University Computer Center Newsletter

University of Minnesota
Twin Cities

March 1985
Volume 19, Number 3

Barbara Wolfe Becomes Assistant Vice President for Information Systems

On January 1 of this year, Dr. Barbara Wolfe took up her new position as Assistant Vice President for Information Systems. Her primary responsibilities are policy and planning for computing services on the main and coordinate campuses. She is also organizing a new Department of Telecommunications. In a recent interview she spoke about the computing environment she foresees for the University, about organizational relationships within computing services, about the responsibilities of her new job, and about her own background.

Dr. Barbara Wolfe

A Seamless Computing Environment

One of the chief attractions of the new position, Dr. Wolfe says, was the opportunity to develop a range of services for computer users at the University. All the components exist here to create "a seamless computing environment, from the workstation on the desk to the supercomputer."

When she considered taking the position, she noted that "supercomputers were a reality" at the University, and that computing center staff already had "four years of learning the operation and management of supercomputers." Also, "telecommunications were well under way, and there seemed to be a commitment to install adequate telecommunications to build that seamless environment." She knew that "there were many workstations being installed and many people here were knowledgeable and interested in that area." She prefers to see all these components working together, rather than distinguishing among the kinds of services offered by the computing centers and the kinds of users served.

The University took an important step when it installed the CRAY-1 supercomputer in 1981. Dr. Wolfe says that this progress was noted around the country, at a time when academic computing centers seemed to be lagging behind their counterparts in private industry. She envisions much greater use of supercomputing resources by faculty and graduate students as these machines and their capabilities become more widely known and better developed. Supercomputing will even play a part in UCC's instructional services as students and faculty members begin to see the usefulness of these machines: Dr. Wolfe says that "graduate students are soon going to expect to be able to use a supercomputer."

The telecommunications system now being installed on the main campus will make Dr. Wolfe's vision of an integrated computing environment a reality. Capable of handling both voice and data transmissions, the communication network will link workstations and the larger mainframes and supercomputers. Her primary responsibility in this area is to oversee the management of the installation project and to organize a new Department of Telecommunications that will manage the system. The people in this department must work with the contractor and consultants so that they will be able to operate the systems after the contractor and consultants have left.

With regard to microcomputers and workstations, Dr. Wolfe wants to offer the consulting and advising expertise of the computing centers to University departments and colleges to help them develop their own microcomputing resources: "Each college should not have to relearn everything we've learned in the last 25 years about managing computing," she says. She would like to see a workstation in the office of every faculty member who wants one, and she understands that different disciplines will have different computing needs. "Different disciplines will decide that certain kinds of machines better suit what they need," Dr. Wolfe says, and her concern is to standardize communications for workstations and to coordinate efforts in this area with the help of College Computing Coordinators appointed in each college. The computing centers' role is to be sure that computers are used effectively to accomplish the instructional and research objectives defined by the academic disciplines. Dr. Wolfe is especially sensitive to the needs of users; she has long been active in user services through her work with national organizations.

Computing Services Organization

Dr. Wolfe is certain that the organizational structures that deliver computing services to the University community will change, if only because technological change in this field makes it imperative. While not prepared to say what reorganizations she is planning, she did state emphatically that if someone's job description had not changed within the last five years, that employee was probably out of touch. She is currently meeting with computing center directors to learn more about their organizations and assess their needs.

Dr. Wolfe is also concerned with the computing centers' relationships with the rest of the University community and with the private sector. After only a short time on the job, she has a wide-ranging knowledge of University departments and colleges, and she is well on the way to establishing productive relationships with department heads and deans. With respect to the private sector, she is alarmed that so much computing equipment is being donated to the University by vendors. The University often cannot find resources for operating and staffing the services required for successful use. At the same time, she hopes to build on the good will that seems to be in the private sector and to maintain close relationships with these companies.

