MTA SZTAKI Közlemények 34/1986 pp. 79-89

MODULA-2 USED IN THE IMPLEMENTATION OF DATA ACCESS CONTROL MECHANISM

B. SZAFRAŃSKI
Warsaw, Poland

SUMMARY

Experience gained by using Module-2 for implementing a data security mechanism based on the data access control is presented. It entails some questions of the data access control idea but its main aim is showing the use of Module-2 in a real application. We show how in the simple way you can amplify capabilities of Modula-2 as a result of introducing data access control features.

1. INTRODUCTION

The problem of data security for any programming language system is considered in the three following aspects:

- security of standard translation facilities (like-compiler, linker, library of subprograms),
- security of translation products,
- security of data units processed by the programs written in this language.

Two first questions belong to the wider security problem of standard library and files. Such kind of data security should be resolved by the operating system and will not be considered in this paper. However, we focus on the third problem. It touches, in general principle, some questions of the language system facilities to define access rules and to detect the violations of these rules. Development of the programming languages points out, that using their capabilities to build

data security mechanisms is absolutely right. Among other things, the necessity of data protection has led to such high level languages features as module structure with IMPORT and EXPORT lists, possibility to define the own structure, SCOPE rules and so on. However in data processing systems the principal question is processing of data stored in the external files. One can find out that above listed facilities refer to internal program objects and not influence directly the data security in the external storage. To confirm above we can state, that the most popular languages as COBOL, FORTRAN, PASCAL, PL/1 have no capabilities of such data security. This paper describes the experiences gained by using Modula 2 for implementation a data security mechanism based on the data access control (DACM). The first implementation was done by Mr Wlodzimierz Kubalski.

2. FILES IN MODULA-2 FOR RT-11

Access operations to data file in this disc storage always depend on the file system, which is a part of the operating system. For this reason and to save machine's independence of Modula-2 system, operations mentioned above can not be determined by the Modula-2 implementation [WIRTH80]. However, to provide program compatibility, the module Files (containing file processing procedures) was created by the authors of Modula-2. Each data file is identified by a value of type File, which in terminology RT-11 is named a channel number. The data file is assigned a channel number by calling either Lookup or Create procedures. In both cases, it is necessary to provide a file name. A file name consists of 12 characters and can be given in a static (as a literal) or a dynamic (by value of variable) form.

3. THE MODEL OF DATA ACCESS CONTROL

The complete model of data access control is presented in [SZAF78]. We show below only those elements of the model, which are important for the aim of this paper. These elements create the simplified model of data access control, which determines capabilities and operational rules of implemented mechanism [SZAF81].

Definition

Data access control model is an ordered triple:

$$S = \langle Z, T, \Psi \rangle$$

where

Z - finite object names set,

$Z = U \cup B$

where:

U - finite active objects set,

B - finite passive objects set

T - finite operation names set,

Ψ - set of model relations,

$\Psi = \{\Psi_1, \Psi_2\}$

where:

$\Psi_1 \subset U \times B \times T$, Ψ_1 is an access relation, which determines access privileges of active to passive objects,

$\Psi_2 \subset T \times T$, Ψ_2 is operation scope relation. To clarify this relation let us introduce formally the definition of the operation scope:

Definition

An operation scope $t_i \in T$ is an operation set $\{t_j\} \in 2^T$ such that, the ability of doing the operation t_i implicates the ability of doing operation $t_1 \in \{t_j\}$.

Definition

The operation scope $\{t_j\}\in 2^T$ of operation t_1 is smaller (equal) than the operation scope $\{t_k\}\in 2^T$ of the operation t_2 if and only if $\{t_j\}\subset\{t_k\}$.

The operation scope relation Ψ_2 is defined on pairs of operation names.

Definition

For $t_1,t_2\in T$ we say, that $(t_1,t_2)\in\Psi_2$ if and only if the operation scope t_1 is smaller (equal) than the operation scope t_2.

