KEV 6-DEC-74 15:30 24688

operating system interface for nsw

this is the most recent version of this document, please read it and let me know of any problems, and/or deficiencies, that you may see.

operating system interface for nsw

A process is an instance of a program with the following characteristics: 1

it is running on one physical machine 1a

it is running under the operating system of that machine 1b

it has one program counter associated with it 1c

it has its own logical address space 1d

it has its own state information 1e

it consists of a collection of procedures and data structures 1f

Thus the environment that a process sees consists of the physical machine that it is running on and the operating system under which it is running. We will call this environment the System Machine (SM). 2

The environment that a programmer sees when writing a program in machine (or assembly) language is usually just the SM. However, when a programmer writes in a higher level language, such as COBOL, PL/1, FORTRAN, or L10, the environment that the programmer then sees consists of the SM and the runtime support for the language being used. Thus, such a programmer sees a COBOL machine, or a FORTRAN machine, etc. We will call this environment the Language Machine (LM). 3

Application programs, written for the LM, then provide a new environment, e.g. an NLS Machine, for their users. 4

Thus, we have the following hierarchy of environments: 5

the physical machine environment at the bottom, 5a

next the system machine environment, consisting of the physical machine and the operating system, 5b

next the language machine environment, consisting of the SM and the runtime support package for the higher level languages, and finally 5c

(perhaps several levels of) application environments, 5d

Since we are concerned here with the writing of applications programs, and thus with providing an applications environment we now can arbitrarily divide the collection of procedures and data structures that make up a process into the following two catagories: 6

those procedures and data structures written for the LM 6a

those procedures and data structures written for the SM 6b

For example (using L10 as our language, and TENEX as our operating system), the statement a _ a+b is a statement written for the L10 LM and should work on any L10 language machine; on the other hand, the statement !gtjfn(a, b) is a statement written for the TENEX SM. 7

We will call the collection of procedures and data structures that are written for the SM (as opposed to those for the LM), the Operating System Interface, OSI. 8

If we have a running applications program (e.g. NLS) written in a higher level language (e.g. L10) that supports an application environment on one SM (e.g. an NLS environment for a PDP-10 running TENEX), and we now wish to provide this same application environment on a different SM (e.g. an NLS environment for a Honeywell-6080 running MULTICS), all that is required is that we examine and possible recode the procedures and data structures that make up the OSI. All the LM procedures and data structures should work as written with only a recompilation required to produce object code for the new SM. 9

Procedures fall into logical groupings by virtue of their functions. Each such group will be called a package. (This is the same terminolgy used by Jim White for PCP (see -- xxx).) Most of the following packages will, for efficiencies sake, live within the process that uses the package's procedures. However, this need not be the case and the procedures could in fact be invoked via PCP. In fact, the only procedures that need to exist in the process, in addition to those procedures implementing the applications environment, are those procedures that implement the LM (e.g., the L10 runtime package) and the procedures that implement PCP. 10

The following packages (in addition to those procedures implementing the application environment) are likely to exist in an NSW environment: 11

a LM runtime package for the implementation of the LM, 11a

a Procedure Call Protocol Package (see -- xxx), 11b

a File System package for the manipulation of data structures external to a process, 11c

an Environmental Control Package for the manipulation of other processes (see -- xxx), 11d

a Debugging Package for the debugging of programs, and 11e

a Miscellaneous Support Package containing procedures that don't fall into any of the above packages, but are of general use to a number of different applications. 11f

Once again, within any package, the procedures can be divided into two groups: 12

a set of procedures written for the LM, and 12a

a set of procedures written for the SM. 12b

For example, in the File System package, there may be a LM procedure to delete a file given a pathname for the file. This LM procedure would then decide if the file concerned exists in the local file system or in some remote (other machine) file system. If the file exists in the local file system, the LM procedure would then call an OSI (SM) procedure to delete the file. If the file exists in a remote file system, the LM procedure would then issue the appropriate PCP call on a remote procedure in the remote machine to delete the file. 13

The rest of this document will be concerned with specifying the procedures and data structures that make up the File System package and the Miscellaneous support package. 14

It is not expected that any one process will contain all of the following packages. A process will contain only those packages that it needs. However, if a process requires one of the functions provided for by one of the following procedures, it is expected to use this procedure and NOT to write one of its own. 15

The procedures described below are intended to be the set of primitives that are needed by NLS in an NSW environment. No attempt is being made at this time to provide a complete and exhaustive set of primitives that can be used by any tool at any time in the future. However, it is intended that this set will be a subset of some final package. 15a

Each procedure will be described using PCP syntax. 16

Introduction to the File System 17

Files 17a

As far as NLS is concerned, a file is just a PCP data structure of a specific USE-TYPE (see -- xxx). The following are the particular USE-TYPEs that we will be concerned with here: 17a1

NLS files 17a1a

An NLS file is a tree structured data structure. This tree has certain global or file attributes associted with it, e.g. the creation time of the file, who last wrote the file, etc. In addition, each node of the tree has he following information associated with it: 17a1a1

structural information which reflects this node's position in the tree, 17a1a1a

statement attributes such as the name of this statement and "information retreival keys", 17a1a1b

implementation specific overhead, and 17a1a1c

properties such as the text property of this node or the graphics property of this node, 17a1a1d

All properties have associated with them the following information: 17a1a2

the name of this property, 17a1a2a

the size of this property, 17a1a2b

implementation specific overhead, and 17a1a2c

the property information itself (text, picture, etc.). 17a1a2d

We will provide primities for the allocation, deletion, and manipulation of these nodes and properties. When an NLS file is first initiallized, a process can indicate how many, and the size of the "information retreival keys" are to be associated with each node. The number and size of these "keys" then become non-alterable attributes of the file. 17a1a3

There exists a well defined algorithm ffor converting this tree structure into a PCP list of the following format (note that the structural information is imbedded in the syntax of the list): 17a1a4

```
<%filename% STRING>
   LIST( %file-attributes% LIST( any, ... ), branch )
branch = <%statement-id% INTEGER>
   LIST( stmnt-att, property, ... [, branch, ...] )
stmnt-att = LIST( %statement attribute% any, ... )
```

property = LIST(prprty-att, %property% any)
prpty-att = LIST(%property attribute% any, ...) 17a1a4a

Sequential files 17a1b

A sequential file is simply a PCP list. The KEY for this list is the file name (pathname) that uniquely identifies the file. The first element of this list is a list that contains certain attributes of the file (e.g. the bit length of individual bytes); succesive elements of the list are the data bytes of the file. 17a1b1

%filename% STRING: LIST(%attributes% LIST(), %bytes% BITSTR, ...) 17a1b1a

Terminal files 17a1c

A terminal file is merely a sequential file with certain attributes and with specific semantics applied to the data bytes. 17a1c1

We will be providing procedures for manipulating the data contained in each of these types of files. While any procedure can be called and passed a file handle of any type of file, this will not always provide meaningful results (e.g. allocating a window in an NLS file). Such mismatches of functions and file types will: 17a2

be simulated as best as possible, or 17a2a

be a slow NOP, or 17a2b

generate an error 17a2c

It should be noted that we are talking about logical files and not the physical devices that the data may actually reside on. 17a3

For example, a terminal file (of which a typescript file is an example) could very well live on a disk. In this case, there would be semantics associated with the data in the file such that if the disk terminal file is ever copied to a physical terminal, it would appear to the user just as if it had been created originally for the terminal. 17a3a

Filenames 17b

Whenever we talk about a filename, we will be referring to a string whose value will be some canonical form that represents the full pathname that describes the file. This canonical form

would probably ideally be the current NLS link syntax, however there are known deficiencies in this form (e.g. the inability to specify devices). We may have to use some form similar to that proposed in RFC645, although my personal feeling is that this form is not a "nice" form to present to users. Of course, it is not neccessary that the form presented to users be the same form that is used internally, but it seems desirable to me to try to use the same format internally and externally. 17b1

In the primitives described below, we will be using the following terms, with the following meanings: 17b2

pathname: a string which fully describes the location of the file, 17b2a

file-id: an integer which is local to the process and is a shorthand way of referring to a pathname, and 17b2b

file-handle: a pathname or a file-id. 17b2c

More on Terminal Files 17c

When we talk of manipulating a terminal, in fact what we do is issue a procedure call, to a procedure in the File System package, requesting that I/O be performed to a terminal file. The called procedure will then decide if the terminal file concerned is currently residing on a physical terminal, and if so issue the appropriate OSI procedure calls (which in turn issue the appropriate operating system calls) to actually manipulate the terminal. If the concerned terminal file is not currently residing on a physical terminal, then the called procedure simulates the requested function as best as possible, e.g. if the terminal file is currently residing on a disk, then the appropriate data bytes will be written into the file so that the terminal actions can be reproduced later. 17c1

When thinking about terminals, we see that they serve the following two basic functions: 17c2

accepting input from the user, and 17c2a

presenting information to the user. 17c2b

On any given terminal, there is a display space available for the presentation of information to a user. (On a typewriter-like terminal, this corresponds to one line; on displays this corresponds to the display screen.) This total display space can be subdivided into informationally functional "windows", and windows can be subdivided into primitive units

for the presentation of atomic pieces of information (e.g. strings, pictures). (Note that atoms are made up of elementary particles such as characters or lines, etc.) 17c3

The functional windows that are likely to exist for an NSW terminal are: 17c4

a status information window, called the status window, 17c4a

an error information window, called the error window, 17c4b

a command feedback and prompting information window, called the command window, and 17c4c

a tool specific information window, called the tool window. 17c4d

In an ideal world, where all NSW users have a super-sophisticated full graphics display terminal with an infinetely large display area, we could afford to reserve unique areas of the display space for each of the above windows. Getting back to earth, however, we are faced with the problem of mapping these logical areas into physical areas on a variety of different physical terminals, including typewriter-like devices. 17c5

One approach to this problem is to make NSW processes aware of the myriad terminals and the ideosyncrocies of each, and to tailor its interaction appropriately for each terminal. This would seem to be a never ending task (and perhaps counter-productive in that there might not be a "standard" interface anymore). 17c5a

Another approach is to provide an NSW Virtual Terminal (NSWVT) and an interface to this logical terminal. 17c5b

A third approach is to provide a small set of NSW virtual terminals and interfaces to each of these logical terminals. Each virtual terminal is then representative of a class of physical terminals with similar characteristics. 17c5c

With approaches 2 and 3 it is then the function of the OSI procedures (in conjunction with operating system calls) to map these logical terminals into physical devices. These approaches have the advantage that supporting a new terminal only requires the addition of a new "device" driver, a much simpler conceptual task compared with changing code within the NSW. 17c5d

NSW will use approach 3, i.e. a small set of NSW virtual

terminals will exist, and we will be specifying the primitives for manipulating these virtual terminals. The classes of NSW virtual terminals will be: 17c5e

Class 0: ascii, half-duplex, line at a time, typewriter-like terminals, 17c5e1

Class 1: ascii, half-duplex, character at a time, typewriter-like terminals, 17c5e2

Class 2: ascii, full-duplex, character at a time, typewriter-like terminals, 17c5e3

Class 3: ascii, alpha-numeric displays, with editing capabilities, 17c5e4

Class 4: terminals that support our line-processor protocol, and 17c5e5

Class 5: sophisticated graphics terminals (these will not be supported in the first year of the NSW). 17c5e6

As mentioned above, windows can be subdivided into atoms such as strings or pictures. When outputting to a terminal, it is these atoms that are actually manipulated (written, deleted, etc.). 17c6

Window names (window-id) are local to a file and atom names (atom-id) are local to a window. Thus when manipulating a terminal it is necessary to give the full path name to describe the actual atom concerned: 17c7

atom-pathname ::= file-id, window-id, atom-id. 17c7a

There are three basic types of windows: 17c8

sequential windows, 17c8a

random windows, and 17c8b

cursor windows. 17c8c

A sequential window, consists of one and only one string, and is a window which simulates an alpha-numeric display: 17c9

characters written to the window are always appended to the end of the window, 17c9a

the characters linefeed, carriage-return, formfeed (which

clears the entire window), backspace-character, backspace-word, and tab are simulated appropriately, 17c9b

writing after the last character position in an individual line causes automatic line overflow, 17c9c

writing after the last character position in a window causes automatic line overflow and causes device dependent scrolling. 17c9d

A facility similar to the TENEX scope terminal will exist for sequential windows (user input sets a count to zero and succesive output to the window increments this count; when scrolling would cause information to leave the window if the count is non-zero, then a pause occurs until the user inputs a (special?) character and the count is reset to zero.) 17c9d1

A facility will also exist so that the user can scroll back and forth through n lines of information that has passed through the window. 17c9d2

A random window, on the other hand, can contain as many atoms as desired by the process. The atoms are allocated, and deleted, etc. under program control. 17c10

A cursor window consists of one and only one atom. The bounds of this window are allowed to change dynamically under program control. The contents of the one atom are also allowed to change dynamically under program control. 17c11

All windows have a process assignable priority associated with them. If two atoms overlap each other, then the atom belonging to the lower priority window will be made invisible. If two atoms with equal priority overlap, then the results are unspecified. 17c12

Now lets take a look at the input functions served by a terminal. A terminal can have a number of discrete devices associated with it, e.g., an alphanumeric keyboard, a mouse, etc. For the purposes of this document (and for the first year of the NSW) we will be concerned with only the following devices: 17c13

an ascii alphanumeric keyboard, 17c13a

(The NLS keyset can and will be thought of as just an extension of the ascii alphanumeric keyboard and input from the keyset will consist of any of 31 ascii

characters. There will be a well defined character set for each chord.) 17c13a1

a pointing device (such as a mouse or cursor buttons on a keyboard), and 17c13b

a binary switch device, such as the buttons on a mouse. 17c13c

A process is then interested in knowing when the state of any of these devices changes, and when such a change takes place is interested in knowing the state of one or more of these devices. Primitives are provided for specifying which state changes are of interest to the process and the "report form" desired when such a change takes place. 17c14

All input from terminal files will be 8-bits wide. The first 128 codes will be used to represent the 7-bit ascii code, the second 128 codes will be used for reporting the states of other devices. 17c14a

(A note on the pointing device: there will exist a short circuit between the input pointing device and the output function of displaying the cursor to the user. This shortcut consists of the cursor window coordinates for the input pointing device being automatically updated.) 17c15

Some Assumptions about the Initial State 17d

When a process is first created, with regard to the state of its terminal files, it will be in what we we call the initial state. When a process is first created it will have file-ids for the following two open terminal files: 17d1

a "Primary Input File" - PIF, with file-id PIFID, and 17d1a

a "Primary Output File" - POF, with file-id POFID. 17d1b

The POF will have the following two windows defined (and the handles of these windows will be known): 17d2

a cursor window, known as the default cursor window. This is the window that will be used for tracking the input pointing device. The single atom for this window will consist of a pre-determined string or picture, and 17d2a

a sequential window, known as the default TTY window (DTW). The bounds of this window will correspond to the bounds available to the process for the presentation of information to the user. 17d2b

The PIF will be set to report only changes in state of the ascii alphanumeric keyboard. 17d3

We expect that the NSW frontend process will do the following as part of its intialization code: 17d4

find out which class of physical terminals the PIF and POF are living on, and as appropriate to this class of terminals 17d4a

make the DTW invisible, 17d4b

allocate an error window, 17d4c

allocate a status window, 17d4d

allocate a command window, 17d4e

allocate a sequential window as the tool window, and 17d4f

set up the PIF for the type of input it desires. 17d4g

The POF that exists for an NSW backend process will have as its DTW the tool window that was created by the frontend. Primitives are provided for finding out the bounds of this window, and the backend can manipulate this window (subdivide it, make it invisible and allocate random windows that are within with its bounds, etc.) to its heart's content. 17d5

The NSW backend process will not be allowed to directly manipulate the error, status, and command windows (or allocate windows that overlap the command window or error windows), but there are primitives provided for displaying error and status messages. 17d5a

The File Manipulation Package - FMP 18

General File manipulation Procedures 18a

get-file-id(partial-pathname, file-use-type, new-old [,parms] => file-id, full-pathname) 18a1

This procedure is used to obtain a file-id for a file given a partial pathname and other information about the intended use of the file. If a file-id cannot be obtained (e.g. if the partial pathname is not sufficient to uniquely identify a file) an appropriate error will be generated. 18a1a

FORMAT: 18a1b

partial-pathname - STRING 18a1b1

this is the partial pathname to be used to try to uniquely specify a file 18a1b1a

file-use-type - INTEGER [NLS=0 / SEQUENTIAL=1 / TERMINAL=2] 18a1b2

this parameter specifies for which use type of file a file-id is desired 18a1b2a

new-old - INTEGER [NEW=0 / OLD=1] 18a1b3

this parameter specifies whether we are trying to get a file-id for a pre-existing file (OLD file) or if we wish to create a new file (NEW) 18a1b3a

parms - %terminal files% class / %new NLS files% keys 18a1b4

This parameter is optional, and only needs to be specified in the following situations: 18a1b4a

if file-use-type specifies a terminal file and the partial-pathname specified does not refer to an actual physical terminal, If this is the case, then this parameter specifies which terminal class the file is to simulate, 18a1b4a1

if file-use-type specifies an NLS file and new-old specifies NEW, If this is the case, then this parameter specifies the number, and size, of the "information retreival keys" to be associated with the file, 18a1b4a2

class - INTEGER [CLASS0=0 / CLASS1=1 / CLASS2=2 / CLASS3=3 / CLASS4=4 / CLASS5=5] / 18a1b5

keys - LIST(%number-of-keys% INTEGER, %size-key-1% INTEGER, ...) 18a1b6

file-id - INTEGER 18a1b7

This result is the file-id for the specified parameters, 18a1b7a

full-pathname - STRING 18a1b8

This result is the full pathname for the input file,

This result is useful in situations where "file name recognition" is being performed. 18a1b8a

release-file-id(file-id) 18a2

This procedure releases a specific file-id, or all file-ids, from use by this process. 18a2a

FORMAT: 18a2b

file-id = INTEGER [ALL=-1 / file-id] 18a2b1

open-file(file-id, open-type [,bytesize]) 18a3

Before input or output to a file can actually be performed, it is necessary to open the file. This procedure performs that function, and opens a fiel for a specific type of I/O. An error will be generated if the file (to which file-id refers) cannot be opened for the type of I/O requested. 18a3a

FORMAT: 18a3b

file-id = INTEGER 18a3b1

open-type = LIST(INTEGER [READ=0 / WRITE=1 / APPEND=2 / EXECUTE=3], ...) 18a3b2

This parameter specifies what type of access is desired to a file. 18a3b2a

bytesize = INTEGER 18a3b3

This optional parameter is required if the file-id specified refers to a sequential file, and in that case specifies the number of bits in each byte of the file. This parameter will be ignored for file types other than sequential. 18a3b3a

close-file(file-id [,release-parm]) 18a4

This procedure closes the specified file(s) and depending on the value of release-parm may release the file-id(s). After a file is closed, no further I/O can be performed to the file until it is opened. 18a4a

FORMAT: 18a4b

file-id = INTEGER [ALL=-1 / file-id] 18a4b1

release-parm - INTEGER [RELEASE=0 / KEEp=1] 18a4b2

This optional parameter specifies whether or not it is desired to keep the file-id after the appropriate file(s) have been closed. If this parameter is not specified, it defaults to RELEASE. 18a4b2a

delete-file(file-id) 18a5

This procedure deletes the specified file and releases the file-id for the file. An error will be generated if the file cannot be deleted. 18a5a

