The Cache Buffer

*

DECUS

This Is The First Subscription Service Issue of the RSTS/E Special Interest Group

Printed in the U.S.A.

The Cache Buffer *

Newsletter of the RSTS/E Special Interest Group

Vol. 10, No. 1 August 1983

SYMPOSIUM REPORT

Beginning on page 33

DECUS Scribe Project Successful

The articles that appear in the special Spring '83 Symposium Report section of this publication were produced through the DECUS Scribe Project. 20 students from the St. Louis area were hired to report on selected sessions. They produced documents using DECmate word processors. These documents were sent to the newsletter editors on machine readable media. I selected a number of articles that would be of interest to RSTS users. Many of them appear in this newsletter, and others were saved for use in future editions. Not all of the articles that I selected are about RSTS sessions. Some are from sessions from SIG's such as BASIC, RSX, Site Management and Training, and others.

The scribes must be commended for successfully completing a very difficult assignment. Although many of the scribes were data processing majors, it is still very difficult to report on very technical subjects that are unfamiliar. From my investigation of the articles, I have found them to be written very accurately and competently. Our thanks must go out also to Ralph Stammerjohn, whose efforts to keep the project running efficiently were monumental.

It is my hope that these articles prove very valuable not only for those who weren't able to attend the Symposium, but also to those who were there and had to miss sessions or were not able to take notes. I am looking forward to the work of the Scribe Project at future Symposia. *

DECUS Subscription Service Begins

This is the first issue of The Cache Buffer under the new DECUS Subscription Service. You probably have already noticed some changes in the newsletter. The paper and the cover are different, and we have eliminated the use of the reduced format (with two 8 1/2 X 11 pages reduced to fit on one page). The newsletter is also 3-hole punched for filing. There will be other changes in the future. These improvements are made possible by the funding coming from the subscriptions.

In return for your subscriptions, I have committed myself to improving my part in the newsletter. I have scheduled 6 issues during the first 12 months of the service. In order to accomplish this, I need help. I am only a volunteer, and have many obligations, both personal and professional. But I pledged my efforts toward the RSTS newsletter, keeping in mind that the newsletter is important to all RSTS users. An easy way to help is to urge other RSTS users to subscribe. Only through subscriptions can we continue to produce this newsletter. Please see "From the Editor" in this issue for more information on how you can help. *

RSTS/E SIG DECUS U.S. CHAPTER

DECUS

SIG Officers

SIG Chairman

Bruce L. Gaarder
Macalester College
16ØØ Grand Ave.
St. Paul, MN 551Ø5
612-696-6138

Symposia Coordinator

Scott Daily
Great Lakes Chemical Corp.
P.O. Box 22ØØ
West Lafayette, IN 479Ø6
317-463-2511

Newsletter Editor

Ray E. Gebbie
Guntert Sales
P.O. Box 1688
Stockton, CA 952Ø1
2Ø9-464-8712

Wishlist Coordinator

Jeff Killeen
Information Design & Management
45B N. Main St.
Sherborn, MA Ø177Ø
617-653-25ØØ

BASIC SIG Liaison

James Hodges
Merrell-National Labs
211Ø E. Galbraith Rd.
Cincinnati, OH 45215
513-948-9111 x2566

EDUSIG Liaison

George Wyncott
Purdue University
Mathematical Sciences Bldg.
West Lafayette, IN 497Ø7
317-494-1787

Small-Users Sub-SIG

Scott Pandorf
Kittle's
86ØØ Allisonville Rd
Indianapolis, IN 4625Ø
317-849-53ØØ

Library Coordinator

Mark Hartman
ImPrint Systems, Inc.
3Ø3 W. Katella, Suite 3Ø3
Orange, CA 92667
714-771-648Ø

INDENT Sub-SIG

Dean Goranson
Cincinatti Gear Co.
Cincinatti, OH 45227
513-271-77ØØ

Tape Copy

Carl Hauger
P.O. Box 4Ø52
Greenville, DE 198Ø7
3Ø2-655-3544

If there is anyone who was left out, or if there are any corrections, let me know as soon as possible, so that the next newsletter can be correct.

In This Issue

Coming in Future Issues:

* Preview of the Fall Symposium in Las Vegas
* More reports from the Spring Symposium
* Reprints of the DEC Handouts from St. Louis
* Introduction to Multi-Terminal Service
* More news and useful information

From the Editor

There seems to be some confusion among the DECUS membership as to what the Commercialism Policy allows and what it does not allow. My interpretation of the policy is that anything that is of a sales-oriented nature from any source is not allowed to be part of DECUS. This means that we can discuss non-DEC vendors if we keep it to a technical orientation.

In "The Cache Buffer", I consider it my duty to edit or reject any submissions which violate the policy. I extend the policy to DEC, along with non-DEC vendors. I feel that any material that has a sales orientation should not be published in this newsletter. Discussion of a product's internal features, or comparison of different aspects of multiple products is gladly accepted, as long as the above interpretation of the Commercialism Policy is followed.

On another subject, I must correct an omission from the Spring 1983 newsletter. My article entitled "RSTS Optimization Checklist" was derived mainly from the handout from the first "Dave and Carl Show", presented by Dave Mallery and Carl Marbach at the Fall '8Ø Symposium in San Diego. These 2 men have become very important in providing information to RSTS and DEC users, and my thanks goes out to them.

Now that "The Cache Buffer" is published bi-monthly, I am in great need of submissions. Without them, we cannot publish a newsletter. I have received a number of promises from people at Symposia that they would provide submissions. In most cases, the submissions are never received. I know that the pressures of work and other things easily push things like newsletter submissions out of a person's mind. But don't let them be forgotten forever! I especially welcome articles of a technical nature. There is no end to the desire by users for more information about their RSTS systems. Other kinds of articles are also welcome, including "War Stories", humor, programs, hints and kinks, product reviews, requests for information, etc. I also welcome your comments or suggestions. This is a newsletter for the RSTS SIG, so the input and participation of the SIG membership is very important. *

How to Submit to "The Cache Buffer"

I urge you to submit articles for the newsletter on a machine-readable medium. I will promptly return all media. The ideal medium for me is 8ØØ BPI tape, but I can handle with greater effort 16ØØ BPI tape or WPS 8 floppy. The format of the article can be RUNOFF, RNO, or plain text file. With machine-readable articles, I can print them myself and include headlines and page numbering and the print quality of the newsletter can be more uniform. But if you are unable to provide it using one of the above formats or media, <u>by all means send it anyhow.</u> For .RNO files, please use the following print settings:

```
    .PAGE SIZE 6Ø,8Ø
    .LEFT MARGIN 8
    .RIGHT MARGIN 72
    .SPACING 1
```

For other formats, use a page size of 6Ø lines by 65 characters, single spaced, one side of the paper only.

The deadlines for submissions to future newsletters is as follows:

ISSUE	DEADLINE
October 1983	Sept. 15, 1983
December 1983	Nov. 1, 1983
February 1984	Jan. 1, 1984
April 1984	Mar. 1, 1984
June 1984	May 1, 1984

Send your submissions to:

Ray Gebbie
RSTS Sig Newsletter Editor
Guntert Sales Div., Inc.
P.O. Box 1688
Stockton, CA 95207
209-464-8712

*

Letters to "The Cache Buffer"

Letters to The Cache Buffer should not be written on company letterhead, in order to avoid any question of the letterhead being commercial and violating the DECUS policy on commercialism. I reserve the right to reject any letters which I feel might be in violation of the policy on commercialism. My address is listed above.

Ray Gebbie
Editor, "The Cache Buffer"

*

Dear Editor:

In an effort to build better lines of communication with the developers of RMS, I am trying to start an RMS wish list. It would be under the auspices of the Data Management Systems Sig because that seems to me to be the most logical one; however, I would like to think of it as a cross-sig endeavor. If members of your SIG use RMS, I would appreciate it if you would print this letter in your newsletter.

Too many sig wishes are responded to with the answer, "Well, that's really an RMS problem.", and from there, the wish disappears. I'm sure that that is as frustrating to the developers of RMS as it is to the users.

A session is planned for Las Vegas to discuss existing RMS wishes, and to accept new ideas and wishes. If any member of any sig would like to send me input, my address is below.

Thank you.

[illegible]

Chuck Evans
Director of Data Processing
Times Publishing Company
Times Square
Erie, Pennsylvania 16534

Ray Gebbie
RSTS SIG Newsletter
Guntert Sales Div., Inc
P.O. Box 1688
Stockton, CA 95207

Dear Ray,

Many owners of 11/44 processors running RSTS have tried the built-in TU-58 tape unit once or twice and given up under a deluge of "Device hung or write-locked" errors. The problem is that 9600 baud is just too fast for RSTS. Some other monitor process runs with interrupts locked out long enough to make the DD: driver drop characters, leading to retries and eventually errors. This is very dependent on system load. Occaisionally, FIT would work, as long as I didn't use the /W (watch) switch. Apparently, the extra terminal I/O was enough to bomb the copy.

DEC's solution is evident: they are discontinuing support for the TU-58 after V8.0. This is tolerable for most users, after all, not even Field Service will use it if there is any alternative device.

We, however, develop code for TU-58 based stand-alone systems, and the necessity of copying software to floppy, and putting it on tape via RT11 (whose driver does work, even at 38.4 KB) got on my nerves. Luckily, a solution is available for most configurations. The 11/44 TU-58 port may be set for 9.6 KB, 38.4 KB (heaven forbid), or to be identical to the console baud rate. Our console is an LA120, so it was a simple matter to reset the tape for 1200 baud (one wire-wrap jumper), switch the TU-58 port to match the console (DIP switches on the M7096), and away we went (slowly, but it is a TU-58, after all...).

Since making the change, we have logged no DD: errors, no matter what the system load was. I have not experimented with any other baud rates, but the 11/44 and the TU-58 can both be configured for 600, 2400 and 4800 baud. Running the TU-58 at a slower console speed is possible, but it might not be practical (20 seconds per block at 300 baud).

