ED 179 205 IR 007 867

AUTHOR Fitzgerald, M. L.
TITLE Computer Science and Technology: Common Command Language for File Manipulation and Network Job Execution: An Example.
INSTITUTION National Bureau of Standards (DOC), Washington, D.C.
REPORT NO NBS-SP-500-37
PUB DATE Aug 78
NOTE 33p.
AVAILABLE FROM Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402 (Stock No. 003-003-01965-8, $1.50)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Information Networks; *Information Retrieval; Information Storage; *On Line Systems; *Programing Languages; Task Performance
IDENTIFIERS Resource Sharing

ABSTRACT

Recognizing that the use of resource sharing capabilities provided by computer networks is inhibited by the requirement that the user become familiar with the varied command languages and protocols of each accessed system, this report presents a general approach to solving the problem using an intermediary system to support a set of common commands for file manipulation and network job execution. Tested in an experiment with four different systems, this common command language permits the network user to easily transfer files between varied systems of the network and provides the capability to execute programs at particular hosts using data that may not initially be available where the program is to be executed. An account of the development and implementation of the command language is followed by a description of its usage in system interconnection, intrasystem file handling, intersystem file transporting, and network job execution. An appendix provides a display of the command language and a list of pertinent references. (FM)

COMPUTER SCIENCE & TECHNOLOGY:

Common Command Language for File Manipulation and Network Job Execution: An Example

M. L. Fitzgerald

Computer Systems Engineering Division
Institute for Computer Science 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 August 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 500-37
Nat. Bur. Stand. (U.S.) Spec. Publ. 500-37, 32 pages (Aug. 1978)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 78-600069

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1978

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20434
Stock No. 003-003-01965-8 Price $1.50
(Add 25 percent additional for other than U.S. mailing).

TABLE OF CONTENTS

4

DISCLAIMER

Common Command Language for File Manipulation and Network Job Execution: An Example

M. L. Fitzgerald

Computer networks provide the capability for sharing resources across many diverse computer systems. Utilizing this capability is inhibited by the requirement that the user become familiar with all the varied command languages and protocols of each accessed system. This report presents a general approach to solving this problem using an intermediary system to support a set of Common Commands for File Manipulation and Network Job Execution. To show the feasibility of this approach, common commands were implemented for four systems using the NBS Network Access Machine.

Key words: Command language; Computer networks; Network access; Macros.

This Special Publication is one of a series prepared as a 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.

I. INTRODUCTION

Computer networks have made available to the programming community the services and resources of many types of computer systems. Unfortunately, each of these systems has its own, sometimes complex, language and methodology for accessing its resources. A network user is faced with the task of becoming familiar with the specific command language for each system before using the services available on that system.

One method for simplifying the usage of computer networks might be to develop a set of standards for network access [NBS 77] [NEU 75]. The vendors of computer services would be required to make their products compatible with these standards if the services were to be available to the networking community. In the absence of such a standard, a different method is required.

An alternative approach to simplifying networking for users is to develop a standard language to which the diverse sets of commands for heterogeneous hosts can be translated. This language should provide a common set of commands to gain access to the computer systems on the network, and to manipulate files on those systems. Since the network user is concerned with many different systems, the language should present a uniform view of the systems through the standard commands and also remove many of the complexities of inter-system interactions. Therefore the language should permit the network user to easily transfer files between different systems of the network and provide the capability to execute programs at particular hosts using data that may not initially be available where the program is to be executed.

In order to show the feasibility of such a language, an experiment was performed in which common commands were developed for the following four systems: a Honeywell 6180 running MULTICS, a PDP-10 running TOPS-10, a PDP-10 running TENEX, and a PDP-11/45 running UNIX. The language was implemented on a PDP-11/45 under the UNIX operating system using the Network Access Machine [ROS 76] as its foundation. Expanding the language to include other systems is very straightforward and requires little programming effort.

