Computer Corner

XINBASE, a database system in the Unix shell

C. R. Muthukrishnan and Jai Kumar
Madras, India

Recent advances in implementing user-friendly systems advocate tools for rapid prototyping. Interactive access to a database using the relational model is a powerful basis on which to develop such a tool. This work presents an integrated set of tools using shell language under Unix and employs the utilities offered by it.

The system, named XINBASE, is a flatfile database system. It combines the general Unix tools (commands) into specialized tools (operators) and is presented in a menu-driven, user-friendly environment. A number of Unix commands (60 of them) have been combined to emulate 21 basic database operations on the lines of the popular dBASE II package.

Data are stored at two levels (pools) to allow for locking, security and simple error recoveries to be incorporated. The results of operations on the flatfiles are sent to temporary files and the user has the option to direct it to named files or to overwrite the original. XINBASE is designed to show that efficient application programs can be developed in the Unix environment.

Introduction

The Unix environment is productive because it provides a clean and systematic interface to programs that run on it. Unix has a repertoire of small and useful programs[1] (called utilities or tools) that can be combined into complex procedures using the Unix shell[2]. XINBASE is a paradigm of such a combination. The existing utilities (60 of them) are connected at a very high level to perform 21 database functions. In this system, each function is called an 'operator'. XINBASE is a relational database system, the relations being in the form of tables. Each line of the table constitutes a record. Embedded within records are user-defined fields, demarcated by field separators[3]. The XINBASE operators manipulate data files consisting of these tables to produce the desired results.

Design of XINBASE

In addition to being a command interpreter, the Unix shell[2] is also a programming language with variables, control flow, subroutines (calling other programs) and interrupt handling features. It permits the user to create programs in a straightforward and well structured manner. These programs constitute a combination of Unix commands and can be tailored to meet specific requirements.

XINBASE is written in the shell language. Each XINBASE operator is a customized shell script which can be invoked via a menu display. Fig. 1 illustrates the structure of XINBASE.

Variable substitution forms the crux of XINBASE operations. A dialogue is initiated between the user and XINBASE. Based on the response, variables are set and passed as arguments to the appropriate operator. The operator then executes sequentially the commands contained in its constituent shell script file. Depending on the task to be performed, each line in the operator file could be a simple command or another shell script containing more commands. This is analogous to the procedure calls of conventional programming languages. Fig. 2 shows the organization of operators and the flow of data in XINBASE.

In addition to its display on the terminal, the output of an operation is sent to a file in the /tmp directory through pipes or I/O redirection. This technique is useful for two reasons. Firstly, it is possible to 'undo' the effect of an operation by instructing XIN-

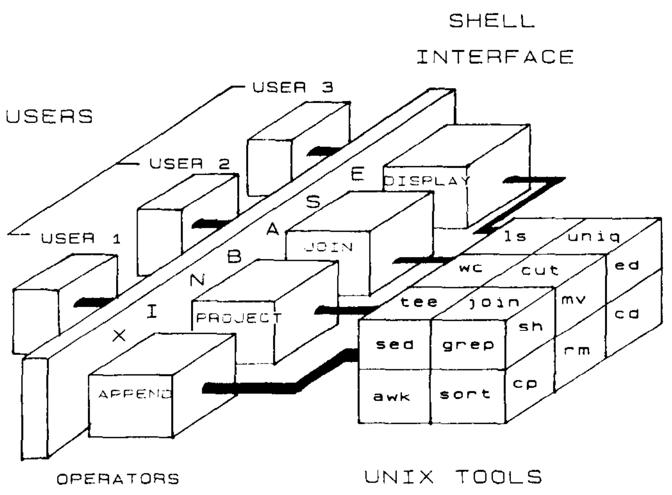

Fig. 1. Structure of XINBASE.

