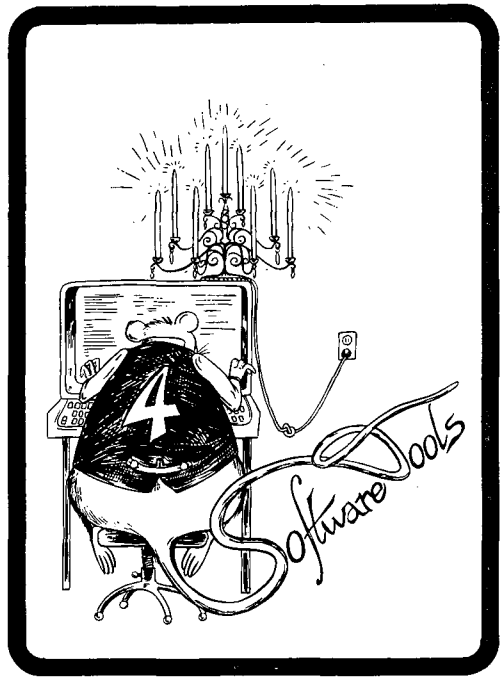
Software Tools COMMUNICATIONS

NUMBER 9 NOVEMBER 1982

STATUS OF THE USERS GROUP

We are (at last) rolling again under our own power. First, let me apologize to those of you who have been patiently waiting for some acknowledgement of your membership or tape request. The six month or so hiatus of operations was a result of a general lack of coordination among your newly elected board members; none of us seemed to have the time or the inclination to get the group moving forward again after LBL/CSAM's withdrawal. It became clear, as other users groups have discovered, that volunteer labor may get software written but it doesn't get clerical work done. Finally, I couldn't stand it any longer. In September I called Debbie Scherrer, who had been gratiously collecting all tape and membership requests, and volunteered to take over the running of the users group. She promptly forward a large box of unprocessed membership forms, tape requests and unanswered correspondence.

Since that time, I have hired part-time clerical support and have more or less gotten caught up with the processing of some 300+ membership forms. I am now working on getting the tape requests filled. Your patience is appreciated.



This work was supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy under contract number DE-AC03-76SF00098.

PUB-402 11-82/1200

NEW FORMS OF SUPPORT

LBL has recently gotten back into the picture; their Real-Time Systems Group (RTSG) has offered assistance and is currently handling the preprocessing of requests, publishing of the newsletter and collecting information for the next STUG software catalog. I sincerely doubt that our debt to LBL will ever be paid...

Many of your letters contain offers of help. To this end we are creating two new forms of membership in addition to the existing $15/yr. individual memberships. They are:

Industrial Membership	$ 150/yr.
Sustaining Membership	$1500/yr. (or more)

All Sustaining Member Organizations (SMOs) will be gratefully acknowledged in each issue of the newsletter and in other STUG publications, such as the upcoming STUG software catalog.

The $1500/yr. figure for SMO status may be met either with cash or with goods of equivalent value. We can use the following items:

- Virgin magnetic tapes, 1200 or 2400 ft. (1200 preferred).
- Bell 212A-compatible modems capable of auto-answer.
- Microcomputer systems with disk storage and RS-232 interface.

Please contact us if you have other items you would like us to consider. We are definitely interested in your help.

REGIONAL ACTIVITIES

Since the last meeting, a group of us on the West coast have gotten together informally to discuss standardization of our various implementations of the Software Tools. These meetings, while sometimes tedious, were quite worthwile and yielded useful results which hopefully will be published in a future issue of the newsletter. To encourage other such geographically-specific meetings we are publishing a list of interested implementors in this issue. The rest is up to you.

DISCLAIMER

JANUARY MEETING

Since everyone on our mailing list should have received announcements for the upcoming UNICOM conference, we won't repeat the details here. If you did not receive any such announcement and would like one, contact:

UNICOM
P.O. Box 385
Sunset Beach, CA 90742

This conference will be held in San Diego, January 25-28, 1983 at the Town and Country Hotel. As of this writing, we have received precious little in the way of abstracts. If you have done something with the Software Tools that is of general interest, PLEASE consider making a short presentation at the conference. Abstracts should be sent to:

Bill Appelbe
Department of EECS B-014
University of California, San Diego
La Jolla, CA 92093

sdcsvax!billr ucbvax!sdcsvax!bill

714-452-3729 or 714-452-3620

PURCHASE ORDERS

Due to the various ambiguous terms and conditions in the purchase orders we have received, and to the group's current unincorporated legal status, we are unable to accept purchase orders at this time. This policy may change as we learn more about the legal aspects of a users group such as ours, but for right now ALL purchase orders will be returned.

For those who have already sent STUG purchase orders, we are sorry to inconvenience you; especially since many of you have been waiting for your tapes or notice of your membership for several months. Although we will not be able to accept your purchase orders, we will proceed to process your requests and send you an invoice for the amount due; if payment was included with the purchase order, we will send a receipt of payment instead.

SPECIAL NOTE ABOUT MEMBERSHIPS

To be fair to those of you who have been waiting several months for some tangible result of your membership (and to keep from going crazy ourselves) ALL membership requests received before January 1, 1983 will be treated as 1983 memberships. This means that everyone gets their 1982 membership gratis.

The membership interval is one year, or four quarters. For membership requests coming in after January 1, 1983 and before April 1, 1982, their membership will begin in the second quarter of 1983 (2Q83).

In the future we will have an expiration date on the mailing labels. For those who have sent in their requests before 1983, your expiration date will be 4Q83, for example.

If you have not yet joined STUG officially, THIS WILL BE YOUR LAST ISSUE OF THE NEWSLETTER. A membership application form is provided in this issue.

NEW PROJECTS

We are very interested in establishing a network of machines running the software tools to provide facilities for electronic mail and software distribution. If you have a suitable machine and would like to participate in a networking project, please contact:

Dave Martin
Hughes Aircraft Co.
MS R1/C320
PO Box 92426
L.A., CA 90045
213-648-9927 (after 11:30AM)

****PLEASE NOTE****

There will be a few distribution tapes
available for purchase at the
Unicom Conference

