

Thesis  
1849.

## Helping UNIX Users:

An assessment of the effectiveness of various forms of  
online help

Lynne Margaret Coventry

Department of Computing Science and Mathematics

University of Stirling



Submitted in partial fulfilment of  
the degree of Doctor of Philosophy

June 1991

**THE BRITISH LIBRARY DOCUMENT SUPPLY CENTRE**

# **BRITISH THESES NOTICE**

The quality of this reproduction is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print, especially if the original pages were poorly produced or if the university sent us an inferior copy.

Previously copyrighted materials (journal articles, published texts, etc.) are not filmed.

Reproduction of this thesis, other than as permitted under the United Kingdom Copyright Designs and Patents Act 1988, or under specific agreement with the copyright holder, is prohibited.

**THIS THESIS HAS BEEN MICROFILMED EXACTLY AS RECEIVED**

**THE BRITISH LIBRARY  
DOCUMENT SUPPLY CENTRE  
Boston Spa, Wetherby  
West Yorkshire, LS23 7BQ  
United Kingdom**

### Abstract

The aim of online help is to make complex computer systems more usable and allow users to exploit more of the system's power. To achieve this aim it is necessary to provide users with information needed to accomplish their current task while also encouraging further skill development to facilitate the transition from novice to expert.

This thesis investigates the relationship of individual differences to the use of computers and online help. An observational study of real users of UNIX showed that very few commands were used by users and there was great variability in the use of UNIX. "Field Dependency" was identified as a potential source of the variation between users. Two experiments were carried out to assess the effects of Field Dependency. The subjects were required to carry out a number of tasks with help provided via a human expert or an online help system. The help system developed could be configured to behave actively or passively. Two different user communities, computer science students and women trainees, were studied.

Both experiments found Field Dependency to be correlated with the number of commands known by users: the more field-independent a user, the more commands are known. In the first experiment it was found that field-dependents were exposed to more help from the human expert than the field-independents. With the help system, the field-independents were exposed to more help. Field-independents were also found to benefit from increased flexibility of the system where both active and passive help was available whereas field-dependents did not.

Conclusions are drawn about the effects of Field Dependency on user interaction with help systems and the effectiveness of two alternative access initiatives.

## **Dedication**

**To my daughter, Rowan Elizabeth, whose entry into this world in the middle of my PhD was untimely, but then the unexpected events usually turn out to be the best. I wish for her the opportunity to be everything she wants to be, in a peaceful and green world.**

## Acknowledgements

There are many people to acknowledge for their help in the completion of this thesis, my thanks goes to them all, even if I forget to mention them be name.

I would like to thank the Human Computer Interaction group at Wang Laboratories in America, especially Carrie Rudman, for their encouragement and much welcome working breaks in Boston and Colorado. My thanks also goes to Wang Stirling who funded me for three years, Dr. Chris Morse and Prof. Peter Henderson for setting up the fellowship.

The Computing Science department for putting up with me through 10 years of degree, MSc and PhD work. I know the department will be a quieter place without me. Specific thanks goes to the Technical Support Group in the guise of Graham and Sam for rescuing me from slight technical hitches, encouraging me to live it up too much and for Sam's never ending comments on work (and everything else) and excellent grammar. Thanks should also go to my supervisor, Jim Cowie, for supervising me and having faith in my ability to finish, and Richard Bland for explaining statistics to me and convincing me they are a necessary evil. Lastly and by no means least, the secretaries for providing me with female support in this male cabal.

I would like to thank my husband for allowing me time to do this work.

Most importantly I would like to thank my Coffee Bar buddies, my life would not have been worth living if it wasn't for their wit, humour, and compliments and rockbuns every morning. Also my office-mate, Steve, who was the only person capable of sharing an office with me for three years and still smiling!

Lastly, I would like to acknowledge myself. If it was not for my bloody-minded determination to be an academic, inspite of all obstacles put in my way, then this thesis may not have been completed and I may have been employed as a bar person or shop assistant. Then again, who is to say that I will not still be!

## Copyright acknowledgements

The following use of trademarks is acknowledged:

MSDOS: Microsoft Inc;  
UNIX, ASSIST: AT&T Bell Laboratories.

## Declaration

The work presented in this thesis is the composed by the author and embodies work carried out by the author and not included in any other thesis. However, it should be noted that part of this work has been published (Coventry, 1989, 1990).

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	The problem of usability	1
1.2	Solutions to the usability problem	3
1.3	Online help systems	4
1.3.1	What is an online help system?	4
1.3.2	Why are help systems needed?	5
1.3.3	Alternative designs for help systems	8
1.4	User diversity within computing	8
1.5	A multidisciplinary approach	9
1.6	Problem statement	10
1.7	Structure of the thesis	10
<b>2</b>	<b>Users and methods of study</b>	<b>13</b>
2.1	Individual differences	13
2.1.1	The size of individual differences	14
2.1.2	User characteristics contributing to performance differences	15
2.2	Cognitive style	19
2.2.1	Similarities between theories	20
2.3	The Field Dependency dimension	26
2.3.1	Field-dependence	21
2.3.2	Field-independence	21
2.4	Field Dependency: Ability, strategy or style?	22
2.5	Why use Field Dependency?	24
2.6	Field Dependency and cognitive models	25
2.7	Field Dependency and Human-Computer Interaction	26
2.7.1	Potential system biases	30
2.8	Field Dependency and online help	32
2.8.1	Minimising variation in user performance	33
2.9	Methods for studying UNIX users	34
2.9.1	Natural studies	34
2.9.2	Controlled laboratory studies	35
2.9.3	Theoretical models	36
2.10	Analysing the activities of computer users	38
2.10.1	The "Wizard-of-Oz" technique	38
2.10.2	Traces of user activities	39
2.11	Statistical evaluation of data	40
2.11.1	The quest for statistical significance	41
2.11.2	Correlation	41
2.12	Summary	42

<b>3</b>	<b>Online Help Systems</b>	<b>43</b>
3.1	Supporting the user	43
3.2	Design issues for online help	44
3.2.1	Access Initiative	44
3.2.2	Access mechanisms	45
3.2.3	Presentation issues	48
3.3	Online help research	49
3.3.1	Online versus written documentation	51
3.3.2	General improvements	51
3.3.3	Context sensitive help	52
3.3.4	Amount of help required	53
3.3.5	Adaptive help	54
3.3.6	Active help	55
3.3.7	Type of information	56
3.3.8	Natural language	57
3.4	Summary of research findings	57
3.5	Intelligent help	58
3.6	Standard UNIX help facilities	59
3.6.1	The man command	59
3.6.2	The help command	61
3.7	"Intelligent" approaches to the UNIX help problem	62
3.7.1	AQUA	62
3.7.2	The SINDX consultant (SC)	62
3.7.3	The UNIX Consultant (UC) Project	62
3.8	An alternative approach	63
3.8.1	SUPERMAN II	63
3.9	Conclusions	64
<b>4</b>	<b>UNIX</b>	<b>66</b>
4.1	The UNIX system structure	66
4.1.1	The user's perspective	67
4.2	Learning and UNIX	69
4.3	Suitability of UNIX for interface studies	70
4.4	Studying UNIX users	72
4.4.1	The subjects	72
4.4.2	The procedure	73
4.4.3	The results	73
4.4.4	Observations	75
4.5	Discussion	81
4.6	Conclusions	82
<b>5</b>	<b>Pilot study</b>	<b>83</b>
5.1	Measurement of Field Dependency	83
5.2	The first help experiment	84
5.2.1	Aims	84
5.2.2	Experimental task	85
5.2.3	Providing help	85
5.2.4	Method	86
5.2.5	Results	88
5.3	General discussion	91
5.3.1	Inefficient interaction	91

5.3.2	Ratings of expertise	92
5.3.3	Group differences	92
5.3.4	Consequences of Field Dependency	93
5.3.5	Experimental problems	94
5.4	Conclusions	95
<b>6</b>	<b>Design of the Help System</b>	<b>97</b>
6.1	Implementation considerations	97
6.1.1	The prototyping life cycle model	98
6.2	Design aim	99
6.3	Overview	100
6.4	Requirements analysis	102
6.4.1	Help system requirements	103
6.4.2	Requirements validation	105
6.5	Choosing commands for the help system	105
6.6	Categorising commands	106
6.7	The user interface	107
6.7.1	Inputs required	107
6.7.2	Output layout	108
6.8	Components of the System	108
6.8.1	Program overview	108
6.8.2	The help scripts	110
6.8.3	Help initiatives	111
6.9	Summary	113
<b>7</b>	<b>Experimental Evaluation</b>	<b>119</b>
7.1	Aims	119
7.2	Method	120
7.2.1	The subjects	120
7.2.2	Experimental groups	120
7.2.3	Experimental design	121
7.2.4	The experimental task	122
7.2.5	The system	123
7.2.6	The procedure	123
7.3	The results	125
7.3.1	User statistics	125
7.3.2	Correlations	125
7.3.3	One-way analysis of variance	128
7.3.4	Interaction effects	130
7.3.5	Qualitative results	132
7.4	Discussion	133
7.4.1	Impact of Field Dependency	133
7.4.2	Impact of interface type	134
7.4.3	Impact of interaction effects	134
7.4.4	Measuring performance variables	135
7.4.5	Experimental problems	136
7.5	System enhancements	138
7.6	Design recommendations	138
7.7	Conclusions	138



<b>8 Summary and conclusions</b>	<b>140</b>
8.1 What has been learned? . . . . .	140
8.2 Main Limitations . . . . .	145
8.3 Looking to the future . . . . .	145
8.4 Interdisciplinary Challenges . . . . .	147
<b>Bibliography</b>	<b>150</b>
A User log extract	160
B Task instructions	170
C Help scripts	173
D Evaluation statistics	193

# List of Tables

2.1	Styles A and B from Entwistle (1981)	20
2.2	Field Dependency and HCI	27
3.1	Different methods of providing online help	50
3.2	Summary of Borenstein's results (1985)	52
4.1	Data from Command logs	74
4.2	Summary of command logs	74
4.3	The 20 most commonly used commands	77
4.4	Size of command sets	78
4.5	Percentage of overlap between individual command sets	80
5.1	Summary of scores	89
7.1	Summary statistics for dependent variables	125
7.2	Table of Pearson Correlation Coefficients	126
7.3	Summary of time taken (secs) to complete the EFT	126
D.1	Subject's raw scores	194
D.2	F value and significance level for Analysis of variance	195

# List of Figures

1.1	Different domains of knowledge	7
4.1	Frequency of Command Usage	76
6.1	The prototyping life cycle (Somerville, 1985)	100
6.2	Overview of the system	101
6.3	Menu structure	114
6.4	The work screen	115
6.5	The command list screen	116
6.6	Passive system screen	117
6.7	Active system screen	118

# Chapter 1

## Introduction

### 1.1 The problem of usability

The growth in the power of computer technology makes it possible to supply more and more sophisticated human-computer interfaces, while decreasing costs have resulted in a proliferation of computer systems which has, in turn, led to a larger and more heterogeneous population of computer users than there has been previously. The life cycle cost of major systems has shifted away from the hardware to the personnel associated with the system development, maintenance and use.

The computer is a demanding tool. Unlike most other technologies it has no single purpose, and because it can be used in so many ways, it can be badly misused. In the past, computer systems such as UNIX were used by computer scientists who were familiar with the terminology and tolerant of poor interfaces. With wider use of computers has come new classes of users, including the computer-illiterate, or naive user, who is perhaps a professional in some other capacity. Businesses, hospitals and defence installations among others are becoming ever more dependent on interactive computer systems. Higher expectations are now being placed on the usability of the system hardware, software and interface. Today's computer users expect to be able to use a computer system after little or no training and without specialised technical expertise, while the computer system and the tasks to be performed may be highly complex. It has become important to develop systems that are not only reliable but also offer interfaces which promote correct use of the systems, are easy to use and resistant to errors.

People often experience serious and frustrating difficulties when they attempt to use computers in their workplaces (Carroll & Campbell, 1989). The tasks which formerly they could accomplish easily must be re-learned: competent secretaries and accountants are, at least for a while, reduced to varying levels of incompetence until they can master the system. Such difficulties continue to obstruct users at all levels of experience. This is a problem of system usability.

Usability from the user's perspective, can be defined as "the extent to which a user can exploit the potential utility of a system". This represents a distinction between functionality and usability. Functionality dictates whether or not a system can carry out the tasks it is required to; whereas usability reflects the extent to which a user can put that functionality to use.

In an insightful article, orientated towards time-sharing systems but applicable to the broader class of interactive systems, Nickerson (1981) considers the question why "interactive systems are sometimes not used by people who might benefit from them". He suggests a number of reasons. These include:

**Lack of training and user aids**

Users are given little useful help in learning the system, and little help when they get into trouble.

**Poor documentation**

Both tutorial and reference documentation is unclear, inaccurate, incomplete, poorly organised and/or out of date.

**Obscure command languages**

The system dialogue is unnatural, difficult to learn and/or difficult to use.

**Inconsistency**

The system has a variety of subsystems with conflicting conventions and command languages.

**Inadequate user conceptualisation of the system**

It is difficult for the user to form a correct and useful cognitive model of the function and structure of the system.

Designers of computer systems face many problems. One problem is the nature of the human partner in the interaction. People cannot be described in precise, mathematical terms, although

there have been some attempts to do so by creating models of users and deriving equations to predict performance. Users have highly individual characteristics and preferences; there is no such thing as a 'typical' user. Consequently, it is difficult to design an interface which suits all users.

Another problem is that human-computer interaction covers such a broad spectrum of issues. It includes aspects such as screen format and dialogue structure. A large range of technology is also available: keyboards, mice, touch screens, gesture interfaces, voice interfaces and so on. There are also different styles of interaction, such as command-driven, menu-driven or forms-based. The interface may be under user control, system control or mixed initiative (Thimbleby 1980).

Currently there is inadequate help available for designers of computer systems faced with a wide range of decisions. They may not fully appreciate the effects their decisions may have on users. One approach is to establish guidelines for design: for instance an Apple publication, "Human Interface Guidelines : The Apple Desktop Interface" (Apple Computer Inc, 1987). However, this has not yet yielded a consensus on detailed, useful, yet manageable guidelines for software designers. If anything, the sheer number, bulk and inconsistencies of the guidelines means that they may be more of a hindrance than a help.

The current lack of a theoretical base means that software systems are generally best developed experimentally; in other words, by prototyping the users' interactions with the system (Norman, 1984). From both the designer's and the users' perspectives this is beneficial as it allows them to experience the interface before extensive time, effort and money has been invested in development. It is easier to see any potential problems through "hands on" experience of a dynamic interface than by reading a static specification. Improvements may then be incorporated in the final design.

## 1.2 Solutions to the usability problem

One possible solution to the problem of usability is to build "intelligence" into the interface. One of the keys to providing intelligence at the interface is through the use of a user model. This involves the system constructing a representation of the user's knowledge, skills and behaviour patterns in such a way that it can respond according to each particular user.

Attempts to embed intelligence in interfaces are meeting with limited success, and it is most often applied in one of two specific domains. The first is intelligent help systems (Chin, 1986;

Croft, 1984; Mason, 1986; Quinn & Russell, 1986; Risaland, 1984; Zissos & Witten, 1985). The second is intelligent tutoring systems.

There is no agreement on the desirability or feasibility of building significant intelligence into interfaces. Robertson (1985) stresses the need for a cognitive psychology of human information processing strategies and styles. It is also necessary to carry out in-depth studies of advisory and request protocols to both human advisors and simulated intelligent systems. Greenberg & Witten (1985), Trevelyan & Browne (1987) and Nathan (1990) note that there are disadvantages as well as advantages to providing adaptive and thus changing interfaces — they may seem to be inconsistent to the users and may cause the users extra effort in adapting to a changing system.

In spite of some reservations about the viability of intelligent systems, research is still proceeding. There are many user characteristics that have been considered necessary to be adapted to. The problems facing such systems include the diversity of users, deciding on the most important user characteristics and how a system should adapt to them.

### **1.3 Online help systems**

This thesis will explore the potential of online help systems as a solution to the usability dilemma.

#### **1.3.1 What is an online help system ?**

An online help system is one or more programs designed to provide users with assistance while they are carrying out computer-based tasks. They can either be completely integrated with the software of the task or separate, running concurrently with the program. There are two fundamental aspects of online help systems: the interface and the content. The interface includes displaying and accessing the help information. The content is the information the help system actually provides the users with. Both aspects are equally important. An online help system will fail to be of assistance if either a poor interface design makes it difficult to use or the information is unhelpful, unclear or inaccurate.

Online help is related to all other forms of documentation provided for the system. The development of the help system is usually the responsibility of the developers of the computer system, but it does not usually rank as their most popular job.

Developers are not the appropriate people to develop the help system. They are often too sophisticated in their field to communicate effectively with relatively naive end-users. They often receive little training for the extensive written communication which the preparation of user-help demands. This is reflected in the usability of existing help systems.

This thesis focuses on the user interface to online help systems. Related topics that will not be discussed in great detail include implementation issues, the task domain and the relation of online help to other areas of the system interface.

### 1.3.2 Why are help systems needed?

People often find it difficult to use the computer facilities that are intended to help them complete their tasks quickly and efficiently (Schneider & Thomas 1983). Users are often reluctant to make use of all the facilities available to them (Nickerson 1981) and those that are used are often used inefficiently (Jerrams-Smith 1987). This suggests that lack of knowledge of the system, and also of system concepts, on the part of the user wastes the user's own time and effort. Therefore, there are good reasons for investigating how to improve interfaces and support for users so that their knowledge and use of the system can be improved.

It is not possible to build computer programs that are so well designed and tested that no user assistance is ever needed. Although such claims are being made, systems designed with such intentions are proving not to live up to their sales claims (Jones, 1990). It is not always possible to anticipate what potential users will do or in what ways they will put a program to use.

Help systems can be thought of as a "safety net" (Kearsley, 1988) that assist users when they do not find the system as "self-explanatory" or "natural" as they were led to believe it would be. A help system may be used to pursue two different goals; the first is short term and concerns facilitating the user in completion of the current task. The second is long-term and concerns enabling the user to increase their knowledge and improve future interactions.



### Reality is complicated

*"Useful computers which are not usable are of little help; but so are usable computers which are not useful"* (Fischer 1987). Useful computers are generally complex systems with a rich functionality, such as UNIX machines. The aim of a help system is to make complex systems usable and thus allow users to be able to exploit the system's potential power. UNIX has more than 700 commands and many more embedded in systems, eg. commands within the vi or mailx environment. Computer hardware running under such an operating system requires a substantial amount of documentation, for example, the Sun workstation is provided with approximately 15 manuals consisting of approximately 5000 pages and an additional beginners guide of around 10 manuals.

The increased functionality of modern computer systems, required as a result of the many different tasks that users want to do, leads to an increase in the complexity of the system. In such richly functional systems, a user is likely to find something that will do what they need, but it may be very difficult to find something specific.

Users of complex systems are faced with a number of problems which prevent them from fully exploiting the potential of such powerful systems. These include:

- not knowing of the existence of some commands;
- not knowing how to use some commands;
- not knowing when to use some commands;
- not understanding the results produced by some commands;
- not being able to combine commands to meet specific needs.

A consequence of these problems is that only a small proportion of the system is used with any frequency (Draper, 1984; Hanson et al. , 1984) and the functionality provided is, to a large extent, wasted. Sometimes what a user can do with a system, what users think they can do with a system, and what the system can actually do, do not coincide. This can be represented as four domains of knowledge. These domains are shown in Figure 1.1 (Fischer, 1985).

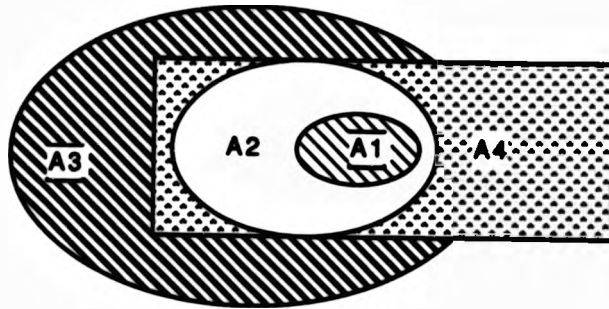


Figure 1.1: Different domains of knowledge

**A1** represents the subset of commands which the user knows and uses regularly.

**A2** represents the subset of commands which the user only uses occasionally and is not too sure about the details.

**A3** represents the user's cognitive model of the system (Norman, 1983; Fischer *et al.*, 1984), ie. the set of actions which the user thinks the computer is capable of carrying out.

**A4** represents the actual system, the commands available to the user. A subset of A4 is not contained in any of the other areas. This represents the subset of commands, the existence of which is not known to the user.

The size of these areas varies between users. An online help system is intended to gradually increase the size of A1 as required by the user, and to gradually match A3 to A4 so that the users' cognitive model of the system corresponds with the actual system functionality. The amount and type of knowledge required by users of UNIX will depend on the tasks they are required to complete — no one user will probably never know, or need to know, all UNIX commands. This requires increasing the user's knowledge of the system while eliminating interference from models of other systems the users may be familiar with, and any other analogies they may be using.

### 1.3.3 Alternative designs for help systems

In designing an effective help system there are many alternatives to consider and trade-offs to be made. These include the following decisions:

- static or adaptive — whether to provide different users with the same or different information;
- context-sensitive or context-free — whether to make inferences as to what the user is trying to do or not;
- passive or active — whether help is provided on request from the user or given, unasked by the system.

Other major design decisions include deciding what information help messages should provide, how help should be accessed, how information should be displayed. Any hardware or software constraints imposed by the computer system used should also be considered.

## 1.4 User diversity within computing

One of the reasons it is so difficult to develop a program that works flawlessly for all users is that of individual differences. Users vary along many dimensions. It is impossible to predict all possible variations of user characteristics. As a consequence of such factors, each user reacts differently to a program and it is impossible to predict what any particular individual's reactions will be.

The incredible diversity of human abilities, backgrounds, personality traits, work styles and motivations is an immense challenge for designers of computer systems. When this is multiplied by the possible range of situations, tasks, technologies and frequency of use, the resultant set of possibilities is enormous. A right-handed male with a degree in computer science who spends a large percentage of his day learning about computers may find it hard to design a system for a left-handed female arts graduate. Understanding the physical, intellectual and personality differences among users is vital, as this will reflect in the user's motivation to learn and become knowledgeable about the system (MacLean *et al.*, 1990). There are also cultural and language difficulties.

"Know the user" was the first principle in Hansen's (1971) list of user engineering principles. It sounds simple, but is in fact a difficult principle to achieve and is often undervalued. All design

should begin with an understanding of intended users. There may be more than one potential user community, so the problem is multiplied. The process of knowing the user does not end, because there is so much to learn and using the system changes the user.

There are several dimensions along which users have been classified. These include:

- Gender
- Experience and expertise
- Age
- Psychological aptitudes
- Personality
- Ability

This thesis is particularly interested in a personality dimension referred to as "Cognitive style". One of the researched theories of this dimension is that of "Field Dependency". A person's level of Field Dependency affects the way information is structured and processed by a person. This may in turn have a profound affect on the way a computer system is learnt and used. Field Dependency is thought to remain relatively static over time and not influenced by environmental factors. A 'field-dependent' approach is associated with a holistic approach and a general acceptance of, or reliance on, the inherent organisation of information. A 'field-independent' approach is associated with a more analytical approach with information being reorganised according to situational demands. The differences in levels of Field Dependency may have bearing on the type of help information required, control of the help system, and the availability and visibility of the help system.

## **1.5 A multidisciplinary approach**

This thesis combines a Software Engineering approach with psychological evaluations to produce and evaluate a user-orientated prototype help system. It focuses on the user interface to the help system. The help system was designed for the domain of command-based operating systems, specifically UNIX. A prototype life cycle model is followed and the thesis can be divided into three

main sections. Chapters 2-5 constitute the system analysis. The size of this section emphasises its importance in User-Centered Design. This section looks at the users, alternative systems and experimentally establishes the user requirements. Chapter 6 discusses the design and development of the prototype, and Chapter 7 describes an experimental evaluation of the prototype.

## **1.6 Problem statement**

This thesis explores the use of online help facilities to enhance the usability of the UNIX operating system. The aim of the thesis is to increase awareness and understanding among experts in computing science and psychology of the procedures, pitfalls and benefits of the other discipline.

This thesis addresses four major problems:

- The need to assess the amount and type of help information required by different users of operating systems;
- The lack of knowledge of how users actually behave when interacting with online help, and even less on their preferences;
- The need to identify specific features of online help provision which affect the interaction;
- The validity of Field Dependency as a measure of individual differences and its effects on the interaction. Field Dependency is being used as a user characteristic to base adaptation on and even to assess an individuals likelihood of success in the computing industry, yet the results of work carried out in this area are inconsistent.

Observations of people using UNIX are used to establish the necessary content of the help system, while empirical studies are used to investigate the effects of different methods of delivering this information to users.

## **1.7 Structure of the thesis**

### **Chapter 1. Introduction**

provides an overview of the problem of usability and introduces help systems as a solution to the usability problem. It also further outlines the purpose and organisation of this thesis.

**Chapter 2. Users and methods of study**

discusses the importance of individual differences, the origins of the differences and the potential effects on different aspects of the interaction. Field Dependency is identified as a possible source of variation which has specific affects on the interaction process. Methods of investigating users are also discussed.

**Chapter 3. Help Systems**

discusses alternative ways to design help systems and it also examines a number of existing systems. Previous research in the area is also reviewed.

**Chapter 4. UNIX**

discusses the use of UNIX as an experimental vehicle. It presents a discussion and summary of UNIX and the interaction pattern of users. Some previous relevant UNIX studies are discussed. An observational study looked at natural usage of UNIX to establish normal usage patterns of computer science students who would form one of the subject groups.

**Chapter 5. Pilot Study**

presents a study carried out to establish the amount and the type of help required by users exhibiting different levels of Field Dependency. The results of the observational study and this experiment assist in the formation of a set of help facilities which will be incorporated into a help system.

**Chapter 6. Help System Design**

discusses the reasons for the design of the help systems and describes their development. The help systems incorporate sufficient information to carry out the task rather than a complete system description. Two types of information were supplied, both 'how-it-works' information to facilitate long-term knowledge and the build up of a cognitive model and 'how-to-do-it' information to allow the user to continue with their current task. Both systems will provide the same information. The first system is a user-driven menu hierarchy. The interaction with this system is dependent on the user, who must request help and then search for the required information. The second system is system-driven (ie. active help). In this case, the system decides when the user appears to require help and what information to provide.

**Chapter 7. Experimental Evaluation**

provides empirical evidence as to the relative effectiveness of the two alternative help schemes and the relationship to users with different levels of Field Dependency. The system is evaluated using two very different groups of potential users. Reasons for the outcome of the experiments are discussed as are future refinements to the help systems.

**Chapter 8. Conclusions**

draws the ideas of the thesis together. It discusses the results of the research and the implications for future research and development. Conclusions are drawn as to the appropriateness of different help schemes and the importance of Field Dependency to the interaction. The original contributions made by this work and future directions highlighted by this study are described. In ending, the future for multidisciplinary work in the design, development and evaluation of systems is outlined.

## Chapter 2

# Users and methods of study

Before the computing aspects, such as design and prototyping can commence, it is necessary to understand the problem in its entirety. This requires a user-orientated analysis to be carried out. The first stage of which is to find out about the potential users, associated problems and how they affect different aspects of interacting with computer systems.

### 2.1 Individual differences

Although the literature on individual differences is vast, relatively few studies have been conducted which explore the effects of individual differences on performance with computer systems. The most profound problem facing the designer wishing to improve the usability of any system is that of individual differences. Users vary in gender, intelligence, training, culture, background, aptitudes, cognitive styles, learning styles, personality traits, age and in expertise. They are naive or expert, casual or regular, alienated or motivated, passive or active, each to varying degree. Any such characterisation depends on the task, the current context and the user's prior experience and motivation. There are no universal solutions that will make all users happy all of the time. Many characteristics have been considered in predicting differences in computer skills. There are two points to consider; firstly the size of the effect of individual differences and secondly the characteristics of the user that can be used to predict the differences in performance.

Many systems evolve without much consideration for any user, let alone the range of capabilities of different possible users. However, Egan (1988) quotes three reasons why attention should



focus on the differences among users:

- individual differences play a major role in determining whether a user can perform effectively using a computer.
- personnel selection is eliminating some individuals from the computing environment without sufficient evidence on which to base the decisions and without successfully eliminating the individual differences in performance.
- understanding and technology has reached a point where more differences can be accommodated.

If individual differences were simply small and random, they would hold few implications for usability. However, if some variation in user performance can be predicted and understood it would be an important step towards the goal of designing systems which accommodate a wider range of users, which in turn would enhance system usability. It is worth noting that Card, Moran, and Newell (1983) in their study of the cognitive psychology of text editing found significant differences across individuals according to their level of experience, but these differences were not nearly as great as the differences due to the design of the text editor. Thus, performance differences due to the software overshadowed performance differences due to experience. However, researchers in the field of individual differences believe that the individual differences outweigh any system differences (for instance, Elkerton & Williges, 1984; Egan, 1988). This difference in opinion is currently unresolved and will be investigated in this thesis. To accomplish this, it is necessary to establish which individual differences are important and whether an appropriately designed system can accommodate such differences. The difficulty from the perspective of designing help systems is how to detect and accommodate important characteristics of the user.

### **2.1.1 The size of individual differences**

It is necessary to establish the size of the effects that are to be considered, if the size is small in comparison to the effects possible by manipulating the system then it may not be important to investigate the potential of individual differences. This is clearly not the case, Egan (1988) summarises the potential effect of individual differences on performance as follows:

*After a group of 30 people take a programming course, one student might take a year to program what another could do in two weeks.*

The effects of individual differences have been measured in a number of different areas. In text editing, Egan & Gomez (1985) measured the performance of subjects using a line or screen editor and found that variations in performance due to individual differences, i.e. age and spatial memory was twenty times greater than variations due to the different editor designs. In information search, Elkerton & Williges (1984) found subjects' previous experience on other systems accounted for more variance in performance than variables such as window size, type of target and length of file. In training, comparing user differences to differences in training procedures shows the same result. The variation due to training types is only a fraction of the variation due to individual differences (Egan, 1988).

### **2.1.2 User characteristics contributing to performance differences**

There are many dimensions along which users may differ. It is important to establish which ones, if any, are important for system design.

#### **Experience**

Differences in experience are controlled to a large extent in many studies, but ratings of experience are generally subjective or assumptions based on past history, for instance, attendance at a certain course or amount of use of the system. However, this underestimates the variability in human computer interactions that occurs in natural settings where users with different backgrounds work side by side. The common sense view is that more time spent with a system will equate with more practice and therefore more commands will be known. However, this may not necessarily be the case with computer systems. Potosnak (1984) found computer use and computer experience to be two separate dimensions. Many researchers treat them as one and the same. Rosson (1984) also found that users do not exploit more powerful features of a system simply as a function of increasing experience (or exposure) to a system. Draper (1984) proposes that expertise does not mean that a user will know more commands but is better able to find the necessary information when it is required. Therefore it should not be assumed that a user will use more efficient methods

as they become more experienced. Draper (1984) found that experts on UNIX did not know more than a very small fraction of the commands available but were characterised by skill at discovering information as and when it was required. Mayes *et al.*(1990) carried out an experiment with users of the MacWrite word processor. There were three groups; occasional, intermediate and frequent users. Subjects first filled in a questionnaire which required them to recall commands and procedures from the menus. Recall was far poorer than expected. However, in a follow-up study subjects were required to carry out the tasks and could do so with little problems, more experienced users showing less hesitation. Thus while using the menus, users only retain sufficient information for recognition, not for recall. Studies of incidental learning have shown that people can often recall few features of very familiar objects. This suggests that it is more functional not to learn unnecessary detail. This may be an important feature of UNIX use which has not been addressed. UNIX is too big to remember all the commands available, therefore rather than exploring recall of commands, users should be shown how to find what functions are available when they require the information.

### **Age**

Most studies are not set up appropriately to study the impact of age on performance. The range of age used in studies is usually restricted and may correlate with experience. It is also the case that the effects of age are not well understood. There are age-related declines in performance, especially with complex tasks, but the reason for this is not known. However, Gomez, Egan & Bowers (1986) found age to be a powerful predictor of performance with a line editor and this prediction power held through at least two days of practise.

### **Culture**

Another perspective on individual differences is that of culture; ethnic, racial or linguistic background. Users who are brought up reading Japanese or Chinese will scan screens in a manner different to Westerners. Little is known about computer users from different cultures.

**Personality traits**

Despite its intuitive appeal, many studies have failed to show that this factor is important. Gomez, Egan & Bowers (1986) failed to find personality traits that accounted for more than 7% of variance on any measure of performance. Allen (1987) also found no evidence that personality measures were strong predictors.

An increasingly popular technique is to use the Myers-Briggs Type Indicator (MBTI) that is based on Carl Jung's theory of personality types. He conjectured four dichotomies;

- extroversion—introversion,
- sensing—intuiting,
- perceptive—judging and
- feeling—thinking.

The theory behind MBTI is that it provides portraits of the relations between professionals and personality types. It has been applied to testing user communities and providing guidance to designers. Yoder (1986, cited in Kearsley 1988) investigated the relationship between an individual's cognitive style, as measured by MBTI and their performance on two tasks using the IBM VM/SP and VM/CMS operating systems. She implemented five different formats of help messages that varied along the following dimensions:

- verbal versus graphical
- explanation versus example
- directive versus tutorial

Participants were asked to complete the tasks using only the help system as guidance. Total completion time, number of help requests, and errors made were taken as measures of performance and the subjects preferences were also noted. It is difficult to draw any firm conclusions because of the number of participants (36) and the large number of possible categories (16), but there were some suggestive results regarding preferences for certain formats depending on cognitive styles.

Perhaps the most important finding was that most subjects performed better with formats based on examples than abstract explanations.

Many hundreds of psychological scales have been developed including :

- risk taking—avoidance
- internal—external locus of control
- reflective—impulsive
- high—low tolerance for stress
- field dependent—independent
- high—low motivation
- left —right brain orientation

#### **Psychological aptitudes**

Differences in ability and skill dimensions may also affect performance. It may be that cognitive factors are more important than personality factors. There is a cluster of broadly related cognitive aptitudes that have been used as measures of individual differences with regards to human computer interaction. These include spatial and reasoning abilities, maths and science achievement, verbal and motor abilities and various personality dimensions. Egan & Gomez (1985) found that spatial memory affected editing ability, Carroll & Carrithers (1984) and Gomez, Egan & Bowers (1986) found that deductive reasoning affected advanced editing ability. Vincente, Hayes and Williges (1987) found that spatial visualisation ability made a significant difference in a file searching task, with subjects of low ability taking twice as long as those with high ability.

#### **Gender**

Few studies have systematically examined the effects of sex differences to the interaction process. Subject groups are predominantly or completely single sexed. However, some ability dimensions such as spatial, verbal and mathematical abilities have been linked with sex differences. Witkin *et al.* (1950 onwards) cognitive style dimension, Field Dependency, predicts a sex difference in

cognitive functioning. Fowler & Murray (1987) review the implications of sex differences for the interface. They predict that sex differences may result in preferences for different styles of interaction, with females preferring a structured, inflexible natural language interaction and males preferring a more flexible and graphical interaction.

## 2.2 Cognitive style

The body of literature on cognitive style is quite large and a variety of cognitive style dimensions have been proposed and investigated. Cognitive styles are usually thought of as characteristic modes of perceiving, remembering, thinking and problem solving. They reflect information processing regularities that develop around underlying personality trends. They are inferred from consistent individual differences in the ways information is organised and processed (Messick, 1985). A definition of cognitive style has been put forward in Robertson (1985). It states that cognitive style concerns

*differences in information processing which are stable across a variety of situations and are reflected in qualitative differences in the processing, storing or recalling of information.*

The paper by Messick *et al.* (1976) provides a glossary of 19 different cognitive style dimensions. Perhaps the best known and most widely-researched cognitive style dimension is that resulting from the work of Witkin and his associates from 1950 onwards. Witkin's research began with an interest in perceptual factors but developed into an attempt to conceptualise and measure an aspect of cognitive style known as Field Dependency. This can be measured using the Embedded Figures Test (Witkin *et al.* 1971). It is this dimension that will be central to this thesis. Other cognitive style dimensions identified include (adapted from Entwistle, 1981):

- conceptual systems approach,
- holistic and serialist approach,
- convergent and divergent thinking.
- reflective and impulsive approach

Style A (Right hemisphere)	Author	Style B (Left Hemisphere)
Holistic	Pask(1976)	Serial
Field dependent	Witkin et al.(1977)	Field-independent
Divergent	Hudson(1966)	Convergent
Impulsive	Kagan(1966)	Reflective

Table 2.1: Styles A and B from Entwistle (1981).

### 2.2.1 Similarities between theories

Despite the differences in the conceptualisation of cognitive style, there are striking similarities between many of the existing dimensions. Entwistle (1981) has reviewed some of this work and pointed to the similarities between various dimensions. He produced, essentially, a two category grouping which places the dimensions in two groups, style A and style B (see Table 2.1). However, despite its appeal, empirical evidence supporting it is limited. In addition to pointing out the similarities, Entwistle also suggests a possible link between the styles and the neurological activity of the brain. The right hemisphere of the brain is reported to be specialised for synthesis and to process information more diffusely than does the left hemisphere. The left hemisphere is predominantly involved with analytical and logical thinking and seems to process information sequentially.