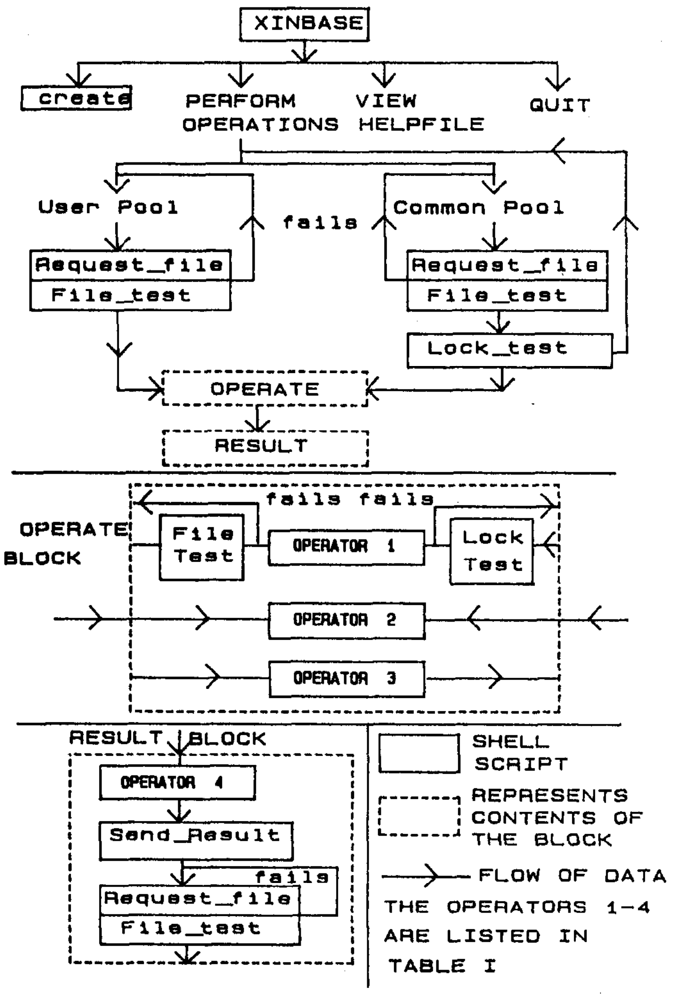

Fig. 2. Organization of operators and dataflow in XINBASE.

BASE to simply delete the temporary file. Secondly, the temporary file can be named at the user's discretion. The use of temporary files also ensures that original files are not accidently overwritten.

Pools and locking

XINBASE allows data to be stored at two levels (pools): the user pool and the common pool (Fig. 3). The user pool contains those files that are local to the user's directory. Files in the common pool can be accessed by all users.

This leads to a need for concurrency control[4] when two or more users request an update on the same file. Mutual exclusion is maintained by the 'lock' command. 'lock' is implemented using the Unix system call 'creat'[5] by allotting a unique temporary file to each user (the file semaphore protocol[6]). When the user completes an operation in the common pool, XINBASE 'unlocks' the file with the 'unlink'[5] system call. Since 'creat' is atomic, only one user would succeed in gaining control over a file from amongst all those users who try to access it simultaneously.

If a user fails to lock a file, XINBASE issues a message requesting the user to try again later. When a user quits XINBASE, all locked files are automatically unlocked to prevent indefinite waiting on the part of other users.

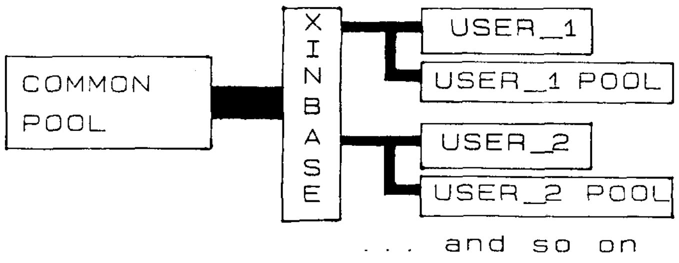

Fig. 3. Data pools in XINBASE.

File types and field types

The files in XINBASE exist with one of the following suffixes:

.db. Data file, contains fields separated by delimiters.

.fdb. Format file, contains information about field names, field types and widths.

.cdb. Crypted file, contains a coded version of the corresponding *.db* file.

.pdb. Print file, contains formatted *.db* file.

XINBASE allows the following field types:

a. Alphanumeric.

n. Numeric.

l. Logical.

Operators

The operators contained in XINBASE and their file requirements are given in Table I. A brief description of the XINBASE operators and some of the Unix commands (given in parentheses) used by each of them will now be given.

