-- --

IEEE P1003.0 Draft 14 – November 1991

4.1 Language Services

Responsibility: Don Folland

4.1.1 Overview and Rationale

While a consistent interface to the operating system is essential for applications portability, the application will have been developed using language and system development tools that, in turn, require support by standards to achieve source code portability.

Those responsible for system or software development will wish to write programs in code supported by an international standard and compile the code using a compiler that has a certificate of conformance issued by an accredited test center. Noncompliant extensions must be avoided if applications portability is to be maintained. Compilers should identify nonstandard-compliant code.

The languages that have been identified in this document are those seen to be in most popular use today for software development. The POSIX.2 shell command language is discussed in 4.10. The standards identified are the most widely recognized today, with significant use in the Information Technology industry on a broad range of processors, or where a large installed base of a particular version is known to exist.

4.1.2 Scope

The services described in this clause cover the most widely used third-generation computer languages in use today for the development of applications; i.e., the languages used to write application programs. Fourth-generation languages are not currently addressed in this guide. In order for a program to address an API to the services described in other clauses of this guide, an appropriate language binding to that interface is required. References to those bindings will be found in the clause describing the relevant service.

4.1.3 Reference Model

This subclause identifies the entities and interfaces supporting language services. The reference model based on the reference model in Figure 3-1 is illustrated in Figure 4-1, but because the language services directly support the binding of the applications to the API, there is no EEI. However, the EEI is shown in Figure 4-1 for consistency.

At the simplistic level, the programmer developing an application that requires only basic operating system services will use a compiler that meets both the fundamental language standard (e.g., ISO 1989: 1985 for COBOL, ISO 1359: 1990 for Fortran) and the binding established for the relevant system calls in POSIX.1 {2}.

As identified in 4.6, an application program may also require database services that will be provided by the Database Manager API. The database vendor will

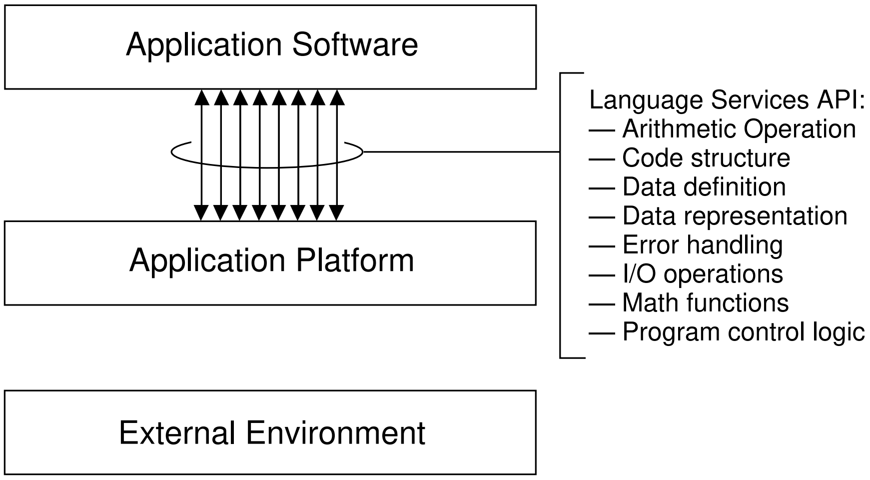

Figure 4-1 – Language Service Reference Model

offer an API to meet the requirements for the popular programming languages.

In a POSIX Open System Environment the intention is that support is provided for all languages identified in 4.1.4.

4.1.4 Service Requirements

Programming language services provide the basic syntax and semantic definition for use by a software developer to describe the desired application software function. While most clauses in this guide provide a comprehensive list of services, in the case of languages many services are a unique function of the language specification. Rather than extend the size of this guide, the detail is more appropriately found in the relevant language manuals and supporting standards.

4.1.4.1 Application Program Services

Programmers require the ability to write and execute a program in the language E
of their choice. The selection of a particular programming language for the development of an application may depend on a variety of factors, including the capability to provide some of the functions listed here:

— Arithmetic operation

— Code structure

— Data definition

— Data representation

— Error handling

— I/O operations

— Mathematical functions

— Program control logic

The programming languages identified in this clause are:

Ada
APL
BASIC
C
C++
COBOL
Common LISP
FORTRAN
Pascal
PL/1
Prolog

As well as making reference to the relevant language standard, where a programmer requires to call other services, e.g., seeks access to graphics kernel system, it will be necessary to refer to the relevant language binding to those services. Language bindings are identified in the Standards subclause, 4.*n*.10, of each service clause in Chapter 4.

