5 Object program input/output.

Input/output (I/O) is the part of SIMULA used for communication. The mode of communication is:

i) user to program
ii) program to user
iii) program to program

An example of i) is the reading of the source program from the user.

ii) is exemplified by printing on a line printer.

Examples of iii) are the writing of a tape to be read by another program or the same program, and the writing and reading of blocks selected at random on a direct-access device.

In the following subsections the basic principles of Data Management with respect to SIMULA I/O are described. Subsections marked with an asterisk cover special features requiring some background knowledge not given in this manual. The references in the heading of these sections refer to the IBM 360/370 documentation that should be understood before reading the section.

The following features are not described at all since they are not related to the fact that the data sets are processed by SIMULA programs:

Operations on data sets by Utility Programs	(see (9))
Concatenation of data sets	(see (6))
Multivolume data sets	(see (6))
Using members of partitioned data sets as sequential data sets	(see (6))
Details of SPACE allocation	(see (6))
Password data sets	(see (6))
Generation data sets	(see (6), (9))

5.1 OS 360/370 data management terminology.

The pieces of equipment which do the physical reading or writing on the external data carrier are called devices or units (peripherals with non-IBM terminology). The word device usually refers to the kind of equipment (e.g. "a 2311 device") whereas unit refers to a particular piece ("this tape unit"). The distinction is, however, not always maintained and is rather unimportant. A detachable data carrier is called a volume (tape reel, a disc pack or even a drum). Data on volumes are read and written in blocks, which are physically recognized by the units. A block will always correspond to an integral number of bytes.

Devices with a fixed block length are called unit record devices (card reader/punch, line printer).

An important function of data management is to maintain a logical structuring of data, which may be different from the physical structuring in volumes, blocks and bytes.

The largest logical data unit is a data set. Several data sets may reside on one volume, but a data set may also occupy several volumes. A data set is uniquely defined by its data set name and the volume on which it starts. For retrieval and checking purposes, the system requires that all volumes have volume labels and that all data sets have data set labels (except for magnetic tapes, which need not have labels, in which case the system assumes that the operator has mounted the correct volume).

The data set label contains a description of the data set characteristics, such as data set organization, logical record format, logical record length, etc. Simula programs process data sets with sequential, direct or partitioned organization. In a data set with sequential organization the records are retrieved in sequence while in a direct data set the records can be retrieved in a random order. A partitioned data set can be regarded as a collection (library) of similar sequential data sets (members).

5.2 Principles of OS 360/370 data management.

Since the SIMULA object program runs under the operating system OS 360/370, it must follow the rules of Data Management of this system, and it also has access to the facilities provided by the system. The most important facilities are device independence of the source program and run-time specification of I/O processing parameters.
The most important rule is that each file must be defined by a data definition statement (DD-statement), which logically connects the file to a data set at run-time.

5.2.1 Device independence restrictions.

Device independence means that the source program does not refer to actual devices, so it can be different devices without recompilation, depending on availability. The device characteristics must, however, be compatible with the functions requested by the program. The allowed devices for the different file classes are given in table 5.1. Infiles, outfiles and printfiles process sequential or partitioned data sets, whereas directfiles process direct data sets.

*. file *. device *.	infile	outfile	printfile	directfile
card reader	X	I	I	I
line printer	I	(X)	X	I
disk	X	X	X	X
drum	X	X	X	X
tape unit	X	X	X	I
card punch	I	X	(X)	I

Table 5.1 Device-file class correspondence.

I: illegal combination.

X: legal combination.

(X): legal but not recommended.

5.2.2 Run-time specification of I/O processing parameters.

A SIMULA source program will, in general, not contain detailed information on the processing mode for a file. OS 360/370 allows a large number of alternatives for such things as record format, record length, block size, number of buffers, buffer length, recording density (tape), etc. to be specified at run-time by means of the DD-statement. Some of these parameters are permanent characteristics of the file (data set), and they need only be specified in the job step that creates it. When the data set is used later, the characteristics will be obtained from the data set label by the control program (this does not apply to magnetic tapes without standard labels).

