AN IMPLEMENTATION OF THE NEW CHILL-I/O

Thomas Mehner, Jürgen F.H Winkler

Siemens AG Munich
Federal Republic of Germany

ABSTRACT

The paper describes the implementation of the new I/O facility of CHILL-84 [2]. Based on the encouraging experiences with this prototype implementation the design of a second implementation for two different operating systems is described. A critical assessment of the new CHILL I/O is given in the last section of the paper.

1 INTRODUCTION

Up to now the language CHILL, as described in Z.200 [1], does not provide any input/output facility. But in general a program communicates with its environment.
Exchange programs eg. communicate with the hardware equipment of the exchange. Support programs, as eg. the compiler, communicate with external files containing the source and the object programs. Every implementor of CHILL had to add his own I/O features to the language. Since the I/O facilities of different implementations differ compatibility and portability are compromised.

During the current study period of the CCITT the sub-working party XI/3-2 developed a draft proposal of an I/O facility for CHILL. It is included in the draft recommendation Z.200 [2]. After being approved by the plenary assembly in October 1984 it will be part of the revised Z.200 recommendation.

The introduction of the new I/O facility has the following goals:

- Improvement of compatibility between CHILL-compilers.
- Introduction of a language feature compatible with the rest of the language and extending the language as few as possible.
- Provision of a tool for safe and easy file manipulation and data transfer.
- Simplicity of implementation.

The proposed I/O facility matches these goals quite well; but some features are still missing. Most important is the fact that text I/O is still not incorporated into the language.

2 OVERVIEW OF THE NEW CHILL-IO

The most significant characteristic of this language concept for input/output is that it is based on a three-state-model. That means that from the viewpoint of the manipulating program an external file can be in three different states.

- in the free state where no connection exists;
- in the file handling state a file is known to the program. It can be created, deleted or its properties can be changed;
- in the data transfer state where read and write operations on a file are possible.

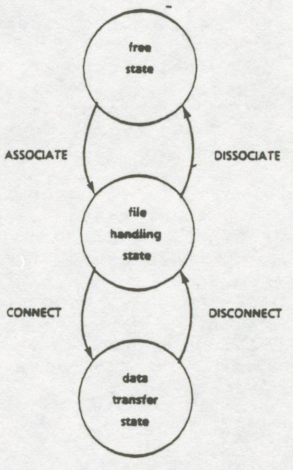

Fig. 1. States of an external file

Two new modes are introduced in CHILL in order to describe and access external data objects from within the program.
An ASSOCIATION object identifies an outside world object (file, printer, reader, terminal etc.) called file in this paper. The ASSOCIATION object has attributes attached, that describe the properties of the associated file.
The operation ASSOCIATE creates an association to a file and initializes the attributes appropriately. After a successful ASSOCIATE the corresponding file is in the file handling state.
The operation DISSOCIATE terminates such an association and returns the file into the free state.

The second of the newly introduced modes is the ACCESS mode. An ACCESS location is used to perform data transfer between files and the internal "world" of the program. The

declaration of an ACCESS location specifies the structure of the data items used in this data transfer.
The operation CONNECT connects an access location to an association. In this operation additional information concerning the positioning of the file associated and the mode of operation (read, readwrite) can be specified. A successful CONNECT puts the associated file into the data transfer state.
The operation DISCONNECT separates an access location from an association. No more data transfer is possible. The file associated is put into the file handling state.

The I/O-facility contains some procedures to read the attributes of an association.

EXISTING (ass) delivers true if and only if a file is associated to the ASSOCIATION ass.

READABLE, WRITEABLE, INDEXABLE, SEQUENCIBLE and VARYING give information about the corresponding attribute of an association.

For file-management three procedures are available.
CREATE with an association as parameter creates a file according to the attributes of the association and associates this newly created file.
DELETE dissociates the file attached to the given association and deletes the file.
With MODIFY properties of files can be changed. The possible parameters and their meaning are implementation defined.

Furthermore three procedures with access locations as parameters are introduced.
GETASSOCIATION (acc) delivers the association object connected to the parameter access location "acc".
GETUSAGE returns readonly, writeonly or readwrite.
OUTOFFILE returns true if the actual positioning of the file concerned is not inside the file.

For data transfer two procedures are available:
READRECORD (acc, ind, st_loc) transfers a data record from the file associated with "acc" into the internal location "st_loc".
The parameter "ind" is used if the associated file is an indexsequential file. "ind" gives the index of the record to be transferred.
WRITERECORD (acc, ind, rec) transfers the data record "rec" to the file associated with "acc". "ind" gives the index in case of an indexsequential file.

