CHI 12-NOV-74 08:28 24479

npg update

RLB2 has been added to npg group 1

CHI 12-NOV-74 08:28 24479

npg update

(J24479) 12-NOV-74 08:28;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /NPG([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG;
Clerk: CHI;

CHI 12-NOV-74 08:32 24480

send-mail work file

Since the send-mail work file is initialized whenever a user goes into sendmail subsystem,, we might just as well make the file temporary or even delete it when user quits send-mail. What think you? has existence of this file caused users any problem? 1

CHI 12-NOV-74 08:32 24480

send-mail work file

(J24480) 12-NOV-74 08:32;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /EKM([ACTION]) HGL([INFO-ONLY]) DSM([INFO-ONLY]
) RLB2([INFO-ONLY]) KIRK([INFO-ONLY]) JDH([INFO-ONLY]) KJM(
[INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: CHI;

KIRK 12-NOV-74 13:20 24482

send-mail workfile

This is in answer to CHI's <24480,> I don't know who he sent it to (when I try to load it, I get the message 'File not on-line; If archived, use EXEC's INTERROGATE') so I'm sending my response to you guessing that you also got the question.

KIRK 12-NOV-74 13:20 24482

send-mail workfile

I don't think the send-mail work file should be initialized whenever a user goes into the sendmail subsystem. I would like to be able to continue a sendmail item after having to reset NLS. I would also like to be able to say "Execute Sendmail Interrogate" and not have all my work go away if I make one mistake and when I say "No" to "(Finished?)". Of course there needs to be some mechanism to let the user know he has an unfinished item. I suggest placing the user in the interrogate command automatically if he doesn't have unfinished mail and/or sending the noise words (and pseudo command) "(you have unfinished mail, initialize?)" Y/N: . 1

KIRK 12-NOV-74 13:20 24482

send-mail workfile

(J24482) 12-NOV-74 13:20;;;; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: KIRK;

KEV 12-NOV-74 17:04 24483

bcpl-deficits

this is the document i wrote back in august

bcpl-deficits

The followings deficits of BCPL are deficits by comparison with L10. The list was obtained through talks with Bruce Parsley, Bill Duvall, and Smokey Wallace at XEROX PARC, and by our reading of the BCPL manual. 1

It is not clear whether or not separately compiled programs in a language other than BCPL can be loaded together. 2

The SWITCHON statement in BCPL is no where near as powerful as the CASE statement in L10. 3

The BCPL FOR loop has the following problems: 4

The condition for exiting the loop is only computed once. Thus if the condition is an expression whose elements change during the execution of the loop, these changes may not be reflected when checking to see if the loop should be exited. (This may be an implementation problem of a specific BCPL.) 4a

The FOR loop index variable increment must be a compile time constant and can not be a runtime variable. 4b

The FOR loop index variable has a scope that is local only to the FOR loop, and thus when you have exited the FOR loop you can not find out how many times you went throught the FOR loop 4c

There is no easy way to do the L10 equivalent of REPEAT CASE. 5

There is NO way to do the L10 equivalent of REPEAT CASE n or REPEAT LOOP n 6

There is NO way to do the L10 equivalent of REPEAT CASE (exp) . 7

There is NO way to do the L10 equivalent of EXIT CASE n or EXIT LOOP n 8

You can't do an assignment as part of an argument in a procedure call, e.g. proc(i_x) 9

There is no equivalant of the L10 := type assignment 10

Procedures can return only one value 11

There is no SIGNALling ability in the language 12

There appears to be no arithmetic AND function, only a boolean AND 13

You can not declare static tables 14

You can not drop into assembly language (not necessarily a drawback) 15

There is no good string manipulation capabilites in the language 16

KEV 12-NOV-74 17:04 24483

bcpl-deficits

(J24483) 12-NOV-74 17:04;;;; Title: Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /CHI([INFO-ONLY]) DIA([INFO-ONLY]) RWW([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: KEV; Origin: < VICTOR, BCPL-DEFICITS.NLS;1, >, 7-AUG-74 17:02 KEV
;;;;####;

KIRK 13-NOV-74 03:26 24484

Help: CENSORED

Information about the availability of certain user-programs is being censored from what is potentially the primary online medium for finding out what is available to help you do what you want to do.

KIRK 13-NOV-74 03:26 24484

Help: CENSORED

Tuesday shortly after 4:00, there was a meeting to discuss user-programs. RWW, JCN, EKM, RLL, DVN, JHB, DSM, NDM, and KIRK attended. 1

There were two basic areas discussed. First, a need was identified for a procedure for reviewing and making-available changes in NLS including user-programs and their documentation. JHB, EKM, JDH, and KIRK are to get together and establish the details of this procedure. A meeting was set for this Friday at 3:00 to discuss the current status of user-programs. 2

Secondly, the question of what programs can be made available in directory <PROGRAMS> and referenced in the user-programs library was discussed. It was decided that only class-one programs "user-programs that are widely used" (as defined in -- 23999,) were to be referenced in the help database and made available in <PROGRAMS>. 3

Reasons for excluding other programs are: 4

Reason 1. We don't have the man-power to provide entries to any user-programs people might want to make available. 4a

Counter argument: I offer my services to do this right, free, on my own time, unpaid by applications or development. 4a1

Reason 2. If someone has trouble with a program clearly marked "Not supported by Knowledge Workshop personnel see: Joe Blow" User Development will have to spend time fielding their questions. 4b

Counter argument: In fact, this is more likely to occur if the programs are not documented in the user-programs library. Not being able to find out information about a user-program via help when they run into problems, the user will be forced to bother Applications personnel. 4b1

If this occurs when the program is clearly marked "not supported" in the user-programs library, it seems a previously journalized message would be appropriate to "Forward" to a user who wants support for something that is clearly marked "not supported". The message would say, -- "We do not support that program, talk to the person listed for it in Help," -- a small, optional and infrequent price to pay for increased capabilities and an open NLS market place. 4b2

Reason 3. Utility customers are currently analogous to 2-year-olds and must be protected from programs we think are unacceptable. 4c

Counter argument: This kind of reasoning seems to me analogous

to "de-augmentation" as described by Doug in the '62 paper. Tie a brick to your pencil. When a user-program allows you to do something with one button push that otherwise takes 15 button pushes; and information about that program is censored from your primary public information medium, you have been, in a very real sense, "de-augmented". 4c1

There are lots of other analogies for this kind of reasoning, but I don't want to get into political science. You can think of your own. 4c2

People access the Help data-base to find out what is available to help them do what they want to do. The absence of user-programs written to help do a job in NLS is at best misleading, and certainly does not look to me like a healthy environment for the evolution and growth of the NLS knowledge workshop. 5

KIRK 13-NOV-74 03:26 24484

Help: CENSORED

(J24484) 13-NOV-74 03:26;;;; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: KIRK;

CHI 13-NOV-74 08:45 24485

Some NSW Frontend Issues Raised at NOV 6-7 NSW Review Meeting

sent via SNDMSG to
carlson,crocker,carlstrom,balzer,warshall,watson,crain,baggiano,lloyd
,wingfield,postel,millstein,stone,white,mayhan,vezza,stubbs,retz,nato
li,triolo,wall,michael,bthomas,walker,lehtman,

CHI 13-NOV-74 08:45 24485

Some NSW Frontend Issues Raised at NOV 6-7 NSW Review Meeting

INTRODUCTION 1

The concept of the NSW Frontend is a very important yet, we think, not fully understood part of the NSW as a whole and is, we believe, very vital to the overall success of the NSW program. This paper attempts to clarify the Frontend concept somewhat, to raise some issues that have come up so far, and to present our current thoughts on those issues. This is a draft document and has been updated to reflect the outcome of the NSW Project Review meeting held Nov 6,7, and 8 at SRI (attendees: Millstein, Balzer, Crocker, Watson, Irby, White, Postel, Waal, Triolo, Michael, Lehtman, Victor, and other SRI staff). The decisions reached at this meeting are denoted by []'s. 1a

THE PURPOSE OF THE FRONTEND FOR THE NSW 2

The Frontend is a buffer between the user and the Works Manager and tools of the NSW. The Frontend provides a logical function (interfacing the user to the NSW) and will initially be implemented on a PDP-11 satellite computer. In the future, the Frontend may consist of a program running on a satellite cooperating with a program running on some larger computer. The primary reasons for having a Frontend, as we perceive them, are as follows: 2a

1) To provide the user with a coherent and consistent command language discipline throughout the NSW. 2a1

No matter whether the user is giving commands to a tool or to the Works Manager or the Frontend itself, he does so using the same methods for specifying which commands he wishes executed, the same methods for specifying arguments or parameters to commands, gets the same type of prompting and requests help in the same way, always. In addition, the general syntactic form(s) should be the same from tool to tool unless there is good reason for the tool to deviate from the standard. Of course the particular commands and vocabularies will vary with the tool and in fact the same verbs may be used with quite different semantics in different tools, but at least most other facits of the command language (including asking for help and being prompted for the proper type of input) should stay the same across tool boundaries. 2a1a

It is expected that initial users of the NSW will have to access some tools in a "transparent" mode [see discussion below in Issue 1], where the Frontend cannot provide the user with the facilities just described. In this instance, the Frontend could take on the nature of a TIP or ELF

terminal concentrator. However, we expect that as time goes on and the NSW grows, the user will be able to use most tools through the unified user interface provided by the Frontend, and it is toward this end that we should build. 2a1b

2) To provide tools with well-formed commands. 2a2

It is proposed that this be done by issuing remote procedure calls to "external" procedures in the tools to actually execute commands. This will be accomplished through the Procedure Call Protocal (see Jim Whites papers on the PCP). 2a2a

Many operating systems and application programs have elected to use half duplex, line-at-a-time terminals because of the increased computer efficiency provided by this approach. Other operating systems and application programs have chosen, instead, to utilize character-at-a-time full duplex terminal disciplines because of the opportunity this provides for utilizing a more human-engineered command language. 2a2b

The NSW Frontend is an attempt to combine these two approaches into a COMMAND-AT-A-TIME system, where the application programs do not directly interact with the terminal, but rather receive fully specified commands from the Frontend. At the same time, the Frontend will attempt to provide the user with the best possible human-engineered command language discipline. This means that interfacing to application programs developed for line-at-a-time terminals should be quite straight forward, even though the user operates from a character-at-a-time full duplex terminal, since such programs tend not to interact with the user extensively [See discussion under Issue 2 below]. 2a2c

3) To provide a terminal-independent interface to the tools. 2a3

Because the Frontend handles all terminal interaction (except in transparent mode), it will present to the tool a virtual terminal. Thus, once a tool is developed, little attention need be given to the type or particular characteristics of the terminal the end user may choose to employ while using the tool. In fact, the cost of creating new tools should be considerably reduced because of the facilities made available by the Frontend. 2a3a

This means that even though the creators of a tool envisioned the user sitting at a typewriter terminal, the NSW user who happens to be using a display terminal with a pointing device may be able to interact with the tool

in a two dimensional sense, pointing to arguments on his screen instead of typing them, etc. [See discussion under Issue 2 below] 2a3a1

For tools which wish to make more extensive use of a display terminal if the user has one, the Frontend presents primitives for allocating windows on the display and allows the tool to write/delete/move/make invisible items displayed within the windows. 2a3a2

4) Possible asynchronous operation. 2a4

In some instances, it may be possible for the execution of the user's commands to be accomplished in parallel with subsequent command specification and execution. This frees the user to do other things while a lengthy command is being executed by a tool. 2a4a

5) NSW-wide macro facilities 2a5

The user should be able to define (text substitution) macros which he can then use with any tool, since the macros will be expanded by the Frontend. 2a5a

[NOV 6-7 DECISION: 2a5b

A macro facility will probably not be made available for first year NSW.] 2a5b1

6) To provide standard mechanisms for presenting status or error conditions to the user. 2a6

an error should consist of the following: 2a6a

a human readable error message 2a6a1

a code indicating whether this error caused the command to be aborted, completed or undefined and whether the tool is now in a state to receive more commands or should be restarted. 2a6a2

certain types of errors will have to be reported to the Works Manager so that it can take action (e.g. file system errors, disk errors, etc). Does the Frontend do this or does the tool? [Could Millstein or Warshall address this?] 2a6b

In particular, from the user's standpoint, the first and last of these justifications are very important. With very few exceptions, most user's would become very frustrated in an

environment where every new tool he chose to use required that he learn a new interaction discipline. (Imagine having to speek Greek to the gas station attendant, Spanish to the grocer, French to the waiter, and so forth.) In fact, it is our belief that this would spell certain doom for the NSW. 2b

The Frontend functions could be much more extensive and could be distributed between the satellite PDP-11 and a larger host. This notion will have to evolve as we gain experience with the NSW as a whole. 2c

ISSUES THAT HAVE COME UP SO FAR 3

Some of the issues that should be raised now are the following: 3a

1) What is to be done for tools that already exist as monolithic packages where the user interaction cannot readily be separated out into the NSW Frontend? 3a1

It is our belief that the more tools which are fully integrated into the NSW framework the better the NSW will seem as a total system. However, we must also allow users to run tools which were not built or modified to run within the NSW. Aside from the impact this has on the Works Manager, the Frontend cannot be expected to help the user when he is using this type of tool. It is envisioned that a transparent mode (with a user-settable escape character) will be provided for such tools and that tool output to the user will appear as though the user had a TTY, even though he may be using a display. 3a1a

[NOV 6-7 DECISION: 3a1b

From the Frontend's standpoint, all tools have a grammar that drives the Frontend's interaction with the user. For some tools, however, this grammar may be very trivial and may cause a trivial backend to be invoked (through PCP) that simply passes characters to the actual tool. This means that the tool itself need not be modified, but that the user interacts with the tool just as he interacts with other tools and that he still has easy access to the Works Manager and to the Frontend itself. 3a1b1

This means that to install a tool into the NSW, someone must write the CML grammar for it (this may be very trivial, such as just collecting a string of characters from the user and passing them to the backend) and construct a simple backend which will interact with the frontend through PCP procedure calls

and ship characters to/from the real tool and perhaps handle some error conditions. It may be possible to use the same grammar and simple backend for several such tools if the simple backend can treat the real tools the same all the time. 3a1b1a

If it turns out to be trivial to also supply a transparent mode that allows for trial, evaluative use of a tool without any investment in a grammar, etc, then this will also be provided.] 3a1b2

2) What classes of terminals should be supported in the first year? 3a2

A Network Virtual Terminal (NVT) was defined for TELNET. We would propose that in addition to this definition of a virtual typewriter terminal, the NSW needs a similar definition of a virtual alpha-numeric display and a virtual display with a pointing device. 3a2a

It is our assumption that it is ADR's responsibility to write drivers for any terminals the NSW management decides the NSW Frontend will support to map them into the appropriate virtual terminal. 3a2b

The Frontend will provide external procedures to present error and status messages to the user and, if he is using a display terminal, to allocate and manipulate text and graphics within windows on the display. A tool will not, however, be able to effect certain areas of the screen which are used by the Frontend for command feedback. 3a2c

While many application programs in conventional operating systems now make use of line-at-a-time terminals, as discussed above, these terminals do not allow the Frontend to actively interact with the user to answer help requests, to provide keyword recognition, noise words, or to prompt him for various types of input. For these reasons, the NSW Frontend should perhaps support half duplex terminals for printing devices only (since the best hardcopy seems to be produced on some of these devices). 3a2d

Proper support of half duplex terminals as interation devices may call for quite a different interaction strategy on the part of the Frontend. Should we be planning on this or just support them awkwardly, as is now done throughout much of the ARPANET? 3a2d1

[NOV 6-7 DECISION: 3a2e

SRI will support half duplex, line-at-a-time terminals as interation devices. ARC currently believes that the specification of the command language for a tool should be, for the most part, independent of the terminal class being used with the tool. There is now a facility in the CML for the command language designer to specify that certain commands (or parts thereof) should not be available for certain terminal classes. This facility will probably have to be expanded to account for half duplex terminals. CHI will think about interaction form for a half duplex terminal -- the Frontend may be able to do special, nicer things for users at such terminals. 3a2e1