Yours truly,

James Van Bokkelen

James Van Bokkelen
Manager, Software Development

PERCEPTION TECHNOLOGY CORPORATION
50 Shawmut Road • Canton • Massachusetts • 02021 • (617) 821-0320

APPLIED BUSINESS SERVICES, INC.
PO Box 417
Christiansburg, Va 24073
(703) 382-0596

May 9, 1983

Ray Gebbie
RSTS Sig Newsletter Editor
Guntert Sales Div.
PO Box 1688
Stockton, California 95207

Dear Ray:

Thanks for the article in the Spring '83 "Cache Buffer" on Disk Rebuilding. Your hints and techniques will be a big help!

I have an idea to help speed the process. After step "19" can you make a "SAVE" copy of the disc? Then, next time the disc needs rebuilding, skip the first 19 steps, "RESTORE" from your pack (or tape) on the shelf, and do only steps 20 and 21.

MY question - Will SAV/RES leave the pre-extended disc directories (from step 17) extended, even if those accounts don't have any files in them? I hope so!

Something else we do, here at ABS, which has been a tremendous help. Our "SY0:" is an RK05 disc. Sounds crazy doesn't it? A slow-poke thing like an RK05 being used as the system disc. But what a joy it is to completely isolate system functions (like swapping) from data functions. When we rebuild a disc, we don't worry about placement of SWAP, RTS, and SIL files, since they are on a device all their own.

We put only the necessaries on "SY0:". CUSPs like LOGIN, UTILTY, PIP, SYSTAT, SAVRES, as well as the RTS and SIL files. This way we can boot RSTS for various system management operations without envoking special procedures. No data goes on "SY0:" (we fill with a DUMMY file). Saving data is simple and straight forward in the (rare) event of a corrupted pack.

Also, since "SY0:" is on a controller of it's own, we effectively have "overlap seek" so that swapping does not have as determental effect as it would if the SWAP file shared a disc with data. Come to think of it, that slow old RK05 helps speed up system performance.

Again, thanks for the tips ...

Bob

Bob Ashcraft

System
Performance
House, Inc.

5522 Loch More Court · Dublin, Ohio 43017 · 614-265-7788

May 10, 1983

Mr. Ray Gebbie
RSTS SIG Newsletter Editor
Guntert Sales Div., Inc.
P.O. box 1688
Stockton, CA 95207
209-464-8712

Dear Mr. Gebbie,

In the spring 1983 issue of Cache Buffer, you had an article entitled "Disk Rebuilding Checklist" in which you solicited articles on well-structured disks. Enclosed is hardcopy of what I believe to be the definitive article on disk structuring for RSTS/E systems. The article is long and is somewhat self-serving, but if you wish to publish it, I can make it available to you on magtape in either WORD-11 or ASCII format.

I understand and appreciate DECUS's rules about non-DEC vendors, but it is time that somebody (not so biased as myself) told the user community that disk optimizers are here, they are cheap and they work! Articles such as yours have been floating around DECUS and elsewhere for years; they are fine accurate articles, but the procedures that they describe are far inferior to any of the popular disk structuring programs. Furthermore, the possibility of user error and wasted time is high.

So how about a good word for disk optimizers? There are at least three good ones and four about which I know very little. I would be happy to survey the market for you or help you however I can. There is too great a need for structured disks in the RSTS/E community.

Sincerely yours,

William R Davy

William R. Davy
President, SYSTEM PERFORMANCE HOUSE, INC.

RSTS/E Disk Optimization in a Multi-User Environment

William R. Davy
System Performance House, Inc.

Abstract

A great deal of existing literature addresses the important matter of RSTS/E disk optimization. This article expands beyond the conventional wisdom to describe previously unpublished optimizations available for multi-user RSTS/E systems. Included is a review of common disk optimization practices; some observations about the multi-user RSTS/E environment; and how these two interact.

I. THE STATE-OF-THE-ART

Most RSTS/E users are painfully aware that their systems tend to be disk bound. They perform far more disk accesses than are needed for mere data retrieval, program loading and swapping. Furthermore, disk seek time for these operations and others is longer than necessary. To minimize these problems, users are generally limited to the following methods.

* Center and pre-extend the UFD's. Some shops also pre-extend the MFD contiguously starting at device cluster one.

* Map files contiguously and give them the contiguous attribute where appropriate.

* Increase file clustersize so that the file is mapped in seven clusters or less, up to the maximum clustersize of 256.

* Center the swap files, run-time system files, etc. Tedious manual procedures generally limit these efforts to the few files which are perceived to be most used.

* Increase the pack cluster size to decrease FIP overhead.

* Run REORDR frequently.

* Use the "new files first" attribute on the disks.

* Allocate some free space near the center of the disk.

Two major focuses of this article will be to correct the misconceptions associated with the above steps and to describe other significant optimizations. It is assumed that the user is familiar with the RSTS/E file structure. There have been excellent descriptions thereof by Mike Mayfield[1], Scott Banks[2] et al.

THE RSTS/E ENVIRONMENT.

- Software Characteristics

RSTS/E systems are by definition multi-user systems: their performance problems arise under multi-user conditions. Consequently, our optimization efforts will focus on these.

The **fundamental consideration of RSTS/E disk optimization** is that **consecutive disk accesses to the same file, UFD or MFD are statistically rare in a multi-user RSTS/E system.**

For example, take a system whose file clustersize has been optimized and whose directories have been recently REORDR'ed. Consider what would happen if there were a number of jobs performing heavy I/O to disk files and a few jobs excercizing FIP through directory accesses etc. (i.e., a typical RSTS/E system). We would find enough idle time that each job would receive the CPU time to queue its next disk access well before the system could process the other pending requests. For most of the time, each job would have a pending request. No given job would be able to receive two consecutive disk accesses, the RSTS/E "fairness" algorithm notwithstanding. If each job accessed a different set of files, no file would ever receive two consecutive accesses.

What about accesses to MFD's and UFD's? It is fairly well known that FIP is single-threaded. That is, it will process any operation to completion before starting another. This guarantees that two jobs will never perform FIP operations simultaneously. Although some FIP operations require multiple UFD accesses (e.g. a file lookup in a large directory), there are other jobs which do file accesses without FIP. There will be file accesses in between the UFD accesses.

Furthermore, FIP always accesses exactly one MFD/UFD block at a time. Monitor statistics show that the number of directory accesses always equals the number of directory blocks transferred. The value of this observation will become apparent when we discuss UFD optimizations.

- Hardware Characteristics

DEC and third-party vendors offer a number of disk drives with different sizes and speeds, but they all have some common characteristics.

First, total time for completion of a disk read or write operation is equal to: SEEK TIME + ROTATIONAL LATENCY TIME + TRANSFER TIME.

Seek time across just one track is significant when compared to either rotational latency or transfer time. Seek time increases less than linearly as the number of tracks increases.

Rotational latency time is the time it takes for the disk to rotate the transferable data under the head(s), after the seek has been performed. The **fundamental consideration** discussed above shows us that it is nearly impossible to pre-determine the rotational state of the disk before any given operation. Therefore, except where special circumstances warrant otherwise, it is accurate to assume an average latency time equal to one-half the maximum latency time.

From a statistical standpoint, transfer time is small compared to the other two components. Consequently, our major hope of speeding up accesses lies in reducing seek time. This is accomplished by accessing blocks on the disk which are "close" together. For now, it will suffice to say that blocks which are close together on the disk have block numbers which are near each other, and vice versa.

THE CONVENTIONAL WISDOM RE-EXAMINED.

Now we are ready to re-examine all of the popular disk optimization methods, paying particular attention to their effect on multi-user systems.

A) Determining the disk center?

In many disk optimizing schemes, the UFD's, free space and certain files are centered. Some programs calculate the center of the disk to be the median numbered block. Others recognize that the entire disk may not be allocated, so that the center is better considered to be closer to the beginning of the disk, say perhaps one-third of the way from the beginning to the end.

These schemes are feeble attempts to guess the optimal "center" of the disk. A much better position can be calculated. Simply stated, the center would be the block number equal to half of the space needed for all of the files on the disk, plus some free space and the space needed for the UFD's and the MFD. An improvement upon this algorithm would be to subtract the size of all of the files not used during time sharing. (They would be placed at the end of the disk where they would not get in the way). The other files, the UFD's, the free space and the MFD would be placed on the disk, starting at the beginning. The calculated center would then be at the center of the active files. In **this article, the term center refers to this.**

B) PRE-EXTENDING THE MFD.

Some installations pre-extend the MFD. Presumably, this is done to make it contiguous. In a single-user system, this can speed up MFD searches, but in a multi-user system, it practically guarantees that MFD seeks will be as slow as possible. The MFD would be entirely contained at the edge of the disk. As the **fundamental consideration** shows, we are unlikely to get two consecutive accesses to the MFD. A much better strategy for optimizing the MFD is described later.

C) PLACING AND PRE-EXTENDING THE UFD'S.

The **fundamental consideration** shows that the main reason to pre-extend a UFD is not to make it contiguous, but to control its general position. **The proper position for UFD's is near the center.** The strategy of placing UFD's near their associated files may make some sense in a single-user system, but it is folly in a multi-user system. It will guarantee that UFD's - the most heavily accessed blocks on the disk - will be scattered as far apart as possible. It will also guarantee that if the most used files are in different accounts, they will also be scattered as far apart as possible, maximizing seek time.

The second question concerning UFD's is how much should they be extended. One strategy is to extend them to their maximum size, 112 blocks. On a system with 100 accounts, this requires 11,200 blocks of prime space on the disk, most of which will never be used.

Another strategy is to use the minimum amount necessary to hold the current directory information. This strategy is poor on two accounts. First, many systems add new files to their accounts without first deleting others. If the UFD is full, it will be extended, not necessarily in a central location. Second, when files are deleted and recreated as they are by editors, compilers, etc., they are often recreated with smaller clustersize which requires more UFD space. If the UFD was created just large enough to hold its previous contents, it will have to be extended to hold the expanded mapping information. So it is clear that some scratch space should be left in the UFD. The UFD optimization section will have more on this subject.

