Character and String Handling in INTRAC

File Handling in Program Package

Essebo, Tommy

1980

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7186)/1-009/(1980)

CHARACTER AND STRING HANDLING IN INTRAC. FILE HANDLING IN PROGRAM PACKAGE.

TOMMY ESSEBO

DEPARTMENT OF AUTOMATIC CONTROL
LUND INSTITUTE OF TECHNOLOGY
FEBRUARY 1980

Dokumentutgivare
06T0
Lund Institute of Technology
Handläggare Dept of Automatic Control
06T0
Författare
08T0
Tommy Essebo

Dokumentnamn
06T4
REPORT LUTFD2/(TFRT-7186)/1-9/(1980)
Utgivningsdatum
06T4
Feb 1980

Dokumentbeteckning
06T4
Ärendebeteckning
06T6

10T4

Dokumenttitel och undertitel
18T0
Character and String Handling in INTRAC

File Handling in Program Packages

Referat (sammandrag)
26T0
This document describes the programming conventions and requirements that are necessary to comply with in order to implement the interactive programs developed at the Department of Automatic Control, Lund Institute of Technology. The plotting library and the interaction module are described elsewhere.

Referat skrivet av
32T0
Author

Förslag till ytterligare nyckelord
44T0
Computer aided design, Interactive programs

Klassifikationssystem och -klass(er)
50T0

Indextermer (ange källa)
52T0
Computer software (Theaurus of Engineering and Scientific Terms, Engineers Joint Council, N.Y., USA).

Omfång
96pages

Språk
68T0
English

Övriga bibliografiska uppgifter
56T2

Sekretessuppgifter
60T0

ISSN
60T4

ISBN
60T6

Dokumentet kan erhållas från
62T0
Department of Automatic Control
Lund Institute of Technology
Box 725, S-220 07 LUND 7, Sweden

Mottagarens uppgifter
62T4

Pris
66T0

DOKUMENTDATABLAD enligt SIS 62 10 12

SIS-DB 1

Blankett LU 11:25 1976–07

1 General

FORTRAN IV has no specific data type for character strings and no operations to handle this type of data. It is thus necessary for a user to define his own character string handling. This document describes the standard used by all programs using INTRAC .

1. All character strings are stored in real arrays with a "packing density" of four characters/real variable

2. Strings are padded with spaces to the nearest real variable boundary

3. It is recommended that if the specific FORTRAN implementation allows more than four characters/real word the rest of the characters are space-filled

4. "String litterals" in the form 4HABCD are used to assign values to variables only in DATA statements

5. A defined set of routines is used to manipulate the strings

6. There are three common types of strings used in INTRAC:
BUFF(20) - Line buffer, max 80 char.
VAR(2) - Variable identifier, max 8 char.
CHAR(1) - Single character

String handling routines:

GAC	- Get a single character from a string
PAC	- Insert a single character in a string
HSTORV	- Assign values to a string
LCOMPV	- Compare strings
FAC	- Decode a character in a string
IFAC	- Initializes FAC with a new buffer

2 Description of the string handling routines

SUBROUTINE GAC(IP,BUFF,CHAR)

Returns single character CHAR from position IP in string BUFF. If IP<1 or IP>80 a space is returned. IP and BUFF are not changed by GAC. The integer code of CHAR (rank) must be returned in commonblock /CRANK/IRANK .

SUBROUTINE PAC(IP,BUFF,CHAR)

Inserts single character CHAR in string BUFF at position IP. IP, CHAR and the rest of BUFF are not changed by PAC. If IP<1 or IP>80 BUFF will not be changed (immediate return from PAC).

SUBROUTINE HSTORV(SOURCE,DESTIN,NR)

Moves NR real words from string SOURCE to string DESTIN. The transfer must be made without any conversion (bit pattern transfer) and this is the reason why HSTORV may be system dependent .

LOGICAL FUNCTION LCOMPV(STR1,STR2,NR)

Compares the strings STR1 and STR2 of length NR real words and returns value .TRUE. if they are equal (bit pattern comparison).

SUBROUTINE IFAC(BUFF,NCHAR)

Initializes subroutine FAC with a new line buffer .
BUFF and NCHAR are stored in a commonblock called IFACOM that is subsequently accessed by FAC. IFAC is not system dependent .
BUFF - Line buffer to be decoded
NCHAR - Current length of BUFF (max 80)

SUBROUTINE FAC(IP,CHAR,ITYPE)

Returns character CHAR from position IP in BUFF. ITYPE is returned as the type of CHAR. IP and BUFF are not changed by FAC.
ITYPE values:

1 Alphabetic character
2 Numeric character
3 Space
4 Delimiter
5 Line terminator (end of line)
6 Unrecognized character

If IP>NCHAR ITYPE=5 is returned.