FORMAT: 18a5b

file-id - INTEGER 18a5b1

copy-file(file-id-1, file-id-2 [,copy-parms]) 18a6

This procedure copies the file specified by file-id-1 to the file specified by file-id-2. If file-id-1 and file-id-2 refer to different use types of files, then the necessary format conversions will be performed in the process of copying. These format conversions can be modified by the value of copy-parms. If file-id-2 refers to a pre-existing file, then the contents of that old file will be lost forever. 18a6a

FORMAT: 18a6b

file-id-1 - INTEGER 18a6b1

file-id-2 - INTEGER 18a6b2

copy-parms - LIST( ) 18a6b3

The value of this optional parameter is not specified yet. It will be specified in the near future when we learn what is necessary. 18a6b3a

move-file(file-id-1, file-id-2 [,copy-parms]) 18a7

This procedure is a shorthand for the following two procedure calls (it may be deleted if we decide it has limited use): 18a7a

copy-file(file-id-1, file-id-2 [,copy-parms]) 18a7a1

delete-file(file-id-1) 18a7a2

(Note that this is effectively a rename file with the appropriate format conversions being performed in the process of the rename.) 18a7b

FORMAT: 18a7c

file-id-1 - INTEGER 18a7c1

file-id-2 - INTEGER 18a7c2

copy-parms - LIST( ) 18a7c3

The value of this optional parameter is not specified yet. It will be specified in the near future when we learn what is necessary. 18a7c3a

convert-file-id-to-string(file-id, convert-parms -> string) 18a8

This procedure will convert the passed file id into a string that represents the full pathname for the specified file. The format for the output string and which fields of the full pathname to be included are governed by convert-parms. 18a8a

FORMAT: 18a8b

file-id - INTEGER 18a8b1

convert-parms - this paramter will be specified later. 18a8b2

string - STRING 18a8b3

read-file-attribute(file-id, attribute-name -> attribute-value) 18a9

This procedure will read an attribute of a file. 18a9a

FORMAT: 18a9b

file-id - INTEGER 18a9b1

attribute-name - STRING 18a9b2

The attribute names will be specified later when we get a better idea of what we want. 18a9b2a

attribute-value - any 18a9b3

The value and type of this result depend on which attribute is being read. 18a9b3a

write-file-attribute(file-id, attribute-name, attribute-value) 18a10

This procedure changes an attribute of a file. An error will be generated if the attribute is not allowed to be changed. 18a10a

FORMAT: 18a10b

file-id - INTEGER 18a10b1

attribute-name - STRING 18a10b2

The attribute names will be specified later when we get a better idea of what we want. 18a10b2a

attribute-value - any 18a10b3

The value and type of this result depend on which attribute is being written. 18a10b3a

NLS File manipulation Procedures 18b

lock-file(file-id, lock-type, user-id) 18b1

This procedure either locks / unlocks / or relocks (a previously locked and unlocked) file. If the file is already locked, then lock and relock generate errors; if the file is locked then the user-id specified must match the user-id of the locker before the file can be unlocked; if a file is unlocked and a relock is specified then the user-id specified must match the user-id of the person who had the file previously locked before it was unlocked. 18b1a

FORMAT: 18b1b

file-id - INTEGER 18b1b1

lock-type - INTEGER [LOCK=0 / UNLOCK=1 / RELOCK=2] 18b1b2

user-id - STRING 18b1b3

This parameter is the NSW user ident of the user who wishes to lock/unlock/relock the file. 18b1b3a

update-file(file-id, user-id, update-parms [,new-file-id]) 18b2

This procedure will update a locked file as modified by the update-parms parameter. If the file is not locked this is a

nop; if the file is locked then the user-id specified must match the user-id of the person who has the file locked or an error will be generated. 18b2a

FORMAT: 18b2b

file-id - INTEGER 18b2b1

user-id - STRING 18b2b2

This parameter is the NSW user ident of the user who wishes to update the file. 18b2b2a

update-parms - INTEGER [NORMAL=0 / COMPACT=1] 18b2b3

new-file-id - INTEGER 18b2b4

If this optional parameter is specified then the file specified will be updated to the file specified by new-file-id; if this parameter is not specified, then the file will be "updated in place". If this parameter is specified and it refers to an old file, then the contents of that old file will be lost; if this parameter is not specified, then the contents of the file specified by file-id will be lost and will be replaced by the value of the updated file. 18b2b4a

For all the following NLS file manipulation primitives the following formats apply: 18b3

file-id - INTEGER 18b3a

node-id - INTEGER 18b3b

this applies to all node-ids such as down-id, up-id, etc. 18b3b1

property-id - ADDRESS 18b3c

this is the address of the property itself and not the address of file system maintained fields such as the length of the property. 18b3c1

property-name - INTEGER 18b3d

key-number - INTEGER 18b3e

key-value - INTEGER 18b3f

initialize-file(file-id, user-id, keys) 18b4

This procedure enables a process to completely initialize a file. It is equivalent to getting a file-id for a new NLS file. 18b4a

FORMAT: 18b4b

user-id - STRING 18b4b1

This parameter is the NSW user ident of the user who wishes to update the file. 18b4b1a

keys - LIST(%number-of-keys% INTEGER, %size-key-1% INTEGER, ...) 18b4b2

allocate-node(file-id -> node-id) 18b5

This procedure will allocate a new node. The only filled in field in the node wil be "key-0", which is a file system maintained key. 18b5a

delete-node(file-id, node-id, subs) 18b6

This procedure will delete a node, and any properties belonging to this node, and if subs is TRUE then any nodes (and associated properties) under this node in the tree. If subs is FALSE and the passed node has substructure, then an error will be generated. If this node is linked in a tree, it will be unlinked from the tree before it is deleted. 18b6a

FORMAT: 18b6b

subs - BOOLEAN 18b6b1

undelete-node(file-id, node-id) 18b7

This procedure will undelete a node. 18b7a

allocate-property(file-id, node-id, property-name, property-size -> property-id) 18b8

This procedure will allocate a property with the specified name and size and associate it with the passed node. If a property already exists for this node with the same name, then as much as will fit of the old property will be copied into the new property and the old property will be irretreivablly deleted. 18b8a

delete-property(file-id, node-id, property-name) 18b9

This procedure will delete a the specified property. 18b9a

undelete-property(file-id, node-id, property-name) 18b10

This procedure will undelete the specified property. 18b10a

The following procedures are used for moving around in the tree structure. They will fail if for example get-down-id is called and the passed node has no down node, etc. 18b11

get-successor-id(file-id, node-id -> successor-id) 18b11a

get-predecessor-id(file-id, node-id -> predecessor-id) 18b11b

get-up-id(file-id, node-id -> up-id) 18b11c

get-down-id(file-id, node-id -> down-id) 18b11d

get-tail-id(file-id, node-id -> tail-id) 18b11e

get-head-id(file-id, node-id -> head-id) 18b11f

get-end-id(file-id, node-id -> end-id) 18b11g

get-origin-id(file-id, node-id -> origin-id) 18b11h

get-next-id(file-id, node-id -> next-id) 18b11i

get-back-id(file-id, node-id -> back-id) 18b11j

link-node(file-id, node-id, relative-node, relationship) 18b12

This procedure will link the passed node into the tree specified by relative-node as either the down or successor of relative-node as specified by relationship. 18b12a

FORMAT: 18b12b

realtive-node = INTEGER 18b12b1

relationship = INTEGER [DOWN=0 / SUCCESSOR=1] 18b12b2

this parameter specifies the relation that the new node (node-id) is to bear to the old node (relative-node). 18b12b2a

unlink-node(file-id, node-id) 18b13

This procedure will remove the passed node from the tree in

which it exists. The proper fixup will occur for brothers of this node. 18b13a

read-key(file-id, node-id, key-number => key-value) 18b14

This procedure will read the value of the indicated key. Note that key-0 is a file system maintained key and is guaranteed to be unique for each node within a file. (This is the sid of the statement in NLS parlance.) 18b14a

write-key(file-id, node-id, key-number, key-value) 18b15

This procedure will change the value of the specified key. (Note that key-number must be greater than 0.) 18b15a

find-key(file-id, node-id, key-number, key-value, type) 18b16

This procedure will find the node that contains as the value of its key-number key the value key-value. The search will start after node node-id and either proceed by following the tree structure if type is STRUCTURE, or in a file system dependent order if type is ANY. 18b16a

FORMAT: 18b16b

type = INTEGER [ANY=0 / STRUCTURE=1] 18b16b1

get-property-id(file-id, node-id, property-name => property-id) 18b17

This procedure will return the property-id for the named property at the passed node. It will fail if this node does not have the named property. 18b17a

get-property-length(file-id, property-id => length) 18b18

This procedure will get the length of the passed property. 18b18a

get-property's-node(file-id, property-id => node-id) 18b19

This procedure will return the node-id for the passed property-id. 18b19a

get-next-property(file-id, parm => property-id) 18b20

This procedure will get either the first property associated with a given node, or the next property given a property. 18b20a

FORMAT: 18b20b

parm = LIST(BOOLEAN[=TRUE], node-id) /
LIST(BOOLEAN[=FALSE], property-id) 18b20b1

Sequential File manipulation Procedures 18c

read-byte(file-id [,byte-name] => byte-value) 18c1

This procedure will read an individual byte from the specified file. 18c1a

FORMAT: 18c1b

file-id = INTEGER 18c1b1

byte-name = INTEGER / STRING 18c1b2

This optional parameter specifies which byte is to be read from the file. If the bytes in the file have string keys, i.e. names, associated with them, then it is possible to read them by specifying the byte's name. If this parameter is not specified, then it defaults to be the current byte position in the file, which is an attribute of the file. Reading (or writing) a byte in a file advances the current character position attribute of the file by one byte for each byte read (or written). 18c1b2a

byte-value = BITSTR 18c1b3

This result is the value of the selected byte. 18c1b3a

write-byte(file-id, byte-value [,byte-name]) 18c2

This procedure will write an individual byte in the specified file. 18c2a

FORMAT: 18c2b

file-id = INTEGER 18c2b1

byte-value = BITSTR 18c2b2

This parameter specifies the new value for the selected byte. 18c2b2a

byte-name = INTEGER / STRING 18c2b3

This optional parameter specifies which byte is to be written in the file. If the bytes in the file have

string keys, i.e. names, associated with them, then it is possible to write them by specifying the byte's name. If this parameter is not specified, then it defaults to be the current byte position in the file, which is an attribute of the file. 18c2b3a

read-string(file-id, termination-condition
[,starting-byte-name] -> string-value) 18c3

This procedure will read a number of successive bytes from the specified file. 18c3a

FORMAT: 18c3b

file-id = INTEGER 18c3b1

termination-condition = LIST(%count%INTEGER, %chars%STRING) 18c3b2

This parameter specifies that either count bytes are to be read if chars is the null string; or that bytes are to be read until one of the bytes matches any byte in the chars string if a zero count is specified; or read bytes until a byte is encountered that matches any of the bytes in the chars string if both count and chars are specified. 18c3b2a

starting-byte-name = INTEGER / STRING 18c3b3

This optional parameter specifies which byte is to be the first byte read from the file. If the bytes in the file have string keys, i.e. names, associated with them, then it is possible to read them by specifying the byte's name. If this parameter is not specified, then it defaults to be the current byte position in the file, which is an attribute of the file. 18c3b3a

string-value = STRING 18c3b4

This result is the value of the selected bytes. 18c3b4a

write-string(file-id, termination-condition, string-value
[,starting-byte-name]) 18c4

This procedure will write a number of successive bytes in the specified file. 18c4a

FORMAT: 18c4b

file-id - INTEGER 18c4b1

temination-condition - LIST(%count%INTEGER, %chars%STRING) 18c4b2

This parameter specifies that either count bytes are to be written if chars is the null string; or that bytes are to be written until one of the new bytes matches any byte in the chars string if a zero count is specified; or write bytes until a byte is encountered in the new string that matches any of the bytes in the chars string if both count and chars are specified. 18c4b2a

string-value - STRING 18c4b3

This is the value for the new string. 18c4b3a

starting-byte-name - INTEGER / STRING 18c4b4

Terminal File manipulation Procedures 18d

Pseudo Interrupts 18d1

A pseudo-interrup normally occurs when the user inputs the appropriate character at his/her terminal. However, in the case of class 0 (half-duplex, line at a time) terminals, the pseudo-interrupt cannot occur until the user has transmitted the entire line and the character is read. 18d1a

(Eventually, it might be nice to have pseudo-interrupts associaed with non-terminal files. However, in the initial NSW implementation, if a file-id is specified which refers to a non-terminal file, an error will be generated.) 18d1b

Enable-Pseudo-Interrupts(file-id) 18d1c

This procedure turns on the pseudo-interrupt system for the specified file. Individual characters can be activated and deactivated independently of the enabled status of the pseudo-interrupt system for the file. However, until the PSI system is enabled for a file, any activated characters will NOT generate pseudo-interrupts and in fact will be dealt with as normal input. 18d1c1

FORMAT: 18d1c2

file-id - INTEGER 18d1c2a

Disable-Pseudo-Interrupts(file-id) 18d1d

This procedure disables (turns off) the PSI system for th specified file. 18d1d1

FORMAT: 18d1d2

file-id - INTEGER 18d1d2a

Read-file-psi-status(file-id -> file-psi-status) 18d1e

This procedure reads the current status of the PSI system for the specified file (i.e. is it on or off), and returns a list of the activated character psi-char-ids. 18d1e1

FORMAT: 18d1e2

file-id - INTEGER 18d1e2a

file-psi-status - LIST(%status% INTEGER [ON=1 / OFF=0], %psi-char-id% INTEGER, ...) 18d1e2b

Activate-char-as-PSI(file-id, char, priority, proc-name -> psi-char-id) 18d1f

This procedure specifies that a character should generate a pseudo-interrupt if the PSI system is enabled. It also specifies the priority of the pseudo-interrupt to be associated with the character and a procedure to be called when the PSI is generated. 18d1f1

FORMAT: 18d1f2

file-id - INTEGER 18d1f2a

char - BITSTR 18d1f2b

this is an 8-bit character 18d1f2b1

priority - INTEGER [P0=0 / P1=1 / P2=2 / P3=3] 18d1f2c

This is the priority to be associated with the PSI for this character. If a pseudo-interrupt of priority n is in progress, then only PSIs of higher priority m (m<n) will be allowed to occur. Any PSIs of equal or lower priority (m>=n) will be remembered and will take place when the current PSI is "debreaked". 18d1f2c1

proc-name - ADDRESS 18d1f2d

This parameter specifies the address of a local procedure to be called when the PSI is generated, (Note that this assumes knowledge of the language that the called procedure is written in so that the proper type of procedure call can be made, Initially in the NSW, we will assume this language to be L10; eventually, this may have to be replaced with an address to receive control rather than a procedure to be called.) 18d1f2d1

psi-char-id - INTEGER 18d1f2e

This result is an id that is to be used for future references to this PSI. 18d1f2e1

Deactivate-char-as-PSI(Psi-char-id) 18d1g

This procedure deactivates the PSI associated with the specified psi-char-id. 18d1g1

FORMAT: 18d1g2

psi-char-id - INTEGER 18d1g2a

Read-char-psi-status(psi-char-id -> char, priority, proc-name) 18d1h

This procedure returns the current status of the PSI associated with the specified psi-char-id. 18d1h1

FORMAT: 18d1h2

psi-char-id - INTEGER 18d1h2a

char - BITSTR 18d1h2b

This result is the 8-bit character associated with the specified psi-char-id 18d1h2b1

priority - INTEGER 18d1h2c

This result is the priority associated with the specified psi-char-id 18d1h2c1

proc-name - ADDRESS 18d1h2d

This result is the address of the local procedure

that will be call if the specified PSI is generated. 18d1h2d1

Change-PSI-Return-Address(return-address) 18d1i

Normally a procedure that is called as a result of a PSI character performs a normal return and then the PSI system returns control to the instruction that was interrupted. This procedure gives a programmer the ability to "debreak" from a PSI routine (e.g. to allow lower or equal priority pseudo-interrupts to occur) and yet have control over where execution will resume. This procedure will generate an error if it is called when no pseudo-interrupts are in progress. 18d1i1

FORMAT: 18d1i2

return-address - ADDRESS 18d1i2a

Echo control 18d2

Initially, the first 128 character codes will be echoed directly without any transformations applied and none of the second 128 codes will be echoed. The following two procedures are provided to modify this initial state. 18d2a

write-echo-status-file(file-id, echo-status) 18d2b

This procedure allows a programmer to specify how characters will be echoed when they are typed at a terminal. (Note that specifying anything but no echoing for class 0 (line at a time) terminals can lead to ugly things appearing on the terminal.) 18d2b1

FORMAT: 18d2b2

file-id - INTEGER 18d2b2a

echo-status - LIST(LIST(class / char, control), ...) 18d2b2b

class - INTEGER [CONTROL=0 / ALPHA=1 / NUM=2 / PUNC=3 / FIRST-128=4 / SECOND-128=5 / ALL=6] 18d2b2c

char - BITSTR 18d2b2d

this is the 8-bit character to which the echo string applies 18d2b2d1

control - STRING 18d2b2e

this is the string that should be echoed when the specified character is input. This can be a null string indicating no echoing; we will have to come up with some meta language to do things such as echoing a class of characters with a mapping of those characters, e.g. echo control-L as <^L>. 18d2b2e1

read-echo-status-file(file-id -> echo-status) 18d2c

This procedure reads the current echo status for the specified file. 18d2c1

FORMAT: 18d2c2

file-id - INTEGER 18d2c2a

echo-status - see above 18d2c2b

Character translation control 18d3

Initially, all 256 character codes will be given to a process exactly as input from a terminal. The following two procedures are provided to modify this initial state. 18d3a

read-input-char-trans(file-id -> input-trans-list) 18d3b

This procedure will read the current status of input translation tables for the specified file. 18d3b1

FORMAT: 18d3b2

file-id - INTEGER 18d3b2a

input-trans-list - see echo-status above 18d3b2b

write-input-char-trans(file-id, input-trans-list) 18d3c

This procedure will write the input translation tables for the specified file. 18d3c1

FORMAT: 18d3c2

file-id - INTEGER 18d3c2a

input-trans-list - see echo-status above 18d3c2b

Primary File Manipulation 18d4

At any time a process has associated with it one and only one primary input file and one and only one primary output file. Normally, and initially, these will correspond to the physical terminal being used by a user. These are the files that are referenced when a process uses the file-ids PIFID and POFID. The following procedures allow processes to divert input or output to or from a physical terminal to other devices, e.g. a disk.file to be read later, or a command file for frequently performed functions. 18d4a

read-primary-file-handles(process-id -> PIF-id, POF-id) 18d4b

This procedure reads the file-ids of the current primary input and output files. If I/O has not been diverted then these will bee PIFID and POFID. 18d4b1

FORMAT: 18d4b2

process-id - INTEGER (see -- xxx,) 18d4b2a

PIF-id - INTEGER 18d4b2b

This will be the file-id for the current primary input file. 18d4b2b1

POF-id - INTEGER 18d4b2c

This will be the file-id for the current primary output file. 18d4b2c1

set-primary-file-handles(process-id, PIF-id, POF-id) 18d4c

This procedure allows a process to change its current primary input and/or output files. If a process does not wish to change its primary input file then it should use the file-id PIFID for the PIF-id parameter and simarily for its primary output file. 18d4c1

