Prompt Comprehension in UNIX Command Production

by

Stephanie Doane, Danielle S. McNamara,
Walter Kintsch, Peter Polson,
Deborah Clawson and R. Gary Dungca

Department of Psychology
University of Colorado
Boulder, CO 80309

ICS Tech Report 91-04

Prompt Comprehension in UNIX Command Production

This research applies cognitive theory to an applied problem in human-computer interaction. Modern computer-based systems make numerous cognitive demands on users. The sheer volume of information presented is often overwhelming, and this can be confounded by obscure human-computer interface designs. Detailed analyses of interface designs in the context of cognitive theories have proven fruitful. For example, Norman (1981) shed light on the trouble that users have producing UNIX commands because they are not mnemonic, and as such are difficult to recall. Researchers espousing theoretical approaches to human-computer interaction recognize that there are multiple determinants to user problems with complex systems, and that a theoretical approach is required for these to be disentangled (e.g., Card, Moran, & Newell, 1983). Theory based approaches seem the most profitable for providing causal analyses of the real difficulties users have with complex systems, and have the best chance for generalizing to other system interface designs. In addition, applied domains are fruitful environments for the investigation of fundamental issues in cognition, and applied problems pose unique challenges for theories of cognition to address (e.g., Polson and Lewis, 1990).

The goal of this study is to provide empirical support for assumptions about user knowledge of UNIX and user memory processes which are proposed by the Doane, Kintsch, & Polson (1989;1990) UNIX comprehension construction-integration model (UNICOM) for generating complex commands in UNIX. In so doing, we specify the role of this theory in understanding user action planning, and we illustrate a general methodology that may be used to examine complex knowledge and process assumptions made by other cognitive models.

In UNICOM, we model the understanding of task instructions and the generation of appropriate action plans. van Dijk and Kintsch (1983) and Kintsch (1988) make the point that comprehension of text that describes a problem to be

file", then the file names would be printed on the line printer. The second method requires fewer keystrokes, and as such is more streamlined. The use of pipes "|" allows commands to be combined to produce the command "ls|lpr". In this example, the output from the "ls" command flows directly into the "lpr" command. Thus, one can think of a pipe as a plumbing pipe, where information flows directly through the pipes.

We wish to determine what kinds of knowledge are necessary to use these advanced features (e.g., input/output redirection). Knowing the specific action (e.g., knowing the command "ls" lists file names) is critical to plan execution, but simply knowing the elements is not sufficient to develop an effective action plan.

Previous empirical work involved providing UNIX users at varied levels of expertise textual descriptions to produce legal UNIX commands requiring the redirection of standard input and output (i.e., "composite" commands) (Doane, Pellegrino, & Klatzky, 1990). Their findings suggest that novices and intermediates have knowledge of the elements of the system; that is, they can successfully produce the single and multiple commands that make up a composite. They can not, however, put these elements together using pipes and/or other redirection symbols to produce the composite commands. The symbols that enable input/output redirection are fundamental design features of UNIX, and these features are taught in elementary computer science courses, but Doane et al. (1990) demonstrate that these features can only be used reliably after extensive experience (e.g., experts had, on the average, 5 years of experience with UNIX).

UNICOM (Doane, Kintsch, & Polson, 1989;1990), was developed to determine why the Doane et al. (1990) users had such difficulties generating composite commands. To facilitate discussion of the UNICOM knowledge base, we will use the command "nroff -ms ATT2>ATT1" as an example. This command formats the contents of the file ATT2 using the utility nroff and the -ms macro package, and

produce a successful command. For example, to produce the command that will print the first ten alphabetically arranged lines of a file on the line printer, "sort file | head | lpr", the model has to first select the items to be used (sort, | , | ,lpr,head, file) and then determine the order of the items. The model must also keep track of which command is first, second, and third, and what the output of the first command is, where its output is redirected, and so on for the remaining command elements. Doane et al. (1989;1990) assumed that the mechanisms involved in integrating and ordering the items to produce a composite would result in memory load problems. In addition, it was assumed that experts possess knowledge structures which decrease this memory load, whereas novices do not.

Item knowledge and ordering processes are involved not only in the composition of UNIX commands but are also important for understanding problem solving and skill acquisition in general. The literature on expertise and skill acquisition (e.g., Chi, Feltovich, P. J., & Glaser, R.,1981; Larkin, 1983) suggests that less expert individuals may know the separate facts in a domain, but are not able to use them productively in the sense described by Wertheimer (1982/1945). Understanding how individuals utilize textual instructions and experience to transform their factual knowledge into effective problem solving procedures is an important general issue in skill acquisition.

