NCE & TECHNOLOGY:

THE DESIGN AND IMPLEMENTATION OF THE NATIONAL BUREAU OF STANDARDS' NETWORK ACCESS MACHINE (NAM)

NBS Special Publication 500-35
U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

-35

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government Agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] — Radiation Research — Thermodynamics and Molecular Science — Analytical Chemistry — Materials Science.

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to users in the public and private sectors to address national needs and to solve national problems in the public interest; conducts research in engineering and applied science in support of objectives in these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the utlimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] — Mechanical Engineering and Process Technology[2] — Building Technology — Fire Research — Consumer Product Technology — Field Methods.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal Agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal Agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following divisions:

Systems and Software — Computer Systems Engineering — Information Technology.

[1]Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington,D.C. 20234.
[2]Some divisions within the center are located at Boulder, Colorado, 80303.

The National Bureau of Standards was reorganized, effective April 9, 1978.

COMPUTER SCIENCE & TECHNOLOGY:

The Design and Implementation of the National Bureau of Standards' Network Access Machine (NAM)

Robert Rosenthal and Bruce D. Lucas

Information Technology Division
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

U.S. DEPARTMENT OF COMMERCE, Juanita M. Kreps, Secretary

Dr. Sidney Harman, Under Secretary

Jordan J. Baruch, Assistant Secretary for Science and Technology

NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued June 1978

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-35
Nat. Bur. Stand. (U.S.) Spec. Publ. 500-35, 50 pages (June 1978)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 78-600055

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1978

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
Stock No. 003-003-01949-6 Price 2.20

(Add 25 percent additional for other than U.S. mailing).

TABLE OF CONTENTS

DISCLAIMER

Certain commercial products are identified in this special publication in order to adequately specify the Network Access Machine. In no case does such identification imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the products identified are the best available for the purposes described.

The Design and Implementation of the National Bureau of Standards' Network Access Machine (NAM)

Robert Rosenthal
Bruce D. Lucas

ABSTRACT

The Network Access Machine (NAM), a programmed minicomputer designed to assist interactive on-line terminal users of computer network services and resources, is discussed in detail. The minicomputer allows the user to specify (or to have specified) network command sequences for execution on a specified network and host connected to that network. Computer responses are analyzed to assure agreement with those anticipated for specific commands. Experiences with the NAM and specific examples of NAM use including a common command language for bibliographic retrieval are presented.

Key words: Computer networks; Minicomputers; Intelligent terminals; Computer access; Communications; Protocols; Command languages.

PREFACE

This Special Publication is one of a series prepared as part of a jointly sponsored effort by the National Bureau of Standards and the U. S Air Force Rome Air Development Center under contract number F 30602-77-F-0068.

1. INTRODUCTION

This report discusses the design philosophy of the National Bureau of Standards' Network Access Machine (NAM) and augments a previous tutorial report that surveyed current and planned work in the area of network access [ROS 76]. That report identified several projects related to network access including work at the Rand Corporation on the Rand Intelligent Terminal Agent (RITA) [AND 77], and the Massachusetts Institute of Technology's work on the networking of interactive bibliographic retrieval systems [MAR 76].

This report updates previous descriptions of the NBS NAM from that tutorial and details the specific software that provides access assistance functions to a wide variety of users including sophisticated programmers, casual timesharing users, scientists and engineers, bibliographic search and retrieval specialists, and others who need and use computers in the normal performance of their jobs.

This report also discusses the implementation of the NAM software and points out how the NAM is used in specific applications including a common command language on heterogeneous host computers for bibliographic retrieval.

1.1 A NAM OVERVIEW

The NAM is a minicomputer system that acts as a network access point for a user at his terminal and assists the user through the automatic execution of access procedures. This minicomputer facility allows the user to specify (or have specified) his own network command sequences for execution on a specified network and host connected to that network. Computer responses are analyzed to assure agreement with those anticipated for specific commands.

The facility is a PDP-11/45 minicomputer with a full complement (128k) of core; a disk provides secondary storage for files. Several different types of communications equipment are used for access to a variety of computer networks. This equipment includes a specially designed automatic calling unit that provides access to the switched telephone network, an interface to the ARPA Network that provides access to ARPA Network resources, and a specially designed interconnection patch panel that provides access to other NBS local computer facilities.

The NAM software is written as an applications program for the UNIX operating system [RIT 74]. The program relies on UNIX for communications support: That is, the automatic calling unit software, the ARPA Network Control Program (NCP), and local computer connection software are integral parts of UNIX -- they are UNIX drivers. These drivers provide the NAM program with the lower level communications support needed to establish and maintain connections.

1.2 NAM DESIGN PHILOSOPHY

The NAM software has been designed with the following three points in mind:

Service providers (vendors) have innumerable product offerings; there are many different services available that meet a wide range of user needs.

Equipment suppliers have enormous selections of product offerings; there are many different terminal types and communication facilities.

Users or customers have diverse needs and demands for the services and equipment offered by vendors; more and more, these customers require the services and equipment provided by more than one supplier.

It should be possible for a large community of users with different terminals and communication facilities to access a wide variety of heterogeneous host computers and services. How does one design a NAM to meet such diverse requirements -- requirements that match user demands with user terminals, access methods and services? One way is to adopt a philosophy where:

A) Lower level interface connection requirements are met directly. UNIX provides these requirements for the switched telephone network, the ARPA Network, and the NBS local patch panel facility.

1) Physical connections to requested services are established and maintained. These physical connections include host interfaces to computer networks, hardwired connections to local computer facilities or, when applicable, dial-

out connections to other computer service providers.

2) Link control protocols are satisfied so that intelligible communication is possible. Suitable link controls enable the NAM to request that connections be established and broken on behalf of the user.

3) Communication control protocols are satisfied so that the transfer of intelligible data in the form of messages, packets, characters, or bits is possible.

B) Higher level user protocol requirements are met directly by the the NAM using the NAM programmed access capabilities. That is, detailed user/host interactions such as "login", "service selection", "logout" and other high-level interactions are generated and tailored for specific computer service providers using a procedural language that is part of the NAM facility.

This philosophy, to provide a procedural language that expresses user (higher) level interactions, enables the NAM to perform complex interactions with diverse user communities, terminal types, and host computers.

How can such a system be useful? It can be useful in three important ways. First, it provides an experienced user with capabilities to tailor procedures that perform complex machine dependent dialog (interactions) for particular user needs. Second, it provides less experienced groups or user communities with predefined procedures applicable across diverse services, terminal equipments, and communication media. Third, it provides a common user level interface so that user inputs have a common syntax and (hopefully) semantics across service boundaries, and so that user outputs are formatted in a consistent way even when messages originate from different communications facilities or service providers.

1.3 IMPLEMENTATION RATIONALE

Implementation of a device that meets these philosophical requirements is surely an aggressive undertaking. But, as users who demand more and more services from different suppliers realize, the complex and cumbersome dialog required to perform the same or similar functional operations on different services is overwhelming.

Why for instance should simple (user) functions like "login", "logout", or "service selection" be so different across different services? Why should the more complex user functions of programming, editing, compiling, and debugging be so different across different host computers? Why should applications packages that perform similar functions be so different? One obvious reason is competition among the service providers.

Encouraging competition among network service providers can be in the user's best interest if it leads to innovation in the amount and quality of service received and in the reduction of costs in providing the service. But, when such differences make access more complex and cumbersome, steps to alleviate those differences can be taken.

The NBS NAM attacks this problem in three important ways. First, the NAM provides user uniformity for the simpler functions like "login", "logout", and "service selection". Apparent user uniformity can be accomplished earlier with a network access machine than through formal standards procedures. Second, the NAM provides a testbed facility for quickly implementing and testing proposed standards at the user level of interaction. This lessens the burden of retrofitting existing systems with new protocols and provides a vehicle that allows user communities to more easily adapt to standards when they become effective. Third, early use of tested and proven user-level protocols increases user productivity.

Other important benefits accrue to users of a NAM. Experience with this machine indicates that communication line utilization increases. This may indirectly increase user productivity at the terminal. Other more tangible benefits accrue through the use of common commands: Users don't waste time searching in manuals for specific commands that perform similar functions on different services.