FORMAT: 18d4c2

process-id - INTEGER (see -- xxx,) 18d4c2a

PIF-id - INTEGER 18d4c2b

This is the file-id for the new primary input file. 18d4c2b1

POF-id - INTEGER 18d4c2c

This is the file-id for the new primary output file. 18d4c2c1

reset-primary-file-handles(process-id) 18d4d

This procedure resets the primary input and output files for the specified process back to what they were when the process first started up. 18d4d1

FORMAT: 18d4d2

process-id - INTEGER (see -- xxx,) 18d4d2a

Window Manipulation 18d5

An attempt to manipulate windows on class 0, 1, or 2 (non-display) terminals or on non-terminal files will generate an error. 18d5a

allocate-window(file-id, window-parms -> window-id) 18d5b

This procedure allocates a window with the specified characteristics in the specified file. 18d5b1

FORMAT: 18d5b2

file-id - INTEGER 18d5b2a

window-parms - LIST(type, bounds, priority, visibility, hit-sensitivity [,typewriter]) 18d5b2b

type - INTEGER [SEQUENTIAL=0 / RANDOm=1 / CURSOR=2] 18d5b2c

bounds - LIST(%x1% INTEGER, %y1% INTEGER, %x2% INTEGER, %y2% INTEGER) 18d5b2d

this list specifies the coordinates of the lower left and upper right hand corners for the window 18d5b2d1

priority - INTEGER [FLOAT=-1 / P0=0 / P1=1 / P2=2 / P3=3] 18d5b2e

the FLOAT priority says that this window has equal priority with any windows it may be overlapping. (The default cursor window has priority FLOAT.) 18d5b2e1

visibility - INTEGER [VISIBLE=0 / INVISIBLE=1] 18d5b2f

hit-sensitivity - INTEGER [SENSITIVE=0 / INSENSITIVE=1] 18d5b2g

This parameter specifies whether or not the atoms that make up this window are "hit sensitive" to the select primitive below. 18d5b2g1

typewriter - INTEGER [FALSE=0 / TRUE=1] 18d5b2h

IF this optional parameter is true (only valid for sequential windows), then any characters output to the file POFID which are not part of any of the other commands specified below (e.g. part of a write-string command), will be sent to this window. 18d5b2h1

window-id - INTEGER 18d5b2i

This is the id that should be used to reference this window. 18d5b2i1

delete-window(file-id, window-id) 18d5c

This procedure deletes the specified window(s), and any atoms that are belong to the window. 18d5c1

FORMAT: 18d5c2

file-id - INTEGER 18d5c2a

window-id - INTEGER [ALL=-1 / window-id] 18d5c2b

If this parameter specifies ALL windows, then this procedure will delete all the windows, with the exception of the default TTY window and the default cursor window, that have been allocated by this process within the specified file. 18d5c2b1

manipulate-window(file-id, window-id, manipulation-parms) 18d5d

This procedure allows a process to manipulate a window. 18d5d1

FORMAT: 18d5d2

file-id - INTEGER 18d5d2a

window-id - INTEGER 18d5d2b

manipulation-parms - LIST(bounds, priority, visibility, hit-sensitivity, typewriter) 18d5d2c

bounds - LIST(%x1% INTEGER, %y1% INTEGER, %x2% INTEGER, %y2% INTEGER) 18d5d2d

This parameter can only be specified if the window being manipulated is a cursor window. If this is the case, then this list specifies the coordinates of the lower left and upper right hand corners for the window 18d5d2d1

priority - INTEGER [FLOAT=-1 / P0=0 / P1=1 / P2=2 / P3=3] 18d5d2e

visibility - INTEGER [VISIBLE=0 / INVISIBLE=1] 18d5d2f

hit-sensitivity - INTEGER [SENSITIVE=0 / INSENSITIVE=1] 18d5d2g

typewriter - INTEGER [FALSE=0 / TRUE=1] 18d5d2h

read-window-parms(file-id, window-id -> window-parms) 18d5e

This procedure allows a process to read the status of a window. 18d5e1

FORMAT: 18d5e2

file-id - INTEGER 18d5e2a

window-id - INTEGER 18d5e2b

window-parms - LIST() see above 18d5e2c

Atom Manipulation 18d6

allocate-atom(file-id, window-id, atom-parms -> atom-id) 18d6a

This procedure allocates an atom, with the specified characteristics, within the specified window. 18d6a1

FORMAT: 18d6a2

file-id - INTEGER 18d6a2a

window-id - INTEGER 18d6a2b

atom-parms - LIST(type, visibility, hit-sensitivity, type-parms) 18d6a2c

type - INTEGER [STRING=0 / PICTURE=1] 18d6a2d

visibility - INTEGER [VISIBLE=0 / INVISIBLE=1] 18d6a2e

If the window in which this atom resides is currently INVISIBLE, then the atom will be invisible indendent of the visibility paratmeter for the atom; on the other hand, if the owning window is VISIBLE, then the visibility of the atom is governed by its visibility parameter. 18d6a2e1

hit-sensitivity - INTEGER [SENSITIVE=0 / INSENSITIVE=1] 18d6a2f

If the window in which this atom resides is currently INSENSITIVE, then the atom will be insensitive indendent of the hit-sensitivity parameter for the atom; on the other hand, if the owning window is SENSITIVE, then the sensitivity of the atom is governed by its hit-sensitivity parameter. 18d6a2f1

type-parms - string-parms / picture-parms 18d6a2g

the value of this parameter is dependent on the type of atom being allocated 18d6a2g1

string-parms - LIST(coord, nlines, nchars) 18d6a2h

coord - LIST(%x% INTEGER, %y% INTEGER) 18d6a2i

this is the coordinate for where the string should start 18d6a2i1

nlines - INTEGER 18d6a2j

this parameter specifies how many lines the string should take 18d6a2j1

nchars - INTEGER 18d6a2k

this parameter specifies how many characters should appear in each line of the string 18d6a2k1

picture-parms - LIST( ) 18d6a2l

the value of this list will be specified later when we can more knowledge about pictures 18d6a2l1

atom-id - INTEGER 18d6a2m

this result is the id that should be used for all future references to the atom 18d6a2m1

deallocate-atom(file-id, window-id, atom-id) 18d6b

This procedure will delete the specified atom(s). 18d6b1

FORMAT: 18d6b2

file-id - INTEGER 18d6b2a

window-id - INTEGER 18d6b2b

atom-id - INTEGER [ALL=-1 / atom-id] 18d6b2c

If this parameter is ALL, then all atoms belonging to this window will be deleted. 18d6b2c1

read-atom-parms(file-id, window-id, atom-id => atom-parms) 18d6c

This procedure will read the current parameters for the specified atom. 18d6c1

FORMAT: 18d6c2

file-id - INTEGER 18d6c2a

window-id - INTEGER 18d6c2b

atom-id - INTEGER 18d6c2c

atom-parms - LIST( ) see above 18d6c2d

manipulate-atom-parms(file-id, window-id, atom-id, new-atom-parms) 18d6d

This procedure allows a process to modify several of the parameters associated with an atom. 18d6d1

FORMAT: 18d6d2

file-id - INTEGER 18d6d2a

window-id - INTEGER 18d6d2b

atom-id - INTEGER 18d6d2c

new-atom-parms - LIST(visibility, hit-sensitivity) 18d6d2d

visibility - INTEGER [VISIBLE=0 / INVISIBLE=1] 18d6d2e

hit-sensitivity - INTEGER [SENSITIVE=0 / INSENSITIVE=1] 18d6d2f

String Manipulation 18d7

(Note that string-id merely means the atom-id for the specified string.) 18d7a

write-string(file-id, window-id, string-id, string-pos, string-parms, string) 18d7b

This procedure will write the passed string at the specifed location (specified by string-id and string-pos). This is a destructive write, i.e., any characters ovewritten will be lost, and if this write is replacing a substring of a string with a different length substring, the resulting string will be opened or closed appropriately. The characters carriage-return line-feed will cause text to appear on succesive lines of the string. 18d7b1

FORMAT: 18d7b2

file-id - INTEGER 18d7b2a

window-id - INTEGER 18d7b2b

string-id - INTEGER 18d7b2c

string-pos - LIST(sline, schar, eline, echar) 18d7b2d

sline - linec 18d7b2e

schar - charc 18d7b2f

eline - linec 18d7b2g

echar - charc 18d7b2h

linec - %first-line% INTEGER [=0] / %current-last-line% INTEGER [=-1] / INTEGER 18d7b2i

charc - %first-char% INTEGER [=0] / %current-last-char% INTEGER [=-1] / INTEGER 18d7b2j

Thus to insert a substring in front of an existing string, specify the string-pos of:

LIST(0, 0, 0, 0)
To append to the end of an existing string use:
LIST(-1, -1, -1, -1)
To replace an arbitrary substring use;
LIST(n, m, p, q)
To replace a substring at the front of a string use:
LIST(0, 0, m, n)
To replace a substring at the end of a string use
LIST(m, n, -1, -1)
To replace an entire string use:
LIST(0, 0, -1, -1) 18d7b2j1

string-parms = LIST(highlight) 18d7b2k

highlight = INTEGER [NORMAL=0 / HIGHLIGHT=1] 18d7b21

This parameter specifies whether or not the newly written string should be made to "stand-out", in any terminal dependent manner, or not. 18d7b211

string = STRING 18d7b2m

read-string(file-id, window-id, string-id, string-pos =>
string) 18d7c

This procedure will read the specified (sub-)string. 18d7c1

FORMAT: 18d7c2

file-id = INTEGER 18d7c2a

window-id = INTEGER 18d7c2b

string-id = INTEGER 18d7c2c

string-pos = see above 18d7c2d

string = STRING 18d7c2e

move-string(file-id-1, window-id-1, string-id-1, file-id-2,
window-id-2, string-id-2) 18d7d

This procedure will move the string from string-id-1 to string-id-2. The entire string at string-id-1 will be replaced by a null string, and the entire string at string-id-2 will be replaced by as much as will fit of string string-id-1. 18d7d1

FORMAT: 18d7d2

file-id-1 - INTEGER 18d7d2a

window-id-1 - INTEGER 18d7d2b

string-id-1 - INTEGER 18d7d2c

file-id-2 - INTEGER 18d7d2d

window-id-2 - INTEGER 18d7d2e

string-id-2 - INTEGER 18d7d2f

copy-string(file-id-1, window-id-1, string-id-1, file-id-2, window-id-2, string-id-2) 18d7e

This procedure will copy the string from string-id-1 to string-id-2. The entire string at string-id-2 will be replaced by as much as will fit of string string-id-1. 18d7e1

FORMAT: 18d7e2

file-id-1 - INTEGER 18d7e2a

window-id-1 - INTEGER 18d7e2b

string-id-1 - INTEGER 18d7e2c

file-id-2 - INTEGER 18d7e2d

window-id-2 - INTEGER 18d7e2e

string-id-2 - INTEGER 18d7e2f

mark-characters(file-id, window-id, string-id-1, string-pos-1, string-id-2, string-pos-2 -> mark-id) 18d7f

This procedure will cause the (sub-)string to be made to "stand-out" in any appropriate terminal dependent manner. The mark-id returned can be used for future references to these characters that are now standing out. 18d7f1

FORMAT: 18d7f2

file-id - INTEGER 18d7f2a

window-id - INTEGER 18d7f2b

string-id-1 - INTEGER 18d7f2c

string-pos-1 - see string-pos above 18d7f2d

string-id-2 - INTEGER 18d7f2e

string-pos-2 - see string-pos above 18d7f2f

mark-id - INTEGER 18d7f2g

remove-mark(file-id, mark-id) 18d7g

This procedure will cause characaters that were previously made to stand-out to no longer standout. 18d7g1

FORMAT: 18d7g2

file-id - INTEGER 18d7g2a

mark-id - INTEGER [ALL=-1 / mark-id] 18d7g2b

If this parameter is ALL, then all marked characters in the specified file will no longer standout. 18d7g2b1

read-marks(file-id, mark-id => mark-id-list) 18d7h

This procedure enables a process to determine which (sub-)strings are currently standing out. 18d7h1

FORMAT: 18d7h2

file-id - INTEGER 18d7h2a

mark-id - INTEGER [ALL=-1 / mark-id] 18d7h2b

mark-id-list - LIST(LIST(mark-id, string-id, string-pos), ...) 18d7h2c

Picture Manipulation 18d8

This branch will be filled in later as we can more knowledge in the area of pictures. 18d8a

Input Control Manipulation 18d9

Send-Coors-with-action(file-id, action-list) 18d9a

Don't-send-Coors-with-actions(file-id) 18d9b

Report-Coors(file-id, cursor-window-id, on-off, criteria-list) 18d9c

Report-mouse-button-status(file-id, criteria-list) 18d9d

Input 18d10

read-byte(file-id -> byte-value) 18d10a

This procedure will read a character from the specified file. 18d10a1

FORMAT: 18d10a2

file-id - INTEGER 18d10a2a

byte-value - CHARACTER 18d10a2b

write-byte(file-id, byte-value) 18d10b

This procedure will write a character on the specified file. Characters written on terminal files by this procedure will appear in windows that have been designated to receive teletype output. 18d10b1

FORMAT: 18d10b2

file-id - INTEGER 18d10b2a

byte-value - BITSTR 18d10b2b

This parameter specifies the new value for the selected byte. 18d10b2b1

read-string(file-id, termination-condition -> string-value) 18d10c

This procedure will read a number of successive characters from the specified file. 18d10c1

FORMAT: 18d10c2

file-id - INTEGER 18d10c2a

temination-condition - LIST(%count%INTEGER, %chars%STRING) 18d10c2b

This parameter specifies that either count bytes are to be read if chars is the null string; or that

bytes are to be read until one of the bytes matches any byte in the chars string if a zero count is specified; or read bytes until a byte is encountered that matches any of the bytes in the chars string if both count and chars are specified, 18d10c2b1

string-value - STRING 18d10c2c

This result is the value of the selected bytes, 18d10c2c1

write-string(file-id, termination-condition, string-value) 18d10d

This procedure will write a number of successive characters on the specified file. Characters written on terminal files by this procedure will appear in windows that have been designated to receive teletype output, 18d10d1

FORMAT: 18d10d2

file-id - INTEGER 18d10d2a

temination-condition - LIST(%count%INTEGER, %chars%STRING) 18d10d2b

This parameter specifies that either count bytes are to be written if chars is the null string; or that bytes are to be written until one of the new bytes matches any byte in the chars string if a zero count is specified; or write bytes until a byte is encountered in the new string that matches any of the bytes in the chars string if both count and chars are specified, 18d10d2b1

string-value - STRING 18d10d2c

This is the value for the new string, 18d10d2c1

Select-char(file-id, coors => window-id, string-id, char-pos, wcoors) 18d10e

This procedure accepts a file-id and coordinates relative to the file and converts them to a window-id, string-id, and line and character position within that string, and to coordinates relative to the selected window. Only strings that are hit sensitive will be considered as possible selection strings, 18d10e1

FORMAT: 18d10e2

file-id - INTEGER 18d10e2a

coors - LIST(%x% INTEGER, %y% INTEGER) 18d10e2b

window-id - INTEGER 18d10e2c

string-id - INTEGER 18d10e2d

char-pos - LIST(%line-number% INTEGER,
%character-position% INTEGER) 18d10e2e

wcoors - LIST(%relative-x-position% INTEGER,
%relative-y-position% INTEGER) 18d10e2f

Select-string(file-id, coors -> window-id, string-id,
wcoors) 18d10f

This procedure accepts a file-id and coordinates relative to the file and converts them to a window-id and a string-id, and to coordinates relative to the selected window. Only strings that are hit sensitive will be considered as possible selection strings. 18d10f1

FORMAT: 18d10f2

file-id - INTEGER 18d10f2a

coors - LIST(%x% INTEGER, %y% INTEGER) 18d10f2b

window-id - INTEGER 18d10f2c

string-id - INTEGER 18d10f2d

wcoors - LIST(%relative-x-position% INTEGER,
%relative-y-position% INTEGER) 18d10f2e

select-window(file-id, coors -> window-id, wcoors) 18d10g

This procedure accepts a file-id and coordinates relative to the file and converts them to a window-id and to coordinates relative to the selected window. Only windows that are hit sensitive will be considered as possible selection windows. 18d10g1

FORMAT: 18d10g2

file-id - INTEGER 18d10g2a

coors - LIST(%x% INTEGER, %y% INTEGER) 18d10g2b

window-id - INTEGER 18d10g2c

wcoors - LIST(%relative-x-position% INTEGER,
%relative-y-position% INTEGER) 18d10g2d

Global and Miscellaneous Procedures 18d11

process-batch(procedure-list) 18d11a

This procedure allows a process to "batch" a group of procedure calls into one transmission. Any results from any of the procedures within the batch will be lost. 18d11a1

FORMAT: 18d11a2

procedure-list - LIST(LIST(pname, pargs), ...) 18d11a2a

pname - ADDRESS 18d11a2b

the name of the procedure to be called 18d11a2b1

pargs - any 18d11a2c

these are the parameters to be passed to the procedure pname. 18d11a2c1

write-error-message(error-string) 18d11b

This procedure enables a process to place a message in the error window. 18d11b1

FORMAT: 18d11b2

error-string - STRING 18d11b2a

write-status-message(status-string) 18d11c

This procedure enables a process to place a message in the status window. 18d11c1

FORMAT: 18d11c2

status-string - STRING 18d11c2a

read-terminal-file-characteristics(file-id =>
file-characteristics-list) 18d11d

This procedure enables a process to determine the logical

class and the physical characteristics of the terminal to which file-id refers. 18d11d1

FORMAT: 18d11d2

file-id - INTEGER 18d11d2a

file-characteristics-list - LIST( ) 18d11d2b

The value of this list will be specified later; however, it will probably include such things as the logical terminal class, whther or not the terminal has lower case, etc. 18d11d2b1

set-terminal-file-characteristics(file-id, new-file-characteristics-list) 18d11e

This procedure enables a process to modify some of the physical characteristics of the terminal to which file-id refers. 18d11e1

FORMAT: 18d11e2

file-id - INTEGER 18d11e2a

file-characteristics-list - LIST( ) 18d11e2b

The value of this list will be specified later; however, it will probably include such things as tab stops, etc. 18d11e2b1

reset-process-terminal-file(process-id) 18d11f

This procedure will reset the PIF and POF files for the specified process back to their initial state (see above -- xxx). 18d11f1

FORMAT: 18d11f2

process-id - INTEGER 18d11f2a

This procedure will return the current date and time in a

canonical form (probably similar to the internal form used by TENEX). 19a1a

FORMAT: 19a1a1

time - INTEGER 19a1a1a

convert-time-to-string (time, format-qualifiers -> time-string) 19a2

This procedure will convert the passed date and time (in canonical form) to a string as specified by format-qualifiers. 19a2a

FORMAT: 19a2b

time - INTEGER [CURRENT=-1 / time] 19a2b1

If this parameter is CURRENT, then the current date and time will be used. 19a2b1a

format-qualifiers - LIST( ) 19a2b2

this parameter will be specified later when we get more knowledge. 19a2b2a

time-string - STRING 19a2b3

convert-string-to-time (time-string, format-qualifiers -> time) 19a3

This procedure will convert the passed date and time string to canonical form. 19a3a

FORMAT: 19a3b

time-string - STRING 19a3b1

format-qualifiers - LIST( ) 19a3b2

This parameter will be specified later when we gain more knowledge; however, we expect it to contain elements that describe the content and format of the passed string. 19a3b2a

time - INTEGER 19a3b3

Buffers 19b

The following primitives provide a process with storage managment capabilities. It is the process' responsibility not to write outside the limits of an allocated buffer. 19b1

Initialize-buffer-pool(pool-address, pool-size, parm => pool-id) 19b2

This procedure informs the OSI of the virtual address space within a process that is to be used for buffers. The pool-id returned should be used in future calls to allocate and delete individual buffers. 19b2a

FORMAT: 19b2b

pool-address - ADDRESS 19b2b1

pool-size - INTEGER 19b2b2

This is the total size of the buffer pool. 19b2b2a

parm - LIST(type, size [,gcol]) 19b2b3

This parameter indicates if all the individual buffers are to be fixed size (and their size) or if the individual buffers are going to be variable in size (and the maximum size of any individual buffer). The element gcol must be specified for variable size pools. If gcol is IMPLICIT and a subsequent allocate-buffer would fail because it could not allocate the requested buffer, a garbage collection will take place and an appropriate SIGNAL will be generated. During this garbage collection, no "fixup" of the contents of any of the buffers will take place. If gcol is EXPLICIT, then subsequent allocates will fail if the requested buffer size cannot be allocated. 19b2b3a

type - BOOLEAN [FIXED=TRUE / VARIABLE=FALSE] 19b2b4

size - INTEGER 19b2b5

gcol - BOOLEAN [IMPLICIT=TRUE / EXPLICIT=FALSE] 19b2b6

pool-id - INTEGER 19b2b7

Allocate-buffer(pool-id [,size] => buffer-id) 19b3

This procedure will allocate a fixed size buffer, or a buffer of the requested size in variable size pools, in the specified buffer pool. The buffer-id returned is the

address of cell that contains the address the first usable word in the allocated buffer. The address of any specific buffer can change due to a garbage collection. A error will be generated if no more buffers exist, or if the requested size buffer cannot be allocated. 19b3a

FORMAT: 19b3b

pool-id - INTEGER 19b3b1

size - INTEGER 19b3b2

buffer-id - ADDRESS 19b3b3

Delete-buffer(buffer-id) 19b4

This procedure returns a previously allocated buffer to the buffer pool. 19b4a

FORMAT: 19b4b

buffer-id - ADDRESS 19b4b1

Garbage-collect-pool(pool-id) 19b5

This procedure will garbage collect a buffer pool. This procedure only has real meaning for varable size buffer pools. This procedure will not do any "fixup" on the contents of indiviual buffers. 19b5a

FORMAT: 19b5b

pool-id - INTEGER 19b5b1

KEV 6-DEC-74 15:30 24688

operating system interface for nsw

(J24688) 6-DEC-74 15:30;;;; Title: Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /NPG([ACTION]) RWW([ACTION]) ; Sub-Collections: SRI-ARC NPG; Clerk: KEV; Origin: < VICTOR, O-S-I,NLS;2, >, 6-DEC-74 15:27 KEV ;;;;####;

KEV 6-DEC-74 15:36 24689

development journal ident???

how come there is no journal ident for arc's development staff?? 1

development journal ident???