We also note that we are analyzing the knowledge and processes required to chain elements together in the context of a theory of comprehension that has been used to explain algebra story problem comprehension (Kintsch, 1988), and solution of simple computing tasks (Mannes & Kintsch, in press). Furthermore, the theory of comprehension used here was originally developed in the context of story comprehension (Kintsch, 1988) and has been widely applied in the context of narration and descriptive texts. Thus, rather than developing an idiosyncratic knowledge analysis, we are performing our research in the context of a general

different amounts of the required four types of knowledge and that displaying the prompts that help with each respective type of knowledge will impact subsequent user performance. We hold similar assumptions about the process knowledge (i.e., keeping track of intermediate results).

Method

Subjects

Twenty-two computer science and electrical engineering majors with between six months and six years experience with UNIX were paid $20 for their participation. A preexperimental questionnaire was administered to determine the users' experience with UNIX and other operating systems, their history of computer training, and their self-assessment of their computer skills. Following the method outlined in Doane, Pellegrino, & Klatzky (1990), subjects were classified into expertise groups. Novices (N=10) had less than 1.25 years of experience with UNIX and no operating systems courses; intermediates (N=8) had between 1.25 and 3.0 years experience with UNIX and some had operating systems courses; and experts (N=4) had greater than three years experience with UNIX and all had taken an operating systems course.

Apparatus and Materials

All production tasks were performed on a Macintosh II with a large-screen monitor. The experiment was controlled by a program written in Supercard[2]. The stimuli were task statements, a fixed directory of file names and a series of help prompts, all displayed on the screen, as well as three "error cards" presented by the experimenter. Subjects responded by typing at the keyboard a command or series of commands to accomplish a given task and then used the mouse to "click" on a displayed button the cause the program to evaluate their answer.

syntax knowledge) required for the problem. Prompts 5 and 7 remind the user of the complete set of items that have already been identified in previous prompts. We assume that the latter prompts will ease memory load for items. Prompt 6 is the first prompt that determines the order of commands and symbols for the user (providing command redirection knowledge and help with tracking intermediate results). This information is repeated in Prompt 8. Finally, Prompt 9 gives the user the correct production. The prompts were displayed in order of severity of the proposed cause of error; for example, the prompt telling the exact command names was always presented earlier than the prompt explaining the concept of redirection. Although the problems varied in the number of prompts that accompanied them depending on the length of the target command sequence, all prompt series followed the same general structure and are categorized into nine ordered prompts.

Finally, we used error cards, that described the type of error made by the subject on an incorrect attempt. The three types were: "Illegal," "Different: Legal but does something other than what the problem asks for," and "Keystrokes: Legal and does what the problem asks for but uses more than the minimum possible keystrokes." (This last error category was intended to force subjects to use composite commands rather than a series of commands with temporary files in order to be correct.)

Procedure

Subjects were run individually for a single two-hour session. The experiment was made up of three sections: a questionnaire, an orientation, and completion of the actual UNIX problems.

Orientation. To ensure that the subjects were familiar with the MAC II and the mouse, they read the experiment's instructions on the screen, using the mouse to click on button images in order to proceed from one page of directions to the next. To familiarize them with the specific procedures used in the composite command

response. When the subject typed the required command sequence, the program displayed a "CORRECT!" screen and then revealed the next composite task instructions.

Results and Discussion

Scoring Correct Command Productions

Productions were scored as correct by the MAC computer (using a Super Card program) if they matched the syntax of the idealized command (spaces were not parsed). Thus, a subject had to produce the command that required the least number of keystrokes (i.e., subjects could not substitute "sort>file1; head file1>file2" for the command sort file1 | head>file2").

Partitioning Productions into Groups

The data were analyzed by grouping all of the problems together, which allowed a comparison of all relevant effects of expertise level and prompt. However, this approach would have ignored the existence of an important learning curve. The learning curve is influenced by two aspects of the problems; their serial position, and the percentage of new knowledge required for solution. A fact in a problem was considered new knowledge if it had not been used previously by the subject in the experiment. A partial correlation predicting percent correct production performance on the basis of the two variables was performed to determine their unique contributions. Serial position was not a significant unique predictor of percent correct performance (r = .008), whereas percent new knowledge was (r=.493). Consequently, we will examine performance as a function of new knowledge. We split the problems into three groups based on arbitrary but sensible criterion, those requiring 0 % new knowledge (8 problems), those requiring 1-59% new knowledge (10 problems), and those requiring 60-100 % new knowledge (3 problems).

ordering information in Prompt 6. To analyze this effect we study the four types of component knowledge demonstrated in each attempt in detail.

