IEEE P1003.0 Draft 13 − September 1991

4.9 User Command Interface Services

Responsibility: Wendy Rauch

HLJ : Wendy has provided a full clause replacement for Draft 13; it is not further diff marked.

4.9.1 Rationale and Overview

Although system-level services are necessary for application portability and interoperability, they are insufficient for many users' system needs. To maximize portability, users also require the commands, command interpreter (shell), compilers, editors, and other utilities that have been traditionally associated with many operating systems. These command interface services facilitate a successful port and help users to manage and maintain applications and to solve problems on an ad hoc basis. The standardization of these utilities allows users and programmers to move from platform to platform without having to relearn the command interface for each application platform.

4.9.2 Scope

This clause describes how a user interacts with an application platform by executing general purpose commands. This command interface is also available to applications so that applications also can execute commands. A standardized command interface provides a consistent, interactive environment across platforms for users and programmers.

Commands that are outside the scope of this clause are:

— System administration and installation commands

— Text formatting programs

— Database commands

— Networking and communications commands

— Graphical user interfaces

Networking commands and graphical user interfaces are described in other clauses of the POSIX Open Systems Environment Guide.

4.9.3 Reference Model

The use of the command interface services presented in this clause is consistent with the reference model in Section 3. The POSIX OSE reference model for the command interface also is consistent with typical implementations for user command languages in traditional UNIX-based systems.

As Figure 4-19 shows, the command interface is available both to users (through the External Environment Interface) and to applications (through the Application

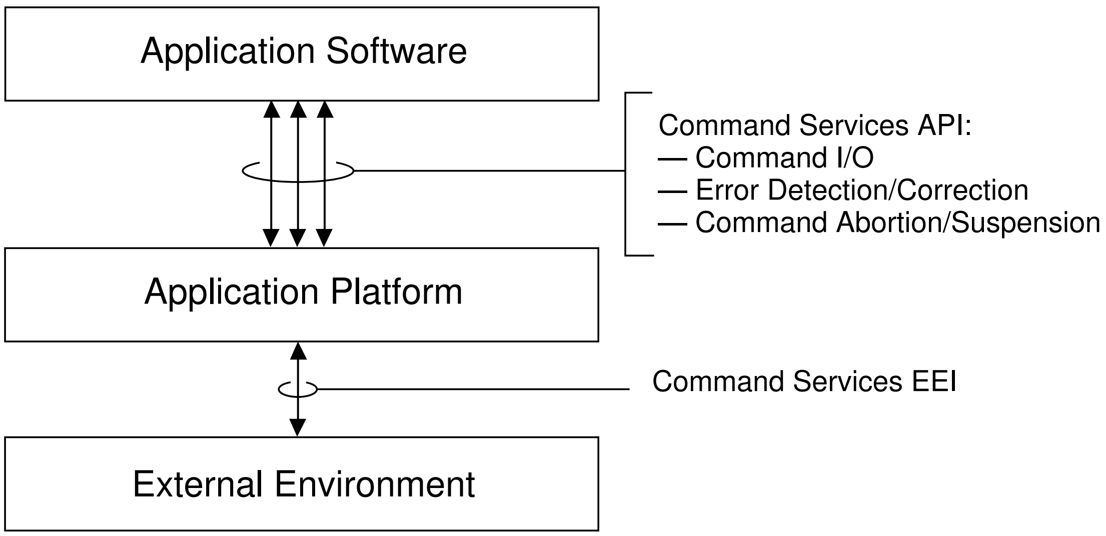

Figure 4-19 – POSIX OSE Reference Model for Command Interfaces

Programming Interface). Any operating system implementation can reside underneath the APIs and EEIs.

The API and EEI command interfaces provide access to a software component (known as a command interpreter or shell) that interprets the commands issued by either the user or the application. The command interpreter acts as an intermediary between the command API and EEI and the base application platform's system-level services. The command interpreter reads the commands entered and parses them. Depending on the type of command (e.g., utility or built-in shell command), the command interpreter either executes the command for the user or application, using the base application platform's system-level services, or it calls on the system-level services to create a new process which executes the command.

None of the methods of executing commands have an impact on the API or EEI specifications.

The commands interfaces may be available to users and applications either locally or remotely. Remote invocation of a system's command interfaces is provided through networking and data interchange capabilities. These are described in 4.3 and 4.5. Alternatively, remote access to a system's command interfaces may be available through certain interapplication services.

4.9.4 Service Requirements

There are three major aspects of command interface services that must be addressed for practical support of multivendor application portability and system interoperability. The first aspect consists of the basic functionality and interfaces provided for generally usefulness. The second aspect of command interface services concerns the ability to move applications, such as script files, between platforms. The third aspect concerns user portability so that the same user interface is available on different platforms.

Since most command interfaces are available at the API and EEI , the service requirements for the API and the EEI are very similar. This clause, therefore, discusses primarily the EEI command interface requirements. The API service subclause discusses only the additional service requirements for applications.