Another potential benefit still to be explored permits users to shop around for the best service for a particular job. The NAM is capable of executing benchmarks on candidate hosts, calculating and tabulating expected response times, and presenting lists of best alternative service providers. The NAM can even migrate user files from one host to another should the user select an alternative to the host where his files reside. Also to be explored is the possibility of supporting certain help assistance functions for the user. Automatic spelling correction is one example.

2. NBS NAM IN THE LOOP

The NBS NAM is the access point to diverse and varied computer services for users at interactive terminals or displays. The NAM acts as a surrogate for the user to establish and maintain physical communications with remote computers; and, when appropriate, the NAM maintains connections by implementing specific communication protocols for a host system or network. These low level communication functions are routinely performed by the UNIX operating system. So, a user can request that the NAM establish and maintain communications with a wide variety of different services.

Having established a communications link with a remote service, the NAM mediates user input and produces service (host computer) dependent interactions to accomplish the user's request. In this way, simple user input requests to the NAM may result in multiple NAM/host interactions. Also, host responses received by the NAM are appropriately formatted (reformatted) and displayed to the user in a consistent and meaningful way. Figure 1 below typifies this exchange of requests.

As an example, consider a user request to log into a particular host on a given network. One user request is sufficient. The nominal response to that request is a response indicating a successful login. To accomplish a successful login, the NAM participates in multiple interactions that request a host connection, provide a user identification and password, and select a service or data base.

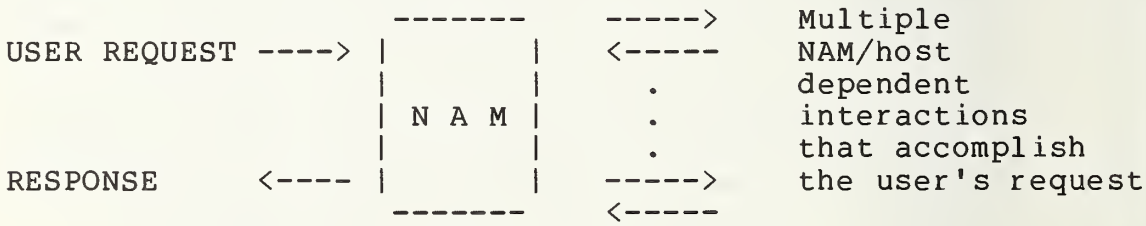

Figure 1: Typical Exchange of Interactions

The service-dependent interactions between the NAM and the host computer are seldom of concern to the user. The user is (usually) only concerned with the syntax and semantics guiding formulation of the user request and the format and meaning of any response to that request. It thus

becomes the responsibility of the NAM to ensure that the user sees a consistent interface to the service selected regardless of the user terminal, communications media, or service vendor.

The NAM program translates a user input request to specific service interactions by interpreting a previously prepared file of NAM directives. These prepared files are referred to as macro files or simply macros. The NAM interprets the directives by expanding the macro file.

Each request entered by the user has a macro file associated with it. The contents of the macro file are directives to the NAM that specify precisely what the remote service interactions ought to be. A rich set of directives that specify what to send, where to send, and when to send is available. Directives that format (reformat) messages both to the host and to the user are available as well as a generalized set of looping and control directives. These looping and control directives provide a convenient way to conditionally expand macro files.

Also, a rich set of utility directives provide for establishing and maintaining connections to services, producing connection status reports, and interfacing to the services provided by the UNIX operating system itself. These services include editors, line printer spoolers, and other utilities.

Users of NAM macros are seldom concerned with the details of the macro directives. Few librarians for instance, would care about the detailed implementation of a common command language for bibliographic retrieval using NAM directives; the librarian is most interested in performing the job of finding and displaying citations in a data base. But, an individual interested in extending such a language or developing new common command languages for other applications would be very interested in the directive details.

So the NAM has two audiences -- a user of NAM commands, and an author of NAM macro commands. The author is interested in how to formulate sequences of service requests for a specific user community, and the specific users of a given community are interested in the commands built by the author. The following sections detail the NAM from the macro author's point of view. Then, after the directives have been identified and explained, several examples giving the user's point of view are presented. These examples are from the common commands for bibliographic search and retrieval.

3. THE NAM PROGRAM

The macro author views the NAM as three cooperating programs: the Command Interpreter, the Macro Expander, and the Response Analyzer. These three programs interface the NAM user to a remote host while allowing the NAM to mediate user/host interactions using data available in macro files. Figure 2 illustrates the relationship that these programs have to the user and to the host.

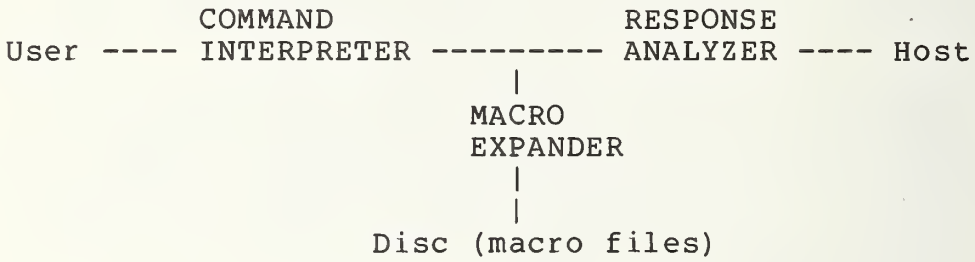

Figure 2 Three Major NAM Components: Command Interpreter, Macro Expander, and Response Analyzer

3.1 THE COMMAND INTERPRETER

The Command Interpreter accepts user input and helps formulate the user's intended command. Once formulated, the command is sent to either the Macro Expander when NAM service is required, or to UNIX when the user desires local editing or other UNIX support. For convenience, a special mode of operation allows user input to be sent directly through the Command Interpreter to host system connections. (Incidentally, more than one host system connection is possible.)

User characters enter the Interpreter and are formed into words. As the words are formed, they are searched for in a keyword table. When appropriate, partially entered keywords, identified by their unique beginning characters, are automatically completed -- their endings are automatically formed. This completion is observed by the user when the terminal displays the ending of the word being searched. The user forces completion by typing the ASCII "ESCAPE" character.

Completed user input commands are identified by a terminating carriage return or line feed character on the last word entered. The completed command is then sent off to the Macro Expander or to UNIX. For the special case in which the user directly interacts with the host system, each character (one at a time) is stored and then forwarded through the Interpreter to the host system. These three paths from the Command Interpreter are illustrated in Figure 3 below.

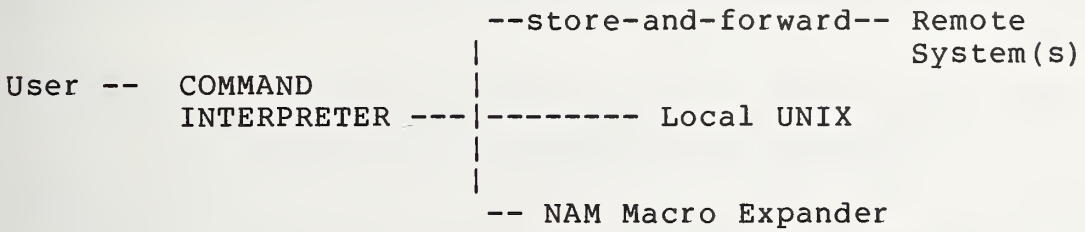

Figure 3: Three Paths Through the Command Interpreter

Once the command is executed either by the Expander or by UNIX, control reverts back to the Interpreter; the Interpreter awaits further user input after prompting the user with a colon ":" character. On the other hand, an explicit return to the interpreter is required to leave (the special case) store-and-forward operation to the remote system. The return is accomplished by typing the ASCII control character "US" -- Unit Separator (sometimes represented as control shift back-arrow).

Several special characters assist the user in the preparation of input lines to the Command Interpreter. The ASCII character back-space "BS" is used to delete a typing error; negative acknowledge "NAK" deletes the entire user line. Also, the "up-arrow" character prefixing an input command, directs the entire line to UNIX for execution.

Command execution by UNIX is beyond the scope of this report but is thoroughly discussed in [RIT 74]. Essentially all of the UNIX commands are directly available to NAM users except "login", "logout", "chdir" (change directory) and several others directly executed by the UNIX Shell program. Command execution by the Macro Expander is important here and is explained below.