5.3 Data definition statement (DD-statement).

The DD-statement serves the following purposes:

i) It defines the correspondance between the program unique filename and the system unique file identification.

ii) It supplies additional information needed to find an existing data set, when this is necessary.

iii) It tells the system how to handle the data set when the job step is finished.

iv) It defines whether write operations on a sequential data set are going to start at the beginning or at the end of the data set, that is whether the program will add to the data set or replace it (outfile).

v) It indicates where a new data set is to be created and how much space it will occupy.

vi) It defines data set characteristics for a new data set.

vii) It can be used to specify special processing options for a new or old data set.

The items above are defined by a number of keyword parameters. The keyword parameter DCB= has a large number of subparameters used for purposes ii), vi), and vii).

The keywords relevant to purposes i) - vi) above are:

i) : DSNAME=

ii) : UNIT=, VOLUME=, LABEL=
DCB subparameters DEN=, TRTCH=

iii) : DISP=

iv) : DISP=

v) : UNIT=, VOLUME=, LABEL=, SPACE=, SYSOUT=

vi) : LABEL=, DCB subparameters DEN=, DSORG=, RECFM=, LRECL=, TRTCH=, BLKSIZE=

vii) : DCB subparameters BUFNO=, EROPT=, HIARCHY=, OPTCD=, MODE=, STACK=

5.3.1 Binding a file to a data set.

Each file object of a SIMULA program will, at execution time, be connected to a data set. The binding is effected by a DD-statement, which defines a data set and a DD-name, and which is supplied to the execution job step. When the file object is created it is given a DD-name which is the FILENAME parameter of the object. The DD-name should be a valid SIMULA identifier, possibly with trailing blanks. 8 characters are significant. When the file is opened it is logically connected to the data set defined by the DD-statement with a matching DD-name. If no such DD-statement exists, the SIMULA program is terminated with a diagnostic. Subsequent calls of OUTIMAGE/ INIMAGE will cause records to be written/read on this data set. Several files may refer to the same data set, i.e. an infile may read data written by an outfile earlier in the program. This can be achieved by either giving the same DD-name to the files or defining the same data set in several DD-statements. The files should not, however, be open simultaneously.

This also goes for different members of the same partitioned. Use of the same dataset in different DD-names in parallell will give unpredictable results at run-time.

> Note: If the catalogued procedures described in 2.4 are used to execute an object program, the DD-name should be preceded by 'GO', indicating that DD-statement is added to the GO step of the procedure.

5.3.2 Creating a data set on magnetic tape, disk or drum.

5.3.2.1 Naming a data set.

A data set which will be used in more than one job step must be given a name when it is created. The name is supplied by the DSNAME= parameter. The name may be simple (e.g. LLIB) or qualified (A.B.LLIB). If the data set will be referred to in other jobs, it is preferably catalogued (5.3.6). In order to catalogue a data set with a qualified name 'qual.sequence.name', the qual.sequence must be the name of an index in the catalogue. Indexes are built with the IEHPROGM utility program (see (9)).

If the data set is only used in later steps of the job, it can be assigned a temporary name of the form &&name, where name is a simple name. Such data sets should be passed (5.3.6) to the later job steps of the job.

5.3.2.2 Allocating a data set.

When a data set is created, a volume or volume class on which it is to be allocated must be indicated. There are two ways to specify the volume: by specific or nonspecific volume request.

Specific request

A permanent data set should be allocated by a specific request identifying the volume on which the data set is to be placed. A specific request is effected by giving either

i) the device number and the volume serial number, e.g.

```
UNIT=3330,VOLUME=SER=111111,
```

when you want to put the data set on the 3330 disc pack with several number 111111, or

ii) the name of a data set catalogued on the same volume, e.g.

```
          VOLUME=REF=MYLIB,
```

where MYLIB is the data set name of a data set catalogued on the volume.

Nonspecific request

For temporary data sets it is usually enough to make a nonspecific request, in which case the system chooses a suitable volume depending on general specifications you supply by means of the UNIT=parameter. In each installation a number of unit group names can be used to specify the type of volume. Common names are SYSSQ for tape or disk, SYSDA for disk or drum, and DRUM for drum.

In applications with several files, a considerable increase of performance can be achieved by carefully considering the times at which each of the data sets are processed, and requesting separate access mechanisms for files operated at the same time.