SRI (Victor) will define the following virtual terminal classes for NSW: 3a2e2

1) Half duplex, line-at-a-time terminal, 3a2e2a

2) NVT (already defined), 3a2e2b

3) Alpha-numeric display without pointing device and with and without editing functions, and 3a2e2c

4) Line Processor-enhanced alpha-numeric display (already defined). 3a2e2d

The definition of a virtual graphics terminal may be deferred until next year (May be able to use Network Graphics Protocol here.). 3a2e3

A facility should be made available for alpha-numeric display terminals to automatically page output (halt output until the user types a character to continue outputting). In addition, it would be desireable from such a terminal to be able to scroll back though recent tty output that is no longer on the screen. The frontend will accomodate this within memory size limits (may have to dump it on a file somewhere).] 3a2e4

3) Is the Command Meta Language flexible enough for first-year NSW (documentation of currently planned language will be available shortly)? 3a3

As currently planned, the NSW CML will be derived from the current CML, used in the NLS system. The re-implementation and generalization of this CML will remove problems that we have found during its use for the past year within NLS but are there other steps we should take to generalize it even further? 3a3a

We are now in the process of writing trial grammars for TECO, WYLBUR, CANDI, NETED, SOS, and two forms of DDT. A discussion of these will be distributed shortly. Are there other interactive systems that people know of that we should try to express in CML at this time? 3a3b

[NOV 6-7 DECISION: 3a3c

It was felt that if the CML could actually be used to describe the above set of user languages that it would be sufficient for first year NSW use. We were able to specify one of the most complicated commands in the Works Manager with no difficulty. We will bring up any problems that arise. 3a3c1

Bob Balzer felt strongly that the user should be able to execute "universal" (always available) commands to the Frontend or Works Manager even if these conflict with tool-specific commands. Please note that this imposes a limitation on tool command languages since they cannot use commands that begin with the same command words as the Frontend or Works Manager commands (Warshall was opposed to this in a meeting in Atlanta, feeling that it would be better to have an escape character which the user typed in order to talk to the Frontend or Works Manager). We have used the "universal" command strategy within NLS with little problem of conflicts. This approach seems (to us) to present a simpler user interface to the overall system but either approach is acceptable to us. It seems more discussion is needed here.] 3a3c2

4) What language form should be the NSW standard? 3a4

We currently envision NSW commands that will consist of command words (which might serve as verbs for commands or as modifiers for partialy specified commands), parameter specifications (pointing operations for users at displays and typing parameters), standard command confirmation (final ok to proceed), standard backup to a previous state in the specification of the current command, standard ways of determining the currently available alternatives or the general syntactic form of commands, and general semantic (data base driven) help for particular tools and commands within those tools. 3a4a

Should the NSW CML attempt to enforce some conventions about the form of commands, such as specifying command words at the begining of commands followed by parameters or vice

versa? Should the recognizers for some low-level parameter types, such as word, text-string, and file name, be built into the CML so that they will be the same for all tools? 3a4b

Should post fix be allowed? Should functional notation be allowed? 3a4c

[NOV 6-7 DECISION: 3a4d

It was felt that the CML should not be changed to be more restrictive in an attempt to enforce a standard NSW command language convention. This can be done later when the issues are more clearly understood. Although this issue was not discussed at length, we feel that certain commonly used recognizers should be built into the frontend (to facilitate writing simple command languages and to present a more uniform "system" to the user) in such a way that a command language designer can override them if necessary. Examples of such builtin recognizers are file names, free text, words, and visibles (words with ajoining punctuation).] 3a4d1

5) What interaction and presentation forms should be provided? 3a5

Should function keyboard simulation or menu selection be provided for tools? 3a5a

[NOV 6-7 DECISION: 3a5b

Function keyboards and menu selection will not be supported in the first year system. Interaction forms will be similar to those now supported by NLS.] 3a5b1

6) What sorts of things should the user be allowed to tailor in the Frontend to his own personal preferences? 3a6

Should he be able to specify the level of verbosity, the amount of prompting, the succinctness of error messages, the recognition algorithm for command word recognition, tools available to him, etc. What should a site or group manager be able to specify on behalf of his people? 3a6a

[NOV 6-7 DECISION: 3a6b

The facilities now supported by NLS for this purpose will be provided. In addition, the list of available tools will be settable by a project leader for his project personnel. A User Profile tool will be provided as a subset of the NLS backend.] 3a6b1

7) How much control should a tool be allowed to have over the user's terminal? 3a7

Our current plan is to provide a standard mechanism for presenting status and error messages and the ability to write text (and perhaps graphics) in or to subdivide windows on a portion of the screen. Is anything else needed or does anything beyond this constitute a violation of the primary reason for the frontends existence -- i.e. to present a standard interface to all NSW tools? 3a7a

[NOV 6-7 DECISION: 3a7b

It was not felt that additional facilities would be needed for the first year system.] 3a7b1

8) Semantic help 3a8

It is envisioned that the user will be able to obtain English help with tools in the NSW. This will be accomplished by providing a separate tool, capable of interacting with the user (via a grammar in the Frontend) and using a structured data base provided along with the tool grammar. This help tool will not run in the satellite machine but will be invoked by the satellite whenever the user asks for semantic help with a tool. The help tool will be provided with the name of the help data base for the tool the user was using and a representation of the user's command state at the time he requested help. (Once a connection has been established to the help tool for a user, the connection will probably be maintained until the user terminates the session.) It is expected that the command language designers will provide the data bases. It is expected that there will be one data base for the NSW as a whole, describing global concepts, organization, and purpose of the NSW. This data base will be available at all times to the user. In addition, we may wish to produce a data base that is a high-level guide to all the tools accessable through NSW. 3a8a

We would propose that for first-year NSW, this help tool is simply a set of calls on the NLS backend, with the data bases being NLS structured files (this approach is now being used within NLS). 3a8a1

In the future it is expected that the help tool will be able to take on the character of a tutor and show the user how to execute commands and what the effects of doing so are as well as providing the user an environment in which he is

free to try things without running the risk of destroying anything or leaving unwanted trash around. (We refer you to the NLS-SCHOLAR project at BBN and to the COTCO system being designed at ISI as examples of more active help facilities.) Initially and for some time, however, we expect the help tool to be more of a browsing aid, using a structured data base to allow the user to more quickly find the information he seeks. 3a8b

[NOV 6-7 DECISION: 3a8c

This was felt to be an acceptable plan for first year NSW.] 3a8c1

9) UNDO and REDO 3a9

What mechanisms must be in the Frontend to facilitate UNDO and REDO capabilities on an NSW-wide basis and what must be done for tools which can undo the effects of previous function executions. 3a9a

[NOV 6-7 DECISION: 3a9b

UNDO was not discussed and will not be available in first year NSW. REDO was discussed briefly and was felt to be valuable. If resources permit, the frontend will provide this for the most recently specified commands for first year NSW.] 3a9b1

10) Checkpointing 3a10

Must the Frontend do anything with respect to NSW checkpointing, backing up to a checkpoint, etc? What does the Frontend do when the Works Manager crashes and restarts? What must the Frontend do to allow the user to pickup where he left off at the end of his last session? 3a10a

[NOV 6-7 DECISION: 3a10b

This was not discussed (we ran out of time). Perhaps Warshall or Millstein could write a short blurb on this? Is it to be available in first year NSW? Suspending and resuming a session will not be provided by the Works Manager for the first year system.] 3a10b1

11) Terminal Linking 3a11

The ability for two or more users to connect their terminals (especially displays) together in order to work together or

simply to type to each other has been found to be a very valuable facility. Should this be offered in the NSW. 3a11a

[NOV 6-7 DECISION 3a11b

Terminal linking, although a very valuable facility, will not be supported in first year NSW.] 3a11b1

THE CML AND ITS IMPLICATIONS FOR NSW TOOLS, TOOL BUILDER, TOOL INSTALLERS, AND COMMAND LANGUAGE DESIGNERS 4

The Frontend system that is being planned for the NSW consists of the following: 4a

1) A formal language (CML) for specifying NSW user interfaces 4a1

2) A compiler for that formal language that runs under TENEX as a subsystem or from NLS 4a2

3) Tool grammars, products of the CML compiler or any other such program 4a3

4) A CML interpreter that processes a CML grammar in order to work with the user in specifying syntactically correct commands to the NSW. 4a4

5) A user profile data base that is used by the CML interpreter while interacting with the user. This data base allows the Frontend to be tailored to the individual preferences of the users. 4a5

6) A user statistics data base, where, if desired, statistics can be accumulated on commands used by a user, error rates, etc. This will be accumulated on a file or perhaps reported to the Works Manager. 4a6

7) Access to a semantic help tool which is employed by the Frontend when the user requests semantic level help with a tool or a command. It is presumed that each tool, in addition to supplying the Frontend with a grammar will also supply it with the name of a help data base file whose structure and content, as with the grammar, are the sole responsibility of the tool builder/supplier. 4a7

This help tool could also be kept informed of the user's dialog with the Frontend, can have access to the tool grammar, the current parse state of the user, and the user's profile. 4a7a

Detailed discussions of the CML and the CML interpreter are being prepared and will be forwarded to you as soon as they are completed. 4b

DOCUMENTATION THAT SEEMS NECESSARY FOR THE NSW TO FUNCTION [Someone (Carlson, Balzer, or Crocker) should specify such a list of necessary documentation and affix responsibility for providing it.]: 5

A system guide to installing and running an NSW Frontend 5a

System specs for tool bearing host execs 5b

A system manual on the flow of control in the NSW as a whole 5c

A system guide for Command language designers 5d

A system guide for tool installers 5e

System documentation on the Procedure Call Protocol 5f

System documentation on debugging NSW tools 5g

System documentation on (PCP) external procedures for each tool 5h

System documentation on CML 5i

System documentation on functions the Frontend provides to a tool 5j

System documentation on functions the Works Manager provides 5k

CHI 13-NOV-74 08:45 24485

Some NSW Frontend Issues Raised at NOV 6-7 NSW Review Meeting

(J24485) 13-NOV-74 08:45;;;; Title: Author(s): Charles H. Irby/CHI; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: CHI; Origin: < NSW-SOURCES, FE-ISSUES,NLS;4, >, 13-NOV-74 07:59 CHI ;;;;####;

RWW 13-NOV-74 08:55 24486

RJE Responsibilities

There was considerable discussion of the RJE process and problems, and which contractor was responsible for what at the recent NSW meeting at SRI. This note is to indicate what I think was agreed on in this area. 1

Protocols in the NSW PCP environment take on somewhat of a different character than they did in the past and become packages of callable procedures that represent standard functions or services available to tools and other NSW processes. SRI as part of its protocol responsibilities will define a cut at a standard RJE model and set of functions. 2

A tool in the NSW environment is defined by: 3

1) a grammar to be used by the Frontend interpreter defining its interactive user interface, if any. 3a

2) its "backend" or set of actual information processing procedures and functions operating in a PCP and Tool Bearing Host environment. 3b

3) use mediation by the Works Manager. 3c

It is envisioned, subject to further thought by the protocol guys, that a user would perform an RJE task by sitting down at a terminal and interacting with the RJE tool. The tool would collect parameters such as source file name, destination file for results, JCL file etc and then would initiate and complete the task while maintaining appropriate progress status information for interrogation by the user. The user interface would probably consist of the RJE protocol functions and possibly some additional features that are a superset unique to a particular system (although one hopes that these can be minimized). 4

It is clear that the B 4700/PDP 11 is a tool and that responsibility for bringing it into the NSW environment rests with ADR, DSDC and the B 4700 software contractor. Therefore, they are responsible for providing the RJE "backend" in the PDP 11 and B 4700. Technically they are probably responsible with SRIs help to provide the grammar also, although SRI as part of its NLS task to get COBOL programs written, compiled, and executed would be willing in this case to write the grammar for the PDP 11/B 4700 RJE tool. SRI is providing the PCP environment for the PDP 11, and ADR will provide an interface to PCP from ELF. 5

RWW 13-NOV-74 08:55 24486

RJE Responsibilities

(J24486) 13-NOV-74 08:55;;;; Title: Author(s): Richard W. Watson/RWW; Distribution: /NPG([INFO-ONLY]) JBP([INFO-ONLY]) DCE([INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: RWW;

JEW 13-NOV-74 10:20 24487

User Program Thoughts Prompted by Kirk's (24484,)

The following comments are largely in support of Kirk's arguments in (24484,). 1

If only "widely used" user programs are made available to the user community, it's not clear how user programs can ever BECOME widely used. One can't try a program and find it useful without first knowing about it. 2

Also, by discouraging the sharing of user-written user programs, valuable feedback about the user community's needs is lost. 2a

Although it seems to me that any user program should be offerable by a user to the entire user community, the HELP system seems the wrong place for documentation of programs unsupported, by the installation since the support of system personnel is then required, since a system file is involved. 3

Kirk's offer to assume personal responsibility for maintenance of the data base seems like the wrong solution. Any user service that depends solely upon someone's unpaid work will lose eventually (when the volunteer leaves), since the installation by definition has no committment to continue support of that service. 3a

A more reasonable use of Kirk's time, it seems to me, would be for him to create an NLS subsystem via which users can publicize and learn of each other's user programs. This subsystem would manage a data base separate from help's, and have simple commands for entering information about a program into the data base (and the program itself into a system directory), updating it, classifying the program, commenting upon it, and so forth (there are prototypes of such subsystems around, I believe). 3b

Although the maintenance of documentation about "unofficial" user programs is understandably a task an installation might not be able to support, the maintenance of a subsystem which does the work seems quite reasonable to include as part of the system. 3c

Of course, it's important that the class of each user program, as outined in (23999,), be carefully identified to the user and that rules about non-support of certain classes be faithfully adhered to. When the author of a private program leaves, the program is either abandoned as soon as it needs modification, or responsibility for it is assumed by another user. And, as it finds resources to do so, the installation should reclassify widely-used programs and assume their support. 4

JEW 13-NOV-74 10:20 24487

User Program Thoughts Prompted by Kirk's (24484,)

(J24487) 13-NOV-74 10:20;;;; Title: Author(s): James E. (Jim) White/JEW; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JEW; Origin: < WHITE, ARCMSG.NLS;2, >, 13-NOV-74 10:18 JEW ;;;;####;

NDM 13-NOV-74 14:00 24489

Proposal for Bibliographiy Subsystem

Request for comments and suggestions on a first step bibliographic citation program.

Proposal for Bibliographiy Subsystem

A number of applications groups are interested in bibliographies. I would like to build a subsyste_m that can take catalog citations and produce number, titleword, and author listings. This would provide a user interface to the current catalog production reformatters (not the catalog production processor, but the techniques we have developed for formatting catalog entries and extracting citations from them) that might be thought of as the first step in providing biliographic tools. 1

I think such a subsystem should take: 2

-a statement, group, or plex of catalog entries from a user's file 2a

-a statement, group, or plex of tjcat entries (or a journal origin statement) from a user's file 2b

and produce: 3

-author citations 3a

-titleword citations 3b

-number citations 3c

-any other format for a citation we care to set up. 3d

It ought to allow for expansion: 4

-user settable formats (fields put in various places) 4a

-take a journal number from the user 4b

-catalog entry building tools 4c

I propose the following initial syntax: 5

```
SUBSYSTEM biblio KEYWORD "BIBLIOGRAPHY"                          5a
   COMMAND                                                       5a1
      fbiblio =                                                  5a1a
         "INSERT"!L1!                                            5a1a1
         ctype _ (                                               5a1a2
            "AUTHOR"!L1! /                                       5a1a2a
            "NUMBER"!L1! /                                       5a1a2b
```

```
        "TITLE"!L1! <"word">                                    5a1a2c
        )                                                       5a1a2d
    <"citation(s) at"> dest _ DSEL(#"STATEMENT")                5a1a3
    level _ LEVADJ                                              5a1a4
    <"using"> stype _ (                                         5a1a5
        "CATALOG"!L1! /                                         5a1a5a
        "JOURNAL"!L1! /                                         5a1a5b
        )                                                       5a1a5c
    <"entries in"> ent _ structure                              5a1a6
    <"at"> source _ DSEL(ent)                                   5a1a7
    Xbiblio (ctype, dest, level, source, stype) ;               5a1a8
END.                                                            5a2
```

Proposal for Bibliographiy Subsystem

(J24489) 13-NOV-74 14:00;;;; Title: Author(s): N. Dean Meyer/NDM;
Distribution: /RWW([ACTION]) JCN([ACTION]) KIRK([ACTION])
JHB([ACTION]) EKM([ACTION]) DSM([ACTION]) JAKE([ACTION])
RLL([ACTION]) SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: NDM; Origin: < MEYER, BIBPROP,NLS;1, >, 13-NOV-74 13:16
NDM ;;;;####;

JMB 13-NOV-74 14:24 24490

RE: Maximizing Alternatives

Kirk, I keep running across people and writers who are saying things similar to your theories.