D) USING THE OPTIMUM FILE CLUSTERSIZE

Optimum file clustersize is generally considered to be the smallest clustersize which will map the file in seven clusters or less. If the file is larger than 256*7 blocks, the optimum clustersize would be 256, the maximum clustersize. The reason for this is that the "retrieval blockettes" in the UFD's each hold pointers to seven clusters. If the file has seven clusters or less, then the minimum number of blockettes are needed to map the file.

Optimum clustersize helps performance two ways. First, since only one retrieval blockette is kept by FIP at a time, fewer "window turns" will be performed to access the file. Second, by having fewer blockettes in the UFD, it will be more compact and cached more efficiently.

There are some disadvantages to using the optimum clustersize. The space allocated to the file is a multiple of the clustersize, regardless of what is actually used. This can increase disk usage up to 7 percent in a typical case, and up to 14 percent in certain pathological cases when compared to using the minimum clustersize.

Raising clustersize also makes it more difficult to place a file exactly where you want it. FIP always places files on cluster boundaries. In cases where it is desirable to pack files of different clustersize close together, with no free space in between, using optimum clustersize often prohibits placing a file exactly where you want it.

On certain files, it may actually be advantageous to lower clustersize below the "optimum" value so that the files may be placed on a otherwise unattainable boundary. It may also be worthwhile to save the allocated but unused space, that is wasted when using larger clustersize. Run-time system and swap files are good examples.

E) POSITIONING FILES

Positioning files can increase system throughput by decreasing seek time. One of the popular optimizations is to position files accessed by a particular program as close to each other as possible, if you cannot place them on separate drives. Here is another case where a single-user optimization causes degradation of performance. As we saw when discussing the **fundamental consideration,** a particular job is unlikely to get two consecutive accesses to the same disk. So what is the purpose of positioning the files? The only value comes from positioning them near the files being accessed by other jobs, which will also cause them to be positioned near each other.

From the above example, we begin to see that we must position files with consideration for all of the files on the disk. The most used files will be placed closest to the center; the least used files will be placed nearest the edges. This algorithm will be expanded into a powerful file positioning strategy.

F) MAKING FILES CONTIGUOUS

At this point, it is necessary to distinguish between the two types of contiguousness. Files whose block numbers are contiguous will be referred to as being mapped contiguously. In addition, there is a RSTS/E file attribute known as the contiguous attribute, which tells RSTS/E that the file is contiguous without FIP examining the mapping. This attribute is what causes PIP and DIR to list a file as being contiguous.

There are three main reasons for mapping files contiguously. First, the file is compactly placed on the disk so that large transfers can be made with one access. Second, the entire file can be placed where desired. Finally, in cases where the file need not be extended, it can be given the contiguous attribute to help reduce window turns (FIP overhead) on files larger than 256*7 blocks. Only on single-user systems is it necessarily true that head movement is minimized by making files contiguous.

G) PUTTING SOME FREE SPACE NEAR THE CENTER

Assuming that most systems create new or temporary files, free space becomes very active file space and should be positioned near the active files. Two questions arise concerning free space: where and how much?

Free space clearly should be positioned near the center of the disk where the most active files should also reside. Most programs, including DEC utilities, do not specify file position when they create new files. FIP searches for a file location when it is created or first extended by starting at the low numbered end of the disk until it finds contiguous space (if possible) for the specified clusters. The ramifications of this are important.

First, if there is free space between the beginning of the disk and the centered free space, it will be used. Therefore, centered free space will only be useful if there is no other free space below it - the low numbered end of the disk must be packed tightly with files.

Second, FIP has considerable overhead searching for the free space. If we have a choice between placing free space on the high or the low side of the center, we should place it on the low side so FIP will find it faster.

H) INCREASING PACK CLUSTERSIZE

Pack clustersize can be considered the minimum file clustersize. FIP maintains SATT.SYS, which is a bitmap for the pack clusters on the disk. It tells which pack clusters are in use. If those clusters are large, there are fewer of them and FIP can search the bitmap easier. Also, SATT.SYS will be smaller and there will be fewer accesses to it FIP only keeps one block of SATT.SYS in memory at a time.

Increasing pack clustersize also results in less variation in clustersize between files. The disk becomes less fragmented and it is easier to pack files tightly, eliminating free space in front of the centered free space mentioned in the previous section.

The disadvantage of increasing pack clustersize is that it tends to waste space. Note that on a pack with pack clustersize 16, even a one block file uses 16 blocks. All storage allocations are rounded up to a multiple of 16.

I) RUNNING REORDR FREQUENTLY

Using REORDR makes systems run faster. The UFD structure can be considered a tree, and REORDR allows FIP to traverse that tree more quickly. However, while there are many ways to organize MFD/UFD's, the one created by REORDR is almost never optimal.

POSITIONING FILES ON A MULTI-USER SYSTEM

Earlier, we defined the center of the disk. If we could somehow determine which blocks of which files/UFD's were accessed the most during a time sharing session, we could then position those files in their optimum static position by placing the most accessed blocks nearest the center.

There is no practical way to know exactly which files are used the most, but one can make some reasonable guesses by examining the nature of the system. What follows is a description of common files in what is likely close to their optimum order. That is, the ones mentioned first should be placed closest to the center.

A) ORDERING FILES AROUND THE CENTER

The swap files are frequently accessed on most systems. With the following exception, there is little reason to have more than one swap file on any given disk. The reason for having more than one swap file would be to optimize a system in which interactive jobs swap frequently with event driven jobs, the usual job count is much less than job maximum, and there is good reason to do all swapping on the same disk. Under these conditions, having SWAP3.SYS just in front of SWAP.SYS (and no other swap files) can decrease lost time by requiring slightly shorter seeks between the out-swaps and the in-swaps.

The files OVR.SYS, ERR.SYS, and BUFF.SYS are frequently accessed, if they are used. If OVR.SYS is not used, then the current SIL should be positioned near the center of the disk. Otherwise, it can go to the back edge with the other unused files.

The non-permanently resident run-time systems and resident libraries are frequently accessed. Note that in addition to being accessed at program load time, they are accessed on certain in-swaps. Under certain conditions, lost time can be decreased by positioning these files as near as possible to the swap files. Permanently resident run-time systems are not accessed once a time sharing session has started, so those files should be placed at the back edge of the disk.

MFD's and UFD's should be placed near the center, as will be discussed later.

SATT.SYS is accessed frequently on disks when files are being created or deleted. It should also be near the center.

Free space should be near the center as mentioned earlier.

Next should be frequently used files. Presumably, someone knows which files those are.

The remaining files should be positioned around the others in decreasing order of use, keeping in mind that the free space in the center will not be used until the free space toward the front of the disk is used.

B) IMPROVEMENTS OVER ORDERING AROUND THE CENTER

The above rules for positioning files are effective until the system is actually used. When files are created and deleted (e.g. when they are edited or rebuilt), the new versions end up where FIP puts them, and free space is created elsewhere on the disk. These new creations are not likely to be well placed. Consequently, several additional steps should be taken.

First, as a general rule, files which are likely to be deleted should be placed near the center. This will force the resulting free space to be in a good location. Furthermore, if there is sufficient free space, the files likely to be deleted, without being otherwise accessed, can be placed near the high numbered end of the disk. The free space generated when they are deleted will be used last; the well-positioned free space will be used first.

It may seem difficult to guess which files are likely to be deleted, but in reality it is quite easy. With few exceptions, **the most recently created files are the most likely to be deleted soon.** Think about it, keeping in mind that most editors etc. do not modify old versions of files, but create new ones, deleting the old ones when done.

C) DETERMINING THE MOST ACCESSED FILES

With the large file processor, it is relatively easy to scan the list of open files. One might conclude that monitoring this list for the most commonly opened files would reveal the most heavily accessed files. Unfortunately, such attempts will produce poor results.

Consider a typical system which runs error logging, the spooling package with batch processors, and uses the CUSP's frequently. Error logging leaves the error file open continuously and only accesses it in the event of errors. The spooling leaves a number of work files open and accesses them only occasionally. To load CUSP's, the system must open them, read them, and close them. However, all this happens so quickly that the monitoring program is quite likely to never see it happen (unless it is absolutely hogging the system). So any likely monitor program would guess the relative frequency of accesses to those files exactly backwards.

There is a large difference between a file being open and being accessed. Accesses are difficult to monitor without large overhead. Fortunately, some common sense observations can help determine which files are likely to be accessed most often.

Compiled basic programs and RT11 and RSX task images, can be loaded into memory with just one file access, provided that they are not overlaid. However, the ones that are overlaid (including PIP.SAV, EDT.TSK, and TKB.TSK), can be accessed many times in the course of one run. TSK files larger than 115 blocks and SAV files much larger than their core images are also overlaid. There is considerable sentiment that suggests that LOGIN should be well positioned because of its frequent use, but consider that in the course of one task build, more accesses are made to TKB.TSK than to LOGIN all day long. The same phenomenon is seen with PIP, EDT, ED2, and LBR.

Files which are accessed by EDT or compilers are typically accessed one block at a time. This implies that even if the file is only opened once, it will be accessed many times. Note that any file with a source file extension likely falls into this category. Data files, especially randomly accessed ones,are typically accessed a few blocks at a time. Object libraries, if they are used, are accessed many times per use. LOG files are not likely to be accessed more than once, sequentially, and possibly many blocks at a time (as PIP would access it).

Programs which access particular files each time they are run are most likely accessed less often than the files they access, especially if they do many accesses to those files.

D) POSITIONING ACROSS CYLINDER BOUNDARIES

Discussions about disk cylinder boundaries are rare in PDP-11 operating systems. On RSTS/E systems, the system takes care of mapping virtual blocks onto physical blocks on the disk without help from the programmer. For the most part, little is gained by positioning files across the minimum number of cylinder boundaries.

Most files on most systems are accessed one or a few blocks at a time. The blocks transferred on any given access are unlikely to cross a cylinder boundary regardless of how files are placed. The **fundamental consideration** shows there is little to be gained from positioning such files across minimum cylinder boundaries, since they are unlikely to be accessed twice in succession.