As an example, consider a program that operates in two passes. The first pass reads a catalogued data set and produces intermediate results on a temporary sequential file. The second pass reads the intermediate results and produces a new data set. Since the input and output data sets are not used in the same pass, they can use the same access mechanism, but the temporary data set should be separated from both the input data set and the output data set:

```
 -----------------------------------------------------------
!                                                           !
!   //STEP1    EXEC  PGM=TWOPASS                            !
!   //SYSOUT   DD   SYSOUT=A                                !
!   //SYSIN    DD   DSNAME=INPUT,DISP=OLD                   !
!   //OUTSET   DD   UNIT=AFF=SYSIN,DISP=(NEW,PASS),         !
!   //              SPACE=(...),DSNAME=&&OUTSET             !
!   //TEMPSET  DD   UNIT=(SYSDA,SEP=(SYSIN,OUTSET)),        !
!   //              SPACE=(...)                             !
!-----------------------------------------------------------!
```

The DD-names of the three files are SYSIN, OUTSET and TEMPSET.

The DD-statement with DD-name SYSOUT is mandatory, and this data set will contain possible diagnostics. The UNIT= parameter of OUTSET specifies affinity with SYSIN, i.e. this data set will be allocated on the same volum as INPUT. The UNIT= parameter of TEMPSET indicates separation from OUTSET and SYSIN and direct-access storage (SYSDA). The separation request will be ignored if it cannot be satisfied.

The SIMULA program may look like the following, with the detailed processing of data removed:

```
        BEGIN        REF(OUTFILE) TEMPOUT,OUTPUT;
                     REF(INFILE) TEMPIN;
                     ...

        PASS1:       TEMPOUT :- NEW OUTFILE ("TEMPSET");
                     TEMPOUT.OPEN(BLANKS(80));
                     ...

        PASS1LOOP:   INIMAGE; ... TEMPOUT.OUTIMAGE;
                     GOTO PASS1LOOP;

        PASS2:       TEMPOUT.CLOSE; TEMPOUT :- NONE;
                     SYSIN.CLOSE;
                     OUTPUT :- NEW OUTFILE("OUTSET");
                     TEMPIN :- NEW INFILE("TEMPSET");
                     TEMPIN.OPEN(BLANKS(80));
                     OUTPUT.OPEN(BLANKS(80));

        PASS2LOOP:   TEMPIN.INIMAGE; ... OUTPUT.OUTIMAGE;
                     GOTO PASS2LOOP;

        ENDPROG:     TEMPIN.CLOSE; OUTPUT.CLOSE;
                     OUTTEXT("TWOPASS FINISHED");
        END TWOPASS;
```

Note: The performance can be increased significantly by specifying proper blocking of OUTSET and TEMPSET (5.3.7). The SPACE requests will be discussed in the next section. Unit affinity can only be requested for removable volumes.

5.3.2.3 Direct-access storage.

Except for the volume or volume type, one must specify the space a data set will occupy. A data set on a direct access device may occupy several disjoint areas, called extents.

The first extents are allocated when the data set is created, and they constitute the prime area of the data set. Additional extents are allocated when all previous extents are filled by a process called secondary allocation. Secondary allocation will occur up to 15 times, but after that the data set must be restructured.

The format of the space parameter is

```
SPACE=(units,quantity)
SPACE=(units,(quantity,increment))
SPACE=(units,(quantity,increment),,,ROUND)
SPACE=(units,quantity,,,ROUND)
```

unit is the unit in which the space request is given. It could be:

  i) average block length in bytes
 ii) TRK: tracks
iii) CYL: cylinders

quantity is the number of units to be allocated as prime area.

increment is the number of units allocated to each additional extent.

ROUND this indicates that the control program will round the size of each extent upwards to an integral number of cylinders. Extents are also allocated on cylinder boundaries.

Notes: Specify units with average block length if the volume request is non-specific, since it may be satisfied by devices with different track capacity.
ROUND can increase performance (see (9), p. 49). There are several alternate space allocation methods, which are given in (see (9), p. 47).
The track and cylinder capacities are listed in (see (7), p. 158).