JMB 13-NOV-74 14:24 24490

RE: Maximizing Alternatives

Quote from: "Geosocial Revolution" in
Fuller, Buckminster, COMPREHENSIVE THINKING; Phase I, Document 3 of WDSD, 1965 1

"In dynamical balance with the inside-outing, expanding universe, man witnesses Earth as a collecting or outside-inning, contracting phase, of universe. In addition to its daily sun radiation income Earth receives a continually increasing inventory of radiation in its lethal, energy concentrates sifting, sorting, and accumulating Van Allen Belts. The succession of concentric terrestrial spheres, e.g., ionosphere, troposphere et al., constitute an extraordinary series of random-to-orderly sorting, shunting, partially accumulating and partially forwarding--inwardly and finally to benign state in the biosphere--of Earth's continual, universal, energy income receipts. 2

Earth also receives an additional one hundred thousand tons of stardust daily. This randomly deposited dust apparently consists of all the 92 regeneratively patterning, chemical elements in approximately the same order of relative abundance as the relative abundance of those elements in the thus far inventoried reaches of universe. 3

The biological life on Earth is inherently anti-entropic for it negotiates the chemical sorting out of the Earth crust's chemical element inventory and rearranges the atoms in elegantly ordered molecular compound patterning. Of all the biological anti-entropics, i.e., random-to-orderly arrangers, man's intellect is by far the most active, exquisite, and effective agent thus far in evidence in universe. Through intellect, man constantly succeeds in inventing technological means of doing evermore orderly--i.e., more efficient, local universe, energy tasks with ever less units of investments of the (what may be only apparently) "randomly" occurring resources of energy as atomic matter, or energy as channeled electro-magnetics. 4

JMB 13-NOV-74 14:24 24490

RE: Maximizing Alternatives

(J24490) 13-NOV-74 14:24;;;; Title: Author(s): Jeanne M. Beck/JMB;
Distribution: /KIRK([ACTION]) ; Sub-Collections: SRI-ARC; Clerk:
JMB;

RWW 13-NOV-74 14:29 24491

Reply to Kirks Note on User Programs

Kirk's note did not fairly describe the thrust of the meeting in a number of respects and utilized misleading emotional terms not part of constructive dialog,. It was generally agreed that user programs should have a mechanism to get advertised and made available, The issue was strictly time frame and mechanism, 1

There are a number of interesting and thorny problems in deciding how to deal with user programs, It is felt by JCN and myself that the Help database is the wrong place at this time for any programs other than those that are felt to be "part of the system" and supported officially by ARC, In fact I still think that when something gets that much use it should stop being a user program and get incorporated into the system, 2

Applications has got its hands full right now with the conversion to NLS 8 marketing and other issues and it will be a year or two before the KWAC matures to the point where they can set up and maintain procedures as a user group like SHARE or others for quality control etc on a user program library, We will also need to think through the process carefully before charging off, Right now we have our hands full with NSW and other things, 3

In the meantime informal mechanisms will have to evolve, The process bby which user progrms get into the system needs carefull thought as it is more than function but coding style and quality etc that has to be quality checked, Already some code has creptinto the system from user programs that hs caused problems unnecessarily, 4

RWW 13-NOV-74 14:29 24491

Reply to Kirks Note on User Programs

(J24491) 13-NOV-74 14:29;;;; Title: Author(s): Richard W. Watson/RWW; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RWW;

RWW 13-NOV-74 14:39 24492

Conversation with Craig Fields on Nov 13

Craig and I discussed two topics the Intelligent Terminal Program and plans for upgrading of the network. The Bob Anderson report is about to be released. The plan to be followed or recommended calls for four areas of work, psychological studies in the user interface area, technology transfer, tools in the terminal and a AI framework to couple user to tools in the terminal and elsewhere. Plans call for a competitive building of item 4 and low cost adaptation of 3 from exiting stuff. They are presently trying to decide which machine to use. They think that the PDP 11 has too small an address space andd are thinking about the Interdata 732, bu it has no good system building software. Craig see us as one of the AI framework competitors and expects feedback on the RAND report.
There are no plans to upgrade he net with higher speed lines high speed modular imps etc. If you want more buffer space on a TIP or TIPS get the money and order it from BBN and don't bother me (the procedure is obviously unclear). Memory for TIPS comes in 4K hunks. 1

RWW 13-NOV-74 14:39 24492

Conversation with Craig Fields on Nov 13

(J24492) 13-NOV-74 14:39;;;; Title: Author(s): Richard W. Watson/RWW; Distribution: /DCE([INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RWW;

CHI 13-NOV-74 15:30 24493

user program maintenance and publicity

I think the ideas and approach suggest by JEW in (mjournal, 24487,1)
is very sound and should be carefully considered by applications. 1

CHI 13-NOV-74 15:30 24493

user program maintenance and publicity

(J24493) 13-NOV-74 15:30;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC;
Clerk: CHI;

JMB 13-NOV-74 15:48 24494

Sendmail workfile & Initializing: Ref: <24482,> <24480,>

In support of Kirk's <24482,>: Sendmail workfile: 1

I have experienced that it greatly troubles new learners of NLS (as well as me sometimes) to lose the sendmail stuff they've already put in, by the stuff disappearing when they accidently or otherwise Quit Sendmail before giving Send command. 2

> If you have to abort the Interrogate command, or use Help in the middle of it (which in effect aborts the command--which in a way is self-defeating in using Help to learn), what you've already input does indeed remain there, as you see if you Show Status. You do have to start the command over, however, and know that you should just skip the fields you've already done, but Help doesn't tell you that. 2a

> Wait, I have just discovered I can't skip the "type of source" field in Interrogate by either <CTRL-d> or <CTRL-n> on my TI. So if Interrogate is all I've learned yet, and I already put everything in (having typed N to "Send the mail now?"), how do I get back through the Interrogate process to send the mail? Why does Interrogate have these quirks? Or, maybe they'll make sense when someone explains it to me so I'll have something to explain to a frustrated new learner. 2b

Now all this wasn't what CHI wanted to open up when he sent <24480,> but to me it contradicts his viewpoint: 3

> "We might just as well delete the work-file since it goes away when you Quit Sendmail anyway" 3a

--i.e, we should, I believe, go to more effort to preserve all we can for the user. I support Kirk's statement in <24482,> that 4

> "there needs to be some mechanism to let the user know he has an unfinished item,"
> [as well as letting him go back to it after Quit]
> "I suggest placing the user in Interrogate automatically if he [has] unfinished mail and/or sending the noise words (and pseudo command): '(you have unfinished mail, initialize?) Y/N:'" 4a

But don't let's put user in Interrogate if he can't skip the previously specified source field. If we're going to have the Initialize command we should make it a true choice. 5

JMB 13-NOV-74 15:48 24494

Sendmail workfile & Initializing: Ref: <24482,> <24480,>

(J24494) 13-NOV-74 15:48;;;; Title: Author(s): Jeanne M. Beck/JMB;
Distribution: /FDBK([ACTION]) CHI([ACTION]) KIRK([ACTION])
SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JMB;

CHI 13-NOV-74 15:49 24495

sendmail bug -- null author

It appears that an illegal author field in a sendmail request results in a null author instead of an error message to the sending user. This should get fixed in (nls,csendmail,jsubmit). -- Charles. 1

CHI 13-NOV-74 15:49 24495

sendmail bug -- null author

(J24495) 13-NOV-74 15:49;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /FDBK([ACTION]) EKM([ACTION]) ; Sub-Collections:
SRI-ARC; Clerk: CHI;

JMB 13-NOV-74 16:54 24496

Can you fix something in <USERGUIDES, Locator,6>?

I assume you're working on Locator because you have it locked.

JMB 13-NOV-74 16:54 24496

Can you fix something in <USERGUIDES, Locator,6>?

In <userguides, locator,> there is a link to the NLS-8 command summary in Branch 6. First, could you change it so it goes to--userguides,commands,definitions>, which will take the person to the Introductory section called DEFINITIONS AND CONVENTIONS which is the last branch in the file, oddly enough, but they should read it first. (definitions is the statement name for the source statement--that's how the link will work). Secondly, do you know why that link in branch 6 has the viewspec x in it; I don't exactly understand the conventions and techniques of locator, but you cant read the text if x is on. Would you find out about that please? 1

JMB 13-NOV-74 16:54 24496

Can you fix something in <USERGUIDES, Locator,6>?

(J24496) 13-NOV-74 16:54;;;; Title: Author(s): Jeanne M. Beck/JMB;
Distribution: /POOH([ACTION]) ; Sub-Collections: SRI-ARC; Clerk:
JMB;

JMB 13-NOV-74 20:00 24497

The Syntax Generator Lies! PLUS Trouble using the Syntax command

When I give the command Show Subsystem Supervisor when in Syntax generatoar subsystem, among the things it prints are:
!TNLS!Jump (to) Name BUG VIEWSPECS OK
!TNLS!Jump (to) File BUG VIEWSPECS OK
Now, even the Questionmark facility, as well as the bell on my TI, knows those aren't possible commands; they shouldn't be listed,
P.S. I couldn't test to see if they come out in the syntax command when in TNLS, because I can't seem to figure out in which subsystem you have to use the Syntax command to be able to ask it for supervisor commands: if you're not attached to Syntax gnerator (as most users aren't) you can't Goto Supervisor to get the Jump command listed in Syntax command, and if you do Syntax in any other system, none of the supervisor commands are alternatives after giving "Syntax". Would you please explain to me how to do this? 1

JMB 13-NOV-74 20:00 24497

The Syntax Generator Lies! PLUS Trouble using the Syntax command

(J24497) 13-NOV-74 20:00;;;; Title: Author(s): Jeanne M. Beck/JMB;
Distribution: /FDBK([ACTION]) BUGS([ACTION]) ; Sub-Collections:
SRI-ARC BUGS; Clerk: JMB;

CHI 14-NOV-74 07:44 24498

take care of 24497

Ken, I assume you will take care of JMB's (24497,). 1

CHI 14-NOV-74 07:44 24498

take care of 24497

(J24498) 14-NOV-74 07:44;;;; Title: Author(s): Charles H. Irby/CHI; Distribution: /KEV([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: CHI;

new pupils on the nls

how to know where the control marker (cm the pointer in a file) is. 1

SRL 14-NOV-74 08:04 24499

new pupils on the nls

(J24499) 14-NOV-74 08:04;;;; Title: Author(s): Susan R. Lee/SRL;
Distribution: /DRL([ACTION]) RJC([ACTION]) EJK([ACTION]) ;
Sub-Collections: SRI-ARC; Clerk: SRL;

EKM 14-NOV-74 08:26 24500

output remote printer

Output remote printer now works are SRI-ARC and will be fixed at Office-1 tonight. Susan tested it vigorously for us at RADC. It wants tip number followed by port number (both decimal). Jim would you notify any of your users that might need to know this? 1

EKM 14-NOV-74 08:26 24500

output remote printer

(J24500) 14-NOV-74 08:26;;;; Title: Author(s): Elizabeth K. Michael/EKM; Distribution: /JHB([ACTION]) SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: EKM;

DVN 14-NOV-74 09:26 24511

Can the Network Control Center Help with Unexpected Changes iin the TIP Intercept Character?

Copy of a message sent to Alex McKensie and others

DVN 14-NOV-74 09:26 24511

Can the Network Control Center Help with Unexpected Changes iin the TIP Intercept Character?

I am writing a users' guide for our Lineprocessor that assumes connection to a TIP. I hear from the field that from time to time something (noise on the telephone connection?) inadvertantly resets the intercept character. It's a rare occurence (once a week or less maybe in regular use) but when it happens it's a bind for our naive and even our expereinced users because in the context of other input it's very hard to realize what is happening. I can tell them to check for what is happening by sending a linefeed (in which case the TIP replis "BAD") but then there is no way that I can see for them to learn the current intercept character so they can reset tne TIP. Can they call the Network Control Center and ask you to tell them their intercept character? If so are you game for such calls? If not, can you suggest some other course? 1

DVN 14-NOV-74 09:26 24511

Can the Network Control Center Help with Unexpected Changes iin the TIP Intercept Character?

(J24511) 14-NOV-74 09:26;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /JOAN([ACTION] dirrt notebook please) &DIRT([INFO-ONLY]) ; Sub-Collections: DIRT SRI-ARC, Clerk: DVN; Origin: < VANNOUHUYS, TIPP.NLS;1, >, 11-NOV-74 20:47 DVN ;;;;####;

DVN 14-NOV-74 10:52 24512

Journal Problem: Ungraceful Failure after Sendmail Abort

Monday night I started to journalize a group in my file <mylin, >. I got, correctly I believe, the message that I had gone over my file space. I deleted some files, reset, and tried again, this tme without error messages. Two journal items resulted:<mjournal,24471,> sent at 8:09 which is my initial file and <mjournal,24473,> sent at 8:14 which is the right group. Of course it is barely possible that I accidentally instructed sendmail to send my initial file, but, since that would mean both that I had sent the wrong thing and used a different command (File instead of Group) it raises the question of wehther the journal system somehow decided on its own to send my initial file. For the next day or so I could not send journal items and got the message "vanNouhuys [SENDM,,,etc. is not an NLS file". Perhaps if the suggestion to retain the sendmail status, with notification to the user (24482),which I strongly support, had been operating I would have known what was goin on. 1

DVN 14-NOV-74 10:52 245

Journal Problem: Ungraceful Failure after Sendmail Abort

(J24512) 14-NOV-74 10:52;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /FDBK([ACTION]) DPCS([INFO-ONLY] a note to explain a journal item of my initial file you received the other day) ; Sub-Collections: SRI-ARC DPCS; Clerk: DVN;

DVN 14-NOV-74 11:01 24513

Bundling Sendmessages and Responsibility for Authorship

I believe in procdures such as those described in (mjournal,24393,) but problems arises when you have a bundle of sendmessages by various authors. First, it is easey to acknowledge authorship only of authors known to the ident system. A more serious problem arises with other authors' intent. On at least one occasion I jornalized just such dialogue and acutely vexed Dean and moderately vexed our good outside friend Duane Stone by immortalizing messages they thought were not ready for posertity. I had, as it happened asked their permision, but misundestood their reply in sendmessages (now lost). All this has made me super careful about getting autohr's consent. Being super careful can take months. 1

DVN 14-NOV-74 11:01 24513

Bundling Sendmessages and Responsibility for Authorship

(J24513) 14-NOV-74 11:01;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /SRI-ARC([INFO-ONLY]) DLS([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DVN;