II. DEVELOPMENT OF THE COMMON COMMAND LANGUAGE

The development of the Common Command Language (CCL) started by determining some of the desired properties of a common command language. Next, a survey of the existing command languages of the systems in the experiment was made to define the required commands. The CCL was designed as an attempt to provide a viable language for network users of those systems.

II.1. COMMAND LANGUAGES

The primary concern of most computer or network users is to get a job done, ideally as simply as possible. Usually, the user has little or no interest in the various peculiarities of the different systems to be used. This includes differences in the syntax and semantics of the command languages of those systems. The user should also be shielded from dealing with the complexities of error messages which may occur during a terminal session.

At best, then, the language provided to a network user should allow maximum access to the features of all the systems available with minimum system dependent interaction. In addition, it must supply a reasonable cross-section of all the available services each system maintains, but it may not necessarily handle commands dependent upon the physical properties of the system due to differences in hardware configurations and equipment.

Extensive study has been made to determine the most desired properties of a common command language [UNG 74]. The properties should include the following:

Obvious	The commands should perform the actions their names imply.
Simple	The commands should be simple in format and machine independent.
Uniform	The functions performed should be equally adaptable to all systems involved.
Transparent	The user should be unaware of any system peculiarities and shielded from different types

	of error or system messages.
Extendable	The command repertoire should be easily extended to add more commands if necessary.

In order to develop the CCL the following three steps were taken:

1. A survey of the systems to be used was made to determine all the different types of (logical) file handling functions available. These functions deal with the files themselves as named entities, rather than with the contents of the files.

2. The command names for the functions were chosen, trying to maintain an intuitively meaningful vocabulary.

3. The commands were implemented on the Network Access Machine.

II.1.1. Command Survey.

Four types of computer systems were used for the development of the CCL. In order to select the commands to be implemented, a survey of the file manipulation commands of each of these systems was made, and for each different function represented, a common command was implemented.

Table II.1 indicates the types of file manipulation functions that are commonly available and the corresponding commands for the systems being surveyed. Dashes in the table indicate that the system does not have a command for the described function. (In some cases there may be a utility that can be invoked on a given system, but not a single command.)

These commands are for logical manipulations on files as entities within an operating system. Functions which use the other resources of the operating system such as editors will not be addressed at this time. Later this language will be incorporated with other functions to provide the capabilities of a Network Operating System (NOS) [KIM 76-1] [KIM 76-2]. The NOS environment will provide the user with the ability to use a network in the same manner that he would use a single system. Included within the NOS will be a network wide Directory System that will describe the user, the user's files on the network, the configurations and structure of the various hosts on the network, and the interactions permitted among them. In addition the NOS will provide functions for remote record access and data translation and transformation [WOOD 78].

Table II.1

Functions	MULTICS	TENEX	TOPS-10	UNIX
append one file to another	---	append	---	cat
compare two files	compare	---	---	cmp
make a new file	create	---	alcfil	---
remove file name from directory	---	delete	---	---
restore file to directory	---	undelete	---	---
remove file from directory	delete	expunge	delete	rm
make copy of a file	copy	copy	copy	cp
obtain directory listing	ls	dir	dir	ls
list contents of a file	pr	type	type	cat
change the name of a file	rename	rename	rename	mv

II.1.2. The Common Command Language Structure.

The CCL consists of four major command types: intra-system commands, inter-system commands, network job execution commands, and system inter-connection commands.

Intra-system Commands

Since each of the functions described in Table II.1 is available on at least one of the systems, it was decided that the CCL would contain a command for each of the functions represented. The names were chosen trying to maintain a correspondence between the functions performed and the command. In addition, some of the command names were chosen in keeping with some of the currently available commands.

append	append one file to another
compare	compare one file to another
copy	make a copy of an existing file
create	make a new file
delete	remove the file name from the user's directory
erase	remove a file completely
list	list the names of the user's files
rename	change the name of a file
type	display the contents of a text file to the user terminal
undelete	return a deleted file name to the user's directory

Inter-system Commands

Along with the commands for file handling listed above, the language contains commands to transfer files between systems.

retrieve	move file from remote host to

	primary host
transfer	move file from primary host to a remote host

System Interconnection Commands

Before a user can actually perform any file manipulations, access must be gained to the system where the files reside. The command language therefore contains commands to connect the user to a remote system and execute the particular login procedure required. Likewise, when the terminal session is over, there exist commands that will log the user out and terminate the connection.