Despite the neatness of this synthesis it may be too simplistic. It is clear that different parts of the brain do tend to specialise; however, it is far from clear how cognitive style can be related to brain function in any simple manner. There is also little evidence to show how the various dimensions of cognitive style relate to each other. It is possible to speculate that individual differences in information processes may arise, at least in part, from individual differences in how the cerebral hemispheres are organised and how functions are lateralised (Kolb & Whishaw, 1980).

### 2.3 The Field Dependency dimension

Witkin et al.'s (1962) theory of Psychological Differentiation is a global theory of personal factors and as such encompasses social, perceptual and cognitive functioning. In addition, it is one of the few theories which predicts a sex difference in cognitive functioning. The research by

Withkin *et al.* suggests the existence of two different cognitive styles, field-dependent and field-independent, which reflect different ways of processing and responding to information. The concept of Field Dependency which Withkin and his colleagues introduced and elaborated has generated an enormous amount of research.

The difference between the field-dependent (undifferentiated) and field-independent (differentiated) style lies in the extent to which the individual structures and analyses information. The field-independent style is characterised by analysis and structure, the field-dependent by a more global processing of information. The styles are correlated with results of the Embedded Figures Test (Withkin *et al.*, 1971), where the subject is required to locate a specific visual figure embedded within a more complex visual figure; field-independent subjects have less difficulty in finding the embedded figure.

The distinction between dependence and independence has been identified with many other test situations, for example using an auditory or tactile form of the Embedded Figures Test. It has also been shown to be closely related to other human characteristics and aspects of performance in the human-computer interaction area (examples are given in Goodenough, 1976 and Lotwick *et al.*, 1980).

### **2.3.1 Field-dependence**

Field-dependent people react to a situation as a whole without analysing it, responding on the basis of what it does rather than what they do with it. This style is associated with a holistic approach and a general acceptance of, or reliance on, the inherent organisation of material. People associated with this style are likely to exhibit greater interpersonal skills than the field-independent type. In learning and problem solving, such people tend to act passively and accept a situation as given. They prefer to be guided, and to rely on external referents.

### **2.3.2 Field-independence**

Field-independent people prefer to keep the different parts of the situation separate from each other, ignoring those parts which are irrelevant to their purpose. This style is associated with good analytical and restructuring skills. They will actively reorganise information according to



situational demands and impose structure where necessary. In learning and problem solving, a field-independent type may adopt a hypothesis-testing approach, actively exploring the situation. They are likely to form a mental model of the situation before proceeding. This model is then refined through experience as more knowledge is acquired.

## 2.4 Field Dependency: Ability, strategy or style?

The underlying assumption of the literature on cognitive style is that cognitive styles are individual differences which are not merely different types of ability. The cognitive-style dimensions are thought to represent differences in the manner in which people process information and so do not simply reflect whether someone has more or less of a certain ability. Messick (1976) suggests some criteria which distinguish 'style' from 'ability':

- Abilities are concerned with the level of performance, whereas styles are concerned with the manner of performance. Goodenough (1976) also emphasise this criterion. There is a qualitative difference in the type of information processing which takes place. This clear conceptual distinction is difficult to reconcile with much of the research conducted.
- Cognitive styles are bipolar. This has two implications: each pole of a style dimension has different adaptive implications for cognitive functioning; one style is not seen as superior or inferior to the other. This is unlike abilities which are unipolar.
- Abilities are value-directional (high amounts of ability are always preferable to low amounts) whereas styles are value differentiated. Witkin & Goodenough (1977) also emphasise this criterion.

McKenna (1984) argues that at both a conceptual and an empirical level, the Embedded Figures Test should not be regarded as a measure of cognitive style. At the empirical level there are doubts because of the correlation with measures derived from the test and measures of general ability. There has also been a high correlation observed between the Embedded Figures Test and measures of spatial and fluid ability, although these correlations have not been observed consistently. Fluid ability is the ability to reason independent of previous knowledge. It can be measured using a combination of intelligence and spatial ability tests. However, care should be

taken in the interpretation of correlation studies as significance can be reached with small effects if a large number of subjects are used.

One possible conclusion that McKenna (1984) suggests after his review of the literature is that the literature in which the Embedded Figures Test is used should be reinterpreted in terms of the above-mentioned concepts of general ability, spatial ability and fluid ability. Yet despite the evidence, practitioners have persisted in referring to Field Dependency as a cognitive style. It is interesting to reflect on why this is the case. McKenna (1984) considers two factors:

- It may be partly due to the attractive nature of the concept of cognitive style. According to the concept, it is not that one person is better than another, it is more that they are different. When this is combined with a general dissatisfaction with psychometric testing, enough impetus for research may have occurred.
- In spite of the debate over the nature of exactly what is measured by the Embedded Figures Test, it is a very good measure of individual differences.

At the operational level the Embedded Figures Test seems to merely separate people who have more of the ability to identify embedded figures. This perhaps would not be a serious criticism if a parallel test existed which could provide the same differentiation of subjects. This criticism however, does not negate the importance of the work done on Field Dependency. The important factor in this research is the relationship of field dependency to the process of interacting with a computer.

Another conceptual difficulty concerns the distinction between cognitive style and cognitive strategy. Messick *et al.* (1976) make the point clearly, "It is important to distinguish cognitive styles, which are high-level heuristics that organise and control behaviour across a wide variety of situations, from cognitive strategies, which are ... a function of the conditions of particular situations". People can and should develop different strategies for different occasions (Tyler, 1974). However, in most situations an individual will usually adopt a strategy which is consistent with their basic cognitive style. However differences may appear in conditions where the task is biased towards a particular strategy and/or where an individual feels stressed and so reverts to a less sophisticated strategy. Such stress situations include time pressure and unfamiliarity with the task or tools.

Although in the literature cognitive styles are often treated as discrete ways of handling information, there is in fact a continuous variation between the extremes which are given the label. It seems likely that both extremes are available to most people, although with differing degrees of effectiveness. Nevertheless, many people do develop a preference for one or other style which may become apparent in their approach to learning etc. In fact, being able to integrate the two styles may prove to be the most beneficial way of dealing with the situation, thus being able to analyse and sequentially collect details while still not losing sight of the global aspect and how it all fits together.

## 2.5 Why use Field Dependency?

It is necessary to establish why Field Dependency, as measured by the Embedded Figures Test (EFT) was used in this study. The following reasons were important to that decision:

- The test is well-established and recognised. It is relatively quick and simple to administer.
- The test is not simply a measure of a person's intellectual ability. It does not correlate highly with measures of general intelligence.
- Longitudinal studies have shown a person's level of Field Dependency to remain consistent over time. Since the characteristic does not change easily, it is useful to investigate how different levels of Field Dependency affect the interaction process so that the differences can be accommodated in future systems.
- It is a theory which predicts a gender difference, with females, on average, being more field-dependent than males. This is an interesting concept in the field of computing, as it attracts so few females and different occupations are known to attract people exhibiting different levels of Field Dependency.
- Field Dependency can be taken simply as a theory of information processing. Different levels of Field Dependency affecting the way in which information is gathered and the representations used internally to store and manipulate that information. However, it can also be interpreted at other interesting levels, for instance, level of Field Dependency has been

explored with relation to other human characteristics including sociability and interaction with other people. Although, not central to this thesis, it is interesting to look at the types of personalities who do well in the field of computing.

- It is already being used by others in the field yet little is known about the validity of its use.

Which category one chooses to place the concept of Field Dependency in, be it style, ability, or strategy, it still remains the case that it is a measurable difference between individuals and it has been linked to differences in the extent to which people use and their preferences for different computer systems. Therefore further investigation of the link between Field Dependency and the use of computers is justified. The following sections discuss some of the reported links.

## 2.6 Field Dependency and cognitive models

The concept of Field Dependency may be useful in relation to the idea of a user's cognitive model (Saja, 1985). This relates to the way in which relevant information is internally represented and organised in users' memories. For any non-trivial system, the cognitive model may be inadequate: because the system being modelled is inherently complex, the cognitive model tends to be incomplete and contain ambiguities and conflicts. At some point, either through reading material prior to any interaction with the system, or experience with other systems, or through actual interaction with the system, users will begin to form a cognitive model of the system. This model contains what the users believe to be the functions, capabilities and limitations of the system. An inaccurate or incomplete cognitive model will lead to user dissatisfaction and lack of confidence in the system, hence, inefficient use of the system and human resources may result. The user interface incorporates all aspects of the system which users come into contact with physically, perceptually and conceptually: it is the interface that provides users with a link to the capabilities of the computer. Thus the interface must present the system to the user in such a manner as to guide the formation of the cognitive model. Since it is the cognitive model which directs users' actions and reaction to the system, Hammond et al. (1982) stress that the better structured the information is, the *"more usable the system is likely to be"*. Therefore the ability to structure and restructure knowledge concerning the system, according to situational demands, could be regarded as a useful

user skill.

It is thought that the field independents, who are more internally orientated may form more efficient and elaborate cognitive models. As well as forming more efficient models, as a result of their more exploratory nature, field-independents may form more complete models. They use the interaction to refine and add to the model, thus seeking out system functionality. The field-dependents, relying on external referents, are more likely to stick to standard procedures or methods which they know to work through experience, although they are not necessarily the most efficient methods. Therefore the use of a rich command language may be wasted on the field-dependents (Boies, 1974).

Carey (1982) saw 'style' differences in the way people approached a particular task. One style involved the development of a cognitive model before carrying out the task, while the other involved developing a model during the task. If this is the case, then pre-task instructions and off-line documentation, might not be fully utilised by field-dependents, preferring to form a cognitive model during task execution. This idea was further backed up by Fowler *et al.* (1985). Their results suggest that field-independents are more likely to develop cognitive models before carrying out the task. On the other hand, field-dependents were more likely to develop the cognitive model later, through experience with the task. These results highlight the need for provision of different modes of access to help information. Field-independents would benefit from being able to browse the help information to get an idea of the functionality of the system BEFORE they interact with the system, whereas field-dependents would benefit from easy access, DURING the interaction, to help relevant to their current task. It is necessary for any assistance, given by the system, to accommodate these different preferences. McDonald & Schvaneveldt (1987) saw the goal of on-line assistance to be to allow users to develop accurate cognitive models efficiently, by interacting with the documentation.

## 2.7 Field Dependency and Human-Computer Interaction

The relationship between the Field Dependency and Human-Computer Interaction has been investigated by a number of researchers (Boies, 1974; Carey, 1982; Fowler *et al.* 1985; Van der Veer *et al.*,1985; Fowler & Murray 1987; Fowler *et al.*,1987). The relationship of Field Dependency to

Field-dependent	Author(s)	Field-independent
Passive role	Goodenough (1976)	Active role
develop mental model as and when information is given	Carey (1982)	develop mental model prior to use and refine through experience
less likely to use analogy	Van der Veer et al. (1985)	more likely to use analogy
less likely to transfer knowledge from other system	Van der Veer et al. (1985)	more likely to transfer knowledge from other system
less flexible/innovative	Fowler et al. (1985)	more flexible/innovative
learns only basic commands	Boies (1974)	likely to discover new commands
relies on external frames of reference	Fowler et al. (1985)	relies on internal frames of reference
prefer structured command language	Fowler et al. (1985)	prefer linear command language
prefers to be system-guided	Fowler & Murray (1987)	prefers user-guided
prefers inflexible dialogue structure	Fowler & Murray (1987)	prefers flexible dialogue structure

Table 2.2: Field Dependency and HCI

different aspects of a Human-Computer Interaction are summarised in Table 2.2.

Some researchers (Fowler & Murray, 1987; Witkin & Goodenough, 1981; Tyler, 1974) have reported that the nature of sex differences lies in the cognitive style most likely to be adopted when processing information. Males are thought to be, on average, more field-independent than females. Despite the interest in this topic, very little empirical work has been undertaken to establish the nature of the importance of this dimension.

Interest in the Field Dependency of users has been confused in investigations into the nature of sex differences between users of computer systems. The paper by Fowler and Murray (1987) concerning gender and Field Dependency differences is both confusing and self contradictory. It is confusing because of the authors failure to clearly separate preferences which they attribute to Field Dependency, such as dialogue structure, from those which they attribute to other sex differences such as a preference for verbal, sequential presentation or pictorial and parallel presentation. Other points of controversy are as follows:

- The authors state that females may prefer natural language content. They also say that because of their relative field-dependency they prefer structured, inflexible interactions. Nat-

ural language as input is neither inflexible nor necessarily structured. Fowler *et al.* (1987) have also reported that field-dependents do not perform well with natural dialogue content. Females have been reported to be, on average, more field-dependent than males.

- The authors say that females prefer a 'natural' (verbal) dialogue whereas males prefer pictorial. In some situations, such as describing a hierarchical file system, a diagram (i.e. pictorial communication) may be the most "natural" way to communicate the necessary information.
- The authors say that question and answer dialogues are less flexible than menu dialogues. If natural language responses are used then the response range is wide-ranging whereas the response to a menu can only be a combination of the given menu items.
- The authors assume that because Field Dependency is associated with greater interpersonal skills, these people will want to be sociable with the computer — "*They seem to need the social contents which are obviously missing from existing computer dialogues*" — Natural language research was originally designed to increase the flexibility of a system's input rather than to increase its sociability.

Fowler *et al.* (1985) carried out an exploratory study which examined the relation between Field Dependency and command structure, as reflected in various measures of performance over two learning blocks. Field dependency was measured using the Group Embedded Figures Test (GEFT). The structure of the command language was manipulated into either a linear structure where commands and arguments were given as a single user response or into a substructured form where a set of prompts were given for the arguments required for each command. Performance measures used were time taken to complete the task, both thinking and doing, the number of 'help' requests, and the number and type of errors made. It was thought that field-dependents would prefer the guidance of the substructured form as the linear form requires a good conception of the overall task which requires the quick formation of a cognitive model.

Fowler's study investigated the performance of 48 clerical staff. The subjects were predominantly female, all with little or no computing experience. The experimental task was a computer-simulated filing task which involved both editing and retrieving files. The results were analysed using correlation techniques. There were significant differences in favour of field-independents on

the first learning block, but by the second block the differences were not significant. This supports the hypothesis that the field-independents formed an appropriate cognitive model quicker and thus initially performed better but as the task progressed the field-dependents formed a cognitive model and their performance improved.

Interestingly, despite the overall difference in the number of errors there was no significant difference between the groups in the number of times help was requested. However, when the order of presentation of the structures is looked at, the field-dependents asked for more help on the second block if the order was substructured followed by linear. This suggests that field-dependents suffer more from the removal of an 'interaction aid' such as structure.

The previous study highlights the idea that differences in performance can be found, especially within the initial learning phase. However, the study has used a mixed sex group, which may lead to experimental error if this variable was not controlled, and makes no mention of the actual scores the subjects achieved on GEFT. Therefore it is impossible to decide the range of Field Dependency the study actually encompasses.

A third study (Fowler et al., 1987) looks at Field Dependency and system expertise as potential predictors of dialogue preferences. The dialogues used were question and answer, menu and command language. Time taken to complete a number of tasks was used as a measure of success. However, the time measurement only included error free time. Time taken for errors and looking up help was not reported. Neither was the number of errors and number of help requests. It is questionable whether error-free time is a sufficient measure of usability. Also in their selection of subjects for this experiment, out of a pool of 60 subjects the 20 subjects whose scores on GEFT fell in the middle of the distribution of scores were eliminated. Only the extremes were used. Yet these were real scores and potential real users and their performance and preferences were not considered. Also in the two groups that were formed four subjects in the field-dependent group and five in the field-independent group lay within one point. In the experimental design only 4 subjects were used in each condition, the elimination of a group of subjects on the basis of their scores means that the groups were not random and so the use of some statistical techniques is not appropriate and can be misleading.

It may be that a persons level of Field Dependency has an affect on the way they interact



with a system. However, the studies carried out in this area are flawed and there is need for more work in the area. It may also be the case that a system may be biased towards a particular level of Field Dependency.

### 2.7.1 Potential system biases

Exploration is required of the conditions under which inherent system biases may result in significant individual differences in performance. Such biases are discussed by Thimbleby (1980) under the term, *dialogue determination*. The relationship between the computer and its user should be well-determined, i.e. well-balanced. This requires neither to be in ultimate control. This balance is not reached if a system is over-determined or under-determined with respect to the particular user. The level of determination a system exhibits will affect the users' feeling of control over that system.

Over-determination is brought about by excessive control on the part of the computer. The computer actually controls and restricts the user's behaviour, thus denying the user choice and participation. The user must formulate the task to suit the computer. This may aid a field-dependent person to interact with the system by providing guidance and structure, but restrict a field-independent person, although it does attempt to reduce the consequence of user errors. This reduction may prove useful while exploring new parts of the system.

Under-determination is brought about by the computer leaving the user with no external referents, eg. insufficient visual feedback, and at a loss how to proceed. The computer may not be exerting control over the user, but it is failing to help or guide the user. The basis of the dialogue is insufficiently defined for the user to operate adequately. This is particularly a problem to a field-dependent type, but allows the field-independent person freedom.

There are several aspects to dialogue determination:

**Flexibility** A system is inflexible (over-determined) if for instance, it requires fixed-length inputs. However such inflexibility has been shown by Fowler *et al.* (1985, 1987) to increase the speed with which field-dependent subjects performed computer-presented tasks. Too flexible a system is formless and could become under-determined, for instance a natural language

dialogue.

**Commonality** is the degree to which an aspect of the system is common to other parts of the whole. This is related to user predictability and consistency. The more consistent the interface language is the better able the users are to generalise existing knowledge. This is appropriate to all levels of Field Dependency.

**Immediacy** is the delay in system response time. This contributes to the under-determining of the system. The shorter the delay, the more likely it is that users will attribute their actions to the consequence. Immediacy is related to the concept of closure, the subjective sense of reaching completion. Without closure users are subjected to cognitive load. This is useful for all levels of Field Dependency.

**Intromission** refers to the system pacing the user. If users reach closure, for instance they notice an error and cannot stop the interaction (no intromission), they may feel that the system is over-determined. Intromission contributes to a system being over-determined but it has been shown to be preferred by field-dependent people (Fowler et al. 1965, 1967).

**Variability** As the user moves from one subtask to the next, the computer may vary the language, for example `rm *` might be responded to by the computer with `Are you sure?`. Here the dialogue is varying toward over-determination. It is assuming that the user may have made an error and restricts the input. However, at the risk of losing system consistency, it is a good safety measure to have extra interaction at dangerous points.

**Feedback** The form of feedback can be broken down into two aspects: visibility and clarity. The visibility of the system is the only way users have of knowing what the computer is doing. There is a need to show users what is happening as things happen. There should be no side effects which are invisible to users. As long as this is combined with good intromission then the system is not being over-determining. If the dialogue lacks immediacy then there should be some indication that the command has been accepted — otherwise the user may repeat it. Clarity of messages and messages expressed in terms meaningful to users is also essential.

If the help system is adaptive, and therefore determining:

1. the user's current state of knowledge
2. what information to present to the user and/or
3. when that information is required by the user

then the help system runs the risk of being over-determined. If on the other hand there is natural-language access to the help system then it may run the risk of being under-determined. Both over- or under-determination are usually bad. Ultimately the degree of determination varies with the individual user and may be dependent on the users level of Field Dependency.

## **2.8 Field Dependency and online help**

Van der Veer *et al.*(1985) use the term metacommunication to describe how a system communicates information about the conceptual model underlying the design of the system. It can serve many purposes which include:

- introducing a novice to the system
- handling user errors
- reminding users about available commands
- informing the user about the current state of the system.

They acknowledge that metacommunication occurs both during the interaction, through online help and error messages, and outside the interaction, through teaching, manuals and other off-line documentation. They suggest that the form and content of the metacommunication, and therefore the online help, should vary across different types of individuals depending on their cognitive style. For example if the user prefers to develop their mental model through hands on experience then it is important that the system can cope with this. If the user wishes to develop a model before interacting with the system then this must also be dealt with. Metacommunication consists of the system's online help and error messages as well as the learning environment the system provides. It is therefore critical that help information is in a form which facilitates formation of an appropriate mental model.

The next aspect of metacommunication to be considered is whether metacommunication should be active or passive. It may also be the case that a field-dependent user would benefit from an active, system-driven help system whereas the field-independent may prefer passive help which allows the users to be in control of requesting help. The form the help information takes is also important. The field-dependent users, who find it difficult to impose their own structure on the information, may require information which aids this process, whereas the field-independent users may require just the facts and will structure them for themselves.

Thus the following points should be investigated:

- structure of the help information
- content of the help information
- the role of the computer — active or passive.

### 2.8.1 Minimising variation in user performance

It is not only the case that individual differences in performance are large and systematic, but individual performance is also modifiable. Patterns of performance change depending on the design of the interface and training received. It should be remembered that a particular interface design may determine who will have greatest success with the system. Egan & Gomez (1985) propose three steps to redesign interfaces to accommodate a wide range of user diversity:

1. *Assay* user differences and performance to find out what characteristics predict differences in performance. This gives a clue to the kind of difficulties experienced.
2. *Isolate* the source of variation in a particular task or system component.
3. *Accommodate* differences among users by redesigning the task procedures or the interface.

These can be illustrated by an example: field dependency may predict lack of knowledge of command names which may limit performance. Performance can thus be improved by providing user-orientated help which eliminates the need to know the exact command name.

## **2.9 Methods for studying UNIX users**

Having discussed the potential of individual differences, the next stage is to establish a method for studying the users of computer systems.

The main aim of this thesis is to evaluate a help system against a number of benchmark tasks. A prerequisite to the provision of a help system is to first "know the user" (Hansen,1971). How should researchers find out about the user community? Three possible methods of studying the user are discussed.

### **2.9.1 Natural studies**

One way to get to know the users and their problems is through the analysis of everyday, natural interactions with the system. This involves no interference with the everyday work of the users on the part of the researcher. Researchers only observe natural occurrences. Such studies are referred to as natural or observational studies. Observational studies allow for the discovery of computer features which are problematic to users and provide data on which to base interface guidelines. They also enable, to some extent, the size of the user's command knowledge and misinterpretations of the system to be evaluated. However, observational studies are not easy and there are several problems associated with them:

#### **No established methodology**

Past researchers have used various methods. This makes it difficult to contrast and replicate work. Even when similar methods are chosen, lack of controls make comparisons questionable.

#### **Lack of control**

Because no intervention is possible, research is restricted to what the user does while carrying out their normal duties. Therefore if a behaviour is not naturally observed, it is difficult to establish why without follow up work. It also means that it is difficult to see how different people would react in similar situations because everyone is doing different things.

#### **The difficulty of collecting data**

Monitoring real-life human-computer interaction is not easy. Source code may not be avail-

able for modification; security measures may preclude monitoring; if the user is told about the monitoring, or monitoring is carried out in a more overt manner such as video taping, then behaviour may change. There is also a moral problem, for, if not told about the monitoring, their privacy is being invaded. Monitoring users takes processor time and physical records of user activities need substantial disk space.

An observational approach is useful if researchers are evaluating the real usability of a system or specific feature. This way, statements such as "it is difficult to remember the correct combination of options for this command" can be noted and discussed.

### 2.9.2 Controlled laboratory studies

The standard methodology in the study of human-computer interaction is the traditional controlled experiment. Until recently, the main method of studying the user-computer combination was that of hypothesis testing. This technique was borrowed from experimental psychology. Experiments are used to test these hypotheses. Hypotheses are usually specific and only valid within the domain in which they are tested, ie. the laboratory setting. Generalisation to normal work conditions is sometimes difficult. Another problem is that it does not deal efficiently with highly multivariate phenomena such as human-computer interaction. In situations where it is difficult to isolate a few controlling variables, investigating people's natural use of a computer system may be a better method.

The particular advantage of laboratory studies is their high degree of convenience and control. In the field it may be difficult to gather a large enough sample of low frequency, but perhaps critical, problem episodes to evaluate potential solutions. Given the belief that the human-computer system is highly complicated and often prone to extreme dependencies on details, studies in natural settings are likely to be very noisy and some of the variability may be reduced in controlled experiments. By the same token, the major criticism of experiments is that the social, motivational and environmental context of real use is important but excluded from the situation. However, Landauer (1989), states that this may not be as large an issue as some would make out:

*I know of very few instances of dramatic differences in the relative overall usability of computer systems, or of the effect of feature variations, or principles between laboratory*

*tests and the real world.*

In fact many useful systems have been shaped by laboratory tests including the Olympic Message System (Gould *et al.*, 1967).

One of the common uses of experiments, and the reason for their inclusion as a method in this thesis, is to test a prototype system on representative users carrying out representative benchmark tests. Difficulties, design flaws and suggestions for improvements from the users' perspective can be gathered from observation of the experiment and user comments.

Experiments facilitate the comparison of different systems over place and time using a benchmark set of tasks. However, it is not always easy to design a representative set of tasks. This allows the researcher to separate the effects of the system characteristics from the task characteristics, for instance to answer the question, "why are some commands not used"?

In an experiment the researcher can also deliberately select groups so that their performance can be directly assessed rather than the performance of some mythical "typical" user (Egan *et al.*, 1986). In the field the researcher is often unable to select the desired sample.

A well-designed experiment is not easy to accomplish. The researcher must be extremely careful to select subjects who are representative of the real user community and benchmark tasks representative of actual common usage.

Examples (Rhyne and Wolf, 1986 and others) suggest that natural studies and controlled experiments complement each other. The most successful efforts have employed a variety of empirical behavioural methods to converge on a system design.

### 2.9.3 Theoretical models

Theoretical models, or formal descriptions have recently gained popularity as a means of describing the way in which the task of the user maps on to the system syntax. These models are used to deduce generalities from existing truths about how people process information. Notable examples are Moran's 'Command Language Grammar' (1981), the Keystroke Level Model (Card *et al.*, 1980) and the subsequent Goals, Operators, Methods and Selectors, or 'GOMS', model developed by Card, Moran and Newell (1983). There is also the Task Action Grammar, or 'TAG' (Payne & Green, 1986), and Kieras and Polson's 'Formal Analysis' (1985).

Basically, the claim is made that the interaction between the user and the computer is centred around a specific task and that the actions required to complete this task can be expressed formally. In essence, it offers a means of specifying goals and sub-goals for a task and mapping these to a command sequence. Following such a process is believed to aid consistent interface design.

The strength of these methods lies in the ability to

- model goal-orientated behaviour,
- model the cognitive limitations of the user.

However, there are several limitations to this approach :

- It is based on a very detailed and sophisticated low-level task analysis and therefore has only been developed for very limited domains (eg. simple text editing) where the user's behaviour is very well understood.
- Formal analysis relies upon the idea that an analysis of the task and a specification of the structure and syntax of the interface is a sufficient description of the interaction, but in fact it does not reflect a full understanding of the psychology of the user. Similarly, the GOMS model does not address individual differences which may play a significant role. In other words, the formal description does not reflect the actual behaviour of the user because, there are always some aspects of behaviour which are contingent upon unplanned events (Briggs, 1988).
- They fail to acknowledge the importance of knowledge and experience as a vital part in determining the user's interaction with the computer despite much evidence to the contrary (Carroll & Mack, 1984; Shrager & Klahr, 1986).
- The GOMS and Keystroke models of Card, Moran and Newell (1983) were based on skilled and error-free performance. The models do not address users who are generally unskilled and frequently commit errors and thus is extremely limited in its applicability. Naive users often develop goals which deviate markedly from the formal task descriptions (Briggs, 1990).

Elkerton (1988) suggests that GOMS can be applied to the design of on-line help and Gong & Elkerton (1990) have applied the GOMS model to the design of a minimal manual. Using GOMS



allows designers to focus on real tasks. Gong & Elkerton (1990) also propose that the GOMS model can be extended to provide error recovery procedures for identified error states. Novices using the manual were observed on tasks representative of those that users would encounter in the field. The results of the study support the claim that minimal manuals improve performance of users learning a specific application, especially if the manual provides task-orientated, procedural instructions. However, this method is still only appropriate for users with very specific goals and therefore may be considered as supplementing the full manual.

As a result of the limitations given previously, at present there are only small scale uses for such models and their validity as a practical method of studying real users is questionable. It may well be that formal methods are only useful if the user is guaranteed complete mastery of a system's functions by being given complete instructions and repetitive practice. They may be totally inappropriate if self-directed, exploratory learning was used.

## 2.10 Analysing the activities of computer users

Having established that the type of study to be employed in this thesis is to be an observational study to establish benchmark tasks for a follow on experiment, the next stage is to establish an actual method for collecting the necessary data.

A problem with studying computer use is the multiplicity of methods for data collection. UNIX is no exception, and data collected in studies range from low level traces of user normal activities through to protocol analysis of a few subjects. This section reviews a number of possible methods including the method employed in this thesis.

### 2.10.1 The "Wizard-of-Oz" technique

The "Wizard-of-Oz" technique (Gould *et al.*, 1983; Kelley, 1983) is one where subjects interact with a computer through standard interactive dialogue at a terminal and believe they are conversing with the computer. However, subjects command lines are sent to another monitor where a human expert, known as the "wizard" sends back a reply to the subjects. This technique may be useful if a full system does not exist. This technique is beneficial as

- it mimics current standard human-computer interaction

- there is no failure of the program as there is no program
- the setting is controlled as the subject has a set number of tasks to accomplish.

This technique has been strongly argued for in the design and evaluation of natural-language interfaces (Guindon, 1988; McKeivitt, 1990). However it is not without problems, including the slow response time of the "wizard" and the need to generate messages quickly leads to the sending of inadequate or incorrect messages.

The technique was tested by McKeivitt (1990) in a study where he asked subjects to carry out a number of tasks and told them they could send a natural language request for help at any time. The "wizard" would respond with an appropriate command line. His data was analysed from the perspective of developing a natural-language interface and not how users performed or what they thought of the system.

### 2.10.2 Traces of user activities

A record of the conversation between the user and the computer can be collected via an unobtrusive software monitor. This is called a trace. A trace records all actions occurring by either the user, the computer or both. This data is then filtered to see if anything interesting comes out of it. The trace may be collected while the user is using the computer normally or the user may be requested to solve problems set by the researcher. A measure of the validity of setting problems as a method of study can be obtained if the traces generated in the normal work condition are similar to those generated in the problem solving phase (Lewis, 1986).

There are a number of methods which generate traces of UNIX interactions. Three alternatives are described.

#### Record all keystrokes entered

Every single character generated by the user is included in the trace. This includes special characters such as < *backspace* > and < *return* >. This is easily implemented and captures everything that the user does. However, unnecessary data is often collected and the command lines are difficult to read. For example `cat fileAHAHAB-n fileAM` translates to `cat -n file`. Another problem with this method is that it records only the actions taken on the part of the user.

**Session transcripts**

An alternative to keystroke collection is to collect the complete transcript of a user session. This includes the user's input and the system's response. The transcript is then reviewed as an animated playback. Transcripts are rich with information, but the information they provide is too voluminous for anything other than small studies of a particular topic. An example of the use of this method is reported by Akin et al. (1987) who studied how users move about in the UNIX directory space. Despite only using two subjects for a half hour period they report the transcripts as lengthy and hard to analyse.

**Record complete command lines**

Instead of capturing keystrokes as they are entered, the complete command line can be captured and filtered to eliminate the noise produced by user editing. This information can then be stored in a file and dated so that all sessions can be recorded separately. They can be merged later into one session if necessary. This method also does not record the systems response and is a tradeoff between recording too much information and too little. However, Desmarais and Pavel (1987) collected and analysed short-term UNIX traces by this method, and applied the information to generate user models.

For the observational study in this thesis, complete command lines were collected and analysed as the aim of this stage was to collect data concerning the order in which commands are learnt and the subsequent frequency of command use. For the experimental study, all keystrokes entered were collected as it may be useful to see the thought process that goes into forming each command line for each benchmark task.

**2.11 Statistical evaluation of data**

One characteristic of measures of user behaviour is that they are subject to chance influences. Within a group of subjects there may be variations due to differences between the subjects. Even within measures of one subject, there will be variations due to uncontrolled changes in the environment, user alertness, motivation or learning. It is thus necessary to report not only the central tendency of group performance such as the mean but also a measure of the range

of individual results. When assessing the results of experiments any difference in performance observed between different experimental conditions may be due to chance fluctuations rather than experimental manipulation. Inferential statistics allow this possibility to be tested.

### 2.11.1 The quest for statistical significance

Significance is achieved if the probability of getting a result, or a better result, by chance is less than some arbitrary significance level. The most commonly used level is the .05 level. This represents an error rate, therefore if it is always set at this level then in 5 % of experiments, where there was no effect, it will be wrongly concluded that there was an effect. If the level is set higher to avoid the risk of false positives then the risk of false negatives is increased. This level has been arrived at by balancing the risks of both types of errors <sup>1</sup>.

### 2.11.2 Correlation

Correlation statistics are used to measure the strength of association between two variables, for instance current performance and experience. Pearson's product-moment correlation, signified  $r$ , will quantify such a trend. If  $r$  is near to 1 there is a strong positive relation. If  $r$  is close to -1 there is a strong negative relation. If  $r$  is close to 0 there is no relation. The significance of  $r$  depends on the number of subjects as well as the strength of the association and refers to the probability of getting a relation of the same sign. There are a number of pitfalls associated with the interpretation of correlations. The three most common are:

- Attributing causality — causality cannot be attributed on the basis of a correlation. There may be some other factor causing the effect and the direction is not always obvious.
- Drawing strong conclusions from a small number of subjects
- Confusing significance with strength — the significance of a correlation depends on the number of subjects and the strength of the association. If the number of subjects is large, then quite small correlations are significant.

---

<sup>1</sup>It should be noted that the samples used in this study are not strictly random and therefore the use of statistical techniques is not strictly appropriate. However, they are presented because of their heuristic value.

## 2.12 Summary

This chapter has reviewed some possible measures of individual differences, in particular that of Field Dependency. The importance of this as a dimension of the user which the system should accommodate is discussed by reviewing how it relates to cognitive model formation and interacting with a computer system. The potential of a help system to accommodate differences in Field Dependency by providing a bridge between user and system biases is also discussed. The next stage is to investigate existing help systems and how they can be designed to improve user performance.

This chapter has also reviewed the methods available to carry out research to establish the importance of individual differences for users of computer systems. Statistical significance does not tell the whole story, therefore, for all information vital to solving the usability problem to be included in the analysis, a combination of formal and informal evaluation is required.

To summarise, there has been a recent move away from the constraints imposed by hypothesis testing and experimental set-ups (Landauer, 1987; Landauer et al., 1989). Such an approach requires years of research to repeat the experiment and gradually test the validity of any generalisations. Such a time scale is not appropriate for the study of systems where technology is changing at its current, fast rate. The reaction has been a move towards observational studies and theoretical models. Theoretical models have not yet reached the stage where they can be applied to research such as that undertaken in this thesis. Field studies may prove useful in answering questions more general than can be answered by specific experiments. Therefore for the purposes of this thesis a combined approach of observation followed by experimentation was applied.

## Chapter 3

# Online Help Systems

The previous chapter examined dimensions along which potential users may differ and the effect these variations may have on the interaction. In this chapter an investigation is made of existing online help systems, which identifies potential problem areas and useful approaches and solutions. Previous research of the usability of help systems is also examined. Previous investigations carried out with these systems will serve as a useful insight into the success of different approaches.

### 3.1 Supporting the user

One way of supporting the user is to provide an online help system. An online help system is designed for those users who know how to use the system but who sometimes become confused or are unfamiliar with specific commands. Traditional online help systems are poorly used and disliked by many. It may be that the pre-packaged explanations of the older help systems are insensitive to differences in users' goals and levels of understanding; or it may be that insufficient thought and effort was put into the development of the help text. Many approaches to improve help systems have been tried.

The information available through an online help system is primarily designed to help the user find out the function of different commands and how they can be used. For example, a user might forget which command is required for a particular task or what a particular error message means. Also, even experienced users, very familiar with part of the system, may be vague or totally unfamiliar with another part. It is usually left up to the users to figure out what they need

to know, locate the required information and relate it to the task at hand.

## 3.2 Design issues for online help

In looking at alternative designs from the user's perspective, there are three major issues to consider. These issues involve whose initiative first stimulates the help activity, how the user may request help (access mechanisms) and presentation of the help information.

### 3.2.1 Access Initiative

The initiative to activate the help system may come from one of two sources: the human user or the computer program. In the first case the help system is known as "passive" and in the second as "active".

Passive help may be activated by some explicit access mechanism: for example, by typing the word 'help' or choosing a menu option. With active help the program will intervene to provide the user with information as a result of suboptimal behaviour, a delay in user response or an error. In practise, it is rare to find a system in which the computer program has complete control. More often the system is totally passive or it has a somewhat mixed-initiative.

Most systems assume that help will be initiated by the user. One aspect of an intelligent help system is for the system to know when the user requires assistance and to actively present the required information rather than passively waiting for a help request. Information is presented as advice or suggestions. It is usually triggered by error conditions or some other evidence that the user has a problem, for instance missing parameters or repeated commands. These systems tend to address the problem of improving users interactions with the system by pointing out shortcuts or unused facilities. However, users may perceive this as an interruption and may prefer to initiate help requests.