DVN 14-NOV-74 11:04 24514

Allocating Joan's Time

Martin is laying a substantial amount of typing and editing work on Joan. She has plenty to do for Development and things like maintianing some of the notebooks get shoved back. I'm sure Martin needs the work, and I don't think we should be too rigid about not doing things for Applications, but I think Joan needs some guidelines in setting priorities and people who as for her help need to know about them. 1

DVN 14-NOV-74 11:04 24514

Allocating Joan's Time

(J24514) 14-NOV-74 11:04;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /RWW([ACTION]) MEH([INFO-ONLY]) JML([INFO-ONLY]) SLJ([INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DVN;

CHI 14-NOV-74 12:57 24515

An NSW Scenario

For a rough draft of an NSW scenario (as a vehicle for understanding the flow of control and dtat in the NSW) see <nsw-sources, scenario,>. 1

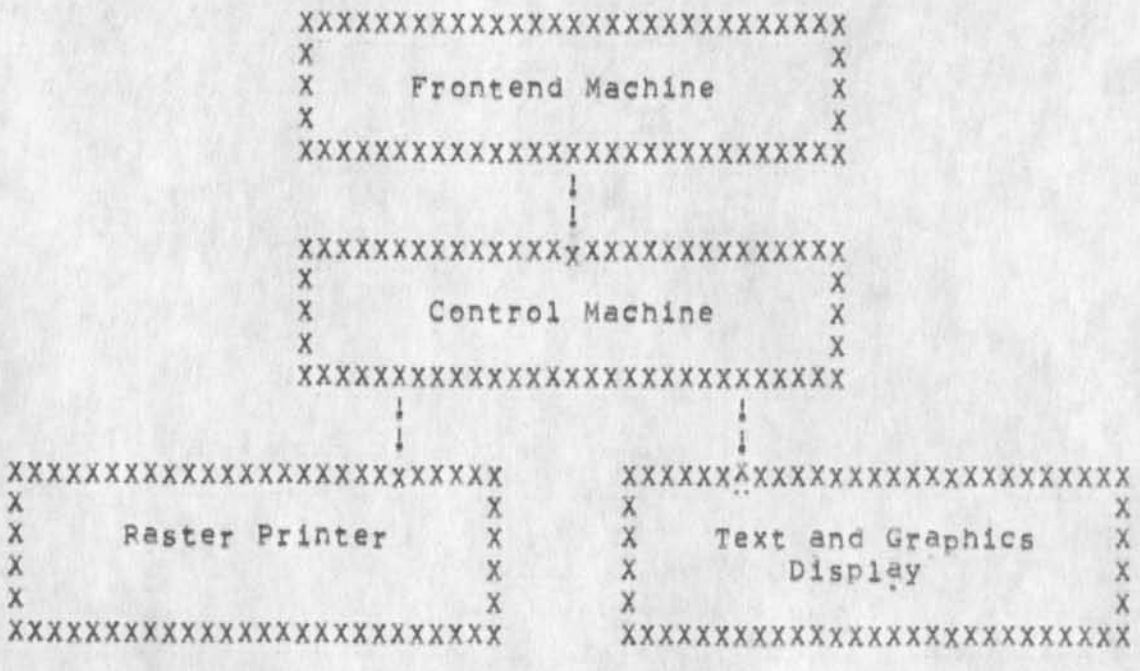
CHI 14-NOV-74 12:57 24515

An NSW Scenario

(J24515) 14-NOV-74 12:57;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /JEW([ACTION]) JBP([ACTION]) RWW([INFO-ONLY])
; Sub-Collections: SRI-ARC; Clerk: CHI;

Dynamic displays for mixed graphics and text.

The question of high quality text and graphics has come up. The following hardware configuration may serve to clarify the problem. 1



1a

RASTER PRINTER 2

DVN is investigation this area in detail. 2a

TEXT AND GRAPHICS DISPLAY 3

Paths of exploration 3a

Systems concepts 3a1

Mike Levitt proposes a 16 terminal cluster video system with the following: 3a1a

16 stations 3a1a1

8 512 x 512 bit maps which can be shared at the statons 3a1a2

512 7 x 9 writtable font memories 3a1a3

4k characters 3a1a4

hardware italics 3a1a5

Dynamic displays for mixed graphics and text.

$100k for the lot w/o a PDP 11 to control it or the keyboards etc. 3a1a6

This all sounds very interesting (although not right now), we shall visit Systems Concepts 11/25 for a demonstration. 3a1b

Evans and Sutherland 3a2

Picture provides the greatest potential for really high class graphics. I am continuing my communication with Mo Gunn of Institute for Advanced Computation (actually employed by E&S) for the general state of their system. 3a2a

MIT and Stanford AI experiments 3a3

At Mit-AI contact Tom Knight (617) 253-6765 or Pitts Jarvis (617) 253-1729 or 8617) 253-7807 3a3a

at SU-AI Jeff Rubin (415) 321-2300 x 4971 3a3b

(message sent 11/14) 3a3c

Ramtek 3a4

Ramtek (of sunnyvale) is probably the leader in video. A GX 100 system could be configured as follows: 3a4a

SX-100 basic controller $6,000 3a4a1

GX-7x7 768x768 bit map (max 4 per controller) $9,000 (845 lines) 3a4a2

GX-100-B PDP 11 interface $1,200 3a4a3

6K characters 7x9 dot matrix (not writable) 3a4a4

Vector General 3a5

Powerfully competer for E&S systems base at $50k. 3a5a

Current recomendations 3b

In addition to the $100K investment for the hardware for these systems, a very sizable investments in programming effort would be required. Particularly for the video displays, the state of graphics art would need to be advanced. For now, implemention of a hardcopy facility through the raster printer is more important than dynamic graphic development. NLS

graphic concepts can be satisfactorily initiated using the TEKETRONIC displays and possibly IMLAC's. 3b1

Dynamic displays for mixed graphics and text.

(J24516) 14-NOV-74 13:08;;;; Title: Author(s): Robert Louis Belleville/RLB2; Distribution: /RWW([ACTION]) DVN([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: RLB2; Origin: < BELLEVILLE, MAJOR-DISPLAY-SELECTION,NLS;1, >, 14-NOV-74 13:06 RLB2 ;;;;####;

RWW 14-NOV-74 16:53 24517

The Bug Facts of Life

Jim Bair and I had a conversation about a number of topics last night, one of which was bugs and the process of getting them fixed and it seems useful to share the main points with others. 1

1) In a system as large and complex as NLS there will always be bugs given the present state of the software art. The quality of software engineering and the skills of the people building NLS is far higher than generally available in the industry. 2

2) The level of resources needed to take a good development product and get it really shaken down is much higher by a factor of from 2-10 than it took to develop it in the first place. ARC does not have that level of resources and our applications clients should be selected on their ability to understand and support the prototype nature of the system. This does not mean that we should tolerate bugs, particularly in areas critical to any application need, but that we must not expect absolute perfection. 3

3) We just came up with a major new system and it daily gets more solid. There is no way we could use the system in all the ways the application people will so we can expect a steady flood of bug reports for some months. 4

4) Dave Hopper is coordinating the bug fixing process and gets help as he needs it. He has limited time and energy so has to make his determinations on bug priorities as do those helping him from time to time. They can only do this from their experience or from info they receive. Therefore, it is important for those reporting bugs to provide any info that can help this priority determinaion process, like, this bug seems obscure and unimportant in ARCs experience but is central to the application of client X because they use this feature constantly and are dependent on it etc. 5

5) Bugs should be reported through feedback and not orally because people who are fixing bugs need to have some buffer to maintain their sanity and perspective on priorities. I am one of the worst offenders here and really feel that no bug should be accepted orally that has not been reported to feedback or is so obscure that when it happens it must be caught now (I think these should be infrequent). 6

6) The Development side of the house has got its neck way out to deliver some very sophisticated things by next July and must concentrate on these tasks and therefore has limited resources to fix bugs that are not absolutely urgent. We have just as much pressure on us as Applications has on them and both sides of the house need to understand and respect the pressures that exist on the other and help each other as we can, understanding may be the best way. We take great pride in our product and would like to stop and fix bugs, fill every hole etc, but we will not be in business long in our highly

competitive world if we can not keep moving on. As part of Applications business plans they must allocate more funds over time for the ongoing bug fixing, and hole filling that will always be there. 7

7) Applications may not realize it but they are getting far more support than they have reason to expect and we are glad to give it because of the importance of applications to long term funding and development strategy. 8

8) The same flavor of comments as above applies to documentation. 9

RWW 14-NOV-74 16:53 24517

The Bug Facts of Life

(J24517) 14-NOV-74 16:53;;;; Title: Author(s): Richard W. Watson/RWW; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RWW;

JAKE 14-NOV-74 21:53 24518

User-programs

I too agree with JEW's suggestion for maintaining user-programs; however, I would like to see us go beyond NLS with this particular activity. One of the things every user wants to know is 'What programs and databases are available for me to use at the utility and on the Arpanet'. Our utility customers, and particularly the Air Force, would really like to know this information (In the past I have averaged two or three requests a week of this type from Air Force personnel). To date most efforts to collect resource data have only tapped the top of the iceberg. The only way I can see that collection of such information will work is if the users who have programs to offer are encouraged to easily enter a description of their program along with other pertinent data into an interactive online file. 1

2

My suggestions for such a program-collecting system would be the following: 3

- The system would be very easy for users to access and to enter data. It should not be several layers down in another system (that is it should not 'appear' to be to the user.). 3a

- It should be accessible for input and viewing from either TENEX or NLS. (and ultimately from any network system) 3b

- The database should be primarily user built with an editor (human) who can oversee the entries for consistency, timeliness, current availability, etc. (whatever is needed) The editor could also add entries, publicize the system, promote its use, and pull off subsets of information for publication. 3c

- A description of an available program should contain: 3d

  - the program name, 3d1

  - a brief description of the program, 3d2

  - a brief scenario on how to access the program, 3d3

  - the author or designer of the program 3d4

  - a contact (if different from the author or designer), 3d5

  - links to pertinent documentation (which would hopefully be online and accessible for ftp) 3d6

  - whether the program is being maintained and to what degree 3d7

- who may use and conditions of use, if any 3d8

The ultimate system would be one that would do what an MIT (I believe) system has attempted. That is, it would let the user search for and select a program, then type the program name after which the system would automatically go across the network, log into the foreign host, load the program, and present it to the user as if it were local (and presumably bill him for any charges.) This, of course, is the ultimate and not likely to happen overnight, but it also is, in the long run, what networking is all about. For that reason I think any design we come up with should not exclude this long range goal, even though the beginning may be something very much simpler. 4

The NIC has collected some information along these lines and would be glad to have a way to maintain it and search through it. Currently there is a very simple program called HACK available in <netprog> that can handle the input of program descriptions in TENEX. It is very simple minded, but is nevertheless alive and usable. Anyone who wants to look at it or play around with it is welcome to do so. 5

User-programs

(J24518) 14-NOV-74 21:53;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /SRI-ARC([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: JAKE; Origin: < NICPROG, SENDMAIL.NLS;2, >, 14-NOV-74 21:51 JAKE ;;;;####;

DIA 14-NOV-74 22:03 24519

Factors in L10/BCPL Language Decision for NSW Frontend

Introduction 1

Sometime in August, 1974, ARC decided to develop an L10 compiler for the PDP-11 for NSW Frontend development. The primary alternative was to use BCPL. This is a reconstruction of the reasons for going the L10 route. 1a

These reasons were weighed at the time of the decision but were only partially recorded at that time. 1b

At this writing, about one man month has gone into writing the L1011 compiler. The compiler is about 90% complete and we expect about one more man month of development and debugging before we have a usable compiler and runtime package. 1c

L10 is a more desirable system programming language in several respects: 2

Notes: 2a

These points are based on our reading of the BCPL manual (BBN, Sept. 1974 and earlier copies that do not have dates on them) and on discussions with Bruce Parsley, Bill Duvall and Smokey Wallace of XEROX PARC (heavy users of BCPL). 2a1

The following statements are made from an L10 programmer's point of view. Some points may not be clear to readers unfamiliar with the L10 language. For clarification, they should see the L10 document (7052,) or an ARC programmer. 2a2

If any of these statements are wrong, we would like to know about it. Send messages to ANDREWS, IRBY, and VICTOR @SRI-ARC. 2a3

We do not want a polarized BCPL vs. L10 situation. We felt reluctant to put L10 on the PDP-11. But putting L10 on the PDP-11 appeared to be the best route for us when we compared the short and long term costs of going each way. 2a4

The results of our comparison of L10 and BCPL follow: 2a5

The SWITCHON statement in BCPL is nowhere near as powerful as the CASE statement in L10. There are deficiencies in the EXIT and REPEAT facilities: 2b

There is NO way to do the L10 equivalent of REPEAT CASE n or REPEAT LOOP n 2b1

There is NO way to do the L10 equivalent of REPEAT CASE (exp) . 2b2

There is NO way to do the L10 equivalent of EXIT CASE n or EXIT LOOP n 2b3

There is no easy way to do the L10 equivalent of REPEAT CASE. 2b4

The above deficiencies would probably necessitate a change in algorithm when translating existing code into BCPL. 2b5

The BCPL FOR loop has the following problems: 2c

The condition for exiting the loop is only computed once. Thus if the condition is an expression whose elements change during the execution of the loop, these changes may not be reflected when checking to see if the loop should be exited. (This may be an implementation problem of a specific BCPL.) 2c1

The FOR loop index variable increment must be a compile time constant and can not be a runtime variable. 2c2

The FOR loop index variable has a scope that is local only to the FOR loop, and thus when you have exited the FOR loop you can not find out how many times you went throught the FOR loop 2c3

Strings are limited to a max length of 255 characters. Also, string operations are limited. 2d

No assembly code is allowed. L10 allows the programmer to place assembly code anywhere in the L10 flow. 2e

This could necessitate the writing of many small assembly code procedures and would reduce efficiency. In the past, we have identified bottlenecks in L10 code and recoded them in assembly statements to make them faster. We would obviously not be able to do that so easily in BCPL. (We would only want to do this in the operating-system-interface code). 2e1

You can't do an assignment as part of an argument in a procedure call, e.g. proc(i←x) 2f

There is no equivalant of the L10 := type assignment (exchange). 2g

Procedures can return only one value 2h

There appears to be no arithmetic AND function, only a boolean AND 2i

You cannot declare static tables 2j

There are no signal mechanisms. 2k

NLS algorithms are generally predicated on the existance of a signal mechanism. Under BCPL, we would create one via assembly code procedures. 2k1

Of course, all of these deficiencies in the language can be circumvented. However, experience has shown that programming around problems leads to bugs and makes system maintenance much more expensive in terms of man-hours and smashed keyboards. 2l

Some of these deficiences would not be considered if ARC programmers were not used to L10, or if NLS were not already written in L10. However, the weak CASE statement, the poor string facilities and the lack of signals are serious from the standpoint of the NSW programming task -- that is, those facilities are important for a reasonable NSW frontend implementation. 2m

Using BCPL would necessitate either making an extra step in the compilation process or interfacing NLS to the BCPL compiler. 3

This is the problem of getting NLS files to a compiler that wants text files. 3a

In addition, BCPL creates an assembly language text file -- which makes an ADDITIONAL step in the compilation process. A meta-generated compiler such as L1011 produces REL files directly from NLS files, whereas with BCPL there would be three compute-bound operations -- NLS->text, Compile, Assemble. 3b

ARC programmers would have to learn a language of quite different syntax. 4

All constructs in BCPL are of different syntax than any other major programming language (except its father, CPL). Also, curly braces and other special characters are used -- creating a character set problem for most of our output devices. 4a

The major syntactic elements in L10 have semi-corresponding elements in BCPL, but there are some subtle differences that programmers would have to learn. 4b

E.G. alternatives in a CASE statement have to be constants; you can say IF <exp> THEN <st> but you have to say TEST <exp> THEN <st> ELSE <st>; etc. 4b1

Breaking into BCPL would probably take 1 or more man months of ARC programmers' time (collectively). 4c