Scoring of Knowledge

Each of the problems given to subjects required a certain amount of the four types of component knowledge. For example, the problem described in Table 1 requires two command syntax facts: nroff -ms as a command name, and that nroff takes a file argument. It requires knowledge of one I/O redirection syntax fact -- that ">" redirects output from a command to a file. The conceptual I/O knowledge required is that redirection of input and output can occur, and that I/O redirection can occur from commands to files. The required command redirection knowledge includes the fact that nroff output can be redirected to a file. Answers for all tasks were scored for the percentage of each type of knowledge displayed by a subject at each prompt level. For example, if a command requires two command syntax facts, and the attempt at prompt 0 (before any prompts) shows evidence of only one of these facts, then the subjects is credited for having 50% of the required command syntax knowledge for that task. When a production was correct, all knowledge types were at 100% levels.

Insert Figure 2 here

Knowledge Analyses

The 60-100% new knowledge problems again provide the most interesting data. The other two problem groups show similar, but attenuated expertise effects, and thus will not be shown. Figures 2 (a) and 2 (d) show the mean knowledge scores for the three expertise groups at prompts 0-9 for the 60-100% new problems. The arrow markers signal the reader what prompt first provided subjects with

problem requires may be more or less difficult to use. This seems to be a function of how new the knowledge is as evidenced by a knowledge by new interaction, $F(6,114) = 11.10$, $MS_e = .008$, $p < .01$. Furthermore, prompts have a differential effect on performance as a function of the percentage of new knowledge a problem requires, $F(18,342) = 28.86$, $MS_e = .014$, $p < .01$. Prompts are less influential on performance in 0% new knowledge problems than other problems because all groups have learned from previous prompts.

The amount of new knowledge interacts with both the prompts and expertise, $F(36,342) = 4.91$, $MS_e = .014$, $p < .01$, and with the prompts and the type of component knowledge required, $F(54, 1026) = 4.78$, $MS_e = .002$, $p < .01$. It appears that the experts are less influenced by the percentage of new knowledge required, perhaps because overall this knowledge is less new to them than to the less expert groups. In a sense, they are not being tutored in this experiment to the extent that the less expert subjects are.

To determine the specific effects of expertise, these data were further analyzed by planned comparisons. Novices and intermediates possess significantly different knowledge scores than do experts. $F(1,19)=5.83$, $MS_e = .295$, $p <.03$. The experts possess unique amounts of knowledge across all of the tasks. The influence of prompts on the four knowledge types is not equivalent for experts and the less expert groups $F(3,54) = 7.35$, $MS_e = .022$, $p <.01$. Novices and intermediates do not show significantly different knowledge scores across problems, $F < 4$.

The arrow markers on Figures 2(a-d) show some of the most interesting interactions between prompts and knowledge scores, especially when compared with the percent correct performance shown in Figure 1(a). Figure 2(a) suggests that for all groups, presentation of Prompt 2, which helps with command syntax (see Table 1 for an example), improves command syntax knowledge and percent correct performance (see Figure 1(a)). However, the change in percent correct performance

different changes in performance as a function of exposure to different types of prompts, and intermediates fall somewhere between the other two groups.

It appears that novices and intermediates need help with I/O syntax, but that this is not a problem for the experts. Experts get help from an abstract statement of I/O conceptual knowledge (see Table 1 for an example), while this statement is less worthwhile to the novice and intermediate groups. This finding is in line with the expertise literature (e.g., Chi et al., 1981) which suggests that experts can process more abstract information than can novices. Certainly Prompt 3 is more abstract in the sense of providing less specific information than do Prompts 2 and 4. Experts also successfully utilize a reminder prompt that lists all of the command and I/O elements together in one prompt. Novices and intermediates don't seem to utilize this information. Experts that have not solved a problem correctly by this point have often represented the problem with one command reversed or dropped; this prompt reminds them that they have forgotten an element, and they can use this information to make a correct production. The less expert subjects seem to have more problems at this stage than just knowing the elements, so that they cannot put them together given just a list of the items. Their response to the ordering prompts (see Prompts 6 & 8 in Figure 1 (a)) make this clear. These are the first prompts that tell users what order they must use for the commands. For example, that first, nroff -ms must be performed on ATT2, and then, that this output must be redirected to the file ATT1, and so on. Although the experts have solved all ordering problems before this information is presented, this ordering information helps the less expert groups quite a bit,.

