MTA SZTAKI Közlemények 39/1988 p. 111-120.

SOFTWARE ENVIRONMENT FOR DISCRETE MATHEMATICAL RESEARCH IN UNIX-LIKE SYSTEMS

K.N. MANEV

Institute of Mathematics, Bulgarian Academy of Sciences

1. INTRODUCTION

The necessity of computers in the scientific researches is obvious nowadays. A lot of problems of the discrete mathematics, for examole, need a computer treatment. The computer experiments became natural, preliminary or final, step of the investigation of some discrete configurations and thus, a part of the reasoning of new results. That is why the requirement towards software for mathematical researches are very strong.

In this paper a general aproach to software environment for discrete mathematical researces in UNIX-like systems is described. It sumarized, also, the autor's experience in implementation of some components of such an environment in UNIX System V and the obtained results.

2. UNIX-LIKE SYSTEMS

The choice of the opperating system is not accidental. The UNIX-like systems possess a lot of usefull features, some of them listed below:

- the UNIX-like systems are, practically, standartized nowadays and are installed in all wide-spread computer's architectures. This allows the exchange of software between researchers and minimizes the efforts for

development:

- the UNIX-like systems have clear multylevel structure (see Fig.) with appropriate interfaces in the differnt levels. so that any user (system analyst, researcher which **now** few programming or such that never will have been acquainted enough with the computer) can ooperate inside the environment on the corresponding level;

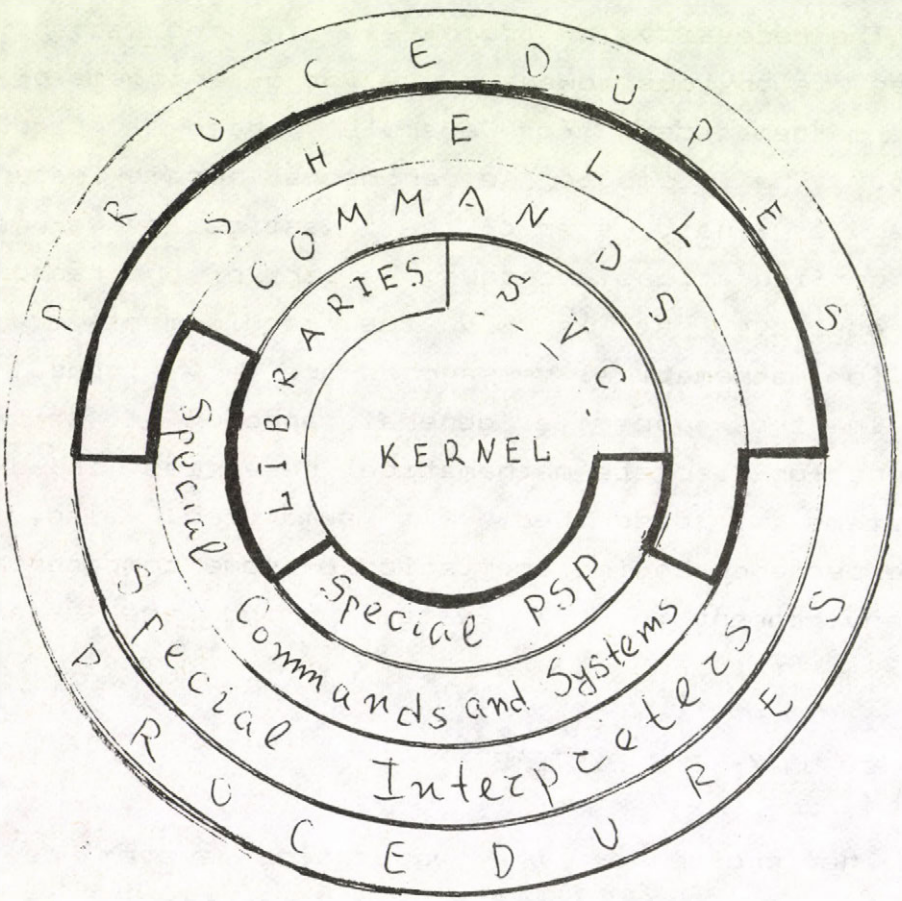

Fig.