There may be some extra mental burden in switching back and forth between languages. 4d

ARC would be dependent on BBN for maintenance of the BCPL compiler. 5

We have the highest regard for our colleagues at BB&N. But since we would not be contracting a service from them, we fear that they would not be very motivated to fix bugs that were not in their way, but in ours. Bug fixing could take months -- hence we would program around bugs (which is bad coding practice and always creates problems later). Problems in our own L10ii compiler would be very important to us and we would fix the compiler. 5a

Future NSW frontend/backend systems would be in two different languages. 6

Staying with one language is more than convenient for programmers. Procedures that are not system-interface procedures could be moved wholesale from the backend to frontend and vice versa, if the same language is used for both parts. (Rewriting the backend of NLS in BCPL for the PDP-10 is clearly are large task!) 6a

In addition, if we did use BCPL for both frontend and backend, we would find several problems in moving BCPL code from a PDP-10 to a PDP-11 and vice versa - such as incrementing vector indices by two on the 11 and by one on the 10, not using quarter word operators on the 11 but OK on the 10, and avoiding crossing word boundaries in record definitions. 6b

We are being as careful as we can to avoid such difficulties with the L10 and L10ii compilers. We are also providing primitives to make writing code for both machines reasonable -- such as builtin symbols for address length and word size. 6b1

We expect to be able to extend these aids to other machines. 6b2

We plan to move the NLS backend to other backend machines in the future. It is more attractive to move the language NLS is written in than it is to rewrite NLS in different languages for each backend -- attractive because it makes system maintenance and further development much much easier. We are carefuly designing the operating system interface to make this possible. 6c

Moving code from backend to frontend is an issue here since 1) the NSW frontend will exist on both PDP-10's and PDP-11's and 2) we plan to move heavily used parts of NSW backend execution modules into the NSW frontend in the future. 6d

DIA 14-NOV-74 22:03 24519

Factors in L10/BCPL Language Decision for NSW frontend

(J24519) 14-NOV-74 22:03;;;; Title: Author(s): Don I. Andrews/DIA; Distribution: /SRI-ARC([INFO-ONLY]) RMB([INFO-ONLY]) SDC2([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: DIA; Origin: < ANDREWS, BCPL/L1011.NLS;6, >, 14-NOV-74 21:55 DIA ;;;;####;

DVN 15-NOV-74 08:51 24520

Writeup on Accideetal TIP Reset

Following conversations withAlex McKensie, I have rewritten the sectionon this problem as it appears in <vannouhuys,novguide,4g1b> et. seqs. I have passed copies for review to Jim Bair and Norton. McKensie has some interest in the problem and is willing t have users call when it happens atleast until they understad it, but not to have that be an intructionin the manual. Network control center can tell you the inercept character set at a given port.

DVN 15-NOV-74 08:51 24520

Writeup on Accideetal TIP Reset

(J24520) 15-NOV-74 08:51;;;; Title: Author(s): Dirk H. Van Nouhuys/DVN; Distribution: /DIA([ACTION]) JOAN([ACTION] for dirt notebook) MEH([INFO-ONLY]) JCN([INFO-ONLY]) RWW([INFO-ONLY]) ; Sub-Collections: SRI-ARC DIRT; Clerk: DVN;

JAKE 15-NOV-74 09:00 24521

Journal delivery

I am still not getting delivery of author items to my journal files. Also on Monday or Tuesday my initial file went bad and I requested that an older version be returned. So far this has not happened. I will request the old file back from the operator again. (Jim, last is just for the record, I will handle. First is apparently a bug which I cannot handle. Jake 1

JAKE 15-NOV-74 09:00 24521

Journal delivery

(J24521) 15-NOV-74 09:00;;;; Title: Author(s): Elizabeth J. (Jake) Feinler/JAKE; Distribution: /FEED([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JAKE;

KJM JDH EKM 15-NOV-74 09:41 24522

Procedures for Splitting NLS

The procedures laid out here were worked out by Dave Hopper and the NLS group with the fervent hope that they will enable us to maintain stable running systems are SRI-ARC and at OFFICE-1 while forging ahead with NSW work. Please read promptly, carefully, and CRITICALLY. Let us know if you detect flaws and cracks.

KJM JDH EKM 15-NOV-74 09:41 24522

Procedures for Splitting NLS

IMPORTANT: MAKE NO (ZERO) CHANGES TO NLS SOURCES UNTIL YOU HAVE READ THIS MEMO. MAKE NO CHANGES TO NIC-NLS SOURCES TODAY, FRIDAY, WITHOUT CHECKING WITH DAVE HOPPER 1

The time has come to split NLS into a Backend and Frontend. In particular, the changes to allow only integers to be passed between CML and xroutines must be done now. We'd like to avoid having several directories of NLS source files in an attempt to reduce the confusion, so the NLS split changes will be done in the directory NLS. 2

The integer passing changes will be done in NLS sources immediately and, when working, will be brought up as the running systems at SRI-ARC and OFFICE-1. Changes to record Line Processor errors will also be done immediately and brought up as the running systems at the same time. From now until further notice, no other activity should take place in the directory NLS. In particular, Bob Belleville should make a private copy of the files he needs to change for his file system changes and also lock those files in the NLS directory so any other attempts to change them are clearly noted and coordinated. This is a bit tricky as the changes for the new CML will probably destroy things for a while. 3

Bug fixes to the system and the creation of a test and running systems at SRI-ARC will be done in the directory NIC-NLS. Some careful procedures should be followed so we (hopefully) won't find ourselves thoroughly confused and fouled up. Dave Hopper is the coordinator of all changes to NIC-NLS. In theory, none of us will make changes except at his request. Unfortunately, that would pretty well clog the works, so we should follow the following procedures to make changes. 4

1. Before changing the NIC-NLS files talk to Dave Hopper about the change, if possible. 4a

2. Make the necessary edits to the sources in NIC-NLS and update the file(s) in NIC-NLS. If it your custom to work with a private copy of the source file, leave the master copy locked. 4b

3. Follow normal procedures for testing the change in the user program environment. 4c

4. Compile the file(s) using to the (todo) branch of NLS,TASKS. 4d

5. Add an entry to the NIC-NLS-CHANGES branch of tasks that contains the following information::
   - what files were changed
   - who and when, if the statement signature is not accurate

- enough of a description to remind you later what the change was about 4e

6. Create a test system (for testing bug fixes to the running system) by processing the commands branch (make-rnls) in NLS,TASKS. This is the system you will get when you type "work". Note that this branch uses the new RUNLDR file in NIC-NLS. 4f

7. Test 4g

8. All bug fixes to the running system (NIC-NLS) must be reflected in the experimental split system, NLS. 4h

9. When it's time to make this test system the running system at SRI-ARC, coordinate with JDH. He'll be maintaining a directory of all the files changed for the current running system over the previous system, and checking that set of files against the notes he received in TASKS, and generally keeping the world in order. Until procedures are developed and demonstrated to work for coordinating this directory, files should only be written in this directory under explicit instructions from Dave Hopper. With any luck at all, a running system can be tested thoroughly and moved to OFFICE-1 before another running system has to be brought up at SRI-ARC. If that's not possible, JDH will have to do some kind of magic to keep the source files straight. DON'T BYPASS HIM! 4i

10. When the running system has been thoroughly tested at SRI-ARC, JDH will move it to OFFICE-1, updating sources at OFFICE-1 and cleaning out the interim directory. 4j

CAUTION: Dave has tried to change the file statement in all the souces to point at the NIC-NLS rel file and we should check to make certain this has been done before doing a compile of a NIC-NLS file. 5

Karolyn Martin is coordinating the NLS split work and should be kept informed so that she can keep us from messing one another up. 6

Procedures for Splitting NLS

(J24522) 15-NOV-74 09:41;;;; Title: Author(s): Karolyn J. Martin, J. D. Hopper, Elizabeth K. Michael/KJM JDH EKM; Distribution: /NPG([ACTION]) RWW([INFO-ONLY]) JCN([INFO-ONLY]) NDM([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: EKM; Origin: < MICHAEL, INFO,NLS;5, >, 15-NOV-74 09:21 EKM ;;;;####;

SCHEDULE F (updated, 11/14/74) 1

COMPUTER SUPPORT COSTS 1a

1) PDP-10 TENEX Computer Time 1b

a) July 1, 1974 to December 31, 1974 GFE 1b1

b) January 1, 1975, to June 30, 1975
5.5 job slots for 6 months * 1b2

16,765 x 5.5 = $ 92,201 1b2a

2) PDP-11 Systems 1c

a) Equipment 1c1

1) PDP-11 Development Machine
$1,483/mo x 12 = 17,796 1c1a

$565/mo x 7 =(tape drive and card reader) 3,955 1c1b

2) ANTs interface = 10,775 1c1c

b) Maintenance (8 hrs/d, 5 d/wk) 1c2

1) DEC PDP-11,
$473/mo x 7 = 3,311 1c2a

3) Terminals 1d

a) NLS workstations (7) 1d1

1) Display (7)
a) Hazeltine display $ 88/mo x 1 x 6 = 528
b) Data Media displays $80/mo x 3 x 6 = 1,440
c) Data Media keyboards $150/mo x 3 x 6 = 2,700
d) Lear Seigler displays $138/mo x 3 x 6 = 2,484
e) Tables for work stations, one time charge

at $100 estimated each x 7 =	700	
f) Tektronix graphics and hardcopy units		
4012 at $300/mo x 6 =	1,800	
4014 and hardcopy unit at $880/mo x 2 =	1,760	
d) Line processor modifications to accept the Tektronix units (estimated)	400	
		1d1a
b) TNLS Terminals		1d2
1) TI (incl maintenance) (4) $165/mo x 4 x 12 =	7,920	1d2a
2) Maintenance, owned TI's (8) $20/mo x 8 x 12 =	1,920	1d2b
3) Acoustic couplers (8) $16/mo x 8 x 12 =	1,536	1d2c
c) Modems		1d3
1) Dial-up (4) $36/mo x 4 x 12 =	1,728	1d3a
d) Leased lines		1d4
1) DIA, data $366/mo x 12 =	4,392	1d4a
2) DIA, voice $22/mo x 12 =	264	1d4b
4) Tasker Display System (10 units for 3 mo)	GFE	1e
5) Miscellaneous (estimated) =	451	1f
	--------	1g
Subtotal (Items 2-5)	$ 65,860	1h

NSW: ISU 74-132 Schedule F Revised and Updated

Total (Items 1-5)	$158,061

11

MEH 15-NOV-74 10:04 24523

NSW: ISU 74-132 Schedule F Revised and Updated

(J24523) 15-NOV-74 10:04;;;; Title: Author(s): Martin E. Hardy/MEH; Distribution: /RWW([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: HDW; Origin: < HARDY, NSWSCHF,NLS;7, >, 15-NOV-74 09:32 HDW ;;;; ####;

KEV 15-NOV-74 14:19 24524

re (24497,)

the first problem is with the CML and not with the SYNTAX GENERATOR in that the CML actually has JUMP NAME BUG for the TNLS JUMP. the other problem is a problem that i have been aware of for some time: 1

supervisor commands are available within the SYNTAX command regardless of which subsystem the user is in at the time of issuing the SYNTAX command 1a

however, do to some bug, this is not the case when you issue the SYNTAX command the first time 1b

conclusively, i have no plans for dealing with any of this stuff, and i think if it is to be fixed someone in the nls maintainae group should have this responsibility 2

KEV 15-NOV-74 14:19 24524

re (24497,)

(J24524) 15-NOV-74 14:19;;;; Title: Author(s): Kenneth E. (Ken) Victor/KEV; Distribution: /JMB([INFO-ONLY]) CHI([INFO-ONLY]) RWW([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: KEV;

JEW CHI RWW 15-NOV-74 14:40 24525

Need for an NSW Design Document

Letter to Warshall and Millstein with copies to all main NSW people by sndmsg.

JEW CHI RWW 15-NOV-74 14:40 24525

Need for an NSW Design Document

Steve and Bob,

With the NSW communication protocol (PCP) design now fairly stable, we are beginning to think at a higher level about how the various pieces of the NSW system (WM, FE, and tools) will be fitted together -- what data structures are required, when and by whom they will be accessed, the flow of control, and so forth. Since it implements the system model which will be presented to the user, the WM plays a major role in dictating the nature of this interconnection.

As we proceed from our end, we are becoming ever more convinced of the need for a written design document laying out our collective model of the whole NSW. Such a document is important for several reasons, chief of which is to provide a tangible design which we can all refine as we find ourselves thinking about issues in greater detail and gaining insights into both how the system should appear to the user, and functon internally. The black and white box descriptions you've supplied us in our role as protocol supplier imply something about the higher-level design, but are not in themselves sufficient to meet this rather different need. We believe the NSW design document should contain at least the following: fairly detailed models of the file and accounting systems, as seen by the user; and a scenario, both from the user's and system's viewpoints, of the use of a typical tool.

We think it appropriate that you, as WM designers, assume primary responsibility for this document. We've been forced to begin a pass at it ourselves, to clarify our own thinking, and will be happy to provide that as input to your effort when it's more complete. However, we feel that you should proceed with your document in parallel and have a draft completed and distributed by the end of November, which we can then discuss at our December meeting. We are finding as we go through this exercise that our understanding is deepening and in fact our concepts of what a tool is is changing and so forth. It is very important that our models and understanding converge before the end of the year. We think this can only happen through a document. Meetings are not in them selves sufficient, although useful as stimulation.

Although the generation of written designs takes time, we think it time that absolutely must be spent, if we're going to pull this thing off. We've taken this approach with PCP and found it absolutely crucial; if we hadn't laid out our thought in writing, we would have produced a completely different -- and, we believe, inferior -- protocol.