There are disadvantages in trying to avoid crossing cylinder boundaries. Due to FIP's alignment algorithm, cylinder boundaries always straddle clusters for any clustersize greater than one. On any large disk and on any optimized disk, the pack clustersize will be greater than one. An attempt to have all files straddle minimum cylinder boundaries would create free space at many cylinder boundaries. The degradation from scattering free space would more than make up for any gains from avoiding cylinder boundaries. Furthermore, attempts to position files with large clustersize across minimum cylinder boundaries tends to position them far from where they are desired. (Explanation of this effect is beyond the scope of this article; the mathematically inclined are referred to the Chinese Remainder Theorem.)

There are benefits, though, to positioning the most useful files. Non-permanently resident run-time systems are ideal candidates for minimum-cylinder positioning. If possible, the entire file is transferred in one access. Since there are just a few of these files, it is worthwhile to straddle the minimum number of boundaries with these, if they are used often.

Similarly, frequently accessed BAC's are good candidates for intra-cylinder positioning. Notice that on large disks, most of these files will miss cylinder boundaries anyway, regardless of the positioning algorithm. Ditto for unoverlaid TSK's and SAV's.

Overlaid task images are another story. They tend to be larger (harder to position), and they are accessed a smaller number of blocks at a time, so they tend not to be accessed across cylinder boundaries. The benefit is small compared to the effort and other resulting degradations.

The swap files are large and will likely cross many cylinder boundaries regardless of where they are placed. Optimum placement can at best avoid one cylinder boundary. Once again, the benefit is very small compared to the harm resulting from the likely off-center placement.

The last "files" to be considered are the MFD's and UFD's. They are always accessed one block at a time, so no single access crosses a cylinder boundary. The **fundamental consideration** shows that it is unlikely to access the same MFD/UFD twice in succession, so there is almost nothing to be gained by avoiding cylinder boundaries with MFD's/UFD's.

OPTIMIZING MFD'S AND UFD'S

UFD's are divided into chunks of eight words called blockettes. There are four different types of blockettes: name, accounting, attribute, and retrieval blockettes. Blockettes are linked together by pointers and except for the requirement that they start on an eight-word boundary, can reside anywhere in the UFD. Any blockette whose first two words are zero is considered unused (free space in the UFD).

The name blockettes contain the name and extension of the file in RADIX-50. The first blockette in a UFD contains a pointer to the name blockette of the first file in the account. Each name blockette contains a pointer to the name blockette of the next file in the UFD. When FIP searches a UFD for a file, it starts with the first blockette of the UFD and follows this chain until it finds the desired file or exhausts the list.

Each name blockette has a pointer to the accounting blockette for its file. This blockette contains the last access date, the number of blocks in the file, the creation date and time, the associated run-time system, and the file clustersize.

Each name blockette also has a pointer to the first attributes blockette if the file has any attributes. The attributes blockettes contain file attributes (up to seven per blockette), plus a pointer to the next attributes blockette for the file. There is a maximum of two attributes blockettes per file.

Finally, each name blockette contains a pointer to the first retrieval blockette. Each retrieval blockette contains the starting device cluster numbers for up to the next seven clusters of the file, plus a pointer to the next retrieval blockette, if it exists.

In a sense, the internal structure of the UFD's is a tree: the name blockettes form the root of the tree, the retrieval and accounting blockettes form branches, and the attributes blockettes are leaves off of the accounting blockettes. This tree can be searched in the forward direction, this is, from the root to the leaves, but not backwards. Since all of the nodes of the tree are found only by following pointers, the nodes (blockettes) can be located in any order in the UFD. However, we will see that some orders are better than others.

A) STRAIGHT FILE COPY

When files are copied into an empty UFD one at a time, their associated UFD blockettes are tightly packed into the UFD, one after another, starting at the first available location in the UFD. To scan the list of name blockettes (i.e., do a file lookup), FIP reads through the UFD sequentially, until it gets to the name blockette it is seeking. Straight file copy guarantees that FIP will only read a block of the UFD once when performing a file lookup.

It is a common belief that this system is efficient. It is much better than the tangled mess that results from creating and deleting files at random. Furthermore, most of the disk structuring utilities leave their UFD's this way. Unfortunately, except for very small directories and a few pathological cases, straight file copy is never best.

B) REORDR

The other well-known UFD optimizing technique is REORDR's algorithm. REORDR puts all of the name blockettes in the requested order starting at the beginning of the UFD. The remainder of the last block of the UFD used for storing name blockettes is left empty. REORDR writes the accounting, retrieval, and attributes blockettes starting in the next block of the UFD. When possible, REORDR writes all of the non-name blockettes in the same block of the UFD so that FIP only needs the minimum number of disk accesses to retrieve them.

For large directories, the above strategy is far superior to straight file copy. To open the last file in a UFD created through straight file copy, every block in use in the UFD must be read (with the possible exception of the last block). To open the last file in a UFD created by REORDR, only the blocks containing name blockettes (usually one-third of the blocks or less) must be read to find the last name blockette. One other block - the one with the accounting, attributes, and the first retrieval blockette - also must be read. So in large directories created by REORDR, approximately one-third the UFD accesses are needed to open files as in directories created by straight file copy.

Unfortunately, REORDR'ed directories are not always optimal. Consider the case in which there are just a few small files in a UFD. If the UFD was created by a straight file copy, all of the directory information is in the first block of the UFD and can be retrieved with one access. If the UFD is REORDR'ed, the name blockettes will be in the first block, and the rest of the directory information will be in the second block, requiring **two** accesses to open any file!

C) DOPTER'ed UFD's

Up to this point, we have hinted that there are a number of improvements to the UFD structure possible. In developing DOPTER, the System Performance House combined the best attributes of straight file copy and REORDR, and added some new optimizations.

DOPTER has an internal routine somewhat similar to REORDR. If all the UFD information will fit into one block, it is put into the first block of the UFD. If there are more blockettes than will fit into one block, the name blockettes are separated from the others as in REORDR, but with some interesting differences.

The most obvious way to reduce the overhead of file lookups is to have the desired files as near as possible to the beginning of the name blockette list. While REORDR makes a good attempt at this with its reverse order sort on creation or access date, it certainly does not do nearly as well as a good heuristic algorithm which will accept help from the user. (Note that sorting in alphabetical order on file name or extension is basically useless for optimization.)

Since DOPTER is very good at placing the most used files at the front of the UFD's, DOPTER leaves room in the first block of the UFD for all of the blockettes of the first two files. Thus, the most used files can be opened with only one UFD access, a 50 percent improvement over REORDR. Except for this improvement, all of the name blockettes remain segregated from the other blockettes.

To appreciate DOPTER's most significant attribute, it is necessary to understand FIP's method of allocating free space in UFD's. When FIP needs free space, (i.e., when it is creating a new file or extending an old one), it first searches the current UFD block in memory for free space. If it finds what it needs there, FIP uses it. Otherwise, FIP searches the UFD sequentially from the beginning, examining each blockette to see if it is in use.

When FIP finds its current block full, it will search every allocated blockette before it finds a free one, if the UFD was created by straight file copy. If the UFD was created by REORDR, it will search through all of the name blockettes until it comes to the small amount of free space left at the end of the name blockettes (if any). If that space has been used (which it will be after a few new files are created), FIP must search to the end of the allocated blockettes, just as in straight file copy.

The solution is to leave some free space near the beginning of the UFD. Since FIP only accesses one block at a time in the UFD's, there is little to be lost with this strategy. DOPTER leaves blocks two through nine empty. Thus, FIP finds free space with a minimum number of reads. In a large UFD, this strategy can cut file open overhead by 80 percent.

There is another important benefit from leaving free space near the start of the UFD. In a typical UFD, files are both created and deleted in somewhat random order. If an otherwise tightly packed UFD has a few files deleted, there will likely be a few free blockettes scattered throughout the UFD. When FIP reallocates the space, it scatters the blockettes throughout the UFD, causing FIP to perform many disk accesses to do file lookups, creations, etc. FIP generates what is commonly referred to as a tangled directory.

When free space is left at the front of the UFD, files still are deleted wherever they are. However, when new files are created, they are built as they would be by straight file copy. As we have seen, straight file copy is not best, but it is far better than the alternative.

D) DOPTER'ed MFD's

MFD's are similar in structure to UFD's, except that the name blockette entries may be for UFD's as well as files. However, MFD's provide special opportunities for optimization.

The first cluster of an MFD must be at device cluster one. MFD's are normally extended contiguously. However, this is a poor strategy because it guarantees that some of the most frequently accessed blocks on the disk are at the very edge. There are several ways to improve the situation.

An MFD may be pre-extended anywhere. That is, clusters two through seven may be placed on any cluster boundary, preferably near the center. Not only does DOPTER place the other clusters near the center, but it only uses the first block of the first cluster. Thus, instead of building the entire MFD at the edge of the disk, it only uses one block there. All the others are where they belong. Once again, the **fundamental consideration** shows us that we have lost very little by removing the contiguity of the MFD because consecutive accesses are seldom made to an MFD. (Note: all blocks of the MFD are available if they are needed, but that is seldom necessary.)

E) Additional MFD/UFD Optimizations

Many files have associated attributes which are unused. The prime examples are task images created by the task builder. The task builder uses RMS I/O to create its task images, but the attributes are never used. EDT Version 1 put attributes on its output files, but in the case of most source files, they were unneeded by the compilers or editors. The unneeded attributes take up space in the UFD's and create extra overhead for FIP. DOPTER effectively eliminates supernumerary attributes by recognizing source file and task image extensions. The same can be done with PIP.

Another UFD optimization can be made with files with the contiguous attribute. Once such a file is opened, RSTS/E knows the mapping for the entire file from having read the first retrieval blockette. UFD accesses can be minimized by putting the rest of the retrieval blockettes for large contiguous files at the very end of the UFD, away from the useful information in the UFD. In this manner, they will never be accessed, not even when searching for free space. This strategy assumes that there is sufficient free space in the UFD, which is almost never a problem.

OTHER TOPICS

A)DLA versus DLW

When initializing a RSTS/E volume, RSTS/E can be made to keep either the date of last access (DLA) or the date of last write (DLW). In addition to the arbitrary needs of the installation, there are several trade-offs when considering one against the other.

