A CHARACTER STRING SCANNER

RICHARD L. ENISON

JANUARY 1971

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND
A CHARACTER STRING SCANNER

by
Richard L. Enison*

January 1971

GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland

*Mr. Enison is a coop student from Pratt Institute who is working in the MTAD under the direction of Mrs. I. Cole.
ABSTRACT

A computer program called Character String Scanner (CSS), is presented. It is designed to search a data set for any specified group of characters and then to flag this group. The output of the CSS program is a listing of the data set being searched with the specified group of characters being flagged by asterisks. Therefore, one may readily identify specific keywords, groups of keywords or specified lines of code internal to a computer program, in a program output, or in any other specific data set. Possible applications of this program include the automatic scan of an output data set for pertinent keyword data, the editing of a program to change the appearance of a certain word or group of words, and the conversion of a set of code to a different set of code.
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. BACKGROUND</td>
<td>1</td>
</tr>
<tr>
<td>2. PURPOSE</td>
<td>1</td>
</tr>
<tr>
<td>3. SYMBOLS USED IN THIS REPORT</td>
<td>1</td>
</tr>
<tr>
<td>4. METHOD</td>
<td>2</td>
</tr>
<tr>
<td>5. INPUT</td>
<td>6</td>
</tr>
<tr>
<td>6. OUTPUT</td>
<td>7</td>
</tr>
<tr>
<td>7. SAMPLE RUN</td>
<td>9</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>10</td>
</tr>
<tr>
<td>APPENDIX - THE FORTRAN SUBROUTINE PARM</td>
<td>11</td>
</tr>
</tbody>
</table>

**PRECEDING PAGE BLANK NOT FILMED**

**Preceding page blank**
A CHARACTER STRING SCANNER

1. BACKGROUND

It was decided to use a static version of DODS as an interim tool for the R and D user of GTDS. This static version of DODS, however, has to be a version that would allow the R and D user to incorporate his changes on either a temporary or permanent basis. It has also to be a version that the OS/360 does not think is the "real" DODS. Several tools were developed to aid in the development of such a special version of DODS. This paper describes one of these tools.

2. PURPOSE

CSS (Character String Scanner) scans a character string, e.g. an EBCDIC record of data, in search of a particular group of characters. The character string scanned is printed out, and each occurrence of the group of characters, if any, is flagged below with asterisks. The entire line is flagged with dashes if the group of characters is found. For an example, see the section OUTPUT.

This is done for each record of an input data set, listing the records with or without flags as appropriate, in succession. The user may specify at load time whether the first character of each record is to be ignored as a carriage control character, and how many four-byte words of data are to be scanned.

3. SYMBOLS USED IN THIS REPORT

LUP — variable containing number of four-byte words to be scanned per record
LCC — one-byte LOGICAL variable indicating whether first character of each record is to be ignored
PC — name of COMMON block containing NUM and F
NUM — variable containing number of characters in PARM field
F — singly-subscripted LOGICAL*1 type variable containing PARM field
PARM — subroutine called to access PARM field
DODS — singly-subscripted INTEGER*2 type variable containing character string 'DODS'
COMPR — singly-subscripted INTEGER*2 type variable containing the group of characters for which to search

TEXT — singly-subscripted variable into which record is read

LTEXT — singly-subscripted LOGICAL*1 variable EQUIVALENCED to TEXT

ITEXT — singly-subscripted INTEGER*2 variable used to break record up into single characters for comparison with COMPR

ILTEXT — singly-subscripted LOGICAL*1 variable EQUIVALENCED to ITEXT

ASTER — singly-subscripted variable containing asterisks to be printed, if any

MARK — singly-subscripted LOGICAL*1 variable EQUIVALENCED to ASTER

LINE — singly-subscripted variable containing dashes to be printed when appropriate

IND — one-byte LOGICAL variable indicating whether any flags are to be printed

N — variable containing the number of characters in COMPR

4. METHOD

The information the user wishes to pass to CSS: 1. whether the first character of each input record is to be ignored, 2. the number of four-byte words to be scanned, and 3. the group of characters for which to search, is passed through a JCL (Job Control Language) feature known as the PARM field. For those readers unfamiliar with this feature, let us review JCL in general.

