UNIVERSITY OF MINNESOTA COMPUTER CENTER

Deadstart Systems Newsletter

08 October 1975 Vol. 1, No. 11

Send all comments, criticisms and contributions to the editor
T. W. Lanzatella

REGARDING MACHINE STABILITY

As all readers know, the CYBER 74 has recently been suffering a mysterious malady rendering it rather unreliable. A good system programming rule for this situation is; "don't rock the boat." In order that we do not introduce any new symptoms or in any way alter machine behavior, the new tape installation is postponed until Tuesday, 14 October.

NOTICE OF CHANGES TO THE OPERATING SYSTEM

E. J. Mundstock supplied a modification to CALLPRG which repairs continuation card processing and fixes documentation.

N. L. Reddy supplied a modification to QFM which repairs processing of FET pointers while dumping a queue file. The problem was actually causing lost information between dumping and reloading queue files. The problem is fixed by CDC at level 10.

Bill Elliott added the following new options to EXAMINE:

1. The B option allows salvage of data located past the EOI.

2. The BS option allows adjustment of block sequence numbers on I format tapes. Trailer labels are also adjusted. Multi-file tapes are properly recovered.

3. The format identification section was enhanced.

K. C. Matthews made two small modifications to PP programs 1AJ and 6DB in order to make the 6638 accessible to the CYBER 74. Equipment 0 in the CMRDECK is now assigned to the 6638.

PROPOSED CHANGES TO THE OPERATING SYSTEM

Ever since the program BATCHER was introduced to the system, operators have complained mildly about the difficulty (if not the impossibility) of purging BATCHER from the ROLLOUT queue. Jim Mundstock has agreed to change BATCHER so that it is easier to purge from the queue.

A user has requested that UCC supply a utility which displays "human readable" date and time in the job dayfile. Dennis Lienke submitted the following proposal regarding this suggestion.

I suggest that an excellent vehicle for such a message is in CTIME. It could easily put all the information on one line:

	CPU Time	Wall Clock	Date
CTIME	xxx.x CPU	0945 AM	29SEPT75.

The organization of the data is probably best to keep in 10 character segments, primarily for ease of reading by programs.

A word as to time - I personally prefer military style time, a 24 hour clock, but some, perhaps many, users may perfer 12 hour am/pm style. With a bit of work, both styles may be listed.

The important part is to put it all on one line.

How about adding this feature to both CTIME and RTIME? - ed.

K. C. Matthews is responsible for the following collection of proposals.

I. Proposal for Higher Job Security

This proposal relates to the security of special jobs on the system. Specifically, it will help Barry Fox write some accounting programs which users may call, but the security measures are general enough to be helpful for other users.

Barry's program will attach a permanent file of accounting data for all users. His program will only list the data for the users account number, but data for all account numbers will be on the file. We propose to make the file a public file with a password. The password will be compiled into the accounting program.

The accounting program will be kept under CALLPRG. When the program file is attached, it will be an EXECUTE only file so that no one can read the program to find the file password. The program will call CPM to set 2 bits in the special one bit word (SOBW) in the control point area. One bit tells the system to clear core before the next control card is executed. This prevents users from examining core to look for secret data. (This bit is already operable in our system.) The second bit is the new one. It tells the system to treat this job like an SSJ= job as far as files are concerned. This means that all files that are attached or created after this bit is set, get the special ID code 74B placed in the FST entry. Then, when the next control card is executed, all files with the 74B ID code are returned. Thus, even if a TELEX user interrupts the accounting program while it is running, he will not be able to examine the special files used by that program.

The CPM function 72B added by Al Johnston will be generalized so that any bit in the SOBW word can be set by the user. This may have to be modified some day if we wish to add some bits in this word that not all users can set.

The following routines will be modified:

COMCMAC - to add an SETSOB macro for setting special bits.

LFM - to allow a job to change a 74B ID file to a 00ID file (00 is normal) without any special validation.

OBF - set ID = 74B when creating a file if the new bit is set.

CPUMTR - Process the 72B CPM function code.

1AJ - Drop 74B ID files on the next control card. Also clear the two bits when control card is loaded.

MODVAL - Use the macro to set the clear core bit.

ACLFAM - Use the macro to set the clear core bit.

CPM - Document the function 72B.

PPCOM - Document the new bit.

II. Password Hashing

This is (at last) the proposal for password hashing. A common deck COMPSPW will take the user number and password and scramble them to form a new 42 bit quantity called the scrambled password. The scrambled password, (not the original password) will be stored in the VALIDUX file.

MODVAL will be changed to call a new CPM function which will perform the scramble. Although it is inefficient to call a PP to do this, I think it's o.k. because it's done infrequently, and then the code to scramble passwords is then only in one place (the common deck COMPSPW). This will be done when account numbers are created or when passwords are changed. MODVAL also has to write the scrambled password to a file when it places all the account numbers on a source file. It must then accept these scrambled passwords when recreating the VALIDUX file from this source. CPM must also be changed to scramble the password when verifying account numbers. A similar change is also needed in 1TA to process the scramble for TELEX log-ins.

The implications of password hashing have been discussed several times in meetings of the Systems' Group, and are therefore, not going to be mentioned here.

III. CPUMTR Track Hangs

In processing the TRT (Track Reservation Table) monitor function requests for PP programs, CPUMTR often "hangs" the PP if there is something fishy about the request. (For example, if a PP drops a track that is not reserved in any track chain.) We sometimes have to unhang the offending PP in attempting to discover the cause of the problem. If there is really something wrong with the TRT, we would like no new files to be created on the device while we are looking around. This can be done by turning the device logically "off" in the system. (When the device is off, no permanent files for that device are honored.) I am proposing that CPUMTR turn the device off when an illegal TRT request is detected. It will still hang the PP. After the problem has been resolved, the device can be turned on by a console command.