3.2 THE MACRO EXPANDER

User input commands passed to the Macro Expander take on one of two forms: an explicit NAM directive, or the name of a macro file containing NAM directives. Explicit NAM directives are statements in the macro language of the NAM. (A complete list of directives is in section 4.5 below.) These directives are interpreted, and executed immediately by the Expander. When execution of the directive is completed, control returns to the Command Interpreter. Thus, all of the NAM directives are directly executable by the user.

The second form -- macro names -- is more interesting. The first word in the user input command names the macro file to be executed. This named file is read by the Expander so that each line in the file can be expanded, evaluated and then interpreted. The remaining words in the input command become parameters to the expansion. The evaluation of the macro file requires that the parameters to the expansion be substituted before interpretation (execution) of the macro line occurs. Finally, the expanded line is executed. Chapter 4 below addresses the importance of macro files in the NAM and completely describes their use with the Macro Expander.

This process of reading lines from the macro file, evaluating the read line, and then interpreting (executing) the result of that evaluation continues until an "end-of-file" condition on the macro file is reached; then control returns back to the Command Interpreter and the user.

3.3 THE RESPONSE ANALYZER

The Response Analyzer buffers characters received from the remote system. As the buffer is being filled, each character received is compared against a predefined anticipated response. Matching the anticipated response with the system response causes the buffer to be marked full and named "response" for use as a string variable by macro directives.

Three directives interface the Expander to the Analyzer: Terminate, Match, and Send. The Terminate directive specifies a list of anticipated responses that cause the buffer to be marked full. This directive optionally specifies a time-out used to force the buffer full if no characters arrive in the specified time. The Response Analyzer uses data from the Terminate directive for determining when the system response is complete. The Match directive plays a similar role; it specifies a list of

anticipated responses that identify substrings within the completed system response. The Response Analyzer returns condition codes to the Expander identifying the substrings for both the matching and terminating conditions. The Send directive causes the specified string to be sent to the remote system initiating these terminate and match processes.

With that overview of the NAM program in mind, attention now focuses on the details of macro files and their relationship to the Macro Expander.

4. MACRO FILES AND THE EXPANDER PROPER

The implementation details of the Macro Expander provide insight into how the NAM carries out its role in the communications link between a user and remote system or service provider. As mentioned earlier, the Expander reads, evaluates, and executes directives found in macro files. This design utilizes an interpretive approach that has proven useful in accommodating the many extensions that inevitably accompany experimental prototype machines (programs) like the NAM.

4.1 EXPANSION PROCEDURES

The Expander parses the command passed to it. Special characters within the command cause syntactically important items to be flagged and acted upon. When the parsing operation is complete, an ordered table of phrases called tokens is available for use and evaluation.

This token table contains substituted formal macro parameters for place holders in the parsed command, substituted character values for the ASCII abbreviations of non-printing and control characters, and verbatim copies of quoted and escaped characters. The example below (presented after a description of the special characters) illustrates how the token table is built and used.

The special characters used by the parser to identify the place holders, character values, and quoted and escaped sequences are defined in table 1 below. Notice that place holders are identified by a dollar sign. Also, notice that tokens are identified by the presence of a separator. These special characters lose their significance if escaped or quoted.

CHARACTER	MEANING
\	Causes the next character to be escaped so that no special meaning is given to it.
"	Causes the QUOTE flag to be toggled: If toggled on, then commas, tabs, spaces and dollar sign have no meaning.
$	If a number follows the dollar sign: It is used as an index into the token table built by the previous call. The string in the table is substituted for the dollar sign followed by a number. If a string name follows the dollar sign and if that name is followed by a period: The contents of that string are substituted for the dollar sign followed by a string name followed by a period.
[...]	Causes the following ASCII control character abbreviation to be substituted for its coded (octal) equivalent. e.g. [SOH] is replaced by the octal value 001.
space tab comma	Causes the previous text to be entered into the token table for this call. Leading separators are ignored.

TABLE 1: Special Characters Used by the Parser

Having built the token table, the Expander examines the first character of the first token in the table. This character is either:

A semicolon ";" indicating that the entire input command is a comment and should be ignored,

a period "." indicating that the command is a NAM directive and must be further evaluated,

an asterisk "*" indicating that the command is a NAM conditional directive and must be further evaluated, or

a different character (none of the above) indicating that the entire input command is the name of another macro to be evaluated much like a subroutine call.

As evidenced from the last item above, (a different character), the command line can even originate from within an expanding macro as well as from the Command Interpreter. For this reason, the Expander was designed as a recursive program. Each recursive call to the Macro Expander requires three arguments: the command line for the new call, the token table from the previous call and a pointer to the text of the call in the macro file.

The command line for the new call is needed so that the parser can again evaluate the input line. This evaluation may result in still another call to the Macro Expander, ..., and on and on recursively until a directive or comment is parsed. The token table from the previous call is required so that parameter substitution based on the index of the ordered token table can be performed during the parser's evaluation of the new command line. Finally, a pointer to the text of the call in the macro file is required so that the loop beginning of the While conditional directive can be remembered (While is defined in section 4.6.7 below). The recursion depth is limited by the number of open files allowed by the UNIX system since each macro is a file that must remain open during its entire expansion. The current UNIX system provides for up to 15 open files.

To clarify the operation of the Expander the following example is offered. In the example, a macro file named OPEN contains comment lines, directives, conditional directives, and the names of other macros to execute. The example assumes that three string variables -- id, password, and account -- had previously been declared and initialized. The example uses the dollar sign to place-hold formal parameters either passed to the macro or passed within the named string variables.

```
;this macro establishes a connection
.connect $1
*if ( connection_made )
       LOGIN $id. $password. $account.
*end
```

The user types the command

```
OPEN bbn-tenexb
```

to the NAM Command Interpreter; here is what happens. The Interpreter recognizes that OPEN is a macro name (as opposed to a UNIX command) and passes the command to the Expander. The Expander begins parsing the command and finds that the

first token does not begin with a period, asterisk, or semicolon: The command is the name of a macro to be evaluated. The token OPEN becomes the zeroth element of the token table array, and bbn-tenex becomes the first element.

The recursive expansion has begun. The first line of OPEN is read and evaluated. A semicolon is parsed and the comment line is ignored. The next line of OPEN is read and a period is parsed. The $1 argument gets replaced by the string found in the token table indexed by 1. This string is "bbn-tenex" so the line is evaluated as

```
.connect bbn-tenexb
```

The NAM executes this directive and attempts to establish a connection to the ARPA Network host bbn-tenexb. The conditional directive on the next line checks that the connection attempt was successful and if it was, LOGIN is parsed and found to be not a directive or a comment, but another macro name. So, another recursive call to the macro expander is made and this time, the token table contains the values of the string variables id, password, and account. LOGIN eventually returns allowing OPEN to continue. In this example, OPEN is finished and control returns back to the Command Interpreter and the user.

One interesting aside to this particular example is that if the user command line begins with a period, the command itself gets executed as though it had come from a macro file. In fact all of the NAM directives are executable directly from the user's terminal via the Command Interpreter.

4.2 EXPRESSIONS IN NAM DIRECTIVES

Expressions in many of the NAM directives allow run-time calculations to be made from executing macros. These expressions are important in conditional expansions for controlling the flow or sequence of directives and for evaluating and setting variables within macros. Expressions consist of mixed mode integer and string operations defined by a rich set of unary and binary operators. Boolean expressions evaluate True or False; zero is False in NAM conditional directives.

Table 2 identifies all of the NAM expression operators including the string and pattern operators discussed below. The more common arithmetic and logical operators are not discussed in detail.

	Operator Symbol	Meaning or "read as"
Arithmetic		
	+	plus
	-	minus
	*	multiply
	/	integer divide
	%	remainder (from integer divide)
Logical		
	~	not
	&	and
	|	or
	>	greater than
	<	less than
	=	equal
	#	not equal
string		
	@	substring extraction
	?	string matching
	.	concatenation
pattern		
	/ /	from the set of characters
	{}	extract
	.	any character
	*	any number of times (including 0)
	' '	the specific string
	|	or
	$	null at end of every string

TABLE 2: NAM Expression Operators

4.3 VARIABLES IN NAM EXPRESSIONS

Two variable types are used in expressions: integer variables and string variables. Integer variables are sixteen bit signed values. String variables are allocated dynamically and are formed as variable length character arrays. Mixed variable types within expressions are perfectly valid; string-to-integer and integer-to-string conversion is done where necessary.