Every job submitted through 360/OS (Operating System) is defined and invoked by JCL cards. The main types of such cards are JOB, EXEC, and DD cards. The JOB card is the first of every job, and in addition to delimiting jobs, it gives the system such information as the name of the job, of the user, the user's account number, box number and other accounting information, as well as such information as whether JCL cards are to be listed and if so, through what output class.

Now, a job consists of one or more steps. The first card of a step is the EXEC card. This card specifies the program to be executed, together with other information. Data sets used by the program are defined by DD cards immediately following the EXEC card. The first field of the DD card is the ddname, which identifies the information coded on the card for use by system I/O routines. The program does I/O by passing the ddname, along with other information, to the system in an internal Data Control Block (DCB). The data set may exist on disk, drum, data cell, tape, cards or may be sent to a printer or card punch. This
information is coded on the DD card. For example, a card data set is defined by a DD card of the form

```
//ddname DD *
```

```
.
.
.
data cards
.
.
.
```

next JCL card

or, if some of the data cards begin with two slashes (//),

```
//ddname DD DATA
```

```
.
.
.
data cards
.
.
.
```

```
/*
```

The program invoked by the EXEC statement must itself be in the form of a load module and must be a member of a partitioned data set called a library. This may be the system program library (SYS1.LINKLIB), a private library, or a temporary library which exists only for the duration of the run. An example of the latter is a typical compile-link-go job. The LINK step invokes the linkage editor, which creates a load module, puts it in a temporary library, and then the GO step invokes that. The compilers, assemblers, linkage editors and utility programs provided by IBM all reside in the system library. They are invoked as follows:

```
// EXEC PGM=progname
```
with other parameters optional. Programs in a private library are invoked the same way, except that a DD card defining the library must be provided immediately following the JOB card, and must have the special ddname JOBLIB. Programs in a temporary library are invoked as follows:

```c
// EXEC PGM=*. stepname. ddname
```

where stepname is the name of the LINK step in which the load module was created. This is indicated on the EXEC card for that step in the stepname field, which is optional and immediately follows the two slashes:

```c
//stepname EXEC ...
```

Frequently used sequences of JCL statements are stored in the system procedure library (SYS1.PROCLIB), which is a partitioned data set whose members are called procedures. Each procedure has one or more EXEC statements, each followed by one or more DD statements. A procedure is invoked as follows:

```c
// EXEC procname
```

Parameters on the EXEC cards in the procedure may be overridden by parameters coded on the above EXEC card, identified with the appropriate stepnames as coded in the procedure. Parameters on DD cards in the procedure may be overridden with DD cards immediately following the EXEC card invoking the procedure, with the appropriate stepname coded as part of the ddname on each such card. For example, the procedure FORTRANG invokes the FORTRAN level G compiler, whose name, incidentally, is IEYFORT. The stepname is SOURCE. The following example shows one of each type of override.

```c
// EXEC FORTRANG,REGION. SOURCE=300K
//SOURCE. SYSIN DD *
```

Since FORTRANG has only one step, the use of the stepname SOURCE is unnecessary and may be omitted. An example of a multi-step procedure is LINKGO, which has a LINK and a GO step. The stepnames are LINK and GO.

Among the parameters which may be coded on an EXEC statement is PARM. The parm field is just a character string with at most 100 characters. It is coded as follows:
EXEC ... , PARM='this is the parm field'

One may pass information to the compiler, linkage editor, and/or one's own program in a compile-link-go job. An example of all three is:

EXEC FORTRAN, PARM='LIST, DECK'

EXEC LINKGO, PARM. LINK='MAP, LIST', PARM. GO='EARTH-MOON'

If one passes information in this way to one's own program, then one's program must contain coding to retrieve this information. To do this, one must understand how the system makes this information available to programs.