At the moment, financing the delivery of computing services is largely on a fee-for-service basis. Dr. Wolfe foresees some shift in the next few years, with more funding coming from central administration and somewhat less from charges to users inside the University.

The Assistant Vice President for Information Systems is responsible for policy and planning for all centrally funded computing centers in the University. It is a new position, created partly in response to the Report on Computation, Communications, and Information published in 1982, and it combines the positions formerly held by Peter Roll, Special Assistant to the Vice President for Academic Affairs, and the late Frank Verbrugge, Director of University Computer Services. A significant part of Dr. Wolfe's responsibility is to advise the University Budget Executive Committee on planning and funding the computing services at the University. She hopes that her new position will gain more recognition for and better representation of computing resources and needs before this important committee. All the directors of the computing centers—UCC, West Bank, Health Sciences, St. Paul, Duluth, Morris, Waseca, and Crookston—now report to her. Dr. Wolfe reports to Vice President for Academic Affairs (and interim President of the University) Kenneth Keller.

Dr. Wolfe obtained graduate degrees from Wayne State University, where she also became Assistant Director of the Computing Services Center. She was responsible for all client services functions in that position. In 1982, she became Associate Vice President for Computing Services at the State University of New York at Albany, where she was responsible for academic and administrative computing and for integrating all computer services including mainframes, minicomputers, microcomputers, software, and communications. Dr. Wolfe has also taught classes and seminars in statistics throughout her career. She is a member of the Association for Computing Machinery and has held leadership positions in the Special Interest Group on University and College Computing Services. She has presented numerous papers at national conferences.

(Parker Johnson)

(Photo by Thomas Foley)

Computer Graphics Workshop

During the week of April 8, 1985, the University Computer Center will sponsor an introductory workshop on computer graphics. The workshop will focus on MOVIE.BYU, a graphics package recently available on the University's CRAY that allows the user to design, manipulate, and display two- and three-dimensional graphics data. Workshop lecture sessions will be held in the evening; the instructor will be one of the developers of the MOVIE.BYU graphics software system, Dr. Hank Christiansen of Brigham Young University.

Topics for the workshop sessions will include:

- An overview of the MOVIE.BYU system
- Model generation capabilities
- Mathematics of scene manipulation
- Shading techniques and raster display capabilities
- System animation capabilities
- Interfacing MOVIE.BYU to other data generation software

Slides and videotapes that illustrate the graphic generation and display capabilities of the MOVIE.BYU system will be utilized in the workshop. Participants will also be expected to participate in a number of "hands-on" sessions during the day.

The price of the workshop is $300. Enrollment is restricted to members of the academic community and will be limited because of hardware constraints. For further information on the workshop and registration, contact Jerry Stearns at 376-8806.

OPTIMIZATION OF CRAY FORTRAN PROGRAMS

Part II: General Source Code Modifications

In Part I of this article, which appeared in January's newsletter, we described a general optimization strategy. In this month's article, we still assume that you have found optimization of your program to be worth the time and effort required and that you have selected and incorporated into your program the appropriate data structures and algorithms. Be sure that you have defined the problem clearly and expressed the solution in its simplest form, using straightforward code and avoiding the temptation to optimize local sections of your program prematurely. Applying sound programming practices to produce good system-independent FORTRAN source statements will usually go a long way toward helping a compiler produce good object code. References 1, 2, 5, and 7 (listed at the end of this article) provide good discussions of optimizing any FORTRAN program. In Part III, which will appear in a future issue of the newsletter, we will use References 3, 4, 6, and 8 to suggest CFT and CRAY-specific optimizations.

Getting Started

We will now narrow our focus to the level of specific regions of FORTRAN statements that are responsible for a relatively large fraction of the total execution time. As we said before, you can identify time-consuming subprograms with a profiling utility like the flow trace option available by using the 'ON=F' parameter on the CRAY FORTRAN compiler statement, CFT. Again, you can estimate the practicality of modifying the code by considering the percent of the total execution time spent in these critical regions, the percent potential improvement that might be realized by changing the code, and the effort these changes will take. The modified source program should be run to validate the results against reference executions and to ensure that a more efficient program was indeed created. You should document all changes, and perhaps even save the original statements as comments, if the new FORTRAN solution obscures the underlying problem.