Fischer, Lemke and Schwab (1984) have developed two related knowledge-based help systems. Activist and Passivist, respectively, are active intervening and passive request-driven systems designed to provide assistance for an EMACS-like editor. Passivist takes natural language questions or requests for help and interprets them in light of the current context of the user, that is with respect to what the user has been trying do. From this information, Passivist tries to deduce what

information the user is seeking.

Activist is an active help system that monitors user behaviour and intervenes when it detects the user performing below some optimal level. There are two kinds of performance level used. First, the user could invoke several commands to do something that could have been done with fewer commands. Secondly, the user may use more than the minimum number of key strokes for a command. For instance, a user might type a full command name when a function key would do the same job. Here the system must infer the user's overall plan and provide information appropriate to that goal. This system is rather narrow-minded, as the metric used to identify suboptimal behaviour is a count of keystrokes.

A problem with the active approach is that of balance. A help system which bombards the user with help messages ceases to be helpful and becomes an annoyance. Yet if help is not given when needed by the user, the user is left at loss what to do next. Another problem is that the user may have some reason unknown to the system for using the longer version of a command and a comment thought helpful by the system may be counterproductive and annoying to the user. In such situations, if the user refuses to change from their preferred method of working then it is necessary for the system to adapt its policies towards that user.

The active approach is qualitatively different from the passive approach even if the passive system is adaptive. If the user looks in a manual or menu system for information and fails to find it then it is frustrating. However, if the system fails to provide information when it is needed then the blame will be placed on the system and confidence in the system will be reduced.

### **3.2.2 Access mechanisms**

There are actually few access mechanisms implemented, although there is room for wide variation in their implementation. The most common mechanisms are:

- keyword access
- menu access
- contextually invoked access
- graphically invoked access



- natural language help requests
- spoken help requests

#### Keyword-based access

Traditionally, online assistance for command line interfaces has been based around command assistance. This requires the user to type "help" or in UNIX, "man", followed by the name of the command. The command name depends on both the concept it represents and the operating system in use. Furnas et al.'s paper (1987) describes how new and intermittent users often use the wrong word for the command name. Knowing the correct command name is no easy task. Therefore an approach which requires the user to know the correct command name to be able to request information about it is of little use. An improvement is to provide a synonym or alias capability or as on the VAX/VMS, typing "help help" produces a list of some of the commands available and "apropos" on some UNIX systems.

DOCUMENT (Girill, 1983) is a help system primarily geared at producing on and off line help from keyword requests. The system has three levels of expertise and the verbosity of prompting and error messages varies according to this.

#### Menu-based access

An alternative to the command access to information is to provide a hierarchical help menu facility which does not monitor the users or what they are trying to do. Commands are grouped together into categories, each representing a class of activities. The user can browse the information and develop an awareness of the size of the system and the type of activities that can be carried out. This enables the user to identify which command is required for the task to be completed. However this may still overwhelm novices with the amount of information they can access. An example of the categories that can be derived is given in Hanson et al., (1984). In this study, multivariate analyses led to a classification of UNIX commands into five categories:

- editing commands that shape text and other objects. These range from text editors to redirection and filters
- orienting commands that inform users about the environment and their current status

- social commands that allow users to exchange information
- process management commands that are used to integrate individual commands into more complex units
- task-specific commands that are required to fulfill specific job assignments, for instance, databases or statistical packages

An extension of the menu-based interaction is illustrated by AT&T's ASSIST software which uses a menu-driven system in conjunction with a form-filling interface to develop command lines for users to issue when they exit from the ASSIST software. Users can also try out the commands, without affecting their workspace as ASSIST forms a copy of their filestore.

#### Natural language access

An obvious possible access mechanism for online help is natural language. Natural language interfaces are computer programs which enable human users to interact with the computer through typed natural language such as English. This technique has been applied to online help systems (for example, Finin, 1983; Wilensky *et al.*, 1984; Jerrams-Smith, 1989). From observing human-human advisory sessions, it was thought that the language aspect of the interaction was important and providing a natural language interface became the vogue in computer research for some time and this was reflected in Human-Computer Interaction research. It was thought that these systems could alleviate the problem of the user not knowing a command name and instead would tell the computer what they were trying to do. An issue of debate is whether natural language interfaces are useful or whether the help system would be better delivered with some other interface, such as a menu-based system or a formal query language. Some would argue that people prefer menus or formal query languages over natural language interfaces. It has also been suggested that a person who is field-dependent would perform better with a more formal and structured method of interaction.

There have been many programs developed over the last 20 years but few of them work for all, or even a reasonable percentage of input. This is known as the brittleness problem (McKevitt, 1990). This may result from the system not being able to understand the input or

the system's lack of world knowledge. Research is now underway that may help alleviate this problem. Interfaces also fail because users use ill-formed and ungrammatical utterances with strange spelling, syntax and semantics (Guindon *et al.*, 1987; Guindon, 1988). It is very difficult to build a natural-language interface that pleases everyone because, just as accents vary for the spoken word, phrasing varies for the written word. An enormous amount of data must be collected before any natural language processor can even attempt to understand the utterances given to it. The main problem is collecting data on how users will form utterances to a computer in natural language before the system exists. There have been few such empirical studies. The 'Wizard-of-Oz' technique (see Section 2.11.1) has been used to collect data for UNIX (Chin 1984), the objective being to investigate written/interactive dialogue and use these investigations to help build natural-language interfaces.

Natural-language systems are extremely difficult to implement and, in fact, the superiority of human consultation over-relies on the delivery mode — interactive dialogue — rather than information content. It is the cooperative process of "zeroing in" on the problem that is important.

The difficulty of constructing natural language interfaces has so far prevented extensive, realistic testing in the context of online help. There are several reasons to suppose that natural language may not be an appropriate form of interaction for help systems; it is ambiguous and under-determining. Ringle and Halstead-Nusaloch (1989) and Guindon (1987 and 1988) have found evidence to suggest that restricted human-computer consultation dialogues may be a better way to proceed. Borenstein (1985) compared several access methods in a single interface called ACRONYM and found other methods to be more successful than a natural language interface to the help system.

### 3.2.3 Presentation issues

Access to the help information is important but so is the manner in which it is presented on the screen. The simplest and most common method of presenting help information is simply to throw it all on to the screen with no regard for what was there, or how much information the user can view at once. Often too much information is presented at once and it simply scrolls off the top of the screen.

Technological advances have allowed modern systems to use multiple windows. This allows the user to maintain separate windows for the work context and the help information. It is also now plausible to include pictures, animation or even videotapes as part of the help message.

The text itself may either be retrieved verbatim from some data structure that contains it, or it may be generated by a natural language composition mechanism action on an underlying knowledge representation. Some researchers take this approach (Rich 1982; Genesereth 1977), however, in the domain of online help it is not the primary goal.

It is also necessary to look at the quality with which the text is presented. It is important that the help text be: readable, well-organised, formatted and headed, and provide reasonable sized chunks of information.

### 3.3 Online help research

Most guidelines for user-oriented software design include recommendations concerning user support, online assistance or online help (Smith & Mosier, 1986). Literature concerned with online help mainly describes help features and the problems of implementation (Sondheimer & Relles, 1982). Currently, help systems are usually custom-designed and implemented with a great deal of effort. They are specific to the application for which they are providing help. This has led to a spate of different help schemes and delivery techniques. These range from providing a manual for users to browse and find the information they need for themselves to providing context-sensitive help and delivering information which relates to each user's current actions. The latter technique assumes that when users request help they need help with what ever they are currently doing. They also assume there will be a close correlation between the documentation and the system. This is not always the case.

There is little empirical knowledge about the effectiveness of different forms of online help, and there are discrepancies in the few results which have been reported. On the one hand, experiments *"show that users without prior computer experience do poorly with online aids"* (Houghton, 1984). On the other hand Mager's data (1983) indicates that people using a well-designed online help system *"completed the computer task in less time, with greater accuracy, and with better resulting attitudes"*. Although online help has been implemented in nearly all software, it is usually

<i>Method</i>	<i>Researcher</i>
On-line manuals	Cohill & Williges, 1985; Dunsmore, 1980; Relles <i>et al.</i> , 1979
Menus	Cherry <i>et al.</i> , 1989; Borenstein, 1985
Minimal manuals	Gong & Elkerton, 1990; Carroll <i>et al.</i> , 1986
Keyword help	Borenstein, 1985; Houghton, 1984; Girill, 1983; Magers, 1983
Query in depth	Fenchel & Estrin, 1982; Mason & Thomas, 1984
Context-sensitive help	Moll & Fischbacher, 1989; Grimm <i>et al.</i> , 1987; Fenchel & Estrin, 1982; Magers, 1983; Borenstein, 1985
Adaptive prompting	Mason & Thomas, 1984; Mason, 1986
Simulations of user's task	Tuck & Olsen, 1990; Grignetti <i>et al.</i> , 1975; Magers, 1983
Task-orientated help	Moll & Fischbacher, 1989; Carroll <i>et al.</i> , 1986; Finin, 1983; Magers, 1983
Window-based help	Borenstein, 1985; Orwick <i>et al.</i> , 1986; Teitelman, 1985; Walker, 1985
Diagrammatic help	Sebrechts <i>et al.</i> , 1983; Shneiderman, 1986
Hypertext-based help	Campagnoni & Ehrlich, 1989;
Natural-language interface	Ringle & Halstead-Nussloch, 1989; Jerrams-Smith, 1989; Borenstein, 1985; Wilensky <i>et al.</i> , 1984
Knowledge-based approaches	Jerrams-Smith, 1989; Aaronson & Carroll, 1987; Fischer <i>et al.</i> , 1985; Hvelplund, 1986; Finin, 1983; Genesereth, 1977

Table 3.1: Different methods of providing online help.

tagged-on at the end with little consideration and Borenstein's criticism is still valid: *"On-line help is a vital part of nearly every computer system of any size, yet it is poorly understood and generally poorly implemented"* (Borenstein, 1985). Instead of carrying out the research required to determine what kinds of online assistance are useful for different classes of users, the literature and developments are largely being technology-driven as more and more sophisticated systems are built (see Table 3.1).

Only a few papers have surveyed the state of the art in online help systems. Sondheimer & Relles (1982) present an excellent survey of the various methods by which help systems can be studied. However, their main concern is with implementation rather than usability issues. Shneiderman (1984) surveys some relevant experimental results. Houghton (1984) purports to survey the area; the paper is brief and includes other topics such as error messages and prompting. The following sections describe some of the major research which investigates the usability of different approaches to providing online help.

### **3.3.1 Online versus written documentation**

Dunsmore (1980) compared the performance of non-computer programmers using an information retrieval system. The subjects were divided into three groups: the first received a brief written summary of the capabilities of the system, commands and available data; the second received the summary and two pages of more detailed online documentation and the third received the summary and the two additional pages in a written form. The results showed that the third group performed best and the second group worse. Dunsmore concluded that valuable time was taken away from problem solving and that important information may have been lost in the process of requesting information.

### **3.3.2 General improvements**

Magers (1983) tested an improved version of the VAX/VMS operating system using 30 computer novices in a file manipulation task. The improvements included providing a help key, providing context sensitive help, more flexible syntax of help commands, additional tutorial information, elimination of computer jargon, use of examples, dictionary of synonyms, paging of information

<i>Condition</i>	<i>Expert (secs)</i>	<i>Novice (secs)</i>
Standard UNIX	168	167
Simulated natural language	not reported	123
Alternative text	116	115
ACRONYM	139	103
Human Tutor	103	60

Table 3.2: Summary of Borenstein's results (1985)

and rewriting help messages to be task-orientated. The results show that the improvements had a dramatic effect on performance in the task. The users of the improved system completed the task faster, used more commands, made fewer errors, used help more often and asked less questions. This study shows that system redesign can be profitable, however from this study it is not possible to assess whether it was any one individual or a specific combination of improvements which caused the improvements in performance.

### 3.3.3 Context sensitive help

Borenstein (1985) produced a context-sensitive help system for UNIX called ACRONYM. He also provided alternative text to that provided by the man command. He compared standard UNIX help (via man) with alternative texts, ACRONYM, simulated natural language and a human tutor. It must be stressed that as well as providing context sensitive help, ACRONYM had several other access mechanisms available to the user; keyword access and a menu. Although Borenstein separated the effects due to the different texts he did not explore the effectiveness or pattern of usage of the different access mechanisms which ACRONYM made available. He used two groups (experienced users and users with no experience). The groups were asked to complete a set of 24 tasks. The mean task completion time was recorded (see Table 3.2). There were large standard deviations associated with each mean. Borenstein describes this variation between subjects as "a common problem in studies of this kind" and chose an analysis which leaves fewer significant differences visible rather than trying to find the cause of the variation. Borenstein's results indicate that simply improving the quality of the help content relative to the standard messages produces better results than ACRONYM. The help provided by the human tutor was

superior to the online help especially for novices. A criticism of the Borenstein study is that only one measure of performance is presented, mean task completion time, which can be affected by the number of help requests and the time taken to find the appropriate help and the number of tasks not completed.

Houghton (1984) reports on an experiment carried out by Ralles in which Ralles compared a single level help system with a sophisticated, multilevel, context-dependent help system. One interesting observation is that any online aid adversely affected the performance of inexperienced users. Perhaps this resulted from their lack of control of the system in general. On the other hand, experienced users performed better when provided with online aids and preferred the more sophisticated system.

#### 3.3.4 Amount of help required

Barnard et al. (1982) set up an experimental task where subjects were required to use a word processor to edit a distorted proverb. They used 48 computer-naive subjects in 4 independent groups. The actual aim of the experiment was to assess the effects of different strategies for command names. The commands could either be named specifically, for instance *front*, *insert* or generally, for instance *move*, *add*. There were two levels of help available to the subjects, either a command name menu or the command name plus a description of the task it carried out. The investigators found that subjects using the general command names used the descriptive help information more often and after a shorter period than those using the specific command names. An interesting finding is that memory for command names was poorer for subjects who had used the general commands in spite of greater exposure to help information. This may result from the fact that asking for help is actually adopting a passive cognitive strategy for learning which leads to less efficient retention of information than spending time actively considering the options. This research serves to illustrate the point that there were individual variations which had a large effect on the user and effectiveness of the help system which may have influenced the results.

The investigators further analysed their data by looking at the cognitive failure score of subjects. This score is obtained from a questionnaire which asks subjects about how often they forget things, such as names or appointments. The analysis shows that subjects with high cognitive



failure scores are less likely to consult the descriptive help level. This may be because they are more impulsive and less likely to take full advantage of the help available to them. Alternatively, it may be attributed to those subjects adopting a more active learning strategy, which leads to a constructive mapping between command names and their meanings.

This study was followed up in 1984 by Hammond *et al.* They carried out the same experiment, but this time did not find significant differences between the groups using the different command naming techniques. Instead, they found wide individual differences in the patterns of help usage. In this case the data from 5 of the 48 subjects dominated the group data. The help calls from these 5 subjects alone accounted for 60% of those issues. These calls were mainly to find the command name. Other individuals hardly used the help facility, instead they adopted the strategy of deliberately making errors to receive help information rather than specifically requesting help. The researchers did not follow up on these experiments to explore the reasons for these differences in behaviour. This illustrates the effects that individual differences can have on experimental results.

Gilfoin (1980) investigated the effects of help referrals on user performance as subjects become more familiar with a system. Subjects could either use menu or command mode. The results show that subject's use of help decreases as they become more familiar with a system, but when they change from menu to command mode there is a sharp increase in the number of help requests. This illustrates the point that in large systems, learning is an ongoing process and as new situations are encountered more help information and perhaps different kinds of information must be accessed.

### 3.3.5 Adaptive help

Mason and Thomas (1984) provide a prototype for an adaptable online help system for UNIX. Users requesting information from the online manual (via the `man` command) are provided with differing amounts of information. This information contains more extensive and sophisticated material for users classified as more "expert" or system-proficient. Their system does not take into account the context of the help request. Also, in the process of this experiment, Mason & Thomas not only provided an adaptive system but improved the quality of the information provided, without making the same improvement to the basic system they were comparing it

with. Therefore it is impossible to assess the true value of the adaptive aspect of the help system. Mason & Thomas recorded the use of the basic system and the adaptive system for 85 users but found no significant differences between the two systems.

### 3.3.6 Active help

Cohill & Williges (1985) studied three variables: user- or system-initiated help, user or system selection of topics and online or offline presentation of information. They used 9 computer-naive users in 8 groups to cover all possible combinations of the three variables. Their results suggest that user-initiated and user-selected groups did better than the system groups and that off line presentation was best. However this last result may be due to poor online presentation such as not allowing the user to maintain their work context.

Carroll and Aaronson (1988) carried out a study of active help and reported on the many usability problems they found. They used the "Wizard-of-Oz" technique (see Section 2.11.1) to monitor users' terminal activity and provide error recovery when necessary. The help can only be initiated actively by the system. If the help given is inadequate, all the users can do is ignore it. Only a further error will trigger assistance again. The "wizard" provided two types of information: how-it-works and how-to-do-it. Subjects were divided into two groups, each receiving one of the types of information.

They reported the following major problems with providing active help:

- Users have their own unique personal histories that can shape their expectations and goals. This causes them to focus their attention on a subset of the stimuli presented to them depending on their current analysis of the situation (Mack et al., 1983). Thus users can always pre-empt and distort any advice given to them which may lead to further errors in the process of recovery.
- Timing of presentation of information was crucial. If the system is the slightest bit slow in delivering a piece of information and the user has already moved on to a new task, then the user may become very frustrated with the system. It is very difficult to keep pace with the user's current situation.

- The users often expect too much. Even the smallest mistake on the part of the system leads to mistrust which is very difficult to overcome. The users start to expect accurate interpretations of their intentions no matter how loosely they express themselves.
- An active help system may actually affect the learning process; some subjects were observed persistently making errors so that the help system would intervene and tell them what to do. In this situation the users had learnt how to get help rather than learning the system.

Other problems with this system are as follows:

- When the user returns to the task, the help window is removed and in the case of a large number of procedural steps, information may be forgotten and cannot simply be recalled by the user. The window should be visible during task execution.
- The users have no control over the presentation of help information. It cannot be explicitly called and only errors will initiate it. This may be disliked by field-independent users who prefer to be in control.
- The paper does not discuss the subjects' overall reaction to the system or compare performance between the two groups.
- There were only four subjects in each group, and the experience level of subjects within the groups varied extensively. There was no control group who only had the software's standard static help available to them.

### **3.3.7 Type of information**

Carroll and Aaronson (1988) provided two types of information in their study of online help. The first, how-it-works information, is strictly a model-based explanation of how the system works, from which users are expected to infer the procedure for how-to-do-it. On occasions, users were not able to do this. This was due, in part, to inconsistencies within the system for which help was being given. The other, how-to-do-it, information provided the procedures for what the users had to do. This method also had problems: if any step was not explicitly stated, the users would either miss it out, or might infer the wrong step. On occasions users, receiving this kind of information, voiced that they would like to know why they had to follow the given procedure.

The two types of help information both have different strengths and deficiencies and may in fact complement each other.

### 3.3.6 Natural language

Chia (1984) carried out an experiment to look at the way people used natural language when interacting with a computer using a system known as the Unix Consultant or "UC". At this time UC (Wilensky *et al.* 1984) could not carry on a simple conversation with users and so the 'Wizard-of-Oz' approach was used. Two groups of intermediate-experienced users were set up. One group were told they were communicating with UC and the other group were told they were communicating with real people. The scripts generated by the users when carrying out a set of tasks which were new to them were compared. The results discuss the type of input made, but not its consequences for providing online help. One interesting finding was that the group who thought they were talking to other people relied on context twice as much as the group who thought they were talking to a computer program.

This paper illustrates the difficulty of using experience level to classify users. The experimenter classified all the users as being intermediate users, but they classified themselves from beginners to experts. Although it was stated that there were 6 subjects in each group, in one group two people worked together. Such an experimental design is flawed as working in a team may lead to a significantly different interaction.

Borenstein (1985) (see Table 3.2) suggests that natural language may not necessarily yield the quickest interaction for novice users of a help system and found other mechanisms to be more successful.

## 3.4 Summary of research findings

In reviewing previous studies of online help, it is difficult to determine what it is that makes an online help system usable and effective. There are a number of research questions which remain unanswered:

- Are there differences in users' attitude towards, and performance with active and passive online help?

- Do users prefer or perform better with a simple or "intelligent" help system?
- How much help information should be generated and should that amount vary between individuals?
- What type of help information should be supplied?
- What form of language should be used to present help information?
- Can an online help system accommodate a wide range of individual patterns of usage?

### 3.5 Intelligent help

A further change in the provision of online help is illustrated in intelligent help systems. Several investigators have analysed naturally occurring help interactions between humans and this has been used as a model for an intelligent help system (Aaronson & Carroll, 1987; Coombs & Alty, 1980; McKendree & Carroll, 1986; Pollack, 1985). This work has identified a variety of ways in which people ask other people for advice and how advice is given. From this, three aspects of providing intelligent help can be identified:

- Providing a natural language interface to the help system (access mechanism);
- Adapting the information to the individual by maintaining a model of the user and tracking user goals and current context (adaptivity);
- Determining when the user requires assistance and intervening when necessary without an explicit request from the user (access initiative).

These three aspects are rarely treated as separate entities which may affect the usability of the help system. The issues of natural language access and access initiative form part of the subject matter of this thesis, but the issue of adaptivity is not discussed (for review see Coventry, 1990). They are usually all used to improve the help system and thus it is not clear if any one aspect affects usability specifically. The reasons for the failure or success of these systems have never been fully analysed. This is due, in part, to the difficulty of analysing usability problems of such facilities when, for the most part, they only exist as demonstration systems.

No evidence has been provided to suggest that the intelligent approach and its components actually gain anything over more traditional approaches but instead authors simply accept it as being obvious. There are yet no complete, working natural language interfaces to online help. There are also no articles discussing real use of a complete intelligent system. For instance, MACSYMA (Genesereth 1977) is described as a nearly completed and untested system. Activist (Fischer et al. 1984) was also never tested with real users. These systems are more concerned with breaking new ground in implementation issues than in the usefulness of their approach.

## **3.6 Standard UNIX help facilities**

### **3.6.1 The man command**

The main method of obtaining help with the UNIX operating system is commonly referred to as the manual. This is basically an online version of the printed manual. It usually has eight sections, each of which contains a different category of information. These categories include, general information, system calls and subroutine calls. Each section is divided into pages and each page is dedicated to a single item (such as a command or sub-routine) or group of closely related items. However, the term 'page' is a historical synonym for 'entry' and bears no resemblance to the amount of information that may be displayed as a result of issuing the command. The page is divided into the following major sections:

- **NAME** : this is an acronym for a command or subroutine and is followed by a short descriptive phrase, eg. "awk—pattern-directed scanning and processing language".
- **SYNOPSIS** : this gives a very short and stylised notation for the correct command syntax, eg. "awk [-Fs] [ prog ] [ file ] ....
- **DESCRIPTION** : this is the longest section and gives further details of the command and elaborates on all the options available with the command.
- **OTHERS** : there are various other sections containing information about related commands, odd features or bugs and any other information deemed useful by the creator of the text.

The manual pages are accessed by typing `man name` where `name` is the exact technical term for the command or subroutine. To overcome this shortcoming, some versions include the `man -k` keyword option for the `man` command which provides a list of all the commands which have the keyword in their full descriptive names and then allows the user to select the appropriate technical term with which to reference the `man` command. Some versions of UNIX also provide the `whatis` command which gives a one-line description of what the command does. Its usage persists because from a developer's perspective, it is easy to change or add entries should the functionality of the system change. There are a variety of problems associated with the UNIX implementation. These include:

- The users must know the command name before help can be accessed;
- The depth of information provided is the same for all users;
- The information is pitched at too high a technical level for many users;
- Information as to how the system works is entangled with information on how they should use the command;
- Too much information is provided at one time and so before a user can make effective use of the manual pages they must be able to either control the screen output or redirect the output of the command;
- The user may lose sight of their work;
- There are no cues to assist the user's conceptualisation of the structure of the manual or the structure of the system;
- There is a lack of indexing facilities in some versions;
- Poor keyword search;
- Poor facilities for browsing;
- very few examples of how and when to use the commands are given;
- the entries are definitions, or statements of functions, rather than explanations or guides as to what they mean.

**3.6.2 The help command**

If the help command is issued without arguments, a menu system is invoked which has 4 categories (although not universally implemented). Each of these categories is also a UNIX command and can be issued in their own right and arguments can be given to facilitate direct access to the required piece of information if the user knows what to look for.

- **STARTER** : lists the commands and technical terms a beginner should learn first. 14 command names are listed and 10 technical terms. It also lists relevant documents and training centres. This section is menu driven.
- **USAGE** : retrieves a command description and usage examples. It can be used in one of two ways. If no arguments are given then a menu system is invoked with a screen prompt for the name of the command. At this point the user is also able to retrieve a list of commands supported or exit back to the shell. After a command is selected the user can then decide to view either a description of what the command can do, examples of typical usage or a description of options. This can be done with menu or command line options. This command provides information for 42 commands.
- **LOCATE** : this allows the user to access command names and related information through the use of keywords. However it is difficult to get to some command names (even if you know their names) for instance, the keyword " print " will find the cat command but " type, list or contents" will not. This command can be driven by a menu system or arguments can be given at the command line.
- **GLOSSARY** : this provides definitions of UNIX technical terms. The menu can be used to select the technical term, or the technical term can be given on the command line to allow direct access to the menu system at that point.



### 3.7 "Intelligent" approaches to the UNIX help problem

#### 3.7.1 AQUA

AQUA (Quilici *et al.*, 1986) is a passive help system based on story understanding techniques. It uses a set of heuristics to generate an English language solution to an English language description of the problem. In some of the examples given in the paper, the reply does not always include any indication of the appropriate command to use. It takes three to five minutes real time to find a solution to a limited number of problems. No indication was given of any user testing.

#### 3.7.2 The SINIX consultant (SC)

The SINIX<sup>1</sup> Consultant (Hecking, 1988; Kemke, 1987) is a help system for SINIX users which allows the user to ask questions in German about concepts and commands for the Sinix operating system. It has both active and passive capabilities. In describing its capabilities, Kemke (1987) says that the consultant is "supposed to" answer natural language queries. A few of examples are given to demonstrate the functionality of the consultant. The system is described as catering for a number of different types of queries as well as having plan recognition capabilities, and will suggest better plans, using fewer commands if the user is interacting inefficiently. However, no user trials were discussed and so what the system actually does remains unclear.

#### 3.7.3 The UNIX Consultant (UC) Project

UC is described as a computer program which answers English queries about the UNIX operating system. In Wilensky *et al.* (1984) the UC system is described as, "... a natural-language help facility that allows new users to learn operating systems conventions in a relatively painless way". In Wilensky *et al.* (1988), after many modifications to UC it is described as an "intelligent natural language interface that allows naive users to learn about the UNIX operating system". The intention is to create a consultation system, not a natural language front end to an operating system. It is intended to help the user learn how to use an existing interface, not replace it. This is seen as more useful as it means that even if the system cannot deal with every task put

<sup>1</sup>SINIX is a version of UNIX produced by Siemens AG.

to it, it can still serve some useful function. It is also said that UC can handle requests in a variety of forms. Also, it is described as having a robust analyser, "... which almost never has a 'hard' failure and which has the ability to handle most elliptical constructions in context". The authors claim that the UC has extensible knowledge bases both of facts about UNIX and about the English language and a mechanism to make sense out of ill-formed inputs. It is also described as having a component which enables UC to learn new vocabulary and new facts about UNIX. A number of examples are given to illustrate the types of dialogues in which UC can take part. It is stated that, "UC is capable of generating the answers shown below [in an example] in response to most forms of the request users might supply". UC also contains a component that builds a model of the user's state with respect to UNIX. In Wilemsky *et al.* (1984) it is pointed out that the system has not been user-tested. The reason given for this is because the knowledge base was not large enough and that the authors were unsure whether the types of questions UC is designed to answer are the ones that users will actually want to ask. Their intention was to modify the system as a result of any future user testing. It would perhaps be better to attempt to find the user requirements for the system before beginning to develop the system.

### 3.8 An alternative approach

As far as UNIX is concerned, the major development effort to improve the standard of online help is directed at finding an "intelligent solution". However, there is little evidence to suggest that this effort will be of genuine value to the users as very little research has been carried out to establish users' reactions to and performance with such systems. However, the work by Magers (1983) on VAX/VMS, Borenstein (1985) and the ASSIST menu system described earlier, suggests that it may be possible to make significant advances in the usability of online help without requiring to generate an intelligent solution. Another example of the 'nonintelligent' approach is explored in SUPERMAN II.

#### 3.8.1 SUPERMAN II

McDonald & Schvaneveldt (1987) built on previous work to develop an interactive documentation guide called SUPERMAN II. This is a passive system, designed to improve access to the infor-

mation contained in the UNIX online manual. The basis of this system is a network of paths of related topics. The users are guided along these paths. The paths are based on empirically-derived representations of experienced users' cognitive models. The system also has multiple levels of abstraction and provides users who have used other operating systems with a 'bridge' to transfer this knowledge to UNIX. The system is of little benefit if the user does not know the name of a command; however, users may be able to find the command if they know a related command.

### **3.9 Conclusions**

This chapter has reviewed some of the research that is relevant to an online help which attempts to assist and instruct users on a computer-based task. The review reveals that many of the efforts to construct such systems to aid the user have met with only mixed success. Many help dialogues can only assist the user on the current computer-based task, while other instructional dialogues can only help users acquire knowledge which improves long-term performance. In fact, many online assistance systems actually fail to improve performance and even when performance enhancements are observed, the characteristics of online assistance which contribute to the improvement often cannot be isolated.

What conclusions can be drawn from the experimental studies? It appears that the effectiveness of a help system can vary with individual differences. It is clear that a well-designed help system can be effective. It is not as clear whether the system or user should have control of initiating the help messages.

The measures of performance used in these studies are important. Total or average time taken is a commonly reported measure, however the number of errors, number of commands used, number of helps required, number of tasks completed could also be measured. In general, all these measures are interdependent but it is possible for a specific feature of the help system to have a specific effect on one of the measures.

Experimental studies are useful, but the correct balance of variables measured must be found. They are particularly useful if used to test a prototype as part of the development cycle. These experiments yield specific answers that can be applied to the design of the final system.

Experimental studies of online help have investigated a number of factors. One important

omission in the experimental studies if the failure to user test the "intelligent" systems. The research reported suggests that the most important feature of a help system, when determining its effectiveness, is the quality of the information provided. In the previous chapter, research on individual differences which indicates that some dimensions of ability and personality appear to affect how well people use computers was discussed. It is suggested that these individual differences should be accommodated by the interface. This includes the help system. Therefore, to design more effective help systems there are at least three things to bear in mind:

- The help content and its presentation;
- The delivery mode (access mechanism and initiative);
- The people requiring assistance.

## Chapter 4

# UNIX

The previous chapters have explored the differences between users and the methods of providing online help. The next stage is to explore the specific environment for which help and to access what users normally do with UNIX.

An operating system is a piece of software which acts as an interface between users and the computer hardware. The objective of the interface is to provide users with an environment to build and execute programs in a convenient manner. A second objective is to make efficient use of the hardware. For most users, the file system is the most visible aspect of the operating system.

UNIX is a commonly-used and well-researched operating system and its command line interface is still the normal means of interaction. According to the definition given by Thimbleby (1980), UNIX is an under-determining system, ie. failing to help or guide the user with sufficient feedback. The collection and analysis of data pertaining to users' behaviour when interacting with the system is necessary if an effective help system is to be provided. UNIX is an appropriate environment for such a study and a variety of research methodologies prove to be appropriate.

### 4.1 The UNIX system structure

UNIX is an interactive system in which users type commands and the system carries them out and displays the response on the screen. It is a widely used multi-tasking and multi-user system, runs on a variety of machines, and is well-described in many publications (Ritchie & Thompson, 1974; Kernighan & Mashey, 1981; Bourne, 1983; Bach, 1986; Morgan & McGilton, 1987; ). It is

a powerful and rich general-purpose environment.

To exploit the communications, data storage and information processing capabilities of computer hardware, application software requires a mechanism which manages hardware resources, accesses files and interacts with users. These duties fall on the operating system: the version used for this study was UNIX System V. This thesis is particularly concerned with the duty of interacting with the users. All operating systems carry out the same function, namely to hold together all the different hardware resources of the computer in such a way that users can do useful work in an efficient manner.

The operating system interacts directly with the hardware, providing common services to programs and insulating them from hardware idiosyncrasies. The system can be viewed as a set of layers. The central core of the operating system is commonly called the *kernel*, emphasising its isolation from user programs. Because the programs are kept independent of the underlying hardware, it is comparatively easy to move them between UNIX systems running on different hardware. Several hundred programs appear in standard system configurations and are known as *commands*. Other application programs can build on top of lower-level programs. Programs such as shells and editors interact with the kernel by invoking a well-defined set of *system calls*. There are about 64 in System V, of which fewer than half are used frequently. All have simple options that provide the programmer with a lot of power. The set of system calls form the body of the kernel. In short, the kernel provides the services upon which all application programs depend and it defines those services.

#### 4.1.1 The user's perspective

##### The file system

This is a hierarchical organisation of the filestore, divided into directories. A directory is a file containing a list of files and directions to where those files can be found. There are different kinds of files, but all are treated as the same by the file system. Users often work within a single "current" directory although resources located elsewhere are generally made available as well. The name of a file is given by a *path name* that describes how to locate a file in the file system hierarchy.

An important feature of UNIX is that there is little distinction made between files containing programs and those containing other things; any file is eligible for execution if the appropriate permission has been set. Permissions are used to protect files, by controlling read, write and execute permission for three classes of users.

From the user's perspective, UNIX seems to treat devices as if they were files. Programs access devices with the same syntax they use when accessing ordinary files. Devices can also be protected by setting access permissions.

### The shell

The shell is the UNIX user interface. The "normal" shell is the command line interpreter that translates the users' requests into actions on the part of the kernel and other utility programs. It should be noted that any program can act as a "shell" to UNIX. The usual form of a request is a command. The first word of the command is taken as the *command name*. The command takes the form of a command name optionally followed by an argument list. The argument list may contain options which modify the standard meaning of the command, or character strings which act as inputs to a command or as the names of files which are to be manipulated in some way. The shell allows three types of commands: commands are executable files which contain object code, a sequence of shell command lines or internal shell commands. Users do not have to be aware of a command's type.

### Building block primitives

The philosophy of UNIX is to provide operating system primitives that enable users to write small programs that can be used as building blocks to build more complex programs. One such building block primitive visible to shell users is the capability tie together resources by *redirecting input and output* between files.

Another such primitive is the pipe. This allows output of one program to be redirected to become the input of another program. The use of pipes frequently makes it unnecessary to create temporary files.

## 4.2 Learning and UNIX

Of all the components of computer systems, the operating system software retains the most mystique (Gooding, 1987). It is a world where commands are cryptic and brevity is seen as a virtue. Operating system writers often appear to have other things to do, such as making efficient and reliable use of the hardware, than to endear themselves to users. Many operating systems do not encourage users to explore. They remain unhelpful, providing inadequate and incomprehensible help and the consequence of getting a command wrong may be too great a risk for users to take. Such factors may discourage even field-independent users from exploring. However, operating systems do have to be complicated, supporting multi-user, multi-tasking and organising resources. Despite this, they need not appear complicated to the user.

Previous research has explored how people learn to use a computer, but has only addressed the problem within a limited domain, typically text editing or e-mail. This domain has a clear end-product and a smaller set of commands available for use. It is also the most well understood by researchers as they actually use it themselves. Such research has identified a need for improved help, a system which is sensitive to the experience level of novice users, for instance by appropriately wording explanations or error messages in terms novice users understand, and providing greater feedback about the result of actions carried out by the user. Such help is more necessary when normal usage of the operating system is of interest.

Some workers have researched the use of UNIX. The following points are of particular concern to this study:

- Very few commands are known to all users, regardless of the similarity of their tasks. Greenberg and Witten (1988) report that less than 1% of commands are shared by 90% of users. Draper (1984) and Anderson et al. (1987) report that about 10 commands fall into this category.
- Command vocabularies are specialised, large numbers of commands are known by very few people. Greenberg and Witten (1988) report that 92%—100% of all commands are known by fewer than 10% of users.
- Command vocabularies vary greatly in size, but vocabulary size is not a good indicator



of expertise (Draper, 1984). Reported vocabulary sizes range from 10 to 236 commands (Sutcliffe & Old, 1984; Draper, 1984; Anderson et al., 1987).

- A small number of commands are used with a high frequency. These commands tend to be concerned with file manipulation. Hanson et al. (1984) reports that 10% of commands account for 90% of those issued.
- Learning UNIX proceeds at a slow, irregular pace. Learning new commands is usually associated with the undertaking of new tasks (Greenberg & Witten, 1988).
- The popular approach to designing command languages is simply for the designer to adopt their favourite single word as the command name, however, this has been shown to result in 80-90% user failure rates (Furnas et al., 1987).
- Changing command names on UNIX is trivial, yet very few people do it (Good et al., 1984).

### 4.3 Suitability of UNIX for interface studies

Why carry out investigations into the use of UNIX, a 20-year-old system with an out-of-date command line interface, which is argued to be full of deficiencies (Norman, 1981). Why not choose a modern graphical, iconic, direct manipulation interface? Studying UNIX is useful for a number of reasons:

#### Generalisation

UNIX is a real system. It is not a "toy" system developed solely for the purpose of experimentation. It is widely used, very powerful and very complex (Kraut, Hanson & Farber, 1983). It is a general purpose computer environment attempting to fulfill many needs; in fact it was not designed for any particular function. Any results gathered from it can be generalised to many other systems. In contrast many graphical interfaces are so customised to particular applications that generalisations are almost impossible to make.