3 System dependent interface for display handler (DISHDL).

ERDIS - Erase display (Also used by plot package)
PLCURS - Position display cursor (line only)
TPOS - Position display cursor (line & column)
TWRITE - Display string
TREAD - Display string & read input line

SUBROUTINE ERDIS(LFLAG)

Erase display if LFLAG true else no action.

SUBROUTINE PLCURS(NL)

Position display cursor to beginning of line NL

SUBROUTINE TPOS(NL,NC)

Position display cursor to line NL and column NC.

There are further differences between PLCURS and TPOS. PLCURS is used before writing information on the display outside the control of DISHDL (usually by formatted WRITE) and the cursor must be physically positioned at line NL before returning from PLCURS. A call to TPOS is always followed by a call to TWRITE or TREAD and the actual cursor positioning can be made in these routines. Special problem: Some systems will always include a LF (line feed) in FORTRAN formatted output lines to a terminal and this will cause bad positioning of output from TWRITE/TREAD compared with formatted WRITE output. This can be solved by letting NL in TPOS (and then in TWRITE/TREAD) start one line lower than in PLCURS. This implies that line Ø (first line on display) will never be used by text output.

SUBROUTINE TWRITE(BUFF,NCHAR)

Writes line buffer BUFF of length NCHAR on logical unit LTO. If LTO=LDIS and LPLOT non-zero the output must start at line NL and column NC defined by last TPOS call.

SUBROUTINE TREAD(OUTBUF,NCHAR,INBUFF,LEOF)

Writes line buffer OUTBUF of length NCHAR on logical unit LTO and reads input line from LKB to line buffer INBUFF. NCHAR returned as length of INBUFF. LEOF returned .TRUE. if EOF dectected in input. Note that INBUFF must be space-filled from pos. NCHAR+1 to the nearest real word boundary. If LTO=LDIS and LPLOT non-zero the output must start at line NL and column NC defined by last TPOS call. Furthermore TREAD must ensure that the input line will follow directly after the last character in OUTBUF on the terminal. Example: OUTBUF='HOW DO YOU DO ? >' , INBUFF=' FINE' On the terminal: HOW DO YOU DO ? > FINE

1 General

All files are used as record oriented sequential mass storage files . There are two different types of files:

A) Data files (unformatted files) .

The first record in a data file is called the filehead and it consists of 10 integers describing the file. The filehead can be extended to the following records. In this case the last integer in the filehead indicates the number of records in the "extended" filehead. The rest of the file consists of constant length records . All I/O concerning data files is made through the subroutines FILES , FILDAT and FILRED . The maximum recordlength currently used in any program package is 51 real words .

B) Text or source files .

Contains variable length records of source text (e.g. ASCII) . Output to a text file is made through formatted WRITE or subroutine WBUFF . Input from a text file is always made through the subroutine RBUFF . The maximum recordlength for text files used in any program package is 20 real words (80 characters).

2 File handling interface

The following system-independent FORTRAN routines constitutes the interface between the program packages and the system dependent file I/O:

FILES - Open and close a data file
FILDAT - Read and write a data file
FILRED - Read a data file
FCHECK - Check if a file exists
FCLOSE - Close a file
FDELET - Delete a file
FENTER - Create a new file for output
FRENAM - Rename a file
FSEEK - Open an existing file for input
RBUFF - Read a text file
WBUFF - Write a text file

There are also a few routines to support the file routines:

FILCHK - Checks if a file is open
FILNAM - Creates system filename (System dep. routine)
LUFIND - Checks if a logical unit number is occupied
TFIENC - Creates temporary filename (System dep. routine)
TFIDEC - Restores original filename (System dep. routine)

Note:

The program packages use a special feature that allows creation of a new file with the same name as an existing file and as long as the new file is open all input from the file is taken from the old file and the output is written in the new file. When the new file is closed the old file (that must already have been closed) will be deleted . If the old file is not closed when the new file is going to be closed the new file will be deleted instead. This feature is implemented within subroutines FENTER and FCLOSE by creation of a file with a temporary filename from the original filename and the new temporary file is renamed when the file is closed. In the code this is done by calling the system dependent subroutine TFIENC. FCLOSE then calls subroutine TFIDEC to recreate the original filename from the temporary name.