TABLE I. Operators and file requirements in XINBASE (Fig. 2)

Definitions: type 1 = operators requiring two or more files; type 2 = operators requiring only one file; type 3 = operators of type 2 which transfer files to the common pool; type 4 = operators of types 1 and 2 whose results are sent to temporary files.

type 1	type 2		type 3	type 4
compare	append	change	transfer	delete
join	create	copy		display
project	duplicate	display		duplicate
	delete	insert		edit
	edit	modify		insert
	locate	sort		join
	rename	sum		locate
	print	transfer		modify
				sort
				sum
				project

append. Adds new records or fields to the end of a database file or a format file ('ed'[1] and the 'here document technique'[2]).

change. Modifies the access rights to a file ('chmod'[1] and 'crypt'[1]).

compare. Compares and lists the differences between two datafiles ('comm'[1], 'cmp'[1], 'compare'[1] and 'diff'[1]).

copy. Duplicates a datafile to a new file ('cp'[1]).

create. Creates a new datafile structure ('cat'[1] and 'echo'[2]).

delete. Removes records or fields selected by their numbers or by user specified patterns ('rm'[1], 'grep'[1] and 'awk'[7]).

display. Prints a formatted datafile ('awk'[7]).

duplicate. Lists duplicate records and removes all except one occurrence ('uniq'[1]).

edit. Invokes one of the three editors ('ed'[1], 'vi'[1] or 'spy'[8]).

help. Displays the XINBASE help file.

join. Combines two datafiles on a common field ('join'[1]).

locate. Searches the database for records that match a specified pattern ('fgrep'[1], 'egrep'[1], 'grep'[1] and 'awk'[7]).

modify. Allows the modification of a field structure or the contents of a datafile interactively ('ed'[1], 'sed'[1], 'grep'[1] and 'awk'[7]).

quit. Terminates the XINBASE session ('exit'[2] and 'break'[2]).

insert. Adds a record or field at a specified location ('fgrep'[1], 'egrep'[1], 'sed'[1] and 'awk'[7]).

rename. Changes the name of a datafile ('mv'[1]).

sort. Sorts a datafile on any field or fields ('sort'[1]).

project. Projects fields from one or more datafiles ('cut'[8] and 'paste'[8]).

print. Prints a datafile according to a chosen specification ('roff'[1], 'nroff'[1] and 'tbl'[1]).

sum. Performs totals on numeric fields ('awk'[7]).

transfer. Sends a copy of files from the user pool to the common pool ('mv'[1]).

Discussion

XINBASE was designed with several goals in mind:

(a) to demonstrate the simplicity of building databases. Much less effort is required compared to using a conventional programming approach;

(b) to lay stress on the development of a cohesive, integrated user-friendly and automated environment;

(c) to increase both programmer productivity and application reliability;

(d) the idea of exploiting the potential of existing Unix utilities instead of writing programs from scratch.

Performance criteria

It is well known that Unix commands have been designed to provide maximal efficiency in execution times[1,5]. Since each XINBASE operator is a combination of such commands, the overall performance is limited by the following factors:

(a) variable assignment and their passing to programs written in shell and awk result in an overhead in processing;

(b) the inherent response delay in any timesharing system — more so in the case of Unix;

(c) the redirection of output to temporary files reduces performance speed but ensures data integrity.

Conclusions

The flexibility of developing programs in the shell facilitates rapid prototyping. A prototype could be built, experimented with and tailored to perform a particular task — all with minimal investment of time and effort. This methodology proved to be very beneficial in the design phase of each XINBASE operator. Shell programs do not need compiling and they produce immediate results. This allows the user to test a shell script by varying input specifications, modifying and retesting until a streamlined and efficient program is developed.

We conclude that the XINBASE approach to application program development proves to be reliable, efficient and requires much less coding than programs written in conventional programming languages.

References

1 *Unix Programmer's Manual,* Vol. 1, Bell Telephone Labs., Inc., New Jersey, 1983.

2 S. R. Bourne, *Bell Syst. Tech. J.,* 57 (1979) 1971–1980.

3 P. A. Bailes, *Software Practice and Experience,* 15 (1985) 1011–1020.

4 J. Ullman, *Principles of Database Management,* Computer Science Press, Potomac, 1981.

5 B. W. Kerninghan and D. M. Ritchie, *The C Programming Language,* Prentice Hall, Englewood Cliffs, NJ, 1977.

6 M. Rochkind, *Advanced Unix Programming,* Prentice Hall, Englewood Cliffs, NJ, 1985.

7 A. Aho, B. Kerninghan and J. Weinberger, *Software Practice and Experience,* 9 (1979) 267–279.

8 *Guide to PNX, Appendix-3: Additional Software Specifications,* ICL, U.K., 1984.

C. R. Muthukrishnan is a Professor and Jai Kumar is a research scholar at the Department of Computer Science, Indian Institute of Technology, Madras 600 036, India.