TENTACLE

SEPTEMBER 1981 Volume 1 Number 2

CONTENTS

Department Overview 1
Computer Operations Division 2
Math and Statistics News 5
NLTSS - What it is 6
Coordination Center Notes 8
CHORS II Meetings & User-Requirements Document . 10
Dynamic Memory Management 12
Dependent Libraries 14
Enterprise Directories 17
Consulting Office Commentary 18
GIFFGAFF 20
Octopus Communiqués
  Utility Routine CHECK 23
  Approximating a Function for Computation 25
Computist's Corner 27
Computer Education 32
New and Newly Revised Documents 30
Wanted 33
Public File Usage Statistics 33
Submitting Articles 34
Octogram Summary 35
In The Beginning 42
The Octopus's Garden 43
Puzzles by Henry Larson 44
Index 46

COMPUTATION DEPARTMENT
LAWRENCE LIVERMORE NATIONAL LABORATORY

Tentacle is published monthly by

User Systems Division
Computation Department
Lawrence Livermore National Laboratory
P. O. Box 808
Livermore, California 94550

Editors
Gail Whitten, Ext. 2-3704
Ted Stullich, Ext. 2-4703

Consulting Editor
Gene Ledbetter, Ext. 2-4316

Editorial Assistant
Sharyn Vantine, Ext. 2-4361

Illustrator
Joanne Perra, Ext. 2-3762

To submit an article for publication in the next issue, contact one of the editors above or send the article to *Tentacle*, L-301, before September 15, 1981.

Material in this issue is current to about August 20, 1981.

The text portion of this issue is available online:
TRIX AC!PRINT!NIP SEP81TNT BOX *ann id*!END
or view on TMDS with:
TRIX AC!OD(SEP81TNT)!TV*mon*

Quick Reference

	Ext.
Software Consulting Office	23724
Coordination Center	24531
Mathematics & Statistics Software	29410
LASL Consulting Office	114*1*267*5745
MFECC Consulting	21544
Missing Output	23705
Computer Documentation Library	24257
Computer Education	24527
Tape Librarian	23734
Television Center	28990
Terminal Repair	23718
TID Library	25277

The Uncial character font, used as headings, was implemented on our system by Bob Kuhn in 1977 (described in UCID-17653), and included in the Hershey set by John Beatty.

DEPARTMENT OVERVIEW

CHORS II – Initial Plans

The Computation Department has requested funding in FY '82 for a "first phase" replacement of the existing CHORS subnetwork. We have established a CHORS II project with a design committee. One goal of the design committee is to produce a detailed plan with accompanying documentation which will address questions of workload, performance, capacity, cost, functionality and availability. The "user requirements" phase of this project is currently underway; the design committee is collecting input from the LCC user community. The current estimate of availability for the first phase documentation (external design document) is the first quarter of 1982.

Intelligent Terminals – A Survey

At LLNL, there are numerous computer-based resources which could, if desired, be connected by an appropriate network. There are significant advances expected in terminal, word processing, and office communication capabilities over the next few years which could have an impact on the way the Laboratory does its work. Currently there is a planning effort underway to investigate orderly and economic methods of providing new data networking and terminal facilities for the Laboratory.

As part of this planning process, an Octopus Intelligent Terminal Committee was formed to study intelligent terminals to be supported by the Computation Department for use in the Octopus Network and perhaps elsewhere in the Laboratory. The committee recently completed a survey which examined the current state-of-the-art of "intelligent terminals". A report of their findings and recommendations has been submitted to the Computation Department. One of their conclusions is that the Computation Department, and the Laboratory, should depart from our long standing practice of allowing only a small number of kinds of terminals connected to the Octopus Network. Rather, we should plan on the widest variety possible of terminals. The report also includes a review of some available micro-processor based terminals. Interested users can obtain a copy of this report by calling the department office at extension 2-3782.

The "G" Machine – A New Home

As most of you know, we retired our serial number 1, CDC 6600 "G" Machine last January. After several months on the government's equipment excess list, it has been determined that there are no government agencies interested in obtaining this classic. As a proper resting place for a machine which served us well for 17 years and represented a major milestone in the hardware evolution of supercomputers, we have made arrangements to ship the "G" Machine to the Digital Computer Museum in Marlboro, Massachusetts where it will be displayed with other "famous firsts" of computer history.

John E. Ranelletti
Department Head
Computation Department

COMPUTER OPERATIONS DIVISION

The Computer Operations Division (COD) is comprised of five groups: the Coordination Center Group (CCG), Mainframe Operations Group (MFG), I/O Operations Group (I/OOG), Training Management Group (TMG), and Operations Interface Group (IOG). The structure of the COD is shown on the outside back cover of this edition of *Tentacle*. Directly or indirectly, each group is involved in the Division's primary mission of providing a wide range of computer services to an advanced, demanding, and varied community of scientists and engineers.

COD personnel apply various skills and backgrounds to the numerous tasks involved in monitoring, evaluating, maintaining, and operating the hardware and software resources of the Livermore Computer Center (LCC).

Because of its exposure to the users, the Coordination Center Group, led by Don Emery, is the most conspicuous and easily recognized COD group. The CCG actively monitors all aspects of the LCC from 08:00 to 17:00, Monday through Friday. Each morning it carefully reviews reports describing the behavior of the center during the previous 15 hours (63 hours during the weekend). Any significantly anomalous behavior encountered during the "production hours" may result in a call to Joe Davison and Bob Bennett, the Division Leader's production accountability staff. Either Joe or Bob will investigate the event, depending on the details and the impact on the successful completion of the scheduled production work.

CCG personnel also monitor the status-information displays maintained on the Television Monitor Display System (TMDS). It is an integral part of the group's mission to interact with customers on any issue relating to the status and use of the LCC. The CCG is also responsible for establishing both the schedules and standards of various hardware maintenance programs and monitoring their effectiveness.

Another activity of the CCG is the pursuit of an active I/O quality-assurance program, headed by Bob Anderson. This program includes: collection and evaluation of individual system performance-history data; preliminary diagnosis of hardware and software errors occurring in the LCC; emergency modification to the divisional bank allocation structure and the initiation of comprehensive monthly performance reports for dissemination to senior Laboratory program management personnel.

The two largest COD groups are the Mainframe Operations and I/O Operations Groups. As their names suggest, they are primarily responsible for staffing and operating the LCC continuously, 24 hours a day, including all weekends and holidays; but the specific responsibilities of each group are quite different.

The Mainframe Operations Group, under the leadership of Chuck Cole, is charged with the operation of large worker systems, the various network components, and the supporting storage subsystems. The worker systems are: the CDC 7600's and the CRI Cray-1's; the major support systems, such as the CALCOMP

7110 Automatic Tape Library; the CDC 38500 Mass Store; and a host of other support systems necessary for the network to function. This last group includes the terminal concentrators, the RJET PDP-11's, the OSTRICH security system, and the TMDS. In sum, maintaining the availability of the immense LCC processing resource network is largely the mission of the Mainframe Operations Group.

The products of codes running on the worker systems must somehow find their way back to their originators. The processing and distribution of all forms of output are the primary mission of the I/O Operations Group, which is led by Curt Klutts. The group staffs sophisticated offline processing systems that convert the digital output from the worker systems operated by the Mainframe Group into the various final forms of output. Hardcopy alphanumeric and graphic output can be obtained from two 18,000-line-per-minute Honeywell printers in the Nonimpact Printer Systems (NIPS). Currently, the NIPS are producing more than 2,500,000 pages per month—only 20% of the LCC's monthly output. Approximately 7,- to 9,000,000 pages per month are produced in the form of 105mm microfiche via four Information International FR80 systems. The I/O Operations Group, besides operating computer-controlled I/O processing equipment, also staffs a modern photographic processing laboratory that processes both black-and-white and color computer-generated films. The Photo Lab routinely processes 105mm microfiche, 35mm slides and motion pictures in black and white or in color, and 16mm motion pictures; it also operates a high-resolution Phototypesetting Camera System producing 11-inch paper film. The Photo Lab maintains a quality control system that consistently yields excellent color work and black-and-white microfiche and substantially exceeds industry standards for sharpness, contrast, and consistency.

Besides staffing the NIPs and FR80's and operating the Photo Lab, the I/O Operations Group monitors the Computer Hardcopy Output Recording Systems (CHORS), administers the output distribution center, and participates heavily in researching and planning for the future evolution of I/O processing systems such as the Electron Beam Recorder. Also included in the I/O Operations Group are a tape-vault librarian, keypunch operators, and the User Services Representative, Forrest Allen. Forrest's main goal is to aid customers who have experienced input- and output-related problems.

The Mainframe and I/O Operations Groups are both supported by the Training Management Group, under the guidance of Doug Holt. Because high levels of operating performance could only be achieved by a well-trained operator staff, the COD embarked on the design, construction, and implementation of a formal computer operator training program. Doug and his group are well on their way toward meeting this ambitious goal through implementation of a Criteria Referenced Instruction (CRI) program. The CRI program, which is basically a self-paced tutorial approach to learning, makes excellent use of the knowledge of our most experienced senior operator personnel to develop course content. This program then employs the talents of Doug Brown to edit and produce curriculum elements and Lynn Groves to acquaint students with the CRI process and monitor and assist them as they progress. TMG's Solomon Plummer is responsible for the initial training of all newly hired computer operators while they wait for their "Q" clearance.

The Operations Interface Group is responsible for providing the COD with the software support it requires to collect and analyze performance history and accounting data. Its leader, Gus Wilgus, has a staff of programmers, programming associates, and computer programming technologists to assist him in the design, development, and maintenance of the wide variety of software tools necessary for accounting and analysis of the LCC's operation. Gus' group also handles the installation of "user numbers," public-file maintenance, I/O statistics, disk statistics, assignment of "alpha" and "effort" numbers, bank allocation schedules, and system-access combinations. This group is also well into the development stages of an automatic logging system using a locally designed minicomputer-based network built around HP 2621 and HP 2626 terminals interfaced through an LSI-11/23 to the Octopus Network. This system, CLARET, will replace a manual logging system that is not amenable to the creation of a large, easily manipulated data base. The data base will be comprehensive enough to permit more thorough, accurate, and timely analyses of the LCC's overall ability to meet the needs of its customers.

COD's varied missions are supported by a staff of four. The administrative, budget, and human resources responsibilities for the division are divided among them. The Division is supported in the crucial area of communications, travel, and clerical assistance by Aileen Dantzler and Elaine Rodriguez, without whom much of COD's work would grind to a premature halt.

In general, the COD's goals, my goals, fall into two broad categories: the first is to provide the greatest access to and encourage the most effective use of the enormous computer resources available at LLNL; the second is to analyze and evaluate the needs of the LCC's clientele so that the continued development of computer resources will meet future requirements efficiently, effectively, and quickly.

George Vranesh, COD
Ext. 2-4008

MATH AND STATISTICS NEWS

Last month we highlighted the services of the Library Software Section of the Mathematics and Statistics Division (MSD). This month we focus on the general mathematical activities of the Division, which can be classified as follows:

1. Consulting services
2. Large-code support
3. Algorithm development

Consulting services may range from a two-minute phone conversation to several man-months of work. MSD attempts to provide expert advice (and software when possible) on the most modern, efficient and powerful mathematical techniques available. Areas of expertise include ordinary and partial differential equations, numerical linear algebra, interpolation and approximation, special functions, and numerical integration. Examples of our longer-term consulting efforts are finding eigenmodes related to seismic behavior, modeling semiconductors, solving a system of ODE's related to viscous fluid flow, improving the implementation of Fowler-Wilson splines in the control of machine tools, and installing several subroutines involving special functions.

