EUGENE D. LOUREY
Research Associate
Bio-Medical Library Mini-Computer Project
University of Minnesota
Minneapolis, Minnesota

Systems Design for a Minicomputer-Based Library Data Management System

In the past, most automated systems employed large, general purpose processors, or very specialized applications software, usually both. An approach employing either method does not support library automation in any but the largest and best-financed libraries. Smaller libraries must either affiliate with these larger libraries or share equipment with nonlibrary applications. In either case, processing methodologies, standards and equipment decisions are made outside the affected library by persons unfamiliar with their requirements.

Current technology and pricing make this approach to library automation unnecessarily restrictive to individual libraries and unnecessarily expensive for the library community. Equipment selected for its suitability in library-type problems combined with generalized data management software designed for library applications will make dedicated, self-contained systems cost effective for most libraries.

THE SYSTEM

At the University of Minnesota Bio-Medical Library, we are building an automated library data management system. The system will encompass all aspects of library processing activities: acquisitions, serials, cataloging, circulation, and reference services. It is an on-line, integrated system built to be a prototype which can easily be exported to other libraries. It is based upon low-cost, dedicated minicomputer hardware and advanced software concepts.

Hardware

For our system we have selected hardware which we believe is particularly well suited to library applications. Our CPU is a DEC PDP 11/40. This is a 16-bit machine with excellent character processing capability. It is very fast, is memory-expandable to 256K bytes, and has memory management which is comparable to virtual memory and memory protect. Also, peripheral attachment options are unlimited. This is important for libraries since some peripherals devices which will be very useful for them are not yet perfected.

We are using removable pack disks as our on-line storage devices. We presently have a single controller and a single 40 million character drive. Both are marketed and maintained by DEC. We can attach up to 8 drives to a controller and multiple controllers to the CPU. This is an area where we are expecting price and technology changes. Prices have been falling steadily for several years and libraries are ideal applications for read-only storage devices which have a high potential for cheap, permanent on-line storage.

We have two industry compatible 9-channel 800 BPI tape drives. The first is for transaction recording; the second is for off-line sequential storage, exchanging data with other systems, and back-up for the transaction recording unit.

Our central printer is a high-speed 132-column line printer. We expect to use this capability only for the production of long hard copy lists and for problem diagnosis memory dumps. We plan to have operational print capabilities decentralized to provide system users direct control over the preparation of such materials as orders, claims, cancellations, cataloging worksheets, catalog cards, fine notices, recalls, and bibliographies. This decentralized print capability will consist of selectable font printer/plotters with independent horizontal and vertical carriage control. These devices can satisfy virtually any hard copy print requirements but are low-speed and thus relatively inexpensive.

Our user terminals are Superbee CRT and keyboard terminals. These devices are excellent for data entry. They have character and line editing capabilities, 2048 characters of terminal memory, a large screen size (25 by 80 characters), and a large number of displayable characters. Their screen size and memory capacity also make them excellent for file inquiry.

These devices, with the exception of the low-speed printers, are already installed and in use for test purposes. We expect to acquire two additional devices, both primarily for circulation: a wand reader for identification of library materials and borrowers, and a special printer for preparation of labels with bar coding to support the wand reader.

Software

The software for our system employs several techniques to minimize development time, development effort, and core requirements, and to maximize system flexibility–modularity, runtime parameters, dictionary descriptions of data items, assembly time parameters, and control by user command. These techniques are not new, but the use of them all in one system, the use of them in library systems, and the use of them with minicomputers is quite uncommon.

Our software is designed and programmed in small self-contained modules. Each module performs a specific activity but is generalized so that it can perform its activity in a variety of circumstances. An example of such a generalized module is the table conversion routine. It searches a user-specified table which it reads from disk storage for an equivalent argument and returns the conversion value. The lengths of the table, argument, and conversion value are all variables. These small modules are easier to debug, easier to change if necessary, and they reduce total system core requirements.

Modules are particularly important for those areas of the system which are common to all applications such as searching indexes, extracting field values from a record for output generation, and converting encoded data to display form. They are also important for software interfacing with peripheral devices. Input and output from all devices will be converted to our internal form. Only the interface software will deal with data in its external form. In this way, adding new peripherals will only require adding a new external interface module to the system repertoire of software modules. Also, interfacing with another system will only require adding a software interface to make the exchange data compatible.