IV. MSAL Word Changes

The MSAL word in low core can be used to force certain types of files to certain devices. Specifically, the five bytes of the MSAL word can force INPUT, OUTPUT, ROLLOUT, LOCAL, AND LGO files to special devices. Since it

is doubtful that we will ever want to force all local (i.e., temporary) files to device, I propose to use that byte of the MSAL word for short rollout files. We will then have the capability of putting short rollout files (from TELEX) on one device (ECS), and longer rollout files on another device (probably the 808). The following routines will be changed:

PPCOM - to reflect the changed byte in MSAL.

SET - to set the correct byte at deadstart time.

DSD - to change the correct byte from the console.

1RI - to look at the two possible rollout devices.

OBF - to stop looking at the old local file byte.

V. User ECS Changes

There is a CMRDECK command which currently specifies the amount and location of ECS reserved for user ECS. We propose to add a field to that command which specifies also the maximum amount of user ECS that a single job may request. This will allow us to increase the amount of user ECS available, while still limiting a single job to 300K of ECS. The routines that will have to be changed are:

PPCOM - to document the maximum ECS field in central memory resident.

SET - to process the new field at deadstart time.

MEM - to use the new field for ECS requests.

PEOPLE AND PROCEDURES: SYSTEM MAINTENANCE

Larry Liddiard highly recommends two new NOS design documents regarding permanent files and control card language appearing in a recent VIM Newsletter. Copies of the documents will be distributed to Systems Group members during the Systems meeting on Thursday 09 October.

REVIEW OF PROPOSAL FOR JOB SECURITY

The first line of the proposal says that it relates to the security of special jobs on the system. It does not. It relates to user programs. It enables user programs to assume in part the status of a special system job (SSJ), deriving thereby whatever file security the system provides for such jobs.

The specific problem here is a small-scale version of the bigger problem of maintaining huge data banks, to portions of which many different people have access and the security of these banks from within and without. Validux is a well-known example of a file which has information pertaining to many thousands of users, who have access to information about themselves but not about each other.

The person who maintains this data file always has to provide a service routine which can be called by different people, who have no direct access to the data file. The service routine however can get direct access to the file. This routine has to be a specialized one depending on a host of factors like the structure of the data file, types of user accesses, etc. increasing in complexity as these factors get complex and as more security is demanded. In the case of validux, the system designers knew the structure of this file and the various types of accesses to it and had at their disposal various security measures that the system already had to protect itself from mischievous users. So they threw in a couple of more things like defining a Fast Attach File type and allowing only SSJ jobs to attach them. Thus the security design for files like validux is an integral part of the security design of the entire operating system.

Now what about the poor user who wants to maintain something like a validux file? If the different categories of information he wants to supply out of this file are only a few, he can make individual files for the various categories (with password if necessary) and make each file available to one group of people so that one group does not learn about the other. Kronos can handle thousands of small files easily, and even a hundred files for this user should not be a problem to the system.

If for some reason a user wants to maintain a single file as a data base and wants thousands of other users to have access to parts of this file under the control of his service program, he has at his command several user controls which will give him all the safeguards he would ever need. By the way, I cannot believe that there exists a need for such a usage (except Barry Fox).

There are two problems: (a) Illegal access to the data file and (b) Privacy of each customer. In fact, with thousands of customers, (b) becomes more important and complicated than (a). We will consider only (a).

The three things that would provide adequate safeguards against a mischievous user gaining direct access to the data file are:

1. No one should be able to "read" the service program to find out how it obtains access to the data file.

2. Make sure the service program can always return the data file, whether it hits a time limit or other errors.

3. Should be able to clear core after its run so that core cannot be dumped to find the method of access to the data file.

The suggested solutions are:

1. Make the service routine an execute-only file so that no one can "read" it.

2. Using the macro EREXIT, ask the system to return control to some section of the program in the event of an error. If telex origin, use the macro DISTC to disable terminal control so that users cannot type in STOP or Interrupt effectively.

3. At the end of the run, the service routine zeroes out the buffers of the data file and clears the code that obtains access to the data file. He may even use the EXCST macro to execute a control cardlike SETCORE to clear all of his memory.

If there is enough demand for it, we can create a common deck to do all of these. We can also assemble this into a FORTRAN callable relocatable routine. It would have several entry points, as for example PRNPLOT has. An initial call would set the error exit, terminal control, etc. The file name, user password, etc. can be put in a common block and a call to a second entry point would attach this file and "remember" the file name in its own memory. A final call would return the file(s) and clear memory.

I believe our responsibility to users should be to help their programming efforts by providing such common decks and subroutines and at the same time to keep the system overhead at a minimum. If the user wants to clear core, it should be with his CPU time. If he wants to return file(s), it should be at his cost. Those who know 1AJ program would be familiar with the contortions it goes thru to process DMP=, SSJ= entry points. Such innovations have made systems programming easier but the system slower. For the sake of 1 out of 10,000(?) users who might need this feature, we should not permanently burden and enlarge frequently used programs like LFM, OBF, etc. These would have to check more words in control point area for each of the tens of thousands of times they are called in the hope that during one of those times a user might want them to do something.

Considering Barry Fox's problem, he can either make it a system routine (O.K. even though he writes in FORTRAN) or use the above suggested safeguards.

(N. Reddy 10/9/75)