- the UNIX-like systems support compilers, practically standartized, from the high-level programming language C, which allows effective usage of the computer's

possibilities - pointers, bit and shift operations and so on. All dependences from the computer's architecture (the size of the computer's word, for example) can be defined as parametrs for the preprocessor of the language. So, the UNIX-like systems provide us with such portability of the software, which is of a big importance for our purposes:

- the UNIX-like systems communicate with users throw the command interpreter, which control running of the executable modules (commands). Any program, written from the, user becomes command. Moreover, the interpreter, itself, is also a command, and in a such a way is not obligatory. The user can provide an own interpreter. The standard command interpreter - shell, combines command with operators like this in C language: for, while, case and if. Whit the stancard commands echo (prints its arguments), read (reads values in its arguments), expr (evaluates an expression, formed by its arguments) and test (evaluates various predicates) shell becomes a programming language. Each procedure writen in this language is considered as a command, i.e. can be included in new procedures and so on.

3. SOFTWARE ENVIRONMENT

Our general idea is the next: to follow the multylevel structure of the operating system and to append on every level the corresponding tools for discrete mathematical researches (see Fig.)

The kernel of the UNIX-like systems is good enough for our purposes. It provides the supervisor calls (SVC), necessery for construction of an environment, dedicated to scientific researches of discrete mathematical objects. Moreover, appending of new SVC will decrease portability of the environment and the possibilities for exchanging of programs and results.

That is why we will not touch the kernel of the operating system.

The variety of architectures, operating systems and various versions, even for well standartized programming languages, did not permit wide-spread use of package standard programs (PSP). The UNIX-like systems give us the possibility for creating portable PSP. Moreover, the portability can be reached following a few simple rules:

- use only standard features of the C language;
- define any machine dependent constant on the preprocessor's level;
- use as subroutines only standard S/C, standard function from the system's libraries or programs from the package (which follow this rules, of course).

It is very helpfull to construct the package by the approach of abstract data type (ADT), i.e. to choose an ADT, appropriate for the goals of the package and to build the programs like operations with this ADT. In such a way, one can easy append new operations (programs) and change existing programs by ones, which implement better algorithms (and no needs to change the software which uses the package).

The PSP have to be created from very experienced system programers, which know very well the features of the operating system, C language and the object of the researches, too.

On the next level there are two possibilities. The first: any researcher, which knows the C language and can formulate a problem in the terms of the operations with the ADT, can write easy a program for resolving the problem, which use the PSP and with the help of a more experienced programer to create even complex systems, with apropriates languages, for resolving a class of problems.

The second possibility is: some standard programs to be transformed into commands of the operating system.

Although the user's systems and the commands lie in the same level of the structure of the operating system we will distinguish them. The commands are more simple and, as a rule, they have not a proper language. They perform usually one step of the resolving of the problem. The using of commands will be discussed bellow. The transformation of the standard programs into commands is more then trivial by the provided scheme for processing the arguments.

On the next level we propose the construction of new command interpreters. The standard command interpreter of the UNIX-like systems - shell, combines commands with the control constructions of the C language - for, while, case and if. For researchers, which dont know the C language there is only one possibility to work on their problems - sequential executing of commands. But these users know better others ways of composition of primitives - superposition of formulae, logical inference and so on.

That is why we propose on this level construction of new command interpreters, more appropriate for mathematicians. The UNIX-like systems encourage such activities, providing tools for automatically generating lexical analizers and parsers, starting from the corresponding formal gramars. Having such interpreters and enough commands (created on the previous level) for the special subject, even not experienced in programming user can formulate problems for resolving by computer.

What's about the mathematicians, which have not experience to work with computers at all. For these users we will create procedures in the languages of the command interpreters and the users could resolve their problems executing this procedures (giving appropriate values to the parameters).

Thus, we obtain an environment corresponding to the structure of the opperating system, which give to any

user the possibility to determine the proper level of experience and to work on his problems with the appropriate tools. This environment is very flexible. It can be easy modified and developped, without changing the existing software.