Scoring Productions for Working Memory Errors

So far, we have provided evidence suggesting that the less expert groups need assistance with ordering. We shall now consider evidence that the less experts need this information because they have a working memory load problem. The work of

prompt as within-subjects variables. These data show significant effects of expertise level (deletion $F(2, 19) = 4.24$, $MS_e = .204$, $p < .03$; substitution $F(2,19) = 6.23$, $MS_e = .427$, $p < .01$). Novices commit the greatest number of errors, relative to the expert and intermediate groups, as supported by planned comparisons (deletion $F(1,19) = 8.04$, $MS_e = .02$, $p < .01$; substitution $F(1,19) = 11.04$, $MS_e = .043$, $p < .02$). There is also a main effect of prompt (deletion $F(9,171) = 15.68$, $MS_e = .094$, $p < .01$; substitution $F(9,171) = 17.61$, $MS_e = 104$, $p < .01$). The deletion errors decrease radically for all groups after the first command syntax prompt (Prompt 2), and substitution errors decrease after the first ordering prompt. It is difficult to explain the rise in substitution errors for the less expert groups following Prompt 3. Recall that this prompt presents the abstract information about I/O concepts (see Table 1.). It is possible that this general statement confuses the less expert groups, and they try new commands and orderings in response. Also note the lack of a drop in substitution errors by the novices at Prompt 5, consistent with our earlier hypothesis that they find little use for the listing of elements. Their memory load problems are not solved by a listing of the elements in a command. If this were the case, Prompt 5 would have an impact on their deletion and substitution errors and on their percent correct performance. Instead they require specific ordering information to decrease their memory load errors and improve their correct performance. Finally, the percentage of new knowledge required influences the number of errors (deletion $F(2,38) = 13.17$, $MS_e = .293$, $p < .01$; substitution $F(2,38) = 38.09$, $MS_e = .35$, $p < .01$). Errors are highest for the 60-100% new knowledge problems. For substitutions, the effect of new knowledge is stronger for the less expert groups than for the novices $F(1,23) = 9.07$, $MS_e = .035$, $p < .01$.

In summary, there is evidence that the less expert groups are making more errors that are consistent with a memory load hypothesis than are the experts. And the extent of these errors decreases radically for all groups when they are helped

novices and intermediates lack both specific syntax knowledge and more general conceptual redirection knowledge. If prompted with specific syntax knowledge, novices and intermediates can increase their chances of producing a correct composite. However, novices and intermediates cannot utilize our more abstract prompt (Prompt 3) which provides the conceptual redirection information knowledge that they need (see Figure 2(b)) to increase their correct performance (see Figure 1(a)). Our experts, in contrast, can use this information effectively to increase their performance. This finding is consistent with the literature on expertise which suggests that experts can utilize abstract information, while novices cannot (e.g., Chi et. al, 1981).

Novices and intermediates need help retrieving the elements that go into making a composite; but for many of them, this is not enough. They also need help ordering the elements -- or constructing a command. When they receive a prompt that helps them order the items, their performance improves. There is evidence that ordering the elements taxes their working memory. They delete and substitute many more components of a composite than do experts, and these memory errors decrease markedly once ordering information has been given. In contrast, experts attain almost perfect performance before any ordering information is given.

Thus, the hypotheses suggested by the UNICOM construction-integration model were supported. There appear to be separate components of knowledge above and beyond simple command knowledge required to produce composites. Further, retrieving the elements is only part of the problem in composing a composite. To chain these elements together is another aspect of the problem, and this requires relating the pieces of knowledge in an ordered fashion and tracking the intermediate results.

Suggesting design solutions. Can our theoretical analysis contribute to system design? It is interesting to note that none of the new graphical user interfaces for

Current theory development. Applied research can also drive a theory to expand, and we have extended the construction-integration theory as a result of this effort. First, we have extended the theory to another, more complex domain. The theory has previously been applied to text comprehension and to mathematical problem solving (Kintsch, et al., 1990; Cummins, et al., 1988), and now it has been applied to the production of novel, complex action planning in UNIX.