Large-code support efforts have improved the speed, accuracy, and stability of the Laboratory's largest, most heavily used codes. Our equation-of-state (EOS) packages on the Cray have given code developers new capabilities while saving huge amounts of machine time. An MSD team working with T-Division developed and implemented a revolutionary one-dimensional radiation-diffusion package that solves many problems that were previously intractable. Much of this work involves developing special linear system solvers, vectorization methods, and accurate but fast approximations.

Algorithm development refers to the need to devise new algorithms providing the capability to solve an ever-widening circle of problems. Currently MSD is enjoying national and international recognition in the area of ODE initial-value problems and in curve and surface fitting. Requests for software implementing these algorithms, such as LSODE, LSOD1, and MONDER, have come from all corners of the Laboratory and many parts of the world. The univariate monotone interpolation algorithm implemented in MONDER has been extended to functions of two independent variables. This new algorithm gives physically reasonable surface approximations where previous methods produced undesirable oscillations. It promises to be a significant improvement for EOS surface representations and has generated considerable interest throughout the world.

The above examples illustrate how MSD has helped others improve their problem-solving capabilities, save machine time, and increase accuracy. MSD will be glad to assist you also.

Ralph Carlson, NMS/MSD
Ext. 2-9359

NLTSS — WHAT IT IS

The following paragraphs are offered in an attempt to dispel some of the fog causing vague impressions of NLTSS.

Somewhere between 1965 and 1975 the various "supercomputers" attached to the Octopus network came to be known as "worker" computers. Each of the major workers (namely CDC 6600, CDC 7600, STAR-100) was supported by a locally designed and implemented timesharing operating system. Each of these had its own "name": CDC 6600 - GOB, FROST, CDC 7600 - FLOE, STAR-100 - BOSS, STAR-OS. During this period various writings describing aspects of the total worker system (hardware, operating system, programming system, Octopus network, etc.) were gathered into a common document series entitled LTSS (Livermore TimeSharing Systems). This document title came to be a colloquial generic term for all worker operating systems.

During his term as Head of Computation Department, Bob Lee attempted to formalize the colloquialism by defining each operating system by its generic/specific function, e.g., LTSS/7600, LTSS/Cray.

As the network expanded and newer hardware became available, the difficulty of adding specific logic for each new component of Octopus increased. This added to the complexity and cost of each system, and to the difficulty for components of the programming system to access new devices. Network Systems Division recognized this situation and began consideration of a new genre of operating systems for future equipment. The generic term applied to the definition of such a new worker operating system was NLTSS, where the N represented the word New. Various groups researched the problem, using differing approaches. As new high-speed communications hardware appeared on the horizon, the concept of a network operating system became increasingly feasible and practical. So, the definition of the N was changed to Network.

In recent years, a great deal of research, design, and implementation has gone into the production of a modern network operating system environment for the LLLCC. The fundamental view of such an environment is provided by LINCS (Livermore Interactive Network Communication System). LINCS defines a set of layered protocols that provide the framework for implementing NLTSS, OCTOPORT, a new central storage facility, and future components of Octopus. NLTSS, then, is an implementation of a local operating system for a major worker computer (current target is a CRAY-1S) functioning in the LINCS environment. NLTSS includes a hardware-specific kernel, a message switch that is at the core of the system design, and several server processes that provide the functionality (user view) of the system. All processes communicate by messages via the message switch. Therefore, the manner of interaction of two processes is logically independent of whether they are on the same or different computers. In fact, most modules are being written in a high-level language for portability purposes. New functionality (services) can be made available by simply providing a new server process without necessarily altering or disturbing the existing menu.

The result is a distributable, extendible network operating system. In this environment, network resources become much more readily and effectively available to user processes, and new resources can be added much more easily than in the current Octopus/LTSS environment.

Note that NLTSS is not being built in a vacuum. All of Network Systems Division participates in design discussions so that all areas of responsibility are represented. Several members of the User Systems Division have recently been included in order to begin a new BASELIB compatible with NLTSS primitives. At times, some aspects of NLTSS are presented to interested groups to get their reaction, input, feedback, etc. This information provides useful guidelines for subsequent design discussion.

The actual NLTSS logic is being exercised in an encapsulator running on the CRAY-1 machines. This is a valuable tool for development of server and user processes.

Pierre (Pete) Du Bois, OSG/NSD
Ext. 2-4007

COORDINATION CENTER NOTES

Some readers may be interested in brief descriptions of the computer network, and what might be wrong with it. Such is the nature of the contribution of the Coordination Center this month.

Cheerful and Efficient Self Diagnosis: Terminals

The message network at LLNL consists of about 1400 terminals attached to locally designed concentrators based on DEC minicomputers. The minicomputers are connected to each other and the PDP-10. Some have connections to the big worker computers. The message network is used to connect terminals to worker computers and for communications by ENTERPRISE to schedule data flow between workers and storage.

Terminals are attached to the network by means of dedicated wires, four per terminal. There are cross connect panels in the computer center and at intermediate points around the laboratory, enabling relatively easy configuration of individual terminals, but requiring careful documentation by the network coordinator's office. Viewed from a distance, the terminal connections and the intercomputer connections look like spilled spaghetti.

"So what has all that to do with my problems?," one might ask. "And why do people ask all those dumb questions when I complain?" The answers to those questions may be found in the following.

In the simplest case, there are six active devices between your fingers and the job you are running on the worker. Your terminal, the concentrator multiplexor, the concentrator computer, the concentrator OCTOBUS adapter, the worker OCTOBUS adapter, and the worker system all have electronic circuits, and some have programs that are acting on the characters you type or the packaged message after you type RETURN. All are possible sources of malfunction. Even the job you are running may contribute. There are terminals on more remote concentrators that add one or more minicomputers and associated line units or adapters to the chain.

There are four groups responsible for maintenance of various parts of the message chain. All four groups are kept busy. It is desirable to guess correctly which group is most likely to repair a specific problem.

The various intelligent active components of the chain offer some clues to the knowledgeable observer. Here are a few clues. (More detailed information will be forth coming in a later issue.)

1. Your terminal echoes strange characters, loses the first two characters on a continuation line, or refuses to echo anything.

   a. If your terminal number is less than 1000: The concentrator is down or your terminal is inoperative. Look at the network display or message

display to check the condition of the concentrator. If it is up, call TTY service.

b. If your terminal is greater than 777: (1) it may be misdefined, type CTRL-D a few times and then the appropriate number:

0 - if you have a minicomputer.
1 - if you have a TTY33
2 - if you have a TI Silent-700
3 - if you have a Data Media Elite
4 - if you have an ADM1
5 - if you have an HP

Note that if your terminal is defined as a video terminal, it may be necessary to type CTRL-C to clear the screen.

2. The terminal seems to be o.k., but echos @BYE whatever key you press.

a. If your terminal number is less than 1000: Your terminal is bad, call TTY repair.

b. If your terminal number is greater than 777: Chances are that your alpha or shift-lock key or uppercase key is on; check and correct the setting.

Don Emery, COD
Ext. 2-4005

CHORS II MEETINGS & USER-REQUIREMENTS DOCUMENT

SCHEDULE OF CHORS II MEETINGS

CHORS-II-Status-Reporting and User-Critique Meetings
First Tuesday of each month, beginning 6 October
Bldg. 113, Rm. 1206
1:30 to 2:30 p.m. (Anyone may attend.)

CHORS II Working Meetings for Staff and Project Leaders
Every Tuesday, beginning 8 September
Bldg. 113, Rm. 1206
1:30 to 2:30 p.m. (Limited to the small working group.)

WHAT IS THE USER-REQUIREMENTS DOCUMENT?

In a memorandum dated 30 July 1981, I gave the following brief description of the user-requirements document: *The document does not specify equipment, utility routines, design, or coding practice. It simply defines CHORS II from the point of view of its users.* (*User is defined in the broad sense: anyone who uses the CHORS II system.*) Now I'll give a more complete definition. Many of the items below are the ideas of Tom DeMarco, but they have been refined to fit the LLNL environment.

A user-requirements document is the most important result of the initial phases of project development; it is important because it describes what is to be implemented. The name user-requirements document has many variants, depending on who you talk to, which book you read, or which training seminar you attend. Some other names used are: external design document, design document, user-specification document, or requirements document. For CHORS II, I'll stick with the term user-requirements document. This document will be the result of an analysis of the question: What does CHORS II look like to the user? This document will establish the goals for the rest of the project. It will describe what the project must deliver in order to be considered a success.

The successful completion of this document involves:

- Selecting an optimal CHORS II description.
- Documenting in sufficient detail so that subsequent implementation can be evaluated to see whether it meets the description.
- Producing accurate predictions of important parameters, including benefits and performance characteristics.

It should also be noted that user requirements are constantly changing. Thus, the user-requirements document should be regarded as an approximation to real user requirements. Therefore, the requirements document itself should be structured so that it can easily change as it is being developed.

To make the requirements document more readable, it will be divided into several minidocuments. Or, more appropriately, CHORS II will be divided into separate subprojects, and these will be examined separately.

When the meetings get under way, expect many changes to and compromises of both CHORS II features and the procedure for conducting the user-requirements-document project. Also expect new committees to form, committee purpose and

goals to change, committee membership to change, etc. Such is the nature of making a user-requirements document: constant change.

My role in getting the user-requirements document written will be that of mediator. I will try to suggest proposals of compromise. I will take the information you have prepared or suggested and try to work it into the requirements document.

Being inclined towards graphics, I plan to describe the CHORS II system in pictures as it develops through the series of meetings. The format for the pictures will be like a tree or network. The nodes will represent some action or process. Data such as files or commands will be the objects that get passed among the nodes.

This should give us a document whose meaning is clear and whose content can be easily modified.

Pete Keller, CGG/USD
Ext. 2-4300

DYNAMIC MEMORY MANAGEMENT

Currently, there are several methods whereby a user can dynamically expand and contract the memory being utilized by a controllee. He may accomplish this by using FORTLIB (subroutine MEMORY), BASELIB (subroutine IZSCMLCM), or by invoking direct system calls. In addition, there are several memory management packages (e.g. MMLIB) available that provide relatively sophisticated capabilities in this area. The afore-mentioned capabilities have provided the user with a great deal of flexibility in managing the use of memory.

On the negative side, most of the currently used memory management methods do not allow for more than one entity in a controllee to be performing memory management. The only exception to this is the controllee that exclusively uses the FORTLIB MEMORY routine to manage memory. In this case, FORTLIB is able to do some limited memory management in order to allocate I/O buffers. This has not been a significant problem thus far, but some imminent developments require us to alter our memory management methods.

In the very near future, the CFT compiler will be utilizing a stack or heap at run time. The CIVIC compiler could also profit from this technique. These compilers could utilize a stack or heap to store vector temporaries and possibly other data (i.e. argument values, local variables, register saves, etc.). The CRAY loader (LDR) could implement a new feature that would significantly benefit some of the larger codes if it could perform dynamic memory management. None of this can take place, however, until there is a common memory management interface.