Upon entry to the program invoked by the EXEC statement, general register 1 contains the address of a pointer to a field of data in core. The field is located on a halfword boundary, and its format is as follows:

<table>
<thead>
<tr>
<th>length of field</th>
<th>the field ....</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>0 1 2 ....</td>
</tr>
</tbody>
</table>

That is, the first two bytes contain a halfword integer whose value is the length of the PARM field in bytes (characters). This is immediately followed by the information coded in the PARM field itself. CSS, which was written in FORTRAN, calls a FORTRAN SUBROUTINE called PARM which accesses this information and returns it in the COMMON block PC. This SUBROUTINE is described in the appendix.

The PARM field retrieved as above is interpreted by CSS as having three fields or less. The formats of these fields is described in the section INPUT. The three fields are:

1. A logical value, true or false, indicating whether the first character of each record is to be ignored.
2. A number, from 1 to 33, indicating the number of words of data to be scanned per record.
3. A character string, indicating the group of characters for which to search.
If one or more fields are omitted, default values are assumed. The default for field one is true, for field two 30, and for field three 'DODS'.

The values of these three fields, whether default or explicitly specified, are placed in the variables LCC, LUP and COMPR, respectively. The variable N is set equal to the number of characters in COMPR. LUP and N, or functions thereof, are used as upper limits in DO loops related to the scan. The logical switch LCC is used to decide which FORMAT statement should be used to read the record. In the READ statement, an iterated I/O list is used, with LUP as the upper limit, to read the desired number of words into the elements of TEXT. These elements are then printed. Then in a DO loop, MARK is initialized as all blanks, and TEXT is broken up into characters by moving the elements of LTEXT into every other element of ILTEXT. The odd elements have already been initialized as blanks in a specification statement. Thus, upon exit from the loop, ITEXT will be an array of halfwords each of the form

<table>
<thead>
<tr>
<th>blank</th>
<th>character</th>
</tr>
</thead>
</table>

byte 0 1

That is, the first byte is blank and the second, a character from the record. This can be compared with the elements of COMPR, which have similar form. This is done because, with some compilers, one cannot compare logical variables for equality.

Next, IND is set to FALSE, so that if its value is not changed during the scan by finding a match, it will be FALSE after the scan and so ASTER and LINE will not be printed. Finally, we have the scan itself, which consists of a DO loop with two inner loops. The outer loop is indexed by the number of the character in ITEXT which is the first of those we are comparing with COMPR. The first inner loop carries out the comparison, character by character, up to N iterations. If a mismatch is found, control is passed to the end of the outer loop. If not, the second inner loop is executed, which sets the appropriate elements of MARK to asterisks. Then IND is set to TRUE, and we drop to the end of the outer loop. Upon exit from the outer loop, we check IND. If it is TRUE, we print ASTER and LINE and branch back to statement 50 where the test on LCC is made and another record is read in. Otherwise, we go directly to statement 50. When the last record has been processed, we take the END exit of the READ statement and STOP.

5. INPUT

CSS accepts two inputs: the PARM field, and a sequential data set of records to be scanned. The latter is read on logical FORTRAN unit 8, so it is defined in
the GO step by a DD card with the ddname FT08F001. It need have no special
format, except that no record may be shorter than LUP words. It is suggested
that a fixed record format be used, to prevent some records, such as pagination
records, from being too short.

The format of the PARM field is:

<table>
<thead>
<tr>
<th>logical switch</th>
<th>ignored</th>
<th>number of words</th>
<th>ignored</th>
<th>group of characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

where logical switch is either T or F, for true or false respectively, indicating
whether the first character is to be ignored; number of words is a two-digit dec-
imal numeral indicating the number of words of data to be scanned per record —
if the number is less than 10, the first digit must be coded as 0; and group of
characters is the character string for which to search. All fields must be en-
tered in the positions shown. The ignored characters may be used optionally as
punctuation or separation to make the field readable. I prefer commas.