The date of last access/write is stored in the accounting blockette. If the DLW option is chosen, this blockette only needs to be rewritten when a file is modified. In the case of read-only files, of which executable programs are a major example, a physical UFD access can be saved by using DLW instead of DLA. Note that all writes to UFD's generate physical accesses - they are not cached.

On the other hand, if REORDR is used to sort the files in the UFD in reverse order by access date, a better order is likely to occur if DLA is used instead of DLW. If DLW is used, some frequently accessed programs/files can have very old last access dates and thus be put at the end of the UFD. With DLA, they would be put near the front.

On balance, DLW is better with all small directories because only a block or two will ever be read to search the name blockette list for any file. The savings from not writing the last access date on read-only files will more than compensate for the slightly longer lookup path after running REORDR. DLA is better only on large accounts where heavily used read-only programs/files will be placed at the front of the UFD by REORDR.

B) New Files First

It is easy to overrate the advantages of using New Files First (NFF). The arguments for using NFF are as follows.

1). It is convenient to have the most recently created files at the front of the directory listing. (This has nothing to do with performance.)

2). Recently created files are frequently accessed, so putting them at the front of the list decreases open time.

Closer inspection shows these arguments against using NFF.

1). In accounts with more than 31 files, it requires an extra UFD access (a physical write access), to create a new file. All of the name blockettes must be read to make sure that the file does not already exist. Then the pointers in both the first and the last name blockette must be rewritten. If NFF is not used, only the last name blockette must be rewritten. Note too, that these are physical accesses to disk that are not cached.

2). Recently created files are very likely to be deleted, resulting in two side effects. First, two name blockettes must be rewritten when the file is deleted, and they are likely to be in two separate UFD blocks. Without NFF, it is likely that they will be in the same UFD block. Second, in many environments such as word processing and development systems. Most files are created, opened only once for reading, and then deleted. The main advantage of NFF, quick file opens, is minimized.

BENCHMARKS

Typical RSTS/E systems are difficult to objectively measure. System load varies from minute to minute and hour to hour. Monitor statistics are easy to gather but difficult to interpret (e.g., what does it mean if directory accesses decrease: less activity, a slower system, or a more efficient UFD structure?). One way to eliminate uncontrollable variables is to run a single job on an otherwise unused system and measure wall clock time. Unfortunately, single-user benchmarks are poor indicators of multi-user performance.

At the System Performance House, we have developed a set of 12 programs which run simultaneously, simulating a multi-user environment. They are heavily disk I/O bound programs which perform a mix of file creations, opens, closes, lookups, logins, logouts, swaps, run-time system loads, and random file accesses. The programs run for a fixed length of time and measure the number of each type of operation they can perform during the allotted time. Although they overstate the true performance differences from changes in disk I/O efficiency, they are very sensitive to small performance changes.

These benchmark programs vividly demonstrate the performance gains generated by the optimizations described above. The benchmark programs showed 50 percent more throughput on this arrangement than they did on the identical system generated by another commercially available "disk structuring" program. This increase was due to both faster disk accesses and fewer directory accesses.

REFERENCES

1. Banks, Scott. "RSTS Disk Directories", RSTS Professional, Vol. 1, No. 1; Vol. 2, No. 1; Vol. 2, No.3; and Vol. 2, No. 4.

2. Mayfield, Mike. "RSTS/E Disk Internal Structures", Proceedings of the Digital Equipment Computer Users Society, April, 1978.

As editor of "The Cache Buffer", I would like to thank Mr. Davy for the preceeding article. It shows great thought and effort on a very important subject. I would like to solicit other articles on RSTS disk structure and optimization. I would welcome descriptions of non-DEC products, especially if they include figures before and after which can demonstrate the benefits of disk re-structuring. Comments about the V8.Ø structure (called "RDS1") are extremely useful at this time. Remember, that if your submission covers non-DEC products please follow my non-Commercialism guidelines as described in "From the Editor", earlier in this newsletter.

To demonstrate the need for information on the new disk structure, I would like to express a "War Story". About 5 days after converting our data pack to the new structure (using the convert utility under V8.Ø), we started to back up the disk when BACKUP told us that the disk was corrupted! We were greatly upset with this revelation. We were able to access files throughout the disk, yet we were told that the disk was corrupt. I suggested that we dismount and re-mount the disk, but this did not change the message from BACKUP. Also, we could not access anything on the disk now.

My choice at this point was to restore the disk from the previous night's BACKUP (and having all the work for the day re-done), or let my expert programmer attempt to re-build the directory structure. Since I had just returned from St. Louis with wonderful describing the new disk structure, I decided to let him embark on a directory re-build.

We discovered that we had only lost the directory to [Ø,1], but since SATT.SYS is located there, the disk could not be accessed. I guess that we were able to access it earlier because the SATT was in memory. We learned a lot about the new structure in searching through disk blocks, and we re-connected the [Ø,1] directory. But we still could not access the disk. We throughly checked the directory structure, but even though all looked good, we were still not successful.

After a number of hours of messing with it, and with complaints coming in that people were getting behind on current work, not to mention the work that had to be re-done, I decided to restore from the last BACKUP tape.

We went to twice daily backups as a protection against future problems, but to my great relief, the problem has not returned.

--Ray Gebbie

Weinberg's Second Law

If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization.

BASIC-PLUS II Solution?

Ray Gebbie
Guntert Sales Div., Inc.
Stockton, CA

We received version 2.Ø of BASIC-PLUS-II with great expectations. But as we all discovered, there were a few problems with the release. The new version simply did not work correctly in a number of areas. We also discovered that there were numerous syntax errors when we tried to compile our old programs, and the compile time for the new version is so long as to make a reasonable conversion impossible.

The BASIC developers were not ignoring these problems. They produced version 2.1, which fixed most of the bugs. There are a few minor problems, but the new version works very well. A number of the new features of version 2.1 are very nice, such as the ability to do proper structured programming, the compiler directives, etc. But the compile times are extremely long. There is no easy way to convert a full system of programs without locking out all users for many days. We were also put in a CATCH 22 situation by the fact that version 8.Ø of RSTS did not support v1.8 of RMS with resident libraries, so we could not go to version 8.Ø and stay with the old version of BASIC. We could not convert to the new version because of compile times, and we could not stay with the old version.

The answer to the problem is to be able to use the old version of the RMS resident libraries under V8.Ø of RSTS. By changing the names of the old RMS libraries and patching the tasks to access the libraries using the new names, we can continue to run the old tasks under the latest version of RSTS, and convert to the new version of BASIC over a longer period of time. We can use the old and new versions of BASIC at the same time. The only disadvantage is having to keep old and new versions of the resident libraries in memory. (We make the new libraries non-resident, until we have a number of programs converted to the new version.)

Two of the programs which follow were provided by Bruce Gaarder of Macalester College in St. Paul, MN. The third program is a modification by us of the first program. These programs produce ATPK command files that patch tasks to refer to the renamed libraries. After patching the tasks, rename the libraries using PIP and install them. In order to run both versions of BASIC at the same time you need to change the LB: account to another account before building the new version. This will insure that the old files are not deleted. Your ODL and CMD files need to refer to the proper accounts for the version that you wish to use.

NOTE

These procedures are not supported by DIGITAL, or by the authors of the programs. But they have worked with no problems at a number of installations. I welcome any comments or suggestions concerning the procedures. My address appears at the end of the article.

1st Program -- patching for the RMS resident library

```
1       ! *** RMSV18.B2S &
        ! Program to convert tasks to use RMSV18 (instead of RMSRES) &
        ! Input to the program is a directory file created by PIP &
        ! using a command such as PIP TASK.DIR=[*,*]*.TSK/DI:NA:EX &
        ! The output is called V18.CMD and should be submitted to &
        ! ATPK using the command ATP V18.  This command file runs &
        ! $ONLPAT to patch the tasks. &
        !
9       DIM A%(512%)
10      PRINT "Enter input file name"; &
        \ INPUT LINE I$ &
        \ I$=EDIT$(I$,4%) &
        \ OPEN I$ FOR INPUT AS FILE 1% &
        \ OPEN "V18.CMD" FOR OUTPUT AS FILE 2% &
        \ PRINT #2%, "RUN $ONLPAT" &
        \ ON ERROR GO TO 1000
20      INPUT LINE #1%, A$ &
        ! These lines may have to be modified to properly parse the &
        ! directory file produced by PIP on your system (i.e. SY: vs. &
        ! DR0:) &
        \ GO TO 20 IF MID(A$,3%,1%)<>":" AND MID(A$,7%,1%)<>"." &
        \ IF MID(A$,3%,1%)=":" THEN &
                ACCT$=EDIT$(A$,4%) &
                \ GO TO 20
30      GO TO 20 IF LEFT$(A$,1%)=" " &
        \ FIL$=LEFT$(A$,10%) &
        \ OPEN ACCT$+FIL$ FOR INPUT AS FILE 3% &
        \ FIELD #3%, 512% AS R$ &
        \ N%=0%
40      GET #3% &
        \ N%=N%+1% &
        \ CHANGE R$ TO A% &
        \ FOR I%=1% TO 509% STEP 2% &
                \ R.1%=A%(I%)+SWAP%(A%(I%+1%)) &
                \ R.2%=A%(I%+2%)+SWAP%(A%(I%+3%)) &
                \ IF R.1%=29339% AND R.2%=29019% THEN &
                        PRINT #2% &
                        \ PRINT #2%, ACCT$;FIL$ &
                        \ PRINT #2%, NUM1$(N%-1%);".*512." &
                        \ PRINT #2%, NUM1$(I%+1%);"." &
                        \ PRINT #2%, "107176" &
                        \ PRINT #2%, "Z" &
                        \ PRINT #2%, "Z" &
                        \ PRINT #2%, "Z" &
                        \ PRINT #2%, "Z"
50      NEXT I% &
        \ GO TO 40
60      CLOSE 3% &
        \ GO TO 20
```

```
1000    RESUME 20 IF ERR=5% &
        \ IF ERR<>11% THEN &
                ON ERROR GO TO 0 &
          ELSE  RESUME 60 IF ERL=40% &
                \ RESUME 1010
1010    CLOSE 1%,2%
32767   END
```