(J24689) 6-DEC-74 15:36;;;; Title: Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /RWW([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: KEV;

DIA 6-DEC-74 16:13 24690

Re: 24639, L10 signals

I think the best way to handle the situation you describe, of being sure cleanup code makes it thru a list of procedure calls, is to invoke a catchphrase around each call. New signal stuff will provide a nice way to do that. 1

DIA 6-DEC-74 16:13 24690

Re: 24639, L10 signals

(J24690) 6-DEC-74 16:13;;;; Title: Author(s): Don I. Andrews/DIA; Distribution: /JEW([INFO-ONLY]) CHI([INFO-ONLY]) KEV([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DIA;

DVN 6-DEC-74 16:57 24691

Final Report Prod

These are the sections of the Final Report where work remains with links to the draft. RWW, DVN, HGL have writing to do; EKM, RLL, JDH, JMB, JCN, AND SRL have reviewing or response to reviewing. 1

Head Matter 2

Done but DVN needs to Review it (documentation,final,1) 2a

ABSTRACT 3

DVN needs to write it, SRL can then review it. (documentation,final,2) 3a

INTRODUCTION 4

DVN needs to write it, RWW can then review it. (documentation,final,3) 4a

LONG TERM GOALS 5

Some Basic Characteristics of a Knowledge Workshop System and Status of NLS with Respect to Them. 5a

RWW needs to finish updating and then JCN review.(documentation,final,4b) 5a1

Aspects of ARC's Technology Transfer Strategy 5b

RWW needs to finish rewritng, RLL can then review. (documentation,final,4d) 5b1

Organizational Development 5c

JDH needs to Review (documentation,final,4e) 5c1

WORKSHOP TECHNOLOGY 6

USER INTERFACE 6a

The NLS Command Language and User Interface: An Overview 6a1

RWW needs to do it so CHI can Review (documentation,final,,5a) 6a1a

Query/HELP Software and Data Bases 6a2

KIRK has done his part, DVN and HGL Need to do theirs. Then JMB can Review it.(documentation,final,,5a3) 6a2a

Final Report Prod

(J24691) 6-DEC-74 16:57;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /RWW([ACTION]) HGL([ACTION]) JDH([ACTION]) JCN([ACTION]) EKM([INFO-ONLY]) RLL([INFO-ONLY]) JMB([INFO-ONLY]) SRL([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DVN; Origin: < VANNOUHUYS, REPORTPROD,NLS;1, >, 6-DEC-74 16:54 DVN ;;;;####;

DVN POOH KIRK JMB 6-DEC-74 19:37 24692

Informal Documentation Report for Week Ending Dec 6th

pooh-monday was spent flapping my wings to get back from nyc. the rest of the week i have been working on revisions in help. i have also been working on finishing up the list of available documentation and getting copies made of certain documents. 1

JMB: I fixed up XHELP to have complete and accurate info on the Feedback mechanism. Also made some changes in XHELP from POOH's edits of the glossary. 2

KIRK: Teaching POOH the conventions and procedures for making changes in the Help database has occupied about 2/3 of my documentation time. The other 1/3 was updating Help re-doing the conceptual interface to content-analyzer patterns and programs and beginning to re-organize the Sendmail concepts including Journal, Numbers, reading mail, and Identification. 3

DVN: 4

COM:I sent to tape at ISI the revised viewspec cards, Larry Day's paper, and the Paper on CML by Ken Victor et al. DDSI said they ran them last night and put them in the mail today. Proofs of two documents, a draft of the DCA paper Susan Lee has been working on, and Martin Hardy's Paper on the Lineprocessor sent to them Nov 18 have not appeared. DDSI claims they were sent to us via UPS November 29th. They are going to rerun the fil m and try gain. 4a

I made slow progress on the fillowing outstanding documents: Command Summary (failed in two more attempts to use the index program, I plan to give up and send it to COM without an index), The Lineprocessor User's Guide, and TNLS Addressing. Preface to NLS, a three-page introductory document, has gone to Applications for approval. 4b

I routinely edit certain changes in Help. 4c

DVN POOH KIRK JMB 6-DEC-74 19:37 24692

Informal Documentation Report for Week Ending Dec 6th

(J24692) 6-DEC-74 19:37;;;; Title: Author(s): Dirk H. Van Nouhuys, Ann Weinberg, Kirk E. Kelley, Jeanne M. Beck/DVN POOH KIRK JMB; Distribution: /JOAN([ACTION] dirt notebook please) DIRT([INFO-ONLY]) ; Sub-Collections: DIRT SRI-ARC; Clerk: KIRK;

CHI 6-DEC-74 23:09 24693

Recommendation for support of line-at-a-time and character-at-a-time half-duplex terminals for NSW

The following is my recommendation for how to handle line-at-a-time and character-at-a-time half duplex terminals in first-year NSW: 1

Line-at-a-time: 1a

Command-word recognition will automatically be set to DEMAND (user types as much of the command word as he wishes, so long as it is sufficient to disambiguate the current alternatives, and terminates them with a <space>. Other recognizers will also allow users at this type of terminal (and perhaps all others) to terminate thier recognition with <space> if this does not introduce ambiguities. 1a1

To obtain help from the frontend (current alternatives or syntax of commands) the user will partially specify a command and end it with the appropriate help key and hit CR, EOT, or whatever it takes to get the line sent to the frontend. 1a2

The frontend will do no echoing to the terminal if the user's input matches the possible command parse states. In addition, it would be nice for the user not to have to confirm his commands except by sending the line to the frontend. However, if the line does not match a possible parse, the the line will be interpreted as best it can and the results will be echoed back to the user and he will have to confirm it by again typing CR or whatever. If the parse that results from the discarding of characters that do not match is not what the user intended, he may type <backspace> a number of times or abort the command specification with the abort key (whatever he defined that to be). 1a3

Since the <backspace> key on these terminals often does not transmit a character to the computer, the user may define some other transmitted character as his <backspace> key and use it as described above. 1a3a

Character-at-a-time, half duplex 1b

As with full duplex, the terminal's bell will be rung when the user's input does not match a possible parse. Also, this type of terminal could receive noise word prompts if the user desired as well as command word completion. These would be up to the user's discretion. 1b1

Your comments and suggestions are invited. 2

-- Charles. 3

CHI 6-DEC-74 23:09 24693

Recommendation for support of line-at-a-time and character-at-a-time half-duplex terminals for NSW

(J24693) 6-DEC-74 23:09;;;; Title: Author(s): Charles H. Irby/CHI; Distribution: /NPG([INFO-ONLY]) RWW([INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: CHI; Origin: < IRBY, LINE-AT-A-TIME.NLS;1, >, 6-DEC-74 22:45 CHI ;;;;####;

CHI 8-DEC-74 20:10 24696

Development Meeting

There will be a Development meeting at 3:00 on Thursday, the 12th. Dick will want to talk about his recent trip and meeting and review our progress. Anyone outside of Development who may be interested should feel free to attend. 1

CHI 8-DEC-74 20:10 24696

Development Meeting

(J24696) 8-DEC-74 20:10;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: CHI;

RLL 8-DEC-74 21:31 24697

BUG: <NUL><NUL> appearing in output when hitting ? key in DNLS

A series of <NUL>'s appars several times whenever I type a question mark to get th options. This has happened in several places and appears to be inconsistent. 1

RLL 8-DEC-74 21:31 24697

BUG: <NUL><NUL> appearing in output when hitting ? key in DNLS

(J24697) 8-DEC-74 21:31;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /FEED([ACTION]) JDH([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RLL;

SRL 9-DEC-74 05:55 24698

New Documentation Should be Reviewed by KWAC as well as APPLICATIONS

Is there an ident for distribution to Applications?

SRL 9-DEC-74 05:55 24698

New Documentation Should be Reviewed by KWAC as well as APPLICATIONS

I notice in the Documentation Report that a document Preface toNLS, a three-page introductory document, has gone to Applications for approval. I don't know the normal procedure (if there is one) but thought it might be a good ideas to get comments from the KWAC group as well as Applications when a new piece of documentation is being reviewed.

1

SRL 9-DEC-74 05:55 24698

New Documentation Should be Reviewed by KWAC as well as APPLICATIONS

(J24698) 9-DEC-74 05:55;;;; Title: Author(s): Susan R. Lee/SRL;
Distribution: /JCN([ACTION]) JHB([ACTION]) RLL([ACTION]).;
Sub-Collections: SRI-ARC; Clerk: SRL;

JAKE 9-DEC-74 10:13 24699

message margins are wierd and delete modifications gives bad file

When I copy my messages from message.txt using the pogram Message,
the margins are all messed up again so that messages with multiple
carriage returns (as most sndmsgs have) are practically unreadable.
Also when I attempted to do a Delete Modifications I got a bad file. 1

JAKE 9-DEC-74 10:13 24699

message margins are wierd and delete modifications gives bad file

(J24699) 9-DEC-74 10:13;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /FEED([ACTION]) FEEDBACK([ACTION]) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: JAKE;

JAKE 9-DEC-74 10:58 24700

AMC-MIS

Stan, Do you think I could be a member of the AMC-MIS group, that is if it is a general interest group and not a private one. I would be interested to keep up with what you are doing. If it is agreeable let me know and I will add my name to the list. Regards, Jake 1

JAKE 9-DEC-74 10:58 24700

AMC-MIS

(J24700) 9-DEC-74 10:58;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /SMT([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

JAKE 9-DEC-74 11:14 24701

Sites on the Network and Network Resources

Dear Dave, I believe what you are after with respect to sites is a list of all the TIPs and their phone numbers. Unfortunately, I cannot supply such a list. Craig Fields of ARPA IPTO has control of who can access what TIP and th phone numbers are not public information (although many of them are around subrosa). To further complicate the situation ARPA is now in the process of requiring a login procedure, account and password for accessing TIPs. Perhaps when this system is installed (next year sometime) the phone numbers will be public since the account, password, etc. will be the deciding factor for use. I will send you a copy of the Arpanet Directory by U.S. Mail. This will give you some indication of who is on and what they are doing. The third page tells you how to access the NIC guest account by typing 'NIC <CR>' at TENEX level (you can do this from your account also). This will take you into the online Resource Handbook which unfortunately is in a bad state of arrears at the moment. I am currently trying to upgrade it but it will take awhile.
(

1

Sites on the Network and Network Resources

(J24701) 9-DEC-74 11:14;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /DAP([ACTION]) FEEDBACK([INFO-ONLY]) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: JAKE;

DVN 9-DEC-74 15:31 24712

Electrostatic Printers: Gould, Versatec, Varian, and Xerox

Three companies compete closely in this field, VERSATEX, VARIAN, and GOULD. 1

GOULD 5000 1a

Cost: $7000 with appropriate options + $1200 for PDP-11 interface, 1a1

Delivery: "30-45" days from order 1a2

Speed: Alphanumeric Mode 7x9 matrix: 1600 lines/minute; 1a3

Graphic Mode (necessary to simulate different type faces 3.25 inches/sec 1a4

Resolution: 100 dots/inch in graphic mode 1a5

Claims their software uses "orders of magnitude" less cpu time than competitors. 1a6

I have brochures and manuals in my office. 1a7

VARIAN VPM V70 1b

Cost: $6,900; Interface to PDP-11 $1,275 1b1

Delivery: "45-60" days 1b2

Resolution: various options. The best is 10x14 dot matrix. In plot mode 100 points/inch. 1b3

Speed: Various options up to 1,320 lines/minute for matrix characters, 2.75 lines/sec for plot mode. 1b4

I have brochures and samples in my office. 1b5

VERSATEC D1200 A 1c

Cost: $9,7000; Interface to PDP-11 $1,275 1c1

Delivery: Unknown 1c2

Resolution:16x16 matrix or 200 points/inch in plot mode. 1c3

Speed: 500 lines per minute in matrix mode; 1 inch per second in plot mode. 1c4

Electrostatic Printers: Gould, Versatec, Varian, and Xerox

I have brochures and samples in my office. 1c5

XEROX XGP 2

The XGP is an LDX with hardware and software attached to accept data from a computer medium rather than a telephone line. The picture as far as buying it is cloudy. The Xerox front office in San Francisco says they are for sale for $40K but only with a Xerox 530 computer for about $80K more. I have information on the 530. Bill English suggests he might be able to arrange for us to buy the XGP alone for about $40K. The XGP running in the ARPA office offers resolution of about 200 dots/inch, good enough to reproduce reports from, Theoretically it can run at 270 dots/inch. The software for a PDP 11 to run the XGP is clouded. ARPA runs on a somewhat klugie version built at ISI on top of software developed at Carnegie-Mellon. Dick Watson is acquainted with the developers at Carnegie-Mellon and plans to discuss the situation with them. 2a

I have samples in my office of documentation printed in the ARPA Office. 2b

COMMENTS: 3

If we want to use any of these machines to simluate the variety of typefaces and formats COM offers, the software to do it must come from somewhere. Perhaps the Carnegie-Mellon software for the XGP could be easilly made to fit out needs. If so, the extra expense of providing fonts and layout for the other three printers must be counted against them in a cost analysis. The brochures of each claim it is easey. Some one with the right programming experience should look at this question (Bob Bellville?). 3a

For that purpose, 200 dots per inch (XGP and Versatec) seems clearily better than 100 dots per inch (Varian and Gould). 3b

Bob Bellvile and I agreed that the optimum paper width is 11 inches which allows reasonable speed and cost with some extra size pages and foldouts. 3c

Varian has been longest in the field Varian and Versatec are made localy. Gould has a very good reputation in the field of printing equipment generally, and has a service representative in Burlingame. 3d

All claim lots of experience running with PDP-11s. 3e

DVN 9-DEC-74 15:31 24712

Electrostatic Printers: Gould, Versatec, Varian, and Xerox

(J24712) 9-DEC-74 15:31;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /JOAN([ACTION] dpcs notebook please) DPCS([INFO-ONLY]) ; Sub-Collections: DPCS SRI-ARC; Clerk: DVN;
Origin: < HAMILTON, ELECTROPRINTSIT.NLS;3, >, 9-DEC-74 15:00 DVN
;;;;####;

proposed standard file formats

Network Working Group
Request for Comments: rrr

J. Postel (SRI-ARC)
dd December 1974

NIC: jjjjj

Standard File Formats

Introduction

In an attempt to provide online documents to the network community we have had many problems with the physical format of the final documents. Much of this difficulty lies in the fact that we do not have control or even knowledge of all the processing steps or devices that act on the document file. A large part of the difficulty in the past has been due to some assumptions we made about the rest of the world being approximately like our own environment. We now see that the problems are due to differing assumptions and treatment of files to be printed as documents. We therefore propose to define certain standard formats for files and describe the expected final form for printed copies of such files.

These standard formats are not additional File Transfer Protocol data types/modes/structures, but rather usage descriptions between the originator and ultimate receiver of the file. It may be useful or even necessary at some hosts to construct programs that convert files between common local formats and the standard formats specified here.

Standardization Elements

The elements or aspects of a file to be standardized are the character or code set used, the format control procedures, the area of the page to be used for text, and the method to describe overstruck or underlined characters.

The area of the page to be used for text can be confusing to discuss, in an attempt to be clear we define a physical page and a logical page.

Physical Page

The physical page is the medium that carries the text, the height and width of its area are measured in inches.

The typical physical page is a piece of paper eleven inches high and eight and one half inches wide.

Typical print density is 10 characters per inch horizontally and 6 characters per inch vertically. This results in the typical physical page having a maximum

capacity of 66 lines and 85 characters per line. It is often the case that printing devices limit the area of the physical page by enforcing margins.

Logical Page

The logical page is the area that can contain text, the height of this area is measured in lines and the width is measured in characters.

A typical logical page is 60 lines high and 72 characters wide.

Code Set

The character encoding will be the network standard Network Virtual Terminal (NVT) code as used in Telnet and File Transfer protocols, that is ASCII in an eight bit byte with the high order bit zero.

Format Control

The format will be controlled by the ASCII format effectors:

Form Feed <FF>

Moves the printer to the top of the next logical page, and to the left edge of the logical page. [Note that this differs from the NVT specification].

Carriage Return <CR>

Moves the printer to the left edge of the logical page remaining on current line.

Line Feed <LF>

Moves the printer to the next print line, keeping the same horizontal position.

Horizontal Tab <HT>

Moves the printer to the next horizontal tab stop.

The default stops for horizontal tabs will be every eight characters, that is character positions 9, 17, 25, ... within the logical page.

Vertical Tab <VT>

Moves the printer to the next vertical tab stop.

The default stops for vertical tabs will be every eight

lines starting at the first printing line on each logical page.

Back Space <BS>

Moves the printer one character position toward the left edge of the logical page.

Not all these effectors will be used in all format standards, any effectors which are not used in a format standard are ignored.