****

Current STUG Members

Aerojet Electrosystems Co.
Ahlqvist, Gunnar
Akin, Allen
Alldredge, Gerald P.
Amey, G. X.
Anderson, Ken
Andose, Joseph D.
Armstrong, Thomas J.
Baker, Robert J.
Banks, Peter S.
Barnes, D. B.
Barnes, M. Wade
Barrowcliff,
Bate, D. G.
Baumann, David A.
Bearman, Steve
Beattie, Brian
Beckett, William F.
Beebe, Nelson H.F.
Belew, John F.
Berkbigler, Kathy
Bishop, Chris
Blanchard, Byron E.
Bogus, Tom
Bourke, Michael
Brown, David
Buhle, Jr., E. Loren
Bunnell, Joanne
Campbell, John-Jude
Cannon, Jr., George R.
Cannon, P. A.
Carey, Mal
Casey, Kevin B.
Cerofolini, Luigi
Chessin, Steve
Choma, Peter
Chow, Eugene
Christopher, John L.
Clarkson III, Thomas B.
Cohen, Martin
Colacino, Michael
Cole, J. H.
Cook, Gene
Coose, James P.
Cornish, Merrill
Countryman, Terry L.
Cowan, John W.
Croeker, Thomas H.
Croft, William J.
Crowfoot, Norm
CSIRO
Cupak, John J.
Daily, Thomas H.
Dearing, A.
DeGood, John S.
Delateur, Steve
DeMattia, Dennis
Deroo, Jerry J.
Dick, David R.
Dill, Dan
Dixon, David A.
Domenico, Ben
Donahoe, J. M.
Donovan, William J.
Downs, H. R.
Edwards, Gregg
Engelberg, Norman
Enslow, Philip
Eriksen, Bjorn
Eufinger, Jr., R. J.
Evans, Barry
Everhart, Glenn C.
Fariss, Gary B.
Favaro, John
Feenberg, Daniel
Ferguson, Edward E.
Fiddian, N. J.
Fishman, Barry
Fitzgerald, Desmond
Forschler, Kerry G.
Franklin, W. Randolph
Frankston, Robert M.
Frew, James
Furman, Roy E.
Gagner, Lee
Gallagher, Ph.D., Joe H.
Georz, Guenther
Gervais, Peter
Gordon, Robert L.
Gore, Willis
Gorlick, Michael
Greenberger, Martin
Groundwater, Neil P.
Guidi, John
Gunn, Keith
Guthery, Scott B.
Hague, S. J.
Hall, Clifford D.
Hall, Harold R. (Bob).
Hamrin, Carl J.
Hanke, F.
Hanshew, Jon
Harris, Gord
Harris, Michael E.
Harris, Robert A.
Hathaway, Steven J.
Head, Stephen
Hecht, Herbert
Heidelberg, Terry
Heinfeld, Blaine
Hengl, Terry
Hethcoat III, Charles L.
Holeman, Chris
Holeman II, C. W.
Holt, Charles C.
Hornbach, Kathy
Howard, James H.
Hubes, Corey F.
Huckster, Kent
Huebner, Doug
Huggins, W. H.
Hurler, Joe
Hutchison, Jr., William G.
Irvine, Hugh
Janes, Robert H.
Janssen, William C.
Jarvinen, Galen
Jenner, David C.
Johnson, Frederick C.
Joint System Development Corp.
Judd, Robert L.
Kanada, Yasumasa
Karpimski, Richard H.
Kasper, Paul R.
Kent, Christopher A.
King, Joseph P.
Klein, Stanley A.
Kochendarfer, Richard A.
Konishi, Yasuo
Kramer, R. O.
Krawiec, Ted
Krieger, Morris
Lawson, Lance P.
Layman, Terry J.
Lazarus, Steve
Leban, Bruce
Levy, Roger
Lewis, Robert R.
Ling, Raymond
Liu, Lon-Mu

Current STUG Members

Livingston, Jr., James W.
Lockheed-California Co.
MacLennan, Mark
Maguire Jr., Gerald Q.
Maguire, R. Brien
Mandelberg, K.
Mangold, Robert B.
Marcum, Robert E.
Martin, Arvid
Martin, Carol C.
Martin, Dave
MCA, Inc.
McGilton, Henry
McKinney, Norris P.
Meyers, Stephen E.
Mikes, Peter
Miles, Larry
Miller, A. L.
Mills, Mark
Morgenthaler, T. W.
Moroney, R. M.
Murchland, John D.
Newberry, Steve
Newmarch, J. E.
Novick, Andrew
Oak Management Corp.
O'Donnell, Susan
Ohashi, Yoshikazu
Okanagan College
Olson, Mark L.
Oyanagi, Yoshio
Parington, David J.
Pearson, Doug
Peck, Jeff
Peterson, Mark H.
Phillips, David M.
Phinney, Mark C.
Poanessa, Lou
Poulton, Ken
Rao, D.C. Vinayak
Rasband, Wayne
Rataj, Walter J.
Ream, Edward K.
Rew, Russ
Rice, D. Lloyd
Richards, Monty
Rijksunivrsiteit
Roberts, Joe C.
Robinson, Gordon D.
Rosenthal, Eric S.
Saisho, Toshiaki
Salazar, Sandra
Sanders, Rex
Sandmayr, Helmut
Saroyan, Allyn
Satterfield, Steven G.
Saxer, Gary
Scherrer, Deborah K.
Schlenger, Alan
Schneider, C. L.
Scott, Anne E.
Shah, Babu V.
Shapin, Ted
Shapiro, Michael D.
Shoemaker, Nancy E.
Sidner, Steve
Silbert, M.D., Jerome A.
Simpson, Anthony U.
Skeen, Jim
Smith, H. Warren
Smith, Walter J.
Spencer, Henry
Stevens, Jack
Stevens, W. Richard
Stevens, W. Richard
Stone, Jeffrey D.
Stordahl, Ron
Story, Peter J.
Strong, Tom
Sturges, Bruce
Sullivan, M.
Sutherland, Dennis
Tabachnick, Murray
Tahim, K. S.
Tanenbaum, Andrew
Taylor, Jeff
Teitel, Robert F.
Thomas, Jr., Richard F.
Thompson, David
Tilson, Michael D.
Tome, Ellen
Tran, T. Lan
Trappe, Scott R.
Triplett, James A. J.
Tsitsivas, H.
Unicorn Systems
University of Nevada, Las Vegas
Upshaw, Bob
Ushijima, Kazuo
van der Hyden, Pieter
van der Hyden, Pieter
Van Tuyl, Robert R.
Velevitch, Christopher G.
Verne, Howard E.
Verschell, Howard J.
Vitek, P. J.
Waite, J. Monte.
Wajih, A. Reza M.
Wand, I. C.
Wan, Peter N.
Webb, Kirk
Weinberger, F. R.
Weiss, S. or Rittberg, I.
White, Andrew N.
White, Ronald G.
Whiting, W. B.
Wiesel, Joachim
Wigan, M. R.
Willard, R. E.
Williams, W. Edwin
Winston, Ira
Witty, R. W.
Wulff, Robert S. M.
Yip, Rodney

MACHINES

The following is a list of developers who have implemented or are implementing the Software Tools on various machines. Many sites are working concurrently; this list represents those with whom we have had the most contact. The names marked with a '*' have been verified as current. If you know of additions or changes to this list, please contact

Tonia Cantrell
Real Time Systems Group
Mail Stop 46A
Lawrence Berkeley Laboratory
Berkeley, CA 94720