JEW CHI RWW 15-NOV-74 14:40 24525

Need for an NSW Design Document

(J24525) 15-NOV-74 14:40;;;; Title: Author(s): James E. (Jim) White, Charles H. Irby, Richard W. Watson/JEW CHI RWW; Distribution: /NPG([INFO-ONLY]) JBP([INFO-ONLY]) DCE([INFO-ONLY]) JCN([INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: RWW; Origin: < WATSON, NSWM.NLS;2, >, 15-NOV-74 14:34 RWW ;;;;####;

JBP 15-NOV-74 17:39 24526

Padlipsky's Common Command Language (CCL)

< POSTEL, CCL,NLS;1, >, 15-NOV-74 16:12 JBP ;;;; 1

M.A. Padlipsky 1a

Beyond the Telephone Line Surrogate:
Specification of the Unified User-Level Protocol 1b

After many discussions of my RFC 451, I discovered that the "Unified User-Level Protocol" proposed therein had evolved into what had always been its underlying motivation, a common command language. There are several reasons why this latter approach satisfies the original goals of the UULP and goes beyond them into even more useful areas: 1c

1. User convenience. As evidenced by the good response to the common editor "neted", the Network Working Group has come to acknowledge the fact that the convenience of non-system programmer users of the Network must be served. Allowing users to invoke the same generic functions -- including "batch" jobs -- irrespective of which Server Host they happen to be using is surely a compelling initial justification for a common command language. Note that the concern with generic functions -- which "all" Servers do, one way or another -- is intended to emphasize the common command subset aspects of the language, rather than the "linguistic" elegance of it all. The attempt is to specify an easy way of getting many things done, not a complicated way of getting "everything" done. 1d

2. "Resource sharing". Another area which is receiving attention in the NWG of late is that of "automatic" or program-driven invocation of resources on foreign systems. A common intermediate representation of some sort is clearly necessary to perform such functions if we are to avoid the old "n by m problem" of the Telnet Protocol -- in this case, n Hosts would otherwise have to keep track of m command languages. For the common intermediate representation to be human-usable seems to kill two birds with one stone, as expanded upon in the next point. 1e

3. Economy of mechanism. In RFC 451, I advanced the claim that a single user-level protocol which connected via socket 1 and Telnet would offer economy of mechanism in that new responders would not be required to service Initial Connection Protocols on socket after socket as protocol after protocol evolved. This consideration still applies, but an even greater economy is visible when we consider the context of resource sharing. For if the common command language is designed for direct employment by users, as the present proposal is, there is no need for users on terminal support "mini-Hosts" (e.g., ANTS and TIPs) to require an intermediary Server when all they actually want is to work on a

particular server in the common language. (This is especially true in light of the fact that many such users are not professional programmers -- and are familiar with no command language.) That is, if resource sharing is achieved by an intermediate language which is only suitable for programs, you would have to learn the native command language of Server B if you didn't want to incur the expense of using Server A only to get at generic functions on Server B. (And you might still have to learn the native language of Server A, even if the expense of using two Servers where one would do isn't a factor.) 1f

4. Front-ending. Another benefit of the common command language proposed here is that it is by and large intended to lend itself to implementation by front-ending onto existing commands. Thus, the unpleasant necessity of throwing out existing implementations is minimized. Indeed, the approach taken is a conscious effort to come up with a common command language by addition to "native" command languages rather than by replacement, for the compelling reason that it would be unworkable as well as ill-advised to attempt to legislate the richness represented by existing command languages out of existence. Further, as it is a closed environment, no naming conflicts with native commands would arise. 1g

5. Accounting and authentication. As evidenced by the spate of RFCs about the implications of the FTP in regard to both accounting for use of Network services and authenticating users' identifications (Bressler's RFC 487, Pogran's RFC 501, and my RFC 505 -- and even 491), this area is still up in the air. The generic login command proposed here should help matters, as it allows the Server to associate an appropriate process with the connection while actuating appropriate accounting and access control as well, if it chooses. 1h

6. Process-process functions. By enabling the invocation of foreign object programs, the present proposal offers a rubric in which such process-to-process functions as "parallelism" can be performed. (See the discussion of the "call" command, below.) Note that the UULP is not being advanced as a panacea: It is assumed that the actual transactions carried out are most likely not going to be in the common command language (although some certainly could be); however, what is furnished is a known way of getting the presumably special-cased programs executing elsewhere. Also, it offers a convenient environment into which can be placed such new functions, which we would like to have become generic, as Day's File Access Protocol. 1i

All of which seems to be a fair amount of mileage to get out of a distaste for remembering whether you find out who's logged in by saying "systat", "users", "s.who:c", "listf tty", or "who"... 1j

Padlipsky's Common Command Language (CCL)

Context 1k

Although ultimately intended to become the general responder to the Initial Connection Protocol, the UULP is initially to be a Telnet Protocol "negotiated option". When the option is enabled, the Server Host will furnish a command environment which supports the common conventions and commands discussed herein. 1l

In a sense, the UULP is a "selector". That is, the common command subset includes commands to exit from the common command environment and enter various other environments, along the lines of CCN's current Telnet Server. To exit from the UULP environment to the "native" command processor, the UULP command is "local" (see also the discussion of Case, below). Note that all commands terminate in Telnet "Newline" (currently cr-lf), unless altered by the "eol" command (below); internal separator is space (blank). (Entrance into other environments -- such as the FTP Server -- is discussed below.) There are two reasons for introducing a mechanism other than the apparently natural one of simply de-negotiating the option: First, it is bound to be more convenient for the user to type a command than to escape to his User Telnet program to cause the option disabling. Second, it is hoped that eventually the UULP will be legislated to be the default environment encountered by any Network login, in which case the natural way to enter the Server's "native" command environment would be by UULP command. 1m

Note: all UULP commands discussed herein are listed in Appendix 1, categorized as to optionality, with brief descriptions given. The appendix may be taken as a first-pass UULP Users' Manual. 1m1

Responses 1n

Any optional commands which are not supported by a particular Server are to be responded to by a message of the form "Not implemented: commandname,", where the variable is the name of the command which was requested. Note that throughout this document, all literals must be sent exactly as specified, so as to allow for the possibility of Servers' being driven by programs (including "automata" or "command macros") in addition to "live" users. 1o

In general, the view has been taken here that a small number of literal, constrained responses is superior to a vast variety of numerically coded responses in which text may vary. Again, the motivation is to achieve an economy of mechanism. For on the coded model, there must be a coordinator of code assignments, which is just as well avoided. Further, as has been experienced in the use of the FTP, when there are many codes there are many ambiguities.

(The sender may have a perfectly valid case for choosing, say, 452, while the receiver may have an equally good interpretation of the codes' definitions for expecting, say, 453.) Experience with a related "error table" mechanism on Multics also bears out the assertion that coded responses create both managerial and technical problems. A final objection to numeric codes might be considered ireelevant by some, but I think that the aesthetics of the situation do merit some attention. And when the common command language is being employed by live users, it seems to me that they would only be distracted by all those numbers flying around. (Nor can we assume that the numbers could be stripped by their "User UULP", for one of the basic goals here is to make it straightforward enough for a user at a TIP to deal with.) 1p

Arguments 1q

During the review process, it became evident that some global comments on arguments were in order. Two areas in particular appear to have led to some confusion: the strategy of specification of arguments on the commannd line, and the question of "control arguments". On the first score, the goal of "front-endability" must be recalled. Consider two native implementations of a particular command, one of which (A) expects to collect its arguments by interrogation of the user, and the other of which (B) expects to receive them on invocation (being invoked as a closed subroutine). Now, it is easy to imagine that a "Server UULP" could feed the arguments to A as needed without requiring A to be rewritten, but it is quite difficult to see how B could be made to interrogate for arguments without extensive rewriting. Therefore, a "least common denominator" approach of specifying arguments in advance incurs the minimum cost in terms of reworking existing implementations. 1r

On the second score, I have borrowed a notion from the Multics command language's convention called "control arguments" because it seems to be quite convenient in actual practice. The key is that some arguments are meant as literals, usually specifying a mode or control function to the command, while others are variables, specifying something like a particular file name or user identifier. A common example is a "mail" command, where the variables are the user identifiers and the Host identifiers, and the "control argument" is the designator that user identifiers have ceased and Host identifiers have begun. The convention used here is to begin the control argument with a hyphen, as this character never seems to be used to begin variable arguments. Thus, we use "-at" in the mail example. Although it is not a deep philosophical point, this approach does relieve argument lists of order-dependency, and feels right to me. 1s

Case 1t

Although it appears to have been legislated out of existence by the specification of the Network Virtual Terminal's keyboard in the Telnet protocol, the question of what to do about users at upper-case-only terminals remains a thorny one in practice. There are two aspects to consider: the alphabetic case of commands, and the ability to cause "case-mapping" in order to allow lower-case input. Some Servers have no local problems with the first aspect, as they operate internally in all upper-case or all lower-case and merely map all input appropriately. (Problems do arise, though, when one is using the User FTP on such a system to deal with a mixed-case sytem, for example.) Other Servers, however, attach the normal linguistic significance to case. (E.g., Smith's name is "Smith" -- not "SMITH", and not "smith".) To minimize superfluous processing for those Servers which are indifferent to case, all UULP commands are to be recognized as such whether they arrive as all upper-case or all lower-case. (They will be shown here as all lower merely for typing convenience.) Note that arbitrarily mixed case is not recognized, as it is an unwarranted assumption about local implementation to suppose that input will necessarily be case-mapped. 1u

On the second aspect, any Server which does distinguish between upper- and lower-case in commands' arguments (a.k.a. parameters) must furnish a UULP "map" command as specified in Appendix 2 in order to support logins from upper-case-only terminals attached to User Hosts which either do not support the Telnet Protocol's dictum that all 128 ASCII codes must be generable, or support it awkwardly. This seems a simpler and preferable solution than the alternative of legislating that upper-case Network-wide personal identifiers (and perhaps even Network Virtual Path Names) be pre-conditions to a usable common command subset. (As noted below, these latter concepts will fit in smoothly when they are agreed upon. The point here, though, is that we need not deprive ourselves of the benefits of a UULP until they are agreed upon.) 1v

User Names 1w

As implied above, the various Servers have their various ways of expressing users' names. Clearly, the principle of economy of memory dictates that there should be a common intermediate representation of names in and for the Network. It is probably also clear that this representation will be based upon the Network Information Center's "NIC ID's". However, it is unfortunately amply clear than an acceptable mechanism for securing up-to-date information cannot be legislated here -- muchless a mechanism for securely updating the implied data base. Therefore, at this stage it seems to be the sensible thing to specify only the UULP syntax

for conveying to the Server the fact that it is to treat a user name as a Network-wide name rather than as a local name, and let the supporting mechanisms evolve as they may. 1x

The prefacing of a name with an asterisk ("*") denotes a Network-wide name. (Such names may be either all upper-case or all lower-case, as with UULP commands' names.) The name "*free" is explicitly reserved to mean that (in the context of logging in) a login is desired on a supported or sampling account, if such an account is available. The response if no such account is available is to be "Invalid ident: *free." When Network-wide names are generally available Servers will either map them into local names or cause them to be registered as local names as they prefer. The point is that a Network-wide name will be "made to work" by the Server in the context of the UULP. 1y

Special Characters and Signals 1z

Another area in which the facts of life must outweigh the letter of the Telnet Protocol if the user's convenience is to be served is that of "erase" and "kill" characters. It is possible that User Telnets will uniformly facilitate the transmission of the Telnet control codes for generic character erase and generic line kill. It is certain, however, that User Telnets will differ -- and users will, if they use more than one User Telnet, be again placed in the uncomfortable position of having to develop too many sets of reflexes. Therefore, the UULP will optionally support the following commands: "erase char" and "kill char", where char is a printable ASCII character (to avoid possible conflicts with "control characters" which are recognized in the innermost areas of particular operating systems). Presumably, unwary users can be instructed not to choose an alphabetic, so as to avoid being placed in a position where they cannot invoke certain commands (erase and kill themselves, for example, in which case they couldn't be changed). 1a@

These commands are supplements to the related Telnet control codes, and have the same meanings. The point here is that it may be far more convenient for a user to be able to say "erase #" and get the "#" to be recognized as the erase character by the Server than for the user to get his User Telnet to send the Telnet equivalent. The commands are designated as optional because they may lead to severe implementation problems on some Servers, and because the equivalent functions do, after all, exist in Telnet. 1aa

> Note: the erasing is assumed to be performed "as early as possible". That is, the sequence "erase x" "erase x" should come out equivalent to "erase x" "erase" -- the second appearance of "x" resulting in the erasing of the space in the

command line. Presumably, this is a sufficiently uncommon path that anomalous results would be tolerated by the user community, but the intent ought to be clear. 1aa1

The Telnet "synch" and "break" mechanisms are, by their very nature, best left to Telnet. End of line, however, might well be a different story. Therefore, as a potential convenience, the UULP optionally supports "eol char" to ask the Server to treat char as the end of line character thenceforth. To revert to Telnet Newline, "eol" (i.e., no argument, current terminator). 1ab

Prompts 1ac

Another aspect in which Servers vary while being the same is how they indicate "being at command level". Some output "ready messages"; others, "prompt characters". For the UULP, where some functions will be performed by means of a command's logging in to another system, the ability to specify a known prompt character is extremely desirable. The UULP command is "prompt char" where char is the character which is to be sent when the user's process (on the Server) is at command level. It is explicitly permitted to prefix char to a line consisting of a "native" prompt or ready message. Also, this command is explicitly acknowledged to be permissible prior to login. (Again, warning must be made of the bad results which can ensue if an alphabetic character is chosen.) 1ad

Note: "prompt", "eol", "erase", and "kill" may all be re-invoked with a new value of char in order to change the relevant setting; all may be turned off by invocation with no argument. 1ad1

Login 1ae

Perhaps the stickiest wicket of them all is the attempt to specify a generic login, but here we go. The UULP login command is "login userident", where userident is either a locally-acceptable user identifier or a Network-wide identifier as discussed above. Note that for utility in contexts to be discussed later, the locally-acceptable form must not contain spaces. Servers may respond to the login attempt with arbitrary text (such as a "message of the day"), but some line of the response must be one of the following: a prompt (as discussed above; indicating, in the present context, successful login); "Password:"; or "Invalid ident: userident." When passwords are required, it is the Server's responsibility either to send a mask or to successfully negotiate the Hide Your Input option. 1af

Note that "login *free" is specifically defined to require no password. (If a "freeloader" has access to a User Telnet and has

learned of the "*free" syntax, it is fruitless to assume that he couldn't have also read the common password.) If a password must be given, acceptable responses are arbitrary text containing a line beginning either with a prompt or with "Login unsuccessful," or with "Account:". If an account is requested, the responses must be either the "Login unsuccessful" message or the text containing a prompt already described. If any errors occur during the login sequence, users are to re-try by starting from the login command. (I.e., it is not required that the Server "remember" idents or passwords.) 1ag

It is explicitly acknowledged that an acceptable response to "login *free" is "Limited access only," (followed by a prompt). This is intended to warn (human) users that the free account on the Server in question exists only to allow such functions as accepting mail and telling if a particular user happens to be logged in. (For objections to "loginless" performance of such tasks, see RFC 491. Note also that nothing here says that a Server must do anything other than return a prompt in response to "login *free" in the event that loginless operation is natural to it.) Given the UULP login discipline and the "prompt" command, it is reasonably straightforward for a program to login on a free account and perform one of these functions, for if the login command succeeded, the program will "see" a guaranteed prompt character. 1ah

To make life simpler for those Hosts which normally have some sort of "daemon" process service mail and the like, a further expansion to login is in order. The point here is that some Hosts may not know what sort of process to pass an unqualified "login *free" to, whereas they'd be sure what to do with an explicit request to process mail, do a who command, or set up console to console communications. Therefore, UULP "login" will allow a "control argument" (as discussed above) of either "-mail", "-who", or "-concom", and the respective UULP commands involved must use the respective strings in any login line they transmit. Again, nothing is being said about what a server has to do with the information, but some Servers need/want it. 1ai

Usage Information 1aj

Most Servers offer some sort of on-line documentation, from calling sequences of commands to entire users' manuals. There are two sorts of information of interest in the UULP environment: "normal" system information, and information about the particular Server's UULP implementation. To learn how to get descriptions of "native" commands, the UULP command is "help -sys" (abbreviation: "?"). Note that "-sys" is viewed as a "control argument" and as such prefaced by a hyphen ("-") to facilitate distinction from

other sorts of name (e.g., command names). To get a description of the Server's UULP implementation, "help -uulp". To get a description of a particular UULP command's implementation, "help comname". To be reminded of how to use the help command, "help". 1ak

Note: as with command names and Network-wide user names, control agruments may be either all upper-case or all lower-case. 1aki

It is specifically acknowledged that "No peculiarities." is an appropriate response to "help comname" if nothing of interest need be said about the Server's implementation of the UULP command in question. (After all, we're sparing users the necessity of studying a dozen or so users' manuals; the least they can do is to read the UULP command list.) Appropriate information for less taciturn Hosts to furnish would be such data as local command invoked (if such be the case), argument syntax (e.g., pathname description, or name of help file about pathnames), "To be implemented.", or even "Not to be implemented." 1al

"Mail" 1am

Even though a separate mail protocol is being evolved for general purposes, the UULP needs to address this topic as, by virtue of being login based, it allows systems which do access control and sender authentication on mail to make these abilities available to users within its framework of generic functions. Therefore, to read one's mailbox, the UULP command is "readmail" . To have "live" input collected and sent to a local user, "mail userident"; to a remote user, "mail userident -at hostname", where the arguments have the "obvious" meanings. To send a previously-created file, "mail -f filename userident -at hostname". Several useridents may be furnished; the delimiter is space (blank). Similar considerations apply to hostnames. If both are lists, they sould be treated pairwise. (A more elaborate syntax could be invented to deal with the desire to send to several users at a given host and then to other users at other hosts, but it seems unnecessary to do so at this point, for multiple invocations would get the job done.) 1an