3 PROTOTYPE IMPLEMENTATION

To test the possibility of realization and to get experience with the new I/O, we decided to implement it as a prototype. That means that we used existing tools as far as possible. One called "portability package" (PP) offers many operating system functions on CHILL-level.
On the other hand our CHILL compiler already implements a simple I/O facility. The compiler runtime system uses the PP and a package for ISAM-file handling.
Further functions needed for the realization of the new I/O are implemented as assembler routines with operating system macro calls.

A further restriction was that the existing CHILL-compiler should not be changed. This is one reason for some restrictions and modifications that we had to adopt in our prototype.

In general our goal was to implement the new I/O according to the CCITT draft proposal.
The built-in functions described in chapter 2 are realized in the module CHILLIO as normal procedures and their names are granted.
The new modes ACCESS and ASSOCIATION are defined as structure respectively pointer of structure mode and are granted with forbid all if the compiler allows this restriction.
Thus our I/O package is constructed according to the principles of data abstraction [5].
The user gets the modes and can define new access locations and association objects, but can only manipulate these objects by means of the granted procedures.

The I/O system contains four modules written in CHILL. They consist of 1200 lines of code, whereof 200 are comments or empty lines.
Additional 7 routines are written in SIEMENS 7xxx assembler. Their length is about 190 lines.

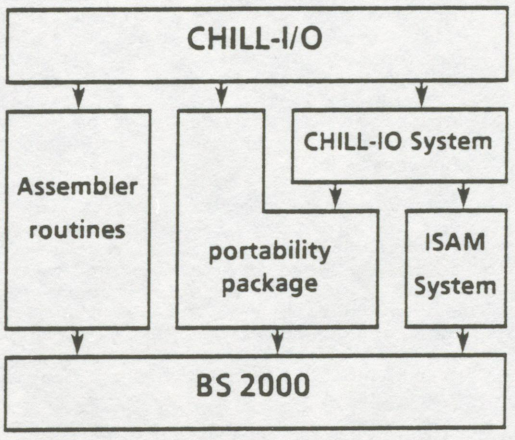

Fig. 2. Structure of the prototype implementation

The CHILL-I/O leaves some parameters of built-in functions to the implementors choice. In our case we have chosen the following:
For ASSOCIATE:
Filename is the name of a file. The format is that of the SIEMENS BS2000 file handling system.
Filemode can be SAM, ISAM, PAM, Temporary, Terminal, or Printer.
For MODIFY:
Newname is the new name of the file concerned.
Accessmode which can be read or write and variability which can be fixed, variable and undefined, to define and change the corresponding attribute of a file.

As a last point we want to give a list of the modifications and restrictions of this prototype in comparison with the draft proposal:

- The I/O-tool used by this prototype needs static declarations of "file variables". That restricts the maximum number of files used simultaneously to a number fixed in advance.
  Furthermore the modes of the records transported to and from the files are predefined in the "file variable".
  These restrictions are quite severe and limit the dynamic aspects of the new I/O-facility.

- The declaration of an access location had to be simplified. Only the synmode identifier ACCESS is used. Without compiler modification the information about the file record mode cannot be included. In our case this has to be done by assigning a mode key to a field of the access location.

- CHILL procedures have fixed parameter modes. The records to be written out or read in have arbitrary modes and thus cannot be parameters of normal CHILL-procedures.

- Our CHILL compiler distinguishes exported and imported names only if they differ in the first seven characters. That caused two minor name changes:
  ASSOCIATE => DOASSOCIATE
  CONNECTED => ISCONNECTED

- File positioning had to be simplified since the PP does not support all possibilities defined in [2].

In addition to the CCITT-proposal procedures for I/O of textlines with variable length have been implemented.

4 IMPLEMENTATION ON TWO DIFFERENT OS

As we have seen in chapter 3, the present implementation has some disadvantages (restrictions, size). Furthermore our prototype is only usable in a BS2000 environment. We have requirements to consider at least BS2000 and BS3000 for the new I/O.
Therefore we decided to design a second realization of the new I/O based directly on the macro interfaces of the two operating systems. In the following we describe this enhanced implementation, which can be conceived as part of the RTS, in terms of CHILL.
There will be one CHILL module new_CHILL_IO granting all the features available in the new I/O to the user program. This module will be the same for both OS. That means it works completely on a machine independent level. It is based on two OS specific packages. They contain the macro calls of operating system functions.
Thus, the I/O system is a typical example of a program with variants [4]. Each variant consists of two program units. One of them is common to both variants, the other is different.
This design principle strongly improves portability.