Second, to build the UNICOM construction-integration model, we had to extend the theory to include causal planning mechanisms that are found in traditional planning systems (see also Mannes & Kintsch, in press). For text comprehension data (e.g., Kintsch, 1988), the associative structure found in the initial conception of the construction-integration theory is sufficient. For the UNIX work, the causal chaining between plan elements representing components of a composite were required.

Future theoretical work. The theory must expand further to remain fruitful in our current efforts to account for differences in knowledge as a function of expertise. Kintsch (e.g.,1988) has proposed that the overlap that exists between a network of propositions created by a text and the corresponding network of the reader determines what propositions will be remembered and processed. This notion can be extended in the UNIX work to suggest that the overlap between a network of propositions resulting from an incoming instruction or prompt and that of the user will dictate their recall of syntax and concepts relevant to the task, and to what they will attend. For example a prompt may not sufficiently overlap with novices' knowledge to dictate their attention or processing of that material. This would explain why, for example, the abstract prompts are not of much use to the novices. This explanation is consistent with previous work on the construction-integration theory, but it must be studied further to be supported.

much in common with the command "head file | wc", which counts the number of words in the first ten lines of a file. However, structurally, these commands are similar in that they both use a pipe in the middle, and require the output of the first command being redirected as input to the second command. While linking these together on the basis of structural aspects is simple computationally, we would like the links to fall naturally out of the cognitive architecture as a function of user knowledge, and for the theory to predict what links of this type will exist. For example, experts may see this structural similarity because of their knowledge of piped commands, whereas novices may not.

Thus, we have made contributions to the theory by researching a complex applied problem, and to the applied problem our theoretical analyses have proposed solutions. We are optimistic about the continued contributions resulting from the interaction between this theory and our applied work in the future.

Mannes, S. M. & Kintsch, W. (in press). Routine computing tasks; Planning as understanding. Cognitive Science.

Murdock, B. B. (1974). Human memory: Theory and data. Lawrence Erlbaum Associates.

Norman, D. A. (1981, November). The trouble with UNIX. Datamation, 139-150.

Newell, A. (1987). Unified theories of cognition. The 1987 William James Lectures.

Polson, P. G. & Lewis, C. H. (1990). Theory-based design for easily learned interfaces. Human-Computer Interaction, 5, 191-220.

van Dijk, T. A. & Kintsch, W. (1983). Strategies of discourse comprehension. New York: Academic Press.

Wertheimer, M. (1982/1945). Productive thinking. Chicago, IL: University of Chicago Press.

Footnotes

[1] UNIX is a registered trademark of AT&T.

[2] Supercard is a trademark of Silicon Beach Software.

[3] The component knowledge shown is higher than the percent correct scores shown in Figure 1. This is because a subject attempt can show high, but not perfect component knowledge, and component knowledge must be perfect for an attempt to be entered as correct in Figure 1.

Table 2.

Production answers for complete problem set.