The mail command prefaces the message with a line identifying the sender (Host and time desirable, but not mandatory). For "live" collection, the end of message is indicated by a line consisting of only a period (".") followed by the regnant line terminator (usually the Telnet Newline, but see also the discussion of the eol command). If remote mail is not successfully transmitted, it is to be saved in a local file and that file's name is to be output as part of the failure message. ("Queueing" for later transmission is admired, but not required.) The transmission

mechanism will follow the general mail protocol. Note that when invoked with a "-at" clause, the mail command will send "login *free -mail" to the remote Host(s), followed by a mail command with no "-at" clause. 1ao

A desirable, but not required, embellishment to "readmail" would be the accepting of a Host name ("-at hostname") to cause the local Host to go off to the named Host (via "login *free -mail") and check for mail there. Several hostnames could, of course, be specified. A further embellishment, which would probably be quite expensive, would be to accept "-all" as a request to check all Hosts (or, perhaps, all Hosts known to have a free account for the purpose) for mail. 1ap

Direct Communication 1aq

The ability to exchange messages directly with other logged in users is apparently greatly prized by many users. Therefore, despite the fact that there is a sense in which this function is not within the purview of the UULP, we will address it, after a digression. 1ar

> Digression: The UULP assumes that there can be straightforward "front ends" at the various Servers which translate generic function calls in a common spelling to calls for specific, pre-existing "native" functions. In the area of console to console communications, however, this premise does not really hold. The problem is that both major "native" implementations known to the author are seriously flawed. The TENEX "link" mechanism is both insecure (you've got no business seeing everything I type even if I'm careless enough to let you) and inconvenient (why should I be forced to remember that pesky semi-colon? how do I get back into phase after I've forgotten one?). It is also likely to be extremely difficult to simulate on systems which do not force Network I/O through local TTY buffers, even if the user interface were not subject to criticism. The Multics "send_message" mechanism, on the other hand, has a more sophisticated design, but is absurdly expensive. Therefore, the UULP mechanism to be described assumes that, for this function, new local implementations will be developed to support it. 1ari

To permit console to console communications: "concom -on"; to refuse, "concom -off". Default is off. To enter message-sending mode: concom useridentt -at hostname" ("-at" clause is optional). To exit from message-sending mode, type a line consisting of only a period (cf. Mail, above). While in message-sending mode, each line will be transmitted as a unit. The first message sent by concom must be prefaced by an identifying line, beginning "From:"

and containing an appropriate address to which to reply. The closing period-only line should be transmitted, so as to allow the other concom to close as well. Acceptable error response is "Not available: userident." (which neither confirms nor denies the existence of the particular user -- a matter of concern on the security front). The command must, of course, do whatever is necessary to transmit the messages; i.e., if locally invoked, access the local mechanism, and if invoked for remote communications, access the remote Host's concom command (via "login *free -concom"). Thus, a user at a TIP would use the local form of concom on the Host of the other party if this is convenient, or would use the remote form on his "usual" Server if the direct use is inconvenient for some reason (such as having no account there, say). 1as

The prerequisites for establishing communications are to find out if the user is logged in, and what "address" to use if so. The mechanism for gathering this information is an expanded "who" command. (Note that "who" is the UULP command to invoke the generic who's logged in function, with no constraints on format of reply.) The syntax is "who userident -at hostname", where both arguments may be multiple. If no "-at" clause, then check local Host only. Response must begin "From hostname: userident:" followed by either an appropriate address (e.g., "11" if local "concom" uses TTY numbers and userident is logged in on TTY 11), or "Not available." 1at

As with mail, a "-all" embellishment might be pleasant. Note that the search for the specified user(s) -- whether or not "-all" is used -- still assumes that a "login *free -who" login will be used on the appropriate remote Host(s), followed by "who userident". This is why responses to the expanded who command must be so rigidly specified. Note also that regardless of whether the inquiry is made in terms of Network-wide or local user name, the response must be appropriate for use in "concom". 1au

"Good" concom implementations will presumably do an expanded who command automatically, so as to spare the user the necessity of having to do it separately. Indeed, the -concom control argument to login is defined to imply the ability to do a who as well as a concom to cater to this possibility. It is tempting to legislate that such an approach be the rule, but the implementation implications are not quite clear enough to do so. The implicit who should be viewed as a strong hint to implementers, though. 1av

File Creation and Manipulation 1aw

The common command subset must furnish the ability to create and manipulate files. Creation is necessary in order to send mail on

the one hand, and to produce source files for subsequent compilation on the other hand. Manipulation (such as copying, renaming, typing out, and the like) is necessary both as a convenience aspect for users who seek to operate only in the common command language and as a means of performing desired batch functions (see below). For file manipulation commmands, the user could enter the File Transfer Protocol environment. However, the FTP user interface is constrained by a very high degree of program-drivability. It also lacks abbreviations and suffers from the lack of mnemonicity dictated by limiting command names to four characters. Further, some valuable functions (such as causing a file to be typed out) are not dealt with. Therefore, various UULP file manipulation commands are given in Appendix 1. They need not be addressed in detail here. However, some context would be useful: 1ax

The file manipulation commands assume that all Servers have some notion roughly corresponding to "the user's working directory". All file names, whether the yet to be invented Network Virtual Pathname or the "local" variety, are taken to refer to files in this directory unless otherwise indicated. That is, the user should not have to furnish "dsk:" or the like; it is taken as given that when he refers to file "x" he means "the file named 'x' in my current working directory" and the Server "knows" what that means. 1ay

At the present stage of development of the UULP, it does not seem fruitful to go into a reasoned explication of the following statement. For now, suffice it to say that those file manipulation commands (a copy of a foreign file, for example) which need to employ the FTP do employ the FTP and let it go at that. As the context and implications of the protocol become more widely understood, the detailed implementation notes will be added to the file commands -- and refined for the other commands, doubtless. In a way, the common file commands may be viewed as a kind of "User FTP" of known human interface when they deal with foreign files. (And, of course, until there's a Network virtual pathname, the issue doesn't really arise.) I expect that an "identify" command might be desirable, so that UULP commands which have to access other Servers in turn on behalf of the specific current user can have the necessary login information available to them. Such a command is included in Appendix 1, but should rank as speculation for now. 1az

On the topic of file creation, matters are rather complicated. It is clear that the ability to create files in the UULP environment is extremely desirable. It is also clear that using mail to a fake address to get the file created, then renaming the "unsent mail" file is too byzantine to expect users to do. Unfortunately, it is

not clear exactly what the alternative is. That is, it's fairly clear that we need a common editor, but it's not at all clear which editor it should be. 1b@

Two widely-known editors come to mind: TECO and QED. However, not everybody has them. Even if everybody did, the "dialects" problem is bound to be a large one. Even if all the relevant system programmers could agree, there remains the question of whether the intended user population would be willing to bother learning a language as complex as TECO or QED. Therefore an optional UULP command to be called "neted" is proposed. (See also RFC 569.) This editor is a line-oriented context editor (no "regular expressions", but also no line numbers). It is copiously documented in Chapter 4 or the Multics Programmers' Manual, including an annotated listing of the (PL/I) source code. A simple user's guide has been prepared (see Appendix 3). Several implentations already exist, and commitments have been made for more. It may also be repugnant to some of the system programmers who would be called upon to implement it -- which is why it is optional, until and unless higher authority makes it mandatory. 1ba

Other Protocols 1bb

The nominal initial impetus for proposing a UULP was to allow new Network user protocols to be invokable through a common mechanism, rather than requiring a new responding mechanism to be built for a new contact socket for each new protocol. Although this goal has been shunted into the background by the admission of the true goal of the UULP, it has not been dropped completely. Therefore, to enter the FTP Server environment, the UULP command is "ftp"; to enter the RJE Server environment, the UULP command is "rje". Exit is as per the respective protocols. (Where possible, exit should be back to the UULP environment.) 1bc

Invoking Foreign Programs 1bd

There are two broad contexts in which it is desirable to cause a specific local program to be invoked from the common command environment: The User side of the connection may itself be a program, and the desired server side program a specifically cooperating one; this is the more sophisticated context, of course. The less sophisticated context assumes that the User side is a "live" user, and the desire is to invoke a compiler or an object program the user has already compiled in the common language -- again as a convenience to the user so that he may operate in a sort of "Server-transparent" mode. (The latter case also covers "batch" use of the Server; see below.) In both contexts, the important role of the UULP is to specify the mechanisms through which the particular programs may be invoked,

irrespective of the idiosyncrasies of the Servers' command languages. 1be

Programming languages are much too big a problem to tackle here. However, assuming that a user somehow manages to create a source program, he still wants some commonality of spelling in invoking the appropriate compiler, or even the object program. As an optional but strongly recommended UULP command, then, "call name" should invoke object program name (where the named program may be a "native" command with agruments specified as appropriate). The values "-pl1", "-basic", "-fortran", "-lisp", etc., should be recognized as requesting the invocation of the appropriate language processor (to operate on a named source file or interpretively/interactively if no source file was named), with "reasonable" defaults in effect. Note that this all is meant to imply that "native" commands are not directly invokable from the UULP environment (other than by "call"), to avoid potential naming conflicts between system commands and new UULP commands. 1bf

> Note that the "call" command in the UULP environment constitues a rubric for "parallel" computation, given any ad hoc convention for the return of completion information. (Writing on the Telnet write socket plus 2 would seem appropriate, provided the initiator has the ability to "listen" for the rfc; but even a response in the data stream would do, as a special-cased program is assumed on the "user" side anyway.) 1bf1

Other Matters 1bg

The topic of "batch" mode merits some attention. As with the file manipulation commands, more consultation is necessary for a firm spec. However, I suspect that a "-batch" control argument to login should initiate batch mode processing by the Server, and given the call and identify commands all we might then require is a convention for designating the output file in order to return it via a copy command in the "job" itself (if output is to be returned rather than stored at the Server). Of course, -batch will probably need some substructure as to password and timing matters. More details will emerge in this area in future iterations. 1bh

An admittedly fictionalized scenario might look like this:

```
login Me -batch -pw xxx -shift 3
copy *452<me>source.text source.pl2
call -pl2 source
call source input output
identify Me2 yyy
copy output *555>root>Me>output452
logout
```

where user "Me" wants the Server receiving the commands (either

directly frm him at a TIP or perhaps from some other Server on which he has created a file containing them) to set up a batch job for him, with password "xxx", to be run on Shift 3 (whenever that is). The job first copies file "source.text" from directory "<me>" on Host 452 into local file "source.pl2", thn compiles it with the local PL2 compiler, executes it (assuming a "Not found" response would go into a known file if compilation had failed) with apecified arguments (presumably the names of files for input and output), then copies the "output" file to Host 555's file hierarchy at the indicated place, using the user identifier "Me2" and the password "yyy". It's not elegant, but it ought to work. 1bi

Finally, on the topic of logging out, the UULP command is "logout". The Server must close the Telnet connection after doing whatever is appropriate to effect a logout. To retain the Telnet connection, "logout -save". Having the Server close is viewed as a convenience for the user, in that it spares him the necessity of causing his User Telnet to close. It is also desirable for program-driven applications, so as not to leave the connections "dangling" and not to require possibly complex negotiations with the User side to break the connection. 1bj

APPENDIX 1. THE COMMON COMMAND SUBSET 1bj1

Syntax Opt 1bK

I. "Set-up" Commands 1bl

login id arg
The id may be Network-wide or Host-specific.
"*free" is reserved.
The arg may be "-mail", "-who", "-concom",
"-batch", or may be absent.
Result is to be either logged in or passed off to appropriate daemon. 1bm

prompt char
Specifies that char is to become or
precede the normal prompt message.
Acceptable prior to login. 1bn

erase char X
Specifies that char is the erase character.
Invocation with no argument reverts to default. 1bo

kill char X
Specifies that char is the kill character.
Invocation with no argument reverts to default. 1bp

eol char X
Specifies that char is the newline character.
Invocation with no argument reverts to default. 1bq

local
Enter the local command environment. 1br

ftp
Enter the FTP environment. 1bs

rje
Enter the RJE environment. 1bt

logout
Logout and sever the Telnet connection. 1bu

logout -save
Logout but keep the Telnet connection. 1bv

map
Apply the case-mapping conventions of Appendix 2.
Required on Hosts to which case is significant. 1bw

identify id arg X
Specifies that id is to be used as the user
identifier in any "fanout" logins required.
If arg is specified, it is to be either the
password to be used in such logins or "-pw", in
which case the Server will furnish a mask or negotiate the Hide
Your Input Telnet option; if no arg, then no password is to be
furnished on fanout logins.
Default id is "*free". 1bx

II. Communications Commands 1by

readmail
Type out "mailbox". 1bz

readmail (id) -at host X
Type out "mailbox" on remote Host host.
Multiple Hosts may be specified,
separated by spaces (blanks).
Implies ability to change working directory
at host to directory implied by Known
user identifier, or (optionally) by id. 1ca

readmail -all XX
Search for mail.
Extremely optional. 1cb

mail id
Collect input until line consisting of only a period (".") for mailing to local user specified by id. 1cb

mail -f file id
Send contents of specified file to specified local user. 1cc

mail id -at host
Collect input until line consisting of only a period (".") for mailing to remote user(s) at specified Host(s). Both id and host may be multiple, separated by spaces. (If multiple, they should be taken pairwise.) 1cd

mail -f file id -at host
Send contents of specified file to specified remote user(s). 1ce

who
The generic who's logged in command. 1cf

who id
Is id logged in? Constrained responses. 1cg

who id -at host
Is the specified user logged in at the specified host. Constrained responses. 1ch

concom -on
Enable console to console communications. 1ci

concom -off
Disable console to console communications. 1cj

concom id
Send messages to specified local user until line consisting of only a period (".") 1ck

concom id -at host
Send messages to specified remote user. 1cl

III. File Commands 1cm

type path
Type out the contents of the specified file. Pathname may be local or Network-wide. Default to current working directory. 1cn

listdir
List the contents of the current working directory. (Local format acceptable.) 1co

listdir path
List the contents of the specified directory. 1cp

rename old new
Change the specified file's name as indicated. 1cq

addname old new X
Give the specified file the specified extra name. 1cr

delete path
Get rid of the specified file.
("Expunge" if necessary.) 1cs

copy from to
Make a copy of the file specified by the first pathname at the second pathname. 1ct

link from to X
If your file system has such a concept, make a "link" between the two pathnames. If no second argument,
use same entry name in working directory. 1cu

status path st X
If your file system has such a concept, give status information about the specified file or directory. 1cv

changewd path X
If no argument, return to the "home" directory. 1cw

typewd X
Type out the pathname of the current working directory. 1cx

neted path X
See Appendix 3. 1cy

IV. Invoking "Native" Programs 1cz

call name (args) X
Invoke the specified program with the
specified arguments (if any).
The following names are reserved to indicate the invocation of the corresponding language processor: "-pl1", "-basic", "-fortran", "-lisp".
(If no source file indicated, invoke "interpretively" if possible.) 1d@

V. On-line Documentation 1da

help name
Type out information about the specified UULP command. If name is "-sys", type out information about how to use the local system's help mechanism; if
"-uulp", about the local system's UULP implementation. If no name given, describe the command itself. 1db

APPENDIX 2. MAP COMMAND CONVENTIONS 1db1

This appendix will eventually contain the case-mapping conventions detailed in RFC 411. 1dc

APPENDIX 3. EDIT COMMAND REQUESTS 1dc1

This appendix will eventually contain descriptions of the neted command requests (a draft of which now exsits), or a reference to the Resource Notebook version, if that gets published first. For now, it should be sufficient to point out that the requests are basically locate, next, top, change, save,and quit -- i.e., it's the "old-fashioned" flavor of context editor. 1dd

1de

JBP 15-NOV-74 17:39 24526

Padlipsky's Common Command Language (CCL)

(J24526) 15-NOV-74 17:39;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

CHI 15-NOV-74 17:59 24527

Sent via sndmsg to warshall and millstein with CC to balzer, crocker, and carlson.