#### An existing body of knowledge

UNIX is appealing to researchers because it has already been studied extensively, thus in

the true spirit of scientific method, previous studies can be replicated and previous results built upon and gradually extended.

#### **A large and diverse user population**

A major obvious advantage of studying UNIX is that large groups of users exist. Until 1981, UNIX was confined mostly to university computing science departments and research institutes. This user community still exists and is useful for ready access to subjects, but the range of users is expanding. No longer are users thought to be computer experts but come from a variety of backgrounds with a variety of jobs to do. They include people learning to program, researchers analysing data, and secretaries using document preparation tools. Yet, little research has been undertaken to assess the usability of UNIX for such users.

#### **The growing acceptance of UNIX as a commercial standard**

Backed by AT&T's desire and effort to establish UNIX System V as a commercial standard, plus marketplace acceptance of UNIX as a vehicle upon which to build and sell application software, the UNIX system has now gained widespread acceptance.

Although graphical interfaces are becoming a vogue for some applications, command line interfaces still prevail in the operating system world. Two examples are VAX/VMS and MS-DOS.

UNIX is now becoming an open system standard on a diversity of machines from mainframes to work stations to personal computers. Even some users of the most famous of the graphic interfaces think they need UNIX, as illustrated by the Macintosh/UNIX fusion.

#### **The lack of full use of UNIX**

The vast majority of users do not use the abundance of power available to them. Why? Is it simply that UNIX is so general purpose that only a few commands are required for a specific job, or that users just do not know of the existence of commands that would make the interaction easier, or even for both of the above reasons.

#### **Difficult to use**

It is well documented that novices find UNIX difficult to use. UNIX is an under-determining

system which often leaves users at a loss on how to proceed. The on-line help commonly available via the man command is unsuitable for most novice users.

#### 4.4 Studying UNIX users

Having described UNIX and determined its viability as a research vehicle, the next stage is to investigate how the users of UNIX would normally interact with the system, and what problems they face. To do this an observational study of normal use of UNIX was carried out.

Over a period of four months, command line data from selected users of the Stirling University Computing Science department's UNIX System V.2 was collected. The start of every login session was noted and all the complete command lines passed to the system were recorded sequentially. From the users' perspective, the monitoring facility was unobtrusive — the modified command interpreter was identical in all visible respects to the standard version. The users were not aware that they were being monitored. Legally, the University is registered under the Data Protection Act. This means that information about the statistical data being collected can be withheld from the subjects. Ethically, not informing the users of the monitoring may be viewed as wrong but as no information was kept with which to identify individual users and knowledge of the monitoring may have changed the users' behaviour, it was deemed a reasonable decision.

An example of a user data log can be found in Appendix A. Shell scripts were then written to analyse these logs. From these logs, the following measures were collected:

- the frequency with which different commands were used
- the order in which different commands were first used by subjects
- the total number of command lines entered
- the total number of login sessions

##### 4.4.1 The subjects

The target group used was 23 third year computing science students. These students had used VAX/VMS for two years and were just beginning to use UNIX. Therefore, they had knowledge

of computers and an operating system, but not specific UNIX knowledge. UNIX was not being formally taught and they were expected to pick it up as they progressed through the semester.

#### 4.4.2 The procedure

The logging of shell command lines entered by a group of novice users was achieved by re-writing parts of the Unix System V Bourne shell source code<sup>1</sup>. The additions to the shell saved each command line as it was entered, and when the process started by the shell to execute the command completed, the command line, the times ('system' CPU time, 'user' CPU time and real time) of execution and a flag denoting 'normal' completion or interruption (for example, by a user sending the 'interrupt' signal to the process) were written to a log file created separately for each shell started by each user. Thus, multiple simultaneous logins were catered for, since each shell created used a separate log file.

Periodically, the set of log files created for each user was collected and analysed using a group of shell and 'awk' scripts written for this purpose (Aho, Weinberger & Kernighan, 1988). These scripts abstracted various information from the raw data, such as total session time (and hence total login time over a semester), number of different commands used, first uses of any particular command, and frequencies of command use. Data collected was processed both for each individual user and for various groups of users.

The only significant limitation of the scheme used was that multiple-line commands had only the first line saved. This was not felt to be a major problem as the users being logged were, in UNIX terms, novices, and it was felt unlikely that they would have any significant use for such complex uses of the shell during the period in which logging took place.

#### 4.4.3 The results

The data was collected over a four month period from a total of 23 third year computing science students. The total number of command lines entered was 45141. 1298 different command lines were recorded as used at least once by at least one user. 75 valid UNIX commands were entered at least once by a single user. The largest number of valid commands recorded for a

<sup>1</sup>The source code cannot be published as part of this thesis as the source code on which it is based is subject to a UNIX source licence non-disclosure agreement.

<i>subject</i>	<i>No. of Sessions</i>	<i>Total Command Lines Entered</i>	<i>Size of Command Set</i>
1	48	997	29
2	35	741	26
3	242	6305	58
4	46	1530	28
5	114	2587	44
6	312	5241	43
7	64	820	32
8	17	226	24
9	9	57	13
10	41	1225	20
11	67	1288	26
12	32	215	11
13	7	89	16
14	56	682	26
15	39	571	27
16	253	4355	46
17	29	378	22
18	130	3385	34
19	71	2518	34
20	2	21	7
21	12	556	24
22	136	4187	47
23	120	7167	51
<b>Total</b>	<b>1882</b>	<b>45141</b>	<b>688</b>

Table 4.1: Data from Command logs

<i>variable</i>	<i>mean</i>	<i>std dev</i>	<i>min</i>	<i>max</i>
sessions	82	84	2	312
lines	1963	2133	21	7167
command set	30	13	7	58

Table 4.2: Summary of command logs

single user was 58. Table 4.1 contains the individuals' actual scores for the number of times they logged in (Session), the number of command lines entered (Lines) and the number of different commands used (set). Table 4.2 contains the summary information derived from the actual scores. Considerable variation was present in the number of command lines entered by individual subjects, the size of individual command sets and the number of login sessions for the monitoring period. Using the Pearson coefficient of correlation it can be shown that there is a significant positive correlation between all three variables ( $r$  is greater than 0.8 for all combinations). Thus despite having similar backgrounds and jobs to do, their use of the system over the four month period varied considerably both in the number of times the system was used and the resultant size of the user's command set. This variation makes it difficult to categorise users according to experience as measured by length of time using the system.

#### 4.4.4 Observations

##### Frequency distribution of commands

Several investigators have examined the frequency of command usage by a user population (Hanson et al. 1984; Kraut et al. 1984; Greenberg & Witten 1988). All studies report results which can be approximated by a Zipf distribution (Zipf 1949). A Zipf distribution has the property that :

- a relatively small number of items are used with high frequency
- a very large number of items are used with low frequency

Users of UNIX have a large number of commands available to them, and yet they use only a small proportion of these with a high frequency. The results of this study followed the same pattern (see Figure 4.1). In this study, although users had over 400 commands available to them, 75% of command lines issued were accounted for by only 12 commands, and 90% of command lines issued were accounted for by 30 commands. The other 10% of commands were accounted for by 4405 command lines. This was further analysed to find that in these command lines were 1268 different command lines. This 10% includes every possible typographical variation of valid commands, user programs, commands valid on other systems, unintentional embedded control characters and other errors as well as some infrequently-used valid commands.

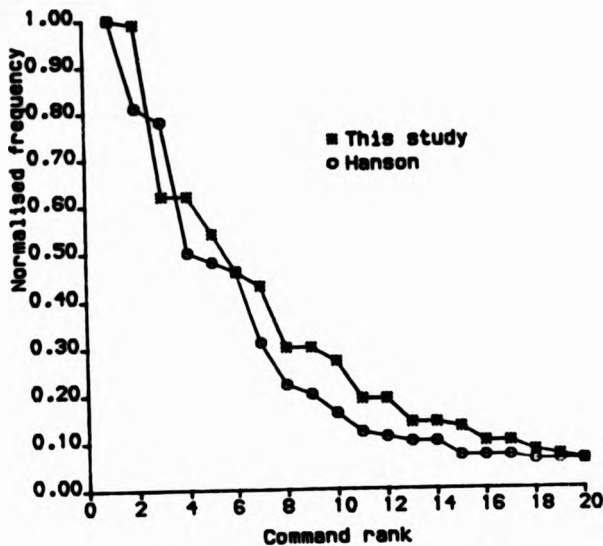


Figure 4.1: Frequency of Command Usage

Sutcliffe & Old (1987) also pursued this line of investigation in a study ranking commands by popularity. They established that the top 20 commands accounted for 73% of the total recorded. The remaining 27% accounted for a further 236 commands. However, their results may be misleading as heavy use of a command by an individual may skew the distribution.

There are many reasons why most commands are used so infrequently:

- redundancy among commands
- difficulty of use
- infrequent need
- the user not being aware of their existence
- the user thinks an alternative method is 'safer'
- the user thinks an alternative method involves less 'mental' effort

The traditional command line and help organisation means that the small number of frequently-used commands are as invisible to the user as the larger set of infrequently-used ones. This is likely to intimidate new users, hinder the learning of the command set, make identification and

Rank	Command	% Used	Normalised	Hanson	% Used	Normalised
1	(null)	12.75	1.00	cd	12.30	1.00
2	ls	12.62	0.99	ls	10.00	0.81
3	vedit	7.94	0.62	cat	9.60	0.78
4	prolog	7.86	0.62		6.20	0.50
5	cd	6.90	0.54	vi	5.90	0.48
6	vi	5.87	0.46	ed	5.60	0.46
7	cat	5.48	0.43	rm	3.80	0.31
8	lp	3.81	0.30	;	2.70	0.22
9	who	3.81	0.30	>	2.50	0.20
10	pascal	3.40	0.27	Mail	2.00	0.16
11	mail	2.43	0.19	nroff	1.50	0.12
12	rm	2.42	0.19	mail	1.40	0.11
13	cp	1.75	0.14	mv	1.20	0.10
14	rogue	1.73	0.14	grep	1.20	0.10
15	emacs	1.64	0.13	col	0.90	0.07
16	man	1.33	0.10	echo	0.90	0.07
17	whois	1.28	0.10	&	0.90	.07
18	date	0.98	0.08	tail	0.70	0.06
19	pas	0.95	0.07	pwd	0.70	0.06
20	chmod	0.72	0.06	awk	0.70	0.06

Table 4.3: The 20 most commonly used commands

location of important commands difficult and lead to input errors. Therefore a help system which covers the basic, frequently-used commands may be useful to novices.

#### Most commonly used commands

Table 4.3 lists the most commonly used commands and the percentage of the interactions with the system they account for. An interesting point to note is that the command most commonly used by third year computing science students is the (null) command. In other words their most common interaction with the system is to send an empty command line. This behaviour may result from a number of causes, for instance, clearing the screen, not knowing what to do next, to see if the previous command is finished. No command accounts for more than 13 percent of the interactions. The next most used commands are `ls`, `vedit` and `prolog`. This is expected as their main reason for using the system was to complete two assignments, one a Pascal program and one a Prolog program.

Hanson *et al.* (1984) collected command usage data from 170 users whose skill level ranged



<i>Command set</i>	<i>Number of Users</i>
1-10	1
11-20	4
21-30	9
31-40	3
41-50	4
51-60	2

Table 4.4: Size of command sets

from novice to several years experience with their main task being document preparation. When compared with the Hanson data, some differences are found in the rank ordering of commands and the actual commands found to be used. However, the Hanson studies were dominated by more expert users. The comparison serves to illustrate that the commands do not maintain their rank ordering over studies.

#### Individual command sets

While studying the nature of expertise in UNIX, Draper (1984) counted the times a command was invoked by each user. Out of a vocabulary of 570 commands available to the population, only 394 were used at least once, with a maximum of 236 commands known to one user. However, the average vocabulary size was distinctly smaller. Draper (1984) quotes "a slight peak in the lower half (of the distribution) around a vocabulary size of 45". In general, he found very few commands used by all the population and a few more to be shared to some degree.

The current study, in general, supports Draper's findings. There were 75 different valid commands used by this user population. This is out of a command set of approximately 400 commands which were available to them. These commands account for 41185 of the command lines entered which was approximately 92% of the interactions. Table 4.4 shows a fairly even distribution of command set size across the user population with a peak in the range 21-30 commands. However, the size of the command set used by the subjects was considerably smaller than those quoted in Draper (1984). The difference with this study is that the subject group is more homogenous in both experience of the system and normal tasks carried out within the monitoring session.

### Common command set

An important observation comes from examining the extent to which one user's vocabulary overlaps another users. Table 4.5 illustrates how few commands are used by all users, even when they are carrying out similar tasks. 75 unique commands were entered but only three commands (4% of commands observed), `cat`, `ls` and `vi` are used by all users, another five by over 90% of users and six by over 80%. Even though these subjects are using the system for solving the same programming assignments, were taught UNIX together, were working in the same room with the same facilities and had apparently similar levels of expertise, they have surprisingly few commands in common.

Drapier (1984) concluded that commands used is a poor measure of expertise. If those with similar experiences do not have many commands in common, then it may not be the case that the novices command set is a subset of an experts.

### Growth of command sets

In the previous sections, a user's command set is taken to be all the distinct commands used within a fixed period of time. However, it is also interesting to examine the dynamics behind the build up of this set. This includes investigating the uniformity of command acquisition and whether new commands are acquired continually or stop increasing the size of their command sets after some period of time.

Sutcliffe and Old (1987) suggest that the size of a user's command set grows with system usage. They found significant correlation between overall command use and the number of unique commands employed. However they do not actually report command acquisition for individual users. This study also found a significant correlation between the number of sessions, number of command lines and number of unique commands used. The correlation was greater than 0.8 for all combinations of these three measures.

There was no predictable pattern to the order that different subjects began to use different commands. However, within each subjects use of commands some sub-orderings were observed. For instance, subjects were noted to use `ls` before they ever used `cd`, `who` before `whois` and `mkdir` before `rmdir`.

<i>Command</i>	<i>% of Users</i>
cat	100
ls	100
vi	100
(null)	91
cd	91
lp	91
who	91
cp	87
chmod	83
mail	83
man	83
pwd	83
rm	83
help	74
passwd	74
mkdir	70
pascal	65
vedit	61
write	61
mv	56
whois	56
prolog	52
date	48
ps	48
time	48
yacc	48
echo	43
kill	43
logout	43
pas	43
rogue	43
lpstat	39
rmdir	39
cancel	35
ed	35
grep	35
pc	35
cal	30
mesg	30
usage	30

Table 4.5: Percentage of overlap between individual command sets

It appears the users have a small command set and new members are acquired slowly and irregularly. It may be that the set is enlarged when new tasks which require new tools are encountered.

#### Use of the online help facilities

83% of users attempted to use the man command at least once. These accounted for only 1.33% of user interactions. 74% of users typed help but this study was carried out before the UNIX help facility was implemented. Therefore the command was valid for VAX/VMS but not for UNIX. 30% of users attempted to use the usage command. At this point in time using this command simply returned the syntax line from the manual for the command given as an argument to the usage command.

## 4.5 Discussion

This study used a homogeneous group of new UNIX users with similar previous experience and tasks to perform. The results agree with previous studies that have found that few commands are shared between all users, despite the subjects having more or less the same tasks to perform during the monitoring period. Outside a very limited set of commands which were used by the majority of users, it would be difficult to predict from the interaction alone, the commands known by individual users. However, observational evidence alone is insufficient to establish whether or not users have knowledge of commands. It may be the case that users know of other commands yet do not utilise them. The major finding so far is that there are large individual differences in the use of UNIX. Thus it may be the case that the statistics which pool users together into different populations may be confusing: the rank ordering of commands are not maintained over studies or between individuals, and an individual's use of commands may vary from the Zipf distribution derived for the group.

It may also be a mistake to assume that users with similar amounts of experience with systems have similar command sets. Users have small command sets and there are different commands which can be used to achieve the same goals, for instance different editors and compilers. The reasons why users elect to use different methods is unclear, but they do not simply use the most

efficient method.

As far as the use of UNIX help facilities is concerned, 83% of users attempted to use the man command. However, this command was not used with any frequency. This may mean that help was not required by users, but judging by the large number of erroneous command lines entered this was not the case. Alternatively, it may be that users find man difficult to use or the information provided is *confusing* or inappropriate for the users' needs and information is found elsewhere by the users.

## 4.6 Conclusions

Studies involving user interactions with command-based interfaces have been discussed and general results replicated. The major findings are as follows:

- Rank frequency distribution of command usage of a population may not be an appropriate approximation of an individual.
- It is a mistake to assume that users have similar command sets. In fact, individuals have small command sets and even fewer commands in common.
- After an initial burst of activity with the system, commands are added to the set slowly and infrequently.
- There are large individual differences in the way the system is used.
- Neither time spent with a system nor command set size are adequate measures of expertise.

These conclusions tell more about individual differences than similarities. It has been suggested that the extent to which a user has knowledge of the system and the manner in which the system is used may be related to the user's level of Field Dependency. This hypothesis will be further investigated in the remainder of this thesis.

## Chapter 5

### Pilot study

Experimental results are not always generalisable to different situations. Therefore, an analysis of the context in which the help system is to be used must be carried out. This chapter discusses a study which was carried out to assess the nature of the problems facing the users of UNIX whom which a help system was to be designed for. This study carried on from the observational study discussed in Chapter Four. It involved an experiment set up to see the amount and type of help required by users exhibiting different levels of Field Dependency when faced with tasks of varying difficulty.

#### 5.1 Measurement of Field Dependency

The measurement technique utilised by this study is the Embedded Figures Test (EFT), as described and utilised by Witkin and his associates (Witkin *et al.*, 1950, 1973). This is a standard psychological test and is generally administered by trained psychologists. Full details of the procedure can be found in the manual (Witkin *et al.*, 1971).

EFT is a paper-and-pencil test in which the subject is required to find a simple geometric figure (for example, a cross) within a complex coloured, geometric figure. The simple figure appears somewhere but is hidden by being incorporated into the pattern of the larger figure. At no time are these two figures presented to the subject simultaneously. Depending on the structure of the complex figure, detection of the simple figure may be very easy or very difficult. To familiarise the subject with the procedure, a practice trial is given at the beginning of the series. The subject's

task is to identify the simple figure as quickly as possible. There are 12 figures in the test. There is a three-minute time limit for each figure. The score is the mean time measurement. This provides a measure of the extent to which a subject is influenced by context.

To complete the task quickly it is necessary to adopt a field-independent approach. This entails separating the picture into parts, ignoring irrelevant stimuli (whether it be colour or shape) and restructuring according to the problem. If a field-dependent approach is adopted, viewing the picture as a whole and not actively restructuring the information it will be more difficult to identify the simple figure and thus result in a slower task completion time (Witkin *et al.*, 1971).

## 5.2 The first help experiment

From the observational study it is difficult to assess whether attempts to access help were successful and what help was needed. Variation in the use of UNIX was also found. The next stage was to look more closely at the type and amount of help required, and whether this was affected by a user's level of Field Dependency.

As a result of the observational study, a number of tasks were drawn up ranging from those frequently carried out by users to those infrequently carried out but thought to be useful for users to have knowledge of. Subjects were asked to attempt to complete these tasks, in an attempt to investigate the type and amount of help they require to enable them to successfully complete the tasks.

### 5.2.1 Aims

The current experiment was designed to examine the commands known and used by users of UNIX and to explore

- the amount and type of knowledge acquired by users
- the amount and type of help requested by users carrying out familiar and less familiar tasks
- whether these variables are affected by the user's level of Field Dependency.

### 5.2.2 Experimental task

The tasks chosen cover a range of UNIX facilities which were thought to be useful for the normal work of the subject group (third year computing science students). The commands utilised include those issued most by students who were monitored while carrying out their normal work. The tasks ranged in complexity from issuing a single command, issuing the command and an operator, to sequencing commands on a single command line. The functionality of the system covered was as follows:

1. Setting up and protecting an individual's work areas (`passwd`, `chmod`)
2. Communicating with other users (`who`, `write`, `mail`, `msg`)
3. Displaying information (`ls`, `cat`, `pr`, `tee`)
4. Organising information (`mv`, `cd`, `mkdir`, `cp`, `rm`, `rmdir`)
5. File analysis (`wc`, `sort`, `uniq`, `comm`, `grep`)

For full details of the tasks given to subjects, see Appendix B.

### 5.2.3 Providing help

Throughout the experiment, subjects could either ask for help or be interrupted by the experimenter. Subjects were free to ask the experimenter to explain anything at any time. If subjects indicated that they did not know which command to use, they were told the command name; if they said they did not know it, the command and its options were explained to them. A similar situation arose if subjects were unaware of the appropriate option. Subjects also asked for the syntax of a command to be explained, for confirmation that they were doing the task correctly ("will that work?") and for verification that the command they had issued had the required effect ("did that work?", "how do I check it worked?").

Subjects were interrupted if they were about to issue a command that would cause an error from which it would be difficult to recover (appending in the wrong order, deleting the wrong file). They were then asked to explain what they thought the command would do. If this was not sufficient for them to recognise their error, the error was pointed out to them. Subjects were also



interrupted if they had used a sub-optimal method to complete a task (using the `mkdir` command twice instead of giving it two directory names on the same command line). They were asked if they knew another way to complete the task. If they did not, the method was explained; if they did know any optimal method, they were asked why they did not use it.

#### 5.2.4 Method

##### Subjects

A pre-experimental questionnaire was administered to determine the subject's own ratings of their computer expertise. Subjects were asked to attribute to themselves the percentage of commands they knew and to categorise themselves from novice to expert.

The eight subjects were volunteers from a 3rd year computing science course. They had been taught VAX/VMS in first and second year and were then left in the third year to learn UNIX for themselves. When this experiment was carried out subjects had completed at least two programming projects in the UNIX environment. They also used UNIX mail and word processing facilities.

All subjects had completed the same courses and were the same age, 21 years (with the exception of one who was 34) and all were male.

After carrying out the Embedded Figures Test, three subjects were found to be more field-dependent than the other five.

##### The procedure

The study was carried out in three stages. These stages were carried out as follows:

- Pre-experimental questionnaire.
- Experimental session.
- Embedded Figures Test.

The questionnaire was not analysed until after the experimental session was complete. The Embedded Figures Test was not administered until after the experimental session. These decisions were taken to avoid experimental bias in the interpretation of the experimental results.

When subjects volunteered to take part in this study, they were issued with a questionnaire. This was used to gather subjective information about the subjects' computing experience. Subjects also signed up for a 45-minute experimental session. On arriving for the experiment, the procedure was explained to them: It was emphasised that the experiment was not meant as an evaluation of what they knew about UNIX, but rather how they went about completing the tasks. Subjects were told to think aloud and explain to the experimenter if they needed help and what exactly they did not know or understand. The subjects did not read the questions in advance, and carried out each one independently.

The experimental session was videoed for later analysis. For the analysis, command knowledge was classified as follows:

- if the subject expressed lack of knowledge of a command and still did not know of the command after being given its name then the command was considered to be unknown by the subject.
- if the subject knew the basic command to use but not the appropriate option then the option was considered to be unknown.
- if the subject issued or was interrupted before issuing a syntactically incorrect command then the syntax was considered unknown.
- if the subject issued a command or issued the command correctly after being prompted by its name only then the command was considered known.

Help requests were classified as follows:

- verification of whether or not a command had been successful
- "how do I do it?", subjects' expression of lack of knowledge of a particular command or option.
- confirmation before a command was issued of whether it was the correct command or correct syntax.
- syntax of commands, "do I need quotes?", "what order do I put the files in?".

Help need not always be asked for. If the subject was seen to be in difficulty, he was interrupted and given the appropriate help (as listed above). After this session, subjects were asked to return to complete the Embedded Figures Test. All agreed and came back for another session, lasting a maximum of 45 minutes.

### 5.2.5 Results

The results were analysed using both Kendall's Tau and Pearson's Product Moment Correlation Coefficients. However, because of an outlier in the results, Pearson's correlation coefficient is artificially high and so the results of Kendall's Tau are reported. A summary of the results for each subject can be found in Table 5.1.

#### Pre-experimental questionnaire

The pre-experimental questionnaire highlights the futility of groupings based on subjective ratings of expertise. Different subjects attributed different percentage of commands known to different levels of expertise. For subjects who classified themselves as fairly competent, the percentage of commands known which they attributed to themselves ranged from 15% to 70%. For those who classified themselves as competent, the range was from 40% to 50% and for those who classified themselves as novices the range was from 1% to 40%. Yet in the study of command usage only 75 unique commands were observed from the entire group. This accounts for less than 1% of the commands available. This suggests that the students are unaware of the extent of UNIX.

#### Field Dependency

Subjects were scored for the Embedded Figures Test (see Table 5.1). The higher the score, the more field-dependent the subject is. Kendall's Tau was then used to compare these scores with the scores from the experiment of command knowledge, help requests and interrupts. In this analysis, Tau must exceed 0.57 for the 8 subjects for there to be significant agreement between the two orderings of scores at the  $p=0.05$  level.

#### Field Dependency and command knowledge

There is a significant negative correlation between Field Dependency scores and the number

Rank	FD score (secs) (total time/no. of tasks)	Commands	Help	Interruptions	Verifications
1	7.30	17	4	5	3
2	8.70	18	1	5	5
3	12.08	15	5	2	0
4	18.47	15	9	0	3
5	19.25	14	9	1	1
6	26.25	17	8	2	7
7	40.40	12	11	4	1
8	87.50	7	14	4	0

Table 5.1: Summary of scores

of commands known by subjects ( $\text{Tau} = -0.67$ ). Therefore, the more field-independent the subject, the more commands they are likely to know. However the number of commands known by all the subjects is limited.

#### Field Dependency and help

There was a significant correlation ( $\text{Tau} = 0.815$ ) between the total amount of help the subject required to complete the tasks and the Field Dependency scores. There is also a significant correlation ( $\text{Tau} = 0.76$ ) between the Field Dependency scores and the number of times the subject explicitly requested help. This correlation may have arisen from the field-dependent subjects knowing a smaller set of commands and therefore requesting to be told the appropriate command name more often.

It was noted that some subjects asked for verification of the command before issuing it. This was noted by questions such as "Will I use `grep`?" when the command they quoted was the correct command. However, no significant correlation was found between Field Dependency scores and the number of requests for verification.

#### Field Dependency and interrupts

There was no significant agreement between the Field Dependency scores and the number of interrupts required. However there was a difference in the type of interrupts required. The more field-dependent subjects were most often interrupted because they did not recognise when the preconditions for a command had not been met and therefore could not under-

stand why the command was unsuccessful. The more field-independent subjects had to be interrupted to stop them doing the wrong thing, for instance, appending in the wrong order or not making the correct changes when using `chmod`.

#### Small common command set

- There were 7 commands known by all subjects, yet not all subjects used all of these commands during the initial observation period.
- A further 2 commands were known by all subjects but some subjects did not know the options which were required.
- 6 of the 8 subjects knew a further three commands, but two of these commands, `mv` and `chmod` were prone to syntax errors which the subject could not correct.

#### Common syntax errors

There are a few commands which caused syntax problems for most subjects. The potential reasons for these difficulties include

- inconsistency in the use of command options (`chmod` and `msg` do not require a minus sign),
- multiplexing of command names, for instance `mail` can be used to read or send mail and `mv` can be used to rename a file or move it to a different directory.
- generalising from previous use of another system in which an alternative format is correct (eg. MSDOS or VAX/VMS).

#### Interrupts and more efficient interaction

Being interrupted and told of a 'more efficient' way to do things was not sufficient inducement for the subjects to change their behaviour. Most subjects reported that they preferred to do things the way they had been attempting and said that they would not use the efficient method. However, this experiment only allowed subjects to display a change in behaviour if they also generalised the learning that had occurred in one situation to a similar situation. The experiment did not provide the subject with exactly the same situation again. Therefore the evidence that the subject would

not change behaviour is only the subject's opinion and may not prove to be the case if the subject was required to use the command again.

#### Lack of an accurate cognitive model

Subjects did not appear to be aware of the fact that a directory is a file and can be treated accordingly. 6 of the subjects were not aware that the order of a directory listing could be altered, although they were aware of other options. But it should be noted that no subject was aware of the sort command for files. Some subjects also were observed getting lost in the file hierarchy and were not aware of the pwd command to find out where they were. Instead they would list the directory contents. They were also not sure of the exact way to use cd to move around. This was especially true of those who had MSDOS experience who tended to start a pathname with a forward slash which took them to the system root when they thought it would take them to their home directory.

### 5.3 General discussion

#### 5.3.1 Inefficient interaction

Subjects did not always elect to use the most efficient mode of interaction available to them. Desmarais and Pavel (1987) suggest that the use of inefficient methods of interaction indicates that more efficient methods are "*most likely not known*" by the user. This experiment found this statement to be only partly true. Subjects were interrupted if they did not use the most efficient form of the required command and asked if they knew of any other way to carry out the task. If they did not, then the other method was explained to them. In certain situations, including giving mkdir or rm more than one file name, subjects were not aware of a more efficient method. In other cases, such as listing a directory without changing directories or using the mv command instead of cp followed by rm, subjects were aware of the other method but preferred the way they completed the task.

### 5.3.2 Ratings of expertise

Many studies categorise subjects according to a subjective rating of expertise. The results of the questionnaire suggest that such categories are too subjective and ambiguous to be of any real use unless they are precisely and explicitly defined. Draper (1984) also suggests that such groupings are useless as the size of command vocabulary is not an accurate determinant of expertise.

### 5.3.3 Group differences

The differences found in this experiment can be interpreted as supporting the hypothesis that Field Dependency results in a qualitative, process difference rather than a quantitative, end-product difference. Thus differences observed in this current study between the field-dependent and field-independent subjects is suggestive of differences in behaviour (Fowler & Murray, 1987; Fowler et al., 1985; 1987).

Field-dependent subjects are thought to adopt a trial-and-error approach to learning rather than pre-planning. They do not naturally impose a structure on information they are given, instead they rely on external referents. In a command-based interface such as that generally provided by UNIX, field-dependents would apparently learn each command and its syntax, independently of other commands. Their learning also appears to be passive, thus they do not actively seek extra information and as a result prefer a system which provides guidance and is well-determined. This may affect the amount of help requested as the field-dependents may be more likely to simply request information from a human helper than explore the system and try things out.

Field-independent subjects are thought to adopt a more flexible approach to learning, developing a model of the system at an early stage and adopting sophisticated learning strategies and actively investigating the system.

The expected differences are reflected in the results of this experiment. The field-dependent subjects were actually not aware of as many commands as the field-independent subjects. This suggests that the field-dependent subjects have not explored the system to the same extent as the field-independent. But those commands that they did know, were known more accurately than the commands issued by the field-independent subjects. Since all subjects are novices as far as UNIX is concerned, this suggests that the field-independent subjects have not yet developed a

complete or accurate model of the system.

Attempts to construct an accurate model of UNIX may prove futile, and field-independent subjects may have to revert to using different strategies, such as rote learning, or develop more sophisticated strategies to deal with the inconsistencies of syntax and double meanings of commands prevalent in UNIX.

### **5.3.4 Consequences of Field Dependency**

This section discusses some of the features that may be useful to incorporate into a help system to accommodate different levels of Field Dependency. The problems outlined and solutions offered are based on observations of the help requested.

#### **Encouraging exploration**

For field-dependent people, it is necessary to find a way to encourage the user to find out more about the system and increase their command knowledge. For instance, a synonym capability incorporated into a question and answer dialogue which would allow a user to ask questions such as "How do I delete a program"? It could also be achieved by providing a menu-driven help system that the user can explore to find out more about the system.

#### **Protective facilities**

Field-independents may require more 'protective' help facilities which stop the user from carrying out an irreparable mistake. Such a facility has been advocated by Carroll and Carrither's paper entitled "Training Wheels at the User Interface" (1984).

#### **Explanation rather than execution of a command**

The results also suggest a need for the user to have the ability to find out what is going to happen if a command is issued, before actually issuing the command. This should allow the user to change the command line, perhaps as a result of either a syntax or logical error being detected by the system before the command is executed.

#### **Verification of the result of executing a command**

There is also a suggestion that users would like more visible feedback as to the result of a



command, with messages such as "password successfully changed" or "non-interrupt mode on", rather than a silent return to the command prompt.

### **5.3.5 Experimental problems**

#### **Choice of method**

Protocol analysis and being videoed is intrusive and not natural for the subjects. Protocol analysis is a useful exploratory tool, given that the type of information wanted is difficult to extract. This information, such as why did the subject do it that way, is the other way known etc, would not have been accessible by secret monitoring of normal usage.

The type of help asked for of the human expert might not be a true reflection of the type of help that would be requested of a help system. A perhaps more appropriate alternative would have been to use a 'Wizard-of-Oz' setup as described in Chapter 3.

#### **Inter-related tasks**

Due to a lack of insight when designing the experimental tasks, the tasks have been inter-related with some tasks being dependent on the successful outcome of previous tasks. This was an oversight, as what became apparent during the course of the experiment was that a "snowball" effect could occur, which meant that failure on one task left the subject unable to proceed with subsequent tasks.

#### **Lack of subjects**

Only eight subjects took part in this experiment and therefore the results provide pointers to a potential conclusion, rather than any definite conclusions. There is also only a limited range of potential UNIX users represented in the subject group. Computing Science students may not fully reflect the great variability present in the user community and a different user group might yield different results.

#### **Lack of comparison groups**

For this experiment to be complete, the following comparison groups would be required. First, a group of female 3rd year students with the same experience (unfortunately a group which is not readily available), and secondly, a group of more experienced users (both male

and female) to allow a comparison to be made between the strategies used; thirdly, a control group, receiving no additional help to that provided by UNIX. Unfortunately suitable subjects were not available.

#### Stress factors

Subjects were placed in an unnatural setting, completing tasks which to some extent were unfamiliar, while being videoed and with a human helper watching what they were doing. Any or all of these features may have affected the subject. As a result of associated stress, the subject may not have performed as normal and may have used different strategies. Perhaps if a Wizard-of-Oz approach had been used, where the user thought that the help was coming from a help system, different results would have been found.

### 5.4 Conclusions

Although of an exploratory nature, this experiment has added weight to the argument that a difference in levels of Field Dependency does exist and that this difference may have an effect both on the way different people approach learning a complicated system and the type of information they learn. This in turn has consequences on the way a help system should function. For field-dependent people, it should be visible and easily accessible, guiding the user to new information. For field-independent people, it should provide "protective" help, thus stopping them making irrecoverable mistakes while exploring the system.

The two components of such a help system are not mutually exclusive. People fall at different places on a continuum of Field Dependency and thus may require different sorts of balances between the different types of help the system provides. It is not an either/or situation.

The following findings are important to note:

#### Experimental results can be misleading

In this experiment, the subjects are not dealing with a real help system and use of such may lead to different findings. Also the statistic used was that of correlation, there may be any number of other factors that also correlated with the results. Correlation indicates a trend but can not attribute cause.

**Different interpretations can be drawn from the results**

The experiment has also served to highlight the fact that although some users learn more about a system than others, as a whole the amount of the system's functionality known to the average user is small. In this case this results from a system which has a large and varied functionality, the system's double meanings of command names and inconsistency of syntax provides the subjects with difficulties in forming a consistent model and the subjects used tended to learn as little as possible to enable them to get their job done.

Some researchers feel that lack of use of UNIX commands is a poor situation and advocate intelligent help systems as a solution (Hecking 1987). Yet the idea of individual differences contradicts this view as does Draper (1984) suggesting that each user learns what they need to know. Intelligent help systems are based on the idea that an expert has knowledge of a set of commands and that a novice should be driven towards acquiring the same set. The UNIX consultant (Chin 1986) stereotypes users into one of four levels of expertise and tailors its advice accordingly. Desmarais & Pavel (1987) use the observation of a command to infer what other commands are known by the user. These approaches are ill-founded. The experiment illustrates how users' knowledge of individual commands differ. They may know just its name, the basic syntax or associated options. In other words, peoples expertise with different commands vary and there is not a general level of expertise with UNIX as a whole.

## Chapter 6

# Design of the Help System

The word design has been used to describe many different aspects of creating a software system, including the choice of functions for a system and the internal structure of the software. It is useful to separate design activities into categories. One such category is *external design* denoting those activities and concepts related to designing the aspects of a system seen by the user.

The emphasis of this chapter is placed on the external design of the help system and the stages progressed through to reach that design. As well as describing how the system looks to the user, it also describes how the user will interact with the system. This implementation is intended to be straightforward, it is not trying to address any unresolved research issues, such as intelligent help systems, tutors and natural language, the help system is designed to address the issue of access initiative.

### 6.1 Implementation considerations

Although the development process for the help system was similar to the general software development cycle, some steps received different emphasis. The collection of data from user analysis and testing is critical to the development of an effective help system and this received more attention. This means that the prototype and tryout stages of the life cycle are very important. This is where the information needed to make final decisions comes from. The interface to the help system is critical, as is its relation to the program for which it is providing help.