edit JOB1<STAT2 | lpr

nroff -ms APON | sort | head>ANAL.C

sort PROP1 | tail>PROP2

ls | sort | head

edit OLD1<STAT2 | sort | head>PROP1

ls | head>EMPT2

edit OLD3<OLDDT

sort STAT1 | nroff -ms

cmp CHORE CHORY | sort

cat OLD1 OLD2 | lpr

nroff -ms APON | head | sort>ATT1

cmp BOLD1 BOLD2 | wc -c>EMPT1

cat BOLDER1 BOLDER2>>BOLD1 &

nroff -ms ATT2> ATT1

sort WRITE | head | wc -c

edit JOB1.P<STAT2 | sort>PROP2

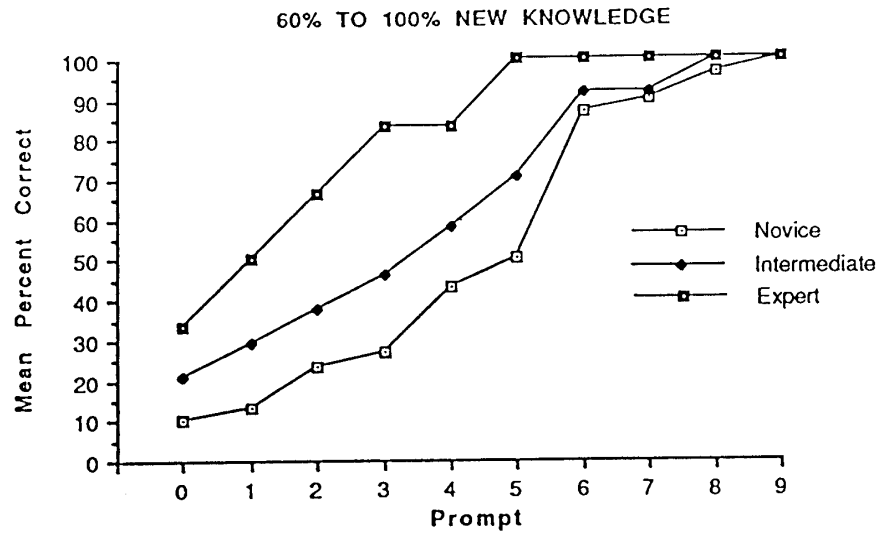
ls | wc

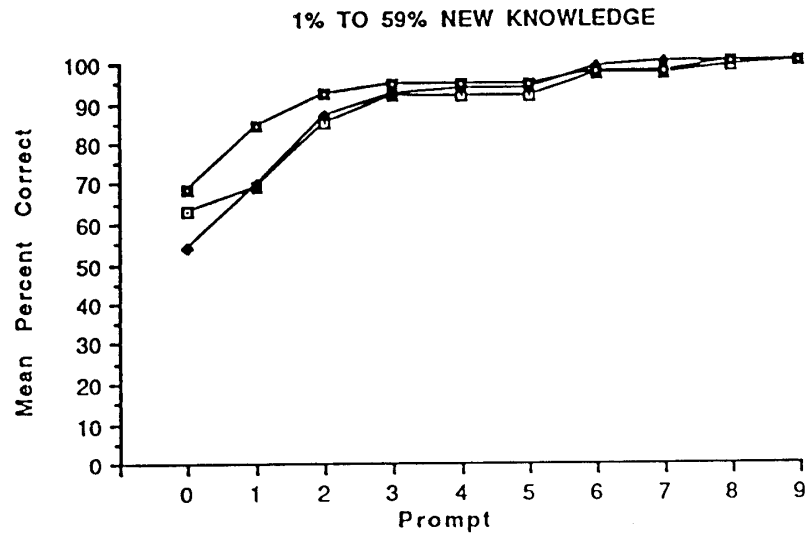
cmp REJECT1 REJECT2 | sort | lpr

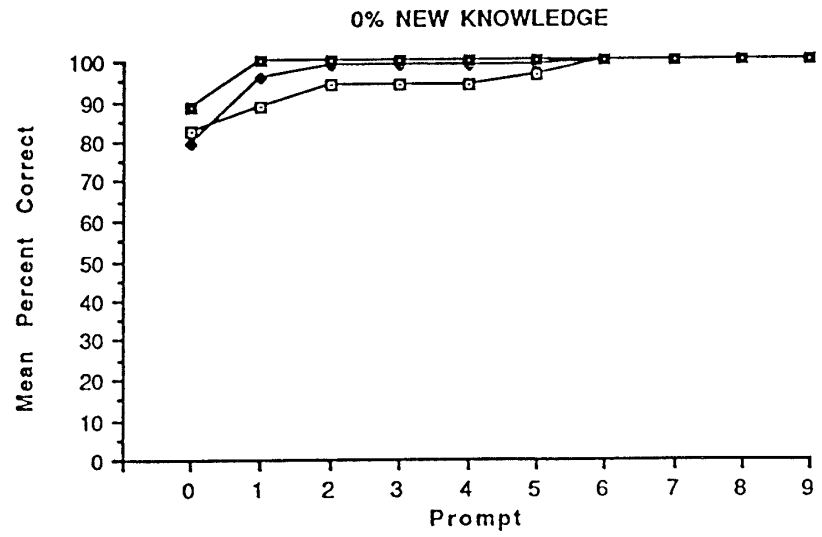
edit JOB1.P<STAT1 | nroff -ms>JOB.P