Network Job Execution

Often, a network user may want to execute a program on a specific host and the desired data files reside on other hosts throughout the network. Ideally the user would like to be able to specify just one command that could indicate where the program is to be executed, where the input data reside, and where the output data should be sent. Such a command could have the format:

```
RUN  PROGRAM@HOST1@HOST2 IN1=INFILE1@HOSTA OUT1=OUTFILE1@HOSTB
```

where

PROGRAM	- program name
HOST1	- where program is to be executed
HOST2	- where program resides
IN1	- program's name for input file 1
INFILE1	- actual name for input file 1
HOSTA	- where input file 1 resides
OUT1	- program's name for output file 1
OUTFILE1	- actual name for output file 1
HOSTB	- where output file will be migrated

One of these commands would constitute a network job step and a sequence of network job steps could be known as a network job.

In order to execute a network job an additional command called 'run' is also part of the command repertoire. This command is unique because its actions are dictated by the user through two utility programs which are called to define the network job to be run.

II.2. IMPLEMENTATION

The implementation of the CCL was facilitated by utilization of the NBS Network Access Machine (NAM) which provides the communication support for this study [ROS 78]. The NAM is a program that runs under the UNIX timesharing system on a PDP-11/45. Through this program the user at an interactive terminal may establish connections with many types of remote computers. The NAM provides directives that permit the user to maintain a dialogue with the systems of a network. Sequences of these directives stored in files called macros have been written that will generate the necessary machine-dependent dialog to perform the file manipulation functions required for each of the host systems. The responses generated by the remote systems are in turn analyzed by the NAM. By permitting the mapping of the varied system responses into standardized messages for the network user to see, the NAM facilitates the uniformity of the CCL.

The commands of the CCL were implemented as a set of simply named NAM macros. Since each system requires a unique dialogue to perform a given function, a macro to generate that dialogue is needed for each system. Thus, the rename command is really implemented as several macros named 'rename', one for each of the systems in the study. The particular macro that is called depends upon which system the user is connected to. Incorporating a new system type merely involves adding a set of command macros for that system.

Figure II.2.1. shows two such macros for the rename command: one for a TOPS-10 system and one for a UNIX system. The TOPS-10 command to perform the file renaming function is 'rename' and provides one of three different responses to the user:

1) "File x already exists" if there already is a file with the new name.

2) "File renamed:" if the rename was successful and

3) "? No file named x" if the file to be renamed doesn't exist.

Macro for the command RENAME for a TOPS-10 system.

```
.term "t30 | '.'"
.match "'.' | 'already exists' | 'No file'" tops10
.send "rename "$2" = "$1"[CR]"
*switch matched
        *case 3:
                .msg "File "$1" not found.[CR][LF]"
                .break
        *case 1:
                .msg "File "$1" renamed to "$2".[CR][LF]"
                .break
        *case 2:
                .msg "File "$2" already exists.[CR][LF]"
*end
.flush
.exit
```

Macro for the command RENAME for a UNIX system..

```
.term "t30 | '%'"
.match "'not found' | 'not exist' | '%'" unix
.send "ls "$2"[CR]"
*if matched = 1
        .send "mv "$1"  $2"[CR]"
        *if matched = 2
                .msg "File "$1" not found.[CR][LF]"
                .flush
                .exit
        *end
        .msg "File "$1" renamed to "$2".[CR][LF]"
        .flush
        .exit
*else
        .msg "File "$2" already exists.[CR][LF]"
        .flush
        .exit
*end
```

Figure II.2.. Sample macros.

Therefore, the rename macro must send the TOPS-10 rename command to the system and analyze the system response. Using the NAM statements 'switch' and 'case', the macro outputs an appropriate message for the resulting system response.