Fields may be omitted by shortening or omitting the PARM field. If number of
words is coded, so must logical switch, and if group of characters is coded,
these two must be also. If no PARM field is coded, all these quantities will as-
sume their default values. If a PARM field is coded and one or two fields are
omitted, the cutoff must be immediately before one of the ignored characters.
To include the ignored character as the last character in the PARM field will
produce unpredictable results. For an example, the following PARM field expli-
citly defines the default values:

// EXEC LINKGO,PARM.GO='T,30,DODS'

6. OUTPUT

The only output of CSS is a listing of the input, or of the input records truncated
by specification of number of words, together with flags when appropriate, as
explained in the section PURPOSE. For example, suppose we are searching for
the characters 'MOON' in a card data set (deck). Then the JCL would be:

JOB CARD

...
/ EXEC LINKGO, PARM.GO='F,20,MOON'

.
.
.

//GO.FT08F001 DD *

.
.
.

DC
EPHEM
EARTH-MOON DISTANCE
SATELLITE VELOCITY
POTENTIAL OF MOON AT PERIGEE, 500 KM FROM MOON
.
.
.

/*

and the output would be:
.
.
.
.

DC
EPHEM
EARTH-MOON DISTANCE
****
SATELLITE VELOCITY

POTENTIAL OF MOON AT PERIGEE, 500 KM FROM MOON

* *** * **

7. SAMPLE RUN

In the run shown at the end of this document, we wish to edit a MAPDISK listing of the disk pack DODS01, flagging all occurrences of 'DODS'. No PARM field is coded, since we want the default values. The first step invokes MAPDISK, which resides in the system library. The output is placed in the data set defined by the DD card with ddname SYSPRINT. This is ordinarily sent to a printer, but we want it to be processed by CSS in a later step, so we code a DD card defining a temporary sequential data set on disk. The remaining steps are just a compile-link-go of CSS. In the GO step, we code the FT06F001 DD card to retrieve the data set created in the MAP step.

This run is typical of a larger class of uses of CSS, but many others are possible, with correspondingly varied JCL deck setups. To learn how to set up the JCL for a particular application of CSS, the reader is referred to reference #1. The other references may provide the reader with a background for understanding how CSS works, depending upon his familiarity with IBM System/360 and its Operating System.
REFERENCES

1. IBM System/360 Operating System, Job Control Language. IBM Corporation, Form C28-6593

2. IBM System/360 Operating System, Supervisor and Data Management Services. IBM Corporation, Form C28-6646

3. IBM System/360, Principles of Operation. IBM Corporation, Form A22-6821

4. IBM System/360 Operating System, Assembler Language. IBM Corporation, Form C28-6514

5. IBM System/360 Operating System, Fortran G and H Programmers' Guide. IBM Corporation, Form C28-6817
APPENDIX

THE FORTRAN SUBROUTINE PARM

PURPOSE

The SUBROUTINE PARM was written to retrieve the PARM field for CSS. There is no reason, however, why it could not perform this service for any other routine. Some caution should be exercised by the would-be user, as there are several restrictions on its use. Here is a list of such restrictions:

1. PARM must be called directly by the MAIN routine.

2. The CALL statement must be the first executable statement in the routine.

3. The CALL statement must have no argument list, i.e., it must be of the form

   CALL PARM

The number of characters in the PARM field is returned in an INTEGER variable called NUM, and the characters of the PARM field themselves are returned in the one-byte elements of the LOGICAL array F, both in the COMMON block PC. Thus, in order to have access to this information, the user must code these specification statements:

   LOGICAL*1 F(100)
   COMMON/PC/NUM, F

Of course, the names NUM and F are arbitrary and may be changed by the user. If there is no PARM field, NUM will have the value 0. In any case, the first NUM elements of F will contain the characters of the PARM field, one per element, and the remaining elements of F will have unpredictable contents.