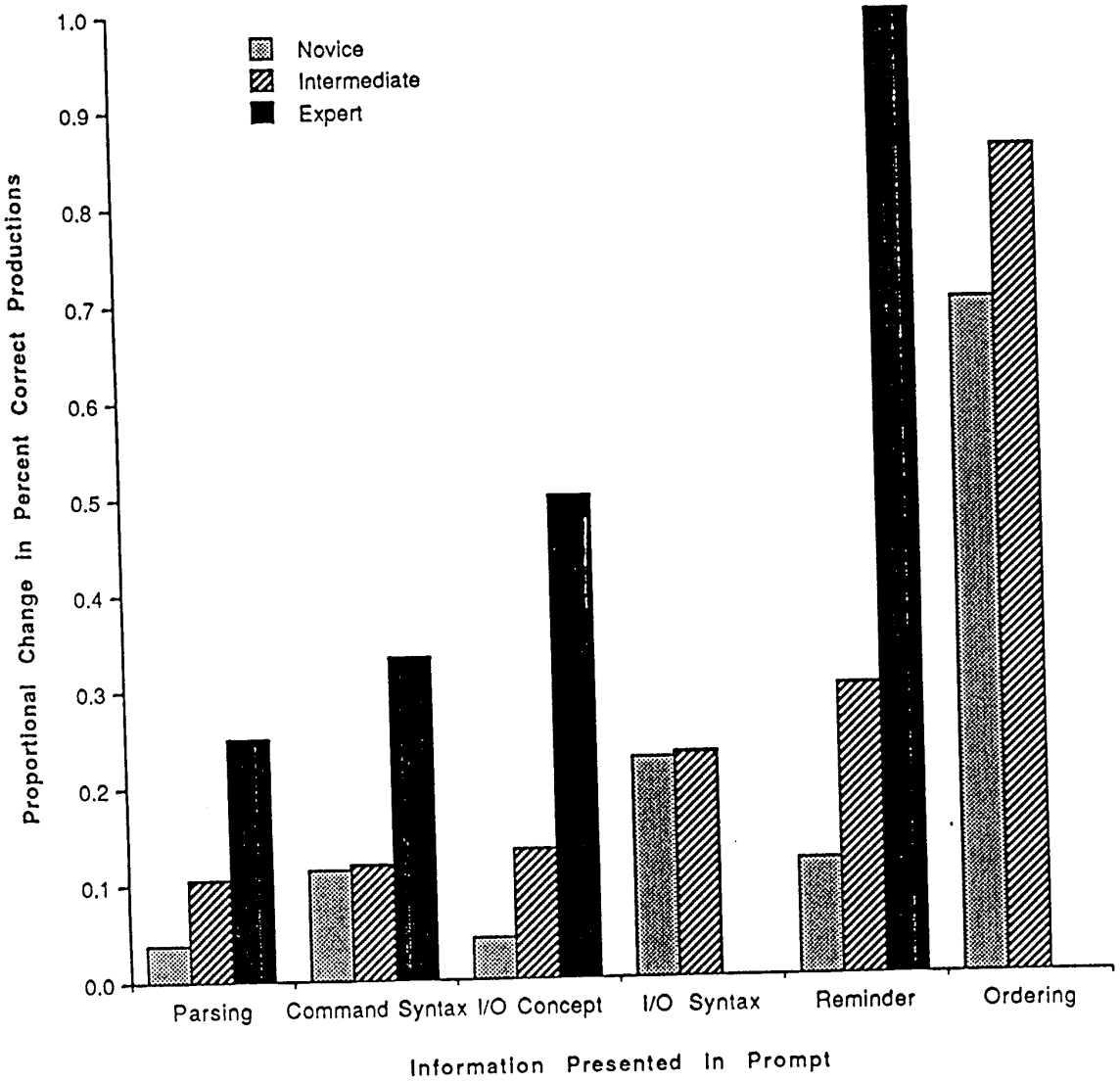
ls | EMPT1

nroff -ms OLD3 | sort>OLDDT


60% TO 100% NEW KNOWLEDGE
Mean Percent Correct
100
90
80
70
60
50
40
30
20
10
0
0 1 2 3 4 5 6 7 8 9
Prompt
Novice
Intermediate
Expert


1% TO 59% NEW KNOWLEDGE
Mean Percent Correct
100
90
80
70
60
50
40
30
20
10
0
0 1 2 3 4 5 6 7 8 9
Prompt


0% NEW KNOWLEDGE
Mean Percent Correct
100
90
80
70
60
50
40
30
20
10
0
0 1 2 3 4 5 6 7 8 9
Prompt


Novice
Intermediate
Expert
Proportional Change in Percent Correct Productions
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Parsing
Command Syntax
I/O Concept
I/O Syntax
Reminder
Ordering
Information Presented In Prompt