Optimization changes targeted at the source statement level fall into two groups: those that are applicable to most FORTRAN implementations and those that are made with a specific compiler and computer in mind. We shall consider compiler-independent modifications this month, and in Part III of this series we will discuss the programming principles appropriate to CRAY FORTRAN programs that help the CFT compiler to produce efficient instructions for the CRAY vector processing hardware.

FORTRAN Optimization

Optimizations attempt to reduce the number of operations, to replace one operation by an equivalent and faster operation, or to organize the data so that it is accessed in a sequential manner during program execution. The first case may arise, for example, when invariant calculations within a DO-loop—calculations that do not depend on the changing conditions defined by the loop—are repeated with each pass. Programs may also contain operations that require the computer to maintain a relatively large number of additional bookkeeping operations. For instance, small subroutines, called repeatedly with many arguments, incur a large overhead.

The second case, replacing slower operations with faster ones, could arise, for example, when a given calculation uses division to achieve the same results that could be reached by multiplication. (Most computers multiply faster than they divide.) Another example is using formatted I/O instead of unformatted I/O for files that will only be read by FORTRAN programs. The final case, in which data is not accessed in a sequential manner, can cause problems with either page faults or memory bank conflicts, depending on the computer you are using.

An optimizing compiler will attempt to apply these same principles. Check the appropriate FORTRAN reference manual to find out what optimizations are being performed by the compiler you are using. Typically, optimizing compilers can perform the following optimizations:

1) Move invariant computations outside of loops.
2) Evaluate constant expressions.
3) Replace constant exponentiation with fewer multiplication operations.
4) Eliminate redundant expressions.
5) Eliminate subscript evaluations involving multiplication by adding multiples of the control variable to a base index.
6) Eliminate variables whose values are not used in later computations.

The programmer with a global understanding of the logic and purpose of the program can often optimize the source statements in ways not available to the compiler. We will now consider some of the techniques that can decrease execution time.

DO-loops

In typical FORTRAN programs, much of the execution time is spent within DO-loops. Therefore, you usually achieve the greatest results from optimizations when you concentrate on modifying the heavily-used loops and the first priority should be innermost DO-loops.

Each iteration of the DO-loop is accompanied by loop-control overhead. Because of the magnitude of the loop bookkeeping, avoid constant parameter DO-loops with low iteration counts, and replace them with their explicit, expanded source statement equivalent. In some cases you can combine several loops into one so that they share the cost of the loop overhead. When you nest DO-loops, make innermost the loop that carries the greatest workload. Furthermore, most of the compiler's register usage optimization is reserved for the innermost loop.

Move invariant code (values and expressions that do not change with each pass) outside the loop to avoid needless repetition. Restructure loops with invariant logical decisions by forming two or more new loops that are executed dependent on the outcome of the invariant decision.

Write common subexpressions identically to help the compiler recognize them, or eliminate them by storing the result in a temporary variable. When you reference arrays within the loop, keep subscript expressions simple, since most compilers can only optimize subscripts of the form "constant*integer variable plus or minus a constant." Another technique, called unrolling the loop, involves increasing the increment step for the next outer loop control variable and adding array references with appropriately adjusted subscript expressions to do more work in the innermost loop. (See Reference 9.)

To make the most efficient use of loop optimizations, you should know the following:

1) Most FORTRAN intrinsic functions are done inline and do not inhibit DO-loop optimization.
2) Try to avoid references to external functions and subroutines. If you must have an external reference inside a DO-loop, then pass as few dummy arguments as possible, usually by putting most or all arguments in a COMMON block.
3) A loop with an external function whose results depend on the loop control variable may be replaced by an array reference previously defined by a subroutine that calculates the corresponding function results as an array.
4) Use logical or block-IFs rather than other branching statements. When you set up a series of logical tests, order them so that the more likely conditions are tested first.
5) If you want your loop to execute quickly, do not do any I/O in your innermost loop.