If more than one entity within a controllee attempted to manipulate memory without informing each other, unpredictable and often catastrophic results would follow. The obvious solution to this problem is the implementation of a common memory management interface. By this I mean a set of modules that would manage memory in such a way that more than one entity within a controllee could obtain and return segments of memory without interfering with each other.

With this in mind, I propose a set of three routines to constitute the memory management interface. There would be routines to allocate memory, deallocate memory, and handle errors. These memory management "primitives" could be called directly by the users, or by more sophisticated memory management packages. They would manage a heap based on a "first-fit" algorithm. The "first-fit" method is certainly superior in speed and seems to produce a better characteristic size distribution of free blocks. The blocks will never be moved. The heap would begin at the FLL+1 for single level codes, and at LWA+1 of the longest chain for segmented codes. The compilers, loader, and FORTLIB would also call these routines to accomodate their memory requirements.

There are some existing codes that could not adapt to this scheme. I propose, therefore, that there be two additional options available to the user. In the first, the memory management primitives would manage only that part of memory that the user does not want to manage . In the second, the user would provide his own memory management primitives.
1. In this case, the user would inform the memory management primitives of the

area of memory that they are to manage. The user would then be free to manage the remainder of memory as he deems necessary. The compilers, loader, and FORTLIB would call the memory management primitives in the normal manner. If the compiler requested a block of memory from the allocate routine that could not be accommodated, the allocate routine would call the error handling routine. If the standard error handling routine was loaded, it would issue a diagnostic and terminate. If the user provided his own error handling routine, it might perform some clean-up operations, or perhaps move its own block of managed memory, increase the size of the block that the primitives are managing, and return to the allocate routine. The user would inform the primitives of the area of memory to be managed via a vehicle similar to the IQ8QDSP method currently used. Both the address and length of the area would be provided. The calling sequence of a user provided error handling routine would have to be compatible with the standard routine.

2. In this case, the user would provide his own set of memory management primitives. They would manage memory in any way the user desires. They would, however, have to have calling sequences that are compatible with the standard set of routines. Furthermore, they would be required to function indentically (from an external point of view) to the standard set. By that, I mean that when the compiler calls the non-standard version of the allocate and deallocate routines, the effect (as far as the compiler is concerned) must be the same as if the standard routines were called.

Following is a brief specification of the allocate, deallocate, and error processing routines. The names would be changed to be consistent with the naming convention of the library they are placed in. One or more of these routines might also be callable as functions. This would be the case if it was determined to be more efficient.

```
CALL ALLOC (ADDR,LENGTH)
      where:  ADDR is returned as the FWA of the allocated block.
              LENGTH is the requested block size.
CALL DEALLOC (ADDR)
      where:  ADDR is the FWA of the block to deallocate.
CALL ERROR (FLAG,DELTA)
      where:  FLAG = 1, bad address in deallocate call.
                     2, negative or zero length in allocate call.
                     3, insufficient memory to satisfy allocate call.
              DELTA is the amount of memory needed so that the allocate call
                    could be satisfied (FLAG=3).
```

Note that the ERROR routine would normally be called only from ALLOC and DEALLOC, not directly by the user.

It is hoped that this proposal addresses the various requirements of the user community, as well as the compilers, loaders, FORTLIB, and any currently existing memory management packages. I am currently having discussions with CRI regarding the compatibility of this proposal with the needs of the CFT compiler. If you have any questions, suggestions, or problems with this proposal, please do not hesitate to write or call.

Rick Johnson, LG/USD
Ext. 2-4496

Dependent Libraries

Last January the 7600 version of BUILD was released, and in June a new version of LOAD was released to public on all CDC 7600's. These two utilities give approximately the same dependent library capability for 7600 users that LDR and the Cray version of BUILD do on the CRAY-1's. This article will explain dependent libraries and some of the details of their implementation and use on both machines.

Definitions

In general, a *library* is a file that contains 1) copies of other files and 2) a *directory* with the names of the included files, their starting address in the library, and their lengths. This article will only discuss libraries for which the included files are binary modules produced by a compiler or an assembler--in particular, libraries generated and maintained by the utility routine BUILD.

In a general sense, such a library is said to have *dependencies* if its modules have calls to routines that are not in the library--loading with such a library could result with an "unsatisfied externals" message.

For example, the LINK subroutine in FORTLIB uses the scan routines from BASELIB (ZSCANLEQ, ZSCANRNE, and so on). However, these BASELIB routines are not included in FORTLIB.

In fact, several system libraries use routines in BASELIB-- FORTLIB, URLIB, and GRAFCORE are prominent examples. These libraries depend on BASELIB to achieve their full function. And, for historical reasons, BASELIB is said to be a *dependent library* of these system libraries.

There is, however, a more precise sense to the term "dependent library". The directory that BUILD puts in a library is divided into two parts--the library header table and the library directory table. Space is proved in the header table for a list of dependent libraries, and anything in this list is by definition, a dependent library. Keeping a list of dependencies in the BUILD directory makes the loading process more convenient by allowing the loaders to pick them up automatically.

Loading with Dependent Libraries

The loaders LOAD and LDR use the dependent library list in the library header table. If there are unsatisfied externals from the libraries and binary files listed on the execute line, the loaders will open the libraries on the dependent library list and try to get all the modules it needs from them. If these dependent libraries in turn have dependent libraries, they will also be opened if necessary. This process could go on through several levels until either all externals are satisfied, or the loader runs out of libraries to examine.

This process is straightforward for single level loads that don't involve too many libraries. In more complicated situations, however, it may be difficult to predict results. For example, suppose there are four libraries—L1, L2, L3, and L4. L1 contains a module named A, and has L2 as a dependent library. L2 contains a module named B, and has no dependent libraries. L3 contains a module named C, and has L4 as a dependent library. L4 also contains a module named B, and has no dependent libraries. (For example, B might be two versions of a square root routine, and the version in L4 is ten times as fast as the version in L2.) Suppose the load line was

```
LOAD Q46 BIN1$ L1, L3, XBIN1+ 1* MBIN1
```

If BIN1 had a call to C, and C had a call to B, would B come from L2 or L4?

For single level loads LDR and LOAD use approximately the same algorithm:

The input line (or file) is scanned, and any explicitly named libraries are put on an internal library list. Then the input binaries are opened in the order they appear on the input line. If any binary is in fact already a BUILD library with dependent libraries specified, then these dependent libraries are placed at the end of the library list, unless the library name already appears on the list. After the input binaries are processed the loader starts on the list of libraries in an attempt to satisfy remaining unsatisfied externals. The libraries are considered in the order they appear on the list, and as each library is opened its dependent libraries are added to the end of the list (unless of course the dependent library is already in the list). The loader continues until all externals are satisfied or it runs out of libraries. So, following this algorithm, module B will be loaded from library L2.

LOAD and LDR process overlaid codes in entirely different ways, and this is reflected in the way they handle dependent libraries. LOAD loads the entire root code block just as described above. After it is finished with the root block it clears the library list, and forgets all entry point names that were not actually used. In order to have routines available to lower level code blocks the libraries must be re-specified--each code block is processed in isolation from the others. Only modules on the direct path from the root code block to the lower level code block will be available to the lower level block. (These modules are available because they will be in core with the lower level block at run time.)

Because LDR is a segmenting loader, it does things differently. Instead of processing each segment separately, the structure of the entire overlay is built at one time. LDR automatically allocates subroutines to segments in an optimal manner—the user has only to specify a binary file or module to be force loaded as the starting point for each segment. Since LDR does not know in advance what modules will go into a given segment, it must keep the same library list during the entire loading process.

At the beginning of the load LDR places all explicitly named libraries on the library list. Then all force-loaded binaries and modules will be loaded, following the order of the SEGMENT directives that the user has specified. Dependent libraries from these binaries are added to the end of the library list

as they are encountered (unless, of course, they already appear in the list.) Following the force-loaded modules any other binaries specified are loaded wherever LDR determines they should go, and their dependents are added to the list. From this point on LDR proceeds as in the single level case, opening libraries and adding dependent libraries to the end of the library list until it either runs out of libraries or unsatisfied externals.

In practice, loads complex enough to cause problems should be rare. However, users with their own versions of standard routines should be careful in using them as dependent libraries—it is entirely possible to get the standard routine loaded instead. Of course, consult the load map if there is any doubt about the origin of a routine.

The structure of BUILD libraries is documented in LCSD-1512, the BUILD write-up. How LDR handles dependent libraries is documented in LCSD-344, the LDR write-up. There is no documentation at present on how LOAD deals with dependent libraries, but LTSS-109, the LOD document, covers multi-level loads, and is generally applicable to LOAD.

Kent Crispin, SIG/USD
Ext. 2-4273

ENTERPRISE DIRECTORIES

The scavenge of expired MASS cartridge directory entries took place the last week of July. We deleted 225,680 expired entries. The scavenge left 10,020 empty directories, which we also deleted after rechecking each directory to ensure that it was still empty. In the near future we will be deleting any other empty directories that have not been altered within the last year.

WARNING: This will affect those people who are using empty directories as identification. See suggestions below.

Our ENTERPRISE directory disks are rapidly filling up. There are several practices we encourage for more efficient use of our resources.

1. Deleting unneeded directories and file entries.
2. Using the CA command in XPORT for identification purposes rather than empty directories.
3. Storing temporary files with the PROP. C option in XPORT to avoid the default of creating another permanent entry.
4. Learn to use duplicate entries to share files rather than keeping individual copies. See the DUP command in XPORT. Try to avoid using the COPY command. (In most cases the DUP command will more than suffice.)

There are tools available for helping the user to trim down directory structures.

1. The XPORT L command will make a list of your directories. For example:

```
XPORT NOTTY!DISK FILENAME!L. LEV. 8!END
```

   will list your directories to 8 levels.
2. DLIST under NEW will take the output from 1. and reformat it.
3. The PURGE command in XPORT will delete expired cartridge entries indicated by FE.
4. A new utility, DMAN, will be available later this year to make cleanup much easier. It will provide an interactive means of altering directories by using the TMDS. The author is Kent Crispin.

Everyone's cooperation is needed to make efficient use of our resources.

JoAnn Watson, SG/NSD
Ext. 2-4313

CONSULTING OFFICE COMMENTARY

TREATMENT OF LABELS BY CIVIC AND CFT

The CIVIC and CFT compilers handle source code labels differently. The Consulting Office receives questions like: "I put a label in subroutine xxx, but it's not there; how come?," or "How do I set a breakpoint (using DDT) at the end of a loop?"

CIVIC's treatment of labels is dependent upon the level of optimization being used. If no optimization level is specified, the default level of G is selected (See LCSD-302 for details). One of the side-effects of G optimization is the removal of unreferenced labels. User-supplied DO-loop labels are also removed (if they are not jumped to). In other cases, CIVIC introduces labels of its own (irrespective of the level of optimization) in handling DO-loops and other language constructs. For example, an internally generated label is inserted before the first executable statement within a DO-loop. These labels can be seen in the compiler listing of the source file. The second column of numbers from the left contains line numbers with an "L" appended. These are L-labels that may be referenced with DDT. To see all the labels available for referencing with DDT, use the LIST LABELS command in DDT (see the following section). The compiler listing is useful in complex cases because the positioning of the compiler-generated labels with respect to the source code is obvious. If you do not want your labels removed, drop the level of optimization below G (e.g., use O=F; however, object code will not be scheduled), or use the OPTION LABEL feature to preserve labels you specify. Of course, to make either of these changes, you must recompile and load your code.