Definition

The process of data processing, from data access control point of view, is defined in the following way:

$$P = \{(\alpha_1,\beta_1,\gamma_1),\ldots,(\alpha_i,\beta_i,\gamma_i),\ldots,(\alpha_I,\beta_I,\gamma_I)\}$$

where:

$$\alpha_i\in U,\ \beta_i\in B,\ \gamma_i\in T \quad \text{for} \quad i=1,2,\ldots,I.$$

Definition

Process P is correct, from data access control point of view, if and only if:

$$\forall[(\alpha_i,\beta_i,\gamma_i)\in P]\exists[(\alpha_i,\beta_i,t)\in\Psi_1]\wedge(\gamma_i,t)\in\Psi_2$$

For implementation reasons it is better to check legality of a process by using the checking function f:

Definition

$f : U\times B\times T\rightarrow\{0,1\}$ in such a way, that

$$f(\alpha,\beta,\gamma) = \begin{cases} 1 \text{ if and only if } \exists[(\alpha,\beta,t)\in\Psi_1]\wedge(\gamma,t)\in\Psi_2 \\ 0 \text{ in other case} \end{cases}$$

Now we can say that:

Process $P = \{(\alpha_1,\beta_1,\gamma_1),\ldots,(\alpha_i,\beta_i,\gamma_i),\ldots,(\alpha_I,\beta_I,\gamma_I)\}$

for $\alpha_i \in U$, $\beta_i \in B$, $\gamma_i \in T$ is correct if and only if:

$$\prod_{i=1}^{I} f(\alpha_i, \beta_i, \gamma_i) = 1$$

4. DATA ACCESS CONTROL MECHANISM - DACM

4.1 Assumptions of DACM

A selective DACM is based on the model S and entirely respects its features describes above. The concrete forms of model elements, which determine the implementation conditions of DACM are the following:

- a set of active object names U is a set of character strings representing the user identifications,
- a set of passive object names B is a set of RT-11 file names,
- an ordered set of operation names which contains operation names provided by system Modula-2 in the module Files (see Fig.1):
  T={Nil,Lookup,ReadBlock,WriteBlock,Delete,Rename,Create},
- ordering of set T, important for operation scope relation Ψ_2 , is defined by above enumerating of set elements. This means that the operation scope is growing from left to right. As you see the file T does not include the Close and Release operations because they only have the technical significance. Moreover, we have introduced the operation Nil to forbid any processing of a data unit.

Such model allows DACM to grant access selectively to the files, depending upon user identification and privileges that are included in the user access privilege file - USER.PRV.

This file is an implementation of access relation Ψ_1. The organization of the file USER.PRV bases on capability list (C-list) and it can be treated as known security matrix [HOFF77]. The checking function f we should consider as two functions, because of static and dynamic features of file name declaration function f_t is a translation time checking function

```
DEFINITION MODULE Files; /*Ch.Jacobi, for RT-11 */
FROM SYSTEM IMPORT  ADDRESS, WORD;
EXPORT QUALIFIED FILE, File Name,Lookup,Release,Create,
Delete,Close,WriteBlock,ReadBlock,Rename;
TYPE FILE  = [0..15] /* chanel number */
File Name  =  ARRAY [0..11] OF CHAR  /*File name*/
Procedure  Lookup  (f:File, FileName, VAR reply INTEGER)
            /*lookup file f in dictionary*/
```

Fig.1 A fragment of definition module Files

and f_r is a run time checking function. The function f_t is called during translation whenever any file access operation occurs. In the case of the static declaration of file name the legality checking is performed. In the other case, the call of function f_r is introduced in the procedure form to the user module. Later at run time f_r is invoked whenever it is required (see Fig.2).