LIST OF SYMBOLS USED IN PARM

PC — COMMON block containing NUM and F
NUM — variable in which length of PARM field is returned
F — singly-subscripted LOGICAL*1 variable in which PARM field is returned

ARRAY — singly-subscripted LOGICAL*1 variable containing system PARM field, including length field

INUM — INTEGER*2 variable into which length is moved

LN — singly-subscripted LOGICAL*1 variable EQUIVALENCED to INUM

METHOD

The SUBROUTINE statement has an argument list with the single argument ARRAY. Now the way argument lists are passed in FORTRAN under IBM System/360 OS is as follows: the address of the argument list is placed in general register 1 by the calling routine. The argument list itself is located on a word boundary and consists of a sequence of one or more single-word address constants. In FORTRAN these would be the addresses of variables. Thus the SUBROUTINE PARM will expect, upon entry, to find in register 1 the address of a one-argument argument list pointing to an array. References in the SUBROUTINE to elements of ARRAY will retrieve the data in the field pointed to by the address constant. Now, since the CALL statement was the first statement executed in the program and since it had no argument list, register 1 will contain whatever it had on entry to the program, namely, the address of a pointer to the system PARM field, which includes a length field. All that needs to be done, is to move the information, in appropriate format, into the COMMON variables NUM and F.

The first two bytes of the system PARM field are the length, in bytes, of the PARM field itself. We move these bytes, the first two elements of ARRAY, into the two elements of LN. This is EQUIVALENCED to the halfword integer variable INUM, which therefore now has as its value the length of the PARM field. We next set the fullword integer variable NUM equal to INUM, and half the job is done. If NUM is 0, it means there is no PARM field and we RETURN. Otherwise, we execute a DO loop indexed by NUM to move the elements of ARRAY from the third on into the elements of F. NUM and F now contain all the necessary information, so we RETURN.

SOURCE LISTING

A source listing of PARM, as well as of CSS, can be found in the SAMPLE RUN below.
COMMON /PC,ACC,KK
CALL PARM
IF (NUM=5) GO TO 99
LJTEMP(1)=F(1)
IF (LTEMP.NE.LTEC) LCC=.FALSE.
LTEMP(2)=F(3)
I=I+1 I TEMP=I TEMP-240
LUMP=I TEMP
LTEMP(2)=F(4)
LUMP=LUMP+I TEMP-240
60 LUMP=LUMP
N=N+1
GO TO 70 K=1,A
70 CMPPH(K)=LUMP(K)
IF (NUM=5) GO TO 99
N=NUM-5
JTEMP=62
DO 00 K=1,N
LUMP(K)+=F(K+5)
80 CMPPH(K)=JTEMP
65 N=N-1
50 IF (LCC.GT.0) GO TO 61
ACAD (5,IC,NU=50) (LTEC(K),K=1,LUMP)
160 FORMAT (J33A4)
GU 10 52
61 READ (5,101) (LTEC(K),K=1,LUMP)
52 WRITE (6,101) (LTEC(K),K=1,LUMP)
130 FORMAT (J33A4)
DO 10 K=1,LUMP
10 LTEMP(2)=LTEMP(K)
IC=0
IND=.FALSE.
GO TO 30 K=1,LUMP
IF (IC).LT.18
41 GO TO 30 K=1
IF (LTEMP(K+K-1).NE.CMPPH(K)). UC TC 20
60 CONTINUE
IC=0
DO 20 ACC KK=1,A
20 MARK(K+K-1)=ASTI
IND=.TRUE.
15 IC=IC-1
26 CONTINUE
IF (IND) WRITE (5,102) ASTC+1 LIN
102 FORMAT (J33A8/J33A8)
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARM</td>
<td>116</td>
<td>LOCMN=</td>
<td>11C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEXT</td>
<td>120</td>
<td>LEXIST</td>
<td>120</td>
<td>LEXIST</td>
<td>LEXIST</td>
<td>144</td>
<td>LEXIST</td>
<td>144</td>
<td>LEXIST</td>
<td>144</td>
<td>LEXIST</td>
</tr>
<tr>
<td>DDOS</td>
<td>2AC</td>
<td>ASP1</td>
<td>2BP</td>
<td>AST2</td>
<td>AST5</td>
<td>2BP</td>
<td>AST5</td>
<td>2BP</td>
<td>AST5</td>
<td>2BP</td>
<td>AST5</td>
</tr>
<tr>
<td>ITTEMP</td>
<td>33C</td>
<td>LTEMMP</td>
<td>33C</td>
<td>UTFEM</td>
<td>UTFEM</td>
<td>33C</td>
<td>UTFEM</td>
<td>33C</td>
<td>UTFEM</td>
<td>33C</td>
<td>UTFEM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUP</td>
<td>340</td>
<td>LUPM4</td>
<td>356</td>
<td>N</td>
<td>LUPM4</td>
<td>356</td>
<td>LUPM4</td>
<td>356</td>
<td>LUPM4</td>
<td>356</td>
<td>LUPM4</td>
</tr>
<tr>
<td>LUPM4</td>
<td>356</td>
<td>IC</td>
<td>356</td>
<td>R</td>
<td>LUPM4</td>
<td>356</td>
<td>LUPM4</td>
<td>356</td>
<td>LUPM4</td>
<td>356</td>
<td>LUPM4</td>
</tr>
<tr>
<td>LCC</td>
<td>364</td>
<td>INC</td>
<td>365</td>
<td>NM1</td>
<td>360</td>
<td>360</td>
<td>360</td>
<td>62</td>
<td>360</td>
<td>360</td>
<td>360</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPR</td>
<td>366</td>
<td>COMPR</td>
<td>424</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
<th>SYMBOL</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>44B</td>
<td>101</td>
<td>64E</td>
<td>102</td>
<td>64E</td>
<td>102</td>
<td>64E</td>
<td>102</td>
<td>64E</td>
<td>102</td>
<td>64E</td>
</tr>
</tbody>
</table>