CFT's handling of labels is different. As with CIVIC, user-supplied, unreferenced labels are removed; however, there is no mechanism to force them back into the code. Also, like CIVIC, CFT does not generate symbol-table labels for user-defined loop labels. Rather, for each loop, CFT generates two labels; one at the beginning of the loop and another immediately after the loop. As an example, for these statements

```
DO 9999 II = 1,100
   XXX(II) = Z1
9999 CONTINUE
```

CFT would insert label 9999A before the code generated for the assignment statement within the loop, and label 9999B after the code generated for the CONTINUE statement, just outside the loop. In the case of the IF/THEN/ELSE construct, CFT generates labels of the form *nnnnn*, where *nnnnn* = 00001,00002,00003,..., as needed. The compiler listing generated by CFT with the ON=G option is helpful in defining the use of labels. (See CFT document, LCSD-304, for further details.) You can also use DDT's LIST LABELS, as with CIVIC codes.

DDT and Labels

When using DDT to run a controllee, breakpoints can be set at absolute addresses (i.e., BKP *nnnnn*B). Typically, a programmer debugging a code would rather reference the source code's symbolic labels, instead of absolute addresses. The labels placed in the symbol table, which DDT uses, are determined by the compiler/assembler being used (see previous section).

Use the LIST LABELS command to list these labels (relative to a code block or subroutine, first type SUB=*cb*). If a label doesn't show up in the list generated, either the wrong code block is being specified, or the compiler didn't put the label in the symbol table. To see what the compiler did or didn't do, check the compiler listing (preferably one with the object code listed). In a large routine with many labels, you can view the output of the LIST LABELS command on the TMDS. Do this with DDT's OUTPUT command (e.g., OUTPUT=TV123;SUB=ABC;LIST LABELS). The LIST command will list other DDT tables, i.e., variables (VARS), common blocks (COMMONS), subroutines (SUBS), etc. See the DDT writeup for details (LCSD-1620). One last point on labels that begin with a numeric character: when referring to these with DDT, either precede the label (as given by DDT in the LIST output) with a $, or enclose the entire label within single quotes (e.g., $100, $99L, '50A').

Date of a File

Users often want to know when a file was written to a storage device (MASS or ATL) or when a file was created, last written, etc. Let's assume you have a file in central storage. If the file was written to MASS (e.g., using the P. C option of XPORT), XPORT's STATUS command will tell you when the file was written. If the file was written to ATL (using the P. L or the default P. B option), there is no inquiry command within XPORT to tell you when the file was written to ATL.

Once the file is staged from storage to disk, there are other things you might do to determine when the file was written. If the file was ever written or modified by TRIX AC or TRIXGL, the machine, date, time, and user number are written into the file in the four words following the EOF. The TRIX AC and TRIXGL "WHEN" command will report this information to you. Compilers, loaders, and library file maintenance routines (e.g., BUILD, LIB, etc.) generally record the date, time, and other information in each file they operate on. CHAT, CIVIC, and CFT will record the date and time of compilation in each binary module; use DDT in sequential mode to search the file for a date and time (in BCD). An executable file produced by a loader will contain the date and time of the load in GOBCOM on a CDC 7600 (words 36B and 37B) and in low-core (GOBCOM equivalent) on a CRAY-1 (words 110B and 111B). Library files have headers and/or directories which contain date and time information and can be querried for this data using list commands of the particular routine used.

Code output files (either binary or BCD) that have not been modified by a text editor routine will not have date/time data stored in the file unless the user has explicitly placed it there.

Ted Stullich, SIG/USD
Ext. 2-4703

GIFFGAFF

Material for this issue was received from approximately forty members of NDD, NSSD, APDI, Guylene Hegarty, and Pat Medsker.

Congratulations to the Consulting Office for its new bug reporting system. It looks like a giant step in the right direction.

Responses Received

Many thanks to Guylene Hegarty and Pat Medsker for responding to our requests for a guide to the TMDS display. Guylene sent me a 1979 *SHOWTIME* writeup which is available from the Coordination Center. She reported that a new version will be available this fall. This was verified by a call from Pat Medsker. She indicated that a new *SHOWTIME* had been planned to be available in time to meet the needs of the summer employees. This was held up due to significant recent changes in the displays. The report should be available in about two and a half months.

Pat suggested that users call Joanne Perra to submit any computer related documentation for publication.

Guylene was also the source of a document entitled, *A Tour of the Computer Facility*. This was produced for the 1979 open house and, while a bit dated, contains some interesting information about our computer facility. This is available from the Coordination Center. An updated version is at the printers and will be available this fall.

Observations

Opinions seemed to vary regarding CRAY-1 performance. We had everything from "things are going smoothly" to "its been the pits lately."

- Some of our users reported inadequacies in CRAY-1 graphics. One user in particular wondered why the CRAY-1 graphics are not equivalent to those on the CDC 7600's.
- The RJETs are extremely important pieces of equipment. We would like to ask for a better maintenance schedule.
- The TMDS was less than reliable. We saw many transmission difficulties.
- Several users had difficulty getting the whole TMDS screen. The requested data was displayed on the upper part of the screen, then out of the blue the user's ID appeared, followed by a blank screen. This phenomenon was noted when using several codes, e.g., DISPLAY, TRIXGL, LASLIB, and FRTV and occurred on both the CRAY-1's and the CDC 7600's.
- A plea for messages was received. When a machine comes up the users appreciate a prompt message from the operators.
- It was also noted that messages frequently get lost in the system. These could be from XPORT or any inter-machine communications.
- Waits of three to four hours were reported by several ATL users.
- One user wondered if there are plans for another ATL or if the present system will be expanded.

I/O

- Our I/O watchers noted that although the HSP situation seems to be quite good right now; it is still taking two days to get fiche.
- Users are bogging down the disks with files that cannot be destroyed until output is received.
- Misfiled fiche was a source of annoyance to several users.

Libraries

A couple of items in the BASELIB documentation would seem to be in need of a bit of clarification.

- The introduction states that all arguments must be in SCM with the exception of data arrays and the lengths for the I/O functions (IZDKIN, IZDKOUT, IZTAPIN, IZTAPOUT). However, ZMOVEWRD states that both AOUT and AIN may be in SCM or LCM.
- ZMOVEBIT section defines "AIN: An input array containing a string of bits." A concerned user asked "What does this mean? Can the input array be declared a bit or a byte? Experience would seem to indicate this, in fact, cannot be a byte."

- A user questioned why GRAFLIB doesn't end the frame when a GRAFLIB family is ended. This user was troubled by receiving a half a frame at the beginning of a fiche and a half at the end. This problem can be remedied by correctly estimating the number of frames to be printed on a given fiche. Our user wondered if there were a simple solution to this problem for the casual user of GRAFLIB and GRAFCORE.

Utilities

- Dissatisfaction with DISPLAY was expressed by a user who felt that the commands were awkward, making it difficult for those not using the code on a daily basis.
- TRIXGL was mentioned by several people this month. It is an oft used tool that would be even more valuable if the following options were added.
- TRIXGL came under fire from a user who reported that in spite of an ALL DONE message, changes were not being made in her file. This problem was experienced by several people recently.
- It would be useful to have an instruction that would type the line that is *n* lines after (or before) the occurrence of a specified pattern. This would be useful in scanning output where there is a header and then the values are printed in a line, some number of lines under the header. Some MEG output is an example of this sort of text. You could ask for the line two lines past each occurrence of "KE" for instance, and receive all the values of KE in the output without reading extraneous data.
- It would be beneficial to be able to search for a string of patterns, such that the user would receive a list composed of patt1 or patt2, etc.

  e.g., TP!patt1!patt2! .....

  Additionally beneficial would be the ability to locate the closest pattern before (or after) a given line number (*n*).

  TPB*n*!patt or TPA*n*!patt

  Also useful would be a command to move text on lines *m* through *n* by *j* columns to the right or the left. This would be particularly useful if TRIXGL could

recognize special cases and thus treat continuations and comment cards appropriately.

Finally it would be advantageous to be able to define a special character to represent blank (b), in order to specify trailing blanks in a "newpatt"; this could also replace, more or less, the column-changing commands previously suggested, e.g., blank = $

RP*n*,*m*!$$$$!$$$$$$$ for 3 shifts to the right.

Requests

- A "B" Division designer would like to request that when utility routines are moved from one machine to another, please do not change the commands. This is a pet peeve which he has in common with many other users.
- We would like to request that no soft copy terminals be provided for operators. We often need hard copy to identify bugs in either machines or codes. It would be helpful if all TTY listings from the operators' terminals could be saved in a log book.
- It would be very valuable for late night and weekend workers to have a complete library of computer documentation in Trailer 2106. Manuals are apt to be outdated, thus information in a central library is important for those working when the Consulting Office is closed.
- To repeat an oldie but goodie—when public files are lost, if the owners of said files can be identified, please notify them.

Wouldn't it be great if:

- Output came in book-like form?
- We could eliminate the glue on printouts?
- You could return to a suffix and receive a complete message. ALL DONE is inadequate. What about stashing messages in a news file for the user?
- There were a wide distribution for any material relating to CHORS II meetings, plans, etc.

CHORS II Collecting Agency

Mike Pratt is collecting data regarding user requirements for CHORS II. IF you would like to make your wishes known, please feel free to contact Mike Pratt at Ext. 2-4699 or Dorothy Freeman at Ext. 2-4702.

I will gladly accept input for this column from anyone who cares to call or drop me a line. I am always more than happy to discuss any issue addressed with anyone who can shed some light on the subject.

Dorothy Freeman, NSSD/NDD
Ext. 2-4702

Utility Routine CHECK

Octopus Communiqué 1569

References: 1. B. Eckert, *A Proposal for Utility Routine CHECK (UR-333)*, Octopus Communique 1486, dated 16 January 1980.
2. *Summary Sheets*, LCSD-50, Rev. 0, dated 1 May 1981.

On Tuesday, 22 September, public file CHECK will be replaced on all CDC 7600's by the version of CHECK that is presently available in library file NEW.

Users who do not have time to adjust to the options in the new version of CHECK, find them impractical to use, or have no wish to learn them, will still be able to retrieve the old version; it will be placed in library file USE on all CDC 7600's. (N.B.—This action is the one proposed in Ref. 1.)

The new version of CHECK uses the universal 29-bit decimal checksum—as opposed to the 30-bit octal checksum; this new checksum is the same as the one used by CSUM. Also, when a file is transported from a CDC 7600 to a CRAY-1 and back, its checksum will be the same.

A partial list of the differing options follows. All of them may be seen in Ref. 2.

CKF. Create file %CKSUMFL and calculate checksums of all files specified. Put names, lengths, and checksums into this file.

CKC. Calculate checksums of files specified and compare them with checksums in %CKSUMFL.

A. Add checksums of files specified to %CKSUMFL.

D. Delete checksums of files specified from %CKSUMFL.

R. Replace checksums (in %CKSUMFL) of files specified with new checksums for those same files. If a file is specified that is not in %CKSUMFL, it will be added.

LIST. Print contents of %CKSUMFL.