Runtime parameters control the execution of all modules. Requests for execution of any module require specifications of the values of the module variables. For example, a request to execute the table conversion routine includes specification of the table to be used, the length and address of the argument and the length and address of the output buffer. The parameterization of these modules reduces the number of modules and the amount of core required at the expense of execution time. For our application this is a very good tradeoff, since execution time is essentially free in a dedicated system which is I/O bound.

Dictionary descriptions control all data entry, all accesses to the data base, and all output generation. Every file, record, field, index, and I/O format is described in a dictionary. Each module which accesses the data base, interprets input, or prepares output uses these dictionary descriptions. For example, the table (file) used in the table conversion routine is described in a dictionary which indicates its length, its address, its ordering scheme, and the records it contains. Each record in this case contains a pair of values, the argument, and the result of the conversion, which are also described in dictionaries. Use of these dictionaries makes all software independent of the data base. Changes to the data base will not require any modification of the software.

Assembly time parameters are used for all program references to the dictionary and to all system-defined values and references. This permits changes to the format and contents of the dictionary and restructuring of core with no changes in existing software. For instance, if a new variable is added to the description of files to accommodate a file which is treated differently than others in one or more modules, a modification of the affected modules and a reassembly of others which access the file dictionary will permit the addition of the new variable anywhere in the dictionary format. Or, if the locations in core of system defined values–such as date and clock time–are changed, a reassembly of the system will accomplish the change.

User commands control the operation of all components of the system except file security, file back-up, and management statistics. In general this is true of all interactive systems, but our software architecture provides a whole new dimension to user command control of system operation. As a result of our use of small generalized modules rather than comprehensive applications programs, users can schedule a sequence of events, with appropriate parameters for each, in much more detail. This gives users a greater opportunity to make the system responsive to their needs. At the same time, users are not required to specify all actions in great detail; default conditions will be operative when not overridden by user specification.

As an example, consider a request for an author index search. If nothing is specified except the author entry, the system will assume: that the user wants to inspect all cross references, that the output format is standard, and that the output medium is the CRT terminal device. However, if the user desired, he could specify the cross references to be shown, specify the categories of author entries to be shown, specify that subject classifications are desired in addition to the standard output, and specify that a hard copy list in order by title is required. In fact, the user can specify any combination of values of the variables he may desire.

We have devoted considerable time and effort to making this system exportable. This has required close attention to two aspects of the system: cost and versatility. An exportable system must be cost effective in many environments and it must be versatile enough to meet many differing processing requirements. There are many aspects to both cost and versatility which influenced our design and implementation decisions.

Cost Considerations

Total system costs include costs of hardware, hardware maintenance, site preparation, installation, operations, software maintenance, and supplies. During this project our principal cost emphasis has been to keep all these elements of cost to a minimum for subsequent installations of the system rather than for the project. Thus we have put extra software effort into making the system flexible, and we have purchased and are purchasing equipment which will be operational for subsequent applications of the system.

HARDWARE

The purchase price of minicomputers is considerably below that of larger processors. But there are other factors which make our approach significantly less expensive. Our modular software design is ideally suited to the hardware modularity of minicomputer architecture. All system hardware can be purchased as needed. The user pays for only what is actually needed and only when it is needed. Any peripheral device can be attached to our system; thus the least expensive device which meets the requirements can be acquired. Since DEC is by far the largest manufacturer of minicomputers, hardware interfaces between peripheral devices and DEC CPUs are usually available when the device is introduced or shortly thereafter. DEC users are seldom required to have these interfaces specially built for their use. Our use of generalized common modules in software implementation keeps core requirements to a minimum. Furthermore, the system executive has memory swapping capability enabling much of the system to be resident on the disk and only pulled into core when needed. All of these factors make the hardware costs of our approach minimal in comparison with any other.