Phone: (415) 486-5873
Arpanet: tonia@lbl-unix

Machine	*Contact (Affiliation)*	*Phone*
Apollo	Jim Ward (Apollo)	—
Burroughs B1700	C.R.Snow (Newcastle upon Tyne)	—
CDC Cyber w/NOS	Mike Shapiro (NCR)	—
CDC Cyber,6000s	C.R. Snow (Xerox)	415-494-4518
	David Hanson (University of Arizona)	602-626-3617
CDC CTSS	Dottie Schmeling (LASL)	—
CRAY-1	*Kent Crispin (LLNL)	415-422-4273
CRAY-1 COS	J. Otto Tennant (Cray Research)	—
DEC IAS	Jessie Howell	703-525-6020
		703-922-7230
DEC RSX-11M	*Joe Sventek (LBL)	415-486-5205
DEC RT-11	*Chris W. Holeman	619-444-0446
DEC TOPS-20	*Chris Peterson (M/A-C Linkabit)	916-453-7007 x454
	*Steve Hathaway (Tektonix)	503-685-3292
	Webb Miller (UCSB)	805-961-4067
DEC VAX/VMS	*Dave Martin (Hughes Aircraft)	213-648-9927
	*Bob Upshaw (LBL)	415-486-6411
	*Joe Sventek (LBL)	415-486-5205
	*Jerry Deroo (Garmaise & Assoc.)	416-978-5396
DG AOS,AOS/VS	Peter Reintjes (Data General)	919-549-8421
DG RDOS	*John Hanshew (CompuCode)	415-339-9463
	Bob Lewis	408-249-5986
HP-1000 w/RTE	*John Campbell	213-831-3938
	Larry Dwyer (Hewlett-Packard)	408-257-7000
HP-3000	*Ken Poulton (Terminal Software)	415-857-8461
Honeywell w/MPE III/IV	Jerry DeRoo (Garmaise & Assoc.)	416-978-5396
IBM OS/MVS	*Bill Meine (Louisiana Land)	303-988-8660
		303-989-5442
IBM VM/CMS	*Ben Dominico (NCAR)	303-494-5151 x559

ISIC	*Robert Calland (U.S.Navy)	714-225-2413
MODCOMP w/MAX-IV	*Bob Upshaw (LBL)	415-486-6411
Multics	Jerry DeRoo (Garmaise & Assoc.)	416-978-5396
NCR V8000 w/VRX	Mike Shapiro (NCR)	—
OS/32	*Michael Brouke (Wollongong Group)	415-692-9224
PDP 11/34	*Frank Bradford (Ftn for Md Care)	714-825-6053
PDP-11	*Walter Brown (Moravian College)	251-861-1300
Perkin-Elmer	*Michael Bourke (Wollongong Group)	415-692-9224
SEL MPX1.4/2.0	*Walt Donovan (NASA/AMES)	415-965-6368
Tandem	Jessie Howell	703-525-6020
		703-922-7230
	Bob Lewis	408-249-5986
UNIX	*Debbie Scherrer (Unicorn Systems)	415-881-4490
	*Dennis Hall (LBL)	415-486-6053
	*Michael Bourke (Wollongong Group)	415-692-9224
	*Neil Groundwater (Analytic Disciplines)	703-893-6140
Univac	*Ben Cranston (Univ. of Ma.)	301-454-2946
WANG VS-80	Side Shapiro (Wang Institute)	617-649-9731
XEROX Sigma 6	Norman Crowfoot (Northern Ariz. Univ.)	—
Z80/8080 CP/M	Phillip Scherrer (Unicorn Systems)	415-881-4490

Software Tools Users Group Meeting

Friday, July 9, 1982

Editor's Note

Tom Strong

The following articles were written to summarize the main points of the presentations made at the Boston meeting for people who did not attend the meeting. Copies of the abstracts, viewgraphs foils, slides, and/or papers were used when available. Suggestions and corrections should be sent to me at the USENIX Association Office or electronically at

ucbvax!g:usenix

Session Chair: David Stoffel, Users Group Coordinator

Software Tools Bulletin Board

Bill Louden
CompuServe, Columbus, OH

[STUG Board perspectives supplied by Dave Stoffel.]

CompuServe offers computing facilities to the public. They provide DEC 10s and 20s running their own operating system with access through several publically-accessible networks. They are offering to set up a clearing house for the exchange of information about Software Tools. Services offered include:

- an electronic mail system for public and private mail,
- a bulletin board,
- megabytes of on-line disk storage for such things as database archival storage,
- central computing resources for Tools development by working groups, and
- central Tools tape distribution.

CIS, CompuServe's own network, is currently available in 110 cities in the USA. CompuServe's machines may also be accessed through Telenet and Tymnet and a public network in Canada [Datapak?]. Network access is expected to become much larger by the end of 1982 due to CompuServe's impending merger with H&R Block. Connections are available at 110→1200 baud. The only charge is for connect time and it is negotiable; their basic charges are $22.50/hr from 8am→5pm and $5/hr from 6pm→5am (local time). There is a $17.50/hr surcharge for connections faster than 300 baud [at all times?...Ed].

CompuServe's disk storage could be used as a source for updates to the basic distribution tape and/or for machine-dependent implementations of the Tools. Group members could login and download what they needed over the phone. Access to the magnetic tape on CompuServe's system would be limited to the group administrator.

The STUG Board is concerned with problems of inter-group communications, central Tools development, and Tools tape distribution. The services offered by CompuServe provide one potential solution to these problems. The Board plans to contact CompuServe after a firm membership base has been established. Any arrangements made with CompuServe will, of course, reflect the desires of the Group members.

The Software Tools On The Data General NOVA

Jon Hanshew
CompuCode, 6147 Aspinwall Road, Oakland, CA 94611

CompuCode has ported most of the Tools to a Data General Nova® 4/C running RDOS®. Mr. Hanshew gave a talk on the project at the Winter, 1982, Software Tools Conference. In this presentation he discussed bringing up *ratfor* and *sh* and general problems he encountered bringing up the Tools.

One major problem was that the system allowed only about 40Kb of memory for user code. He found that the "data set" routines used in *roff*, *sh*, and *macro* keep both the hash table and the definitions in two different locations in the same large array. He tried putting the array on disk but found he got two disk accesses for each defined symbol. He then coded in *ratfor* the scheme employed by the bootstrap *ratfor*: he put the hash table in one array in RAM and the symbol definitions in another array on disk. This saved space and reduced disk accesses to less than one per definition. Still in need of space, he coded five library routines in assembler and commented out code for *ratfor* features that could be handled by DG Fortran IV (e.g., long variable names). In the interests of speed he recoded *putlin* and *getlin* in assembler and caused *ratfor* to read and write a binary *ratdef* file.