On UNIX systems, however, renaming files is accomplished through the 'mv' command and existing files with the new name will be overwritten without warning the user. Therefore, the rename macro for UNIX first determines if a file with the new name exists. If not, the 'mv' command is sent to the system and the response is analyzed to determine if the rename was successful. In this macro, the NAM statements of 'if' and 'else' are used to differentiate among the possible system responses.

This ability of the NAM to analyze system responses, provides the mechanism to standardize messages that the CCL user sees. Each of the systems in the above example has different responses for non-existent files, yet the message the user will see will always be "File x not found" if the file to be renamed does not exist. Moreover, since the use of NAM macros permits multiple NAM-remote system interactions for one user command, the command level capabilities for the systems appears to be extended.

III. COMMON COMMAND LANGUAGE USAGE

This section describes the usage of the commands of the CCL. Each of the four types of commands is covered and a example of its usage is given.

III.1. SYSTEM INTERCONNECTION COMMANDS

In order to perform any file manipulations on a specific host via a network, a user must gain access to that host and execute a successful login procedure. Two types of connections can be made by the network user: primary connections which connect him to the host where most of the session will occur; and secondary connections which are used to allow file transfers between different hosts. For this study, most of the systems were primarily available through the ARPA network. BBN refers to the host BBN-TENEX, while ISI is the USC-ISI TENEX system. NBS-10 and UNIX451 refer to hosts within the NBS Experimental Computer Facility and MULTICS is the MIT-MULTICS. (In addition, the language was expanded to include the UNIX453 machine at NBS and the Multics system at the Rome Air Development Center.)

III.1.1. Establish Primary Connections.

The following commands make and break primary connections on the systems the user wishes to access. A primary connection is made to the system the user expects will be utilized most during the session at the terminal. These commands are identified by the names of the systems to be accessed.

BBN <Identifier> ISI <Identifier>	A user enters this command to establish a primary connection with the TENEX systems at BBN or ISI. The identifier is the user's name as known to the system and the user will be interactively requested to supply password and accounting information.
MULTICS <Identifier>	A user enters this command to estabish a primary connection with MULTICS at MIT. The

	identifier is the name of the user as known to the system, and again password information must be supplied as required.
NBS10 <Identifier>	This will establish a primary connection with the NBS PDP-10 system running TOPS-10. The identifier is the user's project-programmer number on the 10 and password information is requested.
UNIX451 <Identifier>	The command establishes a primary connection with a PDP-11/45 running UNIX. The user will be connected to and logged onto the UNIX451 system at NBS. A password will be requested.

III.1.2. Establish Secondary Connections.

Secondary connections are made when the user wishes to transport files to or from a host other than the primary connection. These commands are the system name prefixed by 'conn' or 'leave' depending on whether the user is connecting to or disconnecting from the system.

CONNBBN <Identifier> CONNISI <Identifier>	A secondary connection is established with a TENEX system at BBN or ISI. The identifier to be supplied is the name of the user known to the system. The password and accounting information must be supplied by the user.
CONNMULTICS <Identifier>	A secondary connection is established at MULTICS. Again the system will request a password which the user must supply.
CONNNBS10 <Identifier>	A secondary connection is made with the NBS PDP-10 running

	TOPS-10. The identifier is the project-programmer number of the user at NBS and the system will request a password.
CONNUNIX451 <Identifier>	The command establishes a secondary connection with the UNIX451 system at NBS. Login commands are issued using the user's identifying name specified in the identifier parameter. The system again will require the user to enter a password.

III.1.3. Terminate Primary Connections.

LEAVE	This is the command issued to terminate communications with any of the four primary connections. The NAM is aware of which system the user has established as the primary connection and will execute this command to log the user out from that host and terminate the connection.

III.1.4. Terminate Secondary Connections.

LEAVEhost	This is the command used to close out the secondary connections. The host part of the command can be either BBN, ISI, MULTICS, NBS10 or UNIX451. The command logs the users off the secondary host and closes the connection with the NAM.