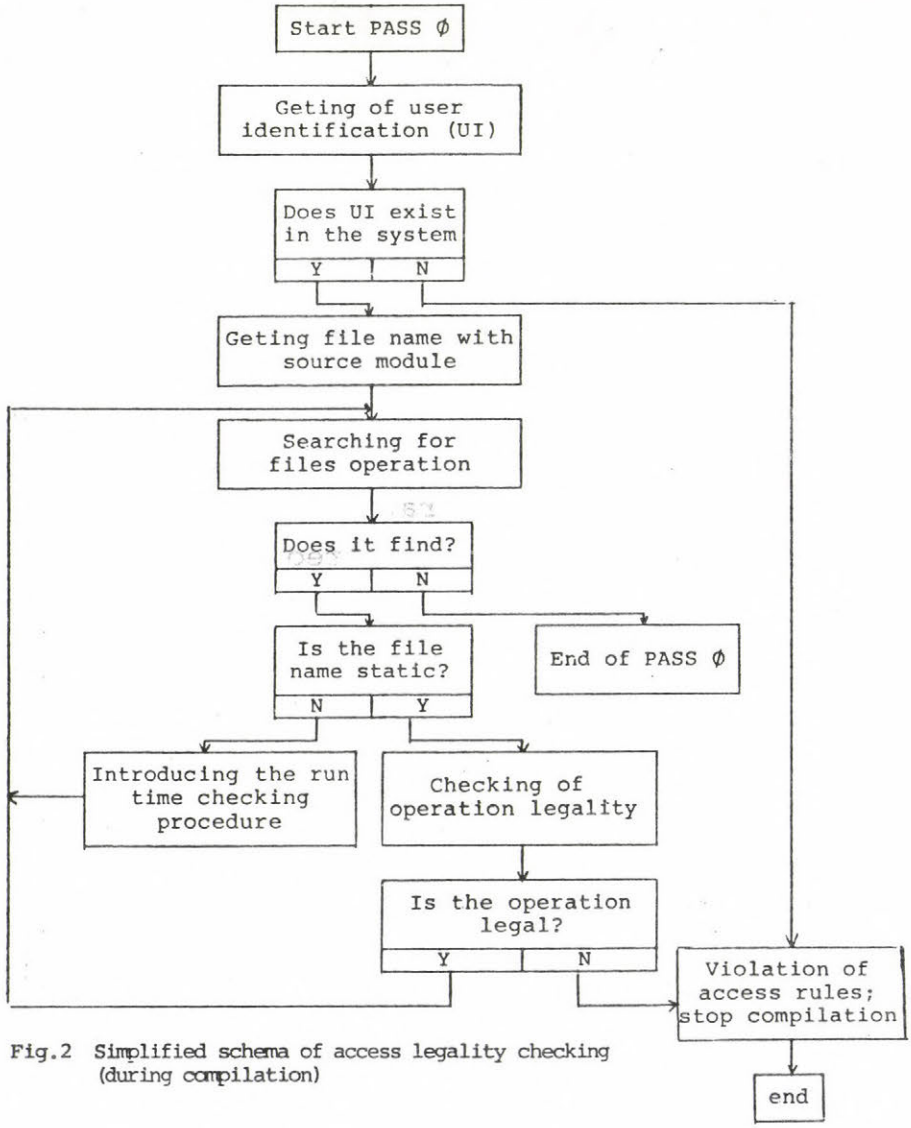

Fig.2 Simplified schema of access legality checking (during compilation)

4.2 Architecture of DACM

Implementation of the functions f_t,f_r and other elements of DACM requires additional language system features, a modification of standard language processors, and/or a language preprocessor. In our case the elements of DACM showed on the Fig.3 create two subsystems:

- subsystem of creating and maintenance file USER.PRV. The main element of this subsystem in the program module EDIPRV, which is used by data administrator to manage access privileges file USER.PRV.
- subsystem of operation legality checking. The main program module PREPROC of this subsystem realizes following functions:
  - •• geting and examining user's identification,
  - •• geting of file name including source user's module,
  - •• searching of file access operations and when occurs:
    - ••• static file name - checking of operation legality (function f_t),
    - ••• dynamic file name - introducing into user's module suitable procedure from module VERYFIER (function f_r) and into IMPORT list the name VERYFIER.