If the data set is going to have direct organization (directfile), you must also specify DCB=DSORG=DA in the DD-statement. No other DCB subparameters may be specified. A direct data set will never use secondary allocation, so the increment is superfluous.

5.3.2.4 Magnetic tape.

On a magnetic tape any SPACE parameter in the DD-statement is ignored. Instead, the data set serial number in the LABEL= parameter must be specified. The format of this parameter is

```
        LABEL=(n,SL),
```

where n is the ordinal number of the data set on the reel, and SL indicates that the data set has standard labels.

A tape volum which is going to have standard labels must be initialized with the IEHINITT utility program (see (9)).

Tapes with no labels should be used only if they are used or produced outside the System 360/370 OS environment.

Note: When a data set with serial number n is written, all data sets on the volume with serial number >n are lost.

5.3.3 System output.

When a data set is created on a line printer or a card punch, it will logically leave the system, since the system does not recognize labels on the corresponding data carriers (printer listings and punched card files). Therefore, the DD-statement will not contain a DSNAME= parameter or a volume request, but the kind of output is specified with the SYSOUT= parameter. The parameter is a letter or a digit, usually A for printed output and B for punched output. In some systems (MFT or MVT) the SPACE= parameter can be used to limit the number of lines if the program is looping. An example of the use of SYSOUT=A is found in section 5.3.2.2.

5.3.4 Retrieving a data set.

A data set is retrieved by the system if it is given the data set name and the volume on which it resides. The volume information can be given explicitly in the DD-statement in the form of a specific volume request, or is implicit if the data set is catalogued or passed from a preceding job step of the job. The data set name is given in the DSNAME= parameter. In addition, the DISP= parameter must indicate that the data set is to be retrieved (5.3.6).

For a magnetic tape you must give the density, which is needed for reading the label, e.g. DCB=DEN=1 and the data set serial number in the LABEL operand.

5.3.5 System input.

When an input data set is small, it is conveniently put in the job input stream among the control statements. The DD-statement for the data set has the following format:

```
        //ddname  DD  *
```

and indicates that the data set follows this line in the input stream.

In a PCP system, this must be the last DD-statement of the job step, and only one such data set can be processed in any single job step.

The line images may not have // or /* in the first two columns.

If you need to input datasets containing e.g. // in the first two positions the DD-statements to be used is

```
        //ddname  DD  DATA
```

The end of the data set is signalled with a line containing /* in the first two columns.

5.3.6 Disposition of a data set.

The DISP= parameter serves several purposes:

i) it indicates whether a data set is to be created or retrieved

ii) it indicates whether it is to be catalogued, passed, kept or deleted

iii) it defines positioning of sequential data sets.

The format of the DISP= parameter is:

```
        DISP=(parm1,parm2), or DISP=parm1,
```

where parm1 is OLD, SHR, MOD or NEW, and parm2 is CATLG, KEEP, PASS, DELETE or is absent.

If the DISP= parameter is missing, DISP=(NEW,DELETE) is assumed, and DSNAME is not necessary.

parm1: meaning:

OLD An existing data set is to be retrieved. The data set will be locked, so that no other tasks in a multiprogramming environment can access it during the job step.

If the data set has sequential organization, it will be positioned to the beginning, so that read operations will read the entire data set and write operations will replace the existing records.

SHR This parameter has the same meaning as OLD, except that other tasks in the system may access the data set concurrently. Do not specify SHR if the job step writes on the data set, unless the write operations are synchronized by ENQ/DEQ macro instructions (these can be issued in an external assembly procedure of a SIMULA program, (see (7))).

MOD The system will check if a data set with the same name has been passed to the step. If this is not the case, or if the system cannot find the volume information for the dataset, a new data set will be created (and the DD-statement must contain enough information to create it). Otherwise the existing data set is used, and it is positioned to the last record, so that write operations will add records to the data set.

NEW A new output data set is to be created. Write operations will add records, starting from the beginning of the data set.

parm2: meaning

CATLG The data set will be catalogued at the end of the job step. It can later be retrieved by name alone.

KEEP The data set will be kept after the job step. It must later be retrieved with a specific volume request.

A sequential data set (infile, outfile) will be rewound after each close, and a later opening of the same data set (not necessarily the same file) will process it from the beginning.