III.2. THE FILE HANDLING COMMANDS (Intrasystem)

The following is a description of each of the file manipulation commands, their usage and parameters.

APPEND FILE1 FILE2	FILE1 is appended to FILE2. If FILE1 does not exist, the message "File FILE1 not found." is given and FILE2 is unchanged. If FILE2 does not exist, a copy of FILE1 is made and called FILE2.
COMPARE FILE1 FILE2	If either FILE1 or FILE2 does not exist a message is given. Otherwise the character files are compared on a line by line basis and a message indicating whether they are different or equivalent is issued. (Comparison of non-character files will be a future option of this command.)
COPY FILE1 FILE2	A file called FILE2 is created that is a copy of the file called FILE1. If there already exists a file called FILE2, it is NOT overwritten and a message indicating that FILE2 already exists is given. If FILE1 does not exist the message "File FILE1 not found." is output to the user's terminal.
CREATE FILE1	The directory entry for file FILE1 is made. If there already is an entry by that name, the message "File FILE1 already exists." is given.
DELETE FILE1	File FILE1 is removed from the user's directory. This implies that the user may no longer access that file and

	may even create another file with the same name. However, the file itself still exists and may be undeleted (restored) at a later time, provided that either the user does not logout or does not do an erase command (see UNDELETE and ERASE.) If a file of the same name has previously been deleted, the message "File FILE1 previously deleted." is given to permit the user access to the previous version of the files. In this case either the previously deleted file must be erased, or in case it is still useful, the name of the current version of the file can be changed before it is deleted. If FILE1 does not exist, a message "File not found." is given.
ERASE	The command completely removes any previously deleted files. Once an ERASE is done no more UNDELETEs can be done on those files. In addition, an automatic erase is done when a user leaves the system.
LIST	All the file names from the user's directory are listed in the format of the directory of the system being used.
RENAME FILE1 FILE2	FILE1 is renamed to FILE2. If there already is a FILE2, the message "File FILE2 already exists." is given and the files retain their names. If there is no FILE1, the user will be informed.
TYPE FILE1	The contents of FILE1 are displayed on the user's terminal.

UNDELETE FILE1	The command returns any previously deleted file to the user's workspace.

III.3. INTERSYSTEM FILE TRANSPORTING COMMANDS

The CCL extends the file handling capabilities of the host computers studied by providing commands that transfer files between systems. Thus the user enters system level commands and has the ability to transfer and retrieve files to and from remote hosts. In order to use these commands, the user must establish a primary connection with one host, and a secondary connection with the host with which the exchange is to take place. In addition a directory listing capability for secondary connections is provided.

LISThost	The directory listing of the secondary host 'host' is obtained.
RETRIEVE FILE1 SECONDARYHOST	A file named FILE1 is sent to the primary host from the secondary host. It will be assumed at present that the file does indeed exist on the secondary host and a file of the same name does not as yet exist on the primary host.
TRANSFER FILE1 SECONDARYHOST	The file FILE1 is sent from the primary host to the secondary host. Again it is assumed that the file does exist on the primary host and a file of the same name does not exist on the secondary host.

III.4. A SAMPLE SESSION

In order to show the usage of the CCL, a scenario for a sample terminal session will be given and the commands used will be listed.

A network user has accounts at MULTICS and the NBS10. A primary connection is established with MULTICS and a listing of the files stored there is made. The files are: mul, mul2, and mul3. The user decides to combine mul and mul2 and just keep the combined file. In addition he decides to transfer this file to the NBS10 as a backup copy. Therefore he appends mul to mul2 and deletes mul. A secondary connection is made with the NBS10, a directory listing is requested, and the new mul2 is transferred there. He finds that in a previous session he made a file on the NBS10 that should have been a copy of the old mul file, but he is not sure. He retrieves that file, mul, from the NBS10 but cannot compare it with the old mul because he deleted it a few moments ago. Also, since the file he retrieved from the NBS10 has the same name as the deleted file, he cannot undelete it yet. So, he renames the retrieved file to mulnbsl and then undeletes the old mul. Now he compares the two files and since they are different, deletes them both. He is now finished and so terminates the secondary connection and then terminates the primary connection.

```
MULTICS WATKINS
LIST
APPEND MUL MUL2
DELETE MUL
CONNNBS10 510/46
LISTNBS10
TRANSFER MUL2 NBS10
RETRIEVE MUL NBS10
RENAME MUL MULNBS1
UNDELETE MUL
COMPARE MUL MULNBS1
DELETE MUL
DELETE MULNBS1
LEAVENBS10
LEAVE
```