2nd Program -- patching for the BASIC resident library

```
1       ! *** BP2V16.B2S &
        ! Program to convert tasks to use BP2V16 (instead of BP2RES) &
        ! Input to the program is a directory file created by PIP &
        ! using a command such as PIP TASK.DIR=[*,*]*.TSK/DI:NA:EX &
        ! The output is called V16.CMD and should be submitted to &
        ! ATPK using the command ATP V16.  This command file runs &
        ! $ONLPAT to patch the tasks. &
        !
9       DIM A%(512%)
10      PRINT "Enter input file name"; &
        \ INPUT LINE I$ &
        \ I$=EDIT$(I$,4%) &
        \ OPEN I$ FOR INPUT AS FILE 1% &
        \ OPEN "V16.CMD" FOR OUTPUT AS FILE 2% &
        \ PRINT #2%, "RUN $ONLPAT" &
        \ ON ERROR GO TO 1000
20      INPUT LINE #1%, A$ &
        ! These lines may have to be modified to properly parse the &
        ! directory file produced by PIP on your system (i.e. SY: vs. &
        ! DR0:) &
        \ GO TO 20 IF MID(A$,3%,1%)<>":" AND MID(A$,7%,1%)<>"." &
        \ IF MID(A$,3%,1%)=":" THEN &
                ACCT$=EDIT$(A$,4%) &
                \ GO TO 20
30      GO TO 20 IF LEFT$(A$,1%)=" " &
        \ FIL$=LEFT$(A$,10%) &
        \ OPEN ACCT$+FIL$ FOR INPUT AS FILE 3% &
        \ FIELD #3%, 512% AS R$ &
        \ N%=0%
40      GET #3% &
        \ N%=N%+1% &
        \ CHANGE R$ TO A% &
        \ FOR I%=1% TO 509% STEP 2% &
                \ R.1%=A%(I%)+SWAP%(A%(I%+1%)) &
                \ R.2%=A%(I%+2%)+SWAP%(A%(I%+3%)) &
                \ IF R.1%=3872% AND R.2%=29019% THEN &
                        PRINT #2% &
                        \ PRINT #2%, ACCT$;FIL$ &
                        \ PRINT #2%, NUM1$(N%-1%);".*512." &
                        \ PRINT #2%, NUM1$(I%+1%);"." &
                        \ PRINT #2%, "107174" &
                        \ PRINT #2%, "Z" &
```

```
                           \ PRINT #2%, "Z" &
                           \ PRINT #2%, "Z" &
                           \ PRINT #2%, "Z"
50      NEXT I% &
        \ GO TO 40
60      CLOSE 3% &
        \ GO TO 20
1000    RESUME 20 IF ERR=5% &
        \ IF ERR<>11% THEN &
                ON ERROR GO TO 0 &
          ELSE  RESUME 60 IF ERL=40% &
                \ RESUME 1010
1010    CLOSE 1%,2%
32767   END
```

3rd Program -- patching of single programs for the RMS resident library

```
1       ! *** RMSPAT.B2S &
        ! Program to convert tasks to use RMSV18 (instead of RMSRES) &
        ! Input task names to be patched (type <RETURN> to exit). &
        ! The output is called V18.CMD and should be submitted to &
        ! ATPK using the command ATP V18.  This command file runs &
        ! $ONLPAT to patch the tasks. &
        !
9       DIM A%(512%)
10      ON ERROR GO TO 1000 &
        \ OPEN "V18.CMD" FOR OUTPUT AS FILE 2% &
        \ PRINT #2%, "RUN $ONLPAT"
20      PRINT "Enter task name"; &
        \ INPUT LINE FIL$ &
        \ FIL$=EDIT$(FIL$,4%) &
        \ GO TO 1010 IF FIL$="" &
        \ OPEN FIL$ FOR INPUT AS FILE 3% &
        \ FIELD #3%, 512% AS R$ &
        \ N%=0%
40      GET #3% &
        \ N%=N%+1% &
        \ CHANGE R$ TO A% &
        \ FOR I%=1% TO 509% STEP 2% &
                \ R.1%=A%(I%)+SWAP%(A%(I%+1%)) &
                \ R.2%=A%(I%+2%)+SWAP%(A%(I%+3%)) &
                \ IF R.1%=29339% AND R.2%=29019% THEN &
                        PRINT #2% &
                        \ PRINT #2%, FIL$ &
                        \ PRINT #2%, NUM1$(N%-1%);".*512." &
                        \ PRINT #2%, NUM1$(I%+1%);"." &
                        \ PRINT #2%, "107176" &
                        \ PRINT #2%, "Z" &
                        \ PRINT #2%, "Z" &
                        \ PRINT #2%, "Z" &
                        \ PRINT #2%, "Z"
50      NEXT I% &
```

```
        \ GO TO 4Ø
6Ø      CLOSE 3% &
        \ GO TO 2Ø
1ØØØ    IF ERR=11% AND ERL=4Ø% THEN &
                RESUME 6Ø &
        ELSE    ON ERROR GO TO Ø
1Ø1Ø    CLOSE 2%
32767   END
```

A similar program to patch for the BP2 libraries could be made from BP2V16.B2S.

Ray Gebbie
D.P. Manager
Guntert Sales Div., Inc.
P.O. Box 1688
Stockton, CA 952Ø1
2Ø9-464-8712

*

BACDIR Patch

Ray Gebbie
Guntert Sales Div., Inc.
Stockton, CA

The following CPATCH command file will patch BACDIR to open files with a recordsize of 518%*4%, which greatly speeds up the searching of disk directories before beginning the BACKUP file transfers.

```
RUN $CPATCH<cr>
<CPATCH's header line>

File to patch - BACDIR.BAS=BACDIR.BAS<cr>
#[logfile=]KB:<cr>
*H/OPEN W$/V<cr>
<tab>\ OPEN W$ FOR INPUT AS FILE 1% &<cr>
*G/1%/I/, RECORDSIZE 518%*4%/V<cr>
<tab>\ OPEN W$ FOR INPUT AS FILE 1%, RECORDSIZE 518%*4% &<cr>
*H/OPEN W$/V<cr>
<tab>\ OPEN W$ FOR INPUT AS FILE 2% &<cr>
*G/2%/I/, RECORDSIZE 518%*4%/V<cr>
<tab>\ OPEN W$ FOR INPUT AS FILE 2%, RECORDSIZE 518%*4% &<cr>
*H/OPEN V$/V<cr>
<tab>\<tab>OPEN V$ FOR INPUT AS FILE M% IF D%<>1% &<cr>
*G/M%/I/, RECORDSIZE 518%*4%/V<cr>
<tab>\<tab>OPEN V$ FOR INPUT AS FILE M%, RECORDSIZE 518%*4% IF D%<>1% &<cr>
*EX<cr>
Patch from KB:[P,PN]CPATCH.CMD complete
#^Z
File to patch - ^Z
```

Use your normal procedures to compile the program. *

Spring '83 Symposium Report

Symposium Summary

Ray Gebbie
Guntert Sales Div., Inc.
Stockton, CA

The St. Louis symposium was very successful. Although the attendence did not match the record set at Anaheim last December, there were over 4000 attendees. Due to the hotel room shortage, some of us had to commute from far parts of the city. The facilities in the Cervantes Convention Center were excellent. There were a large number of meeting rooms in one location. There was lots of space and it never felt crowded. The food was generally very good, and the service was fast (except for the deli stands on Monday and Thursday). The weather was good, although early arrivals on Saturday night were treated with a thunderstorm with heavy rain.

There were a couple of sessions that brought out heated response from the attendees. DEC announced that they are, in effect, getting out of the large systems business (10's and 20's). As you might expect, this caused great concern among the large systems users as to future support. Closer to the RSTS users was our great concern over the problems with the V2.0 release of BASIC-PLUS-II. The developers had to put up with a lot of verbal abuse from unhappy users. (See the article entitled "Basic-Plus-II Solution?" in this newsletter for more on this subject.)

I find that after attending 9 symposia, I still learn many things. The amount of useful information I received in St. Louis was even more than usual. I definitely feel that the investment that an installation must make in order to send someone to a DECUS Symposium is returned many times over in information that can be used to better run its computer systems. I have received more helpful information from DECUS than from any other source. *

Spooling on RSTS Systems

Paul Laba
Digital Equipment Corporation
Merrimack, NH

Scott Daily, Session Chairperson
Great Lakes Chemical Corporation
West Lafayette, IN

Reported by Marty Olevitch
DECUS Scribe Service

Paul Laba, of DIGITAL's RSTS/E Development Group, described features of the new Micro RSTS V8.0 Spooling Package recently announced by DIGITAL, and compared it with the older, slower package it will eventually replace.

The new package runs as a single detached job in 13K words of memory (plus 8K words using RMS resident library) or in 22K words using the overlaid version. Since it is written entirely in MACRO, it runs significantly faster than the current spooling package. It supports concurrent printing on up to four output devices (line printers or keyboard devices), and will print files of any RMS format, as well as standard stream ASCII files.

A single interface program is

used for both operator and user commands, with syntax fully supported in DCL. All user and operator commands are acknowledged, with improved error reporting and better handshaking between devices and processes. A single RMS indexed file for queued jobs and internal work entries is used for improved queue management. The package provides a Forms Definition File for maintaining all necessary form attributes. It is supported from the existing UU.SPL monitor directive. The user interface program can be accessed via DCL, a CCL, or a RUN command. The standard spooling package may be run concurrently with the new one.

Several user commands which do not require a privileged account are available. PRINT is used to submit one or more disk files to be printed. It may be qualified with regard to the priority, the number of copies, the forms to use, and a deadline. DELETE will cancel one or more print jobs on the queue by a job specification. DELETE/ENTRY is used to cancel jobs by entry (job) number. SHOW QUEUE will display either currently running jobs or jobs waiting to run. A job to be displayed may be selected by queue name, owner name (PPN), job name, or form name.