4. SOME RESULTS

Some of our ideas were implemented in UNIX System V on the 32-bit (M68010) computer, with 2 Mb opperating memory - ICL CLAN. Here we briefly discuss the obtained results.

The portable package standard programes COMPACK [1] for combinatorial researches was created arround the ADT called a generated binary matrix (GBM). This ADT is appropriate for representing a very large set of combinatorial objects - graphs, matroids, block-designs, error-correcting codes, etc. The package contains programs, which operate with GBM, dosen't matter which object they represent, and an amount of programs appropriate for combinatorial researches. Many programs from COMPACK were transformed into commands of the opperating system.

The package GFPACK for operations with elements of Galois fields, polynomials and matrices over Galois fields is now under develeopment.

Using COMPACK, a special system - LINCOR, was constructed for computer researches of error-correcting codes [2]. For this purpose the package was extended with programs, appropriate for the special subject and we have enogh experience to extend the package in other directions, too.

The system LINCOR was designed to escape the everyday, routine work. Nevertheless, serious problems were resolved with it, connected with the covering radius of error correcting codes. It was refuted the conjecture

that all binary linear codes, containing all-ones vector meet the lower bound for the covering radius [3] and the covering radius of 73 from 76 cyclic bynari codes with length 33, 35 and 39 were found [4].

Three languages for new command interpreters was proposed [5]. The first of them is extracted from Backus' FP-systems and combine comands in a functional style. The second is extension of the language Prolog, which treat both, commands and files, like predicates. The third is an original language, which combines commands by the usual scheme "backtrack" and it is oriented toward the searching of combinatorial configurations. All these languages are under experimental implementation in UNIX System V.

REFERENCES

1. K. N. Manev, A Portable Package of Standard Combinatorial Programs, Proceedings of the Scientific Seminar "Problems of the Portability of Programming Tools", Varna, 1987.
2. K. N. Manev, LINCOR - A System for Linear Codes Researces, in "Mathematics and Education in Mathematics", Proceedings of the Sixteen Spring Conference of the Union of Bulgarian Mathematicians, Sunny Beach, 1987.
3. S. M. Dodunekov, K. N. Manev, V. D. Tonchev, On the Covering Radius of the (14,6) Binary Codes, Containig The All-ones Vector, IEEE, Trans. on Inform. Theory, 1988 (to appear).
4. E. Velikova, K. Manev, The Covering Radius of Cyclic Codes of Length 33, 35 and 39, Annuaire de l'Université de Sofia, t. 81, l. 1 - Mathematique, 1987 (to appear).
5. K. N. Manev, Automatization of the Combinatorial Researches, Candidate disertation, University of Sofia, 1987.

SOFTWARE ENVIRONMENT FOR DISCRETE MATHEMATICAL RESEARCH IN UNIX-LIKE SYSTEMS

K.N. Manev

Summary

In the paper a general approach to software environment for research in discrete mathematics in UNIX-like systems is described.
The paper summarizes the author's experience in implementation of some components of such an environment in UNIX System V and a review of results is presented. The programs written are in a great extent independent of the combinatorial structure of the problems. However better results can be achieved using some additional sub-bases for particular problems (say coding theory).

DISZKRÉT MATEMATIKAI KUTATÁSOK SZOFTVER KÖRNYEZETE UNIX TIPUSU RENDSZEREK HASZNÁLATÁVAL

K.N. Manev

Összefoglaló

A cikkben a szerző egy általános módszertant mutat be, amely diszkrét matematikai kutatások szoftver környezetének megteremtésére vonatkozik UNIX-tipusu rendszereket felhasználva. A szerzőnek gazdag tapasztalata van az UNIX System V rendszerek esetében.
A cikkben bizonyos eredmények áttekintése is megtalálható. A megirt programok nagymértékben függetlenek a konkrét kombinatorikai strukturától, de jobb szoftver környezet is kialakitható konkrét esetekben /pl. a kódelméletben/.