<HiFi>

The
High Files
Filesystem

Prospectus:

CONTENTS

{brochure}

preface

High Files is a high-level filesystem, intended for casual users of computers. The structures which it supports are sufficient for storing and manipulating large amounts of information, all powerfully accesible thru editors.

A file in High Files is either a table or an outline. Both are highly intuitive versions of constructs which are used to represent information in hard-copy reference sources such as encyclopedias, almanacs and newspapers. We have included a sample of a table and an outline, both prepared using the High Files system.

The system has not been publicly released, although experimental (no warranty implied) copies will be made available soon.

the table editor

The table editor is built on top of a relational database management system, giving the editor a powerful query facility with a very convenient command syntax. Populating a table is exactly analogous to "appending text" using a conventional line-oriented editor. The user can easily re-format a table, add a row, modify the values stored in an entire column - all thru editor commands.

the outline editor

computers can deal with outlines

We were all taught to use outlines in junior high school. But outlines were only practical for people who had fantastic memories. One also had to be quite patient and do alot of writing and copying over. The problem is not in the method, on the contrary outlines are a very natural idea. The problem is that pencil and paper (and eraser) are inadequate tools.

The answer is that computers are adequate tools for dealing with outlines. Using a computer to manipulate outlines we have no concern about putting an idea down, and even exploring it a little. We don't care particularly about where we put something, if we want to move it later we can do so, and let the computer worry about erasing and cutting and scotch-taping.

it goes the other way, too

If you're already a user of computers, you'll find that outlines-on-computers are much more managable than textfiles-on-computers. The sample outline at the end of the brochure shows that outlines can be used to store and manipulate structured reference sources like The Guiness Book of World Records. Other reference sources which would blossom under High Files include

- dictionary
- thesaurus
- encyclopedia
- the yellow pages

If you program in Pascal or in Assembly Language, consider High Files for storing your source code. We are now working on some incredible ways to integrate outlines and compilers, the impact of outlines on programming languages could eventually effect the manner in which large and powerful programs are created.

the outline editor

An "outline" is the High Files equivalent of "textfile". An outline is a string of characters and a set of outlines. When the outline editor prints an outline, it only prints the string of characters not the set of outlines. All editor commands are applied to the set of outlines that belong to the "current_outline". There are commands which can be used to change the value of "current_outline". There is a command which can be used to "traverse" an entire outline, allowing commands that modify or consider all the information in an outline.

the implementation

There are three programs: the operating system, the outline editor and the table editor. All three programs use the swapping machine to access databases.

databases

The largest object which High Files deals with is a <database>. A database is a collection of tables and outlines; each database has a <database name>, which is the name of the Pascal file used to store the database.

the swapping machine
All three programs access databases thru the virtual machine named "the swapping machine". The swapping machine has two memory organs: fast memory and slow memory. Users of the swapping machine are aware of only one memory organ, a database. The swapping machine provides access to database information thru fast memory; a permanent copy of the database is maintained in slow memory.

the operating system
an extended version of the UCSD operating system. the new facilities implemented by the High Files operating system are described below.

process management
some programs running under the High Files OS can spawn new processes. since the Pascal machine is a single user real-time system, the parent process must wait for the child process to complete.

parameter passing
programs running under the High Files OS can recieve a single parameter of type <string>. a more powerful mechanism is available for programs which are willing to admit to being an <editor>.

database management
virtual programs such as "new_db" and "dir" are provided by the operating system.

complete access to UCSD Pascal
only the most superficial changes have been made to the UCSD system, allowing the High Files OS to view the UCSD system as another High Files program. the program "pascal" when called will turn control over to the UCSD system. therefore, we have added a command to the C(ommand level of the UCSD system: Q(uit. Q(uit returns the user to the High Files OS level. the UCSD commands are also available at the High Files OS level, under the names

```
filer = 'F(ile'
compiler = 'C(omp'
linker = 'L(ink'
flat_editor = 'E(dit'
and so on...
```

at present the Pascal compiler will only compile flat files, but we hope to offer (in the near future) a Pascal compiler which accepts its input in outline form.

the editors

the programs running under High Files are split into two classes: editors and other_programs.

editors

can spawn other processes (in response to a ! command)

other_programs

can't

this limitation could be removed if there were a lower-level mechanism for saving the state of a process. in the case of editors, only a small portion of memory represents the <state> of an editor (things like current_outline, current_table). it is therefore practical for the operating system to <save> the state of an editor. any other program with a small state could be given the power to spawn processes.

! command

the syntax is !<string>. when an editor gets a ! command it spawns the shell of the operating system with the parameter <string>. after executing the command, the OS returns to the editor.

the source code

High Files is implemented entirely in Pascal, it runs under and on top of the University of California - San Diego (UCSD) Pascal operating system. Since UCSD Pascal is a very portable system, the High Files filesystem is very portable. This implies that user databases are also portable, a fact which gives High Files the potential for becoming a universal means of transporting information.

The swapping machine is implemented as a UCSD Pascal "unit" named "dbaccess." The two editors were derived from a single body of Pascal code (referred to in the Formal Document as "the portable editor"), which implements the syntax of editor commands and certain "portable commands." The portable editor can be used as the basis for a third or fourth editor for the system; we plan to eventually sell the portable editor commercially (in source code only), allowing the High Files system to be extended by users of High Files.