4.9.4.1 External Environment Interface Services

Users need a number of capabilities in order to work on a system. On a traditional system, these are implemented by providing interactive commands entered via a keyboard. However, as graphical user interfaces evolve, these commands may also be implemented by clicking on a mouse in a particular area of the screen, by a ''touchie-feelie'' screen, a tablet, or other input device.

The major services at the EEI provide the following abilities:

- Capture the output of a command or application into a file
- Redirect the input for a command from a file
- Direct the output of a command to be used as the input to another command
- Execute applications
- Get online help for commands or applications
- Manipulate file contents:
  - Cutting
  - Pasting
  - Concatenating
  - Converting
  - Sorting
  - Reformatting
  - Comparing
  - Searching for regular expression
- Edit files

  - Interactive editors
  - Batch or ‘‘stream’’ editors
- Display files
  - Pausing when necessary
  - Display only selected ranges of files
- Manipulate files
  - Create
  - Delete
  - Rename
  - Move
  - Copy
- Print files
- Perform network functions
  - File transfer
  - Remote execution of commands
  - Remote file printing
- Manipulate and display directories
  - Create
  - Delete
  - Display
  - Destroy (Delete a directory and all its subdirectories and files)
- Control file and directory permissions
- Communicate with other users
  - Electronic mail
  - Online interaction where two or more users communicate with each other simultaneously
- Control the application execution environment
  - Execute applications in the background
  - Abort applications running in the foreground or background
  - Suspend an application
  - Move an application running in foreground mode to the background
- Schedule commands for periodic execution

- Control the users' input equipment, such as a terminal or graphical user interface
- Manage local environment and configuration information
- Query local environment and configuration data
- Configure an environment for an international locale.

4.9.4.2 Application Program Interface Services

In a command API, the output syntax of the commands and command responses (such as error messages) need to be standardized, in addition to the calling sequence and allowable inputs. Such standardization is necessary to allow applications executing a command to reliably parse the output of that command.

The API should be able to access all of the services available to the user at the EEI. The additional service requirements for the API are as follows;

- Ability to provide the input to the command and access the output of the command when necessary
- Ability for the application to detect and correct errors as the command is executed
- Ability to abort or suspend the command as it is executing.

It is also important to have the ability to create script files which are combinations of commands. The scripting language developed for this purpose is an application development language. The scripting language has the following requirements:

- Conditional execution primitives
- Repeated execution primitives
- Ability to display output
- Ability to prompt the user for input
- Ability to execute commands and obtain error information.

The services and standards for the scripting language are described in this clause, rather than in the Languages clause 4.1, because it is so closely related to the command interface.

4.9.4.3 Interapplication Entity Services

These services enable remote users and applications to access and execute a system's command interfaces as if they were directly connected to that system. The major categories of interapplication entity services include the following:

- Login and use hosts on a network as if the users logging-in were directly connected to the local terminal

— Remotely execute a system's shell commands as if the user were directly connected to a local terminal

— Copy files between hosts without going through a network file transfer program

— Find out who else is logged into the machines on a local-area network

— Query the status and uptime of all machines on a local-area network.

4.9.5 Standards, Specifications, and Gaps

There are currently no formal standards for command interfaces. There are, however, several command-interface standards-development activities underway. In addition, there are several consortia-defined specifications and de facto specification standards for commands, shell, and utilities services and interfaces.

Table 4-10 summarizes the shell and utilities standards and specifications and work in progress.

Table 4-10 – Shell and Utilities Standards Activities

Service	Specification	Subclause
Shell and Utilities	IEEE POSIX. 2 {*ref*}	4.9.5.?
User Portability Extension (UPE)	IEEE POSIX. 2a {*ref*}	4.9.5.?
Shell, Utilities, and UPE	X/Open XPG 4.9.5.? OSF OSF/1 System V: 1989 SVID Berkeley BSD 4.x UNIX	

4.9.5.1 Current Standards

4.9.5.1.1 International and National Standards

There are no currently completed or approved international or national standards for commands and utilities.

4.9.5.2 Emerging Standards

4.9.5.2.1 International and Other Formal Standards

When completed, the IEEE POSIX. 2 {*ref*} standard will define a source code interface to command interpretation or shell services and common utility programs for application programs. These services and programs are complementary to those specified by POSIX .1 {2}.

The IEEE POSIX.2a {*ref*} User Portability Extension will supplement POSIX.2 {*ref*} by extending the specifications to promote the portability of users and programmers, in addition to applications, across conforming systems. Toward this end, the POSIX.2a {*ref*} specifications expand the number and type of utilities specified, and enhance the features of a number of POSIX.2-specified utilities, to provide a consistent interactive environment. The consistent interactive environment does not include emerging technologies such as graphical user interfaces, which are under development by different standards groups.