Operator commands can only be issued from a privileged account. The commands are as follows: START/QUEUE/MANAGER to start up the spooling services package. It causes a single detached job to be spawned. STOP/QUEUE/MANAGER will shut down the package, either immediately or at the completion of all active jobs. INITIALIZE/PRINTER defines a device for printing. DELETE/PRINTER removes a printer from the list of defined print devices. STOP/PRINTER halts printing on a specified printer, either immediately, at the end of the current file, or at the end of the current job. START/PRINTER causes printing to resume following a STOP/PRINTER command and supports several options.

Because the package is not quite complete, there are several restrictions on current use. Although batch processing capabilites are planned, they are not currently available. Job modification is not available. No multiple queues are allowed since the system only has the single (generic) PRINT queue in the current version. No operator commands may be issued from a non-privileged account. There is no operator services console, printing of remote files is not allowed, and there is no status display of spoolers. Only the characteristic /FORMS is defined in the first release, so there can be no definable job characteristics. As yet, there are no forms alignment procedures, and no loadable fonts for printers. *

RSTS System Performance Optimization

Michael Mayfield
Northwest Digital Software, Inc.
Newport, WA

Ed McKay, Session Chairperson
Galveston College
Galveston, TX

Reported by Todd Spangler
DECUS Scribe Service

System optimization is always important in the business world. Michael Mayfield of Northwest Digital Software, Inc. presented several ways in which a RSTS system can be optimized.

In system optimization, the ideal is to maximize all features with respect to each other. At no time will all of the features work at 100% operating capacity with no waste. In checking your system's STATUS data, some goals to aim for are:

1. Less than 5% CPU idle time (time that the CPU is not in use).

2. 0% of CPU time lost due to insufficient memory.

3. More than 66% CPU time used for user jobs

4. Less than 15% monitor overhead charged to a job.

5. Less than 55% monitor overhead not charged to a job.

6. Minimum number of characters output to the terminal without affecting user performance significantly.

7. Less than 10% amount of time FIP is in use or waiting.

8. Less than 7% amount of time FIP is in use.

9. Less than 15% monitor overhead for interrupt processing.

10. More than 15% free small buffers.

11. Less than 80% of maximum number of accesses per second for disk type.

12. Less than 80% of maximum transfer rate per second for disk type.

When optimizing the system, it is best to have low monitor overhead time when compared to the actual user time involved, as little swapping as possible (a little swapping is not bad), and reasonable memory management. Disk access is also important. In order to optimize the disk, one needs to decrease the usage of the disk and also the amount of seek distance that the head must travel. To do this there are software products available which can be used to group the most frequently accessed files together in one area on the disk. Secondly, the files can be grouped further on the disk with respect to the number of times that the file is actually opened and accessed. In other words, group together according to UFDs. The UFDs can also be cut down by optimizing the clustersize of files. A formula to calculate the optimum clustersize is:

```
CLUSTERSIZE=-(2^INT(LOG(FILESIZE/
    7)/LOG(2) + .9999))
```

The cluster size should be adjusted to -256 if the formula yields a number equal to or smaller than -256. Cache hit ratio verses cache age also is important because if a cache buffer is required and the proper data is not there, then there is considerable time lost during the search for the proper data. Cache size relates to the same principle, since if the data is not there then the system must search for the proper data. One of the most important factors in RSTS performance is FIP usage. In order to optimize FIP usage one can optimize directory structures, create contiguous files, and use proper clustersizes. This will make a dramatic improvement in the system response.

Small buffers needed by the system can cause an interesting problem. If there are less than 25% of free small buffers available, the system goes into a first level panic. This means that the number of small buffers allocated to separate devices is limited and can cause allocation to be denied to the device. If the percentage of free small buffers goes below 20%, then a second level panic exists. At this time, many more devices will be denied access to the small buffers. If there is less than 10% of small buffers free, the system halts.

Besides STATUS, one product that can be used to collect RSTS performance statistics is RPM (RSTS Performance Moniter). For individual systems, needs will be different, therefore the system optimization will require that different things be adjusted. When proceeding with optimization, change only one factor at a time, since changing multiple factors may result in unreadable results or they may cancel each other out, giving misleading information. The final analysis will be based on your system's needs.

For further information, contact:

Michael Mayfield
Northwest Digital Software, Inc.
Box 2-743
Spring Valley Road
Newport, WA 99156

*

New Users if RSTS/E Hints and Tips

Carl B. Marbach
and
Dave Mallery
RSTS Professional Magazine
Fort Washington, PA

Thomas Robbins, Session Chairperson
Seattle Pacific University
Seattle, WA

Reported by Susan Miller
DECUS Scribe Service

Are you a RSTS Guru? If so, you probably know of the Carl and Dave Show. Carl B. Marbach and Dave Mallery of the RSTS Professional Magazine brought their show to the DECUS Symposium. Their presentation was "New Users of RSTS/E-Hints and Tips."

The audience received a handout, entitled "New User's Manual For RSTS/E." The handout explained some of the hardware of the computer such as:

UNIBUS--"The UNIBUS (DEC trademark) forms the backbone of any PDP-11 computer system. It constitutes a 56 wire party line on which any device of the system can talk to any other device. The ribbon cables interconnect each device from one to the next. Inside each device, the UNIBUS takes the form of a backplane into which the various circuit boards are plugged. Eighteen of the bus lines carry address information; another 16 lines are for the data. The rest of the lines are used for synchronization, handshaking, interrupts, and initialization signals."

CONTROLLERS--"The controllers are devices which exist between a physical device (what you see) and the interface inside the computer. They are found on mass storage devices that need lots of preprocessing of data. For instance, a disk drive reads and writes bits on your disk pack, but the controller groups them into words, counts the words as they are moved, checks the parity, and even corrects the data if it can. It also contains all the control and status registers that you see in an ERRDIS printout. Controllers are also called formatters. Simpler devices, such as terminal interfaces, have the registers right on the interface card and do not process data."

Marbach then gave some information on different processors offered to users of the PDP-11. The terms were explained in their numerical order. Marbach has not found any logical reason for the names of the terminology. But he did list some differences:

1. The 11/2Ø is the first PDP-11 and also the first one to support time sharing.

2. The Micro-11 supports up to 4 megabytes of memory. It uses a Q-Bus system.

3. The 11/23 uses Q-Bus and is packaged a little differently so that additional disks can be attached.

4. The 11/24 is functionally similar to 11/23 but uses a UNIBUS system.

5. The 11/34 is a UNIBUS which only does 18-byte addressing.

6. The 11/44 has cache memory, which makes the memory operate faster, and supports up to four megabytes of memory.

7. The 11/45 is a fast processor and limited in memory.

8. The 11/6Ø is also limited in memory but allows a user to write instructions. This is good for scientific but not commercial use.

9. The 11/7Ø has 22-bit addressing. It has UNIBUS and MASSBUS, which puts it through faster.

Performance information can be obtained on the RSTS operating system. Marbach suggests contacting your local salesman and asking for the Performance Handbook. "If he doesn't know what a performance handbook is, tell him to call the home office and ask for Al Saloky." He's responsible for most of the performance monitoring. His group provides documented information on performance of all Digital processors.

Hints were also given to configure a computer system. Try to establish what you're buying. How many people will use it? Decide what kind of processing and memory you need. And configure statistics information into your system.

The need to have a backup was strongly expressed. The backup should be kept in a different place from the work area. If the building burned, everything would be gone. Mallery had his backup stored in the back seat of a car until he could afford a vault in a fireproof warehouse. Usually backups are needed most because of a human's error, such as a wrong key stroke that happens in a fraction of a second. They recommended having a backup of the previous four days work. Then a weekly backup. And of course a monthly backup for as many months or years that you can keep it.

Mallery gave tips on how to structure disks. This is the single biggest factor in RSTS system performance inprovement. The user file directory, or UFD, should be contiguous, centered and about one-third of the way on the disk.

The most amusing hint and maybe the most useful, was to never use a newly released product until Patch Kit B has arrived. *

Maxi in your Mini

Christopher Johnson
North Shore Sanitary District
Gurnee, IL

Bill Tabor, Session Chairperson
Racal-Milgo
Miami, FL

Reported by Margaret Watters

DECUS Scribe Service

Christopher Johnson, an inexperienced programmer, aimed his talk at other novices in the audience, giving several tips on how to maximize processing in a PDP-11 minicomputer under RSTS. If a programmer is receiving a "maximum memory exceeded" message, he has several options he can take in order to get a large program running with all of the necessary components included.

The first thing to be done when a "maximum memory exceeded" message appears is to think about the ways that the program can be segmented into smaller, more manageable pieces. When the decision is made as to which pieces should go together, they can be put into subprograms which can then be "overlayed" according to the program's ODL file. Each of these separate subprograms must be "logically independent"; in other words, programs at the same level should not call each other.

Programs may be overlayed at several different levels, with one program calling another, but this can defeat the purpose of having the programs overlayed. If a certain subprogram is included in many different paths of the overlay tree, it may be beneficial to include that program on the first level, with the main program. Then it is always in the program's job space and is available for another subprogram to call without control actually passing back to the main program.

If one or two subprograms are quite a bit larger than the others, adding another level to that subroutine path would probably be of help. This would allow different parts of these large subprograms to be overlayed, thus saving space and making these large subprograms more manageable. While deciding how much a program needs to be segmented, one must remember that it is possible to lose more space by having the subprogram than can be gained from the segmentation. It may be very helpful to generate a "Memory Allocation Map" in order to check the effectiveness of the program segmentation.

When fitting a large program into a small space unnecessary line numbers become an unaffordable luxury. Line numbers should only be used where absolutely needed. A blank line is an alternative to line numbers. Another example of a space saver is the use of unmapped constants. There are other measures one can use to save space in a program including co-trees but the speaker was more interested in methods that would help the inexperienced programmer. *

RSTS V8.0 Field Test Panel

John Santos
Evans, Griffiths and Hart, Inc.
Lexington, MA

Bruce L. Gaarder
Macalester College
St. Paul, MN

Carole Lape
Digital Equipment Corporation
Maynard, MA

Rocky Hayden
North County Computer Services
Escondido, CA

Jeff Killeen, Session Chairperson
Information Design and Management
Sherborn, MA

Reported by Susan Miller
DECUS Scribe Service

Producing a final product is a complicated task. A complete test must be performed in order to ensure that the product is usable and reliable. For this task, a field test is done. Field test involves the product development team and the test site, both of which must make commitments. A kit is sent to the field site containing 16ØØ BPI tapes of the testing material, documentation on the material, a cover memo which describes the test, and release notes which explain the differences from the older version of the product.