PASS The data set is passed to the next job step that uses it. The next job step will retrieve it by name, if MOD or OLD is specified in the parm1 field. A sequential data set will not be rewound when closed, and if it is reopened, records can be added to it.

DELETE The data set is deleted at the end of the job step. It is rewound when closed.

A third subparameter can also be used. This will specify what is to be done with the dataset if the job step obnormally terminates (See (6)).

5.3.7 Sequential data sets: characteristics and processing options (outfile, infile).

A sequential data set will have several characteristics which are determined from the DD-statement when the data set is created.

These are record format, logical record length, block size, and, for data sets on magnetic tape, recording density and tape recording technique. They are determined from DCB subparameters RECFM=, LRECL=, BLKSIZE=, DEN= and TRTCH=, respectively.

5.3.7.1 Fixed record length.

If characteristics are not specified the data set will have fixed length and unblocked records, i.e. each time OUTIMAGE or INIMAGE is called, a block of fixed size is written or read. The length of all blocks and records in the data set will be the same as the length of the image passed as a parameter to OPEN. When the data set is connected to an outfile, the image length of the latter must always be less than or equal to the record length. When it is connected to an infile, its image length must be greater than or equal to the record length. In both cases some CPU time and core storage is saved by having image length equal to the record length of the data set.

5.3.7.2 Blocked records.

A data set with fixed length and unblocked records provides the fastest processing of blocks with a given length. It will, however, often be the case that the logical units of information (customers, employees, projects, loads, charges, measurement points, etc.) require much less space than the optimum block size for the device. In these cases one can maintain logical clarity of the source program and still use optimal size of blocking the records (images) together. A blocked data set is obtained by specifying RECFM=FB,LRECL=image length, BLKSIZE=block size in the DCB parameter of the DD-statement. The block size must be a multiple of the image length. It must in no case exceed the track capasity of the device of 32760.

Example: 80 byte images blocked 20/block:

```
         //BLOCKSET   DD  DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600),...
```

5.3.7.3 Variable and undefined record formats. (see (7), p. 58).

If there are considerable variations in the amount of information per image, space can be conserved on secondary storage by writing variable (V or VB) or undefined (U) format records. The length of image determines the length of each record when written.

On input, the image must contain the record if variable format is used, whereas for U format data sets the image length determines the number of bytes read. The LRECL and BLKSIZE parameters are filled in as shown in Table 5.2

RECFM=	LRECL=	BLKSIZE=
V	max+4	-
VB	max+4	> max + 8
U	-	max
F	max	-
FB	max	n*max

Table 5.2: LRECL and BLKSIZE parameters

max is the maximum length of image.

BLISIZE= may never exceed the track capasity or 32760, whichever is smallest.

5.3.7.4 Carriage control character. (see (7), p.60)

A carriage control or stacker bin selector character may be the first character of each record of a sequential data set. This is indicated by appending A (ASA control character) or M (device dependent control character) to the RECFM= code. This character must be supplied in the first position of image. It will be written together with the rest of the image on secondary storage (disc, drum, tape). It will not appear on a printer listing or on punched cards.

5.3.7.5 Additional DCB subparameters. (see (6) and (7))

The following additional subparameters can be used in special applications (none is compulsory):

BUFNO= Number of buffers. If omitted, two buffers are allocated, which is the most suitable in most cases. If there is lack of storage BUFNO=1 is specified. If chained scheduling is used it is advantageous to have many buffers (always less than 255).

DEN= Tape recording density (@, 1, 2, 3 or 4) 2 (800 bpi) is assumed if omitted.

EROPT= Action taken when an erroneous block is read (I/O error):

- ACC : accept block. The image should be investigated character by character to avoid termination in number de-editing procedures.
- SKP : Skip the erroneous block.
- not defined : the program is terminated.

In any case a message is printed on sysout.

HIARCHY= Hierarchy of buffer storage in systems with Large Capacity Storage support (@ or 1). @ (high speed core) is assumed if omitted.

MODE= Mode of cared reader or card punch (e.g. E indicating EBCDIC).