A scenario of an NSW session 1

(J24527) 15-NOV-74 17:59;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /NPG([INFO-ONLY]) RWW([INFO-ONLY]) ;
Sub-Collections: SRI-ARC NPG; Clerk: CHI;

JMB 15-NOV-74 18:15 24528

Bugs in Edit Statement command and Syntax (through Syntax generator)

Bug priority/effects: (1) Stumps potential users of Edit Statement command (how much is this command used?) (2) Slight inconvenience to documenters

JMB 15-NOV-74 18:15 24528

Bugs in Edit Statement command and Syntax (through Syntax generator)

When user asks for syntax of Edit Statement command or of Base subsystem (through Syntax command--<CTRL-s>--or Syntax generator subsystem), It comes out: 1

Edit Statement (at) DESTINATION 1a

It should indicate something ike this: 2

Edit Statement (at) DESTINATION EDITSTRING OK 2a

If one tries to follow what is printed and types an address & <CR> for DESTINATION, nothing happens then. NLS just waits--no prompt, nothing. Isn't this also a bug (or a least a design flaw) in the command itself, i.e., no prompt of any kind when a field is expected? Not only that, but, typing a question mark doesn't work at that point; ? just echoes. User is really stuck. 3

This makes this command useless in the sense you can't find out how to use it through <ctrl-s> or questionmark or by reading a prompt. 4

You can find the appropriate information from Help, but it's cryptic since <CTRL-q> after giving command-word Edit does not take you directly to the explanation of the command--you have to take the see-also [Kirk can you look into this?]. 4a

JMB 15-NOV-74 18:15 24528

Bugs in Edit Statement command and Syntax (through Syntax generator)

(J24528) 15-NOV-74 18:15;;;;, Title: Author(s): Jeanne M. Beck/JMB;
Distribution: /BUGS([ACTION]) FDBK([ACTION]) KIRK([ACTION])
; Sub-Collections: SRI-ARC BUGS; Clerk: JMB;

KIRK 17-NOV-74 00:42 24529

Regarding bugs in Edit Statement command

I think the reason for the neglect of the proper help mechanisms for the Edit statement command is largely due from it's inconsistency with the rest of NLS, from it's lack of users, and from a desire to discourage it's use in favor of the other NLS commands. I agree that this probably results in more general confusion for the user and the bugs you mentioned should be fixed.

As far as help is concerned... If you were a new user, and you were trying to find out how to use NLS as a text editor and you typed the word "edit", would you want to be taken to the Edit command, a general description of how to modify information in NLS, or a choice between the two? It seemed to me, such a user would want a general description about modifying information so that's where I took him with a note makeing it a bit harder to get to the Edit Statement command. Your note did not change my feelings about this, but you are quite welcome to go in and fix it however you wish. 1

KIRK 17-NOV-74 00:42 24529

Regarding bugs in Edit Statement command

(J24529) 17-NOV-74 00:42;;;; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /JMB([INFO-ONLY]) FDBK([INFO-ONLY]) BUGS([
INFO-ONLY]) ; Sub-Collections: SRI-ARC BUGS; Clerk: KIRK;

KIRK 17-NOV-74 01:52 24530

Good and Bad NLS practice reflected in your proposed sendail citation

Robert, I've been thinking about your proposed new sendmail envelope format and it really looks good to me. It really is a close, but NLS-wise efficient match to the tried and true standard format used by most all bibliographers. I think you have done a good job. There is one thing I don't quite understand, that is the reason for placeing carriage returns and three spaces in front of some parameters instead of just having them down a level with the other parameters. 1

I see an unnecessary source of confusion here with level indenting (the ability to turn on statement numbers not with standing). I also see a problem here, not shown in your example but sure to occur, where parameters overflow onto a second line which, of course, cannot be indented and therefore will look ugly and defeat the purpose of the cosmetic indentation in the first place. This is a standard NLS problem which is none-the-less necessary to allow many possible views of a statement and easily move it around to different levels of structure. This is also the reason it is generally bad NLS practice to format by carriage returns and spaces what NLS can already do for you automatically. People who do this are generally always disappointed the first time they move their carefully formatted paragraph down a level, or delete a word from a line. Unless there is good reason for doing this that I am missing, I would not like to see such bad practice encouraged by using it in something as out-front as the default sendmail envelope will be. 2

A one-line, all levels, view of the sendmail envelope with the parameters you currently have appended to the citation placed instead in the substructure would also contain more information at no expense. It would look essentially the same when printed out with a full view. The Message userprogram (for which I am responsible) could be easily made to do the same thing. It would seem to be an improvement all around. 3

P.S. Shouldn't your proposed "REFERENCES" parameter have the link in angle-brackets as well? 4

KIRK 17-NOV-74 01:52 24530

Good and Bad NLS practice reflected in your proposed sendail citation

(J24530) 17-NOV-74 01:52;;;; Title: Author(s): Kirk E. Kelley/KIRK; Distribution: /RLL([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: KIRK;

KIRK 17-NOV-74 04:16 24531

test to see if this gets sent to appropriate person

yes, this is just a test 1

KIRK 17-NOV-74 04:16 24531

test to see if this gets sent to appropriate person

(J24531) 17-NOV-74 04:16;;;; Title: Author(s): Kirk E. Kelley/KIRK;
Distribution: /WUC([ACTION]) KIRK([INFO-ONLY]) ;
Sub-Collections: SRI-ARC WUC; Clerk: KIRK;

Another test, you know

(J24532) 17-NOV-74 05:35;;;; Title: Author(s): ARC FDBK Feedback/FDBK; Distribution: /WUC([INFO-ONLY]) ; Sub-Collections: SRI-ARC WUC; Clerk: KIRK;

A Scenario of an NSW Session

INTRODUCTION 1

The following is a rough draft of a detailed scenario of a user's session with the NSW. It is being used as a vehicle to describe the flow of control and data structures in the NSW system. 1a

A user's session with NSW 2

LOGIN: 2a

A user walks up to an apparently unused terminal and wants to use it to access the NSW. If it is a dial-up terminal, then he must dial the appropriate number and await the carrier. Let's assume that the Frontend responds differently to <CONTROL-G>, based upon whether the terminal is in active use or not. Thus the user types <CONTROL-G>. If the terminal is indeed available the Frontend responds with login questions. If, however, the Frontend thinks the terminal is active, then it simply rings the bell on the terminal. If the user is accessing the Frontend from a dial-up line, the Frontend must ask the user what type of terminal it is so that it can properly pad for the terminal. 2a1

If the terminal is free, the user is asked for a project name, user name (I dislike "programmer" since many users will not be programmers), and password (not echoed). The Frontend will PCP-call the Works Manager (henceforth denoted as WM -- the Frontend PCP-created an instance of the WM when it initially started up) and pass it the info collected from the user. The WM (as with any tool) may return in "help" mode, requesting that the Frontend (the caller) provide it with new data for a particular parameter. In this case, the Frontend will attempt to get correct info from the user and try again (by returning new parameters to the help call). [There are issues of time-out here. What does the Frontend do if the WM goes into an infinite loop? Perhaps the data associated with an open package includes upper bounds on real time required for each routine to complete execution (perhaps this should be "multiplied" by the "load factor" for the machine on which it is executing).] 2a2

The WM looks up PROJECT in its tables (presumably by hashing the string) and, if it finds one by that name, checks to see if USERNAME is a valid name to use with that project. If the USERNAME is in the list, then its PASSWORD is checked against the one the user typed. If all checks are ok, the WM returns a user-id to the Frontend and makes entries in its table(s) of active users. In addition, lets assume that it returns a PCP data structure called the user-profile (or

part of it, actually, since part of it is WM specific), which presumably lives in a file whose name is known to the WM via table entries or is derived from the project-user pair (it is assumed that the WM also provides a primitive which takes a user-id and returns non-secret portions of the user-profile). 2a2a

If any of the checks fail, the WM issues a PCP-HELP to the caller (Frontend), specifying which parameter is bad. The Frontend either returns with a new parameter or with instructions to abort the checking. 2a2b

As discussed above, when the Frontend initially started up, it "created" the WM and retrieved the Frontend and WM grammars from it. Subsequently, whenever the Frontend's interaction with the WM is on behalf of a particular user, that user's user-id will be passed as a parameter of the call. 2a3

Once the user is logged in, he may give commands to the Frontend or to the WM. The commands to the Frontend and WM are always available to the user. [Unless we adopt an escape character approach.] 2a4

SHOW FILE LIST: 2b

The user may decide he would like to see a list of his private files. This is a command to the WM. The WM primitive for doing this would probably take a partially specified name and generate a list of all private (public) files that began with that name. In addition, it should take the list of attributes about the files that the user wants to see (e.g. last write date/time, size of file, type of file, etc.). There are two ways that the list might actually be presented to the user. 2b1

First, the WM might call the show-status primitive in the Frontend, passing it the whole list or a part of the list (where successive calls would append to the already presented list). The approach of appending to the list has the advantage that the user starts seeing the list right away (if you will excuse the overstatement), rather than having to wait while the whole list is generated. This first approach is somewhat of a bad design, however, since the callee assumes who the caller is, rendering this primitive unuseable by other tools. 2b1a

The second approach would be for the WM to return the whole list or PCP-COROUTINE-RETURN parts of the list. [Although this latter approach implies that a SHOWSTATTUS primitive must be added to the CML, it has the advantage that the

callee does not assume that the caller is the Frontend and consequently is a better design. This also has the advantage of perhaps allowing the command language writer the possibility of specifying fixed parts of the status message in a template in the CML and filling in the blanks from the info obtained from the PCP-call to the execution function (this could be done via a PCP-call to the Frontend also, but raises issues such as how to identify the template to use). However, given the potential squeeze in PDP-11 address space, I would prefer that these templates not be used initially.] In any event, if the status message is presented piecemeal, the list of appendages must be closed off by a special entry so that the Frontend can interact with the user to take the status message down (if at a display) and let him continue with other things. 2b1b

RUNNING A TOOL: 2c

Now, lets suppose the user wants to run a tool. This is a command to the WM. The list of tools available to this user is contained in this user's user-profile (set up by project leader). The Frontend PCP-calls the WM to obtain a toolid for this tool and to allow the WM to verify that the user may run the tool and to record this event. If necessary, the Frontend PCP-calls the WM to get the grammar for this tool-id (it may already be in the Frontend because some other user used this tool). The Frontend extracts from the grammar the list of tool backends needed for this tool and PCP-calls the WM to create each backend and associate it with this tool. The WM returns the process handle for each such backend. 2c1

The WM selects a Tool-Bearing Host (TBH) on which to run the backend [using what criteria?] and PCP-creates the backend on that host, allowing it to use a particular work space on that host. [Hopefully, backends being used with the same grammar (tool) will reside on the same host and use the same file space if possible.] (This might actually be done by starting a TBH EXEC and having it create the backend as its subprocess.) The WM interacts with the backend's PCP component and the Frontend's PCP component to give them direct PCP access to each other. (An alternative model would have the WM pass back the process-name to the frontend and it would create the process and connect it to the WM -- assuming they all have to be connected together, that is.) IF necessary, the Frontend can PCP-CALL the WM to get the name of the NSW file containing the help data base for the tool. 2c1a

The Frontend makes the grammar for the tool the current grammar

for this user. The grammars for the WM, Frontend, and current tool are "unioned" together, allowing the user to give commands to any of the three [unless an escape character approach is used]. Commands in such a grammar, in general, specify certian "command words" that the user uses to specify commands or parts thereof, noise words to help the user understand the semantics of the commands, and requests that the Frontend collect certain arguments for the command. The commands may invoke (through PCP) execution funtions which will in general implement the semantics of the command. The command language designer must specify the simple names (which the WM translates into specific PCP process names) for the backends and packages that the execution functions reside in, since procedure name uniqueness cannot be guaranteed across backend boundaries. [Either a declaration statement or a complete-name syntax will be added to CML to accomodate this.] 2c2

SWITCHING FROM TOOL TO TOOL: 2d

At any point the user could give a command to run another tool. The tool the user was running is "detatched" and prevented from outputing to the user's terminal but permitted to run until it attempts to do so [What do we do about timeout here?]. If the tool he wished to run is currently detached, the Frontend simply makes the grammar for that tool the current grammar and allows any output from that tool to proceed. If the tool he wishes to run is a new tool, things proceed as described above. 2d1

STOPPING A TOOL: 2e

If the user gives the stop tool command, the Frontend PCP-calls the WM to stop the tool backends (if the user is actually using the tool). The WM collects accounting info from the TBH Execs or the tool backends, deletes the backend processes associated with this tool, the WM and Frontend throw away the toolid and process handles for the tool, and the Frontend decrements the usecount on the grammar. If the usecount is zero, the space occupied by the grammar may be freed if needed for new grammars. 2e1

FREEZING AND RESUMING A TOOL: 2f

The user may decide to freeze a tool (stop it temporarily), do something else for a while, and then resume the execution of the tool. Both the freeze and resume commands result in PCP-calls to the WM, which in turn PCP-calls the appropriate TBH Execs to freeze/resume the tool backends. When the tool is frozen, the Frontend removes its grammar from current-grammar

status. When it is resumed, its grammar is again made the current grammar. 2f1

ASKING FOR SEMANTIC HELP: 2g

If the user requests semantic help with a tool the Frontend automatically starts the help tool (which is probably loaded as needed rather than at Frontend startup time) and passes it info on the user's parse state, the name of the help data base for this tool, the name of the NSW help data base (containing overall concepts), and the user-id so it can get at the user-profile. The user may interact with the help tool for a while and then resume using the original tool. If he requests help again for the same tool, he merely switches to the help tool which receives new parsestate info but otherwise preserves the state from the last interaction with this user. 2g1

CHANGING A USER PROFILE: 2h

If the user runs the tool which allows him to manipulate parts of his user-profile (or if he is project leader, the list of available tools for his project personnel), this tool manipulates both the copy of the profile in the Frontend [if user is changing his own user-profile, only,] (so the results are immediate) and the actual user-profile file. [what happens if the same user logs in twice from two terminals simultaneously? does the WM prevent this?] 2h1

LOGOUT: 2i

If the user gives the stop session command, the Frontend PCP-calls the WM, which stops any tool backends the user has active, collects acounting info from the TBH Execs or the tool backends, deletes the tool backend processes, moves the user's files to a safe place (if necessary) and removes the user-related entries from its tables and returns accounting info to the Frontend. The Frontend presents this acounting info to the user and frees the space occupied by his user profile and any other user-related data structures it might have. It then waits for another <CONTROL-G> from that terminal. 2i1

If a dialup line hangs up, it is the same as a stop session command, except that the accounting info is not presented to the user. 2i1a

CHI 17-NOV-74 18:24 24534

A Scenario of an NSW Session

(J24534) 17-NOV-74 18:24;;;; Title: Author(s): Charles H. Irby/CHI;
Distribution: /NPG([INFO-ONLY]) WEC([INFO-ONLY]) RWW([
INFO-ONLY]) ; Sub-Collections: SRI-ARC NPG; Clerk: CHI;
Origin: < NSW-SOURCES, SCENARIO.NLS;2, >, 15-NOV-74 17:39 CHI
;;;;####;

xxx

(J24535) 18-NOV-74 07:54;;;; Title: Author(s): James H. Bair/JHB;
Distribution: /JHB([ACTION]) ; Sub-Collections: SRI-ARC; Clerk:
JHB;

RLL 18-NOV-74 10:38 24536

A test of the unrecord feature.

A test a test a test. Does this have a number? Is this marked private? We shall find out. 1

RLL 18-NOV-74 10:38 24536

A test of the unrecord feature.

(J24536) 18-NOV-74 10:38;;;; Title: Author(s): Robert N. Lieberman/RLL; Distribution: /JDH([ACTION]) JHB([INFO-ONLY]) ; Sub-Collections: SRI-ARC; Clerk: RLL;