The result after evaluating a mixed expression is typed by the context in which it is used. For example, printing a string formed by summing two integers causes the sum to be evaluated as a string. Summing two strings of integers and setting the result into an integer causes the result to be an integer. Type string is forced when the string operators concatenation, string matching, and string extraction are used.

Variables used by NAM macros are not allocated automatically. There are specific NAM directives that declare variable names as either string or integer; and, these declarations are valid only for the particular invocation of the expansion and for subroutines local to the current invocation. Further, variables only have local (macro) significance. This implies that the NAM Macro Expander provides a block-structured language.

However, global significance is given to variable names defined explicitly by the user from his keyboard through the Command Interpreter. That is, when the user explicitly executes the NAM directives that declare variable names from the Command Interpreter, those names have global significance. Name conflicts with local declarations are resolved by giving the local name precedence over the global name.

The NAM has predefined variable names that already have global significance. These predefined names and their types -- integer or string -- are identified in table 3 below.

TYPE	NAME	COMMENT
Integer		
	termed	Indicates which termination
	matched	Indicates which match
	connection_made	Indicates connection is made
	error	Indicates expansion error
	flag0	General purpose integer
	flag1	General purpose integer
	nparams	Number of parameters passed
String		
	response	Last remote system response
	userline	Command from the Interpreter
	scratch	General purpose string

TABLE 3: Predefined Variable Names

Any of the NAM directives that require or use expressions may utilize these predefined variable names in NAM expressions.

4.4 NAM EXPRESSION SYNTAX

NAM expression syntax is checked during the evaluation of NAM expressions. The syntax is relatively straightforward; a terse description will suffice.

Unary operators are:	plus, minus, and not.
Binary operators are:	plus, minus, multiply, integer divide, integer remainder, and, or, greater than, less than, equal, not equal, substring extraction, string matching, and concatenation.
A digit is:	0, 1, 2, 3, 4, 5, 6, 7, 8, or 9.
A constant is:	a digit or any number of digits. Note that constants do not include the unary operator minus.
A separator is:	one or more spaces, tabs, or commas.
An alpha is:	any ASCII character other than a separator
A nondigit is:	any alpha other than a digit.
A variable is:	any nondigit followed by none, one, or more alpha characters
A term is: (defined recursively)	a constant or a variable or an open parenthesis followed by an expression followed by a closed parenthesis, or a unary operator followed by a separator followed by another term.
An expression is: (defined recursively)	a term or an expression followed by a separator followed by a binary operator followed by a separator followed by another term.

The value of an expression is the value of the expression on the left side of a binary operator evaluated with the term on the right side of the binary operator. That is, the evaluation is strictly left to right.

The syntax for string expressions requires more deliberation. Three binary operators manipulate user declared and NAM predefined string variables. The substring extraction operator "@" and the string matching operator "?" are similar operations, but produce different results; they perform pattern matching operations similar to these of the

SNOBOL programming language [GRI 71].

In each case, two string operands (which can be variables or constants) are specified. The first is the subject of the string matching operation and the second is the pattern that the string is to match. The "?" operator just determines whether the string matches the pattern. Thus the expression

```
STRING ? PATTERN
```

has the value 0 (False) if STRING does not match PATTERN and has the value 1 (True) if it does match. The "@" operator performs a similar pattern matching operation, but yields a string whose characters are taken from the STRING as specified by the PATTERN. STRING and PATTERN as used here refer to the contents of a string variable or a string constant as evaluated by the NAM Expander. String constants appear between double quotes in NAM expressions.

A pattern consists of elements that are to match characters in the string being tested, combined into lists of consecutive elements. These elements are defined by the following symbols:

.	matches any single character,
/x-y/	matches any character in the range x to y where x and y are single characters,
' '	matches the sequence of characters between single quotes, and
$	matches the null character at the end of the test string.

If several elements are written consecutively in a pattern, then the string matches the pattern if it consists of a concatenation of strings that match each element of the pattern, in the order in which the elements of the pattern occur. For example, the value of the expression

```
"C" ? "/A-Z/"
```

is 1 (True) because the string "C" consists of a character in the range /A-Z/; similarly, the pattern matching operation

```
"C2" ? "/A-Z//0-9/"
```

succeeds because the string "C2" consists of a character in the range /A-Z/ followed by a character in the range /0-9/. (Notice that because these are string constants, they appear between double quotes). Also,

```
"C2" ? "./0-9/"
```

succeeds because the string "C2" consists of any character (namely C) followed by a character (namely 2) in the range /0-9/.

Note that all searches are anchored. That is, the portion of the string matched must be the initial substring. For example,

```
"abcd" ? "'bc'"
```

fails (has the value 0) because, even though the pattern 'bc' is found within the string 'abcd', it is not the initial substring. However,

```
"abcd" ? "'ab'"
```

succeeds, because the pattern 'ab' is an initial substring of 'abcd'. Note also that the pattern need not match the entire string, only an initial substring, in order to succeed. To require the pattern to match the entire string, a "$" is required at the end of the pattern. Thus

```
"ab" ? "'ab'$"
```

has the value 1, but

```
"abcd" ? "'ab'$"
```

has the value 0, because the string 'abcd' does not consist of the string 'ab' followed by the end of the string, which is what the $ in the pattern specifies.

Alternative elements are indicated by the or "|" operator. The pattern

```
"('a'|'b')"
```

matches either an 'a' or a 'b'; thus it is equivalent to /a-b/. Here is another example:

```
"('b'|'br')('e'|'ea')"
```

matches 'be', 'bea', 'bre', or 'brea'.

Note that "(A|B)C", where A, B, and C are sub-patterns, means the same thing as "AC|BC".

The '*' operator, placed after the item to which it applies, means "match 0 or more consecutive occurrences of the preceeding item". The previous item may be a parenthesized pattern. For example,

```
"('a'|'b')*$"
```

matches strings that consist entirely of the letter a and b, including the null string. To exclude the null string, the pattern "('a'|'b')('a'|'b')*" could be used; that is, one occurrence of 'a' or 'b' followed by 0 or more occurrences of 'a' or 'b'.

One useful application of this operator is to make the search unanchored. For example, the pattern ".*'abc'" matches any string with the sequence 'abc' anywhere in it, not just at the beginning.

The "@" operator performs the same operations as the ? operator, but instead of returning True or False, it returns a string of characters extracted from the subject STRING, according to the PATTERN. Specifically, the characters matched by those portions of the pattern between "{" and "}" are extracted. Thus

```
"AbCdEf" @ "({/A-Z/}|.)*$"
```

extracts all the capital letters from the subject STRING producing the string ACE.

Now that the expansion procedures, expressions, variables, and syntax have been covered, attention focuses on the NAM directives.

4.5 NAM DIRECTIVES

Once identified as a NAM directive by the Expander, a NAM routine responsible for interpretively executing that directive is called. Each directive has its own routine as described below. The list below is ordered aphabetically; no significance or transition between directives is implied by the ordering. This provides a handy reference for the NAM macro author. An example use of each directive is given.

4.5.1 Break. Break passes macro execution control outside the scope of the current conditional "While" or "Switch" directives.

```
.break
```

4.5.2 Connect. Connect informs the NAM to make a physical connection to a remote host. Parameters passed to Connect specify the physical and logical connection name, and a possible communication medium for the connection. The first parameter identifies the physical name of the connection. The second optional parameter tags the connection with a logical name for use by other NAM directives. If not specified explicitly by the second parameter, the logical name defaults to the physical name. The third optional parameter specifies a particular physical communications media to use for the connection.

The third optional parameter is meaningful only for specific NAM installations. The NBS installation maintains an ARPA Network host interface, an automatic calling unit interface to the switched telephone network, and a specially designed patch panel interface to other local NBS "hard-wired" computer resources. The third parameter specifies a connection on one of these media; the parameter "-n" refers to the ARPA Network, "-p" refers to the dialout telephone network, and "-w" refers to the local "hard-wired" NBS network. For convenience, the default parameters are "-nwp" in that order. More than one medium can be specified so that in the event of a connection failure, additional physical paths can be tried. This is useful when a host system like the NBS PDP-10 is accessible through more than one communications path. The NBS PDP-10 is accessible by all three media!