Karl Dusenbury, SIG/USD
Ext. 2-4311

APPROXIMATING A FUNCTION FOR COMPUTATION

Octopus Communiqué 1571

Large physics codes frequently use an approximate formula for the integral

$$y = \int_0^x t\,(1 + t^3)^{-1}\,dt$$

to calculate energy deposition. A new formula is both faster on the CRAY-1 computers and more than 300 times more accurate. It is available in the form of a subroutine in my take directory .677150:E1:113. The following table shows how well several approximations of $y(x)$ work.

Table 1. Speed versus accuracy for

$$y = \int_0^x t\,(1 + t^3)^{-1}\,dt$$

Approximation	Time for 10^4 evaluations (milliseconds)	Maximum Relative error * 10^4
(1) Exact formula	33.7	0
(2) Present method	11.8	476
(3) Vectorized version of (2)	3.2	476
(4) Improvement of (3)	3.1	246
(5) Improvement of (4)	3.6	18
(6) NEW METHOD	2.4	1.4

Note: The errors in this table were computed at 10,000 values of x ranging from 0.01 to 100. In all cases the errors are smaller for very large or very small values of x than for moderate values.

The exact formula for this integral (row 1 of the table),

$$y(x) = -1/6 \log\left[(1 + x)^2\,(1 - x + x^2)^{-1}\right] + \tan^{-1}\left[(2x - 1) / \sqrt{3}\right] / \sqrt{3} + \pi / (6\sqrt{3}) \qquad (1)$$

takes too much time to compute on a CRAY-1. This function is asymptotically equal to a polynomial in x for small x and a polynomial in $1/x$ for large x. Therefore, the usual approximation takes $y(x)$ to be a polynomial in x for $x < 1$

and a polynomial in $1/x$ for $x > 1$. The coefficients are chosen to correctly give the first two terms of the asymptotic expansion of $y(x)$ at $x = 0$ and $x = \infty$ and to give the correct value of $y(x)$ at $x = 1$; the result is

$$y(x) \simeq 0.5x^2 - 0.125x^4 \text{ for } x < 1$$
$$y(x) \simeq 1.2 - x^{-1} + 0.175x^{-2} \text{ for } x > 1 \qquad (2)$$

A heavily used code implements this in a DO-loop that will not vectorize (row 2), but another coding of this approximation is much faster (row 3).

But there is no point to speeding up such an inaccurate method. Better coefficients will make (2) more accurate. With subroutine FITPOL from MATHLIB we can compute coefficients that will minimize the error or relative error at a number of fixed points; row 4 of the table shows that these coefficients will cut the error in half.

Finding coefficients this way gives us the flexibility to use any number of terms in the polynomial approximations, trading speed for accuracy. Row 5 of the table shows that five-term polynomials are dramatically more accurate and cost little more than three-term polynomial approximations to this $y(x)$.

Rather than trade speed for accuracy, we can get both at once through a different type of approximation. It is expensive to decide whether to use a polynomial in x or $1/x$, i.e., whether $x < 1$ or $x > 1$. It would be better to approximate this integral $y(x)$ by a <u>single</u> function that still has the correct asymptotic values near both $x = 0$ and $x = \infty$. A rational function fills the bill. An optimization routine in the NAG library helped choose coefficients for

$$p(x) / q(x) \text{ , where } p(x) = x^2 + e_3x^3 + e_4x^4 + e_5x^5 \; q(x) = 2 + d_1x + d_2x^2 + d_3x^3 + d_4x^4 + d_5x^5 \text{ and } e_5/d_5 = 1.2092\ldots,$$

in order to minimize the relative error at a few fixed values of x. The last row of the table shows that this approximation is faster and far more accurate than the alternatives.

Jeff Painter PAS/MSD
Ext. 2-0675

COMPUTIST'S CORNER

by R. K. Cralle

Lore of the Exclusive OR

Many interesting computations can be done by using the logical function, exclusive OR (⊕). This month I shall explore a few of them. A nice thing about the ⊕ is its nondestructive nature, unlike the operators AND and OR (inclusive) which have messy inverses (one needs to remember both operands). It isn't so nice that the ⊕ isn't a complete logical operator. AND and OR aren't either, for that matter. By complete we mean all of logic can be done with a set of logical operators. Such as, NOT and AND, or, NOT and OR.

Sheffer was a logician who asked himself, is there any *single* logical operation with which all of logic can be done? The answer was yes, and the operation was called Sheffer *stroke* (|). (Actually, C. S. Pierce, the great American logician discovered it first.) There are in fact two logically complete operators, the other one being called *dagger* (↓). I suppose *stroke* and *dagger* sounded too clandestine for engineers — they call them NAND and NOR.

Thus, the usual logic operators in terms of only *stroke* (NAND) or *dagger* (NOR):

not P = P|P	*not* P = P↓P
P *and* Q = (P|Q)|(P|Q)	P *and* Q = (P↓P)↓(Q↓Q)
P *or* Q = (P|P)|(Q|Q)	P *or* Q = (P↓Q)↓(P↓Q)

In the early days of digital electronics it was important to have only one logical operator (*gate* to engineers) because it would have been more complicated and expensive to implement all logical operators in hardware. And so the engineers borrowed from the logicians. The ⊕, we will now see, has its own pretty properties. The particular problems I have chosen will be given first and then the techniques, in case anyone wants to think of their own techniques first.

Problems

1. Franz Fizzist comes to us and says he has a calculation that computes three numbers. He guarantees (hmmm) that two of the three numbers are equal to each other, and the third one is not necessarily equal to the other two. Give a fast (best?) technique for picking the third number.

2. Your Micro Peenie I computer has a certain number of registers. They all contain information you don't wish to store in memory now. Thus, being out of registers, you nevertheless wish to exchange the contents of two registers. You want to do this without saving and restoring a register. How? (Your MP-I doesn't have an exchange instruction!)

3. In certain signal processing transformations such as Walsh and others, one needs the integers, 1 to *n*. Unfortunately, one needs each successive integer

with its bits reversed. (Reverse Farsi?) Thus, the calculation requires in successive order (where n, 7 in this example, is less than four bits long),

4,2,6,1,5,3,7 (100.010.110.001.101.011.111 in reverse binary)

The problem is how to make a computer add "left handedly"? That is, when adding a one to the *most* significant bit of a number, cause the carry to ripple to the *right*, not left.

4. Calculate the Gray code of an integer simply. Gray code numbers are binary representations of integers, where going from N to N + 1 entails changing only one bit.

Decimal	Binary	Gray
N=0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Techniques

1. Ignoring computers with a schizophrenic zero, the following gives the desired number: $N = n_1 \oplus n_2 \oplus n_3$

2. The following code exchanges two registers without affecting any other registers.

$$r_1 = r_1 \oplus r_2$$
$$r_2 = r_1 \oplus r_2$$
$$r_1 = r_1 \oplus r_2$$

It follows from this technique one can hold two full word numbers or pointers (P_1, P_2) in a single word. Then with either number or pointer we can retrieve the other.

Set, $W = P_1 \oplus P_2$.
Then, $P_1 = W \oplus P_2$ and $P_2 = W \oplus P_1$

3. Count the number of lead bits in the current generated number n. Add one and $\oplus$ this number of one bits with the original lead bits, like so:

```
1000...0 (where the length is big enough to hold n)
11
0100...0
1
1100...0
111
0010...0 etc.
```

On a CDC 7600 the normalize command (contrary to intuitive belief) counts the number of left shifts until, the most significant bit of the fraction differs from the sign bit. This fact makes the calculation especially fast. A sign extended right shift creates the required string of one bit quite nicely.

For people who like pure TRIX one liners, the following computes a table like the one above (of arbitrary bit length):

```
#DO(<#SM(n,0)#NS(X,#GC(Z,#GC(%SM))<1>#GC(X,99))#DS($X,)>,1,L)
```

where, n has been set to a string of k zeros, $L = 2^k-1$ and Z is set to a sufficiently long string of zeros. (Say, n = <000>↑ and L = 7↑Z=<000000000>↑.)

4. The gray number G, calculated from the integer N:

$$G = N \oplus (N/2)$$

R. K. Cralle, CRG/COMP
Ext. 2-4041

NEW AND NEWLY REVISED DOCUMENTS

TID/USD-Supported Documents

Documentation supported by the Technical Information Department and User Systems Division resides in the central storage system and can be retrieved for viewing and printing on the CDC 7600's. The following are new and newly revised documents released online from July 15 through August 15, 1981, for viewing and printing. (Octopus Communiqués are not included on this list.)

Document title	DOCLIST Reference	Document Number	Title page date
BCON*	BCON	LCSD-516	Nov. 11, 1981
EOS4 Package Manual Rev. 1	EOS4	UC1D-1436	July 7, 1981
GRAFLIB Documentation Description*	GRAFDOC	LCSD-601	July 22, 1981
GRAFLIB: Graphics By Example (Level B)*	GRAFBHO	LCSD-426	July 22, 1981
GRAFLIB: Graphics By Example (Level A)*	GRAFLHO	LCSD-423	July 21, 1981
GRAFLIB Beginner User Manual*	GRAFLUB	LCSD-424	July 22, 1981
GRAFLIB Advanced User Manual*	GRAFLUA	LCSD-425	July 21, 1981
GRAFLIB Appendix A (Sample Programs)*	GRAFLEX	LCSD-432A	July 21, 1981
GRAFCORE Appendix A (Sample Program)*	GRAFCEX	LCSD-434A	July 21, 1981
GRAFCORE Appendix B (DLI Description)*	GRAFDLI	LCSD-434B	April 21, 1981
GRAFSM User Manual*	GRAFSMU	LCSD-1736	July 22, 1981
GRAFCON User Manual*	GRAFCON	LCSD-1735	July 22, 1981
GRAFCORE: Graphics By Example*	GRAFCHO	LCSD-433	July 21, 1981

* Draft

To view the first two documents in this list on a TMDS monitor, execute

TRIX AC!OD!*docref*!TV*number*

(type PTC to see the table of contents and Pi to see page i, etc.,) or to print a hardcopy of any of the documents listed above, execute:

TRIX AC!PRINT!NIP *docref* BOX *ann id*

where ! is the linefeed, *docref* is either the document number without the hyphen (e.g., LCSD158) or the DOCLIST short title, (see OC-1566), *ann* is your BOX number and *id* is any identifying string of alphanumeric characters.

TIDLIST is a list of all TID/USD-supported documentation and is updated weekly. To view or print this list (about 52 pages), execute either of the above execute lines, substituting TIDLIST for *docref*.

Joanne Perra, TID/USD
Ext. 2-3762

Preliminary Documents Not Yet Submitted to TID

In addition to the GRAFLIB and GRAFCORE Drafts released through TID (preceding page) the following have also been updated.

LCSD-434	GRAFCORE Reference Manual
LCSD-434S	GRAFCORE Routing Summary
LCSD-432	GRAFLIB Reference Manual
LCSD-432S	GRAFLIB Routine Summary

These all have dates of July 1981.

CHORS is experiencing difficulty with printing the GRAFLIB and GRAFCORE Reference Manuals and Routine Summaries due to the size of the files; for that reason we are asking people to limit the online printing of those manuals and instead send me a request.