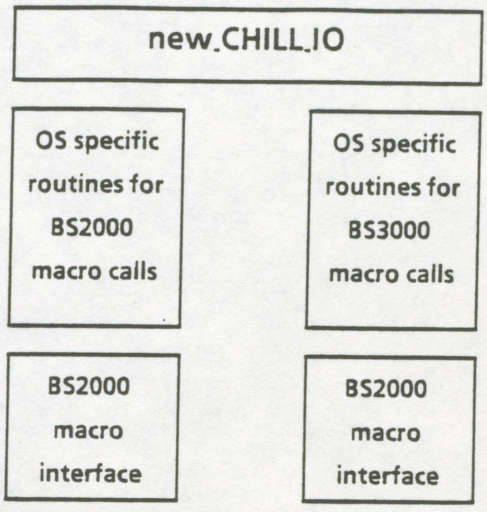

Fig. 3. Structure of the enhanced implementation

5 EXPERIENCES

Our implementation of the new I/O has been used in two projects. According to the feedback, rather good experiences have been made.
The new CHILL-I/O is a very useful and versatile facility for doing input/output. Creation and deletion of files at run time is very helpful. File handling is much more economical: file creation in advance is superfluous and files are deleted as soon as possible (not manually after program termination). Comfortable integration of features for this task into CHILL seems to be very important for many users.

The possibility of changing file attributes proved to be less essential.

The second advantage of this new I/O is the possibility of connecting and disconnecting files at arbitrary points in time. This reduces the time interval, during which a file is locked, to a minimum. This is especially important in cases of multiple access.

To summarize our experience with the new I/O, we can say, it is effective and efficient to use. Especially for library systems [4] and similar systems it is a convinient concept.

6 CRITICAL ASSESSMENT

The most important issue is the lack of text-I/O i.e. the I/O of possibly more than one object including the conversion from and to a human readable form.

A second issue is the introduction of two kinds of new objects for this I/O. A two step protocol as: free state -> file handling state -> data transfer state, can also be enforced if there is only one internal object associated to an external object (file). The internal object is then a kind of reference point or window into the external world. The protocol then defines the possible sequences of operations that may be applied to the internal object. This is the typical approach used with data abstractions [5]. The protocol of the I/O concept of CHILL-84 can be characterised by the following regular expression:

ASS CR ((CON T* (DISC | e))*

((DISS ASS) | e))* DEL DISS

which exhibits also the two step protocol. T stands for data transfer operations and e for the empty sequence.
The protocol is not symmetric because it allows some shortcuts when dissolving the junction between the program and the external file but not when establishing this junction. That means that the program must always pass explicitely through the file handling state even in those cases when it is not necessary. These cases occur quite frequently in practice.

A further issue is the file positioning. Even a programmer accessing a file only in a direct and simple way is burdened with relativ addressing (base index) and a window scheme.
Furthermore it seems to be an inconsistency, that positioning for sequential files is possible at any arbitrary record and for indexed files only three positions can be chosen: first, last and same.

A further critical issue seems to be the omission of an operation to delete file components. There are quite a number of cases where a file is not only written but where components of a file are also deleted. A first example are communication files [6] ("pipes" in UNIX*[7]) where a communication line is modelled as a sequential one-time file. In this case the producer (sender) adds components at the end and the consumer (receiver) removes components at the beginning of the file. A second example is an editor program which must be able to delete a line from a file containing eg a source program.

* UNIX is a trademark of Bell Laboratories

7 REFERENCES

1) CCITT: CHILL, Recommendation Z.200; Genf 1981.

2) CCITT: DRAFT RECOMMENDATION Z.200; Genf December 1983.

3) Sammer, W.; Schwärtzel, H.: CHILL. Eine moderne Programmiersprache für die Systemtechnik; Springer, Berlin 1982.

4) Mehner T., Tobiasch R., Winkler J.F.H.: A Proposal for an Integrated Programming Environment for CHILL. This volume.

5) Winkler, J.F.H.: The Realization of Data Abstractions in CHILL. This volume.

6) Winkler, J.F.H.: A new Methodology for I/O and its Application to CHILL. 2nd CHILL Conference, Lisle Illinois, March 1983, p.217..233.

7) Ritchie, D.M.; Thompson K.: The UNIX Time-Sharing System; C. of the ACM, 7/17 p.365..375 July 1974.

CCITT CHILL

THIRD CHILL CONFERENCE

CAMBRIDGE UNIVERSITY

ROBINSON COLLEGE
&
UNIVERSITY CENTRE

September 23-28 1984

CONFERENCE PROCEEDINGS

Sponsored by:

ITT Europe Engineering Support Centre