The following is a listing of the terminal session for this example.

```
% nam
A Test Version of the NBS Network Access Machine

:multics Watkins
Multics 33.3: MIT, Cambridge, Mass.
Load = 37.0 out of 85.0 units: users = 37
Password:
:list
Segments = 9, Lengths = 6.

mul
mul3
mul2
queen
queen.pl1
Watkins.mbx
start_up.ec
Watkins.profile
watkins.con_msgs

r 1248.7 $

:append mul mul2
File mul appended to mul2.


:delete mul
File mul deleted.


:connnbsl0 510/46
.JOB 22 NBS-10 5.07B NBS 5/IL TTY65
%LGNNOC No operator coverage.
Password:

:listnbsl0 mul

MUL               1  <155>   29-Jul-77    DSKB:    [510,46]


:transfer mul2 nbsl0
File mul2 transferred to nbsl0.


:retrieve mul nbsl0
File mul retrieved from nbsl0.
```

24

```
:rename mul mulnbsl
File mulnbsl created.

:undelete mul
File mul restored.

:compare mul mulnbsl
Files are different.

:delete mul
File mul deleted.

:delete mulnbsl
File mulnbsl deleted.

:leavenbsl0
connection nbsl0 closed

:leave
connection multics closed
```

IV. NETWORK JOB EXECUTION

This section describes the method of providing network job execution capabilities developed in conjunction with the CCL.

IV.1. COMMAND DESCRIPTION

Implementing the RUN command described in section II.1.2 in the same manner as the other commands of the language would be extremely difficult and not within the format constraints of a NAM macro call. In addition, requiring the user to type in such a sequence of complex strings increases the likelihood of error and diverges from the simplicity of commands demonstrated in the rest of the CCL. Therefore, two utility programs were developed for the NAM which, in conjunction with the CCL commands previously described, provide a means for the user to define a network job.

The first utility is called MAKEPROFILE. It is an interactive program that queries the user for information about the various systems the user wishes to access. It requests the user names, passwords and account numbers for the systems to be used, and builds a data file called a ´profile´ for the user which is used by the second utility .

The second utility is called RUNMACRO. It also is an interactive program that asks the user to supply all the information needed to specify a network job. For each job step in the network job, the user is asked to give the program name, host names, and all the rest of the data names as described above in section III.1.2. After the user specifies all the steps, the program builds a NAM macro called ´RUN´ that is a sequence of all the CCL commands that are required to perform the transitions for each function to be executed. Using the profile, the program supplies all the accounting information required to make connections that the user usally has to supply interactively. After executing the RUNMACRO utility, the user merely executes the RUN macro, and the network job is executed.

IV.2. COMMAND USAGE

The following procedure describes the steps a user takes in order to define a network job.

1. Log onto UNIX.

2. Run the NAM program.

3. Enter the accounts and passwords of the systems to be used by running the 'MAKEPROFILE' program. Enter 'MAKEPROFILE' in response to the prompt character and answer the queries. (Note: The identifier required for the NBS10 is the project-programmer number pair separated by a slash: e.g., 510/46).

4. Define the sequence of job steps to be executed in the network job. For each job step the user must specify the name and residence of the program to be executed, where it is to be executed, which input files to transfer, and which output files to transfer by answering the queries of the 'RUNMACRO' program. The output of this program is a NAM macro that will perform the desired functions.

