, different utility and 'display' commands, and finally a set of 'subroutines' to handle files at the CII 10070 from the remote computer (the file-manager sub-system). This is somewhat different and will be treated separately.

A description of the detailed syntax of the commands and their function cannot be given within the scope of this paper. However, a brief summary will be given. In addition the formal syntax definition is given as an appendix for those interested.

Note: In the following paragraphs ORION command words are printed in upper case.

The user presents himself to the system by the LOGIN command in which he gives his name and an account. He may store (SAVE) files only under the account given but he can read (USE, JOIN) files on any account.

He leaves the system by the command LOGOUT which is accepted only if the workfile is cleared. This feature has been implemented as a remainder that he should save his workfile in order not to lose any work done.

After LOGIN the user has an empty workfile into which he might start entering lines of text using the ADD command. He might also USE (bring into the workfile) a file or part of file, for example in order to update it. That file could have been created by ORION or any of the compilers (in the case of a program).

Any time the user wants he can JOIN (bring into a non-empty workfile) the content of another file, fully or in part. This 'joining' can mean an extension at the beginning, at the end, or somewhere in between, the only requirement being that existing line-numbers and joined ones do not interleave or collide. In the USE and JOIN commands renumbering can be done to any line number by any step.

At any time the content of the workfile can be SAVEd (written onto) a user file (named permanent file) or a temporary file (named file that is SCRATCHed (deleted) by ORION at logout).

Also, at any time, any part of the workfile can be PRINTed on the CII 10070 line printer or PUNCHed on the CII 10070 card punch.

The user can, at any time, RENUMBER the workfile fully or in part to any line-number by any step. This process cannot however move line(s) to another position. This must be done using the MOVE command.

To delete user or temporary files there is the SCRATCH command, which also can be specified as an option on a SAVE, and to empty the workfile there is the CLEAR command, which can be specified as an option on USE or FETCH.

On the record level the user has several commands. These commands all operate on a single line or on a range of consecutive lines. They have the function indicated in the table below.

TABLE 1 - EDITING COMMANDS IN ORION

COMMAND	MEANING
ADD	adds line(s) to the workfile.
COPY	copies line(s) to another position.
DELETE	deletes line(s) from workfile.
LIST	lists line(s) on the terminal.
MOVE	moves line(s) to another position.
REPLACE	replaces the text of existing line(s).
CHANGE	changes character strings in line(s).
SEARCH	searches line(s) for specified string.
EDIT	editing within line(s).

Especially during program updating 'content addressing' is extremely useful. With the command SEARCH a user can ask for the occurance of a specified string of characters in a range of lines or in the whole workfile. He can specify whether this string should be looked for anywhere in the record, starting in a specified column or starting within a column range. All lines are listed (displayed) on the terminal. Using the similar command CHANGE a user can replace one string by another (which may be shorter, longer or equal in length). The lines are listed as they are changed.

The command EDIT is used to edit, character by character, one line at a time. During editing the user has several control keys at his disposal for keeping, inserting, deleting, and replacing characters.

For remote job entry there are two commands: PJOB, which activates any of the utility processors at the CII 10070; and SUBMIT which is a 'genuine' remote batch submittal of the workfile. The HOLD option on the submit command allows the user to retrieve his line printer output at the terminal. This is done using the command FETCH which brings the output into a cleared workfile. Related commands are LOCATE, which gives the user status information on his submitted job, and PURGE, which deletes held output files.

Use of the SHOW command gives the user reports on: i) files used in this session and their status, ii) the status, filenames, and free disk space on his account, iii) the status of his or all jobs submitted with the HOLD option on SUBMIT. The remaining commands cannot be mentioned in this limited space.

In case a user is the only user in a PDP-11 which is the case for the 'on-line' PDP-11's, he can use the LOAD command to get his remote computer loaded with an absolute load module that has been compiled and link-edited onto the CII 10070 disk.

There exists as a sub-system a 'file-manager' which allows a user program running in the remote computer to create, retrieve and in general handle files on the disk of the CII 10070.