Modules USEMOD (access procedures to source user's module), USEPRV (access procedures to file USER.PRV), VERYFIER (checking procedures) include the auxiliary procedures which are used by PREPROC and EDIPRV.

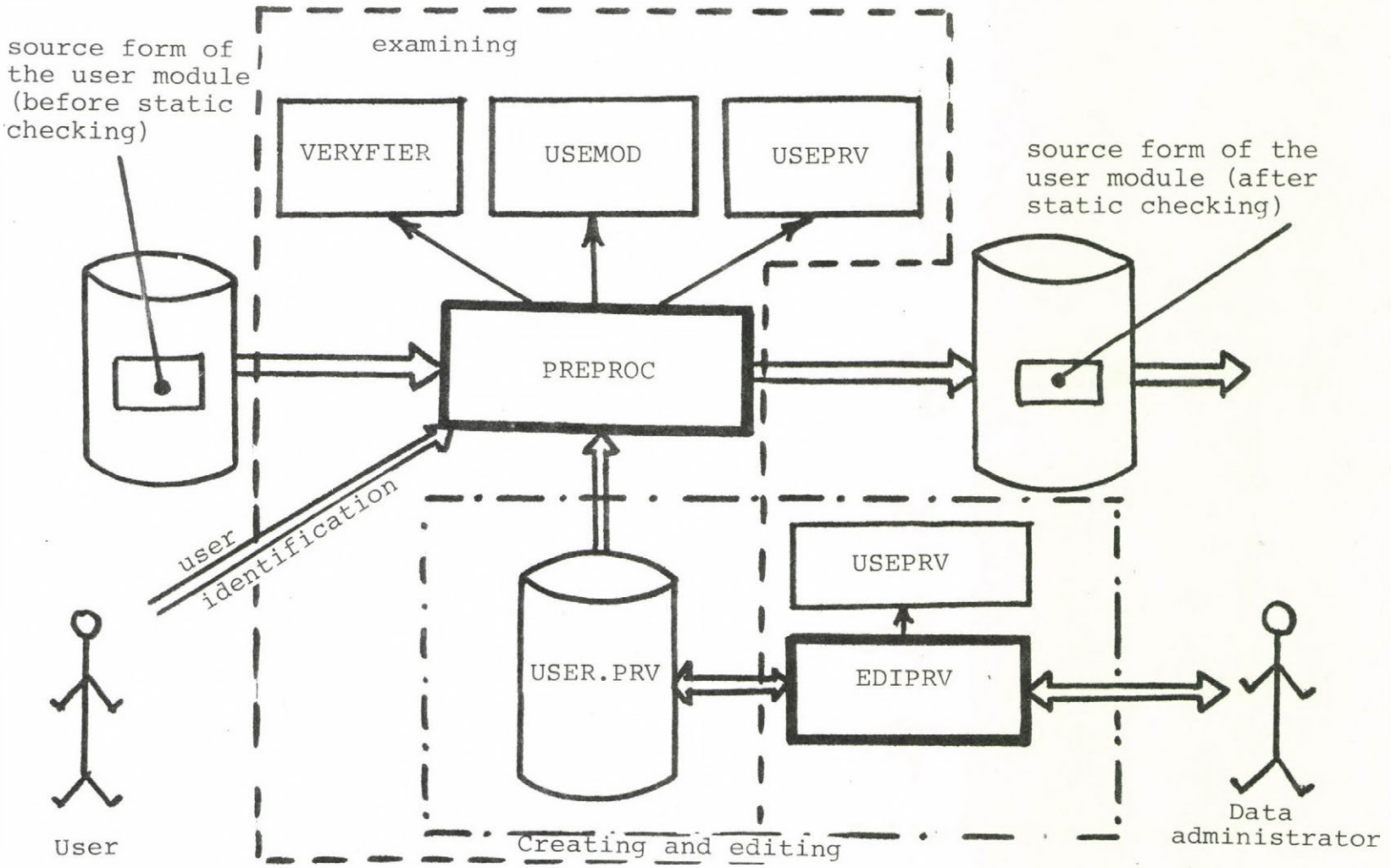

Fig.3 Elements of DACM

4.3 Compilation in MODULA-2 with DACM

The compiler is running on the Modula-2 system. It is written in MODULA-2 itself and generates code for PDP-11. The compiler is organized in a base part and several so called main parts. The base part remains in the memory during the whole compilation and schedules the execution of the main parts, which are called sequentially [WIRTH81]. The compiler has five passes (from 1 to 5) and the interpass files. According to the above, the DACM mechanism in current version was implemented as a PASS Ø of the compiler. The main part PREPROC is linked to COMP. Therefore some changes have appeared in the compile process. The compiler, as before, is invoked by typing the file name COMP but instead of string "source file" will appear the string "user identification". So one must write the corresponding UI. If it is right, the string "input file" and next "output file" will appear. The input file (default extention PRV) consists of source user's module, which will be checked by DACM. The output file is an interpass file between PASS 0 and PASS 1. It consists of the normal source file (module DEF or MOD but default extension is MOD) in terminology Modula-2 system by after checking by DACM. The following passes are running without any changes. If during PASS Ø the violation of access rules occurs, the compilation will stop and the error message will be written.

```
R MODULA <cr>
  COMP   <cr>
user ident >U 124  <cr> /*user identification*/
input file >PROG1  <cr> /*name DK:  PROG1. PRV is accepted*/
output file>PROG1  <cr> /*name DK:  PROG1. MOD is accepted*/
p0                      /*pass of access rule checking*/
p1
p2                      /*indicates succession of*/
p3                      /*activated compiler passes*/
p4
p5