Page Length

The logical page length will be specified in terms of a number of lines of text. This describes the number of lines per physical page available for text. This does not specify the size of the physical page or the font.

Page Width

The logical page width will be specified as a number of characters. This describes the number of characters per line of the physical page available for text. This does not specify the physical size of the page or the font.

Overstriking

Overstriking (note that underlining is a subset of overstriking) may be specified to be done in one or both of the following ways, or not at all:

By Line

The text of the line will be followed by a <CR> then the overstriking will follow as a series of space and overstrike characters followed by <CR><LF>.

By Character

Each character to be overstruck is to be immediately followed by a <BS> and the overstrike character.

Standard Formats

Format 1

This format is designed to be used for documents to be printed on line printers, which normally have 66 lines to a physical page, but often have forced top and bottom margins of 3 lines each.

Active Format Effectors
<FF>, <CR>, <LF>.
Page Length
60 lines.
Page Width
72 Characters.
Overstriking
By Line.

Format 2

This format is designed to be used with hard copy terminals, which in the normal case have 66 lines to a physical page.

Active Format Effectors
<FF>, <CR>, <LF>, <HT>, <VT>, <BS>.
Page Length
66 lines.
Page Width
72 Characters.
Overstriking
By Character.

Format 3

This format is designed to be used with full width (11 by 14 inch paper) line printer output.

Active Format Effectors
<FF>, <CR>, <LF>.
Page Length
60 lines.
Page Width
132 Characters.
Overstriking
None.

Format 4

This format is designed to be used for simulated card input. The page width is 80 characters, each card image is followed by <CR><LF>, thus each card is represented by 82 characters in the file.

Active Format Effectors
<CR>, <LF>.
Page Length
Infinite.
Page Width
80 Characters.
Overstriking
None.

Implementation Suggestions

Overflow

Overflow can result from two causes, first if the physical page is smaller than the logical page, and second if the actual text in the file violates the standard under which it is being processed.

In either case the following suggestions are made to implementors of programs which process files in these formats.

Length

If more lines are processed than fit within the minimum of the physical page and the logical page length since the last top of page action, then the top of page action should be forced.

Width

If more character positions are processed than fit on the minimum of the physical page width and the logical page width since the last left edge action, then characters are discarded up to the next format effector.

or

If more character positions are processed than fit on the minimum of the physical page width and the logical page width since the last left edge action, then the left edge and next line actions should be forced.

References

A. McKenzie "TELNET Protocol Specification," NIC 18639, Aug-73.

"USA Standard Code for Information Interchange," United States of America Standards Institute, 1968.

proposed standard file formats

(J24713) 9-DEC-74 17:52;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /EKM([ACTION]) DSM([ACTION]) JEW([ACTION]) NDM([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP; Origin: < POSTEL, FILE-STANDARDS,NLS;10, >, 9-DEC-74 17:37 JBP
;;;;####;

JAKE 9-DEC-74 18:27 24714

Removal of <CR> during NLSizing message.txt file

The MESSAGE program does not remove <CR>s on input into NLS form and that is what causes the problem I was discussing. The old carriage returns are there but the margin width has been changed from the standard format of sndmsg and therefore the carriage returns from sndmsg are interspersed with the ones put in automatically by NLS. This results in a very messy file to read. 1

JAKE 9-DEC-74 18:27 24714

Removal of <CR> during NLSizing message.txt file

(J24714) 9-DEC-74 18:27;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /FEED([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

JAKE 10-DEC-74 09:14 24715

Weekly project charge sheets

I would like to have a full set of the weekly carge sheets for Project 3803 by tomorrow (Weds.) Also, I would like to receive the copy for the project leader every week when it arrives from now on. This is just to put it in writing so it will hopefully get in your action stack. Cheers, Jake 1

Weekly project charge sheets

(J24715) 10-DEC-74 09:14;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /SLJ([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

EKM 10-DEC-74 09:24 24716

NSW documents vintage 1973

One of two NSW documents written a year ago. Bill Carlson asked to have them journalized.

The National Software Works

December 20, 1973

Robert Balzer, USC/Information Sciences Institute

T. E. Cheatham, Harvard University

Stephen Crocker, ARPA-IPT

Stephen Warshall, Massachusetts Computer Associates

USC/Information Sciences Institute

THE NATIONAL SOFTWARE WORKS

4676 Admiralty Way

Marina del Rey, California 90291 USA

The National Software Works

INTRODUCTION

The production and maintenance of large programs is still an outrageously expensive activity. The costs are not only high, but also difficult to predict or control. Aside from the manifest payoffs derived from the use of compilers and (some) operating systems and a certain amount of improvement experienced by programmers who code interactively, it is not at all clear that the last twenty years of research and development in programming technology have made any serious dent in the problem.

This situation is particularly interesting in the light of a general suspicion that, in principle, the problem ought to be eased by the creation of better software to support the program production and maintenance process for surely a great deal has been spent in the effort to invent just such software. The reasons for our failure are arguable and a variety of hypotheses have been put forward:

- that the necessary tools -- or, at least, many of them -- exist in the research centers but are not being effectively delivered to the practical programming community

- that feedback from the user community has insufficient influence on the research laboratories, so that research emphasis is unrelated to user needs

- that the necessary tools exist, but are diffused over a variety of hardwares in many physical locations; the problem is that of difficulty of access.

Each of these hypotheses -- and the list may readily be extended -- doubtless contains a certain amount of truth, and collectively they surely suggest that dramatic improvements in the way programs are built are less likely to come from marginal improvements in present tools (or the invention of some magical new one) than from better methods of tool access and delivery, and better communication between research laboratory and end user.

The idea of a National Software Works (NSW) on the ARPANET [1] arose fairly naturally from these considerations. If some number of end users were put on the network, and enouth additional off-the-shelf software were brought up on the network to supply a complete set of conventional tools -- compilers, documentation aids, debugging systems, etc. -- for normal program development work, some useful results might be expected to follow:

* The user would immediately have more convenient access to standard tools unavailable on his own hardware (or seldom available if his hardware is often tied up running production).

* The user would find it easy to access novel tools in use at research facilities presently on the network, but not otherwise available to him. 16b

* Contact between the research laboratories and the user community would naturally improve. 16c

In sum, the NSW might both immediately improve the present situation of the user, and in the long term, provide an effective vehicle for the communication of need from user to researcher and of responsive tool from researcher to user. 17

It was soon recognized, however, that a view of the NSW as a mere lash-up of tools which happened to reside on the ARPANET would be extremely short-sighted. The fact that all programmer contact with tools would pass through a common communication mechanism with immense computing resources created a golden opportunity for the study -- and perhaps control -- of the whole process of large program creation and maintenance. This thought was particularly attractive in the light of our feeling that one of the most weakly supported areas in the production and maintenance process is project management: the absence of any tool which keeps track of what is going on, relating particular programmer activities to each other or to the overall picture, appears on the face of it rather a bad idea. 18

To clarify the sort of support we have in mind and to suggest its influence upon the NSW design, we will digress briefly to talk in general about the program production process. 19

THE PROGRAMMING PROCESS

In the production of a large program, numerous programmers cooperate in a venture whose end product is, in some sense, a single entity. In the course of their work toward this goal, programmers prepare, edit, and manipulate a very large number of pieces of "text" of various types: routines in a programming language, data descriptions, structured data objects, modules of object code produced by a compiler, assemblages of such modules linked together by a link editor, items of program documentation, and so on.

To the degree that all of these types of text are either machine-processable or machine-producible, it is reasonable to say that they are all either prepared (and repaired) by programmers or produced by "tools", by which we mean elements of support software invoked by programmers to operate on pieces of text.

The number of such pieces of text which come into existence in the course of a large project can be astronomical, and even the number in some kind of active status at a particular time is likely to be huge. It ought to be clear that any absence of control over this large and shifting inventory of material is an invitation to confusion and the almost total absence of any support software for "inventory control" might have something to do with the high and uncontrolled cost of program production (and perhaps something to do with our difficulties in figuring out what we are doing wrong).

Suppose by contrast that the total inventory of text pieces were explicitly regarded as one logically integrated data base -- the Project File -- and that some piece of support software were charged with the responsibility of managing that data base. This piece of software - for the monent, let us call it the File Manager - would, of course, keep books on the contents of the Project File. These books would include not only the character and status of each item in the Project File, but also its relationship to other items in the File (that A is a later version of B, that C is the object code module corresponding to COBOL text D, and so on).

It should be obvious that, if we have designed the books correctly and arranged matters so that they are always kept accurately and completely, they provide the data crucial to any serious attempt by management to explore or control what is happening in the project.

It is, of course, essential to any interesting use of the project books that they be always complete and correct, that there be no path of entry to the Project Files unguarded by the File Manager. This suggests strongly that an individual programmer's use of his tools -- at least when that use yields a non-transitory (filed) result -- must always be reported to (and, perhaps, controlled by) the File Manager.

To arrange matters so that this requirement is met is extremely difficult when the support software designer is confined to the resources of a particular local hardware: to keep the File Manager and its books effectively on line at all times may be insupportably expensive. Indeed, if a project's development work is performed on several computers with no communication among them, it may be logically impossible to create a reasonable File Manager. Thus, it is not surprising that there has been no serious attempt to provide a facility of the sort we have described: at least the naturalness, if not the feasibility, of the idea depends on a unification and scale of computing resource found only in gigantic machines or in networks. 27

THE NSW ENVIRONMENT

Against the background of our feeling that serious progress in rationalizing large program production will come less from the polishing of particular tools than from a frontal attack on the issue of improved access to tools and centralized management of the vast inventory of text floating around a large project, the logic of our strategy for the National Software Works becomes easy to see.

- First of all, it is our intent to put a project's programmers on line to the ARPANET. This has the immediate effect of giving them access to many tools unavailable on their own local hardware.

- Second, we will supply interactive editing packages, both a general text editor and editors which "speak" one or two common programming languages; the effect of such tools in facilitating program preparation and modification is too well known to require any defense here.

- Third, projects will be able to store these files on very inexpensive on-line mass storage devices (the Datacomputer [2]). This should relieve a considerable part of a project's local off-line file maintenance problems, and facilitate load-sharing, when the project's local computer is busy.

- Fourth, a File Manager will be always on line monitoring the content and structure of the project's files and keeping the books up to date, as text pieces are created and manipulated.

The presence of the first three facilities will permit the project to conduct its business more or less as it does now (using the same languages, the same tools, etc.) with certain improvements in ease of tool access and foreign hardware access, editing, and file management. In addition, the project may, at its option, experiment with the use of different tools scattered around the network.

The fourth facility opens the door to some genuinely new ways of controlling projects in the future. To begin with, a fairly powerful query system will be provided to answer questions about any filed entity: what it is, where it came from, what other entities depend on it, etc. Later we will introduce a variety of experimental tools for project control which use the File Manager's books as their primary data or use the fact of the File Manager's existence as their means of invocation (after all, the later provides a single control point "awakened" every time anything interesting happens). Here are some proposed tools:

* Project Status Reporter: This relates the present status of

the files to the overall project plan (in machine-readable form), identifying bottlenecks, critical paths, etc. 31a

* Project Accountant: This produces reports on the frequency and cost of various patterns of activity interesting to project management. 31b

* Policy Enforcer: Everybody in Section A must use the same version of function X; no programmer may link up two routines until each is adjudged debugged by a section manager; no programmer may start debugging until all his code is written; no programmer may write any code for phase 2 of the project until he has written all his code for phase 1; no programmer may start writing a new routine until his last is documented. The above list of (rather inane) policy dicta are meant to suggest a large family of more reasonable policies which might apply to some or all programmers at various phases of a project. If a plausible way of expressing such dicta in machine-readable form can be developed, it is no great trick to devise a tool which is invoked by the File Manager to verify that the present action of some programmer is consistent with policy, so that the action may be inhibited or permitted accordingly. 31c

The use of such new tools by the project would of course, be optional. In any event, the research community can make use of such tools to collect the data it needs to discover what makes program development and maintenance so expensive. 32

SUPPORTING TECHNOLOGY 33

Virtually all multi-program operating systems have attempted to create a suitable programming environment by providing a set of tools Some merely provided a library from which tools could be selected one at a time by the programmer. Others, like MULTICS [3], CP-67 [4], VS [5], and TENEX [6], have provided an on-line environment for program building and debugging. 34

All of these systems have been built on a single computer and this has severely limited their capability to provide the type of environment described in the previous section. In fact, until recently a combination of several such hardware and software technical problems existed which prevented the conception and implementation of this type of environment. These problems and their solution in the NSW are given below. 35

1. Single machine implementation (all tools provided had to exist on the same machine): Computer networks, such as the ARPA Network [1], have established a communication mechanism whereby cooperating programs in different machines can function together as a single system (the technical basis for eliminating these problems are provided by computer networks, centralized mass storage, the Programmer's Interface [7], ACTORS [8], and Execution Machines (see SYSTEM DESCRIPTION section below). The Programmer's Interface has utilized this net technology to create an on-line programming environment combining tools which run on different machines. 36

2. Non-integrated "tool-at-a-time" systems: current systems either segregate their tools into non-interacting components which are invoked one at a time or else provide highly complex integrated versions of these with the interactions between them built into the systems themselves. The type of programming environment we envision requires that actions or events in one part of the system permeate throughout the rest to maintain consistency and coordination between the component parts. The concept of ACTORS, by externalizing and removing the control and communication between the component parts, greatly simplifies constructing an integrated and coordinated system. 37

3. Machine Independence: although tools running on different machines may be integrated into a single one the technology does not exist to run a single program on several different machines and obtain the same results. Therefore, software being produced must be executed and tested on the machine for which it is intended to run in production mode. Thus, if the software environment is to be used to produce programs for more than one machine, each of these must be hooked in through the computer

network and a small portion of the system replicated on each Execution Machine to provide for translation and run-time monitoring capabilities. The rest of the software environment is common and can be shared independent of the machine for which execution is intended. 38

4. Language Independence: currently, if software is to be produced for more than one language then the tools must either be duplicated in separate and distinct integrated programming environments, or else available in a non-integrated tool-at-a-time mode. The Programmer's Interface has shown that many of these tools are language independent or only slightly language dependent and has demonstrated how such tools can be extended to handle a wide set of programming languages. It utilizes the programming tools (editors, file systems, debuggers, Programmer's Assistant [9], etc.) developed for one language (LISP [10]) for the development of software in other languages (e.g., ECL [11]). It has established interface requirements for other languages which would greatly reduce the effort required to transform these from simple interactive programming languages into an extensive programming environment. 39

5. Economics: in addition to the costs of creating an appropriate programming environment, addressed above, there are several economic factors which currently limit the use and utility of existing programming environments. Most machines are sized for their production requirements not their development ones. Hence, typically they do not contain enough mass storage for the files that would be required in an on-line environment, nor enough memory to support both the code being developed and the tools for that development. Additionally, access to the system is limited by the priorities of the production work load. Networking and economies of scale provide solutions by providing access to a system specifically designed and sized for software development and on which no production work load exists. Charges would be based on usage and development costs for the system spread over a much wider community of users because of the language and machine independence aspects of the system. In addition, very cost effective mass storage can be provided (by the Datacomputer [2]) which provides a trillion bit on-line memory at a cost of about a dollar per megabit per year. 40

SYSTEM DESCRIPTION

The hardware for the NSW, shown in Figure 1, consists of three logical components interconnected by the ARPA Network: The datacomputer, composed of a trillion bit store and a file management system in the Mass Storage component. The Execution Machine component is a set of machines responsible for running the program being developed and for collecting data on its execution. For each program being developed, the Execution Machine chosen is automatically the same as the production machine for that program. Thus, during development, a program is executed on the same (actually a copy of the) machine it will run on during production. This mechanism eliminates all machine-dependence compatibility issues at the cost of replicating the execution software in each machine for which this capability is desired and the cost of having that machine available in the NSW. On the other hand, it provides the great advantage of allowing the final component, the Interactive Machine (or machines), to be independent of the choice of production machine, thereby allowing it to handle a wider set of implementation efforts. This component contains most of the system's software and provides all of the facilities of the NSW except those described above.

The ARPA Network not only interconnects the NSW components, it also provides access for users to the system and supports a variety of terminals. However, the NSW will be oriented towards the use of high capacity video terminals.

Although the system is distributed across the ARPA Network, it is organized so that neither the user nor the component software modules are aware of this. The user sees a single integrated facility. The mechanisms described in the Framework Section enable modules either locally or remotely connected to communicate without knowing each others precise location.

TOOL INTEGRATION

Our discussion so far has concentrated on issues of tool access and integration of project data management; we have not yet talked about the tools themselves. It should be clearly understood that it is not the intention of the NSW to go into massive tool production. Initially, standard tools will be brought up on the various machines in the network, and novel ones will continually be brought up by the research organizations as they have been in the past. In time, improved versions of old tools or quite new ones may be developed specifically for the NSW, as the user's needs become clearer, but there is no present plan for major activity along these lines. 46

This reliance upon tools which already exist -- or will develop -- outside the NSW creates the obvious problem of integration of such foreign objects into the NSW. Here there are a number of issues: 47

* Arranging communication between the tool and the rest of the system (the Project Files, the local Console, etc.). 47a

* Providing standard linguistic conventions for programmers to use when talking to any tool: Surely the highest level command language utterances which invoke tools should be standard over the whole tool population. Whether lower level transactions with various tools can be standardized at reasonable expense is debatable. 47b

* Providing, for each tool, on line instruction in its use. 47c

* Constraining tool contact with the Project Files to be consistent with the File Manager's role as infallible guard. 47d

Certain of these problems must be met squarely: thus, the NSW will enable communication to happen, and will provide a consistent high-level command language. Others may be impractical to solve in any complete or consistent way: thus some tool may have so idiosyncratic a pattern of on-line communication that it would be prohibitively expensive to distort it to what might be called a standard form; on-line instruction might plausibly be quite complete for a small, unusual tool, but we have no intention of demanding a while CAI course in COBOL programming as a necessary accompaniment for a COBOL compiler. 48

The last issue raised may deserve some discussion. The point is that every effort to deliver filed text to a tool or to file text produced by a tool be subjected to whatever inhibitions the File Manager may then be authorized to impose and particularly by the requirement that

enough information be supplied to permit the books to be kept up to date. 49

Ideally, foreign tools should be "naturalized", by which we mean that they should be so modified that this information all gets transmitted behind the programmer's back, through negotiation between the tool and the File Manager: thus a tool might accompany each piece of text produced by a standard chunk of descriptive information. In practice, the modification of some tools may be impractically expensive; in this situation, attempts to file tool output will invoke a standard conversational program which extracts the necessary information from the programmer and blocks filing until it is sufficiently well informed. In effect, we will naturalize tool outputs, if it is too expensive to naturalize the tools themselves. 50

FRAMEWORK

This naturalization and integration will be effected by the framework which is designed: to allow the system to be quickly put together, to obscure the difference between local and remote communication, and to provide a mechanism for smooth, long-term growth for the inclusion of new facilities and improved cooperation and coordination among the tools within the NSW. This framework consists of two components. The first is a slight extension to the traditional subroutine invocation mechanism. It enables subroutine calls both within the same machine and across the ARPA Network to another machine. This mechanism determines which type of linkage to perform, utilizing the name of the routine being invoked to look up its location, using this location determines which type of linkage should be performed.

The second mechanism, and Executive, is more complex and is based on a combination of Actors [8], Ports [12], and Co-routines [13] (see especially Ref. [8]). The Executive acts as an exchange between what is happening and the modules that need to be informed. The Executive, when notified that an action has occurred (or is about to occur), is responsible for invoking (using the mechanism described above) all those modules within the NSW that need to know about this event.