SYSTEM LEVEL IMPLEMENTATION

The hardware configuration at the OMEGA project naturally led to the concept of using the remote computers, the PDP-11's, as front end processors. There are several advantages with such an arrangement: i) the big computer does not have to deal with typed characters or lines but can react on well structured requests, thus using less time. ii) the user is shielded from direct contact with the big computer making the system more reliable and safe. iii) the remote computer can give faster terminal response. iv) much useful work, for example character string scanning, can be done as efficiently in the remote computer as anywhere else, resulting in a smaller CPU load on the large machine.

In addition to the PDP-11's being a good choice for front end processors, the CII 10070, and specifically the SIRIS 7 operating system is very well suited for implementing ORION. In particular, SIRIS 7 has excellent built-in facilities for remote interaction, including submission of jobs to the input stream, and several file-access methods well suited to the ORION file processing. Also, SIRIS 7 manages core efficiently, using a rollin-rollout scheme. If ORION, or any other link connected job is not active within any ten second time frame, it is rolled out until the next link request arrives. Only two changes to SIRIS 7 were necessary; the first allowing ORION to access files on any account, and the second allowing output retreival after a SUBMIT with the HOLD option. Both changes were minor. Using two processors the interface must be clean and well defined. It was felt that everything that could be done in the remote computer should be done there (it is idling most of the time anyhow). The interfaces chosen are vizualized in Figure 1.

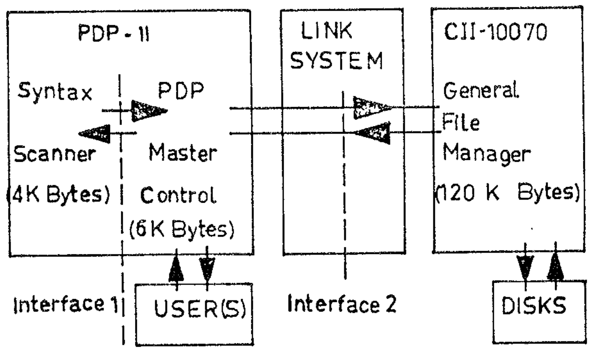
PDP-11
Syntax Scanner (4K Bytes)
PDP Master Control (6K Bytes)
LINK SYSTEM
CII-10070
General File Manager (120 K Bytes)
Interface 1
USER(S)
Interface 2
DISKS

Figure 1 Interfaces Within ORION

A command from the user is accepted by the PDP master control and the syntax analyzer is called. If the syntax is accepted a semicolon (;) is typed and a buffer containing the decoded information is given back to the PDP master control. Using this buffer and a table of default values a link buffer is built and sent to the CII 10070 as a request. The PDP then waits to receive a buffer back from the CII. Consequently the CII 10070 is always the 'slave' computer in the communication, reacting on requests from the remote partner. The requests received at the CII 10070 are completely processed before a message is sent back and another request can be accepted. This arrangement is usually satisfactory since most commands have a real-time duration of less than a second. For some commands however (reading and writing big files) the processing time can be of the order of 30 seconds, which presently blocks 'short' requests from other terminals. This will however be the object of future improvements.

Special attention has been devoted to error detection, the aim being that a user error should result in a 'no-operation' and an explanatory error message. (Due to the limited core memory in the PDP-11's the error messages are retrieved by a special request from the CII 10070 disk.) There are really two types of user errors which can occur, those causing syntax errors and those causing conflicts at the file or record level. The first type is easy to handle as it is dealt with entirely in the remote computer. Recovery from the second type is more difficult. On file operations, for example, a JOIN command causing conflicting line numbers, the workfile is not changed. On record level manipulation, for example, an ADD command, a special 'check' operation is issued to ensure the proper completion of the command.

Another area of great attention has been reliability. A crash in the CII 10070 operating system or in ORION itself does not result in loss of any file. To accomplish this easily all files are named permanent files, and they are closed at all times except when actually read or written. The work-files are always updated on disk, not only in the memory buffer, and this is done in parallel with the transmission of the buffer to the remote computer. For reliability the temporary files are named permanent files which are scratched by ORION at logout, but they can be safely recovered after a crash.

For reliability all tables in ORION are kept only in the CII 10070 and are dumped to disk every time they have been changed. Consequently, after a crash the tables are read in, the workfiles opened and a complete recovery is done.