5. Enter

```
        RUNMACRO
```

In response to the prompt "Enter Program Specifications:" the user enters

```
        pprog@host1@host2
```

where pprog is the name of the program to be run, host1 is the name of the system where the program is to be executed, and host2 is the name of the system where the program is currently located if it is not already at host1. If host1 is the same as host2, enter 'pprog@host1'.

If there are no more job steps to be defined, the user enters a carriage return.

The utility will then query the user for the logical names of the input files, that is, the names by which the program knows the input files. Then the utility will ask for the actual names for the data files and the hosts at which they reside. If the file exists at the execution site, enter a carriage return in response to the host name. If there are no more input files, enter a carriage return in response to the file name. After obtaining the names of the input files, the names and sites for the output files will be obtained, using the same philosophy for entering the responses.

6. After execution of the 'RUNMACRO' program, there exists a macro called 'RUN' in the user's directory. The user enters 'RUN' and the macro will be executed.

IV.3. SAMPLE NETWORK JOB EXECUTION PROCEDURE

The user wishes to execute the program TEST on the NBS10. The source for TEST resides on UNIX451. There are three input files: FILE1 located at BBN, FILE2 located at NBS10 and FILE3 located at ISI. They are to be given the logical names IN1, IN2 and IN3 respectively. The program produces three output files: OUT1, OUT2 and OUT3. They are to be renamed to DATA1, DATA2 and DATA3 respectively and moved to other sites as follows: DATA1 goes to UNIX451, DATA2 goes to BBN and DATA3 will remain at the NBS10. There will be only one Network Job step in this , so only one set of specifications will be entered. (Note: The user's responses are to the right of the ':'. Responses for password entries for the MAKEPROFILE program are not echoed. All other entries that are not seen are responses of carriage-return to indicate no response.)

The following dialogue will transpire:

```
%nam
:makeprofile
Input Identifier at BBN:sww
Input Identifier at ISI:sww
Input Identifier at NBS10:510/46
Input Identifier at UNIX451:fitz
Input Password at BBN:
Input Password at ISI:
Input Password at NBS10:
Input Password at UNIX451:
Input Account at BBN:333
Input Account at ISI:40
Input Account at NBS10:
Input Account at UNIX451:

:runmacro
Enter Program Specifications : TEST@NBS10@UNIX451
Remote file access? (y or n) n
Logical Name for Input File 1: IN1
Actual Name for Input File 1: FILE1
HOST 1 = BBN
Logical Name for Input File 2: IN2
```

```
Actual Name for Input File 2: FILE2
HOST 2 =
Logical Name for Input File 3: IN3
Actual Name for Input File 3: FILE3
HOST 3 = ISI
Logical Name for Input File 4:
Logical Name for Output File 1: OUT1
Actual Name for Output File 1: DATA1
HOST 1 = UNIX451
Logical Name for Output File 2: OUT2
Actual Name for Output File2: DATA2
HOST 2 = BBN
Logical Name for Output File 3: OUT3
Actual Name for Output File 3: DATA3
HOST 3 =
OUT4 =
Enter Program Specifications:
:run
```

The "RUN" macro produced is the following. The user password supplied from the profile is indicated by <PASSWD>.

```
NBS10 510/46 <PASSWD>
UNIX451 FITZ <PASSWD>
TRANSFER TEST NBS10
LEAVEUNIX451
CONNBBN SWW <PASSWD>
RETRIEVE FILE1
LEAVEBBN
RENAME FILE1 IN1
RENAME FILE2 IN2
CONNISI
RETRIEVE FILE3
LEAVEISI
RENAME FILE3 IN3
EX TEST
CONNUNIX451 FITZ <PASSWD>
RENAME OUT1 DATA1
TRANSFER DATA1 UNIX451
LEAVEUNIX451
CONNBBN SWW <PASSWD>
RENAME OUT2 DATA2
TRANSFER DATA2 BBN
LEAVE BBN
RENAME OUT3 DATA3
LEAVE
```

V. CONCLUSIONS

The Common Command Language for File Manipulation and Network Job Execution was developed to facilitate user interaction with the various systems of a network. As presently implemented, it enables a user to perform file manipulations on four different system types after learning only one simple language. In addition, the language provides the ability to gain access to a system , to identify a user to that system and to easily transfer files between systems without learning complicated protocols.