HARDWARE MAINTENANCE

Low-cost hardware will also be less expensive to maintain, since there is usually a direct relationship between total system hardware costs and hardware maintenance costs. Furthermore, minicomputers are structurally and operationally simpler in design and require a great deal less maintenance than large, general purpose systems. Whereas large systems generally have between 10 and 20 hours of regularly scheduled preventative maintenance per week, minicomputer installations require no regularly scheduled preventative maintenance. Mean time to failure on mini systems is measured in months rather than hours. The dependability of minis is explained by their development background. They were created primarily for the military and aerospace markets to operate in spacecraft, in ships, and in mobile land vehicles. The durability of these early minis has been a benchmark for later developments.

SITE PREPARATION

Site preparation for minicomputers is relatively uninvolved in comparison to the major modifications required for large systems. Minis are physically smaller and take less room. They generate less heat and have higher tolerance for adverse conditions, high or low temperature and humidity. They do not require raised floors, special air conditioning or water cooling. Generally they will operate in completely unmodified surroundings. In addition, we are decentralizing operations by attaching remote peripheral devices which require no site preparation except clearing the space and dropping the lines to the remote sites. This reduces space and electrical requirements at the central site.

INSTALLATION

Because the hardware is architecturally simple and reliable, hardware installation and acceptance testing require little staff time. Our software approach will hopefully make software installation equally problem-free. Our methods require less code, result in more thoroughly tested code, enable most changes to be made without any recoding, and enable software changes where required to be made without unforeseen consequences. With good documentation and training materials, the cost of installation of this system should be well below any system of comparable capability.

OPERATIONS

System printing functions are decentralized so no operations staff are required to monitor and oversee the printing activities. System files are all on-line so tape loading and disk mounting of data files will not be required. Errors encountered during system operation can be reported to terminal operators for notification of maintenance staff, so central site monitoring for system failures will be unnecessary. Except for analysis and correction of system failures, no EDP-skilled personnel will be required for system operations. Any tape mounting, disk pack mounting, printer ribbon changing, or paper loading which may occasionally be required can be done by anyone with a few hours of training.

SOFTWARE MAINTENANCE

Less code and more thoroughly debugged code will obviously result in lower maintenance costs. Furthermore, all systems which are installed will be able to communicate with each other, making it possible for maintenance staff from a single site to implement corrections or modifications from any site to all other sites. Any modification to a system which is developed to accommodate any new device or any unique processing requirement will be available, once developed, to all other systems using the same software; and these modifications can be incorporated from any system to any other as the need arises.

SUPPLY COSTS

Since all system files are on-line and available on demand from any system entry point, no hard copy materials will be required for daily operations. Adequate backup will eliminate the need for hard copy data for recovery in the event of catastrophic failure. Where hard copies are needed for orders, claims, overdue notices, etc., the number of copies required can be kept to just the number sent out. No copies for the library need be kept since they can be recreated when needed. The decentralized printers can produce any required form. This will minimize the need for expensive preprinted forms.

Versatility

An exportable system must be able to accommodate changing requirements within a library over time, and different requirements in different libraries. The most important system variables which must be dealt with are: data base size; data base contents; data base structure; staffing, organizational, and processing patterns; number of processing stations or system entry points; volume of activity; data entry and output requirements; network participation; and new hardware technology.

DATA BASE SIZE

Minicomputer hardware can be configured to maintain any amount of data in the data base. On-line storage can be added as needed with no limit. However, if more than 1 billion characters are needed, it might be desirable to add another processor. The size of the data base could only affect the software if a different device were added to increase the on-line storage capacity. In this case, the only change would be a software interface module to handle the new device.

DATA BASE CONTENT

Our software makes no assumptions about the content of the data base; it is entirely defined by the data base dictionary. The content of the data base can be established initially to meet requirements as of that time. Later, it can be changed if the requirements change. For instance, if initially a library does not enter and maintain the physical dimensions of materials, but later decides these items should be included in the data base, the system can modify the file to provide for this data, print a worklist of items for which these values are not completed, and display upon demand records requiring completion of these values so the values can be entered. If the initial decision was to include this data, but later it is found to be unnecessary, the system can modify the data base to eliminate these values.

DATA BASE STRUCTURE

Our data management system includes extensive interfile and intrafile cross reference capability. This provides for as many inverted file indexes as are necessary. It also provides for all library type cross references: see, see also, see from, see also from, etc. In addition, every field can be defined as variable length and as having a variable number of values in a record. Furthermore, every file can have any number of uniquely defined records. These features make the structure of the users data base entirely variable in our system.