Quality control is also vital in the implementation of a help system. If there are inconsistencies

or any other problems, the user may stop using it. Similarly if the information is not completely accurate and current, the usefulness of the system quickly evaporates. The user must be able to trust the help system; if it ever proves to be wrong the credibility of the help system is seriously reduced.

The implementation of good help is very dependent on the collection of user data. The earlier this data is collected, the easier and more cost effective it will be to incorporate any changes necessary. To this end, a prototyping approach may prove worthwhile.

### **6.1.1 The prototyping life cycle model**

The prototyping life cycle model combines the concepts of the structured life cycle model with some of the principles established in engineering (Peters, 1978). A simplified form of the model can be seen in Figure 6.1. The basic notion is to involve the user early in the development cycle by providing them with an example of what the system will look like, so that user reactions and performance can be assessed. This adopts the philosophy that as long as systems will be built twice, it is as well to design it that way (Brooks, 1975). The term "prototype" refers to a skeleton system that contains those elements of the system needed to enable the designer and the user to get an idea of what the system will be like to use. Feedback can be obtained before the final system is delivered.

Prototyping is a particularly useful technique when requirements are not clearly known before the system is built and used. It can also be used to test the validity of requirements from the user's perspective. Prototyping has been successful for new applications where the formulation of requirements is difficult. The evolutionary approach allows the use of an incomplete system which can be modified once requirements become apparent. The modification could be evolutionary or a 'throw away' prototype can be used which allows the derivation of an improved specification which will form the basis of a new system. There are several benefits of the prototype approach. These are noted by Somerville (1985):

- Misunderstandings between the designers and the users are identified as the functionality is demonstrated;
- Incompletely and/or inconsistently specified requirements are exposed as the prototype is

being developed;

- Difficulty of use or functionality which confuses users can be identified;
- A working, although limited, system is available quickly to demonstrate the feasibility and usefulness of the application;
- The prototype may serve as part of a specification for the development of a system where it is difficult to put in writing the effect the customer wishes to achieve.

From the users' perspective the testing of prototypes is useful for the following reasons:

- The users can evaluate the interface in practice and suggest changes;
- The developer can evaluate users' performance and modify the interface to minimise errors;
- It facilitates experimentation with a number of interfaces so that a decision as to the most appropriate can be made;
- It gives users a real sense of the proposed system and therefore encourages them to think more carefully of the consequences;
- It allows users to identify and feel involved with the system, aiding user acceptance.

Given that prototyping can be used to assist in the validation of requirements, a decision must be made on whether the prototype should evolve into a final system or be scrapped and written afresh (Brooks, 1975).

For the purposes of this research, the prototype is not intended to become the final system. A help system, which demonstrated the functional aspects of the interface was developed with the intention of using it to test the effects of different access initiatives rather than building a complete online help system.

## **6.2 Design aim**

The design aim is to find an appropriate method of providing a supportive interface which ensures that novices can learn about UNIX by exploring its capabilities. Such exploration-based learning

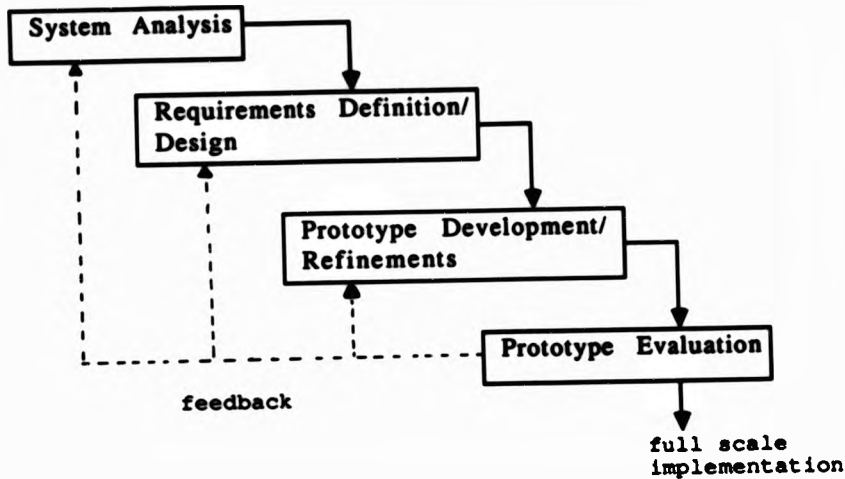


Figure 6.1: The prototyping life cycle (Somerville, 1985)

is believed to be educationally preferable to that of the more usual instruction-based learning (Carroll & Mack, 1984). However, exploration should be enabled in a manner which ensures that the user has confidence in the system. The user has two needs which vary in importance; the first is short term and involves finding the command relevant to the task at hand. The second is long term and involves finding out more about what the system can do so as to improve the user's interactions. The help system was developed to assess the effects of different access initiatives to online help systems on user performance.

### 6.3 Overview

The help system was designed to provide help for the benchmark tasks carried out during the experimental phases (first described in Section 5.2) of this thesis. It is not intended as an end in itself. It is a tool for exploring the effects of different access initiatives on different users.

The help system is written in Microsoft C on a Wang 286 Personal Computer. The PC has a 80286 processor and a 60 Megabyte hard disk. The software allows it to act as a terminal to a UNIX operating system. Therefore the help text files and interaction logging do not take up mainframe memory and the analysis of command lines to provide help does not take up mainframe

processor time. Use can also be made of the colour screen on the PC. The user runs the system as if it were a normal terminal and therefore any learning which takes place is applicable to a basic terminal. There are no noticeable response delays. Figure 6.2 provides an overview of the system.

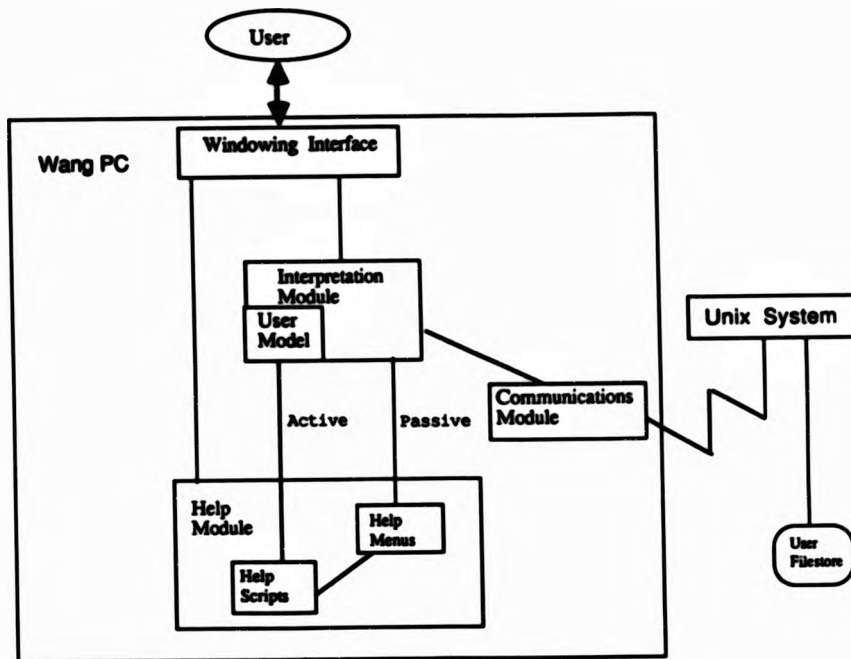


Figure 6.2: Overview of the system

When the user types a command it is interpreted by the interface before being passed to UNIX. The command line is analysed and used to determine the help information to be provided.

The help system can be configured to support to different access initiatives: active or passive. A system which provides both mechanisms was also provided.

For the active system, the initiative lies with the help system to provide help when either an error or inefficient interaction is detected. For the passive system, the initiative should lie totally



with the user.

A menu-driven interface was selected for the passive system. This allowed the initiative to be totally with the user. There were a number of reasons for this decision:

- menus are self-explanatory and require little training,
- the user is not required to know the name of the command to operate the help system,
- a complete working system could be implemented for the thesis,
- if a command-based or natural-language interface was implemented, it may be necessary for the computer to assume some initiative to negotiate with the user the exact command for which help is being requested.

When users made use of the menu-driven help system, they first requested help, then navigated through the menu hierarchy until the exact command and type of help information was located. Therefore, the initiative lay totally with the user.

#### **6.4 Requirements analysis**

The first stage of design is to develop a requirements definition. This definition states what the system should achieve. The requirements of the help system were developed as a result of previous experimental work and the literature survey presented in earlier chapters. These indicated what the goals of the system should be. As far as possible, these have been converted into system requirements. Obviously some goals cannot be precisely expressed in terms of system requirements and so the design was also evaluated with users to discover whether the approach adopted achieves the goals set out. What became apparent in the literature survey is that there are conflicts between some of the requirements, in particular that of access initiative. The help system was developed to investigate this conflict more fully.

**6.4.1 Help system requirements****General requirements**

1. *Should be available both before and during task execution.* This allows the user to browse the information to learn about the system or access information if in difficulty with a particular task (2.6, 2.8).
2. *Should be well-balanced.* This requires an assessment of whether the system should be under user control (passive), system control (active) or mixed-initiative (2.7.1).
3. *Should assist the user in the gradual formation of a correct cognitive model of the system* (2.6 and 2.8)
4. *Should provide both an explanation of how a command works and examples of how to use it* (3.3.7).
5. *Should not pace the user.*
6. *Should allow closure.*
7. *Should allow choice of commands by recognition rather than recall* (3.6).
8. *Should not require any knowledge of either standard UNIX commands or command names to operate the help system* (3.6).
9. *Should not add to the cognitive load of the users*
10. *Should only provide help when it is required* (3.2.1).
11. *Should be easily accessible*
12. *Should be easy to return to work context*
13. *Should guide user to related and possibly new information* (5.3.4).

**Content**

1. *The information provided should be well structured* (3.2.3).

2. *Graded, educative explanations should be provided (3.3.7)*
  - (a) interpretive explanation where the action of the command is defined
  - (b) descriptive explanation of the basic syntax and examples of its use
  - (c) advanced explanation of the options and their use
3. *The information provided should be complete (3.2.3).*
4. *Too much information should not be presented at any one time (3.6).*

**Active requirements**

1. *Should not force an optimal method on the user if the user does not want to change from current method of working (3.2.1).*
2. *Should detect and inform user of inefficient interactions (3.2.1).*
3. *Should give instruction as soon as errors are recognised so that they do not become ingrained (3.2.1).*

**Static requirements**

1. *Should be menu-based (3.2.2).* This allows the users to find help without knowing the particular command name they are looking for.
2. *Should provide categorisation of commands which are meaningful to the user*
3. *Should be no more than 3 levels.* The first level is the category of the command, the second allows choice of specific command and the third access to different types of information for each command.
4. *The breadth of the menu should be determined by the number of commands which belong to each category.*
5. *Should be accessed via labelled function key*

### **6.4.2 Requirements validation**

Once a set of system requirements has been established, these requirements should be tested and changed if necessary. If validation does not occur, errors will propagate into the system design and implementation where they are more expensive to correct. Another role of validation is to assess whether the needs of the user are valid. A user may think that the system is needed to perform certain functions but further thought and analysis may identify that additional or different functions are required.

The user requirements could be met by a number of design alternatives. The next stage will compare two viable alternatives, active and passive help. Both have different problems and benefits associated with them, as described in Chapter 3 and not all design alternatives can meet all the general requirements. The usefulness of the different requirements from the user's perspective can be assessed experimentally by developing a number of help systems which investigate particular issues and testing them with different users. It is clear that the issue of control is the one which has the least agreement as to its validity as a requirement and how it can actually be achieved. The other interesting requirements are that of visibility (for instance, general requirement 1) and availability (general requirement 2). It is necessary to investigate the importance of these requirements as they are difficult to meet with a purely active system. (It is also the case that requirements for an active system might not be met by a passive system and vice versa.)

### **6.5 Choosing commands for the help system**

The subset of UNIX commands supported by the help system is based on the outcome of the observation study, in which the most commonly used commands were identified. These and some other commands thought useful to students are incorporated into the help system. The help system was also designed to support users carrying out the same tasks as described in the first help experiment (Section 5.2). The commands chosen were:

---

cat	cd	chmod	comm	cp
grep	lp	ls	mailx	msg
mkdir	mv	passwd	pr	pwd
rm	rmdir	sort	tee	uniq
wc	who	write		

---

Special operators which can be used to redirect input and output when used in conjunction with some of the commands listed above were also included in the help system. These operators are: |, <, >, and >>.

## 6.6 Categorising commands

Menu categories function like command names. If a menu system is to be implemented, then the commands must be categorised. There are three problems with creating menu hierarchies;

- deciding on a meaningful name for each category,
- deciding what category each command should belong to,
- deciding the order for items within each category.

For the purposes of this thesis the commands were ordered randomly within each category, however alternatives do exist, such as ordering by frequency of use. The categories chosen are:

- general — using a command line system such as UNIX;
- input and output — redirection of input and output;
- directories — organising the filestore;
- files — printing, displaying etc of complete files;
- text file manipulation — rearranging of contents without using an editor;
- ownership and protection — protection of files and access to filestore;
- user-to-user communications — socialising via the computer.

The placement of commands within each category to form the menu structure can be found in Figure 6.3. The categories chosen are similar to those discussed in Chapter 3, and to those used in standard UNIX reference books, with the exception that editing commands are excluded as this would form an entirely separate help system. For each command there is another level of depth to allow the user to access different types of information about the command:

- description of the command and how it fits into the system,
- basic syntax of the command,
- option syntax.

## **6.7 The user interface**

There are two aspects of the interface: the input required by the users and the output given in response.

### **6.7.1 Inputs required**

It is necessary for the users to learn the following interaction procedures:

#### **Using the command list**

- F2 displays the command list,
- any key press when the command list is displayed returns the user to the command line.

#### **Using the help menu**

- F1 gives access to the help menus,
- use either cursor keys to scroll or the number of a selection to move to that menu item, followed by <CR> to make the selection,
- ESC to move back up the menu hierarchy,
- ESC from the top level to return to the command line.

## **6.7.2 Output layout**

### **The work screen**

Figure 6.4 shows how the interface looks to the user when interacting with the system.

### **The command list**

This is a screen which is always available by pressing a labelled function key. It provides the user with a list of the commands covered by the help system and a simple one line description of its basic function (see Figure 6.5).

### **The help system screen**

Figure 6.6 provides an example of how the passive help system appears to the user. Figure 6.7 illustrates how the active help system displays the help information to the users.

## **6.8 Components of the System**

### **6.8.1 Program overview**

The details of the code written does not form a central part of this thesis. What is important is the interface to the user. The code is written in Microsoft C which allows modular development, with separate modules stored in different files. For clarity, the modular breakdown of the program is provided. The system was developed following a modular design with the following modules provided:

#### **Interpretation module**

This module is the control program for the system. It is used to set up the initial screen and links to the UNIX system. The program then loops until the end of a session is signalled. Each time the loop is executed, the following procedures are carried out:

- Accept command line
- If active system provided then interpret command and provide help information if necessary

- send command line to UNIX
- display output returned from UNIX

The interpretation of the command by the active system makes use of two structures. The first stores information about the alternative words which can be used to activate any of the supported UNIX commands. The following information is stored:

- the alternative name,
- the UNIX command the name represents and
- the number of times this alternative has been used.

The second may be considered to be a combination of a model of the user and a model of the system. The structure has an entry for each command supported by the system. This entry contains the following information:

- the command name,
- a list of supported options,
- the name of the help file,
- the position in the help file where the information is stored,
- the number of times the command has been used,
- the number of errors made when using the basic command,
- the number of times the command has been used with options, and
- the number of errors made when using the command with options.

If a passive system is provided then the main control loop can be interrupted by a user request for help. The help module is then executed. The work screen is illustrated in Figure 6.4.

#### **Help module**

This module provides the menu driver for the help system. The menu structure and help information are stored in files which allow the details to be easily updated. When this module is activated, the top level menu is displayed. The user may then traverse the menu by making



a selection at each level until the appropriate information is displayed. An example of how the menu system appears to the user is provided in Figure 6.6. The help scripts can be found in Appendix C and the full menu structure can be found in Figure 6.3.

#### **Windowing module**

The screen is divided into windows. This module provides a library of windowing functions to support the interpretation and help modules when displaying information on the screen within a windows environment.

#### **Communications module**

This module provides the functions required by the interpretation module to interact with UNIX, both to send and receive information.

### **6.3.2 The help scripts**

The help scripts were derived from the information provided in the on-line manual pages, two UNIX reference books (Waite *et al.*, 1983; Morgan & McGilton, 1987). The information provided was reduced to that most relevant to the tasks at hand and reworded to minimise jargon. The number of options displayed for each command was also reduced. It was then divided into the following sections:

- **Description:** gives a description of how the command works and its relation to other commands in the system.
- **Basic syntax:** gives a compact form of the syntax required to issue the command and some examples of how the command will look when issued under different circumstances.
- **Options:** lists the most useful options and gives examples of how to use them both singularly and in conjunction with each other.

The scripts were then given to the Computing Science Department's System Administrator to check for accuracy. The full scripts can be found in Appendix C.

### 6.8.3 Help initiatives

The help system provided two types of help. The first is a static menu system which the user could activate, at leisure, and browse to find the appropriate information. Secondly, active help is initiated by the computer. On the basis of the current command being issued, the computer displays help appropriate to the command being issued.

#### Passive help

Commands presented to this system are sent to UNIX without interference. Users initiate the static help system by pressing the F1 key, which for the purpose of this study was labelled "Help". They can then browse the menu system until they find the command for which they were looking.

The menus have been designed following HCI guidelines (Shneiderman, 1987). There are no more than three levels of depth to the menu system and no more than seven items appearing on each menu. The menu items have task-orientated labels to allow people, who do not know the name of the specific command they are looking for, to find the command name and access help information from it, for example, *list directory contents using ls*. It was hoped that this design would allow users with no knowledge of the command names to find the name by identifying the task they wanted to complete.

The menu hierarchy is listed in Figure 6.3. The first level of the menu system is a general level to allow the user to home in on a particular area of interest, for instance manipulating files or communicating with other users. The second level gives access to the commands appropriate to that grouping, for instance information on the *ls* command will be placed under the directory selection. From here, the user may either be given the required help information directly or be asked to choose a particular type of information relevant to a command; a description, basic syntax, options, error messages or related commands. Selection of a menu item is made by either moving the highlighted bar with the cursor keys over the selection and pressing the return key or typing the number associated with each selection.

**Active help**

This help system provides computer-initiated help by detecting when help is needed and then providing it, rather than waiting for the user to request it. This is particularly important because users do not always ask for help when they need it. For instance, they may not know that they are using the system inefficiently, or they may not be able to formulate a help request in words recognised by the system.

Such an interface should enable novices to complete more tasks successfully. It should also reduce the number of repeated errors resulting by an underlying misconception or lack of knowledge. This help system provides active intervention on the part of the PC. It has a list of the commands for which it can provide help. Each command line is interpreted by the help system. It recognizes the following triggers:

- synonyms for recognized commands from VMS and MSDOS, and other "likely" synonyms;
- some commonly-used alternative names for commands;
- commands not supported by the help system;
- command options not supported by the help system.

A record is kept of the commands used by each user along with a count of the number of times it has been used and whether or not options have been used with it. This allows a decision to be made as to whether a description of the command is given with the basic syntax (command never used), or basic syntax only (command observed before) or syntax with options (options used previously). A record is also kept of alternative names that can be recognised along with a pointer to the appropriate command name. This can be easily updated and command names from other systems included.

In response to a recognised alternative name, the correct syntax is given, a short description and examples of its use. If options are included in the command line, then option examples are given rather than basic examples. Option information is also displayed if a valid command is issued with an invalid option. If the command is not recognized, then the information screen is presented to the user. This gives the user a list of recognized commands and a one line description of what the basic form of each command does.

## 6.9 Summary

This chapter has detailed the active and passive interfaces to the help system. The development environment was not ideal. Code was written in "raw" C, with no User Interface Management System or windowing environment to build on, therefore much of the code written was simply to create an environment in which the help system could be implemented, for instance the windowing module and the communications module. The fact that the code itself resided on a PC rather than on the UNIX system itself meant that the full power of UNIX could not be exploited when providing the active interface. Since the code was written, workstations with X Windows and a full graphical interface have become available in the Department. This would have provided a better working environment. Higher resolution and larger screens would allow separate, non-overlapping help and work windows to be visible at all times.

The next stage in this thesis is to carry out an experimental evaluation (user testing) of the help system. The results of such a user evaluation could result in alterations to the design of the help system.

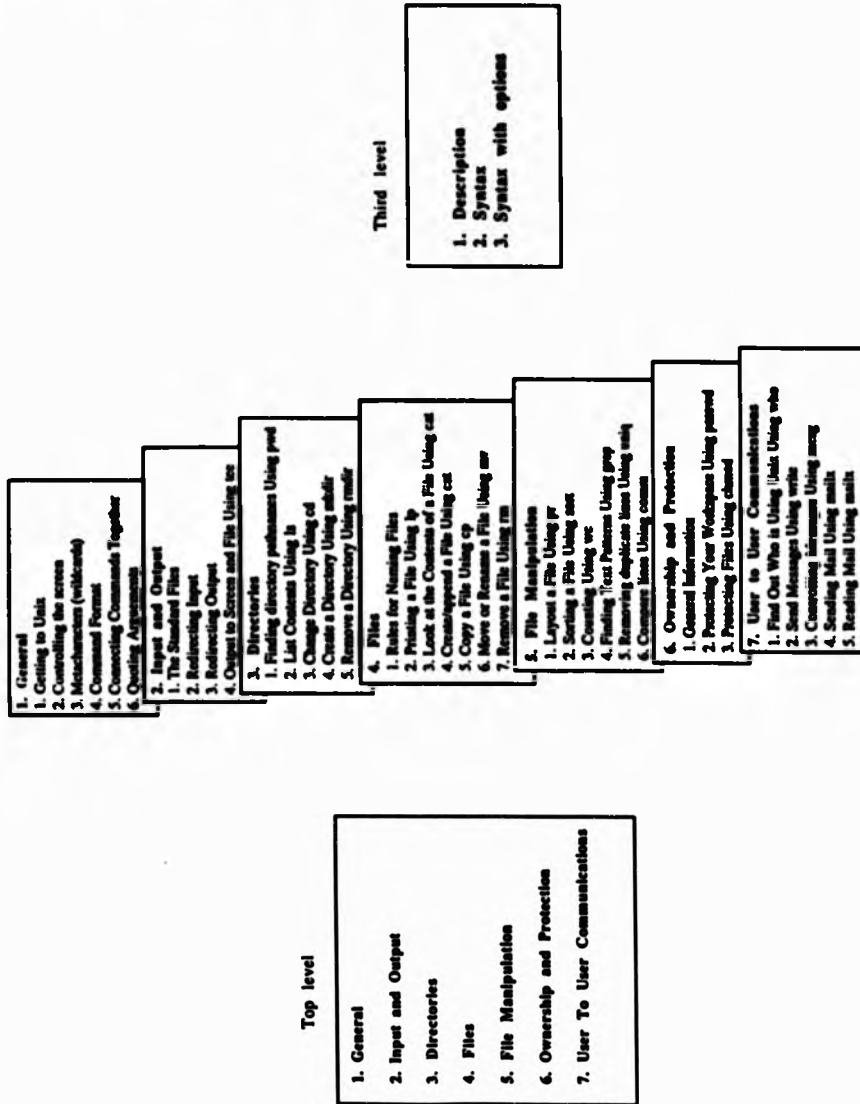


Figure 6.3: Menu structure

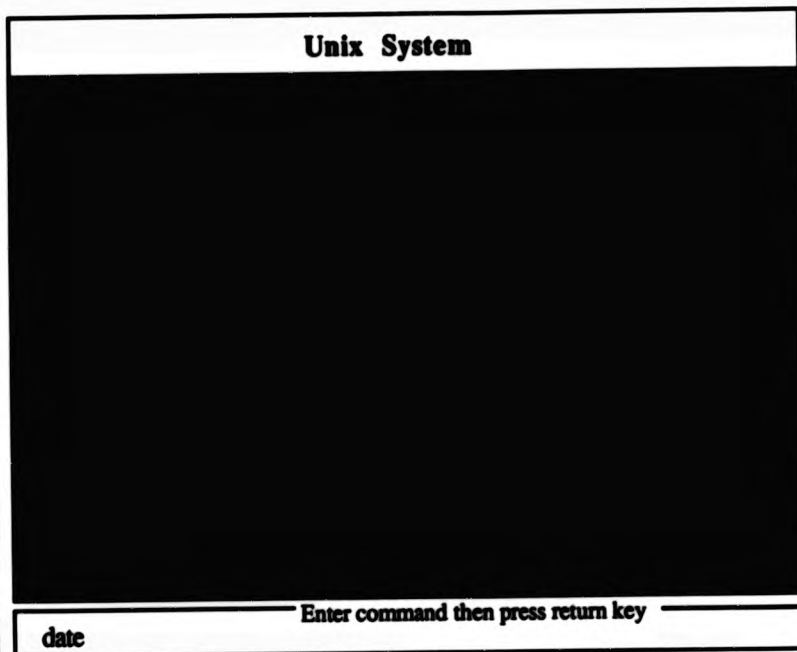


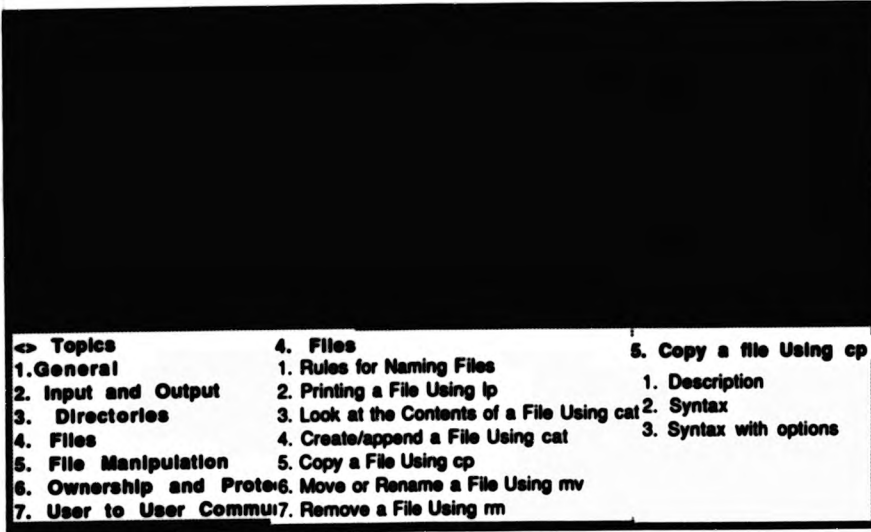
Figure 6.4: The work screen

---

<b>date</b>	give current date and time
<b>who</b>	who is on the system
<b>mesg</b>	permit/stop messages from write
<b>mailx</b>	send/read mail
<b>write</b>	send message to specified user's screen
<b>tee</b>	send output to file and screen
<b>passwd</b>	set/change password
<b>chmod</b>	set permissions on file
<b>cat</b>	display contents of file
<b>cp</b>	make a copy of a file
<b>mv</b>	move or rename a file
<b>rm</b>	remove a file from directory
<b>ls</b>	list contents of a directory
<b>cd</b>	change working directory
<b>mkdir</b>	make a new directory
<b>rmdir</b>	remove an empty directory
<b>pr</b>	format a file for printing
<b>sort</b>	sort contents of a file
<b>wc</b>	count no. of objects in file
<b>grep</b>	search file for pattern
<b>uniq</b>	remove duplicate lines from file
<b>comm</b>	compare lines in sorted files

---

Figure 6.5: The command list screen



<b>&lt;&gt; Topics</b>		
<b>1. General</b>	<b>4. Files</b>	<b>5. Copy a file Using cp</b>
<b>2. Input and Output</b>	1. Rules for Naming Files	1. Description
<b>3. Directories</b>	2. Printing a File Using lp	2. Syntax
<b>4. Files</b>	3. Look at the Contents of a File Using cat	3. Syntax with options
<b>5. File Manipulation</b>	4. Create/append a File Using cat	
<b>6. Ownership and Protection</b>	5. Copy a File Using cp	
<b>7. User to User Communication</b>	6. Move or Rename a File Using mv	
	7. Remove a File Using rm	

Figure 6.6: Passive system screen



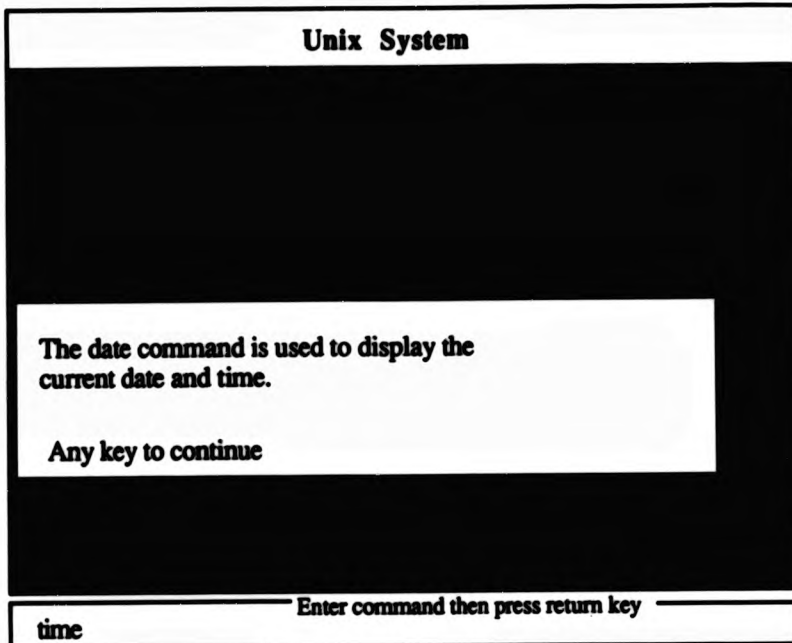


Figure 6.7: Active system screen

## **Chapter 7**

# **Experimental Evaluation**

The help system was evaluated experimentally. The basic purpose of the experiment was to compare the performance of different users of the help system and to assess the effects of individual differences on the usability and acceptance of the help system.

### **7.1 Aims**

The aim of the experimental phase was to evaluate the effect of different types of help on users' ability to complete a set of benchmark tasks. There were three experimental conditions. The first condition was when users had an active help system which remained hidden until the user made an error (system-driven help), the second was when users had a passive help system which could be explored via a menu hierarchy at any point in the interaction (user-driven help), the third is where users had both the passive and active help intervention on the part of the computer (mixed-initiative help). During the experiment, a human helper was also available to the users. The helper's task was to keep the interaction progressing by filling in for any failings of the help provided by the system. The comments made to and questions asked of the expert formed part of the analysis of the help system.

## **7.2 Method**

### **7.2.1 The subjects**

The subjects were drawn from two distinct user communities. The first was formed from second year Computing Science students at the University of Stirling and the other from unemployed women who were training in computing at the Women's Technology Centre in Stirling. Both groups had spent a similar amount of time, approximately three months, developing programs within a UNIX environment and both groups had a minimum of 6 months previous experience with MS-DOS or VAX/VMS.

The women attending the Technology Centre had little or no formal qualifications or training. Therefore, there is a large difference between the backgrounds of the two groups. However, this serves to illustrate the current diversity of the user communities as both groups are real and active users. There are 12 subjects in this group (all female).

The second year students were selected instead of the third years used in the previous studies because of a change in University equipment. Whereas previously the students did not use UNIX until their third year, they are now using UNIX from first year. The students chosen were involved in the transition and used VMS in their first year and in their second year moved on to UNIX. There are 20 subjects in this group, 17 of which are male.

### **7.2.2 Experimental groups**

Subjects were assigned to one of three experimental groups, active help only, passive help only or the joint system. The groups were matched for Field Dependency as much as possible. The procedure for measuring Field Dependency was discussed in Section 2.6. The individual scores for Field Dependency can be found in Appendix D. A problem arose with the group formed from the Technology Centre. At first the management at the Centre would not allow me to carry out any form of assessment on the women as this went against their policy and so I had decided to continue with the next stage as the time they had allotted to spend with me was running out. However, the women themselves were all very keen to do the EFT test and so the management agreed to let me do the test at a later date. So subjects were assigned to experimental groups

at random and their levels of Field Dependency assessed later. All the groups had access to the same help information, the only difference between the groups was the method used to access that information.

**The active group**

There were 10 subjects in this group. Active help is provided under three conditions:

1. If a recognised synonym for a command name is given, then the syntax of the command and examples of its use are presented to the user. When the information is read, the user presses any key to continue.
2. If an unrecognised command is given then a list of valid commands and a one line description of each is presented.
3. If an unrecognised option is given then the syntax and examples of the options is presented.

**The passive group**

There were 11 subjects in this group. Passive help is initiated by the user by pressing a labelled function key (F1). This activates a menu-driven help system which the user can then explore to find the required information.

**The joint group**

There were 11 subjects in this group. This group had both types of help access available to them at all times.

**7.2.3 Experimental design**

The experimental design contains four independent variables, i.e., those deliberately manipulated in the experiment. These are:

1. Field Dependency,
2. gender,
3. training group,

**4. experimental system.**

There are 9 dependent variables measured, i.e., those for which changes were considered to be a consequence of manipulation of the independent variables. These are:

1. The number of help requests (this is the number of times the user initiates the help system) (NHLP).
2. The number of information screen requests (this screen provides a list of the commands for which help is available, and a one line description of the function of the basic form of the command) (NINF).
3. The number of system interrupts (this is the number of times the system initiates help in response to some inappropriate action by the user) (NINT).
4. The number of keystrokes used within the help system (HKYS). It was thought that by counting the number of keypresses while in the help system, the efficiency with which the subject used the help menus and quickness with which the correct information was retrieved could be assessed. It is used as a measure of the usability of the menus.
5. The total time taken to complete the tasks (TME). This includes thinking time, execution time, error time and time spent looking for help information.
6. The number of commands known (but some problems with use), this is the number of command names that were attempted but help information had to be accessed before the command could be successfully completed (CK).
7. The number of commands expert, this is the number of commands which were successfully executed without any help (CE).
8. The number of tasks that were not completed (INCMP). This is an important variable as it affects the scores for the other variables.

**7.2.4 The experimental task**

The benchmark tasks are representative of real tasks an operating system may be used for within a development filestore. They range from those requiring simple and frequently observed commands,

for instance ls to more complicated, rarely observed, but nevertheless useful commands, for instance, tee. The tasks were presented on paper to the subjects. Subjects were asked to work through them in the given order. Each subject worked through the tasks using one experimental system and its use was explained to the subject. The tasks can be found in Appendix B.

### **7.2.5 The system**

Subjects interacted with UNIX V.2 operating system. The interaction was mediated by a PC-based front end, as described in Chapter 6. This provided the three experimental groups, active help only, passive help only or both. There was also a command summary available to all users at all times by pressing the function key, F2. This provides a list of command names and a one line description of what each command is used for.

### **7.2.6 The procedure**

There were three parts to the experimental procedure:

1. Experimental session.
2. Embedded Figures Test.
3. Post-experimental interview.

The Computing Science subjects signed up for an hour long experimental session and both stages were carried out within a single session. This was not possible for the other group and the experiment was carried out over two sessions. On arriving for the experiment, the procedure was explained to them: It was emphasised that the experiment was not meant as an evaluation of what they knew but rather how well the help systems assisted them in the completion of the experimental tasks. Subjects were told to think aloud and explain to the experimenter if they did not understand something and what exactly they did not know or understand. The subjects were not given time to read the tasks in advance, but read each one and then attempted to complete it.

The Computing Science students first took the Embedded Figures Test. Their score from this was used to put them into an experimental group so that the groups could be balanced for Field

Dependency. Each group contains a similar range of Field Dependency scores. Subjects then moved to the computer and the use of the help system was explained, along with an explanation of exactly what their current position was within the UNIX filestore. The subjects were told how to use the help system but could also ask if they forgot during the experiment. The relevant function keys were also pointed out to the subjects.

For the women trainees, it was not possible to measure Field Dependency before the computer tasks were taken and so the women were randomly assigned to the experimental groups.

The experimenter sat beside the subject and encouraged subjects to explain their decisions, what information they were looking for and whether or not the help information was comprehensible and satisfied their needs. The decision to use this method and not to leave the student alone with the tasks was because:

- It maintained a similar experimental setting to that used in the first experiment.
- A system log of the interaction only highlights where the problems lie but not why the problems are occurring.
- It is very difficult to encourage people to think aloud and subjects had to be prodded when they were struggling to explain why they were doing something.
- The tasks were not independent, but led on from each other, therefore a mistake in one may have left the subject unable to continue.

During the interaction with the system, the subjects were videoed so that the comments made by them while using the system could be analysed at a later date. All keystrokes made while completing the tasks were also recorded. This log, along with the protocol analysis provided a complete record of the user's interaction with the system.

After both the tasks and the EFT were completed, the subjects were asked about their impressions of the system. The general questions asked were whether there was anything about the system that the user liked or disliked and whether they could see ways of improving the system.