Sometimes, the physical connection can be implied by the connection's name. A connection name that is 7 digits long is assumed to be a telephone number; so the automatic dialer is used. Also, a list of names associated with telephone numbers cause those numbers to be dialed when that name is used. This is analogous to looking up a number in a

phone book. Similarly, ARPA Network host names automatically cause connections on that medium as do known NBS network "hard-wired" host names (cause connections to the NBS local "hard-wired" resources). An appendix specifies the particular connections used in the NBS NAM.

```
.connect 5551234
.connect mit-multics
.connect nbs-10 connection1 -wpn
```

4.5.3 Disconnect. Disconnect closes the connection specified by the optional logical connection name specified. The default connection is the current logical connection.

```
.disconnect
.disconnect connection1
```

4.5.4 Exit. Exit causes an unconditional return from the macro expander at the current level of recursion. Like a subroutine or function call, the macro returns to the caller. The last return is back to the Command Interpreter.

```
.exit
```

4.5.5 Flush. Flush causes any accumulated input characters from remote host connections to be cleaned out of the NAM host input buffers.

```
.flush
```

4.5.6 Int. Int causes the named integers to be allocated for the duration of the macro expansion. The scope of the integer is local; no global significance occurs unless the integer is declared directly from the user's keyboard via the Command Interpreter.

```
.int i j k anyoldinteger
```

4.5.7 Match. Match defines a data structure representing strings of interest that are likely to occur in responses from remote host connections. As the response to a stimulus is received, the data structure is traversed. Matched, a predefined integer variable, is set to the data structure element contained in the response. Elements are numbered starting from zero and correspond to their position in the Match specification.

Because the syntax of the argument to Match is clumsy, this directive is seldom used; an alternative syntax has not yet been implemented and probably won't be because the string manipulation capability that includes pattern matching is much more powerful. The Match syntax,

informally presented through example, is only done so for completeness.

The first parameter to Match defines a Boolean combination of substrings that occur a given number of times. The second optional parameter specifies the logical name of the connection of interest. For example,

```
.match " 3'hello' ! 'network' & 'broken' " connection1
```

is parsed by identifying the first parameter between double quotes. A data structure is built that represents three occurrences of hello, one occurrence (by default) of any string containing network and broken; this match is useful when responses occur on the logical connection named connection1. Notice that in the match syntax the "or" operator is "!". Also notice the required use of single and double quotes.

This directive and the Term directive defined below are used by the Expander before the Send directive is executed. The data structures defined by Match and Term must exist prior to sending a stimulus so that the response to that stimulus can be evaluated immediately. Once the Match and Term data structures are built, any number of Send directives may be executed. Additional examples of Match are:

```
.match "'user' ! '[CR]@'"  tenex-bbnb
.match " 2'[DC1][DC3]' & 'PROG:'"
```

4.5.8 Msg. The msg directive causes the expression to be sent to the user terminal. The expression is evaluated and the result is converted to type "string". For example:

```
.msg "five equals 1 plus 4"
.msg "five equals " . ( 1 + 4 )
.msg "Result " . ( scratch @ ".* {/0-9//0-9/*} .*$" )
```

4.5.9 Newdir. Newdir causes the user's current working directory to be changed to the specified new directory.

```
.newdir /bib/medline
```

4.5.10 Output. Output works as Msg works except the resultant string is sent to the remote system rather than the user terminal. Normally, Output is not used because no conditions within the macro -- termed, matched, or response -- are set as with Send, Term, and Match; consequently, the macro can not conditionally adapt to system responses that

result from the Output stimulus.

```
.output "520,217[CR]"
```

4.5.11 Pipe. Pipe provides a mechanism for transfering data between remote host connections on a logical name basis. Preconditioning of the host connections to transmit and receive data is usually required. The preconditioning may require macros that log in, and run predefined host programs that transmit and receive the data. So, pipe is only a mechanism for forwarding data received, it is not a mechanism for initializing data transfers; other NAM directives must be used to set up and initialize the transfer.

The transfer is completed when received data meets the criteria established in the data structure specified by the last Term directive. So, like Send, Pipe requires a Term and Match data structure. And, like Send, the predefined integer variables termed and matched are set accordingly. An example is:

```
.pipe connection1 connection2
```

4.5.12 Printall-off. Printall-off causes the NAM to print, without modification, all characters received from the current remote system connection. The rules for verbosity as described in Verbose-on and Verbose-off apply.

```
.printall-off
```

4.5.13 Printall-on. Printall-on causes the NAM to print all characters received from the current remote system connection in the following format: All control characters are counted and displayed with that count and with there standard ASCII abbreviations. The control characters are displayed inside of square brackets.

```
.Printall-on
```

4.5.14 Remote. Remote causes the Command Interpreter to enter store-and-forward mode. In this mode, all characters typed by the user are sent (character-at-a-time) to the remote connection specified by the logical name provided. The default connection is the current connection.

The user must explicitly request to leave store-and-forward mode by typing a control shift back-arrow ASCII character.

```
.remote
.remote connection2
```

4.5.15 Rremote. Rremote (return from remote) works almost the same as Remote except that the character required to return the user from store-and-forward mode back to the NAM is specified as the parameter.

```
.rremote [CR] connection1
```

4.5.16 Send. Send causes the NAM to stimulate the current remote connection with the value of the specified expression converted to a string. Additionally, the Response Analyzer, using the data structures previously built by the Term and Match directives, sets the predefined NAM integer variables termed and matched based on the content of the received response.

Since control passes from the Expander to the Analyzer, the user is "out-of-the-loop" until the Term data structure has been successfully traversed. To keep the user "in-the-loop", a special store-and-forward data path through the NAM between the user and the remote connection is established for the duration of the Send directive. This path gives the user control over the host even when the NAM Analyzer is busy.

Occasionally, on initial connections, the remote system responds with a herald message or other status message before the NAM stimulates the host. Sending a null stimulus is required to force the NAM to enter the Response Analyzer. Some example Send directives follow:

```
.send ""
.send "/logon Rosenthal NBS-GenAcct[CR]"
.send "Explode " . ( sl @ "'MESH NO.'{/0-9//0-9/*}" )
```

4.5.17 Set. Set causes the named variable to be set to the value of the expression specified. A noise word between the variable name and the expression is required -- "to" makes a good noise word. For example:

```
.set connection_made to 0
.set flag0 to "100" + 10 / flag1
.set GETMESHNO to ".* 'MESH NO. ' {/0-9//0-9/*}"
```

4.5.18 Status. Status causes the NAM to print the current connection status on the user's terminal. An example is:

```
.status
```

4.5.19 String. String causes the named strings to be allocated for the duration of the macro expansion. The scope of the string is local; no global significance occurs unless the string is declared directly from the user's keyboard via the Command Interpreter. For example:

```
.string s1 s2 anyoldstring GETMESHNO
```

4.5.20 Term. Term (short for terminate) defines a data structure representing strings of interest in the responses from remote host connections. These strings are descriptions of anticipated responses for the Command Interpreter to match against the actual responses received. When a match occurs, the actual response that trickled into the NAM is buffered and stored in the NAM predefined variable named "response". Response is available for processing by other NAM macro directives.

So, in effect, the Term directive specifies, through strings of anticipated responses, conditions that make a completed response -- a response that becomes a completed buffer addressed using the NAM string "response". In addition to string match conditions, a time-out value can be specified. The time-out causes all previously received data that trickled into the NAM to be buffered and made available in the response variable just as the string match conditions do. The condition causing the termination -- either string match or time-out -- is specified in the predefined NAM integer variable called "termed".

"Termed" is set to the data structure element matching the substring element contained in the response. The syntax is similar to that of the Match directive with the addition of the time-out value. For example

```
.term " t30 ! 2'[CR][LF]' & '.' "
```

causes a data structure to be built representing a termination condition of 30 seconds, or two occurrences of the string carriage return line feed and the occurrence of one period. The time-out value applies to time between received characters. In the example above, 30 seconds would have to elapse between characters received by the NAM before the response is considered complete or terminated. And, since the time-out specification occupies the zero position (not the first, as numbering starts at zero), the variable "termed" would be set to zero if the time-out occurred.