STAFFING, ORGANIZATIONAL, AND PROCESSING PATTERNS

The system contains a very flexible data base security system. Limits upon use can be based on particular user, class of user, terminal in use, type of data, source of data or any combination of these factors. Further, use can be subdivided into data entry, validation, modification, or examination with security restrictions applying differently to each. Security options are established at installation time, although they can be redefined later if that becomes necessary. It is up to the system users in each operating environment to establish who can perform which functions on what data from which terminal sites. Thus, the system can accommodate any breakdown of tasks, responsibilities, and authorities.

NUMBER OF PROCESSING STATIONS OR SYSTEM ENTRY POINTS

As with most interactive applications systems, the number of terminals serviced is not a critical factor for the software. Applications software is identical for one terminal as for hundreds. Systems software may be affected by a large number of terminals, but our systems software is designed assuming a variable number of terminals. The only constraint on the number of terminals attached to our system is the cost of the hardware. As the number of terminals attached increases, it may be necessary to add multiplexers, concentrators, or even additional processors. None of these will require any modification of the software.

VOLUME OF ACTIVITY

Activity volume has implications for the hardware, but does not affect system software. If high volumes are creating a CPU bottleneck, causing unacceptable response times, more core or a parallel CPU can be added at minimal cost (under $10,000). If on-line storage access is overtaxed by high volumes, additional controllers or serial processors can be added. If high volumes are creating terminal communication bottlenecks, higher speed circuits could be installed, or multiplexers, concentrators, or serial processors could be added. None of these hardware expansions would require any software modification.

DATA ENTRY AND OUTPUT REQUIREMENTS

Input and output formats and contents are user defined within broad limits. However, it is impossible to develop generalized software for data entry and output preparation which will provide for all possible forms of data exchange. Even if it were possible, the overhead would be too high to make it feasible. Our system can interpret input and prepare output for all local purposes, but software interface modules will be required for many external communications activities. For instance, our software would not be able to accept as input or prepare as output a magnetic tape for photo composition without a special module for this purpose. Similarly, it might not be able to communicate directly with other automated systems without a specially built software interface for formatting output and reformatting input.

Another form of output which the user can define within limits but cannot completely control without software modification is management reporting. Exception reporting with variable limits will be used to notify system users of a potential problem: an acquisitions account being expended too quickly; a user with many materials borrowed and overdue attempting to borrow more materials; or a consistent variation between estimated price for a class of materials and their actual price—a variation great enough to affect long-range budget planning. However, if observation of different factors than those provided is desired, program modifications will be required. Establishment of control limits for exception reporting entails sufficient system overhead that we will only provide those currently thought to be desirable.

NETWORK PARTICIPATION

Hardware interfaces are either already available or can be developed at reasonable cost to attach any on-line systems together. With the addition of the necessary hardware and perhaps a special purpose software module, our system can be attached to any other system with no changes required on the other end. This link can be to give them access to our data base, to give us access to theirs, or to do both. Thus our system can serve as a self-contained node in a large network of libraries, as a central source for a network of users, or in both capacities simultaneously for different purposes.

NEW HARDWARE TECHNOLOGY

As previously mentioned, the inclusion in the system of any new (to the system or to the market) peripheral device creates no difficult problems. In some cases, a software interface module will be required for intercepting and translating input and output, but these are inexpensive to develop once the characteristics of the new device are known. However, if it becomes desirable to use a different CPU, it will be necessary to recode but not to redesign the system unless a symbolic code conversion package or a hardware PDP 11/40 emulator is available for the new processor.

It should be clear that our system, in concept and implementation, is quite unlike previous library automation systems. We employ very inexpensive hardware, only that which is actually required to do the job at hand, and software tailor-made for library applications, but general enough to serve the needs of a wide variety of library environments. The project was proposed and funded as a prototype readily exportable to other libraries. We are still more than a year away from project completion, but are increasingly confident that our system will be able to fulfill that promise.

We expect our approach to make automation technology available to libraries with very modest resources and budgets. Furthermore, these individual libraries will be able to tailor their system to serve their particular needs, as they define them.