end compilation
*
```

Fig.4 An example of compilation process with DACM

5. CONCLUSIONS

The problem of data security exists not only in data base but also in general data management systems (for example in file systems). Therefore, we should develop data security mechanisms in such systems. The best approach to design that (in my opinion) relies on capabilities built into programming language. The DACM implementation described in this paper provides a nearby satisfactory solution of this problem. However, this work should be completed and generalized in a number of directions. The following two directions are particularly important:

1. The problem how to add the DACM capabilities to MODULA-2 definition.

2. The problem how to expand possibilities of DACM for checking data access not only files but also to elements of files (for example record occurences, field of record).

REFERENCES

[HOFF77] Hofmann, L.: Modern methods for Computer Security and Privacy. New Jersey, USA, 1977.

[SZAF78] Szafranski, B.: The questions of program's data security. Diss., MAT, Poland, 1978.

[SZAF79] Szafranski, B.: A data security model in data base. ICS PAS Reports, Warsaw, Poland, 1979.

[WIRTH80] Wirth, N.: Modula-2. ETM, 1980.

[WIRTH81] Wirht, N.: Overview of the Modula-2 Compiler. M2 RT11, ETM, 1981.

[WIRTH82] Wirth, N.: Programming in Modula-2. Springer-Verlag, Berlin, 1982.

Adat-elérési vezérlési mechanizmusok megvalósitása Modula-2 segitségével

B. Safranski

Összefoglaló

A szerző azokat a tapasztalatokat ismerteti, amelyeket a Modula-2 felhasználásával megvalósitott adat-biztonsági mechanizmusok terén nyert.

Применение языка МОДУЛА-2 в осуществлении механизма контроля доступа к данным

Б. Шафраньски

Р е з ю м е

В статье представляется опыт собранный во время осуществления механизма защиты данных опирающегося на управлении доступа к данным на языке МОДУЛА-2. Этот механизм дает возможность контроля операции доступа к данным во время компиляции и исполнения программы. Механизм построен на основе формальной модели процесса защиты данных.