On the other hand, if two carriage return line feed sequences and one period occur before 30 seconds elapse between characters, then the termination of the response is complete and the variable "termed" is set to one.

To be useful in identifying responses to stimuli, the NAM Response Analyzer must have both a match data structure and a term data structure predefined by the Match and Term directives. These data structure definitions must occur before the stimulus so that the response can be analyzed in real time (as the response happens on a character-at-a-time basis). The following examples illustrate typical sequences of Term, Match, and Send directives. Notice that once the data structures are built, they remain for the duration of the session unless changed by another Term or Match directive.

```
.term " t30 ! '[CR][LF]@' "
.match " 'TENEX' "
.send "[ETX]"
.send "login Rosenthal[CR]"
.term " 2'*' ! t60 "
```

4.5.21 Tmatch. Tmatch produces a data structure for both Term and Match using the same Boolean combination of strings.

```
.tmatch " t5 ! '.' ! ':' ! '@' "
```

4.5.22 Transcript-off. Transcript-off causes the file opened by Transcript-on, to be closed.

```
.transcript-off
```

4.5.23 Transcript-on. Transcript-on causes a file named "transcript" to be created in the user's current working directory. This file contains all of the characters received by the NAM from the remote system connection. Thus, a transcript is maintained. Transcript-off closes this file so that other UNIX commands can manipulate the transcript data.

```
.transcript-on
```

4.5.24 Unix. The Unix directive provides an interface to the UNIX operating system from NAM macros. Any UNIX command may be specified with the exception of those commands interpreted directly by the UNIX Shell such as "login", "logout", "chdir" (change directory), and "newgrp".

```
.unix lpr transcript
.unix banner "hurray"
.unix /mntl/rmr/myprogram
```

4.5.25 Verbose-off. Verbose-off causes characters received by the NAM from remote host connections to be only sent to the Response Analyzer. For example:

```
.verbose-off
```

4.5.26 Verbose-on. Verbose-on causes characters received by the NAM from remote host connections to be sent to the user terminal as well as to the Response Analyzer. This capability allows the user to watch the expansion of a macro as it generates and sends stimulus to remote connections. For example:

```
.verbose-on
```

4.5.27 Wait. Wait suspends execution of a NAM macro for the number of seconds specified. For example:

```
.wait 5
```

4.6 CONDITIONAL EXPANSIONS

Execution of NAM Macro directives proceeds sequentially through a macro until an "end-of-file" condition is reached. Conditional directives alter the sequential flow by evaluating expressions for their Boolean significance -- False is 0 and True is 1. Three constructs provide for the conditional expansion of macros: If-then-else, While, and Switch. The conditional directives are distinguished from other directives by the presents of an asterisk proceeding the directive name.

4.6.1 Case. Case identifies an alternative branch within a Switch directive. Case requires an expression that identifies the alternative. For example:

```
*case 2
*case "connection established"
```

4.6.2 Default. Default identifies the alternative branch within a Switch directive not covered by a particular Case directive. For Example:

```
*default
```

4.6.3 Else. Else identifies the Else alternative of an If-then-else construct. For example:

```
*else
```

4.6.4 End. End delimits the scope of an If-then-else, a Switch, or a While conditional directive. Nested conditional directives require specific End directives. For Example:

```
*end
```

4.6.5 If. If identifies the beginning of an If-then-else construct. The expression specified is evaluated and tested for True or False. If the expression is True, the Then portion is expanded; and, if the expression is False, the Else portion is expanded. The keyword Then is never used, but when the optional Else clause exists, *else must be present. An End directive is required to delimit the scope of the If-then-else construct. For example:

```
*if ( ~ connection_made )
        .connect mit-multics hostl -n
  *end


*if ( response ? ".* ( 'net busy' | 'net trouble' ) " )
        .disconnect hostl
        .msg response
        .exit
  *else
        .term " t5 ! '.' "
        .send "where rmr[CR]"
  *end
```

4.6.6 Switch. Switch identifies the begining of a Switch construct that optionally contains Case and Default directives. The Switch expression is evaluated and compared against the evaluated expressions in the Case directives. When the two expressions are equal, directives within the scope of the Case are expanded. If no Case expression evaluates to the expression of the Switch, the directives following the Default directive are expanded. The Break NAM directive is useful for transfering control outside the scope of the Switch construct. Without Break, all Case directives would be evaluated. For example:

```
*switch key
*case "("
*case ")"
```

```
        *case "or"
        *case "and"
        *case "not"
                .set result to result . " " . key
                .break
        *case ""
                .break
        *default
                .flush
                .send "select " . key . "[CR]"
                .set ssno to response @ ssnopat
                .set result to result .  " " . ssno
                .break
        *end
```

4.6.7 While. While identifies the beginning of the While construct. The directives between the While and the delimiting End are expanded repeatedly as long as the While expression is TRUE. The expression is checked before the directives are expanded. For example:

```
   ;while the string s is not null
   *while s # ""

        .set key to s @ next
        .set s to s @ rest

        *switch key
        *case "("
        *case ")"
        *case "or"
        *case "and"
        *case "not"
                .set result to result . " " . key
                .break
        *case ""
                .break
        *default
                .flush
                .send "select " . key . "[CR]"
                .set ssno to response @ ssnopat
                .set result to result .  " " . ssno
                .break
        *end
   *end
```

The previous examples include many NAM directives used to implement several of the common commands presented below.

5. COMMON COMMAND LANGUAGES

The NAM has been successfully used in a number of experiments including common command language development for bibliographic retrieval. The purpose of this section is to emphasize the applicability of the NAM through example macros; no pretense is made to describe or justify bibliographic search and retrieval. In particular, the details of host system interactions are not compared and contrasted -- there are obvious differences in the syntax and semantics used by the hosts involved and it is beyond the scope of this report to address these differences. While this section only addresses a common command language for bibliographic retrieval, other NAM experiments are currently underway that deal with common commands languages for file manipulation and job execution [FIT 78].

The bibliographic retrieval community sparks excitement among NAM experimenters because of the diversity and background of users interested in performing searches and because of the diversity of the services provided. Data bases are often so large that more than one or two seldom fit on one machine and consequently users are forced to learn and manipulate data with more than one command language through different communications media -- a perfect test bed to try out the kinds of access assistance techniques proposed by the NBS NAM.

For this experiment five target services were chosen and attention focused on a small, but "relevant" and "doable", subset of commands or operations. The five services are MEDLINE, ORBIT, RECON, DIALOG, and BASIS. Table 4 briefly overviews the scope of this experiment; only the more important operations are listed with their user syntax. Complete documentation and motivation for this work is provided in [TRE 78].

OPERATION	EXAMPLE USER SYNTAX
connect	connect to system -|-- medline |-- orbit |-- recon |-- dialog |-- basis
display	display -|-- file names |-- terms related to <search term> |-- <n> citations
access	access file -|-- <file name> |-- <file number>
find	find <search statement>
stop	stop session

TABLE 4: PORTIONS OF A COMMON COMMAND LANGUAGE FOR BIBLIOGRAPHIC RETRIEVAL

The actual NAM macros that implement this user language are organized in the UNIX tree structured file system under directories named according to the host system. Figure 4 diagrams this file structure. The diagram shows a parent directory named "bib" that contains a macro file named connect and the other directory entries.