When implementing *sh* he found that some conditionals get null arguments, which cause problems to the DG compiler. He implemented a way to swap programs and to pass messages between swapped programs.

He found that several tools have programs with the same name but different functions. Consequently, each tool had to be developed in its own directory. He found it he could save space by creating a directory with a library file of the compiled subroutines and linking it to the separate tool-development directories.

Mr. Hanshew went on to discuss the costs of implementing the Tools and the advantages of purchasing them from a vendor. He pointed out that there is a large amount of code and that development of the tools on a new system takes many hours per program. A reasonable vendor would have optimized the Tools for the system and would provide support, updates, and machine-dependent tools like command files to build the tools. CompuCode is selling their implementation of the Tools for the Data General Nova 4/C running RDOS. It is portable to the DG Eclipse.

The speaker has submitted a paper for the Conference Proceedings.

Software Tools for TOPS-20

Steve Hathaway
Tektronix, Inc., Delivery Station 63-333, PO Box 500, Beaverton, OR 97077

Tektronix® found it needed a better text and software control system. They investigated several and decided to implement Software Tools and Neil Groundwater's text control system (TCS) utilities on their TOPS-20® operating system. This talk described their goals, extensions, and problems.

The hardest utility to install was the *sh* command processor. Their implementation does not allow background processing but it handles all other functions of the portable shell. The portable version supports only *ratfor*—compiled programs; they modified it to address all the TOPS-20 features so they could load and run non—*ratfor* programs. Their shell also allows command files to be nested to arbitrary depth with redirected I/O acquired from the parent process.

Their implementation of the Tools allows tools to be invoked by the TOPS-20 utilities *batch* and *pcl* by using the *run* command. Because of differences in design the method of passing arguments to a tool is different for *sh* and *run*.

The biggest problems they had in implementing the Tools were the TOPS-20 limits of six characters in a variable name and 40 characters in a file name, and several tools lack of checking for array boundaries when handling large files.

Tektronix has implemented a project management scheme. It uses a software version management database on TOPS-20. An arbitrary number of concurrent product, test, and custom modification versions can be handled without conflict using a few routines and the standard Tools. The scheme

defines conventions for directories, file types, and file naming. Their database contains shell command files that call upon unmodified TCS utilities. Their version management philosophy can be implemented without TCS in a UNIX environment that contains *sccs* utilities.

The speaker has submitted a paper for the Conference Proceedings.

Portability in the Virtual Operating System

Bob Upshaw (bobup@lbl-unix)
Real Time System Group, Bldg. 46A, Lawrence Berkeley Laboratory, 1 Cyclotron Road, Berkeley, CA 94720

This talk focused on three meanings of the word "portability" provided by the Tools environment: (1) "people portability", the ability for people to move to other machines with minimum effort; (2) "program portability", the ability to move programs to other machines without modification; and (3) "functional portability", the ability to move programs to other machines because of the use of machine-independent user interfaces for the primitives. Mr. Upshaw went on to discuss the latter two meanings and common violations of portability in those contexts.

Portable programs must, by definition, be written to conform to the standard of the language. This is difficult, as few programmers are well versed in the standard of the language(s) they write in. Portable programs can only operate in a "portable environment" in which all low-level operations are defined in a machine-independent manner. This environment must remain constant across different machines, making it difficult to change or upgrade the environment. Portable programs are usually desirable, but not always cost effective. When portable programs and or people are desired the Software Tools environment can be very cost effective, as it normally takes only a few person-months to provide a portable environment with many person-years of tools.

Portable programs require well-defined portable user interfaces to the primitives (e.g., *open*, *close*) and well-defined actions for those primitives. The Tools implementor must know the machine well enough to obtain the desired action from the system without deviating from the specification of the primitive. Since there is no facility for changing the Tools standard, improvements to the standard environment are difficult to implement. Significant amounts of time and agreement among some number of the standards' users are required for changes to specifications. However, the costs involved in maintaining and improving the Tools environment pale when compared to the cost of building the facilities of the Tools from scratch.

Mr. Upshaw went on to list a number of common non-portable practices that have been observed in *ratfor* programs. He discussed these practices with a view to possibly setting firmer standards for the Tools environment. [The practices are discussed in more detail in the Proceedings...Ed.]

Major Violations

(1) The use of quoted and hollerith strings; the *ratfor* character data type is preferred and is the **only** portable data type useful for string manipulation available in all standard *ratfor* implementations.

(2) The use of non-standard library routines or primitives.

(3) The use of Fortran 77-specific constructs.

(4) The use of non-standard Fortran conventions or constructs (e.g., the *implicit* statement or names that are longer than six characters).

(5) Taking advantage of machine-dependent concepts such as the order of byte packing in an integer variable.

(6) Mis-typing the results of functions or parameters to routines. In particular, the use of a character variable when an integer is expected.

Minor Violations

(1) Neglecting to handle a lone '?' as an argument to a tool.

(2) Non-standard Fortran, such as assuming DO loops always execute once or violating statement-ordering standards.

(3) The use, with Fortran 66, of hollerith strings anywhere but in DATA statements, as parameters to routines, and in FORMAT statements.

(4) The assumption that variables in subroutines are static.

(5) The assumption that real and integer variables occupy the same amount of space, in spite of what the Fortran 66 standard says.

(6) Attempting to simulate *structures* by using *equivalence* statements to store data of different types.

Violation of portability standards is not always "bad"; it is a matter of being aware of violations and weighing the benefits. Life is much easier for all concerned if violations are minimized, localized, and documented.

The speaker has submitted a paper for the Conference Proceedings.

Proposed Extensions to the Primitives and Library — Panel Discussion

Bob Upshaw, Debbie Scherrer, Dennis Hall, Joe Sventek
Lawrence Berkeley Laboratory,
Dave Martin
Hughes Aircraft,
Michael Bourke
Perkin-Elmer,
Mac Chandler
Helmsman Systems,
Phil Scherrer
Unicorn Systems

The group listed above have been trying to put together a set of proposals for extensions to the primitives and library. They perceive four major problem areas.

Data Types — start moving towards allowing binary
add the following types: *byte* — smallest addressable unit, *real*, *double_int* (not totally portable), *double_real* (not totally portable), and *quoted string* (ASCII array)

Environment
ways to set and get environmental variables

I/O reading and changing directories, getting information about files, more routines for reading and writing files

Terminal Handling
need ways to set UNIX-type *cooked*, *raw*, and *rare* modes; need way to read binary data

Dynamic Storage
should allow for true dynamic storage and handle multiple data types.

The group's suggested additions to the basic library are listed at the end of these meeting notes.

Comments and suggestions are required; who else can represent your needs to the Software Tools community?