4.1.4.1.1 Ada

Ada is a procedural language based on the Pascal programming language. It is capable of processing both numerical and textual data and has the key attributes of:

— Strong data typing

— Data abstraction

— Structured constructs

— Multitasking

— Concurrent processing

Although Ada was developed initially for military purposes, it is considered suitable for a variety of business and industrial applications.

4.1.4.1.2 APL

APL is a language and interactive programming environment oriented around multidimensional arrays of characters and numbers. It uses an extremely compact notation based on powerful primitive functions and function-combining operators. Revisions to the language are in preparation to permit single array elements to contain arrays.

4.1.4.1.3 BASIC

BASIC is an interactive and procedural language with some similarity to FORTRAN. It is readily learned by non-computer-literate individuals. Commonly used for educational purposes, it has also been adopted in a variety of business and commercial applications running on small business systems. BASIC offers:

— Conversational statements

— Free style input

— Segmentation of complex statements

— Six significant digits of accuracy

— Mathematical functions

4.1.4.1.4 C

C is a general purpose procedural language that was developed for the UNIX operating system. It offers the control and data structure of a high-level language and the efficiency of primitive operators that have made it very suitable for system programming.

4.1.4.1.5 C++

C++ has evolved as a superset of C and may be viewed as a procedural language, while at the same time offering the capability for object-oriented programming. The concept of an object-oriented language is to define data objects that include sets of operations to manipulate the data, and so direct these objects to apply the necessary operations which comprise the application.

4.1.4.1.6 COBOL

COBOL is a procedural language designed originally to meet the needs of business. It permits use of natural words and phrases, enabling the language to be adopted by non-technical writers with a basic appreciation of information processing. The language offers file organization features, variable data length, input/output procedures, and report generation.

4.1.4.1.7 Common LISP

LISP is an interactive nonprocedural language. The basic entity is the symbolic expression which is either an atomic symbol or a list structure. A list is a set of items in a specific order. Lists can be variable length and dynamically adjusted; the items can be of different type.

4.1.4.1.8 FORTRAN

Though originally developed for processing scientific problems the language is widely used in commercial and educational applications. It is a procedural language whose grammar, symbols, rules, and syntax are simple mathematical and English-language conventions. Its focus is on numerical computation, using simple concise statements, operating on small amounts of input data and little text.

4.1.4.1.9 Pascal

This is a procedural language that is particularly effective in structured programming and was designed to help programmers in rapid error detection. It is highly efficient, handling both numerical and textual data. It is considered very suitable for small system applications such as typesetting, editorial work, computer aided design (CAD), and manufacturing processes.

4.1.4.1.10 PL/1

This is a procedural language introduced to offer in one language the strengths of both COBOL and FORTRAN; i.e., serving both the business and scientific communities. It has the FORTRAN strength of simple statements, coupled with the ability, as in COBOL, to manipulate data and organize files. It is block structured, facilitating good programming techniques.

4.1.4.1.11 Prolog

This language, like LISP, is nonprocedural and has an emphasis on description rather than on action. It is described as pattern-directed role-based programming using definitions of conditions established within the program to satisfy a query. It is of particular value in applications of artificial intelligence, for constructing expert or knowledge-based systems.

4.1.4.2 External Environment Interface Services

Not applicable.

4.1.4.3 Interapplication Software Entity Services

Not applicable.

4.1.4.4 Language Resource Management Services

Not applicable.