We estimate that there is 15000 lines of Pascal code accounting for the High Files system - it is difficult to be exact since much of the code appears in two or more places, usually with only minor differences. About 1300 man-hours went into the development process including writing, testing, re-writing and restructuring, and documenting the system. The code has been used to form many different systems (it took a few trys to arrive at the current design). Therefore we believe that it could be used (in pieces) to solve a variety of problems which might come up in implementing a compiler, for example.

One project which we hope to attempt is to use a database to store heap-allocated variables ("pointer variables") in Pascal. This would only require minor modifications to the Pascal compiler; all that's needed is an interface from the generated code to dbaccess (dbaccess has routines called Block_Alloc, Block_Release, Block_Ref, Block_Assign). This would imply a trade-off of program speed for an extremely large and permanent heap. With this feature in the Pascal compiler, the source code for High Files would shrink tremendously and its clarity would improve manifold, since database objects would be referenced directly in expressions without having to use procedure calls followed by "moveleft"s. Future programs might not need to ever use "files" for permanent storage.

the implementor
David M. Winer

mailing address
29-31 170th Street
Flushing, New York 11358

credentials
Master of Science
University of Wisconsin - Madison, 1978
Computer Science
Bachelor of Science
Tulane University - New Orleans, 1976
Mathematics, Computer Science

oscar night

silent partners

Although this has been primarily a solo effort, High Files has clearly benefitted from work done by other organizations:

University of California - San Diego

provided a very reasonable machine to implement High Files on. There is absolutely no question that their innovation, "units" is responsible for the modularity of the source code.

Bell Labs

Alot of the design work behind High Files was done by the implementors of the Unix Operating System. Many of the features incorporated in the operating system and in the editors (including the syntax for editor commands) were patterned after their counterparts in unix.

co-conspirators

thanks to Charles Fischer, Pete Ferron, Gary Sevitsky, Joel Kamerman and Paul Pierce who contributed their opinions and encouragement at various stages of the project.

supporters

Allison Allen, my brother Peter Winer, my parents Eve Winer and Leon Winer, and my pal Pete Vanderveer, who patiently listened to my often incoherent ramblings on subjects which only a computer could find interesting.

Thanks to Rudy Kiesler for financing this project and for keeping me off the payroll and on the keyboard.

{guiness}
sample outline

The Guiness Book of World Records
by Norris and Ross McWhirter
1973 Edition published by Bantam Books

The Human Being
The Animal and Plant Kingdoms
The Natural World
The Universe and Space
The Scientific World
Physical Extremes
Loudest Noise

The loudest noise created in a laboratory is 210 decibels reported by NASA in the US in October, 1965. The noise came from a 48-foot steel and concrete horn. Holes can be bored in solid material by this means.

Quietest Place

The "dead room" measuring 35 feet by 28 feet, in the Bell Telephone System Laboratory at Murray Hill, New Jersey, is the most anechoic room in the world, eliminating 99.98 percent of reflected sound.

Largest Computer

The world's most powerful computer is the Control Data Corporation CDC 7600 first delivered in January 1969. It can perform 36 million operations in one second and has an access time of 27 nano-seconds. It has two internal memory cores of 655,360 and 5,242,880 characters (6 bits per character) supplemented by a Model 817 disc file of 800,000,000 characters. Commercial deliveries have been scheduled from 1972 at a cost of $9,000,000 - $15,000,000 depending on peripherals.

The Arts and Entertainment
The Business World
The World's Structures
The Mechanical World
The Human World
Human Achievements
Endurance and Endeavor
Ball Punching

Ron Reunalf (Australia) equaled his own world duration ball-punching record of 125 hours 20 minutes at 10:20 pm on December 31, 1955, at the Esplanade, Southport, Queensland, Australia.

Best Best Man

The world's champion "best man" is Wally Gant, a bachelor fishmonger from Wakefield, Yorkshire, England, who officiated for the 50th time since 1931 in December, 1964.

Commuter

Bruno Leuthardt commuted 370 miles each day for 10 years (1957-67) from Hamburg to teach in the Bodelschwingh School, Dortmund, West Germany. He was late only once, due to the 1962 Hamburg floods.

Ice Cream

7 lbs. 3 oz. (46 2.5 - oz. scoops) in 30 minutes by Peter Morrow, Brisbane, Australia, on April 27, 1970.

Honors, Decorations and Awards

Accidents and Disasters

Sports, Games & Pastimes

city	state	highest	lowest	avg precipitation
New Orleans	Louisiana	100	14	5.39000E1
Spokane	Washington	108	-25	1.71900E1
Atlantic City	New Jersey	106	-8	4.23600E1
Salt Lake City	Utah	107	-18	1.39000E1
Omaha	Nebraska	107	-17	2.75600E1
Albuquerque	New Mexico	110	-8	8.13000
Cleveland	Ohio	98	-19	3.53500E1
Huron	South Dakota	112	-39	1.73300E1
Oklahoma City	Oklahoma	108	1	3.68200E1
Portland	Maine	100	-39	4.28500E1
Little Rock	Arkansas	108	-4	4.86600E1
Juneau	Alaska	86	-22	5.46200E1
Minneapolis	Minnesota	99	-34	2.47800E1
Raleigh	North Carolina	98	0	4.35800E1