Variable	Name	Mean	Std Dev	Min	Max	N
Commands known	CK	9.91	4.26	3	17	32
Commands expert	CE	7.06	3.53	1	14	32
Help keys used	HKYS	81.36	36.27	20	163	22
Number of helps	NHLP	10.64	4.24	3	20	22
Number of infos	NINF	3.56	3.11	0	12	32
Number of interrupts	NINT	4.05	2.99	0	11	21
Number of tasks incomplete	INCOMP	3.66	3.75	0	13	32
Time taken (mins)	TME	37.16	13.04	16	65	32

Table 7.1: Summary statistics for dependent variables

### 7.3 The results

Data was analysed using SPSS-X. Correlations between each of the variables were investigated. Separate one-way ANOVAS were performed for each of the independent variables against each of the performance variables. Two-way ANOVAS were used to investigate the interaction between the experimental system and each of the other independent variables for the total amount of help required and the total number of tasks completed.

#### 7.3.1 User statistics

The scores for each subject on each variable can be found in Appendix D. A summary is given in Table 7.1. The summary information for Field Dependency is given in Table 7.3. There is a wide range of scores for each variable. This is an indication of the effect of individual differences on the interaction.

#### 7.3.2 Correlations

Table 7.2 shows the Pearson correlation coefficients found in the data. The following points are of particular interest:



	<i>CK</i>	<i>CE</i>	<i>HKYS</i>	<i>NHLP</i>	<i>NINF</i>	<i>NINT</i>	<i>INCOMP</i>	<i>TME</i>
<i>FD</i>	-.52	-.55	NS	-.39	NS	NS	+.62	-.31
<i>CK</i>		+.90	NS	NS	NS	+.42	-.76	NS
<i>CE</i>			NS	NS	NS	+.43	-.64	NS
<i>HKYS</i>				+.95	NS	NS	-.43	+.75
<i>NHLP</i>					NS	NS	-.48	+.76
<i>NINF</i>						NS	NS	NS
<i>NINT</i>							NS	NS
<i>INCOMP</i>								-.48
<i>TME</i>								

Table 7.2: Table of Pearson Correlation Coefficients

<i>Variable</i>	<i>Name</i>	<i>Mean</i>	<i>Std Dev</i>	<i>Min</i>	<i>Max</i>	<i>N</i>
Field score	FD	29.64	25.33	5.67	102.42	32

Table 7.3: Summary of time taken (secs) to complete the EFT

- Field Dependency was found to have a significant negative correlation with the commands known, commands expert, the number of help requests and time taken for the tasks. Thus the more field-independent the subject, the more commands are likely to be known and be expert with and the more time will be taken and more help requests made.
- Field Dependency was found to have a significant positive correlation with the number of tasks not completed. Thus the more field-dependent the subject, the more tasks were uncompleted. This may explain the previous correlation when field-independent subjects knew more commands but took more time to complete the tasks.
- Field Dependency had no significant correlation with the number of keypresses while in the help system, the number of times the information screen was called, the number of times the subject had to be interrupted.
- Commands known had a significant positive correlation with commands expert, and the number of interrupts. This is an interesting observation: the higher the subject's command knowledge, the more likely they were to be interrupted. Perhaps this results from greater knowledge resulting in greater confidence with the system, and using educated guesses which were not actually appropriate.
- Commands known had a significant negative correlation with the number of tasks not completed. Thus the higher the subject's command knowledge, the less likely the subject was to leave the tasks unfinished.
- Commands known and commands expert had no significant correlation with the number of help requests or the number of help keys used within the help system. Thus if command knowledge is being used as an indication of a subject's level of expertise, then level of expertise does not have a significant effect on the number of times help is requested or the amount of help being accessed. However this may reflect the fact that different help was accessed by those having more command knowledge.
- The number of keypresses while in the help system had a significant correlation with the number of help requests, time taken and time per task. Therefore using the help system

actually increases the time taken to complete the tasks.

- The number of keypresses within the help system and the number of help requests had a significant negative correlation with the number of tasks not completed. Thus the more help that was accessed, the less likely tasks were to be left incomplete.
- The time taken had a significant positive correlation with time per task, however there was a significant negative correlation between the number of tasks not completed and time. Thus decreasing time tends to indicate that more of the tasks were left incomplete.

### 7.3.3 One-way analysis of variance

The statistical design adopted for the 4 independent and 9 dependent variables was to carry out a one-way analysis for variance for each of the independent variables against each of the nine dependent variables.

#### System effects

There was no significant difference between the experimental groups on Field Dependency scores, commands known, commands expert. Therefore the experimental groups can be considered to be well matched.

There was no significant difference between the groups for the number of keystrokes used within the help system and the number of tasks not completed.

An interesting result is found when comparing the number of times help was requested by the experimental groups. There is no significant difference between the subjects in the experimental group which only saw the static system and the experimental group which had both the static and the active system. In other words, the active system was not alleviating the need for help. This arose from the subjects' tendency to verify the command by looking up the help system before starting to execute the task. This may result from normal use of UNIX in which mistakes can be costly.

There was a significant difference ( $p = .0009$ ) between the experimental groups for the number of interrupts. Those subjects who only had the active system and thus had to rely on the interrupts received significantly more interrupts than the subjects who had both the active and static system

available to them. This suggests that use of the passive part of the system was reducing the need for the active part.

#### Field Dependency effects

When scores for the Field Dependency (EFT) test are broken down into two groups using the median a significant difference ( $p = .0000$ ) is found between the field-independent group (scores lower than the median) and the field-dependent group (scores higher than the median) on EFT scores. Although there was a significant negative correlation between the number of help requests and field dependency scores ( $-.39$ ), there was no significant difference between the field groupings in the number of help requests. This shows how easy it is to misinterpret significance in statistics and is partly due to the artificial categorisation of subjects into two groups and outliers on the correlation. There was a significant difference between the field dependents and independents on the commands known and commands expert ( $p = .0067$  and  $.0136$  respectively) with the field-independents knowing and being more expert on, more commands than the field-dependents.

There was no significant difference between groups with regards to the number of help keystrokes and time taken. Field Dependency groups did not differ significantly on the number of tasks not completed and the number of interrupts required.

#### Gender effects

There was a significant sex difference ( $p = .0023$ ) found with the females being on average more field-dependent than the males. However, it is worth mentioning that the females within the University group scores are in line with the field-independent mean rather than the field-dependent.

There was a significant difference between genders for the commands known and commands expert variables, with males, on average knowing and being expert with more commands than females ( $p = .0000$  and  $.0001$  respectively). There is also a significant difference for gender ( $p = .004$ ) on the time taken variable with the females taking significantly less time. However, they had also required more help in that time and gave up. It proved impossible to get them to continue on to the more difficult questions.

There was a significant difference ( $p = .0000$ ) between the sexes for number of tasks not

completed with females leaving more tasks incomplete.

There was no significant difference between genders with regards the number of help requests, number of helpkeys and the number of interrupts mainly because the women gave up before completing all the tasks.

#### Training group effects

There was a significant difference ( $p = .0000$ ) between the two groups for field dependency scores though this may be accounted for by the technology centre providing the majority of female subjects.

There was a perhaps unexpected significant difference ( $p = .0002$ ) between the University and technology groups with the University students on average asking for more help. This can be partly explained by the University students completing significantly more tasks ( $p = .0000$ ), and those latter tasks being more difficult than earlier tasks and thus help was more than likely going to be needed. In fact, no subject knew the commands necessary for completion of the last 6 tasks. The majority of the technology students did not complete these tasks. There was also a significant difference ( $p = .0013$ ) for the group with the University students using more keystrokes within the help system. However this may be due to the University students browsing the available information more fully.

Despite the University group asking for more help, there was a significant difference ( $p = .0001$ ) between the University and technology groups with the University group knowing, on average, more commands than the technology group and also being more expert on commands ( $p = .0025$ ). This again may be due to the fact that University students knew the commands for the early tasks, which the technology group did not and then required help on the latter tasks which the technology group did not complete.

There was no significant difference in the number of interrupts required for the different groups.

#### 7.3.4 Interaction effects

Two-way ANOVAS were carried out to investigate the interaction between the experimental system groups and each of the other independent variables. The effect of this interaction on the

number of tasks completed and the amount of help required was assessed.

### **Field Dependency Interactions**

An analysis was carried out to investigate the effect of the interaction between Field Dependency groups and experimental system groups.

There was no interaction effect for the passive and active system groups but there was a pronounced interaction effect for the joint system groups, with field-independents benefitting more from the use of the joint system than the field-dependents. Field-independents using the joint system completed more tasks than the field-dependents using the joint system and more tasks than field-independents using the other two experimental systems. To compare the active and passive system, it is necessary to treat interrupts and help requests as an exposure to help information. When the total help exposure for each subject was calculated as the sum of the number of interrupts, the number of help requests and the number of requests for the information screen, then no effect was found for the amount of help. However, if the number of requests for the information screen is eliminated from this calculation, then field-independents were found to be exposed to more help information especially when using the joint system. This result cannot be taken in isolation, for the other interaction effect was for the number of tasks completed, with field-independents using the joint system completing more tasks than field-dependents using the joint system. The field-dependents did not attempt the latter tasks which every subject required help to complete and thus their number of help requests is artificially low.

### **Gender Interactions**

As far as gender is concerned, the females completed fewer tasks with all systems than males. When investigating the interaction with the amount of help received, females were exposed to less help than males did when using the active or joint system, when access to the information screen was not taken into account, and significantly less help with the static and the joint system when the information screen was included in the calculation for total help exposure. This suggests that the females made more use of the information screen when using the active system than the males. As far as help when information screen requests were included is concerned, it could be that the

females completed fewer tasks and so required less help.

#### **Training group interactions**

The university group completed significantly more tasks than the training group. The interaction was particularly pronounced for the combined group. This supports the interactions found for Field Dependency as most of the university group were field-independent. The university group were also exposed to significantly more help information. This interaction is particularly pronounced for the passive and joint systems. This may arise from them completing more tasks.

### **7.3.5 Qualitative results**

#### **Analysis of the interaction**

During the interaction, subjects were encouraged to express verbally their thoughts about why they were following a particular line of action, any difficulties they were having and whether the information they accessed was what they required. This proved to be a useful technique and allowed effects due to the tasks and problems with the system to be separated from effects due to the access initiative.

It became clear that many subjects preferred to access help information and know exactly what they should be doing before they issued the command. They were naturally cautious about issuing a command that might have been wrong.

Comments were also made about the menu labels and the interaction method that proved useful in assessing the quality of the help system interface, more so than any quantitative performance result could have.

It was also useful to know why the Technology Centre group completed less tasks. They expressed the wish not to continue as they were tired of not being able to do the tasks without help. They saw this as a reflection on themselves and it was clear that it was not because they could not complete the tasks with the help provided. If such qualitative data had not been recorded it would be easy to attribute the lack of completion to inadequacies in the help system.

**Interview comments**

After the experiment had been carried out, the subjects were asked whether they liked the system and what improvements they thought could be made.

**The active system**

This concept was totally novel to all the subjects and there was a clear difference of opinion as to its usefulness. The women from the Technology Centre were divided equally as to whether or not they liked it. The subjects from Computing Science had three opinions:

- 6/13 subjects simply did not like the active help
- 4/13 felt that active help should only be used for protection and not to suggest improvements to the user
- 3/13 felt that active help may be useful as long as they had the ability to switch it off.

No other suggestions as to how the active system could be improved were put forward.

**The passive system**

All the subjects stated that, in general, they liked the menu system but they felt the following features would improve the system:

- Multiple windows to enable both the help information and work context to be visible (10/11)
- A return to command line option, rather than stepping back through the menus (8/11)
- Keyword access to information (7/11)
- The top-level menu headings could be more meaningful(4/11)

**7.4 Discussion****7.4.1 Impact of Field Dependency**

The correlations studies show that Field Dependency correlates with the number of commands known and with the number of tasks they complete: the more field-independent, the more commands known and the more tasks completed. It does not correlate with the use of the help system



as a whole. Field Dependency also shows no correlation with the number of active interrupts required. Although there was no significant difference between the Field Dependency groups on the number of help requests, this may have resulted from the field-dependent group attempting fewer tasks.

#### **7.4.2 Impact of interface type**

Active help is difficult to provide until users get over their fear of trying things out. Users will look up information before they begin to execute commands and the system cannot always use the previous command as an indicator of what help to provide. It may be easier to encourage this way of interacting if the system has an undo capability which allows the user to fully reverse any action if the consequences are not what is wanted.

All the female subjects completed fewer tasks than the male subjects, although they actually asked for less help. This failure to complete the tasks may be due to the females' lack of confidence as they could not complete any of the simple tasks and so asked for help on every task. The men, on the other hand, could complete some of the simpler tasks and so did not always have to ask for help.

The interview comments note a clear preference difference between the Computing Science students and the Technology Centre trainees. The students did not like the idea of active help, they were more interested in browsing the information and seeing how much there was to learn. The trainees, on the other hand, accepted the active help because it allowed them to complete the tasks faster than if they had been required to look for the information for themselves. It was also much easier to get comments on the design of the system from the students than from the trainees. The trainees simply accepted what they had been given, but the students were quite prepared to suggest improvements to the interface. This highlights a problem for designers using a user-centred approach.

#### **7.4.3 Impact of interaction effects**

When looking at the use of help as a whole, no difference was found between the field-dependent and field-independent groups. However, if the interaction between Field Dependency and the

experimental systems is examined, an interesting interaction effect is found. Firstly, there is an effect for use of the joint system, with field-independents performing better, i.e. completing more tasks, with this system than with any other system. This difference was not found for the field-dependent group. Secondly, field-independents were exposed to more help information when using the joint system than any other system, and more than the field-dependents were exposed to.

In this experiment the field-independents were generally exposed to more help information, be it actively or passively initiated. This may be a result of them completing more tasks, or it may be a result of their more active, exploratory nature, and a desire to build a complete mental model of the system, rather than just get on with the tasks.

#### 7.4.4 Measuring performance variables

This study emphasises the complexity and inter-relationships of different measures of performance. The correlation results show how use of a single measure might have led to misinterpretation of what was actually happening. For example, without using the measure of the number of tasks completed other measures such as number of help requests and time taken could be misinterpreted. This could lead to unfounded acceptance of a hypothesis, or design. In previous studies it is assumed that the time taken to complete the tasks has a negative correlation with expertise. However in this case, the opposite effect was found as speed actually had a negative correlation with the number of tasks which were neither attempted nor completed successfully by the subjects.

When looking at the experimental conditions it could be said that it did not matter what system was used, as there were no significant differences for number of tasks completed and time taken, but what is clear from the comments made by subjects after the experiment is that there was a clear dislike for the active system by some users, which was not present for the static system. Suggestions for improving the usability of the static system were offered by subjects, but not for the active system. Here, there was a clear dichotomy of tastes, which are not well researched. It is also noted that in the combined situation, the interrupts did not lead to a reduction in help requests. This cannot be simply interpreted as a failing of the active help information since the information in both cases was exactly the same. It is a problem with timing, because in this experiment, the user wanted the information before they would issue a command. Hidden in the

statistics is the problem that the human helper had to push users to enter a command in the active-only situation much more than the other situations as the users had inherited a distrust of UNIX from their previous experience. The short-term nature of this study meant that this fear could not be overcome for the purposes of the experiment.

#### **7.4.5 Experimental problems**

##### **Access to subjects**

Even within the University environment, it was difficult to get students to volunteer to take part in the experiment. Unlike psychology, where all students must complete a specified number of hours as an experimental subject, Computing Science has no such agreement with the students. The typical subject population of first year psychology students may have provided enough subjects for a more scientifically sound experiment, but would have given little enlightenment on the world of real computer users! Despite the experiment being a useful introduction to UNIX for subjects only 2 students out of a class of over 100 first-year Computing Science students volunteered to take part.

This situation is worsened if experimentation is to include users from industry or Training Centres where any experiments must be done with a minimum of disruption and fit in with any existing work schedules and breaks. In this case, there is not only the biases of the potential subjects to deal with but also the managers in the organisation.

##### **Separation of confounding variables**

The results for gender and training group are in line with what would be expected according to Field Dependency theory. However, whether the observed effects can be attributed solely to a person's level of Field Dependency is not tested in this experiment as most females were field-dependent and all of one of the training groups were female. To separate the effects of gender and training group a number of other subject groups would be required:

- female Computing Science students. Unfortunately, recent UCCA figures show that only 159 females were admitted into British Universities to study Computing Science in 1985, rising to 228 in 1988. With so few female admissions, and these being dispersed

over the whole of Britain, it is little wonder why this subject group would be hard to establish.

- **male technology centre trainees.** Unfortunately, or fortunately depending on individual politics, current Technology Centres are being set up by Councils with a strong feminist influence. This means they are being set up to meet the needs of women who wish to return to work and ignore the needs of men with no formal qualifications, for whom the training might also be extremely useful.
- **female field-independents.** A wider range of Field Dependency scores was desired, but not met by the sample population.
- **male field-dependents.** The males used were nearly all field-independent. It may be that Computing Science attracts those that adopt this way of thinking or it may be that exposure to Computing Science and its constructs invokes a more field-independent approach; what ever the case, the males measured on this dimension were, on the whole, field-independent. A group of Computing Science PhD students were also scored for the EFT test and their scores were even more at the field-independent extreme.

As the experiment stands, it is impossible to untangle the effects due to group and gender from field dependency effects. Politics and biases in the proportion of males and females studying computing science may mean these groups may never be available.

#### **Non-measurable effects**

What is not obvious from the statistics, but was noticeable from observing the experiment was a motivational difference between subjects. Many of the subjects from the Technology Centre group had to be continually encouraged to continue and wanted the tasks to be over as quickly as possible. They were not interested in learning any more about UNIX than they needed to. Many of the Computing Science students were interested in finding out about commands they did not know and were keen to explore all the information. As a result of the previously discussed experimental problems this motivational difference is not clearly attributable to any of the measured user variables.

## **7.5 System enhancements**

There were four main usability issues which arose from the interviews:

1. Keyword access to information about known commands was desirable.
2. A Quit menu option was preferable to stepping back up through the menus when the desired information had been found.
3. Both the work and the help context should be visible at all times to reduce cognitive load.
4. Users should have the ability to switch off the active part of the help system.

Some users were very able, once they had experience of the system, to point out missing (eg. keyword-based access) or disliked functionality (eg. active initiative).

## **7.6 Design recommendations**

Although not explicitly tested in the experiment, interview comments suggest that help provided through a more traditionally designed (passive) help system which runs in a separate window is preferable to an active help system. Given the complexity involved in designing and implementing an acceptable active help system, easier gains could be made by ensuring the quality of the help information and easy and flexible access to such information than following the active approach.

## **7.7 Conclusions**

Although, in general, Field Dependency has been shown to correlate with command knowledge, with the more field-independent subjects having a greater command knowledge, the experiment itself is rather limited. Firstly, the affects of gender and training group confound the results. Secondly, since only a correlation statistic is used, there may be some other variables causing any of the effects noticed, for instance some other sex difference not investigated. An interesting interaction effect was found between Field Dependency and the experimental systems. Field-independents were found to benefit most from the joint system in terms of number of tasks completed, an effect not shown for the field-dependents. Also with the joint system field-independents were exposed

to more help information than with any of the other systems. The field-independents were also exposed to more help information than the field-dependents for all the experimental systems. This may be a direct result of the field-independents completing more tasks, or it may be that the active, explorative nature of the field-independents meant that they looked up more information, perhaps adding to their mental model, rather than just accessing sufficient information to complete the task at hand.

The results of this experiment seem to contradict the results of the experiment discussed in Chapter Five, where field-dependents on average knew less commands and asked for more help. In this case all subjects completed the same number of tasks. This apparent contradiction may reflect the fact that field-dependents are more sociable and more likely to ask another human for help than field-independents. Field-independents are more likely to explore the system and have a greater desire to solve the problem for themselves.

Future studies will require a wider range of subjects, and the effects of increased task difficulty will also have to be considered more carefully. However, even this limited experiment has provided some information regarding the usability of the help system. The experiment may not be scientifically sound, but can still lead to design improvements. A compromise has to be made between getting access to real users and using artificial subjects simply to increase the size of the groups.

Also a longitudinal study is required to establish whether constant access to help systems improves system knowledge. If the system is to be used for long-term learning then it may also be profitable to explore the possibility of creating links between related commands which allows the user to follow a path through the commands rather than dealing with each one individually.

The system itself can also be improved with additions such as keyword-based access and constantly visible help information may improve its usability and acceptability. It should also be concluded that a help system should not be used to compensate for poor interface design. In the case of UNIX, an established operating system, the commands users required most help on were the ones with syntax which did not follow the norm (eg. `chmod`) or which had multiple meanings, for instance `mv`.

## **Chapter 8**

### **Summary and conclusions**

The research began with a simple goal: to explore how important individual differences and different access initiatives are to the usability of online help. Such a broad goal can be elusive, but the approach adopted allowed for some discoveries to be made.

An exploration of the differences between users and how these differences affect their interaction with a computer was made. UNIX was then chosen as a research vehicle and an observation of users in a natural setting was made. A large variation in use of UNIX by different users was noted. Field Dependency was identified as a potential source of the variation between users. An experimental investigation was then carried out into the type of help required for users exhibiting different levels of Field Dependency. At this stage help was provided by a human expert. The major issue to arise from the work carried out to this stage was whether an online help system should provide help passively or actively. The research so far provided input for the design of an online help system and an online help system was developed to investigate the issue of access initiative more fully. The help system was then evaluated experimentally using two distinct user communities as subjects.

#### **8.1 What has been learned?**

**In surveying studies of individual differences; (Chapter 2)**

The idea of individual differences is certainly not new, but the process of applying this theory to the domain of operating systems and how online help systems can be used to reduce the impact

of individual differences is. The effects of individual differences have been measured in a number of areas of Human-Computer Interaction. However, there is disagreement as to the size and relevance of these effects to the interaction process. The main difficulties lie in determining which user characteristics can be used to predict performance differences. The characteristic explored by this research is that of Field Dependency. The nature of this characteristic is controversial. Some previous studies of Field Dependency in Human-Computer Interaction have been flawed and their limitations have been discussed. However, Field Dependency theory has important implications for cognitive model formation and system exploration. The findings of this study suggest that a user's level of Field Dependency affects their eventual level of command knowledge: the more field-independent a user, the greater their knowledge of commands. It also impacts on where a user will look for help, from another user or some form of documentation and whether the user will gain from increased system flexibility.

**In surveying studies of online help systems; (Chapter 3 and Chapter 7)**

Many different approaches to designing online help have been attempted, yet the designers appear to pay little regard to the needs of the users. A large gap has developed between technological developments and user testing of online help systems. This is particularly noticeable in the case of UNIX where a large void in online help systems exists. Researchers of UNIX help systems either criticise the interface to the online manual available via the man command or put major resources into the development of "intelligent" help systems, which cannot currently be supported by the technology available, and the viability of these from the user's perspective has not been assessed.

The experiment reported in Chapter 7 looked at one aspect of intelligent help, access initiative, and tried to assess its viability from the user's perspective. As far as access initiative is concerned, progress could be made with the active approach by providing different triggers to activate the help system. Some of the triggers explored in this research were particularly useful. For instance:

- A synonym capability is particularly useful to aid transfer of learning between systems
- display of options and their use when options are being used is particularly useful with UNIX as there are so many.

However, extreme care must be taken if any new triggers are to be incorporated into the design,



were not all acceptable to the subjects. Any attempts to build an intelligent system which infers what the user is trying to do, or suggests more efficient methods of interaction, were not universally accepted. Any new trigger must be tested with a representative user community for usefulness and acceptability. It may also be necessary to give the user control over whether or not active help should be displayed.

Before any headway can be made with the active initiative, more effort must be put into the design of the passive aspects of the system so that enough flexibility is incorporated into the system to accommodate different styles of working. The following points are important:

- help should be available at all times
- it should be easily accessible
- it should appear in a separate window from the work context
- the information provided should be well organised and presented
- different types of information should be separately available (how-to-do-it and what-it-does)
- different levels of information should be available ( a one line description and basic syntax to more complicated syntax and related commands).

A system so designed could fulfil two functions: firstly to act as online help with the current task and secondly as a learning tool. The user would then have control of their learning experience. They would know when they are working and need just enough information to complete the task at hand and when they have time to browse and learn more about the capabilities of the system.

#### **In surveying studies of UNIX use; (Chapter 4)**

Individual use of UNIX varies enormously, with individual users electing to learn different parts and amounts of the system, and most users learning few commands and using even less with any great frequency. Research has failed to establish whether this is a result of information not being readily accessible by users when it is required or whether they have learnt as much as they require to know to complete their work.

In the experimental studies; (Chapter 5 and 7)

In the study of natural use of UNIX, the results from this research are similar to those reported in other studies. In the two experiments which investigated Field Dependency, Field Dependency and the amount of command knowledge acquired by an individual appear to be related: with more field-independent users acquiring greater command knowledge. A user's level of Field Dependency was also found to affect the amount of help they were exposed to either as a result of a direct request on the part of the user or as a result of active intervention by the help system. In the first experiment, when a human expert was used, field-dependents were found to be exposed to more help information. At first this was attributed to their lack of command knowledge in comparison to the field-independents. However, in the second experiment, the comparative lack of knowledge was still observed, but in this case the field-dependents were exposed to less help information. A possible reason for this was that the field-dependents completed fewer tasks in the second experiment whereas in the first, all subjects completed all the tasks. Failure to attempt a task may have reduced the number of help requests required. An alternative explanation is that field-dependents are more sociable and more likely to ask another person for help, whereas field-independents are less sociable and have a greater desire to solve the problem for themselves and so are less likely to ask another person for help. It is also the case that field-independents are more likely to use instructional material and build up a mental model of the system and so may actively explore the help information to build this model rather than acquire only enough information to complete the task at hand. Field-dependents maybe less concerned with building a mental model and more concerned with simply completing the tasks.

In looking at multiple measures of performance, the problems of using a limited number of measures were identified. Many measures are inter-dependent and using one in isolation can lead to the misinterpretation of results. As far as statistical methods are concerned, it should be noted that statistical results may not tell the whole story. Many studies (including Borenstein, 1985) try to normalise the data to eliminate the effects of individual differences, instead of trying to find out the underlying cause of the differences. Other studies (including Fowler et al., 1987) try to force a categorisation of subjects into groups when no such natural groupings exist and all studies fail to remind readers that statistical techniques rely on random sampling when their samples are

far from random.

**In the development of a online help system; (Chapter 6)**

Using a prototype development cycle to develop a help system was extremely useful. Quick implementation of the prototype allows users to try out the system and design decisions can either be validated or proved wrong. If this is done in conjunction with system development, it can also serve to illustrate potential system flaws as, if it is difficult to design and write the help, then it may be that the system itself has difficulties which may be avoidable. The help system should not be seen as a way to compensate for erroneous system design decisions or used to adapt the user to the system when the system can be changed to suit the users.

**In the evaluation of the online help system; (Chapter 7)**

A well-designed, passive, online help system can go a long way towards alleviating the effects of individual differences. For instance, an online help system with a simple access mechanism and well-written text can provide necessary support for novice users. There are two vital requirements to providing online help. The first is that both the help and current work context should be visible together. If the work context is lost when accessing help then it is more difficult to relate the information provided to the current task and if the help information disappears before the user can return to the command line then the command and/or options required must be memorised. Although not directly tested, most subjects put this forward as a possible improvement to the system. Having both visible reduces the cognitive load necessary to maintain either context. The second requirement is that the help system should always be accessible and not only accessible as a result of an error. This allows the users to use the online help information as a source of information for browsing and learning about the system before they begin any tasks.

The two forms of access initiative are not mutually exclusive, but in fact complement each other. The user can activate help if they want to explore system capabilities without actually issuing commands, and the system can initiate help if suboptimal or dangerous behaviour is observed. If active help is provided then users should have the ability to switch it off. There is a difference of opinion as to whether or not this form of help is useful and this cannot be

accommodated any other way. It should be noted that field-independents are more likely to make use of any increase in flexibility of a system than field-dependents.

## **8.2 Main Limitations**

During the period of this research, a compromise had to be made between involving sufficient numbers of subjects to achieve statistically significant results and using subjects representative of the population the research was designed to help. Therefore, as has been discussed in the sections relating to experimental problems, sufficient control populations were not set up. Another problem caused by lack of subjects was that it was impossible to carry out pilot studies to practise the experimental method before the full experiment was conducted. A trial run was always carried out with non-representative subjects, but this is still not the desired scenario. Another problem with access to subjects was that of time. The subjects were not willing to spend a lot of time on the experiment and with one group, their manager would not allow more than half an hour to be spent away from work.

The active help system created was very simplistic and this simplicity may be the reason for some subjects' dislike of it.

## **8.3 Looking to the future**

There is still a great deal of interesting work to be carried out, which was beyond the scope of this thesis to tackle. Particular areas of interest are outlined below.

### **In the area of cognitive style**

Whether Field Dependency is a cognitive style or not is still a controversial issue, but it is an issue not addressed by this research. An oversight in this work, which occurred because of the order in which the project was completed, was that of not measuring the Field Dependency of users who took part in the observation of natural use of UNIX. It would be interesting to see if a relation existed between Field Dependency and the commands chosen for normal work. It might be useful to compare multiple measures of cognitive style with performance on different

interaction problems, for instance learning system commands or using a help system. It would be interesting to see if they predict the same differences in performance, and if any have a stronger link to measures of performance.

Another interesting point with Field Dependency is whether different occupations attract different individuals displaying different levels of Field Dependency. PhD students in Computing Science are extremely field-independent. It would be interesting to compare this with Field Dependency scores of individuals from different areas of expertise.

#### **In the area of user differences**

There is still great scope for new research in user differences. The question remains as to what user characteristic or combination of characteristics are the best predictors of individual differences in performance. This thesis suggests that variance in user performance is due more to differences between users than differences between designs and training methods for computer systems; therefore further research is warranted. This thesis has made a start by establishing the problems and pitfalls associated with measuring user behaviour. There is also scope for investigating the ability of different system designers to accommodate these differences. Field Dependency was investigated as a possible source of user variation, and does appear to have some predictive capabilities, particularly in the area of command knowledge. However, there are many more user differences which could be investigated.

#### **In the area of online help systems**

Little is known about how people organise activities on-line or whether on-line help would reduce the impact of individual differences, such as levels of Field Dependency or gender, on performance in the long term.

There is also scope for further investigation into online help systems. It may be that a more refined active help system will be more readily accepted by those users who showed dislike of the system in this research. It would also be interesting to carry out a longitudinal study of online help system use, comparing command knowledge for users subjected to different access initiatives over an extended period of time. It may be that a passive online help system which provides links

to unknown commands can facilitate increasing knowledge in a manageable way. This gives some control to the user as to the order in which they accumulate new information. There is also the option to incorporate the design recommendations put forward as a result of the experimental evaluation. This includes the facility to keep both the work context and help information visible and usable at all times.

The ideal way forward may be to create flexible systems which can be customised, or used differently by individual users. However, with the current climate of interface copyright protection, this may not be possible, especially if the process of customising the interface allows it to be made to look like or imitate another interface.

#### **In the area of experimental methods**

There are many difficulties associated with carrying out pure scientific research within the field, especially with the technology changing rapidly, inability to control the work situation and lack of convenient access to potential user groups. Therefore it is necessary to establish whether more informal research can produce significant and valid results which can lead to increased usability of systems. Attempts to normalise data so that strict statistical techniques can be adhered to may lead to the loss of information concerning individual differences.

### **8.4 Interdisciplinary Challenges**

One way forward with similar studies of usability and the building of usable systems is to exploit the interdisciplinary nature of the Human-Computer Interaction problem. The research reported earlier in this thesis shows a large gap between those who design systems and those who test them with real users. For example, the elaborate active systems built either do not actually exist or have not been evaluated with a representative user community.

There has been considerable debate (Carroll & Campbell, 1986; Carroll & Campbell, 1989; Newell & Card, 1985; Newell & Card, 1986; Whiteside & Wixon, 1987) as to the sort of field Human-Computer Interaction (HCI) is, its relation to academic psychology and the role of psychology within HCI and the design of computer systems. Viewed narrowly, HCI can be thought of as a branch of Applied Psychology. However, the problems of usability and HCI in general are

inherently interdisciplinary: developers of computer systems must incorporate psychological analysis and direct investigations at all stages of design and development in order to achieve reasonable coherence and usability (Gould *et al.*, 1987). Conversely, psychologists working in HCI must first develop computer facilities in order to investigate psychological questions about usability. Curtis *et al.* (1987) use the psychology of programming as an example of the need for collaboration between the two fields. There are common misconceptions about both fields: many computer scientists believe that psychology is just another name for common sense and just as many psychologists believe that it is straightforward to write a program and control a system. Psychologists in HCI are very good at criticising existing systems but are short of theoretical foundations for improved design. Computer scientists are not very good at carrying out investigations of how systems will be or are being used. Both the fields of Psychology and Computer Science can make substantial contributions to HCI.

The Psychologist may use empirical measurement by carefully running experiments which compare and quantify differences in performance, usability and learnability between systems. The domains of experimental design and statistical evaluation are immense. It is difficult to define and carry out meaningful experiments and the process should not be left to a computer scientist attempting to be a psychologist. There are many methodological pitfalls. Even when an experiment is carried out using the best method, there are narrow limits to the experimental approach. It is impossible to test even a small subset of the possible variations in the system and its potential users. Therefore, the psychologists can also bring other branches of psychology to the process, and observational studies may be used alongside more controlled experiments to provide a richer range of data. Cognitive psychology can be incorporated and predictive models of behaviour may eventually be derived.

The computer scientists, also, bring many and different skills to the field of HCI. As well as being suitable for users, a system must also be robust and make the most appropriate use of the technology. The computer scientist is responsible for designing and constructing the system, as well as knowing what technology is available and how it can be used. Thus the computer scientist can provide the materials for study. It may be necessary to build many different prototypes of a piece of software if alternative designs are to be evaluated fully. For the experiments to be useful,

they must keep pace with changing technology, and thus building and testing such systems must be done quickly. The computer scientist can also participate in the experiments. This may give them greater insight into the users' needs and the way the system is used, which may lead to improved systems in the future and ensure that experimental results are understood and used correctly. Understanding of experimental results will be improved and this should lead to improved systems in the future.

Thus major projects will require individuals who have a through training in both areas, or a collaboration between individuals from the different areas. The result of such collaboration may lead to the much needed unification of the behavioural and experimental aspects of psychology, and the mathematical and design aspects of computing science.



# Bibliography

- [1] A.P. Aaronson and J.M. Carroll. The answer is in the question: a protocol study of intelligent help. *Behaviour and Information Technology*, 6:393-402, 1987.
- [2] A.P. Aaronson and J.M. Carroll. Intelligent help in a one-shot dialog: a protocol study. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, pages 163-168, Association of Computer Machinery, 1987.
- [3] A.V. Aho, P.J. Weinberger, and B.W. Kernighan. *The AWK Programming Language*. Addison-Wesley Publishing Co., 1988.
- [4] O. Akin, C. Baykan, and D. Radha Rao. Structure of directory space: a case study with a UNIX operating system. *International Journal of Man-Machine Studies*, 26:361-382, 1987.
- [5] M.P. Anderson, J.E. McDonald, and R.W. Schvaneveldt. Empirical user modelling: command usage analysis for deriving models of users. In *Proceedings Of The Human Factors Society 31st Annual Meeting*, pages 41-45, 1987.
- [6] M.J. Bach. *The Design of the UNIX Operating System*. Prentice Hall Inc., New Jersey, 1986.
- [7] P.J. Barnard, N.V. Hammond, A. MacLean, and J. Morton. *Learning and remembering interactive commands*. Technical Report HF055, MRC Applied Psychology Unit, Cambridge, 1982.
- [8] S.J. Boies. User behaviour in an interactive computer system. *IBM Systems Journal*, 13:2-18, 1974.
- [9] N.S. Borenstein. *The Design and Evaluation of On-Line Help Systems*. PhD thesis, Carnegie Mellon University, Pittsburg, Pennsylvania, 1985.
- [10] S.R. Bourne. *The UNIX System*. Addison-Wesley Publishing Co., 1983.
- [11] P. Briggs. Do they know what they're doing? an evaluation of word-processor users' implicit and explicit task-relevant knowledge, and its role in self-directed learning. *International Journal of Man-Machine Studies*, 32:385-398, 1990.
- [12] P. Briggs. What we know and what we need to know: the user model versus the user's model in human-computer interaction. *Behaviour and Information Technology*, 7(4):431-442, 1988.
- [13] A. Brooks. *Evaluation: Where formality demands informality?* Technical Report HCI-1-90, University of Strathclyde, 1990.
- [14] F.P. Brooks Jr. *The mythical man-month: Essays on Software Engineering*. Addison-Wesley Publishing Co., 1975.
- [15] F.R. Campagnoni and K. Ehrlich. Information retrieval using a hypertext-based help system. *ACM Transactions on Information Systems*, 7(3):271-291, 1989.