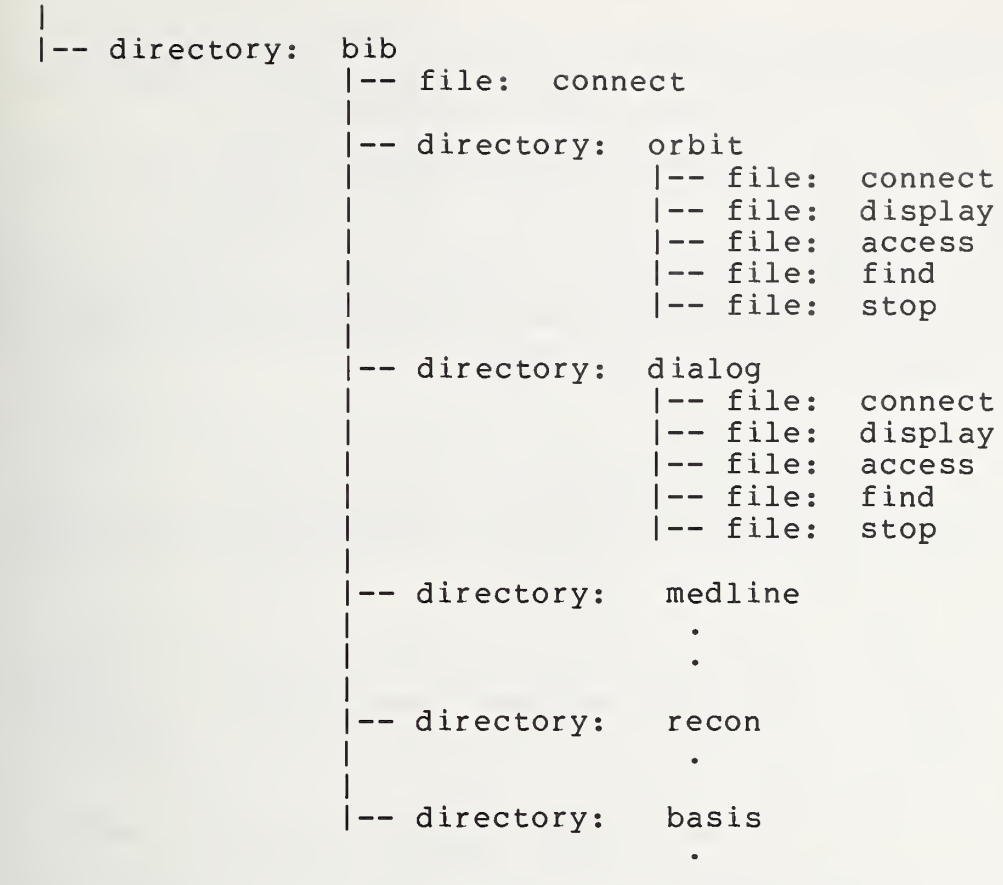

FIGURE 4: DIAGRAM OF UNIX TREE STRUCTURED FILE SYSTEM CONTAINING NAM MACRO FILES

The macro file "connect" in directory "bib" checks the syntax of the user's input command and then, using the ".newdir" directive, switches to the new directory named as the object of the "connect to system" common command. This is easy to accomplish using NAM directives as in the following specification of the bib/connect macro:

```
.set sl to ".*{'medline'|'orbit'|'recon'|'dialog'|'basis'}"
.set name to userline @ sl

*if ( name = "" )
  .msg "connect to system <name>"
  .msg "<name> := medline | orbit | recon | dialog | basis"
  .exit
 *end

.newdir $name.
connect
```

After executing the .newdir directive, the user's current working directory becomes the directory named; and, when the final line "connect" is executed as a macro file name, the connect file in that subdirectory is executed.

The UNIX file structure allows files of the same name to appear in different directories. This provides a convenient method that allows the same user NAM command name to generate completely different sequences for different remote systems. The connect macro in directory bib/orbit is completely different from the connect macro in bib/dialog. So, for instance, if the user types:

connect to system orbit

then the current working directory is changed to orbit and the macro named connect in directory orbit is executed. The orbit connect macro is listed below.

Again, keep in mind that the connect macros in the dialog, medline, recon, and basis directories are different -- they generate different NAM-host interactions -- but all of these connect macros ultimately result in the physical establishment of a connection to the host with the user "logged in" and ready to use the service -- the msg directive .msg "login successful" appears in each connect macro. Here is the orbit connect macro. (The user identifiers and passwords have been changed to protect the integrity of real accounts.)

```
.set connection_made = 0
.connect tymnet orbit -p

*if ~ connection_made
        .msg "sorry, please try again"
        .newdir ..
        .exit
     *end

.term "t30 ! 'identifier' ! ':' ! ';' ! '|'"
.match "'.'"

.send ""
*if ( response ? ".*'|'" )
        .msg "remote connection error[CR][LF]"
        .msg "no response, please try again"
        .disconnect
        .newdir ..
        .exit
     *end

*if ( response ? ".*'identifier'" )
        .send "e"
    *end

*if ( response ? ".* ( 'name' | 'log in' ) " )
        .send "mylogname[CR]"
    *else
        .msg "remote connection sequence error"
        .msg "please try again"
        .disconnect
        .newdir ..
        .exit
    *end

.send "mylogpassword[CR]"

.term "t60 ! 'USER:'"

.send "/login myname[CR]"

*if ( response ? ".* 'USER:'" )
        .send "n[CR]"
        .msg "Login successful[CR][LF]"
        .exit
    *end

.msg "Problem logging into ORBIT[CR][LF]"
.msg response
.disconnect
.newdir ..
```

While the connect macros for the other host systems generate completely different sequences to the host, the messages to the user are the same regardless of the host. In this way the common command language provides consistent user response messages for anomalies that may be encountered while attempting to "connect"; anomalies like busy signals on telephone calls, busy ports on front-end concentrators, and other contingencies that may or may not be uncommon. In addition, responses to nominal host behavior -- messages like "login successful" -- reassures the user that progress is being made.

Once connected to a remote host (service) the user may display the names of files or data bases supported by the host. The display macro for MEDLINE is representative of the display macros for the other systems. First, using NAM directives, the macro determines if the user typed "display file names", "display terms related to <name>", or "display <n> citations"; each case is handled differently. The medline display macro is:

```
.string s
.int i

.term "t60 ! 'user:[cr lf dc3 dc3 dc1]'"
.match "':'"

*if ( userline ? ".* ( 'database names' | 'file names' ) " )
        .send "
        .exit
    *end

*if ( userline ? ".* ( 'terms' | 'words' )" )
        .set s to userline @ ".* 'related to '{.*} '[LF]' "
        .send "
        .send "0[CR]"
        .exit
    *end

*if ( userline ? ".* 'citation'" )
        .set i to userline @ ".* {/0-9//0-9/*} ' citation'"
        .send "\"prt " . i . "\"[CR]"
        .exit
    *end
.msg "display <object>"
.msg "<object>    := file names | database names | terms related to |
.msg "               words related to | <number of> citations"
.msg "<number of> := a number"
```

The access file macro is also straight forward; the example

for ORBIT is given below:

```
.term "t60 ! 'user:[CR][LF]' ! 'USER:[CR][LF]'"
.match "':'"
.send "
.msg response @ ".* 'PROG:' {.*} 'SS ' /0-9/ "
```

The find command for DIALOG is completely different from the other find commands which are relatively straight forward; and, special treatment is given to this example. The find strategy used by DIALOG is to first select the key words that represent "hits" against words in the data base. Having selected the key words, a DIALOG combine command is formulated using the sequence numbers of the selected words. So, in this example, the NAM must extract the search statement number from the response and use that extracted number in the formulation of a new command to the host system.

```
.term "t40 ! '? '"
.match "'?'"

.string s result key next rest ssno ssnopat hits hitspat

.set s to userline @ "'find' {.*} '[LF]'$"
.set next to "' '* ( {/a-z//a-z/*}(' '|'('|')'|$) | {'('|')'} )"
.set rest to "' '* ( (/a-z//a-z/*)(' '|{'('!')'}|$) | ('('|')') ) {.*}$
.set ssnopat to ".*'[CR]'.*{/0-9//0-9/*}' '"
.set hitspat to ".*'[CR]'.*/0-9/.*' '{/0-9//0-9/*}' '"
.set result to ""

*while s # ""
        .set key to s @ next
        .set s to s @ rest
        *switch key
        *case "("
        *case ")"
        *case "or"
        *case "and"
        *case "not"
                .set result to result . " " . key
                .break
        *case ""
                .break
        *default
                .flush
                .send "select " . key . "[CR]"
                .set ssno to response @ ssnopat
                .set result to result .  " " . ssno
                .break
        *end
*end
.send "combine" . result . "[CR]"
.msg ( response @ hitspat ) . " hits on " . ( userline @ "'find'{.*}$"
.msg "Enter next search (# " . ( response @ ssnopat ) . ")"

.set flag0 to ( ( response @ ssnopat ) - 1 )
```

A few observations about this macro are worth making. First, the macro allows for the bibliographic search statement to contain the Boolean search primitives "or", "and" ,and "not". Further, the parenthetical juxtaposition of the input user command is maintained so that only the default words -- those that are keys for the search -- get selected.