SYSTEM PERFORMANCE AND RESULTS

Since ORION has only recently been introduced to the user community at the OMEGA project many results from system performance measurements are not yet available. A continuous supply of information on the performance is ensured as gathering of statistics is part of the ORION system in the CII 10070. For every request sent over the link a statistics record is stored containing all relevant parameters for the command, its real and CPU time consumption and its time of arrival. The last quantity is useful for calculating the typical user's 'think-time'. From the preliminary results it can be concluded that ORION can support many more than the 10 simultaneous users that can be considered the absolute maximum with the present OMEGA equipment. The information from the statistics gathered also indicates those parts of the system in which fine-tuning efforts will be worthwhile. The implementation of the currently running version of ORION has required less than two man years of work.

REFERENCES

1. R.D. Russell; The OMEGA Project: A Case History in the Design and Implementation of an On-line Data Aquisition System, CERN 72-21, (1972).

2. D. Wiegandt, P. Villemoes, R. Cooper, A.P. Jeavons; The OMEGA Data Link Software, Private Communication and OMEGA Software Development Note SW-12 (1970).

3. J. Joosten, C.F. Parkman; The OMEGA Data-link Hardware, Private Communication and Omega Hardware Development Notes HW-13, HW-16 (1973).

4. R. Fajman, J. Borgelt; WYLBUR: An Interactive Text Editing and Remote Job Entry System,

Communications of the ACM, Vol. 16, No. 5, pp. 314-322 (1973).

5. WYLBUR Reference Manual, Columbia University Computer Center, Columbia University, New York, N.Y. (1970).
6. INTERCOM Reference Manual, Control Data Co., Publication No. 60307100 (1972).
7. Instant FOCUS, Data Handling Division, CERN, (1971).
8. Procedures Systemes Sous SIRIS 7/SIRIS 8, Manuel d'Utilisation, Compagnie Internationale pour l'Informatique Document 3642E4/FR (1973).
9. A.P. Jeavons, Private Communication.
10. R.D. Russell; PL-11, A Programming Language for the DEC PDP-11 Computer, OMEGA Software Development Note SW-29 (1971).

THE SYNTAX OF THE ORION COMMANDS

General Remarks and Notation

Comma (,), blank and equal sign (=) are equivalent as keyword and option delimiters.

After the command keyword, options may appear in any order, with two exceptions: i) range specification may appear directly after the command keyword without the keyword (IN).
ii) wherever present, the <ACCT> specification must follow <FILENAME> or <USERNAME>.

Parentheses () enclose optional items.

All keywords are variable length, with the minimum length being that number of characters which make the keyword unique. For example, the command word (COPY) could be typed (CO) since (C) could also mean (CHANGE).

Care must be exercised in naming files. The parameter <FILENAME> may not be any of reserved keywords which are allowable in the SAVE or USE commands. These keywords are those in <OPTIONS> and <SAVOPTIONS>. Also not allowed are the keywords: ALL, FIRST, CURRENT, LAST, BY, TO, IN. This has been implemented this way to avoid syntactic ambiguity, and to encourage meaningful naming conventions of <FILENAME>.

The ORION Options

```
<STRING>::=        Any symbol between two quotation
                   marks. A quotation mark within
                   the string is represented as two
                   consecutive quotation marks.

<IDENTIFIER>::=    A group of up to seventeen
                   adjacent symbols, where a symbol
                   may be any alphanumeric or the
                   symbols (:) or (-). An identifier
                   may not start with a numeric.

<DIGIT>::=         0|1|2|3|4|5|6|7|8|9

<NUMBERS>::=       From one to five consecutive
                   digits.

<SMALL-NUMBERS>::= From one to three consecutive
                   digits.

<LINE-NUMBERS>::=  <NUMBERS>
                   | <NUMBERS> .
                   | <NUMBERS> . <SMALL-NUMBERS>
                   | . <SMALL-NUMBERS>

<LINE-NAME>::=     FIRST | LAST | ALL | CURRENT | *

<LINE>::=          <LINE-NUMBERS> | <LINE-NAME>

<LINE1>::=         <LINE>

<LINE2>::=         <LINE>

<RANGE>::=         <LINE1> | <LINE1> /| <LINE1> / <LINE2>

                   **LINE2 must have a greater value
                     than LINE1**

<OPTIONS>::=       (DISPLAY | NODISPLAY)
     |             (NUMBERS | NONUMBERS)
                   (TEXT | NOTEXT) (VERIFY | NOVERIFY)

<PRINTOPTION>::=   NOEJECT | CONTROL

<FILEOPTION>::=    EBCDIC | COMPRESSED | TEMPORARY

<USERNAME>::=      <IDENTIFIER>

<FILENAME>::=      <IDENTIFIER>

                   **see note above for restrictions
                     on filenames**

<ACCT>::=          <IDENTIFIER>

<JOBNAME>::=       <IDENTIFIER>

<SAVOPTIONS>::=     FORTRAN | METASYM | PL-70 | PL-11

<SETOPTIONS>::=     TERSE | NOTERSE | VERBOSE
                   | NOVERBOSE | LENGTH | NOLENGTH
                   | NUMBERS | NONUMBERS | TEXT
                   | NOTEXT | TABS | NOTABS
                   | TABSTEP | NOTABSTEP | CURRENT
                   | NOCURRENT

<SHOWOPTIONS>::=   SPACE | FILES | HOLDQUEUE
                  | ACCOUNT | ALLHOLDJOBS | LENGTH
                  | TABS | TABSTEP | CURRENT

<SYSID>::=         <IDENTIFIER>

<COL-NUMBERS>::=   <NUMBERS> | <NUMBERS> .
```