- [16] S.K. Card, T.P. Moran, and A. Newell. The keystroke level model for user performance time with interactive systems. *Communications of the ACM*, 23:396-410, 1980.
- [17] S.K. Card, T.P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Assoc, Hillsdale, New Jersey, 1983.
- [18] T.T. Carey. User differences in interface design. *IEEE Computer*, 15:14-20, 1982.
- [19] J.M. Carroll and A. Aaronson. Learning by doing with simulated intelligent help. *Communications of the ACM*, 31(9):1064-1079, 1988.
- [20] J.M. Carroll and R.L. Campbell. Artifacts as psychological theories: the case of human-computer interaction. *Behaviour and Information Technology*, 8(4):247-256, 1989.
- [21] J.M. Carroll and R.L. Campbell. Softening up hard science: reply to Newell and Card. *Human Computer Interaction*, 2(3):227-249, 1986.
- [22] J.M. Carroll and C. Carrithers. Training wheels in the user interface. *Communications of the ACM*, 27(8):800-806, 1984.
- [23] J.M. Carroll and R.L. Mack. Metaphor, computing systems and active learning. *International Journal of Man-Machine Studies*, 22(1):39-58, 1984.
- [24] J.M. Carroll, P.L. Smith-Kerker, J.R. Ford, and S.A. Masur. *The Minimal Manual*. Technical Report RC 11637, IBM T.J. Watson Research Center, Yorktown Heights, NY, 1986.
- [25] J.M. Cherry, M.J. Fischer, B.M. Fryer, and M.J. Steckham. Modes of presentation for on-line help: full screen, split screen and windowed formats. *Behaviour and Information Technology*, 8(6):405-416, 1989.
- [26] D.N. Chin. An analysis of scripts generated in writing between users and computer consultants. In *AFIPS Proceedings of the National Computer Conference 53*, pages 637-642, 1984.
- [27] D.N. Chin. User modelling in UC, the UNIX consultant. In *Proceedings of the ACM SIGCHI '86 Human Factors in Computing Systems*, pages 163-187, 1986.
- [28] A.M. Cohill and R.C. Williges. Retrieval of help information for novice users of interactive computer systems. *Human Factors*, 27:335-344, 1985.
- [29] M.J. Coombs and J.L. Alty. Face to face guidance of university computer users—II: characterising advisory interactions. *International Journal of Man-machine Studies*, 12:407-429, 1980.
- [30] L. Coventry. Cognitive style and intelligent help. In *Interact '90 — Third IFIP Conference on Human-Computer Interaction*, pages 1007-1008, 1990.
- [31] L. Coventry. Some effects of cognitive style on learning UNIX. *International Journal of Man-Machine Studies*, 31:349-365, 1989.
- [32] L.M. Coventry. *Adaptive Interfaces: A Review and Bibliography*. Technical Report TR. 63, Department of Computing Science, University of Stirling, 1990.
- [33] B.W. Croft. The role of context and adaptation in user interfaces. *International Journal of Man-Machine Studies*, 21:283-292, 1984.

- [34] B. Curtis, E.M. Soloway, R.E. Brooks, J.B. Black, K Ehrlich, and H.R. Ramsey. Software psychology: the need for an interdisciplinary program. In R.M. Baecker and W.A.S. Buxton, editors, *Readings in Human-Computer Interaction: A multidisciplinary approach*, pages 150-164, Morgan Kaufmann Publishers, Inc, 1987.
- [35] M.C. Desmarais and M. Pavel. User knowledge evaluation: an experiment with UNIX. In H.J. Bullinger and B. Shackel, editors, *Interact '87 - Second IFIP Conference on Human-Computer Interaction*, pages 151-156, Elsevier Science Publishers B.B., North Holland, 1987.
- [36] S.W. Draper. The nature of expertise in UNIX. In *Interact '84 — First IFIP conference on Human-Computer Interaction*, pages 465-471, 1984.
- [37] H.E. Dunsmore. Designing an interactive facility for non-programmers. In *Proceedings of the ACM National Computer Conference*, pages 475-483, 1980.
- [38] W. Dzida, C. Hoffman, and W. Valder. Mastering the complexity of dialogue systems by the aid of work contexts. In H.J. Bullinger and B. Shackel, editors, *Interact '87 - Second IFIP Conference on Human-Computer Interaction*, pages 29-33, Elsevier Science Publishers B.B., North Holland, 1987.
- [39] D.E. Egan. Individual differences in human-computer interaction. In M. Helander, editor, *Handbook of Human-Computer Interaction*, chapter 24, pages 543-580, North Holland Press, New York, 1988.
- [40] D.E. Egan and L.M. Gomez. Assaying, isolating and accommodating individual differences in learning a complex skill. In R. Dillon, editor, *Individual Differences in Cognition Vol 2*, Academic Press, New York, 1985.
- [41] D.E. Egan, L.M. Gomez, K. McKeown, E. Soloway, B. Reiser, and C. Marshall. Dealing with diversity: approaches to individual differences in human-computer interaction. In *CHI '88 Conference Proceedings*, pages 79-81, 1988.
- [42] J. Elkerton. *Online aiding for Human-Computer Interfaces*, chapter 16. Elsevier Science Publishers, North Holland, 1988.
- [43] J. Elkerton and R.C. Williges. Information retrieval strategies in a file-search environment. *Human Factors*, 26(2):171-184, 1984.
- [44] N. Entwistle. *Styles of Learning and Teaching: An integrated outline of Educational Psychology*. John Wiley & Sons, Chichester, 1981.
- [45] R.S. Fenchel and G. Estrin. Self-describing systems using integral help. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-12:162-167, 1982.
- [46] T.W. Finin. Providing help and advice in task-orientated systems. In *IJCAI 83 Proceedings*, pages 176-178, 1983.
- [47] G. Fischer. Making computers more useful and more usable. In G. Salvendy, editor, *Cognitive Engineering in the Design of Human-Computer Interaction and Expert Systems*, Elsevier Science Publishers B.V. Amsterdam, 1987.
- [48] G. Fischer, A. Lemke, and T. Schwab. Active help systems. In G.C. van de Veer, M.J. Tauber, T.R.G. Green, and P. Gorny, editors, *Readings in Cognitive Ergonomics: Mind and Computers*, Springer-Verlag:Berlin, 1984.

- [49] G. Fischer, A. Lemke, and T. Schwab. Knowledge-based systems. In L. Borman and B. Curtis, editors, *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, pages 161-167, 1985.
- [50] C.J.H. Fowler, L.A. Macaulay, and S. Siripoksap. An evaluation of the effectiveness of the adaptive interface module (AIM) in matching dialogues to users. In *Proceedings of the BCS Human-Computer Interaction Conference*, 1987.
- [51] C.J.H. Fowler, L.A. Macaulay, and J.F. Fowler. The relationship between cognitive style and dialogue style. In P. Johnson and S. Cook, editors, *People and Computers: Designing the Interface*, Cambridge University Press, 1985.
- [52] C.J.H. Fowler and D. Murray. Gender and cognitive style differences at the human-computer interface. In *Interact '87 - Second IFIP Conference on Human-Computer Interaction*, pages 709-714, 1987.
- [53] G.W. Furnas, T.K. Landauer, L.M. Gomez, and S.T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964-971, 1987.
- [54] M. Genesereth. An automated consultant for macsyma. In *IJCAI 77 Proceedings*, page 789, 1977.
- [55] T.R. Girill and H.L. Clement. Document: an interactive online solution to four documentation problems. *Communications of the ACM*, 26(5):328-337, 1983.
- [56] L.M. Gomez, D.E. Egan, and C. Bowers. Learning to use a text editor: some learner characteristics that predict success. *Human-Computer Interaction*, 2:1-23, 1986.
- [57] L.M. Gomez, D.E. Egan, E.A. Wheeler, D.K. Sharma, and A.M. Gruchacz. How interface design determines who has difficulty learning to use a text editor. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computer Systems*, pages 176-181, ACM, New York, 1983.
- [58] R. Gong and J. Elkerton. Designing minimal documentation using a GOMS model: a usability evaluation of an engineering approach. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, pages 99-106, ACM Press, 1990.
- [59] M.G. Good, J.A. Whiteside, D.R. Wixon, and S.J. Jones. Building a user-derived interface. *Communications of the ACM*, 27(10):1032-1043, 1984.
- [60] D.R. Goodenough. The role of individual differences in Field Dependence as a factor in learning and memory. *Psychological Bulletin*, 83:675-694, 1976.
- [61] C. Gooding. Too grand for the grubby Hoi Polloi. *Computer Weekly*, 1987.
- [62] J.D. Gould, S.J. Boies, S. Levy, J.T. Richards, and J. Schoonard. The 1984 Olympic message system: a test of behavioural principles of system design. *Communications of the ACM*, 30:758-769, 1987.
- [63] J.D. Gould, J. Conti, and T. Hovanyecz. Composing letters with a simulated listening typewriter. *Communications of the ACM*, 26(4):295-308, 1983.
- [64] S. Greenberg and I.H. Witten. Adaptive personalised interfaces: a question of viability. *Behaviour and Information Technology*, 4(1):31-45, 1985.
- [65] S. Greenberg and I.H. Witten. Directing the user interface: how people use command-based systems. In *Proceedings of the 3rd IFAC Conference on Man-Machine Systems*, 1988.

- [66] M.G. Grignetti, C. Hausmann, and L. Gould. An intelligent on-line assistant and tutor, NLS-SCHOLAR. In *Proceedings of the National Computer Conference*, pages 775-781, AFIPS Press, 1975.
- [67] S. Grimm, J. Malicki, and S. Obermeyer. A user needs approach to context sensitive help. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, page Poster Session, 1987.
- [68] R. Guindon. How to interface to advisory systems? Users request help with a very simple language. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computer Systems*, 1988.
- [69] R. Guindon, K. Shuldberg, and J. Conner. Grammatical and ungrammatical structures in user-advisor dialogues: evidence for sufficiency of restricted languages in natural language interfaces to advisory systems. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 41-44, 1987.
- [70] N. Hammond, J. Morton, J. MacLean, P. Barnard, and J. Long. *Knowledge fragments and users' models of systems*. Technical Report HF071, MRC Cambridge, 1982.
- [71] N.V. Hammond, J. Morton, P.J. Barnard, and I.A. Clark. *Characterising user performance in command-driven dialogue*. Technical Report HF093, MRC Applied Psychology Unit, Cambridge, 1984.
- [72] W.J. Hansen. User engineering principles for interactive systems. In *Proceedings of the American Federation for Information Processing: Fall Joint Conference*, pages 523-532, 1971.
- [73] S.J. Hanson, R.E. Kraut, and J.M. Farber. Interface design and multivariate analysis of UNIX command use. *ACM Transactions on Office Information Systems*, 2(1):42-57, 1984.
- [74] M. Hecking. How to use plan recognition to improve the intelligent help system SINIX consultant. In H.J. Bullinger and B. Shackel, editors, *Interact '87 - Second IFIP Conference on Human Computer Interaction*, pages 657-662, Elsevier Science Publications B.V., Amsterdam, 1987.
- [75] R.C. Houghton. Online help systems: a conspectus. *Communications of the ACM*, 27(2):126-133, 1984.
- [76] H. Hvelplund. The ESPERIT Eurohelp project: an intelligent help system. In *Proceedings of the 1986 Conference on Knowledge Based Systems*, pages 61-70, Online Publications, Pinner, UK, 1986.
- [77] Apple Computer Inc. *Human Interface Guidelines: The Apple Desktop Interface*. Addison-Wesley Publishing Co. Ltd., USA, 1987.
- [78] J. Jerrams-Smith. An attempt to incorporate expertise about users into an intelligent interface for UNIX. *International Journal of Man-Machine Studies*, 31:269-292, 1989.
- [79] J. Jerrams-Smith. An expert system within a supportive interface for UNIX. *Behaviour and Information Technology*, 6:37-41, 1987.
- [80] A.H. Jorgensen. The trouble with UNIX: initial learning and experts' strategies. In H.J. Bullinger and B. Shackel, editors, *Interact '87 - Second IFIP Conference on Human-Computer Interaction*, pages 847-854, Elsevier Science Publishers B.B., North Holland, 1987.

- [81] J. Karat. Evaluating user interface complexity. In *Proceedings of the Human Factors Society 31st Annual Meeting*, pages 556-570, 1987.
- [82] G. Kearsley. *Online Help Systems: Design and Implementation*. Ablex Publishing Corporation, Norwood, New Jersey, 1988.
- [83] J.F. Kelley. An empirical methodology for writing user-friendly natural language computer applications. In A. Janda, editor, *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, 1983.
- [84] C. Kemke. Representation of domain knowledge in an intelligent help system. In H.J. Bullinger and B. Shackel, editors, *Interact '87 - Second IFIP Conference on Human Computer Interaction*, pages 215-220, Elsevier Science Publications B.V., Amsterdam, 1987.
- [85] B.W. Kernighan and J.R. Mashey. The UNIX programming environment. *IEEE Computer*, 14(4):25-34, 1981.
- [86] D. Kieras and P.G. Polson. An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies*, 22:365-394, 1985.
- [87] B. Kolb and I.Q. Whishaw. *Fundamentals of Human Neuropsychology*. W.H. Freeman and Co., San Francisco, 1980.
- [88] R.E. Kraut, S.J. Hanson, and J.M. Farber. Command use and interface design. In *Proceedings of the ACM SIGCHI '83 Human Factors in Computing Systems*, pages 120-124, 1983.
- [89] T.K. Landauer. *Relations between cognitive psychology and computer system design*, pages 1-25. MIT Press, 1987.
- [90] T.K. Landauer, J.M. Carroll, B.E. John, J. Whiteside, and C.G. Wolf. The role of laboratory experiments in HCI: help, hindrance or ho-hum? In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computer Systems*, pages 265-268, 1989.
- [91] J.M. Lewis. *Analysing the actions of UNIX users*. Technical Report DAI working paper 185, Department of Artificial Intelligence, University of Edinburgh, 1986.
- [92] G. Lotwick, A. Simon, and L.O. Ward. Field dependence-independence in male engineering, science and education students. *Perceptual and Motor Skills*, 51:1289-1290, 1980.
- [93] R.L. Mack, C.H. Lewis, and J.M. Carroll. Learning to use office systems: problems and prospects. *ACM Transactions on Office Information Systems*, 1:254-271, 1983.
- [94] A. MacLean, K. Carter, L. Loustrand, and T. Moran. User-tailorable systems: Pressing the issues with buttons. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computer Systems*, pages 175-182, 1990.
- [95] C.S. Magers. An experimental evaluation of on-line help for non-programmers. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, pages 277-281, 1983.
- [96] M.V. Mason. Adaptive command prompting in an on-line documentation system. *International Journal of Man-Machine Studies*, 25:33-51, 1986.
- [97] M.V. Mason and R.C. Thomas. Experimental adaptive interface. *Information Technology: Research Development Applications*, 3(3):162-167, 1984.

- [98] J.T. Mayes, S.W. Draper, A.M. McGregor, and K. Oatley. Information flow in a user interface: the effects of experience and context on the recall of MacWrite screens. In J. Proece and L. Keller, editors, *Human-Computer Interaction*, pages 222-234, Prentice Hall International (UK) Ltd, 1990.
- [99] J.E. McDonald and R.W. Schvaneveldt. *The application of user knowledge to interface design*. Technical Report Memoranda in Computer and Cognitive Science MCCS-86-73, Computing Research Laboratory, New Mexico State University, 1987.
- [100] J. McKendree and J.M. Carroll. Advising roles of a computer consultant. In M. Mantei and P. Orbeton, editors, *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, pages 35-40, 1986.
- [101] F.P. McKenna. Measures of field dependency: cognitive style or cognitive ability? *Journal of Personality and Social Psychology*, 47:593-603, 1984.
- [102] P. McKeivitt. *Data acquisition for natural language interfaces*. Technical Report MCCS-90-178, Computing Research Laboratory, New Mexico State University, 1990.
- [103] S. Messick. Personality consistencies in cognition and creativity. In S. Messick and associates, editors, *Individuality in Learning*, pages 4-22, Jossey-Bass, San Francisco, 1976.
- [104] S. Messick. Style in the interplay of structure and process. In N. Entwistle, editor, *New Directions in Educational Psychology 1. Learning and Teaching*, pages 83-98, The Falmer Press, London, 1985.
- [105] T. Moll and U. Fischbacher. Online assistance: the development of a help-system and an online tutorial. In G. Salvendy and M.J. Smith, editors, *Designing and using human-computer interfaces and knowledge based systems*, pages 97-103, 1989.
- [106] T.P. Moran. The command language grammar: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, 15:3-50, 1981.
- [107] R. Morgan and H. McGilton. *Introducing UNIX system V*. McGraw-Hill Book Co., USA, 1987.
- [108] M.J. Nathan. Empowering the student: prospects for an unintelligent tutoring system. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computer Systems*, pages 407-414, 1990.
- [109] A. Newell and S. Card. The prospects for psychological science in human-computer interaction. *Human-Computer Interaction*, 1:209-242, 1985.
- [110] A. Newell and S. Card. Straightening out softening up: response to Carroll and Campbell. *Human-Computer Interaction*, 2(3):251-267, 1986.
- [111] E.S. Nickerson. Why interactive computer systems are sometimes not used by people who might benefit from them. *International Journal of Man-Machine Studies*, 15:469-483, 1981.
- [112] D.A. Norman. Cognitive engineering principles in the design of human-computer interfaces. In G. Salvendy, editor, *Human-Computer Interaction: Proceedings of the first USA-JAPAN Conference*, pages 11-16, 1984.
- [113] D.A. Norman. Some observations on mental models. In D. Gentner and A.L. Stevens, editors, *Mental Models*, pages 7-14, Lawrence Erlbaum Associates, Hillsdale, N.J., 1983.
- [114] D.A. Norman. The trouble with UNIX. *Datamation*, 27(12):139-150, 1981.

- [115] P. Orwick, J.T. Jaynes, T.R. Barstow, and L.S. Bohn. DOMAIN/DELPHI: retrieving documents online. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, pages 114-121, 1986.
- [116] G. Paak. Styles and strategies of learning. *British Journal of Educational Psychology*, 46:128-148, 1976.
- [117] S.J. Payne and T.R.G. Green. Task-action grammars: a model of the mental representation of task languages. *Human-Computer Interaction*, 2:93-133, 1986.
- [118] L.J. Peters. Relating software requirements to design. In *Proceedings of the Software Quality and Assurance Workshop*, pages 67-71, ACM, New York, 1978.
- [119] M.E. Pollack. Information sought and information provided: an empirical study of user/expert dialogues. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, pages 155-159, 1985.
- [120] K.M. Potosnak. Choice of interface modes by empirical groupings on computer users. In *INTERACT-84*, 1984.
- [121] A.E. Quilici, M.G. Dyer, and M. Flowers. AQUA: an intelligent UNIX advisor. In *Proceedings of the 1986 European Conference on Artificial Intelligence*, pages 33-38, 1986.
- [122] L. Quinn and D.M. Russell. Intelligent interfaces: user models and planners. In *Proceedings of the CHI '86 Conference*, pages 314-320, 1986.
- [123] N. Relles, N. Sondheimer, and G. Ingargiola. A unified approach to online assistance. In *Proceedings of the AFIPS National Computer Conference 50*, pages 383-388, 1979.
- [124] J.R. Rhyne and C.G. Wolf. Gestural interfaces for information processing applications. In *Interact '87 - Second IFIP Conference on Human-Computer Interaction*, 1987.
- [125] E. Rich. Programs as data for their help systems. In *National Computer Conference Proceedings*, pages 481-485, AFIPS, 1982.
- [126] E. Rich. User modelling via stereotypes. *Cognitive Science*, 3:329, 1979.
- [127] M.D. Ringle and R. Halstead-Nussloch. Shaping user input: a strategy for natural language dialogue design. *Interacting With Computers*, 1(3):227-244, 1989.
- [128] E.L. Risland. Ingredients of intelligent user interfaces. *International Journal of Man-Machine Studies*, 21(4):377-388, 1984.
- [129] D.M. Ritchie and K. Thompson. The UNIX time-sharing system. *Communications of the ACM*, 17(7):365-375, 1974.
- [130] I.T. Robertson. Human information processing strategies and style. *Behaviour and Information Technology*, 4(1):19-29, 1985.
- [131] A.D. Saja. The cognitive model: an approach to designing the human-computer interface. *SIGCHI Bulletin*, 16(3):36-40, 1985.
- [132] M.L. Schneider and J.C. Thomas. The humanization of computer interfaces. *Communications of the ACM*, 26(4), 1983.
- [133] M.M. Sebrechts, J.G. Deck, and J.B. Black. A diagrammatic approach to instruction for the naive user. *Behaviour Research Methods and Instrumentation*, 15:200-207, 1983.



- [134] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Reading, M.A., 1987.
- [135] B. Shneiderman. *Human Factors Issues of Manuals, Online Help and Tutorials*. Technical Report CS-TR-1446, Department of Computer Science, University of Maryland, 1984.
- [136] J. Shrager and D. Klahr. Instructionless learning about a complex device: the paradigm and observations. *International Journal of Man-Machine Studies*, 25(2):153-189, 1986.
- [137] S. Smith and J. Mosier. *Design Guidelines for User-System Interface Software*. Technical Report, MITRE Corporation, Bedford Mass, 1986.
- [138] N.K. Sondheimer and N. Relles. Human factors and user assistance in interactive computing systems: an introduction. *IEEE Transactions on Systems, Man and Cybernetics*, 12(2):102-107, 1982.
- [139] A.G. Sutcliffe and A.C. Old. Do users know they have user models? Some experiences in the practice of user modelling. In *Interact '87 - Second IFIP Conference on Human-Computer Interaction*, pages 35-41, 1987.
- [140] H. Thimbleby. Dialogue determination. *International Journal of Man-Machine Studies*, 13:295-304, 1980.
- [141] W. Tietelman. The Cedar programming environment. *SIGGRAPH Video Review*, 19:8, 1985.
- [142] R. Trevelyan and D.P. Browne. A self-regulating adaptive system. In *CHI & GI 87*, pages 103-107, 1987.
- [143] R. Tuck and D.R. Olsen. Help by guided tasks: utilizing UIMS knowledge. In *Proceedings of the ACM SIGCHI Conference: Human Factors in Computing Systems*, pages 71-78, ACM Press, 1990.
- [144] L.E. Tyler. *Individual Differences*. Meredith Corporation, USA, 1974.
- [145] G.C. Van der Voer, M.J. Tauber, Y. Waern, and B. Muylwijk. On the interaction between system and user characteristics. *Behaviour and Information Technology*, 4:289-308, 1985.
- [146] K.J. Vicente, B.C. Hayes, and R.C. Williges. Assaying and isolating individual differences in searching a hierarchical file system. *Human Factors*, 29:349-359, 1987.
- [147] M. Waite, D. Martin, and S. Prata. *UNIX Primer Plus: User Friendly Guide to the UNIX operating system*. Howard W Sams & Co., Inc., Indiana:USA, 1983.
- [148] J. Walker. The document examiner. *SIGGRAPH Video Review*, 19:4, 1985.
- [149] J.A. Whiteside and D.R. Wixon. Improving human-computer interaction: a quest for cognitive science. In J.M. Carroll, editor, *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, pages 353-365, MIT Press, Cambridge, M.A., 1987.
- [150] R. Wilensky, Y. Arens, and D. Chin. Talking to UNIX in English: an overview of UC. *Communications of the ACM*, 27(6):574-593, 1984.
- [151] W. Wilensky, D.N. Chin, M. Luria, J. Martin, J. Mayfield, and D. Wu. The Berkeley UNIX consultant project. *Computational Linguistics*, 14(4):35-84, 1988.
- [152] T. Winograd and F. Flores. *Understanding Computers and Cognition: A New Foundation for Design*. Ablex Publishing Corporation, Norwood, NJ, 1986.

- [153] H.A. Witkin. Individual differences in ease of perception of embedded figures. *Journal of Personality*, 19:1-15, 1950.
- [154] H.A. Witkin, R.B. Dyke, H.F. Faterson, D.R. Goodenough, and S.A. Karp. *Psychological Differentiation*. Wiley, London, 1962. reprinted by Erlbaum, 1974.
- [155] H.A. Witkin, R.B. Dyke, and P.K. Oltman. Psychological differentiation: current status. *Journal of Personality and Social Psychology*, 37:1127-1145, 1979.
- [156] H.A. Witkin and D.R. Goodenough. *Cognitive Styles: Essence and Origin*. International University Press, 1981.
- [157] H.A. Witkin, D.R. Goodenough, and S.A. Karp. Stability of cognitive style from childhood to young adulthood. *Journal of Personality and Social Psychology*, 7:291-300, 1967.
- [158] H.A. Witkin, H.B. Lewis, M. Hertzman, K. Machover, P. Bretnall Meissner, and S. Wapner. *Personality through perception: An Experimental and Clinical study*. Greenwood Press Publishers, Connecticut, 1973.
- [159] H.A. Witkin, C.A. Moore, D.R. Goodenough, and P.W. Cox. Field-dependent and field-independent cognitive style and their educational implications. *Review of Educational Research*, 1-64, 1977.
- [160] H.A. Witkin, P.K. Oltman, and S.A. Karp. *A Manual for the Embedded Figures Tests*. Consulting Psychologists Press Inc., California, 1971.
- [161] R.M. Young. The machine inside the machine: users' models of pocket calculators. *International Journal of Man-Machine Studies*, 15:81-85, 1981.
- [162] G.K. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, 1949.
- [163] A.Y. Zissos and I.H. Witten. User modelling for a computer coach: a case study. *International Journal of Man-Machine Studies*, 23:729-750, 1985.

## **Appendix A**

### **User log extract**

```

USER      njd
LOGIN    Mon Sep 19 12:59:10 1988

|NRM|    52.07    0.45    0.91    |passwd|
|TOT|    65.48    0.47    0.91

LOGOUT   Mon Sep 19 13:00:15 1988

USER      njd
LOGIN    Mon Sep 19 13:00:30 1988

|TOT|    43.12    0.02    0.00

LOGOUT   Mon Sep 19 13:01:13 1988

USER      njd
LOGIN    Mon Sep 26 21:16:01 1988

|NRM|    0.20    0.04    0.01    |d|
|NRM|    0.26    0.16    0.05    |ls|
|NRM|    29.42    0.95    0.56    |man mail|
|NRM|    2.87    0.04    0.02    |
|TOT|    80.67    1.24    0.65

LOGOUT   Mon Sep 26 21:17:22 1988

USER      njd
LOGIN    Mon Sep 26 20:14:11 1988

|NRM|    0.43    0.14    0.24    |who|
|NRM|    0.06    0.05    0.01    |home|
|NRM|    0.17    0.12    0.03    |ls|
|NRM|    0.06    0.06    0.00    |rogue|
|NRM|    0.44    0.11    0.26    |man|
|NRM|    0.01    0.00    0.01    |cd ..|
|NRM|    0.11    0.03    0.04    |/games|
|NRM|    0.07    0.05    0.02    |rogue|
|NRM|    0.38    0.21    0.11    |ls|
|NRM|    0.01    0.00    0.01    |cd ..|
|NRM|    0.19    0.14    0.05    |ls|
|NRM|    0.10    0.05    0.04    |games|
|NRM|    0.07    0.03    0.04    |/games|
|NRM|    2.11    0.57    0.40    |man d|
|NRM|    0.39    0.09    0.01    |games rogue|
|NRM|    0.18    0.12    0.04    |ls games|
|NRM|    0.20    0.14    0.06    |ls|
|NRM|    3904.49  17.08  128.42 |games/rogue|
|NRM|    0.00    0.00    0.00    |
|NRM|    0.37    0.18    0.19    |who|
|TOT|    4573.03  19.32  130.01

```

LOGOUT Mon Sep 26 21:30:24 1988

USER njd

LOGIN Fri Sep 30 11:32:15 1988

NRM	22.19	1.03	0.57	man sh > shelltex
NRM	0.35	0.13	0.02	ls
NRM	2.10	0.67	0.24	lp shelltex
NRM	0.52	0.21	0.23	who
NRM	105.11	0.44	0.04	cat shelltex
NRM	0.06	0.05	0.01	
TOT	234.41	2.57	1.12	

LOGOUT Fri Sep 30 11:36:09 1988

USER njd

LOGIN Fri Sep 30 11:43:07 1988

NRM	0.81	0.17	0.24	who
NRM	262.02	0.83	0.62	man mail
INT	57.30	0.34	0.25	mail djr
INT	17.09	0.28	0.22	mail cs djr
TOT	771.17	1.64	1.36	

LOGOUT Fri Sep 30 11:55:58 1988

USER njd

LOGIN Sun Oct 2 13:58:41 1988

NRM	0.01	0.00	0.01	cd ..
NRM	0.01	0.00	0.01	cd ..
NRM	0.01	0.00	0.01	cd games
NRM	1664.34	6.94	47.64	rogue
NRM	0.00	0.00	0.00	
NRM	603.32	4.62	28.19	rogue
NRM	433.59	3.16	20.42	rogue
NRM	0.00	0.00	0.00	
NRM	901.81	5.26	35.07	rogue
NRM	0.01	0.01	0.00	cd ..
NRM	0.01	0.00	0.01	cd ..
NRM	0.24	0.17	0.02	ls
NRM	0.01	0.01	0.00	cd usr
NRM	0.01	0.01	0.00	cd njd
INT	0.01	0.01	0.00	ls
NRM	0.29	0.14	0.06	ls
INT	0.08	0.03	0.01	\cd names
NRM	0.41	0.17	0.24	who
NRM	48.50	0.24	0.08	write sjr
NRM	0.36	0.22	0.03	mail
TOT	4543.85	21.15	131.84	

LOGOUT Sun Oct 2 15:14:25 1988

USER njd  
LOGIN Mon Oct 3 15:52:17 1988

NRM	0.61	0.26	0.01	mail
TOT	41.67	0.26	0.02	

LOGOUT Mon Oct 3 15:52:58 1988

USER njd  
LOGIN Sun Oct 9 19:43:53 1988

NRM	0.01	0.00	0.01	cd ..
NRM	0.01	0.00	0.01	cd ..
NRM	0.01	0.01	0.00	cd games
NRM	7695.69	37.79	298.71	rogue
TOT	7739.05	37.84	298.73	

LOGOUT Sun Oct 9 21:52:52 1988

USER njd  
LOGIN Mon Oct 10 18:23:16 1988

NRM	0.51	0.20	0.22	who
INT	0.01	0.01	0.00	cd games
NRM	0.01	0.01	0.00	cd ..
NRM	0.01	0.01	0.00	cd ..
NRM	0.01	0.02	0.00	cd games
NRM	103.16	0.92	5.68	rogue
NRM	1198.09	7.58	54.86	rogue
NRM	0.45	0.18	0.22	who
NRM	2230.38	14.37	109.50	rogue
NRM	0.01	0.00	0.01	
NRM	118.87	0.68	5.13	rogue
NRM	0.01	0.00	0.01	cd ..
NRM	0.21	0.16	0.05	ls
INT	0.02	0.00	0.02	cd names
INT	0.01	0.00	0.01	cd sys
NRM	0.20	0.13	0.07	ls
NRM	0.08	0.05	0.00	pwd
INT	0.01	0.00	0.01	cd home
NRM	0.19	0.14	0.05	ls
NRM	0.01	0.00	0.01	cd src
NRM	0.22	0.11	0.04	ls
NRM	0.01	0.01	0.00	cd ..
NRM	0.01	0.00	0.01	cd ..
NRM	0.18	0.12	0.06	ls
NRM	0.01	0.01	0.00	cd bin
NRM	0.51	0.19	0.18	ls
NRM	0.01	0.01	0.00	cd ..
NRM	0.19	0.15	0.04	ls

NRM	0.01	0.00	0.01	cd usr
NRM	0.20	0.17	0.02	ls
NRM	0.01	0.00	0.01	cd bin
NRM	1.72	0.36	0.26	ls
NRM	0.01	0.00	0.01	cd ...
NRM	0.38	0.17	0.21	who
NRM	0.20	0.13	0.06	ls
NRM	0.01	0.00	0.01	cd tmp
NRM	0.17	0.11	0.03	ls
NRM	0.01	0.01	0.00	cd ...
NRM	0.01	0.00	0.01	cd lib
NRM	1.00	0.22	0.16	ls
NRM	0.01	0.00	0.01	cd ...
NRM	0.23	0.14	0.06	ls
NRM	0.01	0.00	0.01	cd man
NRM	0.24	0.15	0.02	ls
NRM	0.01	0.00	0.01	cd ...
INT	0.03	0.01	0.01	cd names
NRM	0.01	0.00	0.01	cd pub
NRM	0.18	0.13	0.02	ls
NRM	0.01	0.00	0.01	cd ...
NRM	0.01	0.01	0.00	cd ...
NRM	0.19	0.14	0.05	ls
TOT	4113.08	27.18	177.28	

LOGOUT Mon Oct 10 19:31:49 1988

USER njd  
LOGIN Mon Oct 10 19:31:59 1988

NRM	0.16	0.13	0.03	ls
NRM	0.45	0.20	0.00	pwd
NRM	0.14	0.06	0.01	rogue.save
NRM	0.01	0.00	0.01	cd ...
NRM	0.01	0.01	0.00	cd ...
NRM	0.01	0.01	0.00	cd games
NRM	348.43	2.89	19.56	rogue
NRM	0.44	0.26	0.18	rogue rogue.save
NRM	3390.98	21.56	175.92	rogue
NRM	0.01	0.00	0.01	
TOT	3868.46	25.18	195.75	

LOGOUT Mon Oct 10 20:36:27 1988

USER njd  
LOGIN Fri Oct 14 23:26:20 1988

NRM	0.48	0.18	0.21	who
NRM	13.72	0.21	0.08	write nrj
NRM	0.06	0.01	0.00	cd ...
NRM	0.05	0.00	0.01	cd ...
NRM	0.01	0.00	0.01	cd games

NRM	81.20	0.71	3.06	rogue
TOT	160.44	1.16	3.39	

LOGOUT Fri Oct 14 23:29:01 1988

USER njd  
LOGIN Fri Oct 14 23:30:07 1988

NRM	0.01	0.00	0.01	cd ..
NRM	0.01	0.01	0.00	cd ..
NRM	0.07	0.04	0.03	rogue
NRM	0.01	0.00	0.01	cd games
NRM	1029.26	6.71	53.95	rogue
NRM	0.01	0.01	0.00	
NRM	55.78	0.57	2.41	rogue
NRM	981.85	5.73	45.87	rogue
NRM	5977.91	29.19	262.14	rogue
NRM	0.00	0.00	0.00	
TOT	8126.54	42.34	364.44	

LOGOUT Sat Oct 15 01:45:34 1988

USER njd  
LOGIN Mon Oct 17 23:02:18 1988

NRM	0.01	0.01	0.00	cd ..
NRM	0.01	0.01	0.00	cd ..
NRM	0.01	0.01	0.00	cd games
NRM	520.49	4.30	33.26	rogue
NRM	2277.13	15.43	138.65	rogue
NRM	5748.17	36.53	317.94	rogue
TOT	8586.80	56.33	489.85	

LOGOUT Tue Oct 18 01:25:25 1988

USER njd  
LOGIN Sat Oct 22 17:45:53 1988

NRM	1.03	0.23	0.24	who
NRM	107.81	0.95	0.57	man who
NRM	0.09	0.06	0.00	who-a
NRM	0.56	0.17	0.19	who a
NRM	2.92	0.39	0.53	who -a
NRM	0.77	0.30	0.30	who -u
NRM	0.42	0.10	0.05	chmod go= *
NRM	0.84	0.20	0.28	ls -l
NRM	0.01	0.01	0.00	cd ..
NRM	0.04	0.00	0.01	cd ..
NRM	0.01	0.00	0.01	cd games
NRM	544.29	6.71	34.82	rogue
NRM	2407.49	17.02	137.91	rogue



MFM	145.02	1.02	10.18	rogue
TOT	3438.84	27.25	185.14	

LOGOUT Sat Oct 22 18:43:11 1988

USER njd  
LOGIN Sat Oct 22 19:11:36 1988

MFM	0.01	0.00	0.01	cd ..
MFM	0.04	0.01	0.00	cd ..
MFM	0.01	0.00	0.01	cd games
MFM	2834.13	21.15	167.93	rogue
MFM	0.01	0.01	0.00	
MFM	456.40	4.54	30.43	rogue
MFM	1247.64	11.87	89.16	rogue
MFM	2190.78	18.54	137.43	rogue
MFM	0.00	0.00	0.00	
MFM	5242.06	35.25	289.01	rogue
MFM	0.00	0.00	0.00	
MFM	5560.91	36.89	331.95	rogue
TOT	17652.68	128.35	1045.97	

LOGOUT Sun Oct 23 00:05:48 1988

USER njd  
LOGIN Sun Oct 23 16:36:14 1988

MFM	0.54	0.17	0.01	ls
MFM	2.27	0.70	0.38	man vedit
MFM	172.67	0.90	0.57	man vi
MFM	0.89	0.25	0.22	who
MFM	1.45	0.33	0.56	who -a
MFM	51.92	0.95	0.53	man ex
TOT	412.63	3.34	2.28	

LOGOUT Sun Oct 23 16:43:07 1988

USER njd  
LOGIN Sun Oct 23 16:28:37 1988

MFM	0.39	0.14	0.23	who -max
MFM	0.44	0.16	0.25	who
MFM	0.85	0.24	0.27	ls -l
INT	6486.49	1.68	1.28	vedit assign
TOT	6573.18	2.28	2.04	

LOGOUT Sun Oct 23 18:18:10 1988

USER njd  
LOGIN Tue Nov 1 11:51:03 1988

NRM	0.75	0.19	0.24	who
TOT	31.59	0.20	0.24	

LOGOUT Tue Nov 1 11:51:35 1988

USER njd  
LOGIN Tue Nov 1 22:04:58 1988

NRM	0.01	0.00	0.01	cd ..
NRM	0.04	0.01	0.00	cd ..
NRM	0.03	0.01	0.01	cd games
NRM	343.98	3.45	24.92	roguel
NRM	2028.91	14.37	130.86	roguel
NRM	111.36	0.54	1.97	roguel
NRM	4548.28	29.66	420.12	roguel
TOT	7075.22	48.08	577.91	