Spelling Checkers, Compound Words, and Variant Spellings

Eric S. Rosenthal
IMI Systems, 1500 Broadway, New York, NY 10036

Existing spelling checkers do not adequately handle either compound words or words with variant spellings. These two classes of words are difficult to handle because authorities disagree on their preferred spellings and because the same word can have different spellings or forms depending on its meaning or grammatical function. For example,

indices	indexes
look up	look-up
well known	well-known

This talk discussed methods of checking the spelling of these two classes of words without more sophisticated processing of natural language that is performed by existing spelling checkers.

Only one variant spelling of a word should be used in a document, unless different spellings are required for different meanings. For example, either "ax" or "axe" may be used, but not both, so a checking program should flag all occurrences of both spellings if (and only if) both occur. [This can get complicated because the checking program must also look for conformity in related words such as "axman" and "battle-ax"...Ed.] Another example: both "indices" (for superscripts and subscripts) and "indexes" (for parts of books) may appear in a document, but the checking program cannot know (without knowing the context) which is correct so should flag any occurrence of either.

Compound words may be "open", hyphenated, or "solid", depending on whether they are written as separate words, with a hyphen, or as one word. Distinguishing open compounds from hyphenated compounds of the same words and natural occurrences of the components as consecutive but otherwise unrelated words requires more sophisticated natural language processing than was being presented. For example,

The cross-examined witness is typically a cross examined witness.

The treatment of these classes of words can be expressed by rules specifying that certain Boolean combinations of occurrences of words or sequences of consecutive words should trigger warning or error messages. Warnings are required for those cases where the checker cannot determine correctness from the rules alone. A rule must be present for each combination that the author wishes to check, and must reflect the author's preferences. Three approaches to identifying compounds and variant spellings are presented in Mr. Rosenthal's paper.

The speaker has submitted a paper for the Conference Proceedings.

QDP - A Quick Plotting Tool

Kenneth R. Anderson
M.I.T., Applied Seismology Group, Lincoln Laboratory, 42 Carleton Street, Cambridge, MA 02142

QDP is a ratfor program that is used to produce plots of scientific data files. It was designed to be easy to use, to interface easily to other tools, and yet to be flexible enough to produce a variety of plotting styles. Although flexibility was reduced in favor of ease of use, QDP can still be used to produce complex plots of publication quality.

All specification to QDP are made at the command level through a variety of flags. Defaults have been selected to produce reasonable exploratory plots easily. The data may be either columnar or continuous and the field(s) on each line containing the value(s) may be specified. QDP will produce scatter plots, bar plots, or continuous lines. Several plotting symbols and dotted line types are available. Scaling can be either linear or logarithmic. Tick marks, axes and labels are supported.

QDP is implemented in *ratfor*. Only six device-knowledgeable primitives are required. Versions of QDP exist for Tektronix 4014-class terminals and the Megatek 7000.

The speaker has submitted a paper for the Conference Proceedings.

NBS Projects On Software Technology and Computer Based Office Systems

Mike Chernick
National Bureau of Standards, Building 225, Room B-226, Washington, DC 20234

[I missed this talk and there was no paper submitted; the abstract is reproduced below...Ed.]

The purpose of the Software Technology Project is to develop software engineering guidelines and standards for the Federal Government in the area of advanced program development techniques. The development of guidelines and standards is re-enforced by actual hands-on experimentation with software tools, programmer workstations, and user environments. Products developed within the project include a taxonomy of software tool features, a software tools database, and a guide for introducing tools into medium and small programming environments.

Computer Based Office Systems (CBOS) represents a major application area for computers and networks. The CBOS program at NBS includes development of Federal Information Processing Standards (FIPS) for such applications as Computer Based Message Systems (CBMS). The first standard is a message format for CBMS and the second is a message transfer protocol. The CBOS program also has a laboratory for testing proposed FIPS protocols and for implementing developmental applications.

Navy Software Development With *ratfor -T* and Software Tools

Neil P. Groundwater
Analytic Disciplines, Inc., 8320 Old Courthouse Road, #300, Vienna, VA 22180,

Ratfor -T is one of a set of development and testing tools developed by ADI for the Navy. These tools were developed to reduce program development costs and assist the user in program verification. The major tools developed are:

ratfor -T which discerns enough about a program's structure to provide handles for determining how well test data tests the possible paths through a given set of program modules;

pretty which standardizes *ratfor* style and will prompt the user for comments to insert before *ratfor* keywords;

prodoc which knows about the program structure and the prompts the user for standard subprogram header documentation to supplement what it has determined;

tree which prints a plot that traces all possible subprogram calls from the top down;

see which displays possible program paths within each subprogram; and

sum which prints a summary of the possible paths in a program that were executed in a run.

Rocky Mountain Area Implementors Group

Ben Domenico
NCAR, PO Box 3000, Boulder, CO 80307

The Rockies Association for Tools (**RAFT**) is a group of people in the Rocky Mountain area are meeting on a regular basis to solve common problems with the Tools. The members have implemented the Tools on a variety of systems. The goals of the group are:

- Publish a list of known problems with possible solutions.
- Collect the fixes into a "local distribution" and get it running on all the member's machines.
- Discuss enhancements, implement those that can be agreed on, and include them in the local distribution.
- Coordinate their work with the Tools Group and other local groups.

RAFT has written a paper that contains many problems and solutions, a Tools wishlist, and the names and address of the group members. A copy of the paper that has been updated from the one available at the Boston meeting will be available in the Conference Proceedings. The paper is required reading for anyone implementing or having problems with the Tools.

Future Directions - Discussion

A general discussion followed the talks. Listed below are some incomplete notes on the topics covered.

STUG Network

CompuServe's presentation is being discussed by members of the Board.

Editors

There is no known portable full screen editor; it appears each user is stuck with using the native one on the system their Tools are on.

roff

Macros would be very handy to have; Bob Upshaw's *roff* (at LBL) has a "crudely implemented" form of macros.

ratfor Improvements

Many improvements are wanted; suggestions were discussed in the discussion of Proposed Extensions and in the talk on what RAFT is doing.

A better *ratfor* could be built "quite easily", given a parser generator.

YACC and LEX

Tools versions of these are being worked on.