If you can hold off awhile longer, the Advanced User Manual will be completed and you will no longer need the reference manuals. It was never intended that users would have reference manuals. The old reference manuals should fill the gap until the user manual appears.

If you have any questions, please call me.

Kelly O'Hair, CGG/USD
Ext. 2-4296

COMPUTER EDUCATION

It's time to plan your participation in fall quarter Continuing Education Courses. We are offering two courses in Computer Graphics, one new course and one old favorite. Kelly O'Hair will be the proctor for CE1004, Computer Graphics using GRAFLIB. This ten week course starts on October 5 and will be presented on Monday and Wednesday at 2:00 in Bldg. 113, room 1206.

Don Vickers and Nelson Max will present the new course, CE1006, Computer Animation Showcase. Take this course to get excited about computer animation and see state-of-the-art computer movies. This course also runs for ten weeks on Monday and Wednesday starting October 5, Bldg. 113, Rm. 1206, from 11:00 to noon.

Bob Hughes will once again proctor CE1010, Basic Introduction to the Octopus Time Sharing System. We offer this course each quarter at some location in the red badge area and suggest that new employees or any employee who expects to start using the Octopus Computer System should take this short ten-hour course. The class schedule will be announced later.

CE1202-A Introduction to Programming and Fortran which was taped by Paul Dubois will be proctored by Daryl Dell and offered again this fall. These twenty class periods will be offered on Tuesday and Thursday starting October 6 at 11:00 in Bldg. 113, Rm. 1206.

Last month's discussion of our PLATO terminal stirred up considerable interest. I would appreciate any feedback users could provide on the value of the various courses. The *PLATO Catalog of Courses* lists over 50 courses under computers and data processing. A partial listing of courses and subjects include:

Introduction to Machine Language Programming	PL/1 related courses
A complete 60-hour course in structured Fortran	Courses in BASIC
Introduction to Computer Math	Short courses in APL

The catalog has a section on mathematics and statistics. Most of these are short courses on an introductory college level. Included under mathematics is a session on Boolean algebra, various trigonometric functions, set theory, and Fourier series.

The section on statistics includes about 20 offerings. Some of the areas covered include:

Matrix Manipulation	Eigenvalues and Eigenvectors
Analysis of variance	Binomial distribution
Multivariant analysis	Reliability and validity

By next month I hope to be able to list some additional opportunities in computer science that are not currently available to us.

Daryl Dell, DEG/USD
Ext. 2-4394

WANTED

A sorting routine for the CRAY-1 similar in performance to the SORT command in TRIX AC on the CDC 7600.

(Wanted by Stan Bumpus, Ext. 2-0316)

The first person who turns up a routine matching the description will receive a *Tentacle reward*. Contact me or write to *Tentacle*, L-301.

If you are looking for a piece of software that performs a particular function, generates a certain kind of output, etc., chances are good that it exists on our system.

To help avoid duplication of effort, and to possibly save you some time, this section is for announcing such "wanted" software.

If you wish to use this *Tentacle* service, write a brief description of the kind of software wanted and mail it to *Tentacle*, L-301; or contact Ted Stullich or me. Your ad will appear in next month's issue.

Gail Whitten, Ext. 2-3704
Consulting Office, USD

PUBLIC FILE USAGE STATISTICS

The public file usage statistics for August 1981 are now available and can be read from storage with the following command:

Statistics for the CDC 7600's

```
XPORT RD .999900:PFUSTAT:OUTPUT[AUG81]
```

To print this report from a CDC 7600, type: ALLOUT HSP AUG81 BOX *ann id*

Statistics for the CRAY-1's

```
XPORT READ .999900:PFUSTAT:OUTPUT[AUG81C]
```

To print this report from a CRAY-1: ALLOUT HSP AUG81C BOX *ann id*

To print this report from a CDC 7600: ALLOUT HSP 8BIT. AUG81C BOX *ann id*

Larry Sears, SIG/USD
Ext. 2-0824

SUBMITTING ARTICLES

To encourage authors to submit articles *online*, I have outlined below the few steps that are needed to make an online file.

On the CDC 7600's:

```
TRIX AC
.RED7                          (allows upper/lowercase from a TI 700, use the
                               SHIFT KEY to get uppercase)

.C(filename)
.BL1                           (TRIX prompts with the "." or "&")
&text - - - -
&text - - - -
&#PH                           (for a new paragraph, or leave a blank line)
&text - - - - , etc.
&Name, Phone Extension
&Group/Division
&.                             (period ends "text mode")
.END
```

On the CRAY-1's

Use TRIXGL in the same way but DO NOT type RED7 (upper/lowercase is the default).

No macros please.

Once you have the online file, submit for publication by:

XPORT!WR .960262:TENTACLE:*filename*!END

This will help us considerably.

Gail Whitten, USD
Ext. 2-3704

OCTOGRAM SUMMARY

REDPP Update – July 21, 1981

REDPP has been updated on the R, U, and S machines (this is version 377, old version was 372). Many bugs that have been reported over the last year or so have been corrected (not all of them), however there are some problems that need to be pointed out:

1. REDPP was never designed to output to more than one medium at a time, for example going to the RJET and the FR80. REDPP should be run once for each output medium: TMDS, RJET (or UX80), FR80 (or DICOMED), or NIP. Although REDPP is device independent, it cannot be expected to output to multiple devices and create correct results for all of them. Every device has its own characteristics and peculiarities, and REDPP takes all of this into account when plotting. It is recommended that you generate UX80 files directly from REDPP and then use the UX80 system to view the files on the TMDS (UXTV), send them to an RJET (UXRJ), or send them to the FR80 (UXFR). You could also generate FR80 files and use FRTV to do the same thing. There are no guarantees for users who insist on running REDPP to multiple devices.

2. REDPP still has some problems including UX80 pictures in your report. If you are not using the options XNICE. or NICE. on your REDPP input line, don't worry about this problem because you will never see it. If you are using XNICE. or NICE. and you experience trouble, please see me about it. This problem only shows up on output from the NIP printer.

3. The following REDPP options do not appear in the REDPP document:

   NIP. [KEEP.] or NIPS. [KEEP.]
   Create output for the NIP printer.

   UX80.
   Create UX80 files and leave them in your file index.

   XNICE. [KEEP.]
   Send output to the NIP printer (8.5 inch paper) rotated and using DEFONT. 4.

   NICE. [KEEP.]
   Send output to the NIP printer (8.5 inch paper) rotated and using DEFONT. 1. You must follow this with the DEFONT. *n* option and must not use fonts 0 or 1 (I will try to fix this).

   REPORT.
   Prepare output for the FR80 hardcopy camera with cut marks (keeps the FR80 files in your file index).

4. Sometimes REDPP has trouble sending output to new or recently updated RJET stations. If you have trouble like this, the best course of action for you is to generate UX80 files (with the UX80. option) and run UXRJ yourself. REDPP's communication with UXRJ is not very good (REDPP runs UXRJ to send pictures to the RJET station.)

Please contact me if there are any problems with version 377 of REDPP.

Kelly O'Hair, CGG/USD
Ext. 2-4296

Update of Public File XPORT - July 21, 1981

The version of XPORT that has been on the U machine for the last week has been moved to the R and S machines. Also, the version of XPORT that has been on the C machine for the last week will be modified slightly for compatibility with the new CRAY-1 system and will be placed into public on all CRAY-1's. See OC-1568 for details about this new version of XPORT. Please contact us immediately if you have any questions or comments.

Neale Smith, Ext. 2-0822
Rich Belles, Ext. 2-4697
SIC/USD

REDPP Update - July 22, 1981

REDPP has been updated on the R machine (this is version 378, old version was 377). I have tried to fix several problems:

1. I have partially fixed the UX80 picture problem, but only for certain cases; at least it is better than it was. If you are not using the options XNICE. or NICE. on your REDPP input line, don't worry about this problem because you will never see it. If you are using XNICE. or NICE. and you experience trouble please see me about it. This problem only shows up on output from the NIP printer.
2. The NICE. option now uses font 2 (not font 1), so you will no longer get sleeping characters on NICE. output.
3. The cut marks that are plotted by REDPP (using the FRAME. option) are now plotted 6 times so that they will be darker. The automatic paper cutter was experiencing trouble detecting faint cut marks.

Please contact me if there are any problems with version 378 of REDPP.

Kelly O'Hair, CGG/USD
Ext. 2-4296

GRAFCON: 2D CONTOURING PROGRAM - July 22, 1981

GRAFCON will accept commands and data files from users and construct 2D labeled and smoothed contour maps. Users have control over many of the contouring options and also the format of the picture.

GRAFCON uses GRAFLIB and GRAFCORE and runs on the CDC 7600 and the CRAY-1. To get a user manual for GRAFCON type:

BCON GG:DOCOUT CON. BOX *ann*

Where *ann* is your box number (only type your box number, no ID).

Kelly O'Hair, CGG/USD
Ext. 2-4296

FRTV on the CRAY-1 - July 23, 1981

A bug that caused FRTV to blow up with error 204 when outputting frames to NIPS has been fixed and updated on the CRAY-1.

Steven Williams, CGG/USD
Ext. 2-4299

DDT Update - July 23, 1981

DDT version 7.2 has been placed in public on all machines. The update was done early to fix a serious TMDS bug. Also included in this version is the use of registers in conditional breakpoints. Using the .REG. function, you can specify a register in the if-part of a BKP command (e.g., $100 IF .REG.(S1) .EQ. 6). On the CRAY-1 you can reference any A, B, S, T, or V register. On the CDC 7600 you can reference any register except the PC and PSD.

Dave Seberger, LG/USD
Ext. 2-4038

LIB Update - July 28, 1981

The version of LIB which has been in file NEW was put into public today. This version has the new option to list the contents of a library to disk instead of the teletype. The option is invoked by adding the DISK. qualifier to the L or LL command. A long list of the library is then written to a file named P-LIB. For example, to list a library file called XYZLIB to disk, type

```
LIB XYZLIB!L DISK.!END
```

Please contact me if you have any questions or comments.

Rich Belles, SIG/USD
Ext. 2-4697

Bug Reports - July 28, 1981

The Consulting Office is beginning a formal bug reporting system for software/documentation bugs and anomalies. Each bug reported will be written up on a special form; one copy will remain in the Consulting Office, another will go to the owner of the software/document in question. On a regular basis, we will check on the status of the bug and report back to whomever reported it when some action has been taken or the bug has been fixed.

This will insure follow-up and improve our current system of keeping track of reported bugs.

Gail Whitten, Consulting Office/USD
Ext. 2-3704

LDR - July 28, 1981

A new version of LDR was placed in public on all CRAY-1's today at 1:30 pm. This version fixes a bug that affected the EQUIV command, allocates a bigger symbol table file for codes using the "SIZE=L" command, and does some additional linkage verification on segmented codes. There are no changes in the user interface to LDR.

Rick Johnson, LG/USD
Ext. 2-4496

Clarification on BASELIB Version 5A - July 29, 1981

There have been several inquiries about a change to CRAY BASELIB Version 5A regarding the removal of the Q8IOCS file name check from IZDKIN, IZDKOUT, and IZDKPTRN. This check was removed to bring the CRAY-1 BASELIB more in line with its design goals.

The result of this change is that any attempt to do I/O on a file that is not attached to an IOC will generate a nonrecoverable error 300 from the operating system. To remedy this situation, (a) attach files via an open or create prior to issuing the above calls, or (b) check common block Q8IOCS for a nonzero entry corresponding to the IOC number.