Parts of POSIX.2 {*ref*} go beyond the current service requirements and include a number of software development and debugging commands and utilities services. These are included in the POSIX.2 specification because of the traditional development orientation of UNIX systems. These software development and debugging services are not included in this clause because this clause includes more general and universal services, such as copying a file and reading a directory.

Although the POSIX.2 {*ref*} and POSIX.2a {*ref*} specifications are still in draft stages, they are relatively complete, and portions of the emerging standard are believed to be mature and stable.

When the commands, shell, and utilities specifications are completed and approved, the resulting IEEE POSIX.2 {*ref*} and POSIX.2a {*ref*} standards will be submitted to ISO/IEC JTC 1 for adoption as international standards. At that time, POSIX.2 and POSIX.2a will be combined into a single integrated international standard (ISO/IEC 9945-2).

4.9.5.3 Gaps in Available Standards

There are no formal interapplication standards that address the remote access and execution of a system's command interfaces. The Berkeley BSD UNIX de facto standard addresses all these service requirements, however.

4.9.5.3.1 Public Specifications

Public specifications that include the POSIX.2 {*ref*} and POSIX.2a {*ref*}, and go beyond these standards to also include the traditional UNIX-based command interfaces for software development, system administration, the UUCP (UNIX-to-UNIX Communications Protocol) utilities, and parts of the POSIX.4 realtime extensions are available from a number of organizations. These include:

— X/Open's XPG3 specifications, Volume 1 and part of Volume 3

— OSF's OSF/1 Application Environment Specifications (AES), software documentation, and software

— University of California at Berkeley BSD UNIX

— AT&T System V Interface Definition (SVID) and System V UNIX.

4.9.6 POSIX OSE Cross-Category Services

4.9.6.1 Internationalization

The ANSI C, IEEE POSIX.1, and IEEE POSIX.2 specifications contain limited capabilities for obtaining and defining locale-specific information. In addition, some of the utilities described in the POSIX.2 specifications contain requirements for standardized multilingual and multicultural support (e.g., localization requirements such as date formats and collation sequences, and support for international character sets).

4.9.7 Related Standards

Several standards activities are in progress to address additional commands interfaces that are traditionally included in UNIX and UNIX-like systems. The following standards are examples:

- IEEE 1003.4 Realtime Extensions for POSIX
- IEEE 1003.7 POSIX System Administration Extensions
- IEEE 1003.18 Platform Environment Profile (PEP) standard, which identifies and combines the standards dealing with POSIX.1, POSIX.2, POSIX.2a, and the related standards

4.9.8 Open Issues

Conformance Testing

Command interface standards must address conformance testing. Although the POSIX.2 standard is still in a draft stage, work has already begun in the IEEE 1003.3.2 Conformance Test Methods Group to define test assertions for the POSIX.2 standard.

UUCP

The UUCP (UNIX-to-UNIX Copy Protocol) services and commands, for electronic mail and file copying, which are traditionally included in UNIX and UNIX-like systems are not addressed by any standards effort. Among other reasons, UUCP is not currently being addressed because of the inability of the POSIX groups to decide whether the UUCP services and commands should be standardized in the POSIX.2 Group (since UUCP is a traditional UNIX service with traditional command interfaces) or in the networking groups (since UUCP is an electronic mail and file copying facility that works on networks).

4.10 Transaction Processing Services

Responsibility: Bob Gambrel

Based on input created by Carl Hall of P1003.11, with contributions from Jeremy Denny, John Penny, Patrick Prange, and Jeff DeMent Revised in Draft 13 with input from Dorm Fulton of P1003.11.

4.10.1 Overview and Rationale

The database management clause (see 4.4) described some transaction processing service requirements (specific to databases). This clause describes the complete set of transaction processing services from the application software point of view. Note that transaction processing services have long been been regarded, variously, to be within the domain of databases or within the domain of operating systems, and more recently within the domain of interconnect. These services are more broadly applicable and so are treated here as a separate clause.

Transactions (''units of work'') have boundaries (start points and end points) that are determined by the action of the transaction application program. The transaction application program can request to either commit or rollback the work done in the transaction when it identifies the end point. The system will complete a commit operation only if all operations performed during the transaction can complete successfully. Otherwise the system will abort the transaction (rollback the work done by it) and notify the transaction application program of this action.

The following is quoted with a few editorial changes from ISO/IEC DP 10026-1 {*ref*}, the ISO Distributed Transaction Processing standard draft:

> A transaction is characterized by four properties: atomicity, consistency, isolation, and durability. These are the *ACID* properties.
>
> Atomicity implies that the operations of a unit of work are either all performed, or none of them are performed.
>
> Consistency implies that the operations of a unit of work, if performed at all, are performed accurately, correctly, and with validity, with respect to application semantics.
>
> Isolation implies that the partial results of a unit of work are not accessible, except by operations which are part of the unit of work. Isolation also implies that units of work which share bound data are serializable.
>
> Durability implies that all the effects of a completed unit of work are not altered by any sort of failure.