OPTIONS IN EFFECT: NOJESCC,EOCITY,RAWLIST,EXCEED,MAC

OPTIONS IN EFFECT: NAME = MAIN  LINEMNT = 33

STATISTICS: SOURCE STATEMENTS = 07, FILE 9 0126 2210

STATISTICS: NO DIAGNOSTICS GENERATED
0001   SLQKOLNC, PARX(ARRAY)  *
0002   LOGICCAL = 1 (COPT,ARRAY(102)YCN(2))
0003   INTEGER 2 LAY
0004   EQUIVALENCE (INUPTCN(1))
0005   COMMUN /PC/NUM,F
0006   LN(1) = ARRAY(1)
0007   LN(2) = ARRAY(2)
0009   IF (NUM = EQ. 0) RETURN
0010   DO 10 K=1,NUM
0011    F(K) = ARRAY(K+2)
0012   RETURN
0013   END
<table>
<thead>
<tr>
<th>NAME</th>
<th>ORIGIN</th>
<th>LENGTH</th>
<th>NAME</th>
<th>LOCATION</th>
<th>NAME</th>
<th>LOCATION</th>
<th>NAME</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>00</td>
<td>88C</td>
<td>NAME</td>
<td>LOCATION</td>
<td>NAME</td>
<td>LOCATION</td>
<td>NAME</td>
<td>LOCATION</td>
</tr>
<tr>
<td>PARM</td>
<td>8C0</td>
<td>1A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMCECOM**</td>
<td>46E</td>
<td>F31</td>
<td>IMCECOM</td>
<td>19A0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMCECOM2</td>
<td>19A0</td>
<td>46E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF-CMOV**</td>
<td>1F26</td>
<td>1155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMCEPMT</td>
<td>30C0</td>
<td>612</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMCEFOS</td>
<td>35D8</td>
<td>124C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF-CUOPT *</td>
<td>4878</td>
<td>350</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMCEMNT</td>
<td>48C6</td>
<td>98C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF-CUOPT *</td>
<td>518C</td>
<td>63U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMCEMTCH</td>
<td>57C0</td>
<td>2DE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>5A5C</td>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