The ORION Commands

The following is a formal description of the ORION command language syntax. Reference is made to the options listed above.

```
ADD         (IN <RANGE>) (BY <LINE-NUMBERS>)
            (<OPTIONS>)

CHANGE      (IN <RANGE>) (FROM) <STRING>
            (TO <STRING>) (COLUMN <COL-NUMBERS>
            (/ <COL-NUMBERS>)) (<OPTIONS>)

CLEAR

COPY        (IN <RANGE>) TO <LINE>
            (BY <LINE-NUMBERS>) (<OPTIONS>)

DELETE      (IN <RANGE> ) (<OPTIONS>)

FETCH       <SYSID> (CLEAR)

FILE         FILENAME  (<ACCT>) (<FILEOPTION>)
            (KEY <COL-NUMBERS>)

HELP        (<IDENTIFIER>)

JOIN        <FILENAME> (<ACCT>) (IN <RANGE>)
            (RENUMBER) (TO <LINE>) (BY <LINE-NUMBERS>)
            (CLEAR)

LIST        (IN <RANGE>) (<OPTIONS>)

LOAD        <FILENAME>  (<ACCT>) (CLEAR)

LOCATE      <SYSID>

LOGIN       <USERNAME> <ACCT>
            **LOGON is equivalent to LOGIN**

LOGOFF      (CLEAR)
            **LOGOUT is equivalent to LOGOFF**

MOVE        (IN <RANGE>) TO <LINE>
            (BY <LINE-NUMBERS>) (<OPTIONS>)

PJOB        <JOBNAME> (<ACCT>)

PRINT       (IN <RANGE>) (<PRINTOPTION>)
            (NUMBERS | NONNUMBERS) (SPACES <NUMBERS>)
            (COPIES <NUMBERS>)

PUNCH       (IN <RANGE>) (NUMBERS | NONNUMBERS)

PURGE       <SYSID>

RENUMBER    (IN <RANGE>) (TO <LINE>)
            (BY <LINE-NUMBERS>)

REPLACE     (IN <RANGE>) (<OPTIONS>)

SAVE        <FILENAME> (<ACCT>) (IN <RANGE>)
            (SCRATCH) (<SAVOPTIONS>)
            (NUMBERS | NONUMBERS)

SCRATCH     <FILENAME> (<ACCT>)

SEARCH      (IN <RANGE>) FOR <STRING>
            (COLUMN <COL-NUMBERS> (/<COL-NUMBERS>))
            (<OPTIONS>)

SET         <SETOPTIONS> (TO <NUMBERS>)

SHOW        <SHOWOPTIONS> (<ACCT>) (<STRING>)

SUBMIT      (NUMBERS | NONNUMBERS) (HOLD | NOHOLD)

TALK        <USERNAME> (<STRING>) (REPLY)

TIME

USE         <FILENAME> (<ACCT>) (IN <RANGE>)
            (RENUMBER) (TO <LINE>) (BY <LINE-NUMBERS>)
            (CLEAR)
```