LOGOUT Wed Nov 2 00:02:54 1988

USER njd  
LOGIN Wed Nov 2 00:05:03 1988

NRM	2.23	0.56	0.43	man latex > text
NRM	1.83	0.50	0.44	man tex > text
NRM	6.61	0.77	0.55	man vi > vip
NRM	2.41	0.65	0.29	lp vip
NRM	0.21	0.15	0.01	ls
NRM	0.29	0.12	0.01	rm shelltex
NRM	0.31	0.11	0.02	rm rogue.save
NRM	0.15	0.13	0.02	ls
NRM	0.12	0.09	0.02	cat text
NRM	0.25	0.10	0.00	rm text
NRM	65.48	0.87	0.06	latex
NRM	0.15	0.13	0.02	ls
NRM	0.15	0.09	0.02	rm texput.log
NRM	0.19	0.19	0.00	pwd
NRM	8.86	0.41	0.13	vi letter.tex
NRM	0.15	0.13	0.02	ls
NRM	28.53	0.41	0.18	vi let.tex
NRM	192.78	1.20	0.68	latex let.tex
NRM	0.15	0.12	0.03	ls
NRM	0.11	0.11	0.00	cat let.tex
NRM	0.20	0.09	0.03	rm let.log
NRM	0.17	0.09	0.03	rm let.tex
NRM	0.01	0.01	0.00	cd ..
NRM	0.01	0.01	0.00	cd ..
NRM	0.01	0.01	0.00	cd games
NRM	10434.36	63.93	602.97	roguel
NRM	0.00	0.00	0.00	
TOT	11448.89	71.15	606.01	

LOGOUT Wed Nov 2 03:15:52 1988

USER njd  
LOGIN Sun Nov 6 21:00:07 1988

NRM	0.43	0.19	0.19	who
NRM	0.01	0.01	0.00	cd ...
NRM	0.06	0.01	0.00	cd ...
NRM	4646.84	30.59	310.20	games/rogue
NRM	0.43	0.17	0.21	who
NRM	0.27	0.10	0.02	games
NRM	0.01	0.01	0.00	cd games
NRM	0.15	0.11	0.04	ls
NRM	0.12	0.08	0.03	cat lib
NRM	0.00	0.00	0.00	
NRM	0.10	0.08	0.02	lib
NRM	4601.17	31.16	473.40	rogue
NRM	6.95	0.73	0.54	man cpl
NRM	0.01	0.00	0.01	cd ...
NRM	0.43	0.17	0.04	ls
NRM	0.01	0.01	0.00	cd fsl
NRM	0.04	0.01	0.00	cd ...
NRM	0.35	0.16	0.03	ls publ
NRM	0.34	0.15	0.06	ls fsa
NRM	0.22	0.14	0.01	ls fsci
NRM	0.20	0.13	0.07	ls
NRM	0.29	0.14	0.07	ls fsa/pub
NRM	0.03	0.02	0.00	cd fsl
NRM	0.50	0.20	0.14	ls
NRM	0.01	0.00	0.01	cd njd
NRM	0.26	0.11	0.03	cp usr/fsa/pub/y5ass.pl
NRM	4.81	0.73	0.52	man cpl
NRM	0.13	0.09	0.04	cp /usr/fsa/pub/y5ass.p assig.pl
NRM	0.00	0.00	0.00	cd ...
NRM	0.35	0.20	0.15	ls
NRM	0.01	0.00	0.01	cd ...
NRM	0.26	0.16	0.06	ls fsa/pub
NRM	0.30	0.15	0.02	cp fsa/pub/y5ass1.p fsl/njd/assig.pl
NRM	0.02	0.00	0.02	cd fsl/njd
NRM	0.15	0.11	0.04	ls
NRM	0.01	0.00	0.01	cd ...
NRM	0.01	0.01	0.00	cd .
NRM	0.39	0.19	0.12	ls
NRM	0.01	0.00	0.01	cd ...
NRM	21257.37	114.17	1234.13	games/rogue
NRM	0.01	0.00	0.01	
NRM	0.19	0.12	0.07	ls
NRM	0.02	0.00	0.02	cd games/lib
NRM	0.15	0.13	0.02	ls
NRM	0.11	0.11	0.00	cat topten
TOT	31342.14	180.95	2020.42	

LOGOUT Mon Nov 7 05:42:29 1988

USER njd  
LOGIN Thu Nov 10 23:05:20 1988

NRM	0.23	0.13	0.04	ls
NRM	93.98	0.51	1.14	vedit vip
NRM	0.19	0.09	0.04	rm vip
NRM	1222.79	5.75	4.75	vedit greaser
NRM	2.19	0.70	0.23	lp -dtosh greaser
NRM	260.11	0.99	1.09	vedit greaser
NRM	0.15	0.12	0.03	ls
NRM	0.13	0.11	0.01	cat greaser
NRM	1.64	0.67	0.26	lp stat
NRM	28.49	0.71	0.20	lp -stat
NRM	0.00	0.00	0.00	
NRM	20.01	0.78	0.60	man lp
NRM	0.52	0.19	0.02	lpstat -dtosh
NRM	1.53	0.42	0.50	lpstat
NRM	6.54	0.76	0.51	man rm
NRM	1.72	0.45	0.44	lpstat
NRM	2.65	0.72	0.57	cancel mpr-3094
NRM	1.58	0.68	0.47	cancel tosh-3093
NRM	1.13	0.41	0.45	lpstat
NRM	657.17	3.87	3.45	vedit lithium
TOT	2694.03	18.22	14.83	

LOGOUT Thu Nov 10 23:50:15 1988

USER njd  
LOGIN Sat Nov 12 15:24:16 1988

NRM	0.21	0.12	0.03	ls
NRM	1.96	0.66	0.25	lp -dtosh greaser
NRM	9.42	0.32	0.38	vedit lithium
NRM	0.15	0.14	0.01	ls
NRM	1.50	0.67	0.22	lp -dtosh lithium
NRM	0.99	0.29	0.28	lpstat
NRM	12.69	0.75	0.59	man lp
NRM	0.23	0.15	0.01	lpstat -dtosh
NRM	2.61	0.78	0.20	lp greaser
NRM	1.52	0.62	0.25	lp lithium
NRM	0.80	0.29	0.27	lpstat
NRM	0.68	0.29	0.25	lpstat
NRM	1.63	0.68	0.24	lp -dtosh lithium
NRM	0.69	0.24	0.27	lpstat
NRM	358.91	1.10	2.08	vedit greaser
NRM	243.39	1.09	2.36	vedit greaser
NRM	89.69	0.78	1.78	vedit lithium
NRM	47.63	0.82	0.56	man lp
NRM	1.81	0.71	0.27	lp -dtosh greaser lithium
NRM	1.05	0.34	0.27	lpstat
NRM	2.30	0.68	0.28	lp -dtosh greaser lithium

NRM	0.53	0.15	0.02	ls
TOT	2088.84	11.85	10.93	

LOGOUT Sat Nov 12 15:59:05 1988

USER njd  
LOGIN Sun Nov 13 02:42:54 1988

NRM	0.20	0.14	0.02	ls
NRM	5.72	0.69	2.91	pascal assig.pl
NRM	1.94	0.48	0.46	man pascal
NRM	0.15	0.12	0.02	pascal assig
NRM	37.00	2.76	7.47	pascal assig.p assig.pl
NRM	0.16	0.14	0.01	ls
NRM	0.17	0.08	0.04	rm assig.o
NRM	4.37	0.52	2.99	pascal assig.pl
NRM	0.16	0.11	0.04	ls
NRM	0.16	0.06	0.02	assig.o
TOT	174.32	5.14	14.00	

LOGOUT Sun Nov 13 02:45:49 1988

USER njd  
LOGIN Sun Nov 13 12:19:04 1988

NRM	0.15	0.11	0.04	ls
NRM	0.27	0.14	0.01	rm assig.o
NRM	5.58	0.74	2.91	pascal -p assig.pl
NRM	0.23	0.22	0.01	pascal -r assig.o
NRM	0.34	0.18	0.00	ls
NRM	0.01	0.01	0.00	cd ..
NRM	0.01	0.00	0.01	cd ..
NRM	0.71	0.22	0.06	ls
NRM	0.01	0.01	0.00	cd fsa
NRM	0.44	0.14	0.10	ls
NRM	0.01	0.01	0.00	cd pub
NRM	0.43	0.20	0.05	ls
NRM	0.01	0.00	0.01	cd ..
NRM	0.07	0.01	0.01	cd ..
NRM	0.01	0.00	0.01	cd fsb
NRM	0.01	0.01	0.00	cd njd
NRM	0.22	0.13	0.04	ls
NRM	428.17	0.93	0.85	vedit source
NRM	0.82	0.17	0.29	pascal -r assig.o source
NRM	1.14	0.19	0.05	ls
NRM	893.98	2.01	4.26	vedit assig.pl
NRM	1981.31	8.63	9.24	vedit graphite
NRM	0.41	0.14	0.04	ls
TOT	4552.11	14.42	18.02	

LOGOUT Sun Nov 13 13:34:56 1988

## **Appendix B**

### **Task instructions**

## Introduction

- Please attempt each of the following questions using the computer.
- At no point should you access the editor.
- To rubout characters on a command line, use the back space key.
- You may ask for help at any time.

Your user name is qwe

Your password is expt12

Please complete as many tasks as you can.

## The Tasks

1. Change the password to reply12.
2. This computer can be used by different people at the same time, find out who else is using the computer at the moment.
3. Stop other people interrupting your work.
4. Display the current time of day on the screen.
5. You are currently positioned within your working environment (home directory). There is a variety of information stored for you.  
Show on the screen the names of all files stored in this directory.
6. Show, on the screen the names of the files, in the order of when someone last changed them. ie the file which was most recently changed should be at the top of the list.
7. Display the contents of the file called "intro.cs" on the screen
8. An extra copy of this file is required. Make a copy and call it "update"
9. The information in the files "intro.cs" and "intro1.cs" should really be kept in a single file, with the contents of intro.cs being followed by the contents of intro1.cs.  
Complete this change storing the result in "fullintro.cs".
10. It is sometimes necessary to group related information together in a specific place (like a folder for a particular subject). Demo is such an area, find out what files are stored in this area.
11. Create 2 such areas called computer and private
12. Move all the files ending with .pr to private.
13. Now remove all private files (ie. those ending in .pr) and any work space created for them from the computer.

14. Show, on the screen, the number of words and lines in the files `fullintro.cs` and `ending.cs` followed by the total of the two added together.
15. How many lines contain the word "usability" in the file `fullintro.cs`. Your answer should ignore the case of the letters.
16. It is necessary for you to protect some information from other people.  
Ensure that no-one can change the contents of `refs2.bs` although everyone can read it.
17. Sort the contents of `refs1.bs` and `refs2.bs` into alphabetical order, storing the result of each sort in `sort1` and `sort2` respectively.
18. There are some duplicate lines in `sort1`, remove duplicate lines from the file, storing the resultant information (ie minus duplicates) in `sort3`.
19. Find all the lines which are in both `sort2` and `sort3`.
20. Print out the files, `sort2` and `sort3` on the screen side by side.
21. Now see if you can repeat the last task while simultaneously sending the information to a file called "table" ie the information is printed both in a file and on the screen at the same time.
22. Delete the two files `sort2` and `sort3`.



## **Appendix C**

### **Help scripts**

<comms1.1>

The "who" command finds who is currently using the system. The response is the user's login name, the terminal name and when the user logged on.

If you ask "who am i" the system gives you information about who you are logged in as.

If you do not recognise a login name you can ask the system whois user name. The system responds with the persons full name and the group they belong to.

<comms1.2>

The "who" command tells you who is using the system.

who	finds who is using the system
who am i	finds who you are currently logged in as
whois lmc	finds the real name of user lmc

<comms1.3>

OPTIONS

-u also prints how long it has been since the last command  
-q prints the number of users logged in and their usernames.

<comms1.4>

No error messages.

<comms2.1>

The "write" command transmits lines from your terminal to another user providing that user is currently logged on.

The line is transmitted when you hit the return key. If the other user wants to reply to your transmission he can also issue the write command. Then messages can be sent between the two users. The end of this chat session is signalled by typing ctrl-d (End-of -input). This sends EOF to the other user.

<comms2.2>

The "write" command allows you to send an interactive message to another user.

write login-name

Example

```
write lmc
hello - is anyone there?
ctrl-d
```

<comms2.3>

There are no options to the command write.

<comms2.4>

:Writing to someone not logged on  
This means the person named is not currently using the system.

:Permission denied  
The person does not want to receive a message.

<comms3.1>

The 'mesg' command can be used to control attempts to interfere with your work by sending you messages using the write command.

The system default will allow messages to be sent to you but this is changed by using the argument n to stop the messages.

<comms3.2>

The 'mesg' command permits/stops messages from write.

mesg                   UNIX replies with 'is y' or 'is n' what ever the case may be.

<comms3.3>

mesg -n               forbids other users from communicating with you using write.  
mesg -y               allows others to communicate using write.

<comms3.4>

No error messages

<comms4.1>

The command 'mailx' followed by the usernames of the people who are to receive mail sends messages to those users.

After issuing the command you are prompted for the subject (topic) of the message. Input a topic and press <RET>.

After giving this the message is then typed in and ended by issuing ctrl-d on a separate line.

<comms4.2>

The 'mailx' command allows you to send mail.

mailx username(s)

Examples

mailx lmc               sends the message to lmc  
mailx lmc dbd           sends the message to lmc and dbd.  
mailx lmc < letter sends the contents of the file, letter, to lmc

<comms4.3>

No options.

<comms4.4>

\$mailx xxx

send-mail: unknown local user:xxx

This means that the system doesnot recognise the user name xxx

<comms5.1>

The command 'mailx' with no login-name argument allows you to display messages which have been sent to you.

There are options which can be used within the mail environment to read, save and delete the messages.

<comms5.2>

The 'mailx' command allows you to read mail sent to you.

mailx

This command opens the mailbox.

<comms5.3>

mailx -H : allows you to get a list of the messages in the mailbox.

These are the options to use within the mail environment, ie. in response to the mail: prompt. These are

.	display the current message
number	display message with this number
<Return>	display the next message.
s file	save the current message in file.
d	delete the current message from the mailbox.
x	exit from the mailbox without making any changes
q	leave the mailbox and make any changes required.

<comms5.4>

No error messages.

<end>

<diret1.1>

The full name of a file includes the name of the directory it is in. This is called the pathname of the file because it is the path through the directory system to the file. Pathnames can be full or relative.

The command "pwd" can be used to find your current full pathname.

The response to the command "pwd" is the directory you are currently in, ie. the full pathname

<diret1.2>

There are three ways to specify a pathname.

1. The full pathname specifies how to get to a place from root.

eg /usr/fsa/lmc/report/memo

Note: Root is indicated by the / at the beginning of the pathname.

2. The relative pathname specifies how to get to a place from where you are.

eg if you are in /usr/fsa/lmc then report/memo

3. If you require a file in the current directory then only the file name is required.

eg. file1

Thus file1 can be referred to as /usr/fsa/lmc/report/memo/file1 or file1.

pwd this prints the full pathname of the current directory.

<diret1.3>

There are no options to the command pwd.

<diret1.4>

No error messages.

<diret2.1>

"ls" displays the information about one or more files in a directory.

If no directory name is given as an argument, then the current directory is displayed with the files in alphabetical order.

This order can be changed with the use of an option.

If the prompt sign is returned and no other output, then the directory did not contain any files

<diret2.2>

The "ls" command lists the contents of a directory.

ls directoryname

#### Examples

ls	lists the current directory
ls memo	uses relative pathname to specify a subdirectory
ls /usr/fsa/lmc/memo	uses a full pathname to find memo

<diret2.3>

#### OPTIONS

-R	lists the contents of the named directory and contents of any subdirectory contained within it.
-a	lists normal files and those prefixed by a dot.
-t	lists files in order of the time they were last altered.

(Most recently changed file first)  
-l lists other information as well as the filename.  
This information is displayed in the following order  
file mode, no. of links, owner, group, file size, time of last change and name  
eg.  
ls -l produces output similar to that below.  
-rw-r--- 1 lmc staff 9121 Mar 3 12:11 file.name  
<diret2.4>

ls memo  
:file not found  
means that neither a file nor subdirectory call memo exists  
ls -P  
ls: illegal option --P  
means that -P is not a valid option to the ls command.  
<diret3.1>

The "cd" command allows you to switch between different directories.  
This is done by specifying the path from the current directory to  
the destination. Full or relative pathnames can be used.

If no pathname is given, you are moved to your home directory.  
<diret3.2>

The "cd" command allows you to switch directories.

cd pathname

Examples

cd takes you to your home directory (indicated by username)  
cd .. takes you back up one level of the directory  
cd /usr/fsa/lmc/reports uses a full pathname and can be used from  
anywhere within the system.  
cd reports uses a relative pathname and can only be used if reports  
is a subdirectory of the current directory.

<diret3.3>

There are no options to the cd command.

<diret3.4>

:bad directory

means that either the owner of the directory has denied you access  
or that a file rather than directory has been specified.

<diret4.1>

The "mkdir" command creates subdirectories from a given directory.  
This allows the formation of a hierarchical structure with the newly created  
directories being positioned one level below the current directory.  
Subdirectories can then be created from subdirectories and so on.

<diret4.2>

The "mkdir" command allows you to make a new directory.

mkdir directoryname(s)

### Examples

<code>mkdir reports</code>	creates a single subdirectory.
<code>mkdir reports memos</code>	creates two subdirectories.
<code>mkdir reports reports/briefs</code>	creates a subdirectory containing a subdirectory.

<diret4.3>

There are no options to the `mkdir` command.

<diret4.4>

`mkdir progs/c progs`  
`mkdir: cannot access progs/`

The directory `progs` must exist before a subdirectory is made for it thus these arguments are in the wrong order.

`mkdir progs`

`mkdir: cannot make directory progs`

means that a subdirectory called `progs` already exists.

<diret5.1>

A directory can be deleted from the system using the `rmdir` command. However, this command can only operate on empty directories, i.e. those which contain no files or subdirectories.

Therefore all files must be deleted using the `rm` command first.

This can be overcome by using `-r` option on the `rm` command which deletes all the contents of a directory followed by the directory.

<diret5.2>

The `rmdir` command allows you to remove an empty directory.

`rmdir directoryname`

Full or relative pathname of the directory to be deleted can be given.

### Examples

`rmdir /usr/fsa/lmc/demo`  
`rmdir demo - removes the subdirectory demo (if it is empty).`  
<diret5.3>

There are no options to the `rmdir` command.

<diret5.4>

`rmdir:dirname not empty`

The directory `dirname` is not empty.

All files and subdirectories must be removed first using `rm`.

`rmdir: dirname non-existent`

The directory `dirname` does not exist.

<diret6.1>

<diret6.2>

<diret6.3>

<diret6.4>

<diret7.1>

<diret7.2>





until ctrl-d is read.

cat file1 file2 > newfile  
will create a file called newfile which contains a copy of file1  
and file2 appended together.

cat file1>>another  
will add the contents of file1 on to the end of the file, another,  
assuming that the file, another, exists.  
<files3.3>

There are no options to the cat command when used in this manner.  
<files3.4>

No error messages.  
<files4.1>

The "pr" command allows you to layout a file for printing (either on screen or  
printer).  
The result of using this command is to separate the file into pages.

Each page has a 5 line header. This uses the filename as a title and gives you  
the page number and date.  
Each page also has 5 blank lines at the bottom.

This result is sent to standard output unless directed elsewhere.  
<files4.2>

The "pr" command allows you to format files.

pr filename            sends the formatted file to the screen  
pr filename >pretty   stores the formatted file in the file called pretty  
pr filename |lp        sends the result of the pr command to the line printer.  
<files4.3>

pr -h others people : changes the title of the file people to others.  
If the new title consists of more than one word, the words must be quoted.  
If the quotes contain nothing ie. "" then the title is eliminate,  
(only page numbers are printed).

pr -t filename : eliminates the header and bottom blank lines.

pr -m file1 file2 : will print the named files side by side on the screen.  
<files4.4>

No error messages.  
<files5.1>

The command "cp" file1 file2 creates a copy of the first file (file1)  
and gives it the name file2. If the file file2 already exists, then  
it is replaced by the new one.

Alternatively the "cp" command can be used to make copies of all the  
files listed and placed in a named directory.  
<files5.2>

The "cp" command allows you to make a copy of a file.

cp file1 file2 or cp file1 directory

#### Examples

cp first second makes a copy of the file first and calls it second.

cp first /usr/fsa/lmc/memo makes a copy of the file first and places the copy the in directory memo.

<files5.3>

No options to the cp command.

<files5.4>

If you try to copy a file which is write protected or if you try to copy a directory which does not exist you get the message:

cp: cannot create dir/file

If you do not have read permission on the file you are trying to copy you get the message:

cp: cannot open filename

If you are trying to copy a file which does not exist you get the message:

cp: cannot access filename

<files6.1>

The command "mv" changes the name of the file.

However it can also be used to move the file to a different directory.

<files6.2>

The "mv" command allows you to move or rename files.

mv file1 file2

#### Examples

mv file1 file2 changes the name of file1 to file2

mv file1 dir2 moves file1 into dir2, maintaining its name as file1

mv file1 dir2/file2 moves file1 into dir2 and changes its name to file2.

<files6.3>

No options to the mv command.

<files6.4>

No error messages.

<files7.1>

"rm" removes links to one or more files. When the last link is removed, you can no longer access the file and the system releases the space that file occupied on the disk for use by another file.

(ie. the file is deleted)

To do this you must have execute and write permission.

<files7.2>

The "rm" command allows you to remove files.

rm filename(s)

**Examples**

rm lyn                   removes the file lyn  
rm lyn tom dennis beth   removes the three named files

Pathnames can also be used to name the files.  
<files7.3>

**OPTIONS**

- i the system asks the user whether each file listed is really to be deleted.  
The user responds with y or n.
- r deletes a directory and every file and subdirectory in it.

**Example.**

rm -r sub   removes all files in the directory sub and then the directory.  
<files7.4>

If you try to delete a file which does not exist you get the message:  
rm: filename non-existent  
<end>

<ctrls1.1>

To get a control character, you hold down the control key (marked ctrl, cntr, cntrl or control, on left hand side of the keyboard), while simultaneously pressing the character you want. In documentation this is often expressed as ctrl-c  
<ctrls1.2>

The Line Kill = ctrl-u  
This is typed to tell the system to ignore the line typed so far. No new line is given, you just continue to type.  
<ctrls1.3>

Halting output temporarily = ctrl-s  
This suspends the screen scrolling process.  
<ctrls1.4>

Resume output = ctrl-q  
This resumes the suspended output.  
<ctrls1.5>

Stop a command = ctrl-c  
This completely stops a command in progress and returns the prompt.  
<ctrls1.6>

Signing off / end of input = ctrl-d  
This is used to signal the end of text input eg to mail and to indicate that the user wishes to end the session ie. logout.  
<format.2>

Wild cards give you the capability to use a short hand notation for operating upon groups of files and directories in a single command. For instance, they can be used to identify all files ending in .c

The characters are used to match filenames or parts of filenames.  
\* matches any character string, including the empty string.  
eg c\* all files beginning with c.  
? matches any single character  
eg c? all files whose name is two characters long and begins with c.

NB \* on its own matches all files, so take care when using it.  
<format.3>

The general format for commands is a sequence of words, with each word separated by one or more blank spaces. These words indicate the command name followed by arguments of the command. The end of the command is indicated by pressing the return key<ret>.

Arguments to Commands can be one of three things:

1. A filename (upto 14 characters long).
2. An option (introduced by a minus sign) which modifies the action of the command in some way.
3. An expression which describes a character string which is to be used as input to the command.

These are generally ordered as  
command options expression filename(s).  
<format.4>

Commands can be connected in one of two ways;

1. Several commands can be put on a single line.  
These commands must be separated by a semi colon (;).
2. The output of one command can be used as the input to the next.  
These commands must be separated by a vertical bar (|).  
This is known as piping.  
<format.5>

Arguments can be enclosed in either double or single quotes.  
In general it does not matter which ones you use, but if  
one kind is present in the string you must use the other.

eg. "Sally Smith's"  
<format.6>

The "date" command is used to display the current date  
and time on the screen.  
<end>

<manip1.1>

The "sort" command sorts and/or merges one or more text files into alphabetical or numeric order. The default is to sort alphabetically. An option must be used if numerical sort is needed.

The order of the sort is controlled by the options.

The "sort" command does not expect fields on a line to be in fixed columns instead it expects only that they are separated by spaces or tabs.

<manip1.2>

The "sort" command allows you to sort the contents of files.

#### Examples

sort filename           reorders the lines within a file into alphabetical order using the first word in each line as a basis.

sort filename > sorted   sends the sorted lines in the file called sorted  
<manip1.3>

#### OPTIONS

sort -n filename        treats all fields in the file as numerical and sorts smallest to largest.

sort -r filename        sorts in reverse alphabetical order

sort -rn filename       sorts in reverse numerical order (largest first)

sort -o newname oldname  sorts the oldname file into alphabetical order and stores the result in the newname file.

<manip1.4>

No error messages.

<manip2.1>

The "wc" command counts the number of lines, words and characters contained in one or more text files.

The options on this command restrict the counting to one of the object types.

The result appears on the screen and is the number of lines followed by the number of words then characters and then the filename. If more than one file name was specified then the result of each file count appear on separate lines followed by a total line.

eg.    10 30 200  file1  
      20 50 300  file2  
      30 80 500  total

<manip2.2>

The "wc" command counts the number of characters, words and lines in files.

wc filename(s)

#### Examples

`wc /usr/fea/lmc/memo`

`wc memo` :gives the number of lines, words and characters  
in the file memo.

(Full or relative pathnames can be used.)

`wc file1 file2` :gives the number of lines, words and characters  
in each of the named files and then totals them.

<manip2.3>

#### OPTIONS

`-l` restricts output to the number of lines  
`-w` restricts output to the number of words  
`-c` restricts output to the number of characters.

Options can be combined to eliminate one option type

#### Examples

`wc -lw file` restricts output to lines and words.

`wc -lwc file` is equivalent to `wc file`.

<manip2.4>

No error messages.

<manip3.1>

The "grep" command searches one or more files, line by line, for a certain pattern. Lines which contain that pattern are then printed on the screen.

The pattern looked for is not whole words eg if looking for paper then papers, papering etc will be found.

If the \$ appears and no reply then the pattern was not found within the file.

<manip3.2>

The command "grep" allows you to find a pattern in files.

`grep "pattern" filename(s)`

#### Examples

`grep lyn lyn.bib` : find the word lyn in the file,lyn.bib, and print the lines containing "lyn" on the screen.

`grep "two words" filename` : if pattern contains a space then it must be quoted. The reply is the line containing pattern.

`grep pattern file1 file2` : searches the two named files.

`grep pattern *.f` : searches all files in current directory ending in .f.

The reply is the filename followed by the line containing pattern.

<manip3.3>

#### OPTIONS

`-n` gives the line number followed by the line.

`-c` gives a count of the number of lines containing pattern.

Note : It does not print the line.

`-i` means that the command ignores the case of the letters in pattern.

**Examples**

`grep -ni pattern file1` will print the lines preceded by their numbers.  
<manip3.4>

No error messages.

<manip4.1>

The 'uniq' command compares adjacent lines in a file and removes any duplicate lines. The result is printed on the screen or stored in a file. To work effectively the file should be sorted into an order which puts the duplicates together.

The options are used to determine which lines are saved and which are eliminated.

<manip4.2>

The 'uniq' command eliminates duplicated lines from a file.

`uniq infile outfile`

**Examples**

`uniq infile` prints a copy of the lines (minus duplicates) on screen  
`uniq infile outfile` stores a copy of the lines (minus duplicates)  
in the file outfile.

<manip4.3>

**OPTIONS**

-u selects only the lines which have no duplicates.  
-d selects only those lines which have duplicates.

**Examples**

`uniq -u infile outfile` : stores the lines from the file infile which  
have no duplicates in the file outfile.  
`uniq -ud` is the same as `uniq` with no options.

<manip4.4>

No error messages.

<manip5.1>

The 'comm' command compares the lines in two files and to find which ones are common to both. However the command produces output which also tells you which ones are not common.

The command produces three columns of output. The first tells which lines are in the first file but not the second. The second tells which lines are in the second but not the first and the third tells which lines are common to both.

To be effective the contents of both files should be stored in the same order.

The options to the command are used to suppress different columns of output.

<manip5.2>

The 'comm' command allows you to find lines common to two sorted files.



comm file1 file2

**Examples**

comm first second : compares first with second producing the output as:  
the first column is all lines in first file which are not in second  
the second column is all lines in second file which are not in first  
the third column is all lines which are in both files.  
<manip5.3>

**OPTIONS**

- 1 suppresses column 1 ie lines in file1 which are not in file2
- 2 suppresses column 2 ie lines in file2 which are not in file1
- 3 suppresses column 3 ie lines which appear in both files.

**Examples**

comm -1 file1 file2 : prints lines in file2 not in file1 and lines in both.  
comm -12 file1 file2 : prints only lines in both files.  
<manip5.4>

No error messages.  
<end>

<gener1.1>

There are two ways to protect your workspace and the files within it from other users. The first is to stop people using your user account by setting a password. This does not stop people accessing your files from their own identification and therefore protection can also be placed on individual files and directories.

<proct1.2>

<proct1.3>

<proct1.4>

<proct2.1>

The "passwd" command is used to change your password. After issuing the command you will receive a number of prompts. The first is for your current password. If this is input correctly, you will then be asked for your new password. If the password given is accepted you will then be asked to repeat the new password. The old and new password must be sufficiently different from each other to allow the change to occur.

If the process has been completed successfully, the \$ prompt is returned.

<proct2.2>

The "passwd" command allows you to set/change your login password.

passwd

Generally passwords are at least 6 characters long, with at least one alpha numeric.

eg pig666 cowboy

<proct2.3>

No options to the passwd command.

<proct2.4>

Sorry.

means that you didnot get the old password correct.

They don't match; try again

means that you didnot type the same new password on both attempts.

Too many tries; try again later.

means that you have had three unsuccessful attempts to change passwords

Too short. Password Unchanged.

means that you have used too few characters (less than 6).

<proct3.1>

"chmod" gives you a means to assign protections to files and directories. "chmod" either adds (+) or removes (-) permission for three classes of users; the owner(u) - usually the person who created the file, the group(g) - all others in the same group as owner eg undergrads and the public(o) - all others who have access to the system.

Every file/directory has three types of permission;

read (r) : allows users to look at the contents of a file/directory.  
write (w) : allows users to change the contents of a file.  
          remove it or copy it from a directory.  
execute (x) : allows users to use the file as a command  
             copy it from a directory or move to it.

<proct3.2>

The "chmod" command allows you to change permissions on files.  
chmod instruction-string file or directory.

chmod [ugo+-rwx] name

Example

chmod go-w filename  
      Removes write permission from everyone  
      except the owner.

Note: go-w with no spacing, forms the instruction string.  
<proct3.3>

OPTIONS

The options are not started with a minus and there is no spacing.  
This forms an instruction string.

u user  
g group  
o other users  
+ add permission  
- remove permission  
r read permission  
w write permission  
x execute permission

Example

chmod ugo-w remove write permission from everybody

<proct3.4>

chmod : can't access nosuch.  
- the file nosuch does not exist.

chmod : can't :change notmine.  
- you do not own the file notmine.

chmod : invalid mode.  
- wrong syntax or arguements in the wrong order

<proct4.1>

<proct4.2>

<proct4.3>

<proct4.4>

<end>

<stands.1>

When issuing a command, System V usually expects input(data) and produces output (results). If no file is specified, then it expects keyboard input (standard input) and outputs the results to the screen (standard output).

Input and output can be redirected either to another file or the output of one command becomes the input to the next (piping);  
<stands.2>

To redirect input to come from a file rather than from the keyboard, the filename is prefixed with a less than sign,<. eg mail <message  
<stands.3>

To redirect the results of a command from the screen (standard output) to a file, the filename argument to the command should be prefixed with a greater than sign,>. eg. ls -l > dirconts

NOTE: If the filename already exists then its contents will be over written.

If you don't want to lose the contents of the existing file, but want to add the output on to the end of it(append), prefix the name with >>. eg ls -l >>dircont

To redirect the output of a command to become the input to another command, the pipe (|) should be used, eg ls -l | wc  
<stands.4>

Sometimes it is necessary to save the output of a command to file while still seeing what is happening on the screen. This is achieved by using the "tee" command which splits the output.

eg. ls -l | tee dirconts stores the result of ls -l in the file dirconts and also prints it on the screen.  
<end>

## Appendix D

### Evaluation statistics

Table D.1 gives the raw scores of each of the subjects. The details in each of the columns are as follows:

1. The training group, 1 represents the Womens Technology Centre and 0 the University.
2. The experimental system, S represents the static system, A the active and B the joint system.
3. The gender, 0 represent female and 1 male.
4. The field category, 1 represents field-dependent and 0 field-independent.
5. The score on the EFT, the average number of seconds to complete a task.
6. The number of commands known by a subject.
7. The number of commands known well by a subject.
8. The number of keystrokes while within the passive help system.
9. The number of subject requests for the passive help system.
10. The number of interrupts by the active help system.
11. The number of tasks left incomplete.
12. The total time spent doing the tasks.

Table D.2 contains the significance results for the analysis of variance.

1	S	0	1	49.83	5	1	65	10	0		8	34
1	A	0	1	41.67	8	8			7	1	6	25
1	B	0	1	45.89	8	6	78	8	1	1	6	26
1	S	0	0	14.6	6	5	61	7	4		6	18
1	A	0	1	40.08	7	7				6	6	31
1	B	0	1	98.58	6	2	46	6	2	4	10	29
1	S	0	0	12.8	8	8	40	6	1		6	16
1	A	0	1	102.42	7	5			5	4	8	25
1	B	0	1	44.42	6	3	45	6	4	0	8	20
1	S	0	1	31.5	5	5	55	8			2	23
1	A	0	1	60.5	6	4			6	3	8	26
1	B	0	1	71.83	6	4	20	3	8	3	8	30
0	B	1	1	38.25	7	5	131	15	0	2	2	60
0	S	1	0	5.67	16	11	96	12	4		0	45
0	B	1	0	14.58	17	12	50	8	0	2	0	36
0	S	1	0	15.83	16	8	77	9	1		2	40
0	B	1	0	12.67	9	5	118	15	6	2	2	65
0	S	1	0	7.5	14	9	86	10	8		0	47
0	S	1	1	51.92	10	7	163	17	0		2	52
0	S	1	1	17.92	16	13	67	10	5		0	39
0	B	0	0	11.5	10	5	127	17	0	2	0	65
0	B	1	1	23.67	10	5	84	11	12	2	0	58
0	A	1	0	5.92	16	14			1	5	2	29
0	S	0	0	11.33	3	2	145	20	0		6	43
0	A	1	0	11.58	12	8			5	8	0	38
0	A	1	0	12.58	17	12			3	9	0	31
0	A	1	0	11.67	17	14			3	9	0	30
0	A	1	1	25.25	10	7			5	11	0	38
0	A	0	0	16	6	6			4	5	13	45
0	B	1	0	13.67	9	6	83	12	1	3	1	46
0	S	1	1	20.58	10	8	83	13	9		1	42
0	B	1	0	6.25	14	12	70	11	1	3	0	37

Table D.1: Subject's raw scores

<i>Breakdown</i>	<i>F value</i> (2,29 df)	<i>Significance</i>
field by system	.7999	.4590
helps by system	.2440 (1,20 df)	NS
helpkeys by system	.2464 (1,20)	NS
interrupts by system	15.5060 (1,19)	.0009
commands known by system	.2417 (1,30)	NS
commands expert by system	1.3365 (1,30)	NS
tasks incomplete by system	.2036 (2,29)	NS
time to complete by system	2.0732 (1,30)	.1440
field by field	32.5604 (1,30)	.0000
helps by field	1.0132 (1,20)	NS
helpkeys by field	.4528 (1,20)	NS
interrupts by field	1.2212 (1,19)	NS
commands known by field	8.4820 (1,30 df)	.0067
commands expert by field	6.8817 (1,30)	.0136
tasks incomplete by field	4.1181 (1,30)	NS
time to complete by field	.9785 (1,30)	.3305
field by gender	11.7020 (1,30)	.0023
helps by gender	2.592 (1,20)	NS
helpkeys by gender	2.5986 (1,20)	NS
interrupts by gender	3.1059 (1,19)	NS
commands known by gender	43.8235 (1,30)	.0000
commands expert by gender	21.725 (1,30)	.0001
tasks incomplete by gender	79.2331 (1,30)	.0000
time to complete by gender	9.7068 (1,30)	.0040
field by group	23.637 (1,30)	.0000
helps by group	20.2844	.0002
helpkeys by group	14.0509 (1,20)	.0013
interrupts by group	2.6302 (1,19)	NS
commands known by group	19.6590 (1,30)	.0001
commands expert by group	10.8766 (1,30)	.0025
tasks incomplete by group	35.7375	.0000
time to complete by group	32.0152 (1,30)	.0000

Table D.2: F value and significance level for Analysis of variance