Suggested Additions to the Software Tools Basic Library

* indicates a primitive
# indicates a portable routine which might benefit from system dependencies

I/O

* getbyt	get a byte/smallest-unit from file
* putbyt	write a byte/unit to file
# printf/prints	formatted write
# scanf/scans	formatted read
ngetch	get a (possibly pushed back) character
pbinit	initialize for push-back I/O
pbstr	push back a string
putbak	push back a character
getblk	get block of lines from file
putblk	write block of lines to file
setsep	indicate block separators
logpmt	prompt (with history)
fskip	read past 'n' characters/bytes on file
acopy	copy from one file to another (text or binary)

FILE MANIPULATION

* stat/fstat	pick up information about file (type, size, dates, etc.) (this would replace/work in conjuntion with fsize, gettyp, etc.)
# access	see if file exists and is available for access

INFORMATION RETRIEVAL

* getdir	pick up specific directory names
# getuid	get user name
locfil	search for full pathname of file

DIRECTORY MANIPULATION

* opendr	open directory for reading
* closdr	close directory
* gwdir	get name of current directory
* cwdir	change to different directory
* gdrprm	read next file name from directory

TERMINAL HANDLING

* setmod	set terminal mode (RAW, RARE, COOKED)
curses package	(3 or 4 are available)

ARGUMENT HANDLING

(none have been submitted, but several interfaces are under consideration)

DATA TYPE MANIPULATION

rtoc	convert real to character
ctor	convert character to real
putrl	output real number
dtoc	convert double real to character
ctod	convert character to double real
putdbl	output double real
ditoc	convert double integer to character
ctodi	convert character to double integer
putdi	output double integer

(there are also some macros which support double integer arithmetic)

* htos	convert from Hollerith to string

stncmp	compare 'n' characters of two strings
isin	same as *index* only knows about escapes and quotes
stins	insert substring into string
stdel	delete substring from string

ENVIRONMENT HANDLING

getenv	get item from environment
setenv	store item in environment

DATE MANIPULATION

atodat	convert from ASCII to integer date

(some integer date handling (pseudo Julian date) packages are being evaluated)

DYNAMIC STORAGE

dsmove	move a block to one of a different size

(A rewrite of the entire package is also being considered)

LINKED LIST HANDLING (or QUEUE HANDLING?)

maklst	create a linked list
frelst	remove a linked list
push	push item onto list
pop	pop item off list
xtract	read item from list (but leave there)
inject	insert an into at a specified place in list
rmnode	remove a node from linked list
nxtnod	determine next node in linked list
prvnod	determine previous node in linked list

RANDOM NUMBER GENERATION

(a complete package has been submitted)

SORTING

shell	shell sort (integer)
bubble	bubble sort (integer)
quick	quick sort

MISCELLANEOUS

* exec	execute a task (for those who cannot do *spawn*)

SUGGESTED CHANGES/EXTENSIONS TO EXISTING LIBRARY

prompt	should return a status (stat = *prompt* (...))
endst	should return a status (call *endst* (stat))
rat4	should be able to handle quoted strings in some reasonable way
dynamic storage	should handle multiple data types and allow for true dynamic storage
symbol table routines	should be able to handle at least the *character* data type, if not others
fmtdat	should also be able to return a sortable date string
readf/writef	should use *byte* instead of integer/character
pattern-matching package	should be rewritten

New Tools for the Virtual Operating System

Bob Upshaw
Van Jacobson

Real Time Systems Group
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

The Real Time Systems Group at Lawrence Berkeley Laboratory has spent over four years developing software tools for use on our various machines. There are many good reasons for putting the Software Tools on a computer system, but our main one is the desire to increase productivity in the areas of program development, data handling, and data analysis. We chose the Software Tools because they were UNIX-like and UNIX [4] is well known for improving productivity. (For just a few of many references on this topic, see [5].)

Unfortunately, not every machine runs UNIX (yet), but every machine can (and many do) run the Software Tools. So, in striving to build ourselves better tools, we often look to UNIX for guidelines and to the Virtual Operating System for a vehicle. This paper describes a few of the tools we have developed or are in he process of developing. Naturally, these tools are designed to run in the Virtual Operating System environment and therefore are (should be) portable. Any similarities between these tools and UNIX tools are purely intentional; we do not intend to re-invent the wheel.

1. YACC

In the May, 1981 issue of *IEEE Transactions on Software Engineering* there is a paper describing a *portable* LR(1) parser generator[1]. The parser generator (named 'LR') was developed at Lawrence Livermore Labs and is in the public domain. Naturally, we got our hands on it and performed some experiments. It is portable and does indeed work, but suffers from the shortcomings of most parser generators, including

- The inability to assign token values to terminal symbols, making it difficult to write a scanner which would work regardless of the changes to the grammar.
- The inability to attach an action (semantics) to a production, making it necessary to modify the code generation section of a compiler whenever the slightest changes were made to the grammar.
- The requirement to order recursive productions in an awkward way to force the desired associativity.
- The requirement to write the grammar productions in an awkward way in order to specify precidence or to avoid ambiguities in the parser (such as shift-reduce conflicts.)

Yet, despite LR's shortcomings it represented an impressive program. Since the above problems are solved by Yacc[2], we decided to make LR work like Yacc. Hoping to avoid

code modifications to LR as much as possible, we designed a set of tools which did the following:

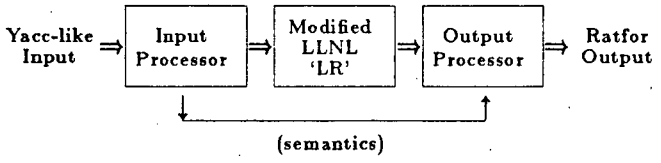
- Took a Yacc-like input grammar and split it into two pieces:

  1) A BNF grammar to be processed by LR.

  2) A file containing the semantics associated with each grammar production.

- Took the output of LR (the parse table) and the semantics generated by the first tool and put them together to form a ratfor program — the desired compiler.

Here is a picture of how our Yacc tool works:



We have since used our Yacc tool to build a significantly inhanced dc tool, and we are using it to develop a new shell.

2. LEX

Using the dragon book[3] as a reference, we have implemented a Lex tool similar to the UNIX Lex tool (except in ratfor.) For those who may not be familiar with Lex, it is to scanners what Yacc is to parsers. The input to Lex specifys regular expressions which describe the tokens recognized by a lexical analyzer (i.e. a scanner) and the actions to be preformed when those tokens are recognized in the input stream. The output of Lex is the scanner written in ratfor.

There is still more work to be done on our Lex, mostly in the area of performance optimization, but we have already used it to generate a 'typeof' tool (which looks at a source file and determines it's language), scanners for some simple languages, and a scanner for our new shell (below).

As with Yacc, our Lex input is similar to the UNIX Lex input except we followed the tools standard regular expression syntax (sigh). The conversion of a UNIX-style Lex input specification to a tools-style Lex input specification is straight forward, thus allowing the use of existing Lex input files which are in the public domain.

3. SHELL

We are currently working on a new shell specification which is to be upwards compatible with the current VOS shell. Our current spec contains control structures, variables, procedures (a mechanism to replace 'alias' as suggested by Bill Joy in the csh manual), a 'history' mechanism, wildcards, command substitution, and so forth. Most of what we have done has very much the flavor of the C-shell. Our 'sh' spec has been frozen and we have made a first pass on the grammar and semantics.