Arithmetic Expressions

Arithmetic expressions offer several opportunities for optimization. Strength reduction is the replacement of loop arithmetic expressions by equivalent but faster statements using faster operations. You can apply strength reduction, remembering that addition and subtraction are usually faster than multiplication, which is faster than division, which is faster than exponentiation. (See Table 1.) Replace exponentiation expressions of the form "x**n.5" with an equivalent expression using the intrinsic SQRT function: "x**n*SQRT(x)". Reduce constant expressions and eliminate unneeded temporary variables by combining statements. You can often use algebraic manipulations like factoring expressions or Horner's rule to simplify computations.

Strength Reduction Table

operations, sorted by speed	example loop statement	replace with:
+ −		
*		I3 = 0
	DO 10 I = 1,K	DO 10 I = 1,K
	I3 = I * 3	I3 = I3 + 3
	...	...
/		ZI = 1.0/Z
	DO 10 I = 1,K	DO 10 I = 1,K
	X = V(I)/Z	X = V(I) * ZI
	...(Z invariant)	...(Z invariant)
**		XI = 1.0
	DO 10 I = 1,K	DO 10 I = 1,K
	XI = X**I	XI = XI*X
	...(X invariant)	...(X invariant)

TABLE 1.

Avoid mixed-mode arithmetic as this incurs additional overhead. However, this does not apply to real**integer exponentiation.

Miscellaneous

Where possible, DATA statements should be used to initialize variables and arrays to save on execution time. PARAMETER statements allow the definition of symbolic constants that can be used to define array bounds and loop parameters. Using symbolic constants is more efficient than using variables that are initialized and never changed.

In summary, concentrate your optimization efforts on inner DO-loops. Make sure you are actually getting improvements while maintaining correct results. Keep in mind the following principles for making your programs faster: (1) reduce the number of operations, and (2) replace slower operations with faster ones.

REFERENCES

1. "Optimization of FORTRAN Programs," Appendix H of the *M77 Reference Manual*, first edition, (UCC, 1983).
2. Jon Louis Bentley, *Writing Efficient Programs*, (Prentice-Hall, 1982).
3. *CFT Reference Manual*, Publication No. SR-0009, Cray Research, Inc.
4. Bonnie Gacnik, "Report on Efficient I/O on the CRAY-1 Computers at NCAR," *The Record*, Vol. 5, No. 9, September 1, 1984.
5. Michael Metcalf, *FORTRAN Optimization*, (Academic Press, 1982).
6. *Optimization Guide*, Publication No. SN-0220, Cray Research, Inc.
7. Dennie Van Tassel, *Program Style, Design, Efficiency, Debugging, and Testing*, (Prentice-Hall, 1978).
8. Jack J. Dongarra & Stanley C. Eisenstat, "Squeezing the Most out of an Algorithm in CRAY FORTRAN," *ACM Transactions on Mathematical Software*, Vol. 10, No. 3, September 1984, pp. 221-230.
9. Jack J. Dongarra & A. R. Hinds, "Unrolling Loops in FORTRAN," *Software Practice and Experience*, Vol. 9, No. 3, March 1979, pp. 219-226.

(*Kurt Richards*)

CRAY News

MINOS 5 NONLINEAR PROGRAMMING PACKAGE

We have installed the MINOS (Modular In-Core Nonlinear Optimization System) Version 5 package on the CRAY as a user library named MIN5LIB. You can access MIN5LIB with the LDR control statement as LDR(LIB=MIN5LIB). The package can solve problems in linear programming, unconstrained optimization, linearly constrained optimization, and nonlinearly constrained optimization. The optimization problems must have smooth nonlinear parts; that is, they must have known gradients. Gradients needed for optimization can be provided by the user, or approximated by MIN5LIB, or a combination of the two. A gradient-checking option is available to help users test their code. Problems can be large-scale but must be able to fit in memory, although sparse matrix methods are used to minimize the amount of memory required. You can pick a starting guess for the solution or let MIN5LIB choose one.