Please do not hesitate to call me with questions about BASELIB.

Vicki Scott, SIG/USD
Ext. 2-0573

MODEL - July 29, 1981

A new version of the MODEL compiler was placed in public on the C machine today. This version handles function return values correctly, regardless of the number of words needed to contain the result or the context of the function call. Previously, such results were limited to 64 words, and they had to be assigned to a variable before use. A bug that caused bad code to sometimes be generated when the index variable of a "for" loop had the same name as a declared variable has also been corrected.

If no problems are reported, MODEL will be updated on the D & E machines.

If you have problems or questions with the MODEL compiler, contact one of us.

Bonnie Toy, Ext. 2-3752
Dave Cox, Ext. 2-1329
LG/USD

Public File EOS4 Update - July 31, 1981

Public file EOS4 was updated today. The reason for this update is to replace the inverse table lookup package with a modified version.

INVSEL and INVSET have been modified to fix a bug that occurred if one restarted from a dump without reselecting. This change has no effect on normal users. A bug was fixed in EOSVINV that caused an error if there was an EOS table in memory which did not correspond to any zone.

If there are any questions about these changes, call Paul Dubois, Ext. 2-4237.

Jim Hegarty, NSSD
Ext.2 4678

New Versions of SLOPE2, FTN, and FTN5 - August 5, 1981

The files that compose the SLOPE2 system, the FTN/LTSS system, and the FTN5/LTSS system were updated in public today at noon. The files to be updated include:

SLOPE	FTN	FTNLIB1
SLOPE2	FTN5	FTNLIB5
SLP2SYS	FTNLIB	FTNLMAK
		SYS2FTN

Note that after this update, SLOPE and SLOPE2 will contain the same version of SLOPE. Users needing the old version of either SLOPE or SLOPE2 may obtain it from the .999900:OLD directory (see OC-1526). The new default system file is SLP2SYS.

Machine-readable documents are available for SLOPE2 and FTN, and shortly, FTN5. To print your own copy, use the commands:

For SLOPE2:

```
XPORT RD .381388:SLOPE2:DOCS:SRM1END
ALLOUT HSP SRM 8BIT. NOPG. BOX ann id
```

For FTN:

```
XPORT RD .381388:NEWFTN:DOCS:FTNRM1END
ALLOUT HSP FTNRM 8BIT. NOPG. BOX ann id
```

These documents supplement but do not replace the CDC manual for NOS and FTN.

This version of SLOPE2 corresponds to the CDC PSR level 531 of NOS. The FTN/LTSS system is a minor update. The FTN5/LTSS system is a new compiler that is an implementation of ANSI '77 FORTRAN, with some LASL-Library I/O restrictions.

Terry Heidelberg, LG/USD
Ext. 2-4154

TV80LIB to GRAFLIB Translator, "TVGRAF" - August 7, 1981

A TV80LIB to GRAFLIB converter, begun by Kelly O'Hair and Pete Keller, and updated by Bob Herbold and Ernie Pyle, is now available. The program's name is "TVGRAF" and it translates typical TV80LIB calls in programs that are compiled by the CHAT, CIVIC, or CFT compilers. The translator has been tested on several physics codes and does roughly 85% of the conversion.

"TVGRAF" expands a TV80LIB call to one or more GRAFLIB calls. This often produces a larger program. However, the disadvantage of a larger program will be offset somewhat by speed, since GRAFLIB routines run 20 times faster on the CRAY-1 than TV80LIB routines. All in all, the translator provides a quick and dirty way to convert a code using TV80LIB to a code using GRAFLIB.

Once the translated program is up and running, some "fine tuning" of graphics output may be needed. If this is the case, contact the Graphics Group for help. If the translated code is heavily used, the Graphics Group can also aid in optimization.

For details on the translator and how to use it, contact Kelly O'Hair, Ext. 2-4296.

Ernie Pyle, CGG/USD
Ext. 2-0545

Memory Manager - August 12, 1981

The Memory Manager has been updated today on all the CDC 7600's and CRAY-1 computers. A number of changes were made to the library. Except that some of these changes detect more error situations. The users should not be adversely affected.

1. MALLOC, MALLOCB, MALLOCD, and MALLOCT were changed. When any array of zero length is allocated, the pointer, index, or descriptor is set to an illegal address.
2. MGETLPD(name,upntr) was added to the library. This function returns a descriptor for variable "name" in upntr. "name" may be allocated by any of the four allocation routines.
3. MGETLP, MGETLPB, and MGETLPD will return a non-zero function value if the input array name is not found, is inactive or has a zero length. If an error is detected by these routines, the user's pointer, index, or descriptor will be set to an illegal address.
4. MGETATT now returns the name of the variable in atts(1).
5. MSETLOG, MSETLOGB, MGETATI, and MSRCLOG will be withdrawn at a future date. These routines make use of an internal MMLIB index. It is dangerous for users to access the Memory Manager's data base in this manner. With the elimination of these routines, access to the Memory Manager will be solely symbolic. MSETLOG and MSETLOGB have been changed so they do not use the first argument either as input or output. Also, before they were subroutines: now they are functions and may return errors.
6. MSETLG and MSETLGB have been added to MMLIB TO REPLACE MSETLOG and MSETLOGB

respectively. The only difference is the removal of the first argument in the calling sequence, the Memory Manager's table index. USERS ARE ENCOURAGED to change their codes to call these new routines before the old routines are deleted from the libraries.

Documentation has been updated and can be obtained by executing the following:

```
XPORT RD .177891:MLIB:MMDOC

TRIX AC!PRINT!NIPS MMDOC BOX ANN ID!END
```

If you experience any troubles with the new library or have any questions, please call.

Bob Cooper, NSSD
Ext. 2-4653

IMP Version 2 - August 17, 1981

IMP Version 2 will be placed in public on all CDC 7600's and CRAY-1's on Monday, August 17, at 10:00 am. New features in this version include: a history file, automatic user macro file saving, a new Macro Control Language, and a change to the SWITCH command.

The history file records all user type-ins and IMP processing. It is optional, and may be turned on and off. A mechanism has been created whereby IMP will save and automatically retrieve a user macro file. The new Macro Control Language includes IF, GO TO, labels, and controllee message testing.

All of the new features are reflected in the document, which you can obtain by starting IMP and type "1". If you have any problems with the new version of IMP, please call me.

Rick Johnson, LG/USD
Ext. 2-4496

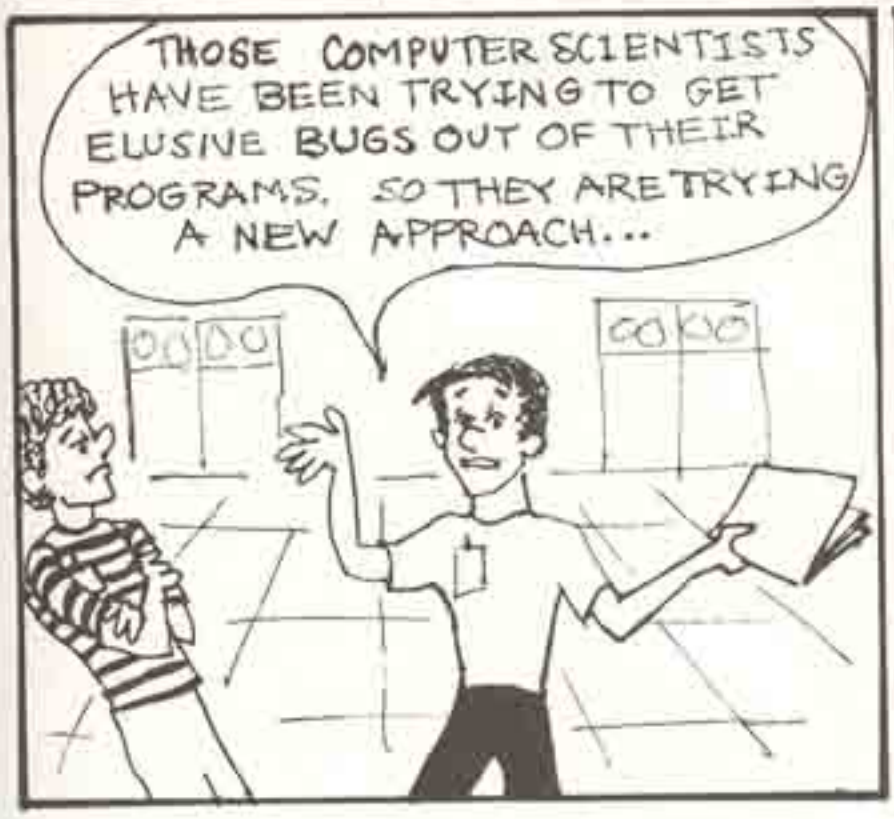
IN THE BEGINNING

Thirty years ago – 1951

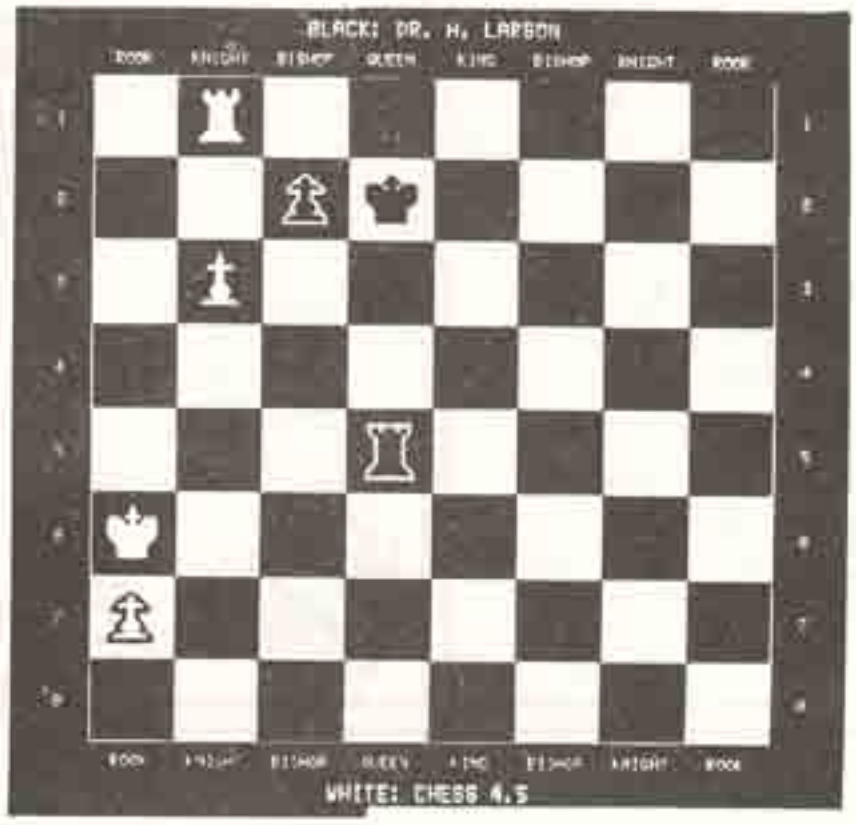
The UC Berkeley Radiation Laboratory had opened up an office at the former Navy base in Livermore (where LLNL is now), and was advertising for a "girl Friday" for the Berkeley engineering group commuting from UC. The Navy base was occupied by the California Research & Development (Standard Oil) group, known by everyone as CR&D and Livermore's total population was about 3,000. There was constant traffic, even then, with people from Los Alamos, University of Chicago, Washington DC, etc.; in fact, I was interviewed by a Dr. Fields from the University of Chicago. That was the year that Edward Teller was trying to convince the politicians that a laboratory, other than Los Alamos, was needed; Earnest Lawrence was trying to convince Dr. Teller that he should come to U.C. and I was trying to convince U.C. that I'd make a good "girl Friday." Dr. Teller got his laboratory; Dr. Lawrence got Teller; and I got the job in my home town.