4. PRINTF

Yes, Virginia, there is a C-like 'printf', 'fprintf', and 'prints' written in ratfor. Our routines are *almost* portable and will probably work on almost all the machines running the VOS. The only rule we violate is passing data of one type and declaring it as another. However, this should work almost everywhere since we are very careful not to *use* a data item except in routines where it is declared correctly.

5. PEEK

We are writing a portable tool similar to UNIX 'more' which we are calling 'peek'. ('More' is a UNIX tool which works like 'crt' but uses raw-io and allows additional commands following each prompt.) Peek is not named 'more' because the commands to peek are slightly different than those of more. Peek will allow the user to jump forwards or backwards in a file or a group of files, search forwards or backwards for a regular expression, etc.

6. Conclusion

The purpose of this note was to make the members of STUG aware of what we are doing in the hopes that they will

- not duplicate our efforts needlessly,
- tell us what they are doing so we will not duplicate their efforts,
- and give us suggestions or feedback when they believe we may be on the wrong track.

Any correspondence is welcome and can be addressed to the authors at:

Real Time Systems Group, Build. 46A
Lawrence Berkeley Labs
1 Cyclotron Rd.
Berkeley, Ca. 94720

By the way, this paper was produced completely by the use of the Software Tools. Our newest project is a tool which filters 'roff' input and produces 'TEX' [7] input, which is then run thru a TEX formatter. Thus, our roff documents can be output to a phototypsetter-quality printer, as was done here.

7. References

[1] "LR — Automatic Parser Generator and LR(1) Parser", Charles Wetherell and Alfred Shannon. *IEEE Transactions on Software Engineering,* May, 1981.

[2] "YACC — Yet Another Compiler-Compiler". Stephen C. Johnson, Bell Labs.

[3] *Principles of Compiler Design,* Aho and Ullman, Addison-Wesley.

[4] UNIX is, as we all know by now, a trademark of Bell Labs.

[5] Even the vendors who used to deny the existance of UNIX now admit it's strengths:

DEC: "VNX Plan Bridges VAX/VMS, UNIX Environments", *Insight,* Vol. 2, Number 8, October, 1982.

AMDAHL: "UNIX On Mainframes: Amdahl", *commUNIXations,* August, 1982. Page 13.

IBM: "How Data Flow Can Improve Application Development Productivity", *IBM Systems Journal,* Vol. 21, No. 2, 1982. Page 167-178.

[6] "A Parser Generator for a 'Production' Programming Environment", Theresa Breckon, 1982. (To be published.)

[7] TEX *and METAFONT — New Directions in Typesetting,* Donald E. Knuth, American Mathematical Society and Digital Press.

Tools in C (or Pascal?)

Deborah Scherrer

[Reprinted, with permission, from the Unicorn Systems Newsletter]

There has been much discussion in the Tools community, especially among micro enthusiasts, about the need for a version of the VOS package in C or Pascal. When the project was initially designed, a prime requirement of the language chosen was that it be available on virtually all operating systems from micros to large mainframes. At that time, FORTRAN was the only language meeting this criteria. To make up for FORTRAN's lack of elegance and high-level constructs, the ratfor preprocessor was developed to provide the control structures and readability features of a high-level language while still retaining the portability of FORTRAN. The C-like support library was designed to supplant FORTRAN's incomplete textual, file manipulation, and I/O capabilities.

The choice of language is not critical to the virtual operating system approach. Now, with the computer industry rapidly developing new machines and more elegant language, perhaps it is time to reconsider the original choice. FORTRAN, even with the ratfor preprocessor, simply does not provide the functionality or elegance of other higher level languages like C or Pascal. It was designed more than 25 years ago for large-scale mathematical computations, and not the text processing and file manipulations now more common to computing. FORTRAN has some severe deficiencies, lack of recursion and data structures, for example, which make coding of complex text manipulation utilities difficult. Furthermore, the micro industry, providing us with the machines of the future, has rightly judged the inappropriateness of FORTRAN to most micro applications and is not providing FORTRAN compilers on many of the newer machines. It would appear that remaining tied to FORTRAN may limit the future usefulness of the Virtual Operating System.

We see two possible approaches to moving the tools to a new language, each with some disadvantages. One would be to define (and possibly extend) ratfor as a language itself. The preprocessor would then become a compiler which might have several code-generating backends, FORTRAN, C, Pascal, Ada, or even assembly code, in the Vrije University (Andy Tanenbaum) style. The advantage with this approach is that no rewriting of existing utilities would be required and there would be only one language to write in and thus one version of the tools to maintain, while still making the package available in whatever language is appropriate to any particular machine. The second approach is to translate the existing

tools to a new language and subsequently write new tools and extensions in either ratfor or the new language. This has the advantage of enabling use of the newer language constructs. However, the disadvantage is having to maintain two versions of everything, and of having to translate new utilities from one language to another. If the new language required changes in calling sequences to the library, this maintenance/translation task could become considerably difficult. The new language would most likely have capabilities not available with the ratfor/FORTRAN combination, thus making translation back to ratfor difficult or impossible. Alternately, once the original utilities were translated to the new language, ratfor itself could be phased out and all future development would continue in the new choice. But how many (generally older) systems would we be abandoning that did not support the new language? It is clear from the interest in the Software Tools Users Group that program development in ratfor will continue for some time. Therefore, there would be a considerable advantage in not losing the connection to ratfor.

Thus, the first approach maintains portability and allows movement of the package to new micros, but does not give the full benefit of a more powerful and modern language. The second approach allows use of a more powerful language but requires either maintenance of multiple versions or abandonment of systems not able to support the new choice and isolation from future program development in ratfor.

Should the Users Group decide that remaining tied strictly to FORTRAN is undesirable, we must then agree not only on the approach but also the language. Should approach #1 be chosen, then a full syntax for ratfor as a language would have to be determined, keeping in mind that it would have to be translatable to FORTRAN, C, Pascal, or whatever else was deemed necessary. The existence of public-domain versions of YACC and LEX (at LBL) would make this a reasonable task. Having the Users Group agree upon a syntax, however, could be difficult.

Alternately, should approach #2 be taken, the new language would have to be chosen. Pascal has been suggested as a possibility, especially since a version of the original Kernighan-Plaugher book has appeared in Pascal. However, the conversion was only a limited success and ay least one of the authors feels that Pascal is simply not powerful enough or appropriate enough for utilities of this type. (See "Why Pascal is not My Favorite Programming Language" by Brian Kernighan, Bell Labs Technical Note 100.) The existence of the original utilities in Pascal is not a significant benefit towards the conversion either, for many, many man-years of effort have gone into extending and developing the VOS package since the original book appeared. Upgrading the Pascal tools to match the standards would be essentially starting over from scratch.

Other languages like Ada and Smalltalk may be appealing, but they simply are not available on enough machines yet to make them viable alternatives.