** ENTRY ADDRESS: 0C **
TOTAL LENGTH: 5AB6

**** GSFC DOES NOT EXIST ULT HAS BEEN ADDED TO DATA SET
<table>
<thead>
<tr>
<th>CONTENTS ON VOLUME=SEN=UDD501 UNIT=544</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DATA SET NAME</strong></td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>DSGRTGSCN</td>
</tr>
<tr>
<td>DSGRTGSCN</td>
</tr>
<tr>
<td>DSGRTGSCN</td>
</tr>
<tr>
<td>DSGRTGSCN</td>
</tr>
<tr>
<td>DSGRTGSCN</td>
</tr>
<tr>
<td>DSGRTGSCN</td>
</tr>
<tr>
<td>DSGRTGSCN</td>
</tr>
<tr>
<td>DATASET NAME</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>NUTC6</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
The image contains a page from a document that appears to be a system support utility output. The text is formatted in a structured way, likely representing data or output from a computer system. The page includes various columns and entries related to file system data, such as volume numbers, file names, and allocation sizes. The content is technical and likely intended for system administrators or IT professionals to understand the allocation and management of files on the system.
```sql
null
```
<table>
<thead>
<tr>
<th>FIRST TRACK</th>
<th>LAST TRACK</th>
<th>LENGTH</th>
<th>EXTENT</th>
<th>DATA SET NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>C001C</td>
<td>C0010</td>
<td>01</td>
<td>VTCC</td>
</tr>
<tr>
<td>00011</td>
<td>C0012</td>
<td>C0000d</td>
<td>01</td>
<td>W00H×DML1H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00013</td>
<td>C0016</td>
<td>C0004</td>
<td>01</td>
<td>VACCAFP</td>
</tr>
<tr>
<td>00017</td>
<td>C0017</td>
<td>C0001</td>
<td>01</td>
<td>TSDTCAP</td>
</tr>
<tr>
<td>00010</td>
<td>C0010</td>
<td>C0001</td>
<td>01</td>
<td>F3EG</td>
</tr>
<tr>
<td>00019</td>
<td>C0019</td>
<td>C0001</td>
<td>01</td>
<td>NULLFACT</td>
</tr>
<tr>
<td>00020</td>
<td>C0020</td>
<td>C0003</td>
<td>01</td>
<td>W00H×VA.TIF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00023</td>
<td>C0033</td>
<td>C0011</td>
<td>01</td>
<td>DTFCDO</td>
</tr>
<tr>
<td>00034</td>
<td>C0135</td>
<td>C0106</td>
<td>01</td>
<td>AUTOCB</td>
</tr>
<tr>
<td>00040</td>
<td>C0374</td>
<td>C0040</td>
<td>01</td>
<td>W00H×GENCB</td>
</tr>
<tr>
<td>00190</td>
<td>C0375</td>
<td>C0203</td>
<td>01</td>
<td>W00H×JERCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 00430       | C0436      | C0267  | 01     | FREE SPACE **
| 00037       | C0437      | C0001  | 01     | V3GCT         |
| 00048       | C0438      | C0002  | 01     | W00H×UTALIN   |
|              |            |        |        |               |
| 00440       | C0465      | C0410  | 01     | SAECEL        |
| 00047       | C0454      | C0023  | 01     | MGCLCMH       |
| 00048       | C0429      | C0205  | 01     | SAE×PECLU     |
| 00050       | C0374      | C0240  | 01     | W00H×GENCMU   |
|              |            |        |        |               |
| 00740       | C0115      | C0280  | 01     | W00H×ACCLIC   |
|              |            |        |        |               |
| 01020       | C0127      | C0003  | 01     | W00H×ET       |
|              |            |        |        |               |
| 01023       | C0147      | C0257  | 02     | FREE SPACE **
<p>| 01022       | C0155      | C0001  | 01     | D00H×VA.CUT   |
|              |            |        |        |               |
| 01036       | C0134      | C0015  | 01     | W00H×MCSP1S   |
|              |            |        |        |               |
| 01115       | C0125      | C0001  | 01     | D00H×LMMHT    |
|              |            |        |        |               |
| 01126       | C0136      | C0001  | 02     | W00H×ALICE    |
|              |            |        |        |               |
| 01127       | C0137      | C0001  | 01     | FCAHF         |
| 01128       | C0136      | C0033  | 01     | V00DM          |
| 01171       | C0174      | C0004  | 01     | T52LLG         |
| 01175       | C0175      | C0001  | 01     | D00H×VA.CUT   |</p>
<table>
<thead>
<tr>
<th>Track</th>
<th>Segments</th>
<th>Type</th>
</tr>
</thead>
</table>
| 01175 | 01175    | 00109 | 01 | FULL SPACE **
| 01160 | 01175    | 00109 | 01 | FULL SPACE **
| 01160 | 01175    | 00109 | 01 | FULL SPACE **
| 01160 | 01175    | 00109 | 01 | FULL SPACE **
| 01160 | 01175    | 00109 | 01 | FULL SPACE **
| 01160 | 01175    | 00109 | 01 | FULL SPACE **
| 01160 | 01175    | 00109 | 01 | FULL SPACE **
| 01160 | 01175    | 00109 | 01 | FULL SPACE **
| 01160 | 01175    | 00109 | 01 | FULL SPACE **