4.1.5 Standards, Specifications, and Gaps

4.1.5.1 Current Standards E

See Table 4-1. E

Table 4-1 – Language Standards

Service	Type	Specification	Subclause	
				E
Ada	S	ISO 8652	4.1.5.1	E
APL	S	ISO 8485	4.1.5.1	E
BASIC	S	ISO 6373	4.1.5.1	E
C	S	ISO/IEC 9899	4.1.5.1	E
C++	E	n/a	4.1.5.2	E
COBOL	S	ISO 1989	4.1.5.1	E
Common LISP	G	n/a	4.1.5.1	E
FORTRAN	S	ISO 1539	4.1.5.3	E
Pascal	S	ISO 7185	4.1.5.1	E
PL/1	S	ISO 6160	4.1.5.1	E
PL/1 (GP Subset)	S	ISO 6522	4.1.5.1	E
PROLOG	G	n/a	4.1.5.3	E

Ada

ISO 8652: 1987 is the current version of the international standard for Ada, which was an endorsement of the ANSI standard 1815A-1983.

APL

ISO 8485 is the current version of the international standard for APL.

BASIC

ISO 6373: 1984 is the current version of the international standard for minimal BASIC.

C

ISO/IEC 9899: 1990 is the current version of the international standard for the C language.

COBOL

ISO 1989: 1985 is the latest version of the international standard for COBOL, which was an endorsement of the ANSI standard X3.23-1985. An Addendum is in process at present entitled ‘‘Intrinsic function module.’’

Fortran

ISO 1539: 1990 is the latest revision of the international standard for Fortran.

Pascal

ISO 7185: 1983 is the current version of the international standard for Pascal, which was an endorsement of the British standard BS 6192-1982.

PL/1

ISO 6160: 1979 is the current version of the international standard for PL/1, which was an endorsement of the ANSI standard X3.53-1976. ISO 6522: 1985 is the current version of the international standard for a General Purpose subset of PL/1, which is an endorsement of ANSI standard X3.74-1981. A revision of this standard is at Draft IS stage.

4.1.5.2 Emerging Standards E

BASIC E

CD 10279 is a proposal for Full BASIC. E

C++ E

ISO/IEC JTC 1/SC22/WG21 has a work item for standardizing C++. This will be E
based on the standard under development in ANSI X3J16. E

Pascal

DIS 10206 is a draft international standard for extended Pascal.

E

4.1.5.3 Gaps in Available Standards

4.1.5.3.1 Standards and Specifications outside the POSIX OSE

None. E

4.1.5.3.2 Unsatisfied Service Requirements

There is a requirement for standardization of the following languages:

- C++
- LISP
- Prolog

4.1.6 OSE Cross-Category Services

Not applicable.

4.1.7 Related Standards

Many of the services within the POSIX OSE require APIs with bindings to languages identified in this clause; e.g., Graphics, Database. Reference to the particular language binding standard is to be found in the relevant service clause.

4.1.8 Open Issues

While there are occasional calls for 4GL standards, there has been little effort applied so far.

4.2 System Services

Responsibility: Patricia Oberndorf

4.2.1 Overview and Rationale

This clause describes the system services component of the application platform. It presents a reference model for this component and describes the services provided to application software. Those services are those usually considered as part of an operating system or executive and also those services that may be provided by system level entities such as spoolers and device drivers. Standards, current and emerging, that specify the interface to those system services are also described.

System services are a key component of the application platform and represent the focus of the IEEE effort to produce POSIX base standards. A common set of system services provides support for the portability and the interoperability of application software. While other common services can aid application reuse, system services are those that are common to the largest number of applications.

4.2.2 Scope

System services cover those features that users have come to expect from operating systems or executives. They cover the areas of process management, file management, input/output, memory management, and print spoolers. Because there is a wide variety of platform users, ranging from large general purpose time-shared systems to small time-critical, special-purpose systems, services such as timers and clocks, event management, logical device drivers, and system initialization/reinitialization are included. Services related to distributed systems are also discussed, since application software sees these capabilities through the platform.

4.2.3 Reference Model

This subclause identifies the entities and interfaces specific to the system services of the POSIX OSE. The reference model presented here is consistent with and expands upon the reference model of Section 3. It provides the context for the discussion of System Services in this clause. The basis System Services model is shown in Figure 4-2.

This clause describes the system services portion of the application platform as viewed by a software developer (not necessarily the viewpoint of the end user). This view corresponds to the program design level of abstraction.

The system services API provides the interface between the application software and the system services from the source code point of view. The API defines the program designer's means of accessing the functions, objects, and services of the system.