For optimization problems, you must provide MIN5LIB with a subroutine to evaluate the objective function (FUNOBJ) and a subroutine to evaluate the constraints, if any (FUNCON). As an option, you can also provide a subroutine for problem modification (MATMOD). The only routine in MIN5LIB called by the user is MINOS1, which runs complete problems defined for the linear part by data files and for the nonlinear part by subroutines FUNOBJ, FUNCON, and MATMOD. MIN5LIB requires you to provide a work space in blank COMMON and to call MINOS1. You must provide a data file (called the SPECS file in the user's manual) for specifying information to MIN5LIB about the number of unknowns, number of constraints, number of nonlinear variables, tolerances, etc. Within this file is the location of the file containing the coefficients of the linear part of the problem in a special format called MPS (Mathematical Programming System). This format is used by commercial linear programming packages, such as APEX, as a standard way to input data, particularly for large problems. Also included in the SPECS file are locations of optional files for saving solutions to be used elsewhere, like start-up values for other runs.

MINOS Version 5 was written by Bruce Murtagh and Michael Saunders of the Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA 94305. Together with the UCC document WRITEUP(MIN5LIB), their *MINOS 5.0 User's Guide* provides the information needed to use MIN5LIB on the CRAY. A copy of the *User's Guide* is on reserve in the UCC Reference Room, 140 Experimental Engineering, and is available from the Systems Optimization Laboratory at Stanford. Also on reserve in the Reference Room are additional reports giving algorithm details.

We tested MIN5LIB on programs provided with the package, and all the test programs ran satisfactorily. UCC regards MIN5LIB as a high quality, full capability optimization package. It has a good user's manual and provides a large number of user options. It is, however, somewhat more complicated to use than its main competitor, the GRG2 (Generalized Reduced Gradient) optimization library. Questions about MIN5LIB should be directed to Mike Frisch, 612/376-1636. To see sample job runs, with problem descriptions, input, and MIN5LIB output, refer to WRITEUP(MIN5LIB).

Note: Our contract with Stanford limits use of MINOS to non-commercial research purposes and to University students and staff.

(*Mike Frisch*)

GAUSSIAN 82 ON THE CRAY

We have installed the Gaussian 82 package on the CRAY. It performs molecular orbit calculations used in chemistry. Along with WRITEUP(GAUS82), the Gaussian 82 *User's Manual* gives information about using the package. The manual is on reserve in the UCC Reference Room, 140 Experimental Engineering.

(*Mike Frisch*)

SHAZAM INSTALLED

SHAZAM 4.5, a sophisticated batch-oriented econometrics analysis package, has been installed on the CYBER CA and the CRAY-1. SHAZAM may be accessed on the CYBER by typing:

```
FETCH,SHAZAM.
SHAZAM.
```

It may be run on the CRAY simply by including the command

```
SHAZAM.
```

in your CRAY job control deck.

A copy of the SHAZAM manual may be obtained by typing:

```
WRITEUP,SHAZAM/L=SHAZMAN,
  PT=AS.
ROUTE,SHAZMAN,DC=PR,
  EC=A9, UN=site,BIN=bin.
```

The manual runs about 150 pages.

A more comprehensive article on SHAZAM including examples will appear in a subsequent newsletter; please bear with us. Questions regarding SHAZAM may be addressed to:

Professor Henry Hwang
Agricultural and Applied
  Economics
130d Classroom Office Building
(376-2936)

CYBER Notes

NEW VERSION OF PASCAL ON CYBERS

On March 24, we will upgrade versions of CYBER Pascal. Future Pascal (4.1.C) will become the current version, and a new future version will be put up. Past Pascal (3.4) will not change. WRITEUP(PASCALF) describes the differences between current and future Pascal.

