ACCESS MECHANISMS IN BLOCK STRUCTURED ENVIRONMENTS

MATS LÖFGREN 1) and BORIS MAGNUSSON 2)

1) Headquarters of Swedish Telecommunications Administration, Dept. of Software Technology (UGD), Box 4548, S-203 20 Malmö, Sweden

2) Lund Institute of Technology, Dept. of Computer Science, Sölvegatan 14A, 223 62 Lund, Sweden

Abstract.

Statistics from dynamic analysis of programs are used to compare the efficiency of different methods to access global variables. The method where static links are used for access is shown to be at least as efficient as the display method. The use of the linked list method also leads to a simpler runtime system and allows separately compiled modules to execute at any level in the resulting program. The results of the dynamic analysis are presented in a form which allows comparison of the access methods in language implementations on specific computers. Experience with using two different access methods in an implementation of Simula67 concludes the paper.

Keywords: programming languages, implementation, recursion, block-structure, display, access mechanism, Simula, Ada, separate compilation, dynamic analysis, compiler.

1. Introduction.

When processing programs written in block-structured languages allowing recursive procedures or other means to generate more than one dynamic instance of a given entity, the compiler cannot predict the location in memory of variables during runtime. Examples of languages with such facilities are Algol 60 [14], Simula67 [7], Pascal [17] and Ada [1]. In these languages identifiers declared in the currently active block or in one of its statically enclosing blocks are directly accessible by their names. The problem discussed in this paper is how to access entities denoted by identifiers in the enclosing blocks in an efficient way.

During runtime there may simultaneously exist several activation records representing different activations of the block (sub-block, procedure etc.) with distinct data. The static binding of variables to names defines the environment of an activation record. The environment of an activation record does not vary during its lifespan.
A common method of providing access to variables in the environment is to maintain a table of references to the accessible activation records. This table is usually called the display [11], and the method has been used in many language implementations [8, 18, 16]. Improved methods for maintaining and representing the display have been described in [3, 19]. Another access method is to use a link stored in each activation record referencing the activation record of its closest surrounding block, the static link. The environment of an activation record is resolved by descending through the linked list of activation records.

An earlier evaluation of a Pascal system [19] indicates that the linked list method is preferable for that language. In this paper we will show that this is also true for languages such as Simula where programs normally contain more nested block structures than in Pascal. In Ada programs there will probably be at least as complex structures as in Simula programs, which indicates that the results are valid also for Ada.

The reason for the advantage of the linked list method is that accesses to variables in activation records other than the current one are usually to the outermost block, which can be accessed in a more efficient way. This means that the overhead to maintain the display does not pay off in faster access often enough. Other advantages of the linked list method are the considerably simpler runtime system and the simpler implementation of separate compilation.

In section 2 of this paper we discuss the access methods in more detail and calculate costs for a few basic operations like updating the display or accessing a non-local variable. The calculations are done in machine independent units.

In section 3 we study the runtime behaviour of 20 programs in order to compare the access methods. We do the comparison for ten different situations or versions of the basic access methods. We also present a cost expression which can be used to compare cases other than the ten presented. We conclude the section with some program statistics that help to explain the result but also are of more general interest.

In section 4 we report on the results obtained in an implementation of Simula using two different access methods.

The reader should have a working knowledge of at least one of the programming languages discussed and some knowledge about language implementation techniques.

2. Access methods

The term "block" is used here to denote sub-blocks, procedures and similar constructs. The storage space needed for execution of a block is allocated during runtime, when such a construct is entered. The variable-size allocation units are called activation records. The environment which consists of the directly accessible activation records can be represented in basically three different ways [Fig. 1].