Work is currently underway to combine the CCL with a Networkwide Directory (NWD) to develop the idea of a "network user." A network user will be connected to the different systems on the network through the NAM and be able to access the files resident on the different hosts the same way a single system user can access files resident on diffent devices. Address resolution of the files will be done through the NAM using the NWD and will be transparent to the user. Incorporated within this system will be a record level access capability for files on remote systems along with facilities to control user access to those files.

APPENDIX 1: THE COMMAND LANGUAGE

Connection Establishing Commands

```
BBN <IDENTIFIER>
ISI <IDENTIFIER>
MULTICS <IDENTIFIER>
NBS10 <IDENTIFIER>
UNIX451 <IDENTIFIER>
```

File Handling Commands

```
APPEND FILE1 FILE2
COMPARE FILE1 FILE2
COPY FILE1 FILE2
CREATE FILE2
DELETE FILE1
ERASE
LIST
RENAME
TYPE
UNDELETE
```

File Transfer Commands

```
RETRIEVE FILE1 HOST
TRANSFER FILE1 HOST
```

Connection Termination Commands

```
LEAVE
```

Network Job Execution

```
Makeprofile Utiltity
Runmacro Utiltity
RUN
```

REFERENCES

[KIM 76-1] Kimbleton, S. R., Final Report, Contract F30603-75-C-0222, Rome Air Development Center, January 1976.

[KIM 76-2] Kimbleton, S. R. and R. L. Mandell, "A Perspective on Network Operating Systems", Procedings of the AFIPS National Computer Conference, vol. 45, 1976, pp. 551-559.

[NBS 77] National Bureau of Standards, "User - Terminal Protocols, ENTRY and EXIT Procedures between Terminal Users and Computer Services, Federal Register, vol. 42, No. 238, pp. 62408 - 62416.

[NEU 75] Neumann, A. J., "Proposed Implementation for Development of User- Terminal Protocols for Computer Network Access, NBSIR 75 - 744, July 1975.

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

[ROS 78] Rosenthal, Robert, and Lucas, Bruce D., The Design and Implementation of the National Bureau of Standards Network Access Machine (NAM), NBS Special Publication Series 500, in preparation.

[UNG 74] Unger, Claus (editor), "Command Languages", Procedings of the IFIP Working Conference on Command Languages, Lund, Sweden, 1974.

[WOOD 78] Wood, Helen M., and Kimbleton, S. R., "Remote Record Access: Requirements, Implementation, and Analysis," in preparation.

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP-500-37	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

COMPUTER SCIENCE & TECHNOLOGY:
Common Command Language for File Manipulation and Network Job Execution: An Example

5. Publication Date

August 1978

6. Performing Organization Code

7. AUTHOR(S)

M. L. Fitzgerald

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)*

13. Type of Report & Period Covered

Special Publication

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 78-600069

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.)*

Computer networks provide the capability for sharing reso rces across many diverse computer systems. Utilizing this capability is inhibited by the requirement that the user become familiar with all the varied command languages and protocols of each accessed system. This report presents a general approach to solving this problem using an intermediary system to support a set of Common Commands for File Manipulation and Network Job Execution. To show the feasibility of this approach, common commands were implemented for four systems using the NBS Network Access Machine.

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 language; computer network; file manipulation, macros; network access; network job execution; network operating systems.

18. AVAILABILITY [X] Unlimited

[ ] For Official Distribution. Do Not Release to NTIS

[X] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C03-003-

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

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PAGES
UNCLASSIFIED	32
20. SECURITY CLASS (THIS PAGE)	22. Price $1.50
UNCLASSIFIED	

USCOMM-DC 29042-P74

[illegible]