Each of the modules within the NSW can be thought of as a self-contained asynchronous unit which utilize these communication mechanisms to keep informed about events which affect it and to announce its behavior to those concerned. Within this framework each module has four responsibilities: to identify those actions that it performs; to notify the Executive when one of these occurs (or is about to occur); to supply appropriate information to the Executive for interpreting this action; and to request notification from the Executive of other actions of concern.

This control structure is highly asynchronous, being driven by the actions that occur and the modules that need to be informed. Although it is specified in an interpretive form, the control structure can be "compiled" into normal intermodule invocations. The important point is that each module does not need to know what other modules are affected by its actions, but only has to agree to announce those actions.

Initially the NSW will be composed of large modules with few, if any, actions identified. As such, it will operate in a largely conventional manner without much cooperation between the modules. Over time, the coordination and cooperation between the modules will be gradually tightened through the replacement of these modules and the incorporation of new ones that identify and report more of their behavior and which utilize the framework to keep informed of the

behavior of other modules. In this way, the NSW can smoothly evolve toward a more unified and comprehensive set of capabilities. 56

ACKNOWLEDGEMENTS 57

The design reported here is the result of an extensive committee effort spanning several months and involving an on-going network conference as well as frequent meetings. We would like to acknowledge the effort and contribution to the NSW of these committee members: 58

Barry Boehm (TRW) 58a

John Brown (TRW) 58b

Michael Busch (CSC) 58c

F. J. Corbito (MIT) 58d

Peter Deutsch (XEROX PARC) 58e

Jerry Feldman (Stanford) 58f

Cordell Green (Stanford) 58g

J. C. R. Licklider (MIT) 58h

Tom Lippiatt (Rand) 58i

Barbara Liskov (MIT) 58j

Richard Watson (SRI) 58k

Clark Weissman (DaSDC) 58l

We would also like to acknowledge the inputs from the following people who took the time and effort to enlighten us on their software production problem and requirements: 58m

William Clark (NAVSHIPS) L/C Robert O'Keefe (USAF-ESD) Major Harold Arthur (USAF-ESD) Norman Glick (NSA) John Mott-Smith (USAF-ESD) Major Zara (USAF-ESD) 58n

REFERENCES 59

[1] Roberts, L. G., B. D. Wessler, "Computer Network Development to Achieve Resource Sharing," AFIPS Conference Proceedings, Vol. 36, 1970, pp. 543-549. 60

[2] Datacomputer Software Architecture, Datacomputer Project Working Paper 5, February 1972, Computer Corporation of America. 61

[3] Corbato, F. J., and Vyssotsky, V. A., "Introduction and Overview of the Multics System," AFIPS Conf. Proc. 27 (1965 FJCC), Spartan Books, Washington, D. C., 1965, pp. 185-196. 62

[4] International Business Machines Corp. 1969 (June), CP-67/CMS, Program 360D-05.2.005, Hawthorne, New York. 63

[5] Auslander, M. A., J. F. Jaffee, A. L. Scherr, and J. P. Birch, "Functional structure of IBM virtual storage operating systems," IBM Systems Journal, Vol. 12, No. 4, 1973, pp. 368-413. 64

[6] Bobrow, D. G., J. D. Burchfiel, D. L. Murphy, and R. S. Tomlinson (Bolt Beranek and Newman Inc., Cambridge, MA), TENEX, A Paged Time Sharing System for the PDP-10, August 1971, 29p. 65

[7] Balzer, R. M., A Language-Independent Programmer's Interface, USC/Information Sciences Institute RR-73-15, November 1973. 66

[8] Hewitt, C., P. Bishop, and R. Steiger, A Universal Modular ACTOR Formalism for Artificial Intelligence, 3IJCAI, 1973, pp. 235-245. 67

[9] Teiteman, W., Automated Programming - The Programmer's Assistant, FJCC 1972, pp. 917-921. 68

[10] Teitelman, W., D. G. Bobrow, A. K. Hartley, and D. L. Murphy, BBN-LISP TENEX reference manual, BBN Report July 1971. 69

[11] Wegbreit, B., "The ECL Programming System," FJCC 1971, pp. 253-262. 70

[12] Balzer, R. M., Ports - A Method for Dynamic Interprogram Communication and Job Control, The RAND Corporation, August 1971. 71

[13] Conway, M., "Design of a Separable Transition-Diagram Compiler," CACM, Vol. 6, No. 7, July 1963, pp. 396-398. 72

NSW documents vintage 1973

(J24716) 10-DEC-74 09:24;;;; Title: Author(s): Elizabeth K. Michael/EKM; Distribution: /EKM([ACTION]) RWW([INFO-ONLY]) DCE([INFO-ONLY]) JCN([INFO-ONLY]) NPG([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: EKM; Origin: < MICHAEL, NSWDESIGN.NLS;4, >, 5-DEC-74 11:26 EKM ;;;;####;

EKM 10-DEC-74 09:32 24717

old NSW document

This is the second of the documents from Bill CArlson.

old NSW document

PREFACE

As Military systems become more sophisticated and complex, the development and maintenance of computer software is becoming increasingly critical to the Air Force in terms of cost, reliability, and development time. This project addresses these issues by developing for operational Commands a totally new approach to software development. This approach is based on providing programmers and documentors with continuous access to a shared interactive facility which is completely separate from the Command's production hardware and is specifically designed for software and documentation production. This facility contains interactive tools running on several different machines and working together through a computer network with a uniform user interface for accessing and controlling the tools. Most of the software building process is performed by tools running on the most appropriate machines, only the actual execution of the software being developed need be run on the production machine or a copy of that machine.

Developers of software destined for many different production machines are thus able to share the tools available in the central facility. Each tool addresses a larger audience and users are able to pick the best tools available rather than being restricted to those which exist on their production machine. This wider audience and freedom of tool location make possible the creation of a competitive commercial marketplace in which private industry, with its own risk capital, produces those tools which it feels are economically viable and whose costs can be recovered by usage charges.

The first year and a half of this project is aimed at providing the basic system for accessing and coordinating tools and a useful, if limited, initial set of tools for two selected Air Force programming centers to demonstrate the viability of this approach to both Air Force users and potential industry tool vendors. This first phase will be supported entirely by the Air Force Data Automation Agency and ARPA. The later phases of the project will extend the availability of the facility to a wider set of Air Force users and will allow and encourage the incorporation of tools produced by private industry.

During its early stages of usage, the facility will address the cost and development time aspects of the Air

Force's software development problem by providing a continuously available interactive facility for programmers and documentors. Later, as a more complete and sophisticated set of tools gets incorporated into the facility, improvement of software quality and further cost reduction should be possible.

OUTLINE 8

I. Introduction
II. Long-Term Objectives
III. Background
  A. Related ARPA Projects
  B. Related Air Force Projects
IV. Short-Term Implementation
  A. Summary of Short-Term Objectives
  B. Discussion of Short-Term Objectives
  C. Size of Initial Demonstration
  D. Management Plan
V. Projected Actions to Achieve Long-Term Objectives
  A. Administration
  B. Tools
  C. Framework
VI. Budget
  A. Minimum Budget to Meet Short-Term Objectives
  B. Desirable Budget Items
VII. Projected Benefits
VIII. Schedule
  A. Milestones
  B. Projected Expansion of Number of Users
IX. Attachments 8a

I. INTRODUCTION

This project is a joint effort by the Air Force Data Automation Agency (AFDAA) and the Department of Defense Advanced Research Projects Agency (ARPA) to expedite the construction of computer software within DOD. The software production process includes such varied activities as problem analysis, system design, coding, testing and debugging, project control and accounting, documentation, and release and distribution of completed systems. Automated support can increase the efficiency of many of these activities. By doing the actual coding on-line and using an editor which understands the structure of the programming language, programming standards can be enforced and detailed program documentation prepared at the same time. Testing and debugging are simplified if the programmer can step through his program on-line, interactively examining and changing the values of variables, and even changing the flow of control. Test data generators and statistical tools which measure the time spent executing each part of the program help managers guarantee that the product is thoroughly tested. The preparation and distribution of system documentation can be expedited by text entry and publication capabilities which are not very different from those required to automate office and administrative functions unrelated to programming. Most of these tools are well understood; they exist in various combinations and with varying quality on a number of computer systems. But because tools are for the most part not easily transferable from one hardware system to another, economics, contractual barriers, and local scheduling priorities may force programmers to program with essentially no assistance.

With the exception of testing before the product is released, there is no reason why the target hardware must be used for program development. In fact, program development is a separate activity which should not be constrained by the same hardware restrictions as a production program which will run at many sites for a long period of time. A common commercial practice now is to use a large general purpose computer to compile and generate object code to run on mini-computers. An extension of that concept is needed so that programmers can use, through a common system, any available tool, be it an editor, compiler, test data generator or whatever, regardless of the specific hardware the tool executes on. Computer networks make such a concept feasible.

This project proposes to construct an integrated collection of tools which expedite the software production process. Each tool will be a program (or programs) which executes under one of the hardware/operating systems connected to the Advanced Research Projects Agency Network (ARPANET). A software system, called the Framework, will be constructed which provides a consistent interface to these tools and centralizes the accounting for their use. The set

of tools, together with the integrated interface, appropriate accounting and maintenance arrangements, and facilities which allow managers to control the use of tools and files, will be called the National Software Works (NSW). 12

To purchase and incorporate all desirable tools into the NSW would be very expensive. A large organization would be needed to examine all candidates, choose the most desirable ones, identify deficiencies, and fund the development of specific tools to satisfy the deficiencies. Instead, the success of the NSW depends on creating an open marketplace which will encourage vendors, on their own initiative and with their own capital, to incorporate new and useful tools into the environment. The Framework will have clearly documented procedures by which vendors can add new tools. After a vendor installs a tool and publicizes it, each user (or user organization) will decide how much to use it and pay a charge proportional to actual use. The NSW will include a catalogue which informs both users and potential vendors of existing tools and their costs. 13

There are two key uncertainties in this project. The technical uncertainty is whether the Framework can be designed so that there is no practical bound on the number of users who can be simultaneously interacting with the system. The NSW is to centralize accounting and control of files. Sufficient parallelism, backup, and recovery must be designed into the system so that reliability is no worse than if each group of users were to be operating its own computer. The management uncertainty is whether the NSW can be made sufficiently attractive to vendors to provide users with real choices among competing tools. 14

Recognizing these uncertainties, the project has been organized as a collection of interrelated subprojects, or tasks, where each task will produce something of intrinsic value. The choice of tasks to bring the NSW to a point where vendors will want to supply tools is discussed in detail in Section IV, the Short-Term Implementation Plan. Because the pieces of the NSW are themselves valuable, the risks of a catastrophic failure are small. On the other hand, if all tasks are completed as expected, the integrated system will guarantee that every Air Force programmer with access to the NSW has the use of the best available tools to do his job. 15

II. LONG TERM OBJECTIVES

16

The long range goal of this project is to develop a powerful, integrated software production facility which will enable the development of less costly and more reliable software systems which are adequately documented. Specific objectives are to:

17

A. Allow software development aids (tools) to execute on machines on which they run best, independent of the particular piece of hardware (target machine) for which the program is being written. Such a concept has two fold power. First, the systems programmer will be able to bring to bear on his problem the most appropriate tool or set of tools, regardless of what is implemented on his local machine. Second, the tool designer will be able to develop his tool primarily to run on the machine most suited to do the job, instead of worrying about being able to transfer it to a miriad of different machines. Thus, instead of designing for the lowest common denominator, he will be able to take full advantage of the special powers of his particular tool bearing host, allowing better tools than ever before feasible.

17a

B. Provide a consistent interface to tools including the automatic transfer of files as necessary for use and storage, so that it is reasonable, from a training and human factors standpoint, to use the powerful tools residing on several different hardware/operating systems in the course of a single project.

17b

C. Make tools available on a usage basis and decentralize the decisions of what tools to use for a particular problem, reducing the cost of using advanced and specialized tools and making them more economically attractive to both the developer and the user.

17c

D. Decentralize the decisions of what tools to include, making the NSW act as a competitive marketplace, with the competition spurring the development of better tools at lower costs.

17d

E. Drovide the project manager with much tighter controls over his software projects by providing access control mechanisms which will adhere to policies set by the project manager.These policies typically will specify who may access and/or change which modules, what crosschecks must be carried out whenever a module is changed, and what consequent actions are required for each change.

17e

III. BACKGROUND 18

A. Related ARPA Projects 19

The NSW incorporates products of other ARPA research programs, some complete and others on-going. While this approach results in an NSW budget which is a small fraction of what it would cost to achieve the objectives from scratch, it has the disadvantage of requiring coordination with constituencies whose uses for the products, and therefore their priorities for development, may differ from those of the NSW. Three components of the NSW will be either partially or completely funded by other programs: the filing system, the intercomputer communications protocols, and the user interface. This plan assumes that ARPA will coordinate the various programs in such a way that the interface between these three components and the rest of the NSW will not change drastically during the development program. Each of the three components is discussed in greater detail in the remainder of this section. Because they should contribute to the overall performance of the NSW, ARPA projects investigating computer security and network reliability are also discussed. 20

1. Filing System 20a

For its filing system, the NSW will use the Datacomputer which is being developed by the Computer Corporation of America (CCA) for ARPA. The Datacomputer will provide controlled access to mass storage devices with capacities in the range of a trillion bits. It is a very large data management system running on a dedicated PDP-10 computer. The Datacomputer's access mechanisms control the sharing of information among processes executing on different hardware and under different operating systems. Network traffic is reduced because processes can request logical subsets of files. An initial version of the Datacomputer software, operating on conventional disks, is running at CCA in Cambridge, Ma. An Ampex Tertiary Storage Device will be installed at CCA in JULY 1974 and will be integrated with the software. A different tertiary store, manufactured by Unicon and located at the NASA Ames Research Center in California, will also be controlled by a copy of the Datacomputer software. 21

2. Intercomputer Communications Protocols 21a

The ARPANET has a hierarchy of rules, or protocols, for intercomputer communication. The lowest level protocols involve reliable message exchange between ARPANET Interface Message Processors (IMPs). At higher levels, the protocols are concerned with functional interactions between the operating systems of computers attached to the network, or between programs running under those operating systems. For example, there are protocols by which a program running in one machine can log into another machine and initiate a program in

it. There are protocols by which one operating system requests another to transfer a file across the network. The NSW Framework development requires new ARPANET protocols to be defined. The protocols are discussed in detail in Attachment 2(see note B at the back) which is the proposal from Massachusetts Computer Associates to build the Framework Executive. The ARPA Network Working Group is the organization which defines standard function oriented protocols. 22

3. The User Interface 22a

ARPA has substantial current research studying man-machine interaction and human interfaces. The NSW will initially have only a simple interface which allows a user to specify a tool and the files it will use; initially each tool builder must provide an interface for all interactions required by the specific tool. The longer term goal is to simplify both the user's interactions with the NSW and the construction of the tool by providing a uniform way to initiate a tool, communicate with it, get help like determining what options are currently available, and examine the tool's results. The term "Front-End" is frequently used to refer to such user oriented subsystems. Attachment 3 is a discussion of the kind of Front-End which may eventually be produced by the other ARPA projects, and can then be integrated into the NSW. 23

4. Computer Security 23a

ARPA has a current research program in computer security. The TENEX operating system, under which the NSW Framework Executive and the Datacomputer software will both operate, is being made more secure. ARPA is also looking specifically at the security of the Datacomputer's access control mechanisms. Because the NSW centralizes access to so much information, privacy and security considerations are crucial. The NSW should be a major beneficiary of the computer security research program. 24

5. ARPANET Reliability 24a

ARPA is continually striving to improve the reliability of the ARPANET. This work is directed both at improving the reliability of the individual components and at providing redundant data paths which give the overall system higher reliability than the parts taken separately. Since the NSW cannot be more reliable than the ARPANET, it should derive substantial benefits from this program. Its design already has incorporated techniques for dynamic reconfiguration to utilize available components. 25

B. Related Air Force Projects 26

At the present time, six Air Force organizations have Terminal

Interface Message Processors(TIPS) which give them access to ARPANET host computers from remote terminals. The USAF Range Measurements Laboratory at Patrick AFB, Florida is the managing agency for the ARPANET communications service. A summary of ARPANET use by these organizations is contained in Attachment 4 . 27

Air Force Systems Command currently has a plan to link AFSC computers with the ARPANET. In the first phase, which is scheduled to be completed in July, 1975, three CDC6600 computers will be established as ARPANET hosts. The AFSC long-term plan calls for the investigation of a common network command language, accounting, and automatic resource sharing and file transfer; but the budget does not include funds for those efforts during FY75. Since the NSW will address those problems during FY75, these plans complement each other. 28

RADC has indicated an interest in installing its Multics and GCOS machines as ARPANET hosts. Any addition of government owned computers to the ARPANET makes the NSW more viable by increasing the number of available tools, and makes the NSW more necessary by proliferating incompatible login sequences, accounting systems, filing systems and command languages. AFDSC is especially interested in having access to an AF owned unclassified Multics system. 29

IV. SHORT-TERM IMPLEMENTATION 30

A. Summary of Short-Term Objectives 31

1. Five to ten personnel at the Air Force Data Services Center(AFDSC) and another five to ten at the Air Force Data Systems Design Center(AFDSDC) will be trained to use the NSW to prepare and publish administrative-type documents. 31a

2. AFDSC and AFDSDC personnel will gain experience using the ARPANET to manage a project by communicating and preparing correspondence for this project on the network. 31b

3. A mechanism will be created for taking documents from the NSW and publishing them on the Linotron at Wright-Patterson AFB, Ohio. 31c

4. Reliable access to the ARPANET and the NSW will be established for terminals located at AFDSDC and AFDSC. 31d

5. A Burroughs B3500 at AFDSDC will be connected to the ARPANET as a user host. A hardware/software interface will be developed to allow entry of jobs from the net. 31e

6. The foundation for the open-ended marketplace system will be laid by constructing a Framework and incorporating the above document preparation, publication, and message passing tools (items 1,2, and 3). This is the primary goal of this phase and will allow the evaluation of the concept and if the results indicate, the implementation of a full scale production version of the NSW. 31f

B. Discussion of Short-Term Objectives 32

The first phase of the NSW consists of three parallel tasks: build a skeletal Framework, incorporate a few tools to demonstrate the concept, and connect one AFDAA computer to the ARPANET. For each task, alternative implementation choices were available, but all three tasks are necessary parts of the overall plan. 33

1. Skeletal Framework 33a

The key to the NSW is a software system which acts as a Framework onto which an arbitrary number of heterogeneous tools can be attached. It will represent a small part of the total system cost, but the NSW cannot operate without it. Therefore, the Framework development must begin immediately. Attachment 2 is a proposal from Mr. Stephen Warshall, president of Massachusetts Computer Associates, to build a skeletal version of the executive (and control) portion of the Framework. Mr. Warshall identifies

the functions to be performed by the Framework Executive. It will be connected to the user interface portion of the Framework and the rest of the NSW through well-defined interfaces. 33b

A major issue regarding the Framework Executive is what hardware and operating system it is to be built under. The PDP-10 with the TENEX operating system was chosen for the following reasons: 33c

a. A PDP-10 with the TENEX operating system is a state-of-the-art machine for developing interactive software. There are many TENEXs on the ARPANET, and their implementation of network protocols is the best available. 33c1

b. A single Multics system, the one at M.I.T., is available on the ARPANET. It provides necessary operating system flexibility, but the machine is heavily loaded and the network protocol implementation is not as advanced as that of TENEX. 33c2

c. Other machines on the ARPANET lack both the TENEX operating system's flexibility and its standard and well-developed network protocol software. 33c3