C, then, represents an obvious next choice. C offers power and functionality suitable to the VOS needs. C compilers are available on most micros and an increasing number of minis and mainframes. C is also particularly attractive because of its resemblance to the ratfor syntax. Having the VOS code available in C, would seem desirable.

Assuming we decided on C, the next problem would be translating the existing package. Rewriting the 60,000 or so lines of code by hand is possible, though not appealing. (However, with the help of the C Users Group it might be done.) On the other hand, Unicorn Systems has done a preliminary analysis of the ratfor used in the tools and determined that an automatic translator could be developed which would generate C from normal ratfor input. The few instances impossible to translate (toggled lines, FORTRAN built-ins) are infrequently used and could be flagged from human intervention. The translator would be no more difficult to write than the ratfor compiler of approach #1. The code produced by automatic translation would not take advantage of C constructs such as structures or pointers, but it would provide a working version of the existing utilities.

There is, however, one more consideration if translation to C were decided upon. The cling sequences of the ratfor library and primitives were designed to match FORTRAN's capabilities. Maintaining the same sequences would be clumsy and inelegant in C. Should we choose to move to C, and eventually abandon ratfor, we might want to adjust out library to be more appropriate to the new language (and perhaps more consistent with the equivalent Unix routines). This, however, would further broaden the gap between the ratfor and C versions and make maintenance/translation between the two more difficult.

Thus, moving to a new language requires several major decisions: a "ratfor" compiler with back-ends; an automatic translator, or hand-translation; C, Pascal, or ?; maintain the same library or adapt to the new language; support ratfor and the new language, or phase out ratfor?

Because of its committment to the micro world, Unicorn Systems is interested in the possibility of tools in a more appropriate language and is very seriously considering developing an automatic ratfor-to-C translator. We also feel the library should be re-evaluated and calling parameters adapted to be more appropriate to C and more consist with UNIX, with the ratfor versions eventually being phased out. We are very interested in the response of the Users Group, and Unicorn customers, to this proposal and welcome feedback either to us directly or to the Users Group via their newsletter.

Date: _____

Software Tools Users Group

Application for Membership or Change of Address

Name: _____ Title: _____

Affiliation: _____

Address: _____

City: _____ State/Country: _____

Telephone: ( ) _____ Zip Code: _____

Computer Mail Address: _____

☐ Check here if you wish to allow this information to be included in the published Software Tools Users Group mailing list.

☐ Check here for Change of Address only.

Machines and systems running the Software Tools: _____

_____

_____

Utilities / library functions you have implemented:

☐ The standard package (as distributed by STUG)

☐ The original package (Kernighan-Plauger)

☐ Additions/Other: _____

_____

_____

Other systems for which you are interested in obtaining/implementing Tools packages: _____

_____

Special interests: _____

_____

_____

Membership fees are $15 per year. Add $5 for overseas airmail, if desired. Make check payable to the *Software Tools Users Group* and mail to:

the Software Tools Users Group
1259 El Camino Real, #242
Menlo Park, Ca. 94025

1983 Software Tools Catalogue

If you have done work with Software Tools and would like to be listed as an implementor in the 1983 Software Tools Catalogue, please return this questionaire to:

the Software Tools Users Group
1259 El Camino Real, #242
Menlo Park, Ca. 94025
attn: 1983 Catalogue

Name _____

Affiliation _____

Address _____

City _____

State/Country _____ Zip Code _____

Telephone _____ Computer Mail Address _____

Machine(s) Running Tools: _____

_____

Please check areas where you have:

- Done work to develop new tools or extend existing tools
- Have need of new tools

Work On	Need		Work On	Need	
☐	☐	Editors	☐	☐	Languages
☐	☐	Text Formatters	☐	☐	Macro Processors
☐	☐	Shell	☐	☐	Tools in other than Ratfor
☐	☐	Ratfor	☐	☐	Other:

Description of Your Tools Work: _____

_____

_____

_____

_____

_____

_____

Availability (including fee, if any): _____

_____

_____

PLEASE POST PLEASE POST

Winter UNICOM 1983

Pre-announcement

UNICOM is the conference and exhibition of the *UNIX** community sponsored jointly by */usr/group*, the *USENIX Association*, and the *Software Tools Users Group*.

Place:	Town and Country Hotel San Diego, California
Dates:	Tuesday through Friday, January 25-28, 1983
Vendor Exhibition:	Tuesday through Thursday

/usr/group is a non-profit corporation formed by and for people who have a commercial interest in *UNIX* and *UNIX*-like operating systems and associated software tools. Key */usr/group* activities include sponsoring twice-yearly conferences where, in addition to meeting contacts, attendees have the opportunity to learn about software and hardware products being offered in the *UNIX*-related marketplace. Also provided are products catalogs and a bi-monthly newsletter.

The *USENIX Association* is an organization providing a forum for the exchange of technical information about the *UNIX* system and associated hardware and software. Such services are provided through semi-annual conferences, bi-monthly newsletters and software exchange.

The *Software Tools Users Group* is focused on a set of license-free, *UNIX*-like utilities and system calls, written in Ratfor and Pascal. When run in conjunction with almost any local operating system, the Tools package presents a virtual operating system interface consisting of a virtual machine (system calls or "primitives"), utility programs, and a command language, thus achieving inter-system uniformity over a variety of operating systems. Originating from Kernighan and Plauger's book *Software Tools*, the enhanced package now includes programs for text formatting, mail systems, enhanced Ratfor preprocessors, a source code control system, command line interpreter similar to the *UNIX* Shell, and many other utilities.

If you wish to receive the pre-registration packet (and did not get a copy of this announcement by mail) send your name and address to:

UNICOM
P O Box 385
Sunset Beach, CA 90742

Local Arrangements Coordinator is
Judith DesHarnais
ucbvax!sdcsvax!sdchema!jfd
(213) 592-3243

---

* *UNIX* is a trademark of Bell Laboratories.

UNICOM Conference Proceedings
Order Form

Proceedings of the January, 1983 UNICOM Conference are available from the Software Tools Users Group. Members of **STUG**, **USENIX** or **/usr/group** receive a $5 discount on each copy they order.

Please make checks payable to the *Software Tools User Group* and mail them, together with this form, to:

the Software Tools Users Group
1259 El Camino Real, #242
Menlo Park, Ca. 94025
attn: UNICOM Proceedings

☐ STUG, USENIX or /usr/group member: _____ copies at $15 each.

☐ Non-member: _____ copies at $20 each.

(**Note:** Overseas postage is an additional $5 per copy.)

Total number of proceedings ordered: _____
Total amount enclosed: $_____

Ship to:

Name _____
Address _____
City _____
State/Country _____ Zip Code _____

Lawrence Berkeley Laboratory
CSAM – 50B/3238
University of California
Berkeley, CA 94720