(*Dave Bianchi*)

CLEAN77 INSTALLED ON CYBERs

CLEAN77, a program from Purdue University to reformat and clean up FORTRAN source code, has been installed on the CYBER systems at UCC. CLEAN77 is similar to the TIDY program that is already on the UCC CYBERs. It can process both FORTRAN-77 and FORTRAN-66 programs while TIDY can only process FORTRAN-66 programs. CLEAN77 has some additional options that are not available in TIDY. CLEAN77 can accept commands from a separate COMMAND file, indent IF -THEN -ELSE blocks, add blank comment statements before and after comment statements, and control the number of characters per line. However, there are some TIDY features that are not available in CLEAN77. TIDY can change FORTRAN II function names to FORTRAN IV names and can check and correct a variable with more than seven characters.

Information on using CLEAN77 is available in a 10-page document you can obtain with the command:

```
WRITEUP(CLEAN77).
```

This sample SUBMIT job shows how CLEAN77 transforms a program into a more readable form.

```
/JOB
JOBCLEA. CLEAN FILE
PROOLD USING DEFAULT
VALUES OF COMMANDS
USER,0,0.
NOEXIT.
UNLOAD,OUTPUT.
ACQUIRE,PROOLD.
CLEAN77,P=PROOLD,
N=PRONEW,
L=CLNOUT.
DAYFILE,CLNOUT.
RETAIN,PRONEW,CLNOUT.
/EOF
```

The program in PROOLD is

```
      PROGRAM TRIAL
C     HOW NICE CLEAN77 IS ???
      READ(5,91)A,B
   91 FORMAT(2F8.2)
      CALL SETC(A,B,C)
      WRITE(7,91)C
      STOP
      END
      SUBROUTINE SETC(A,B,C)
      IF(A.GT.0.0)THEN
      IF(A.GT.B)THEN
      C=A
      ELSE
      C=B/A
      ENDIF
      ELSE
      C=-A/B
      ENDIF
      RETURN
      END
```

The result in PRONEW is

```
      PROGRAM TRIAL
C
C     HOW NICE CLEAN77 IS ???
C
      READ (5,10) A,B
   10 FORMAT (2F8.2)
      CALL SETC (A,B,C)
      WRITE (7,10) C
      STOP
      END
      SUBROUTINE SETC (A,B,C)
      IF (A.GT.0.0) THEN
         IF (A.GT.B) THEN
         IF C = A
         ELSE
         IF C = B/A
         ENDIF
      ELSE
         C = -A/B
      ENDIF
      RETURN
      END
```

(*Sharad Gavali*)

MINITAB 82 TO BECOME CURRENT

On March 26, 1985, MINITAB 82, currently the FUTURE version, will become the default version of MINITAB on all the CYBERs, and the current version of MINITAB will become the past version. MINITAB 82 manuals should be available in spring quarter in the Minnesota Bookcenter in Williamson Hall or the H. D. Smith Bookstore on the West Bank.

(*Bruce Center*)

The Classifieds

FOR SALE

DEC VT-100 terminal with manuals, $600. Call Dave at 378-9497.

