

COMMAND PROCESSORS FOR THE DYNAMIC CONTROL
OF SOFTWARE CONFIGURATIONS

by

ROXANNA MAY FUNDIS

B.M., Kansas State University, 1974

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

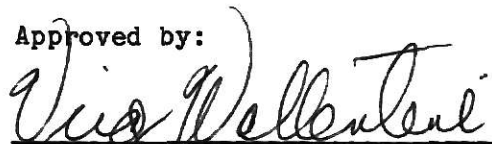
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1980

Approved by:


Major Professor

SPEC

COLL

LD

2668

.R4

1980

F86

C.2

TABLE OF CONTENTS

ILLUSTRATIONS	iii
ACKNOWLEDGMENTS	iv
INTRODUCTION	1
Chapter	
I. NADEX FACILITIES	3
II. THE DO COMMAND PROCESSOR	12
III. THE STATIC COMMAND PROCESSOR	15
IV. THE UNIX COMMAND PROCESSOR	21
V. THE MIRACLE COMMAND PROCESSOR	32
VI. SUMMARY AND CONCLUSIONS	43
Appendix	
A. SYNTAX GRAPHS	50
B. NADEX NATIVE PREFIX	61
C. SAMPLE USER SESSIONS	63
REFERENCES	72

ILLUSTRATIONS

1. PCD1	5
2. PCD2	5
3. PCDHIER.	5
4. PCDSAME	6
5. PCD3	6
6. Resulting Configuration of PCD2 ! PCD3	6
7. Command Processor Configuration	10
8. SOLO Configuration	13
9. Execution of EDIT(CARDS,TAPE)	13
10. Examples of OS/32 MT File Descriptors	16
11. NADEX Static Fast Commands	17
12. NADEX Static Commands with Parameters	20
13. UNIX Commands and Resulting Configurations	22
14. UNIX Commands using Sequencing Operators	24
15. Examples of UNIX Fast Commands	26
16. Hierarchical Nodes in CMD.RES	29
17. Resolved PCD Files written out by Command Processor	30
18. Completed Command Configuration	31
19. Comparison between UNIX and MIRACLE Commands	34
20. Simulation Configuration	35
21. Dining Philosophers	36
22. Examples of MIRACLE Fast Commands	38
23. Comparison of Command Processors	46

ACKNOWLEDGEMENTS

This research was supported in part by the Army Institute for Research in Management, Information, and Computer Systems under grant number DAAG 29-78-G-0200 from the Army Research Office.

Special thanks go to Dr. Virgil Wallentine, Dr. Elizabeth Unger, Dr. Paul Fisher, Quincy, and the OS Group for their contributions to this work in the areas of critiques, moral support, picture drawing programs, and the Symbolic Debugger.

INTRODUCTION

Command language facilities for the construction and execution of software configurations--networks of communicating processes--are very limited today because current operating systems do not support this level of complexity. The Network Adaptable Executive (NADEX)[11] is an operating system which was designed to support dynamic configurations--those configurations which are constructed at command interpretation time--of cooperating processes. These dynamic configurations may be composed of arbitrary graphs which can contain cycles. Three command processors have been developed to explore the sufficiency of the NADEX facilities to support dynamic configurations. Users' guides to these command processors will be presented in this document, together with syntax graphs and sample user sessions.

A command allows the user to query the state of a program and/or the computer and to manipulate its resources. For example, file maintenance may be achieved through a command such as

```
DELETE MYFILE
```

which deletes a file named MYFILE from the file system; or programs may be executed with a command like

```
PAS32 MYFILE
```

which compiles the Pascal program named MYFILE. The program which reads and interprets these commands as requests to execute operations or other programs is commonly known as a command processor, command interpreter, or shell.

In chapter I, NADEX facilities, an overview of the Job Control System, and the command processor configuration are presented. Much of the basic structure of the NADEX command processors is taken from a command processor called D0 which was written by Per Brinch Hansen[3]. This command processor originally ran under the Solo[3] operating system and the parts of it that prove relevant to further command processor development are described in chapter II. Although the Solo operating system does run under NADEX, it is not used for practical purposes. Chapter III contains a description of the NADEX version of D0.

Since the UNIX* shell[2] seems to be the only commercially available command processor that allows the user to dynamically configure commands, a small subset of the UNIX command processor was implemented to test the sufficiency of and demonstrate the use of NADEX facilities. This subset of UNIX is documented in chapter IV. In chapter V, the NADEX implementation of Gray's MIRACLE[5] (Machine Independent Resource Allocation and Control Language) is described. This network command language supports named ports which allows arbitrary configurations to be constructed. Implementation features and conclusions are found in chapter VI. Syntax graphs for the command languages, the NADEX Native Prefix, and sample user sessions are listed in the appendices.

*UNIX is a trademark of Bell Laboratories