The resultant combine command is dynamically formed from the previous select command by concatenating the string variable named result. Finally, after exhausting the user input string, the combine command is transmitted to the host

and the common user message that displays "hits" gets sent to the user.

The stop session command is typical for all systems. Here, the medline example is presented.

```
.term "t30 ! '[DC1]'"
.match "'[DC1]'"
.send "
.term "t20 ! 'good-bye!' ! 'GOOD-BYE!'"
.match "'good-bye!'"
.send "yes[CR]"
.wait 3
.disconnect  medline
.newdir ..
```

6. BIBLIOGRAPHY

[AND 77] Anderson, R. H., Gallegos, M., Gillogly, J. J., Greenberg, R., Villanueva, R., RITA Reference Manual, A report prepared for the Defense Advanced Research Projects Agency (ARPA order No.: 189-1, 7P10 Information Processing Techniques) Sept., 1977, 68p.

[FIT 78] Fitzgerald, M. L., Common Command Language for File Manipulation and Network Job Execution NBS Special Publication, To be published.

[GRI 71] Griswold, R.E., Poage, J.F., and Polonsky, I.P., The SNOBOL 4 Programming Language, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, 256 p.

[MAR 76] Marcus, R.S., Reintjes, F.J., The Networking of Interactive Bibliographic Retrieval Systems MIT Report ESL-R-656, Massachusetts Institute of Technology, Electronic Systems Laboratory, Cambridge, Massachusetts, March 31, 1976. 164p.

[RIT 74] Ritchie, D. M., and K. Thompson, "The UNIX Time-Sharing System," Comm. ACM 17, 7, July 1974, pp. 365-375.

[ROS 76] Rosenthal, Robert, A Review of Network Access Techniques with a Case Study: The Network Access Machine, NBS Technical Note 917, July 1976, 36p.

[TRE 78] Treu, Siegfried, A Testbed for Providing Uniformity to User-Computer Interaction Languages NBS Special Publication, To be published.

7. APPENDIX

7.1 More on the Response Analyzer

A key problem in building the Response Analyzer is knowing when the system response is completed. Two criteria are used to make this decision: a timeout, and string match -- both criteria are specified by the NAM macro author. The specification is made using the Term directive. Here is an example Term directive:

```
.term   " t5 ! '[CR][LF]@' ! 3'.' & '*'"
```

This directive specifies that the system response is complete when five seconds elapses between the receipt of host characters or when the string carriage return line feed at sign is received or when three occurences of a period and an asterisk is received. The string match can be any Boolean combination of substrings as defined in the Term directive in section 4.5.21.

To provide the capability to terminate on any string, a special read operation is performed on the interface used to connect the NAM to the remote host. This read operation is performed in "raw" mode on UNIX. Each character one-at-a-time is read, and buffered; each time a character arrives from the host, NAM software compares the accumulating buffered characters against each substring in the term directive until a match occurs.

When the condition for termination is met -- either time out or string match -- the buffer of accumulated characters is made available in the string variable named "response". Because of buffer space limitations, "response" only accumulates the last 4096 characters of the system response.

The time out field and each substring field in the Term directive are numbered from left to right starting at zero. In the example above, the time out field is numbered zero, the field with the carriage return is numbered one, and on and on. The number of the field that caused the termination of the system response is made available in the integer variable called "termed".

Before the string manipulation directives were implemented, the Match directive played an important role. NAM macro authors defined substrings that identified possible patterns within terminated system responses. Using the same numbering scheme for fields within the Match

directive, authors could identify a particular anticipated string within a terminated response. The integer variable "matched" contains the field number. However, since the powerful string manipulation operators were implemented, the Match directive is seldom used.

One additional point is worth mentioning. The Send directive causes the NAM to output a stimulus message to the remote system. Control within the NAM is then passed to the Response Analyzer. The Analyzer does not return control to the Macro Expander until a termination condition is met. In this way, message flow between the NAM and the remote host is controlled. In particular, only one Send directive can be executed at one time.

7.2 NBS NAM Physical Connection Names

The NAM minicomputer is part of an experimental computer facility at the NBS. It is connected as a host on the ARPA Network and to other less publicized local NBS Networks. Additionally, through an automatic calling unit, the NBS NAM can place direct dial telephone calls to any remote access service. The following table represents the current names given to many of the commonly accessed resources by the everyday users of the facility.

COMMUNICATIONS MEDIA	PHYSICAL NAME	LOCAL PORT
Switched Telephone Network Names	dialog hotline nbs-10 tip tymnet univac unix451 unix453 <any phone number>	automatically allocated by the automatic calling unit software
Local NBS Facilities	NBS Tip NBS Tip NBS Unix 1 NBS PDP-10 Dial out Dial out Dial out Dial out	DH - 8 DH - 9 DH - 10 DH - 11 DH - 12 DH - 13 DH - 14 DH - 15
ARPA Network	Standard ARPA Network Host Name table entries	ARPA Network Host Interface (IMP-11)

NBS-114A (REV. 11-77)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-35	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

COMPUTER SCIENCE & TECHNOLOGY

The Design and Implementation of the National Bureau of Standards' Network Access Machine (NAM)

5. Publication Date: June 1978

6. Performing Organization Code

7. AUTHOR(S)
Robert Rosenthal and Bruce D. Lucas

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No.
6502129

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address *(Street, City, State, ZIP)*

Same as 9.

13. Type of Report & Period Covered
Interim
June 1977 - Present

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

The Network Access Machine (NAM), a programmed minicomputer designed to assist interactive on-line terminal users of computer network services and resources, is discussed in detail. The minicomputer allows the user to specify (or to have specified) network command sequences for execution on a specified network and host connected to that network. Computer responses are analyzed to assure agreement with those anticipated for specific commands. Experience with the NAM and specific examples of NAM use including a common command language for bibliographic retrieval are presented.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Command languages; communications; computer access; computer networks; intelligent terminals; minicomputers; protocols.

18. AVAILABILITY

[X] Unlimited

[ ] For Official Distribution. Do Not Release to NTIS

[ ] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Stock No. SN003-003

[ ] Order From National Technical Information Service (NTIS) Springfield, Virginia 22151

19. SECURITY CLASS (THIS REPORT)
UNCLASSIFIED

21. NO. OF PAGES
50

20. SECURITY CLASS (THIS PAGE)
UNCLASSIFIED

22. Price
$2.20

USCOMM-DC 66035-P78

☆ U. S. GOVERNMENT PRINTING OFFICE: 1978—261-238/142

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, D. C. 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology, and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent NBS publications in NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $17.00; foreign $21.25. Single copy, $3.00 domestic; $3.75 foreign.

Note: The Journal was formerly published in two sections: Section A "Physics and Chemistry" and Section B "Mathematical Sciences."

DIMENSIONS/NBS

This monthly magazine is published to inform scientists, engineers, businessmen, industry, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on the work at NBS. The magazine highlights and reviews such issues as energy research, fire protection, building technology, metric conversion, pollution abatement, health and safety, and consumer product performance. In addition, it reports the results of Bureau programs in measurement standards and techniques, properties of matter and materials, engineering standards and services, instrumentation, and automatic data processing.

Annual subscription: Domestic, $12.50; Foreign $15.65.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a world-wide program coordinated by NBS. Program under authority of National Standard Data Act (Public Law 90-396).

NOTE: At present the principal publication outlet for these data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St. N.W., Wash., D.C. 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The purpose of the standards is to establish nationally recognized requirements for products, and to provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, D.C. 20402.*

Order **following** *NBS publications—NBSIR's and FIPS from the National Technical Information Services, Springfield, Va. 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Services (Springfield, Va. 22161) in paper copy or microfiche form.

BIBLIOGRAPHIC SUBSCRIPTION SERVICES

The following current-awareness and literature-survey bibliographies are issued periodically by the Bureau:

Cryogenic Data Center Current Awareness Service. A literature survey issued biweekly. Annual subscription: Domestic, $25.00; Foreign, $30.00.

Liquified Natural Gas. A literature survey issued quarterly. Annual subscription: $20.00.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $30.00. Send subscription orders and remittances for the preceding bibliographic services to National Bureau of Standards, Cryogenic Data Center (275.02) Boulder, Colorado 80302.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

OFFICIAL BUSINESS

Penalty for Private Use, $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

SPECIAL FOURTH-CLASS RATE
BOOK