OPTCD= Special services requested

- C: Chained scheduling. Several I/O requests are grouped together and performed in one channel operation. This is of advantage if the data set has been given a small blocksize because of the core requirements of one program and is also processed by a program with small core requirements. Specify channel separation and many buffers for better performance.
- Q: Conversion EBCDIC/ASCII. The conversion is from EBCDIC to ASCII on output, and the opposite on input. Note that the maximum block size which can be used is 1600. Note also that only non-labeled tapes can be handled.

STACK= Stacker bin selection on a card punch (1 or 2).

TRTCH= Magnetic tape recording technique (C,E,T). Note that the parameters TRTCH, STACK and MODE are mutually exclusive.

(See (6) for further details)

5.3.8 Printfile data sets.

All printfile data sets have variable length and blocked records with ASA control character.

The control character is inserted automatically and is controlled by standard procedures (lines per page, spacing, eject).

Performance can be increased by giving

```
            LRECL = (maximum image length + 5) and

          BLKSIZE = (>=maximum image length + 9)
```

The other DCB parameters (except RECFM) of section 5.3.7 can be specified, but most of them will usually not be applicable.

5.3.9 Direct data sets.

A direct data set will consist of a number of fixed length blocks. The block length will be equal to the image length when the file is first opened. The first time the data set is opened, it is initialized by consecutively writing the open text parameter until the prime area is filled. No DCB subparameters except DSORG=DA may be specified, and this must be specified when the data set is created. The image length must at all times be equal to the block size of the data set.

The access method is BDAM. Addressing is by relative block number.

When INIMAGE or OUTIMAGE is called and LOC is out of range, the end of file text is supplied if INIMAGE was called, but image is unaltered if OUTIMAGE was called. The Boolean procedure ENDFILE indicates whether the last i/o call on the file was successful (FALSE) or not (TRUE). ENDFILE is reset after a successfull i/o operation. Two unsuccessful calls may not occur in sequence.

Keyword	infile	outfile	printfile	directfile
BLKSIZE=	1)	2)	max.blocksize	
BUFNO=	3)	3)	3)	
DEN=	0,1,2,3,4(tape)	0,1,2,3,4,(tape)		
DSORG=				
EROPT=	ABE,SKP,ACC 5)			
HIARCHY=	0, 1 9)	0, 1 9)	0, 1 9)	0, 1 9)
LRECL=	1)	6)	7)	
MODE=				
OPTCD=				
RECFM=	1)			
STACK=	1 or 2 8)	1 or 2 8)		
TRTCH=	C,E,T (tape)	C,E,T (tape)		

Table 5.3: DCB subparameters.

1) must be specified for tape without standard labels if the default values were overriden when the data set was created.

2) for V and U formats, a maximum blocksize can be specified when the data set is created.

3) The system default values (two buffers) may be overriden.

4) must be specified.

5) Defines system action on bad blocks.

ACC: accept the bad record

SKP: skip the bad records

ABE: terminate processing

6) U format: do not specify LRECL
V format: maximum length of image + 4
F format: length of each record, can be defined when data set is created.

7) maximum length of image + 5.

8) card punch.

9) 0 is default.

5.4 End-of-file text.

When all records of a sequential input data set have been read, an end of file condition exists. On the next call of inimage, the Boolean variable endfile (sensed by the Boolean procedure ENDFILE) is set and the end of file text is stored in image. The end of file text consists of /* in positions 1 and 2 and of blanks in the other position. If the length of image is greater than 260 bytes, the character after the 260th are not blanked.

Be careful with making programs depend on the EOF-record. The EOF-record definition in SIMULA Common Base is the text value "!25!". Programs making use of this value will not be compatible with other SIMULA systems.

5.5 Sysin and sysout.

The standard files sysin and sysout are defined with the ddnames 'SYSIN' and 'SYSOUT', respectively. DD-statements for these files must always be supplied. If a program does not use sequential input, sysin can be defined as a dummy data set by the statement

```
        //SYSIN  DD  DUMMY
```

and core storage can be released by closing it at once.

The standard output file sysout will be used to produce any run-time error or warning diagnostics. If sysout is closed, these diagnostics will appear on the console, but if SYSOUT, OUTIMAGE is called, the program is terminated.

If sysout is closed, a dump or trace should not be requested since the console typewriter is fairly slow and is not supposed to be used for such purposes.