Cecilia A. Larsen, ADP R&I/COMP
Ext. 2-4232

THE OCTOPUS'S GARDEN



puzzles by Henry Larson



This is a betting problem. You may choose to play either color. White moves first. If you take the White pieces, you must mate to win the bet. If you take the Black pieces, a draw will do. Which side do you take?

Editor's Note

Anyone wishing to contribute solutions or comments should send them to *Tentacle*, L-301. Responses received before the next *Tentacle* deadline can affect Henry Larson's solution, scheduled to appear here next issue.

Solution to Bridge Problem from *Tentacle* I

Several readers responded to this. A few minor variations in the layout of the cards were proposed. One respondent confided that the problem "forced" him to spend 1-1/2 hours of Lab time solving it. Another solver commented, "Ideally the mark - namely you - in a scam like this should never suspect a thing, not even after it is all over. However, in this case, the following things may be at least slightly suspicious: the distribution of the cards, the likelihood of the contract, the basis for the double, and the wisdom of the opening lead. On the other hand, there need be no questionable play by your partner."

With this in mind, the layout below seems to go about as far as possible in the direction of allaying suspicion. And, of course, whatever the bidding was, it must have been spirited, so that the necessary voids might not seem too unexpected.

♠	♡	♢	♣
A	K	-	A
K	Q		K
	J		T
	T		4
	9		2
	6		

Possible Bidding (?)

W	N	E	S
1D	2D	P	2S
5D	6H	D	P
P	6S	P	P
7D	P	P	7S
D	P	P	P

♠	-
♡	84
♢	AKQJT9843
♣	Q6

Partner		You
	Dummy	
		♠ J97532
		♡ A7532
		♢ -
		♣ J9
	Declarer	

♠	♡	♢	♣
Q	-	7	8
T		6	7
8		5	5
6		2	3
4			

In view of the bidding, partner can hardly be faulted for leading a heart, (a club lead would be equally fatal). In any case, declarer continues hearts (discarding the two low clubs, then diamonds) until you cover, and then ruffs with the 4. Declarer next returns to the board, leading high hearts and clubs until you are reduced to nothing but spades. The last six tricks are made by cross-ruffing, and your half-dozen pieces of trump are rendered as worthless as a deadbeat's check.

INDEX

BASELIB Version 5A
  Q8IOCS file name check, 38.
BCON document update 30.
Bug Reports 37.

CDC 7600
  Statistics for, 33.
CHECK
  New version, 23.
CHORS II
  Initial plans, 1.
  Status-reporting, 10.
  User requirements document, 10.
  User-Critique meetings, 10.
CRAY-1
  Statistics for, 33.

DDT
  Conditional BKP., 37.
  REG function, 37.
  Update version 7.2, 37.

EOS4
  Document release, 39.
  Update, 39.

FRTV Update 37.
FTN5
  New version, 39.
FTN
  New Version, 39.

G Machine
  New home, 1.
GRAFCON
  2D Contouring program, 36.
GRAFLIB
  Convertor from TV80LIB, 40.
  Documentation, 31.

Integral
  Approximate formula, 25.
Intelligent Terminals
  A survey, 1.

LDR Update 38.
LIB
  DISK. option, 37.
  Update, 37.
LINCS 6.

MATHLIB
  FITPOL, 25.
MODEL Update 38.
MSD-Mathematics & Statistics
  Algorithm development, 5.
  Consulting services, 5.
  Large code support, 5.

NLTSS
  Description, 6.

Octopus Communique
  OC-1569, 23.
  OC-1571, 25.
Online Documents
  BCON, 30.
  EOS4, 30.

Physics integral formula 25.
Puzzles
  Bridge solution, 45.
  Chess, 44.

REDPP
  FRAME. option, 35-36.
  NICE. option, 35-36.
  REPORT. option, 35.
  UX80 files, 35.
  UX80. option, 35-36.
  Version 377 update, 35.
  Version 378 update, 36.
  XNICE. option, 35-36.

SLOPE2
  New version, 39.
SORT for CRAY-1 33.

Tentacle
  Submitting articles, 34.

TV80LIB
  to GRAFLIB translator, 40.
TVGRAF release 40.

XPORT
  Update (Ref. OC-1568), 36.

NOTICE

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under contract number W-7405-ENG-48.

ORGANIZATION OF THE COMPUTATION DEPARTMENT

ADMINISTRATIVE STAFF
H. Wayne Hudson—Program Manager
Sherry Abraham—Department Secretary
Carol White—Affirmative Action Coordinator

COMPUTER RESEARCH GROUP
George Michael—Group Leader

COMPUTATION DEPARTMENT

ADMINISTRATION

DEPARTMENT HEAD

JOHN E. HANELLETTI

SPECIAL PROJECTS GROUP
Ira Morrison—Group Leader

APD REVIEW AND INVENTORY OFFICE
Ray DeSaussure—Group Leader

COMPUTATION PROJECT GROUP (EX-DEPARTMENT)
And Other Support

COMPUTER OPERATIONS DIVISION

Division Leader
George Vranesh

Division Secretary
Aileen Dantzler

Division Staff
Owen Brovont
Ed Schoonover

Production Accountability
Robert Bennett
Joe Davison

Coordination Center Group
Don Emery

Training Management Group
Doug Holt

Operations Interface Group
Gus Wilgus

Mainframe Operations Group
Chuck Cole

I/O Operations Group
Curtis Kluttz

NETWORK SYSTEMS DIVISION

Division Leader
Sam Mendicino

Division Secretary
Sharon Murray

Operating Systems Group
Pete Du Bois

Support Processor Group
George Sutherland (PL)

Communications Group
John Fletcher

Storage Group
Garrett Boer

Research Group
Richard Watson (PL)

(PL) Project Leader

USER SYSTEMS DIVISION

Division Leader
Richard Swakenberg

Division Secretary
Gail Sims

Division Staff
Gail Whitten

Systems Interface Group
Stan Solbeck

Documentation & Edu. Group
Dave Thompson

Data Management Group
Keith Johnson

Language Group
David Seberger

Computer Graphics Group
Donald Vickers

MATHEMATICS AND STATISTICS DIVISION

Division Leader
Nora Smiriga

Division Secretary
Judy Tate

Division Staff
David Margolies

Statistics Section
Nora Smiriga (acting)

Library Software Section
Richard Hesinger

Programmatic Applications Sec.
Paul Dubois

Numerical Mathematics Section
Ralph Carlson

APPLICATIONS PROGRAMMING DIVISION-I

Division Leader
Tad Kishi

Division Secretary
Terry Contreras

K-Programming Group
Landon Bruce

C-, J-, H-Programming Groups
Bob Gulliford

T-Programming Group
Ron Litteral

Y-Programming Group
Bill Warren

G & L-Programming Group
Pat Weidhaas

NUCLEAR SOFTWARE SYSTEMS DIVISION

Division Leader
Chris Hendrickson

Division Secretary
Marlene Wagner

A-Programming Group
Mike Pratt

B-Programming Group
Dick Giroux

D-Programming Group
Mike Uyemura

APPLICATIONS PROGRAMMING DIVISION-II

Division Leader
Ted Michels

Division Secretary
Phyllis Schmitt (acting)

Minicomputer Software Support
Ted Michels (acting)

Mini/Micro Systems Group
Joe Brandt

MFE Mini/Micro Systems Group
Wall Ng

MK/KS Group
Jack Matthews

M Group (large codes)
Tom Brengle

Weapons Engineering Group
Tom Kuczmarski

Chemistry Group
David Belshaw

C-DEPARTMENT PERSONNEL: 402.9 FTE — NON C-PERSONNEL: 58.2 FTE

COMPUTER OPERATIONS DIVISION

GEORGE VRANESH - Division Leader

Division Secretary - Aileen Daniels
Secretarial Support - Elaine Rodriguez

Staff
Owen Bryant - Division Administrator
Ed Schoonover - Human Resources Coordinator

Production Accountability
Robert Bennett
Joe Dolson

COORDINATION CENTER GROUP

GROUP LEADER
Don Reary

Robert Anderson
Pam Gomes
Walter Headley
Guylaine Hegarty
Doug Johnson (EE)

INPUT OUTPUT OPERATIONS GROUP

GROUP LEADER
Curtis Kiutis

William Howard-Staff

User Services
Forrest Allen

Tape Vault Librarian
Julius "Lou" Bratney

Keypunch
Sandra Bosas
Carole Watson

Input/Output Operations

DAY SHIFT
Charlene Frey-Supervisor
Valerie Hill
Dale McDonald
Clarence Reese
Willie Shakelfoot
Steve Stinson
Debra Thomas
Jonel Vollenweider

SWING SHIFT
Ed Long-Supervisor
Jeanette Cruz
Michael Lazzerini
Gerald Mueller
Angela Rocha
Billy Payne
Roilene Peraony
William Stanford

OWL SHIFT
Jack Smith-Supervisor
Willie Harrison
Rodney Kent
Roy Wedelton
Rudy Miller
Bettie Myers
Lutgarde Villarosa
George Williams

Photo Lab
Donald Birte-Supervisor
Charles Bechtell
Dennis Bruce
Donald Culhan
James Gomes
Dennis Jorgenson
James McElhaney
Sherwood Thomas
John Vlassakis

MAINFRAME OPERATIONS GROUP

GROUP LEADER
Chuck Cole

Barbara Costello-Staff

Mainframe Operations

DAY SHIFT
Lee Spencer-Supervisor
Dave Anderson
Chuck Bowman
Wayne Bunker
Robert Cordero
Bruce Frame
Joseph Hammett
Irene Harris
George Mitchell
Ronald Monson
Roberto Morgan
Frank Paquette
Frank Salinas
Jennie Shen

SWING SHIFT
Rene Becker-Supervisor
Dennis Azuar
Susan Brown
Lorene Draper
Richard Everll
William Harris
Russell Hale
Tom Hutchinson
Yvonne Nelson
Joseph Perez
Michael Peray
Virginia Quintana
Josephine Stuart

OWL Shift
Victor Ceballos-Supervisor
Mark Hewitt
Joseph Kato
Judith Manoukian
Raul Reyes
Antonio Ruiz
Karen Tootle
Sandra Wyatt

I/O Red Books
Susanna Barbero
Steve Chavez
Dennis Gronner
Jeff Johnson
Kasper Kurmis

OPERATIONS INTERFACE GROUP

GROUP LEADER
George Wilgus

Phillip Eckert
Richard Frobose
ken Johnson
Alice Pitts
George Peoles

TRAINING MANAGEMENT GROUP

GROUP LEADER
Doug Heil

Doug Brown
Lynn Graves
Solomon Plume