Thus, the NSW Framework Executive should be developed to operate under the TENEX operating system. Note that use of TENEX for the Framework Executive does not restrict the choice of machines on which tools will execute. 34

The first Framework Executive will be skeletal in the sense that project control capabilities will not be fully developed, and in the short term no attempt will be made to balance the load on interchangeable resources. The initial Framework is expected to undergo an extensive tuning phase to improve efficiency and response time during the demonstration period. Although functional correspondence to design specifications is planned for the initial system, the system is complex enough, different enough from existing systems and patterns of usage, that no such guarantee can be given now concerning performance specifications. Considerable energy is expected to be spent tuning the system once these issues are better understood and the bottlenecks and inefficiencies discovered. However, it should not be considered a prototype, as the initial version will be expanded and modified to produce more advanced versions. 35

2. Initial Tools 35a

The first tools to be added to the NSW must be chosen with care. Their current availability on the ARPANET is equally as important as their usefulness to AFDAA. To minimize the potential loss

associated with a failure of the Framework development, as little money as possible should initially be spent developing software which depends on it. Thus, as explained below, document preparation, program editing, and remote job entry capabilities have been selected as initial goals; while compilation, debugging, and project management tools were not. 35b

a. Burroughs B3500 programming tools which would execute on a large-scale ARPANET computer were considered. Final testing must take place on the target machine, so a B3500 at AFDSDC would have to be placed on the ARPANET. The most desirable tools, such as a cross-compiler which runs on a larger, faster machine on the net and produces B3500 object code, do not exist and would have to be developed. The conclusion was that the actual compiling and testing of B3500 COBOL programs is not the best short-term objective for the NSW. However, centralized tools for program creation and editing, file management (through Framework Executive capabilities), and remote job entry to the B3500, will be part of the initial system. Additionally, AFDSDC should have some Burroughs B6700 time on the ARPANET so it can investigate using B6700 COBOL for preliminary syntax checking and testing of B3500 programs. 35b1

b. Coding and testing of Honeywell H6000 COBOL programs was rejected as a short-term goal. No GCOS machine is on the ARPANET, so to do H6000 programming requires placing an H6000 on the Net and the procurement or development of the necessary tools. 35b2

c. The project control facilities of the NSW Framework will contain a centralized record of each project's status and history: what programs and data exist in the project's files, where they came from and when, how they relate to one another, and so on. Project management tools will eventually be one of the strongest components of the NSW. But for them to work, most of the project must be developed on-line. Until the NSW is populated with sufficient tools, the project management facilities will not have enough information to adequately control the behavior of programmers. The initial Framework development will include only primitive validation and access control procedures, and simple accumulation of accounting information. 35b3

d. Tools to prepare and publish documents already exist on the ARPANET. A substantial fraction of AFDSDC resources are devoted to the publication and distribution of manuals documenting standard Air Force data systems. At the same

time, AFDSC is heavily involved in automated photocomposition systems, and has on-going projects to investigate new ways to provide text processing support to Air Staff users. Thus, document preparation and publication tools satisfy the dual criteria of availability and usefulness. 35b4

1) ARPA has been funding the development at Stanford Research Institute of an on-line text processing system called NLS. Rome Air Development Center (RADC) is also supporting that effort. NLS is near completion and ready for operational use and testing. It is available at two sites on the ARPANET. 35b4a

2) The primary disadvantage of NLS is the cost of its photocomposition facility. AFDSC is using the Linotron at Wright-Patterson Air Force Base for photocomposition. Wright-Patterson AFB has an ARPANET terminal interface. It seems desirable to add a magnetic tape to the Wright-Patterson AFB ARPANET interface, and then use the Linotron to publish documents prepared on NLS. Similarly, AFDSDC is investigating micropublishing. When an Air Force owned COM facility is available, it can be used in place of the commercial COM facility available with NLS. It must be emphasized that each photocomposer or COM machine is somewhat unique, and special programming is required each time a new machine is to be used. 35b4b

35b5

3. Machine Connections 35c

In order to do programming on the ARPANET, the target machine must be connected. Connecting a new operating system to the ARPANET is a substantial undertaking which requires a hardware interface and new software to interpret the network protocols. Attachment 5 is a discussion of alternatives for connecting a Burroughs B3500/4700 and a Honeywell H6000 to the ARPANET. Attachment 6 is a proposal from Burroughs for the hardware interface. Honeywell submitted the manual for the PWIN IMP Interface System (IIS) as the proposed hardware interface to connect an H6000 to an IMP. Consultants working for ARPA have suggested other alternatives which seem preferable and they are being investigated. The first priority should be to connect a B3500. Ideally, work should begin to connect both the B3500 and the H6000 so that when vendors are given the opportunity to add tools to the NSW, they will take both machines into account. 35d

C. Size of Initial Demonstration 36

A demonstration of the NSW concept, including the executive, a prototype user interface, and document preparation and publication tools, is scheduled for July, 1975. An important issue is how many simultaneous users the system should support at that time. It is important to do substantive work, but it It is also important not to disrupt too much on-going work. Between five and ten users each at AFDSDC and AFDSC seems to be a reasonable compromise. The AFDSDC users will prepare the documentatiion for an automated data system. AFDSC users prepare correspondence and will use the system to publish an AF manual. 37

D. Management Plan 38

1. Steering Committee. There will be a continuing need to evaluate the progress of the NSW as it develops, formulating new short-term objectives and initiating new tasks. This overall planning and policy formulation role will be performed by an NSW Steering Committee consisting of two representatives each from AFDSDC, AFDSC, and ARPA. Present members of the steering Committee are Major Tony Baggiano and 2Lt. Larry Crain from AFDSDC, Major Jim Lloyd and 1Lt. William Carlson from AFDSC, and Mr. Stephen Crocker and Dr. Robert Balzer(an employee of the Information Sciences Institute of the University of Southern California) representing ARPA. Mr. Al Mayhan is Maj. Baggiano's alternate and attends most meetings. 38a

2. Project Management. It is necessary to have project officers who will maintain control over all aspects of the NSW effort on a day to day basis. These individuals will make certain that regular progress reports are presented to the Steering Committee, maintain coordination among the various tasks, identify potential problems, and propose new short term objectives which further the overall goals of the project. ARPA will appoint one Project Officer and AFDAA will select one Project Officer each from AFDSC and AFDSDC. Initially these will be Mr. Crocker, Lt. Carlson, and Lt. Crain. 38b

3. Project Engineer. Under guidance of the project managers, the project engineer will coordinate the technical interfaces among the various components of the NSW, review the development of the tools, front-end and network connections, and facilitate communication with industry groups which need to evaluate the NSW in order to prepare tools for it. Dr. Balzer will be the project engineer. 38c

4. Specific Tasks. The specific responsibilities for the various facets of the short-term NSW implementation are as follows: 38d

a. Procurement: ARPA has overall responsibility for all procurement actions, and will select the appropriate vendors. For hardware and software which is to be installed at their sites, AFDSC and AFDSDC will assist ARPA in preparing necessary paperwork and forwarding it to the contracting office of ARPA's choice. 38d1

b. Connection of B3500 and H6000 to ARPANET: AFDSDC will have primary responsibility for taking actions necessary to install a Burroughs B3500(or B4700) and a Honeywell H6000 as ARPANET hosts. ARPA will provide consulting services as required, and select vendors for necessary hardware procurements and software contracts. 38d2

c. Document Preparation and Publication Tool Implementation: AFDSC will be responsible for implementing necessary software to publish documents from the ARPANET using the LINOTRON at Wright-Patterson AFB. AFDSC and AFDSDC will be responsible for formulating in-house training plans to teach the use of NSW document preparation tools. When user interface problems are identified, ARPA will work with AFDSC or AFDSDC and the appropriate contractor to correct them. 38d3

d. Computer Services at ARPANET Host Sites: ARPA will see to it that the following computer services are provided: 38d4

1) Accounts for AFDSC and AFDSDC users at ARPANET Host Sites to investigate available software. Known requirements exist for computer time on the 360/91 at UCLA and the B6700 at the University of California at San Diego. The $200,000 budget for computer time is intended to be divided between this item and item 2. 38d4a

2) Computer time on the hosts to be used for document preparation and publication. Availability for up to five simultaneous AFDAA users must be guaranteed, the mean time to failure must be greater than eight hours, temporary files should not be lost when the system crashes, and the overall availability of the system should be greater than 95% during 0800 to 1800 daily. 38d4b

3) Computer time for the development and operational testing of the framework and the Datacomputer. 38d4c

e. NSW Framework Development: ARPA will have overall responsibility for the development of a Framework which guarantees the long-run viability of the NSW. The four facets of the system are the central framework, necessary

modifications to hosts where tools will execute, the user interface, and the Datacomputer. The AFDAA Project Officers will work closely with ARPA in this effort. 38e

f. Coordination With Other ARPA Projects: One of the real potentials of the NSW is its ability to assimilate the products of other ARPA funded projects and make them available to operational users. The Datacomputer and the Management Systems Technology projects are two which are known to be relevant to the NSW. The Project Engineer will assist the ARPA Project Officer in determining how best to integrate these efforts. 38f

g. Coordination With Other Air Force Projects: The AFDSC Project Officer will maintain contact with the project officers for Air Force projects involving the ARPANET. 38g

V. PROJECTED ACTIONS TO ACHIEVE LONG TERM OBJECTIVES 39

In the long term, the NSW should become a self-sustaining system which gracefully accepts new users, new tools and new hosts. The short term plan outlined above addresses some of the technical issues and provides for an early check on the utility of the NSW concept, but much work will remain before the full capabilities of the NSW can be exploited throughout DOD. It is planned to spend the bulk of the next year on the short term plan, and develop a plan to achieve the objectives stated in Section II at a pace consistent with the progress and assurance of success of the short term efforts. A longer term plan with the same level of detail as the short term plan should be available in December 1974. The following action areas have been identified for consideration in the preparation of that plan. 40

A. Administration 40a

A concerted effort will be made to involve other parts of DOD. It is expected that a different form of project management will be necessary in the long-run, and this issue will have to be addressed . How to implement the usage based charging concept within the constraints of the DOD procurement regulations is one of the more challenging problems, and must be discussed in detail in the long-range plan. 41

B. Tools 41a

The range of tools in the initial system will be quite limited. Additional tools to populate the NSW will be obtained from three sources. First, an informational meeting will be held in August 1974 to present the NSW concept to industry and attempt to elicite their interest in providing tools on a usage basis. A more formal meeting at which the government presents the exact procedures for adding tools to the NSW should be held in November 1974. The hypothesis is that most standard tools will be procured by this method. A second source of tools will be the research community. The basic research offices for the three Services, the AF Office of Scientific Research, the Office of Naval Research, and the Army Office of Research are forming a joint committee to guide software research. ARPA has been asked to join that committee. If contractors use the NSW in the process of developing new systems, and then install the products as NSW tools, a rapid transfer of new technology to operational organizations could occur. The NSW's catalogues would provide an indication of missing capabilities and suggest areas for development both by industry and DOD. Finally, there will probably be some tools which do not currently exist, but which could be developed with little risk and have high payoffs. Examples might be a cross compiler from a large computer to small ones or a simulator for a

proposed computer so that the software-first design approach can be used. An economic analysis of each proposal of this type can be made, and if it is favorable, a funded development effort can be initiated. 42

C. Framework 42a

The initial skeletal NSW Framework will need to be expanded in several ways. The accounting system must accurately charge for all resources used so that the marketplace can be created. This will require close cooperation between the NSW and the operating system of the tool bearing hosts. The terminal handler, or front-end, will become much more sophisticated and will begin to act as an assistant for the user correcting his (simple) mistake, providing tutorial service, controling his terminal, remembering previous commands and reissuing them with or without modifications when asked, expanding his macros, and maintaining a description of the preferences for various system options. The administrative control will be tightened as a better model of the programming and documentation process is created within the system and more and more of the user activities automatically provide data for this model. Similarly the validation mechanisms will become more discriminating as better categories and attribute descriptions for tools, files, and users are entered into the system. 43

These increased capabilities are beneficial in several ways. Not only do they directly benefit the users, they facilitate the creation of new tools by removing the need for corresponding capabilities within each tool, and also provide a management aid for industry and DOD by suggesting areas of the programming and documentation process which are not adequately handled by existing NSW tools. 44

VII. PROJECTED BENEFITS

The potential benefits from the NSW are enormous. Each AF organization can have easy access to tools which run on a variety of kinds of hardware and under many different operating systems. Installations which have more than one computer system will be able to easily move files among them over the network, instead of maintaining multiple copies. When procuring new systems, the requirements to make them interface with existing tools are well defined; a hardware connection to the ARPANET and software to satisfy both ARPANET and NSW protocols. The only software development tools which must be replicated for every type of hardware are those directly related to the creation and testing of object code. The existence of language specific tools which are to be used to produce code for several operating systems, for example a COBOL editor or shorthand whose output is a COBOL program, will tend to enforce standardization and machine independence where it is possible.

VIII. SCHEDULE 48

A. MILESTONES 49

PLAN APROVED, INHOUSE EFFORT BEGINS	APRIL 74	49a
INITIATE PROCUREMENT OF AFDSC TERMINALS, AFDSDC TERMINALS,ANTS,AND IMP	MAY 74	49b
CONTRACT FOR FRAMEWORK EXECUTIVE LET	MAY 74	49c
INITIATION OF B3500 INTERFACE PROCUREMENT AND INHOUSE PROGRAMMING EFFORT	JULY 74	49d
CONFERENCE OF SOFTWARE VENDORS	AUG 74	49e
SECOND VENDORS MEETING	NOV 74	49f
PLAN FOR PHASE II RELEASED	JAN 75	49g
INITIATION OF H6000 INTERFACE PROCUREMENT AND INHOUSE PROGRAMMING EFFORT	JAN 75	49h
CONTRACT FOR FRAMEWORK EXTENSIONS LET	MARCH 75	49i
LINOTRON PUBLICATION TOOL OPERATIONAL	APR 75	49j
DEMONSTRATION NSW BEGINS OPERATION	JULY 75	49k
INITIATE PROCUREMENT OF AFDSC IMP AND ANTS	JULY 75	49l
REMOTE JOB ENTRY FOR B3500 ONLINE	AUG 75	49m
REMOTE JOB ENTRY AND TERMINAL ACCESS FROM NET TO H6000	DEC 75	49n

B. Projected Expansion of Number of Users 50

Assuming the July, 1975 demonstration is successful, decisions will have to be made as to how fast to expand the NSW. The two expansion paths are adding more users and adding more tools. The fundamental technical barrier to the availability of the NSW for continuous service for large numbers of users is the reliability of the Datacomputer. All other NSW software will be written so that multiple copies can run on as many systems as necessary to handle the load. The Datacomputer itself will exist at two ARPANET nodes and plans are underway for a "mini" version of the system which uses conventional disc storage. Other barriers to NSW expansion will be the procurement of additional terminals and PDP-10 computers and the

connection of new DOD host computers. A reasonable expansion path might be as follows: 51

July 1975	20 users
July 1976	60 users
July 1977	200 users
July 1978	1000 users 51a

Thus by mid-1978 the system would be able to support all programmers at AFDSDC plus a substantial number of users at AFDSC and from other sites. 52

EKM 10-DEC-74 09:32 24717

old NSW document

(J24717) 10-DEC-74 09:32;;;; Title: Author(s): Elizabeth K. Michael/EKM; Distribution: /EKM([ACTION]) DCE([INFO-ONLY]) RWW([INFO-ONLY]) JCN([INFO-ONLY]) NPG([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: EKM; Origin: < MICHAEL, NSWPLAN,NLS;6, >, 9-DEC-74 16:34 EKM ;;;;####;

Clear (Tty Window) command

When this command is used, it clears your display window instead of the tty window. 1

POOH 10-DEC-74 11:12 24718

Clear (Tty Window) command

(J24718) 10-DEC-74 11:12;;;; Title: Author(s): Ann Weinberg/POOH;
Distribution: /FDBK([ACTION]) POOH([INFO-ONLY]) ;
Sub-Collections: SRI-ARC FDBK; Clerk: POOH;

JAKE 10-DEC-74 11:39 24719

Weekly charge sheets on Project 3803

I would like to have a complete set of the weekly project sheets on project 3803 by Weds, 12/11/74. From here on in I would like to receive the project leader copy of these sheets each week when they come in. This is a written reminder to get this hopefully into your immediate stack. Thanks, Jake 1

JAKE 10-DEC-74 11:39 24719

Weekly charge sheets on Project 3803

(J24719) 10-DEC-74 11:39;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /SLJ([ACTION]) MEH([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

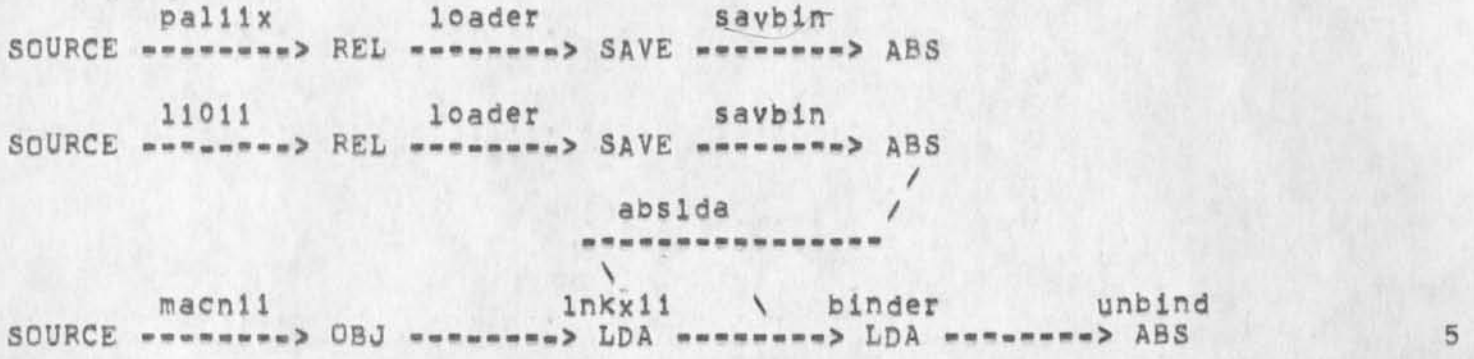
bootstrap 11 procedures

The following picture represents the flow control to compile, link and get ready for loading programs on the pdp-11. Capital words represent fource file types, and lower case words represent processors. 1

The MACN11 path (MACN11 source compiled by macn11 compiler producing OBJ files which are linked by lnkx11 producing LDA files which are bound by binder also yielding an LDA file which is input to unbind which yields an ABS file) will only be used by us for generating new versions of ELF. 2

Source programs written in either PAL11X or L1011 are compiled by the appropriate compiler and REL files are produced. These REL files are then linked using the TENEX loader, producing a SAVE file. This save file is then processed by the savbin program which outputs an ABS file. This is where you stop if you are only interested in running stand-alone programs on the 11, i.e., programs that do not run under ELF. If this is not the case, then the resulting ABS file must be processed by abslda to yield an LDA file that can then be bound with ELF and subsequently unbound, by unbind, to yield an ABS file that can be loaded in the 11. 3

Regardless of how the final ABS file is arrived at, it must be called <ELF>11PGM.ABS to be loaded, in the 11, by our bootstrapping procedures. 4



5

The bootstrap loading sequence is as follows: 6

key in on the 11 the bootstrap loader 6a

run the bootstrap loader to read, from the tty tape reader, the absolute loader paper tape 6b

run the absolute loader to read, from the tty tape reader, the ELF loader paper tape 6c

The ELF loader will start itself up, send a ^C to the 10, log in as

user ELF, start the 10 program C1011, and will then work in conjunction with C1011 to load the program <ELF>11PGM.ABS. The program <ELF>11PGM.ABS will start automatically if the load is successful. After the load is done, C1011 will detach (necessary so that the 11 doesn't see the logoff message) and then log itself out. 7

KEV 10-DEC-74 13:35 24720

bootstrap 11 procedures

(J24720) 10-DEC-74 13:35;;;; Title: Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /NPG([INFO-ONLY]) RWW([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: KEV; Origin: < VICTOR, 11-PROCEDURES.NLS;1, >, 10-DEC-74 13:32 KEV ;;;;####;

JBP 10-DEC-74 15:53 24721

Line at a Time Terminals

Has anyone tested your suggested solution to the line at a time terminal interaction mode on a real 2741 ?
--jon.