PHONE NUMBERS

Access:	
CYBER(CA)—10, 30 cps	376-5730
—120 cps	376-5706
MERITSS(ME)—10, 30 cps	376-7730
—120 cps	376-7120
VAX/VMS(VA)—(autobaud)	376-9770
Budgets	373-2521
Computer-Aided Instruction	376-2975
Computer Hours (recorded message)	373-4927
Consulting	
HELP-line	376-5592
7 a.m.-7 p.m., Monday-Friday	
Statistics Packages	376-1761
1-2 p.m., Monday-Friday	
Data Bases	376-1761
10-11 a.m., Monday-Friday	
Microcomputers	376-4276
9:30 a.m.-noon and 1:30-4 p.m., Monday-Friday	
Text Processing	376-2944
1-4 p.m., Monday-Friday	
Contract Programming	376-1764
Data Base Applications	376-1764
EDUNET Liaison	373-7745
Engineering Services	376-1023, 376-8153
Equipment Purchase/Information	376-8153
Experimental Engineering I/O	373-4596
Graphics Software	638-0541
HELP-line	376-5592
7 a.m.-7 p.m., Monday-Friday	
HOURS-line (recorded message)	373-4927
Information, Experimental Engineering	373-4360
Information, Lauderdale	373-4912
Instructional Labs	376-2703
Instructional Services	373-7745
Lauderdale Computer Room	373-4940
Lauderdale Services	638-0523
Lauderdale Services Manager	373-7538
Lauderdale Users' Room	373-4921
MERITSS	373-7745
Newsletter Subscription	376-1491
Permanent File Restoration	376-5605
Professional Services Division (PSD)	376-1764
Project Assistance	376-1764
Reference Room	373-7744
Remote Batch (RJE) Services	376-2703
Short Courses	376-8806
Shuttle Bus Service	376-3068
System Status (recorded message)	373-4927
Tape Librarian: see Lauderdale Services	
Text Processing Services	376-2943
User Accounts	373-4548

OPERATING HOURS

	CYBER (CA)	Low rate	CRAY (CR)	MERITSS (ME)	VAX (VA)
M-F	7 a.m. - 4 a.m.	8 p.m. - 4 a.m.	7 a.m. - midnight	7:45 a.m. - 3:30 a.m.	8 a.m. - 6 a.m.
Sat	4 a.m. - 5:15 p.m.	4 a.m. - 5:15 p.m.	7 a.m. - 5 p.m.	7:45 a.m. - 3:30 a.m.	24 hours
Sun	4 p.m. - 1 a.m.	4 p.m. - 1 a.m.	4 p.m. - midnight	4 p.m. - 3:30 a.m.	24 hours

PUBLIC LABS—TWIN CITIES CAMPUS

Location	Batch	Interactive	Micro
East Bank			
Arch 148		X	X
CentH		X	
ComH		X	
DiehlH 270, 207		X	
EltH 121, 125		X	
EltH N640	X		
Exp Eng 130		*	
FolH 14, 14a	X	X*	X
LindH 26	X	X	
MechE 308		X	
Physics 69		*	
SanfH		X	
TerrH		X	
VincH 4		X	
WaLib B9		X	
West Bank			
BlegH 25		*	
BlegH 90	X		
BlegH 140		X	
MdbH		X	
OMWL 2		X	
SocSci 167			X
St. Paul			
BaH		X	
ClaOff 125	X	X	

* Research cluster; access to CYBER CA and VAX/VMS

X in interactive column indicates access to MERITSS

Contents

Michael M. Skow, Acting Director

The *UCC Newsletter* is published monthly by the University Computer Center. Deadline for articles is the 10th of the month preceding publication; deadline for short announcements is the 15th. The *Newsletter* is edited and coded for typesetting at the Computer Center, then typeset on a Linotron 202 and printed at the University of Minnesota's Printing and Graphics Arts Department.

Comments, suggestions, articles, and announcements should be directed to the editor, 227 Experimental Engineering, (612) 376-1491.

The University of Minnesota adheres to the principle that all persons should have equal opportunity and access to facilities in any phase of University activity without regard to race, religion, color, sex, national origin, handicap, age, or veteran status.

Nonprofit Org.
U.S. Postage
PAID
Minneapolis, Mn.
Permit No. 155

University Computer Center Newsletter

User Services
227 Experimental Engineering
University of Minnesota
208 Union Street SE
Minneapolis, Minnesota 55455

UNIVERSITY ARCHIVES
10 WALTER LIBRARY
UNIVERSITY OF MINNESOTA
EAST BANK
117 PLEASANT STREET SE
MINNEAPOLIS
MN 55455