3 File names and extension .

In the program packages a filename consists of a 2-word, 4 char/word Hollerith name FNAME and an integer extension IEXT . The max allowed length of FNAME can be changed to suit the system by changing the command decoding routine LTLONG . (It must however be less than 9 characters). The purpose of IEXT is to identify to the program packages what type of file FNAME should be. IEXT can be any integer from 0 to 9. The following numbers are currently used:

IEXT=1 - Data file
IEXT=2 - Text file with all output to it through WBUFF
IEXT=3 - Text file that may be written into by formatted WRITE
IEXT=4 - Temporary data file
IEXT=6 - Temporary text file
IEXT=8 - Aggregate file (special type of data file)

All conversion of FNAME+IEXT to form a system acceptable filename is made in FILNAM .

Example: Filenames in EXEC 8 on UNIVAC 1100
Max FNAME length is 8 char. In FILNAM FNAME is appended with -D for IEXT=1, nothing for IEXT=2 and 3, -A for IEXT=8 and -<IEXT> for all other values.

4 Logical Unit Numbers and system I/O .

In FORTRAN LUN's are used as a simple and convenient way to refer to a specific I/O device (e.g. a file on disc or a cardreader) in READ/WRITE statements . Program package file handling assumes that the system supplies functions available to a user program to connect/disconnect a LUN to any named file any number of times. If this is not feasible the interface routines can be modified so that I/O operations can refer to files by using their names instead of LUN's . This is done in the following way:

1) Data files
   FILDAT/FILRED can use filename retrieved from /FCTAB/ when LUN is known . Subroutine FILES will also have to be modified in the same way (writing/reading of filehead) .
2) Text files
   Since all input is made using RBUFF this routine should be modified by including /FCTAB/ and using filenames instead . The problem is somewhat larger for text file output since formatted WRITE might be used (IEXT=3). In program package SIMNON all text file output is made through subroutine WBUFF ,that can be modified in the same way as RBUFF . Other program packages (e.g. IDPAC) uses formatted WRITE for creation of system description files . Most FORTRAN systems have an ENCODE statement (makes core-to-core data transfers under FORMAT control). All WRITE statements directed to a file then could be changed to an ENCODE to an internal buffer and then a call to WBUFF with the internal buffer .

5 System dependent file operations .

The following basic file functions must be supplied to support FORTRAN-written interface routines:

```
SEEK(LUN,FILE)
ENTER(LUN,FILE)
RCLOSE(LUN)
WCLOSE(LUN)
FSTAT( FILE,LFIL)
DELETE( FILE)
RENAME( FILE1,FILE2)
RDINT(LUN,IVNAME,NNR,IEND,IND)
RDREAL(LUN,VNAME,NNR,IEND,IND)
WRINT(LUN,IVNAME,NNR)
WRREAL(LUN,VNAME,NNR)
```

LUN - Logical unit number (file reference number) used to reference the file in I/O operations

FILE - System dependent file name created from program compatible file name and file extension type , IEXT

SEEK - Connect LUN to FILE and open file for input . Input is always made from RBUFF (text files) or FILDAT/FILRED (data files) using RDREAL or RDINT .

ENTER - Create FILE (that does not exist earlier) , connect LUN to FILE and open file for output . Output is made from formatted WRITE or WBUFF (text file) or from FILDAT (data files) .

RCLOSE - Close input file connected with LUN and release LUN . Closing should include a rewind operation so that a new SEEK will start at the beginning of the file .

WCLOSE - Close output file connected with LUN and release LUN . Closing should include writing some kind of EOF and a rewind operation .

FSTAT - Check if FILE exists . LFIL returned .TRUE. if FILE exists .

DELETE - Deletes FILE (or makes it unavailable) .

RENAME - Renames FILE1 to FILE2 . On some systems this may have to be implemented by creation of FILE2 and copying FILE1 to FILE2 and then deletion of FILE1 .

RDINT - Read a record of integers from an open file on logical unit LUN and store it in IVNAME. IEND is returned .TRUE. if EOF dectected. IND controls the type of reading performed:
IND=Ø - NNR specifies the desired recordlength and if the actual recordlength differs from NNR a terminating error

should occur.
IND=1 - At input NNR is the max allowed recordlength and if the actual record is longer it is truncated after NNR words. At return NNR is the actual recordlength (or NNR if truncation occurred).

RDREAL - Same as RDINT but for a record of real variables .

WRINT - Write a record of integers in an open file on logical unit LUN from the array IVNAME of length NNR .

WRREAL - Same as WRINT but for a real array .