1

Line at a Time Terminals

(J24721) 10-DEC-74 15:53;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /CHI([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

JAKE 10-DEC-74 17:45 24722

Identification subsystem

The Identification subsystem does not seem to be loadable. Was this program not brought up with the new system. Also, how about all the other NIC programs that just were just upgraded - do they run or not? 1

Identification subsystem

(J24722) 10-DEC-74 17:45;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /FEEDBACK([ACTION]) RWW([INFO-ONLY]) KJM([INFO-ONLY]) CHI([INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: SRI-ARC FEEDBACK; Clerk: JAKE;

KIRK 10-DEC-74 17:51 24723

Where to send help stuff

Let me know if there is anything funny with this message, I sent it 'unrecorded'. I dn't know what happens in this case.

KIRK 10-DEC-74 17:51 24723

Where to send help stuff

The proper ident to be sure help suggestions go to the right people (currently just me), is the group ident "HELP". 1

KIRK 10-DEC-74 17:51 24723

Where to send help stuff

(J24723) 10-DEC-74 17:51;;;; Title: (Unrecorded) Title:
Author(s): Kirk E. Kelley/KIRK; Distribution: /FEEDBACK([INFO-ONLY])
; Sub-Collections: SRI-ARC FEEDBACK; Clerk: KIRK;

Proposal ISU 74-253 NLS Workshop Support for AFAA

Signed hardcopy printed today, to be sent to RADC and AFAA tomorrow 12/24

I INTRODUCTION

A. Brief scope statement

The purpose of this proposal is to request support for continued use of knowledge workshop technology developed at the Augmentation Research Center by the Audit Agency (AFAA). The service would be used by those AFAA-selected people who are willing to undertake exploratory use of knowledge workshop techniques through continued use of the on-line system (NLS) at Office-1.

The support is required for two activities: computer services and technical services.

The computer services are being supplied through the ARPANET to geographically distributed user groups from the computer facility maintained and operated by Tymshare, Inc under a subcontract with ARC. As prime contractor, ARC handles all service subcontracts.

The technical services provided by ARC personnel have the following objectives:

Maintain and update the "utility" version of our application software (NLS).

Support the user groups in learning how to use these tools.

Descriptions of the applications being suggested for exploratory use are given in a paper by Engelbart, Watson, and Norton [3] and in an earlier paper by Engelbart [2]. Copies of these documents are included with this proposal as Attachments A and B.

B. Organization of this Proposal

This proposal is divided into two parts, each of which is broken down into several sections.

Part One is the Technical Proposal, covering the proposed work and its background and context.

Section I is the introduction.

Section II is a summary outline of proposed project activity.

Section III is an extended discussion of proposed project activity.

Section IV is a list of selected references.

Part Two contains the Contractual Provisions, with sections covering such topics as estimated time and charges, reports, contract form, acceptance period, and a cost estimate with supporting schedules.

The Attachments contain additional supporting material.

C. ARC's "Community Plan"

Introduction

ARC is a one-organization community of researchers and system developers, supported by several different contracts. The research and development activities of ARC are aimed at exploring the possibilities for augmenting individuals and groups in the performance of knowledge work with the help of computer aids. These aids range from off-line batch to on-line real-time. Exploratory development and operation of augmentation systems have been our substantive work.

ARC's Research and Development Strategy

A new stage of application has now been established with the first year of Workshop Utility service almost completed. We are involving a wider group of system users so that we can begin to transfer the results of our past work to others, and so that we can obtain feedback needed for further evolution from wider application than is possible in our Center alone. We have been providing Workshop support Service to selected groups who are willing to take extra trouble to be exploratory, but who:

1) are not necessarily oriented to being workshop system developers (they have their own work to do),

2) can see enough benefit from the system's application and from the experience of trying it so that they can justify the extra risk and expense of being "early users," and

3) can accept our assurance that reliability, system stability, and technical application help will be available to meet their conditions for risk and cost.

Establishment of a Workshop Utility and promotion of the type of

service work proposed herein are part of ARC's long-term commitment to pursue the continued development of augmented knowledge workshops in a pragmatic, evolutionary manner. Note that our last few years of work have concentrated on the means for delivering support to a distributed community, for providing teleconferencing and other basic processes of collaborative dialogue, etc.,--consciously aiming toward having experience and capabilities especially applicable to support remote and distributed groups of exploratory users.

II SUMMARY OF PROPOSED PROJECT ACTIVITY

The proposed project work will include:

Providing training to AFAA-selected users as appropriate in the use of Display NLS (DNLS), Typewriter NLS (TNLS), and Deferred Execution (DEX) software subsystems.

Providing technical assistance to an AFAA-selected "workshop architect" in the formulation, development, and implementation of augmented knowledge work procedures within user groups.

Providing appropriate terminal equipment for AFAA use as mutually found to be necessary.

The technical assistance will include help in the development of NLS use strategies suitable to the client's environment and procedures within its organization for implementing these strategies.

The service will also include the availability 20 hours a day, 7 days a week of Workshop Utility service via the ARPANET from a PDP 10 TENEX system operated by commercial facility management. This expanded schedule is offered in response to requests from users.

III EXTENDED DISCUSSION OF PROPOSED PROJECT ACTIVITY

A. Objective

The objective of this effort is to work with AFAA personnel in the mutual development and use of procedures, methodology, software features, and other on-line tools; and in the training of users in NLS that will allow their exploratory use of our Workshop system. This objective has the following key components:

1) Building an AFAA user group whose members will find real value in applying the service, and whose participation will contribute to AFAA research goals both directly (by making the users' AFAA-related activities more effective) and indirectly (by accelerating the maturation and acceptance of augmented knowledge workshop techniques).

2) Developing ARC's know-how and capability for integrating innovation with new-development transfer.

B. Background

The Augmentation Research Center has developed, over a period of years under government sponsorship, a general-purpose interactive augmentation system centering about what we now call an "Augmented Knowledge Workshop," abbreviated below as "Workshop." The goal of ARC's work has been to evolve a prototype Workshop system that will significantly improve the performance of individuals and teams engaged in knowledge-work activities, where the Workshop "system" involves daily use of coordinated tools, procedures, methodologies, and languages.

For further background discussion, see [2] and [3], and the references in Section IV.

While the discussion in Attachment B is oriented toward communities of discipline or mission oriented users, the same types of services and knowledge workshop orientation apply to individuals and groups of workers in a local environment.

C. Scope of Proposed Work

Introduction

The types of workshop services that we are beginning to support at varying levels of capability are described in [3] under the headings:

Collaborative Dialogue
Document Development, Production, And Control
Research Intelligence
Community Handbook Development
Computer-Based Instruction
Meetings And Conferences
Community Management And Organization
Special Knowledge Work By Individuals And Teams

Our present capabilities in the above areas are briefly indicated in [2] and [3]. For each area, there is an immediate applicability of the basic NLS provisions for composing, modifying, studying, publishing, and collaborating, and we have additional special provisions specifically supporting almost every area.

Technology Transfer

We are beginning to transfer technology from our local group of experienced users to a wider group of inexperienced, geographically separate users. This technology consists of on-line software capabilities; a coordinated repertoire of on-line-assistance tools; associated concept and language additions dealing with the tools and with the information organization and task processes associated with their use and new aspects to intragroup organization and working methodology. Training a group in these new matters is necessary to the transfer; and to help others learn to train people in the new technology requires a transfer of the additional technology used to support the training.

For any group of users we expect evolutionary growth of their Workshop service application, in both quantity and range. This growth will take guidance and support of the sort that in the commercial computer world would be offered by the applications specialists and "systems engineers."

Services Offered

The proposed Workshop Utility service consists of two

components: computer support and people support. We discuss these components in detail below.

Computer Services

The Underlying Computer Service Support

Starting the second year of service (January 18, 1975), we are offering a Workshop Utility version of ARC's on-line system (NLS), serviced over the ARPANET, at least 20 hours a day, seven days a week. NLS features are described in the documents listed in Section IV.

This service is provided by a computer system operated and managed by a commercial timesharing utility company rather than from a system directly operated by ARC. There are two important reasons for this arrangement:

1) A commercial firm has the experience, facilities, leverage on vendors, and redundant equipment that make possible more reliable service than can be produced in a research and development environment.

2) It will be possible to expand the service in a more flexible manner in increments of whole or partial machines as usage grows.

Service Partitioning

We are now using a "group allocation" scheme for partitioning on-line access and service among groups of users. This guarantees each group its fair share of access to system resources while preserving both adequate responsiveness and independence for each group to plan its own usage loading. During this coming year, we plan to further develop the resource allocation system, working toward allocation of central processing unit (CPU) time, rather than login access.

File Privacy

The Workshop Utility provides the necessary standard TENEX software and facility operating procedures to ensure some privacy of file access. In addition, user-controlled NLS privacy features allow useful dialogue attended with flexible privacy restrictions. However, it is important to note that the visibility and availability of planning information and other recorded dialogue in ARC's currently open Journal System provide

some of the more significant potential of our Workshop system.

We assume that ARC on-line-service personnel may occasionally have to access clients' user files (at a client's request only) as required from an operational standpoint; however, other users of the Workshop Utility Service will be denied read, write and list access to a client's files, unless he specifically releases files for general use.

People Support Services

We are still learning about the amount and nature of people support services that a successful Workshop Utility needs, particularly in the direct client support category. The levels specified in this proposal seem to us to be minimal. Charges for such service will be made as delivered to each client.

Overhead Services

The entire operation, including the interface between the Utility and the clients, needs competent administration.

Documentation of the basic user features of the system and of their application techniques needs to be complete and will have various special versions tailored for particular types of users.

The version of NLS that runs on the Utility needs maintenance and quality assurance. A systematic means is being provided for features found useful in the development version of the system to be integrated into the version running on the Utility. This includes the handling of user feedback, a significant effort on the part of ARC Utility staff, providing service to users and important input to system builders.

Clerical support of various types is needed.

Direct Client Support Services

Our clients' users must be trained to varying levels of competence, depending upon the nature of their jobs and the tasks they perform. New procedures and methods will have to be developed and learned to allow effective use of the system in their working environments. Specifying

these procedures will require help in analyzing the group's needs and present operations.

Therefore the following types of necessary services will be provided.

Assistance in training Utility clients to make special use of the system for applications that are peculiar to their user environments.

Assistance to Utility clients in developing related documentation, procedures, records, and methods as needed locally to support their special use of the system.

Help for the above areas will come in several forms:

Sessions at SRI for training and application-system design.

Temporary residency of SRI personnel at client sites to offer analytic or design help and training.

"Circuit riders" who periodically visit client sites to discuss problems, receive feedback on how to improve the service, and offer training or analytic help.

IV SELECTED REFERENCES

1 ARC 3906, D. C. Engelbart, "Augmenting Human Intellect: A Conceptual Framework," Summary Report, Contract AF 49(638)-1024, SRI Project 3578, Stanford Research Institute, Menlo Park, California, AD 289 565, October 1962.

2 ARC 12445, D. C. Engelbart, "Coordinated Information Services for Discipline- and Mission-Oriented Communities," Stanford Research Institute, Augmentation Research Center, 12 December 1972. Also published in "Time Sharing: Past, Present, Future," Proceedings of the Second Annual Computer Communications Conference at California State University, San Jose, California, January 24-25 1973, pp 2.1-2.4.

3 ARC 14724, D. C. Engelbart, R. W. Watson, J. C. Norton, "The Augmented Knowledge Workshop," AFIPS Proceedings National Computer Conference, June 1973.

Proposal ISU 74-253 NLS Workshop Support for AFAA

I ESTIMATED TIME AND CHARGES

It is proposed that the work outlined herein be performed during a period of twelve months commencing 18 January 1975.

The proposed project will result in Workshop Utility service being made available to offices selected by AFAA.

The costs of the total Workshop Utility service will be accounted for separately by the Institute, with the amount charged to AFAA under this contract being determined as a proportion of the total common cost of the Workshop Utility operation based on its availability for AFAA-directed use together with direct charges for people services as incurred.

We propose to provide guaranteed access to one user access connection (jobslot) 20 hours per day, 7 days per week, from the start of the contract period until 17 January 1976.

Pursuant to the provisions of ASPR 16-206.2, attached is a cost estimate and support schedules in lieu of the DD Form 633-4.

The estimated costs shown in the cost attachment are for the total Workshop Utility service operation. Costs expected to be borne by AFAA over twelve months are estimated to be about $40,000 as shown in the attached cost estimate. We anticipate partial funding by AFAA to cover FY'75 to be followed by funding of the remaining FY'76 period as was the case in the current contract.

II UTILITY COMPUTER SUPPORT SUBCONTRACT

Tymshare, Inc. in Cupertino, California was selected by the Institute as the Computer Support subcontractor for the first year of service. A formal subcontract is being negotiated for continuation of the service through the second service year.

III REPORTS

Because of the support nature of the efforts proposed herein, there will be no technical reports produced under this contract. Rather, documentation will be provided along the lines outlined below.

The technical documentation will include:

TNLS and Deferred Execution User Guides and updates

DNLS User Guide and updates

IV CONTRACT FORM

Because of the nature of the work proposed, it is requested that any contract resulting from this proposal be awarded on a cost-plus-fixed-fee basis.

V ACCEPTANCE PERIOD

This proposal will remain in effect until 18 January 1975. If consideration of the proposal requires a longer period, the Institute will be glad to consider a request for an extension of time.

COST ESTIMATE FOR SECOND YEAR WORKSHOP UTILITY SERVICE
(total OFFICE-1 facility)

Personnel Costs

Supervision	1000	hrs.	
Prof	6867	hrs.	
Technical	2999	hrs.	
Clerical	1000	hrs.	
Total Direct Labor			$ 84,337
Payroll Burden @ 29.0 %			24,458
Total Labor and Burden			108,795
Overhead @ 107.0 %			116,411
Total Personnel Costs			225,206

Direct Costs

Travel		16,051
27 trips Wash @ $368 =	$ 9,936	
122 Days Subsistence @ $42.50=	5,185	
Auto Rental 62 days @ $15 =	930	
Communications		3,000
Materials and Supplies (tape, paper)		3,000
Utility Online Support Subcontract		710,280
[256k core, 3 drums, 20hrs/7days		
12 mo @ $ 59,190 = $ 710,280]		
Documentation Costs		3,997
Total Direct Costs		736,328
Total Estimated Cost		961,534
Fixed Fee		57,692
Total Estimated Cost Plus Fixed Fee		$1,019,226

AFAA CONTRACT SHARE: 1 slot = $ 40,000

See following Schedules.

Proposal ISU 74-253 NLS Workshop Support for AFAA

SCHEDULE A
DIRECT LABOR

Direct labor charges are based on the actual salaries for the staff members contemplated for the project work plus a judgmental factor applied to base salary for merit increases during the contract period of performance. Frequency of salary reviews and level of merit increases are in accordance with the Institute's Salary and Wage Payment Policy as published in Topic No. 505 of the SRI Administration Manual and as approved by the Defense Contract Administration Services Region.

SCHEDULE B
OVERHEAD AND PAYROLL BURDEN

The payroll burden rate is based on the Institute's best prediction as to financial performance for the calendar year 1975. The overhead rate has been found acceptable by the Department of Defense for billing and bidding purposes for the calendar year 1974. We request that these rates not be specifically included in the contract, but rather that the contract provide for reimbursement at billing rates acceptable to the Contracting Officer, subject to retroactive adjustment to fixed rates negotiated on the basis of historical cost data. Included in payroll burden are such costs as vacation, holiday and sick leave pay, social security taxes, and contributions to employee benefit plans.

SCHEDULE C
TRAVEL COSTS

Air fares and car rental rates are established in the current Official Airline Guide.

Domestic subsistence rates and travel by private auto are established standards based on cost data submitted to DCAA.

SCHEDULE D

DOCUMENTATION COSTS

Report costs are estimated on the basis of the number of pages of text and illustrations and the number of copies of reports to be produced, in accordance with the following rates per page:

Editing	$ 2,53	
Composition	2,66	
Coordination	,72	
Proofreading	,77	
Illustration	21,92	
Press and Bindery	,022	per impression

The following is a breakdown of the estimated cost of report production:

Text preparation, 439 pages at $ 6,68 per page (including editing, composition, report coordination and proofreading)	$ 2,933
Illustration, 40 pages at $ 21,92 per illustration	877
Press, binding, and photography for 8,500 printed pages at $,022 per printed page	187
Total Estimated Documentation Costs	$ 3,997

SCHEDULE E

UTILITY COMPUTER SUPPORT SUBCONTRACT COSTS

As per SRI/Tymshare quotaton dated 11 December 1974.

Basic system:	$ 54,790 per month
RM-10B's x 2	$ 4,400 per month
Total	$ 59,190 per month

Proposal ISU 74-253 NLS Workshop Support for AFAA

22-DEC-74
SRI-ARC 24724

SRI Proposal No. ISU 74-253

NLS Workshop Support for AFAA

Part One---Technical Proposal

Prepared for:

USAF Audit Agency
AFAA/DOV
Norton Air Force Base
San Bernardino, California 92409

Attn: Mel J. Draper

Rome Air Development Center
Griffiss Air Force Base
Rome, New York

Attn: D. L. Stone

Prepared by:

James C. Norton, Assistant Director
Augmentation Research Center

Approved:

Douglas C. Engelbart, Director
Augmentation Research Center

Bonnar Cox, Executive Director
Information Science and Engineering Division
Stanford Research Institute

22-DEC-74
SRI-ARC 24724

SRI Proposal No. ISU 74-253

NLS Workshop Support for AFAA

Part Two---Contractual Provisions

Prepared for:

USAF Audit Agency
AFAA/DOV
Norton Air Force Base
San Bernardino, California 92409

Attn: Mel J. Draper

Rome Air Development Center
Griffiss Air Force Base
Rome, New York

Attn: D. L. Stone

Proposal ISU 74-253 NLS Workshop Support for AFAA

(J24724) 23-DEC-74 21:10;;;; Title: Author(s): James C. Norton/JCN; Distribution: /DLS([ACTION]) MJD([ACTION]) JHB([INFO-ONLY]) RLL([INFO-ONLY]) SRL([INFO-ONLY]) JMB([INFO-ONLY]) JDH([INFO-ONLY]) DCE([INFO-ONLY]) RWW([INFO-ONLY]) MEH([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JCN; Origin: < NORTON, AFAAPROP.NLS;1, >, 22-DEC-74 13:47 JCN ;;;;

####;