** Contents on volume 1B:1 = 00001 **
<table>
<thead>
<tr>
<th>02553</th>
<th>02566</th>
<th>C0010</th>
<th>01</th>
<th>G4ENP</th>
<th>02</th>
<th>G44ECE</th>
<th>02</th>
<th>G44ECE</th>
<th>02</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
</tr>
</thead>
<tbody>
<tr>
<td>02569</td>
<td>02574</td>
<td>C0010</td>
<td>01</td>
<td>G4UTL</td>
<td>02</td>
<td>G44ECE</td>
<td>02</td>
<td>G44ECE</td>
<td>02</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>02579</td>
<td>02174</td>
<td>C0001</td>
<td>01</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>02580</th>
<th>03569</th>
<th>C0120</th>
<th>02</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
</tr>
</thead>
<tbody>
<tr>
<td>03100</td>
<td>03128</td>
<td>00003</td>
<td>10</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03199</td>
<td>03157</td>
<td>C0031</td>
<td>01</td>
<td>G44ECE</td>
<td>02</td>
<td>G44ECE</td>
<td>02</td>
<td>G44ECE</td>
<td>02</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>02200</td>
<td>02239</td>
<td>C0000</td>
<td>02</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>02340</th>
<th>03339</th>
<th>00160</th>
<th>02</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
</tr>
</thead>
<tbody>
<tr>
<td>02380</td>
<td>03364</td>
<td>C0354</td>
<td>01</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03390</td>
<td>03419</td>
<td>C0030</td>
<td>02</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03490</td>
<td>03479</td>
<td>C0030</td>
<td>04</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03570</td>
<td>03569</td>
<td>C0031</td>
<td>03</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03410</td>
<td>03400</td>
<td>C0001</td>
<td>04</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03480</td>
<td>03594</td>
<td>C0003</td>
<td>02</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>03625</th>
<th>03565</th>
<th>C0010</th>
<th>01</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
</tr>
</thead>
<tbody>
<tr>
<td>03650</td>
<td>03662</td>
<td>C0031</td>
<td>01</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>03680</td>
<td>03585</td>
<td>C0037</td>
<td>31</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>03940</th>
<th>03565</th>
<th>C0110</th>
<th>02</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
<th>**</th>
</tr>
</thead>
<tbody>
<tr>
<td>03950</td>
<td>03554</td>
<td>C0006</td>
<td>04</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03955</td>
<td>03544</td>
<td>C0009</td>
<td>11</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03960</td>
<td>03565</td>
<td>C0031</td>
<td>03</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03970</td>
<td>03565</td>
<td>C0030</td>
<td>04</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03980</td>
<td>03554</td>
<td>C0003</td>
<td>02</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>03995</td>
<td>03565</td>
<td>C0002</td>
<td>02</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
</tbody>
</table>

| 02557 | 02565 | C0032 | 02 | ** | ** | ** | ** | ** | ** | ** | ** | ** | ** |

| 03650 | 03665 | C0001 | 12 | ** | ** | ** | ** | ** | ** | ** | ** | ** | ** |

END OF UTILITY - USE THIS ARE MAPPED