In field test the use of Quality Assurance Reports (QARs) becomes important. As problems in the material are found, the information must be relayed to the field test division where quick handling occurs. Patches, which solve the current problem or simply explain what is going on, are sent back to the test site. This procedure is continued until the test ends. It is expected from the test sites that an appropriate amount of time be spent on the product, QARs are sent, and documentation is maintained. To become a test site, do one of the following:

1. Contact your local office.

2. Contact the proper people at the DECUS Symposia.

3. Contact the field test people.

When giving information on the field test application one should include all configurations of the available system as hardware will be crucial in the test.

John Santos of Evans, Griffiths and Hart, Inc. described his experience with the field test of RSTS/E Version 8 as being good. The advantage of field testing is that one sees the software before it is released, with the trade off being the time consumption. Some of the problems discovered during the field test of the Version 8 is that SYSTAT still produces '??' for the uptime when the system is on for more than 1ØØØ hours, HELP.HLP was copied to the wrong place, one should say BUILD instead of BUILD/PATCH during the system build using BUILD.CTL. EDT leaves some files in the system that are not explained and SAV/RES does not function properly. Overall, the new version runs well. Some compliments expressed by Mr. Santos were that bugs which were found before the end of field test were patched properly and the new disk structure is fine.

According to Bruce Gaarder of Macalester College, an added advantage is the fact that the run time was less than for the previous version. The minor problems were easy to get around. The same problem encountered was that of SAV/RES, which yet remains unsolved. Here again field test was generally considered to be an advantage. Rocky Hayden of North County Computer Services stated that the installation and the SYSGEN went well and included an increase in throughput by 1Ø to 15 percent with the new disk structure.

Some other problems encountered by Mr. Hayden were that a bootable tape could not be created using HOOK and SAV/RES would not work. The task builder worked OK except with overlays - so use 7.2 task builder. Also the task builder must be used with the new SYSLIB. There can be no combination of the task builder and the old SYSLIB or the old task builder and the new SYSLIB.

(Note--for more information on the field test program, see "Field Test: What is it and How Do We Pick Sites" later in this newsletter. --ed.) *

Tech Tips II - Bit Pushers

(There were 3 "Tech Tips" sessions presented at the Symposium. These sessions give the users a change to bring up questions to be answered by the DEC developers or by other users. There is a different orientation to each session: "SPR's", "Applications", and "Bit Pushers", which allows users to pick the sessions that they wish to attend, based on their interests. The following is a sample of one of those sessions.--ed.)

Jeff Killeen, Session Chairperson
Information Design and Management
Sherborn, MA

Reported by Joseph Lowery
DECUS Scribe Service

Q. On EMT logging in the switch register, now that you've given us everything on the switch register between 1 and 15, why not let us select 6 or 7, or however many switches are left, so we can turn the subset of the EMT loggers off and on?

A. We're looking at a more generic way of handling that.

Q. What are some hints with regard to what I need to look at in the DDB of a modem controlled line to tell if someone is connected to the line but not logged in?

A. There's a variable offset in the DDB called MODCLOCK. You have to look up that value in your monitor SIL. If the most significant bit is set then the modem line is hung up. In terms of turning off and on a subset of the EMT logging you can do it yourself from your BASIC program by peeking at the switch register location.

Q. When RSTS runs low on small buffers, what starts happening?

A. As it starts to run out, it will quit doing things. Around 75 or so it will logging errors, at 40 it will quit logging in, and as it keeps getting lower it will keep slowing down until it hits zero, when it will stop until things free up. If any interrupts occur and DSQs get freed up from disc IO, then things will gradually come back.

Q. Is there any other reason you can think that would cause the system to keep people from logging in at about 24 users online when there is sufficient swap space and the jobmax is 30?

A. Small buffers, swap space, or if, after you added the swap file, you forgot to reenable logins to force recalculation of the new maximum.

Q. I have a tip with regard to the HELP program. If you put $HELP.HLP as the first file in the $ directory to make it contiguous, it helps quite a bit. Do you know why this is?

A. It does a sequential search through the directory to find your HELP file. If it's at the end of the directory, it takes a while to get to it. Sequential caching would help as well. Also, depending on the indirect references to other HELP files within the HELP, you could position the more frequently accessed HELP files at the beginning of the directory.

Q. We've had some problems with NPR devices, specifically the DMR-11. It will start hitting the bus with heavy NPR activity, with almost no way to notice what's going on. RSTS never seems to get the time to notice that. Is there any way to tell if anything is actually hitting on the NPR?

A. Use the logic analyzer, since the software can't tell what the bus is doing.

Q. If RSTS is keeping count of small buffers, as was earlier mentioned, is there a global peak address I can look at so that I can keep count and broadcast a warning message to appropriate terminals?

A. FREES+2, but that will change in each of your SILs.

Q. What happens when the buffer levels get down to 20% free and 25% free?

A. Those two percentages deal with buffer quotas for character oriented devices. As long as you are over both 20 and 25 percent free, character oriented devices will be allowed to use as many small buffers as they like to buffer their IO, with the exception of a terminal that has been stopped with XOFF. When you start to drop below those values, then it will no longer be allowed to exceed its buffer quota. The difference between the 20% and 25% figures is that at 20% it will no longer let it get its quotas, and you'll get caught in a buffer stall in the BF state.

Q. On disabling terminal lines, we often use a DH port, and found that setting speed zero on a DH works fine for shutting it down and trying to run login on both sides. On DZs, which don't have zero speed on the hardware, how can you set zero speed? Is the monitor looking at the speed?

A. If the monitor sees speed zero, then it will disable transmits and ignore receives.

Q. With an 11/70 with four RM03s, two of them set up as DMs and two as DRs, two different controllers, how awful is this? Is the CPU being beaten on by both of those controllers so much that it is not really thinking very much?

A. It's actually a pretty good configuration, since you can not only overlap seek on both drives, but you can overlap the IO operations as well through the two controllers.

Q. What exactly is a missed error?

A. A missed error is when you get below 75 buffers and it can't log it, or if the error copy program has exceeded its message limit and gone into hibernation. This can also happen if you are in the error logger when you get another one; in this case the second error is missed. *

Field Test: What is it and how Do We Pick Sites?

Stephen R. Beason
Digital Equipment Corporation
Maynard, MA

Angela J. Cossette
Digital Equipment Corporation
Maynard, MA

Thomas E. Davis
Digital Equipment Corporation
Maynard, MA

Emily Kitchen, Session Chairperson
A.H. Robins Company
Richmond, VA

Reported by Margaret Watters
DECUS Scribe Service

Three DIGITAL representatives gave a session on what a field test is, and how installations can become field test sites. Stephen R. Beason, the Central Quality Group Manager, began the session by giving a general discussion on what constitutes a field test, why they exist, and why a customer would want to become involved in one.

A field test is the last step in the qualification period. The main objective of a field test is to increase the quality of DIGITAL's products. What cannot be tested in the DIGITAL environment, can be tested in the field. The field test also is helpful, because there are uses of products that DIGITAL had not considered which may create some problems in the product that DIGITAL can then look into. DIGITAL can also evaluate to what extent the customer's requirements and expectations have been met by the new product.

The customer would benefit from a field test at his site, because he would be able to be directly involved in the development of a product that he uses every day, and thereby to increase the quality of that product. Also, there is the possibility that the site will experience a competitive edge by having the product in use up to a year earlier than the shipping date.

Thomas Davis, a Software Services Representative, felt that the major advantage to field tests is the increased customer satisfaction they foster. He also mentioned the general guidelines that DIGITAL uses to choose test sites. Since DIGITAL wants real testing of their products, the main criteria is that the site will test the product well, and that it have a technical staff that will be able to communicate effectively with the DIGITAL engineers. It is also important for the site to have the necessary resources for the product to be tested (hardware etc.). He also said that the site must have a good relationship with DIGITAL.

Angela Cossette, the Field Test Administrator, discussed how the field test works. The three major segments of the process are setting up, testing, and termination. During the testing period the people working at the site send QARs to DIGITAL which are worked on immediately. After the termination period, a questionnaire must be filled out. In terms of the cost to a chosen site, the site must spend the time using, testing, and providing feedback, but the use of the product is otherwise free. One member of the audience said that the site must also pay transportation costs associated with attending a training seminar, but there was no comment from the speakers, so the matter remained unclear.

A site that wishes to become a field test site should get in touch with its local Software Services Manager or with Angela Cossette directly:

Digital Equipment Corporation
P.O. BOX F
Maynard, MA, Ø1754 *

Impact of Regulatory Environment on Digital's Products

Richard Amann
Digital Equipment Corporation
Maynard, MA

Debra Young, Session Chairperson
Boeing Commercial Airplane Company
Seattle, WA

Reported by Joseph Lowery
DECUS Scribe Service

The laws governing the potential hazards and problems of a computer system were discussed in an hour long session Monday, May 23. The speaker, Richard Amann of Digital Equipment Corporation, began by describing some of the situations which might arise and cause personal injury to the workers. The examples cited ranged from the immediacy of falling equipment and fire to the long term effects from working with a VDT, such as eye and back strain as well as the still uncertain effects of the small amounts of radiation emitted from the systems.

The precautionary measures which Digital takes in preventing all of these occurrences was then discussed in terms of four major areas: acoustic, electrical, electro-magnetic and ergonomic, or human engineering. Of these four, the latter two were dealt with in greater detail.

The electro-magnetic radiation (EMR) emitted by computer systems is similar to that emitted by all major appliances and hence is commonly known to supply interference to radio and television broadcast signals. Thus, should a computer system be located too near a receiver of the broadcast signals, interference may occur and the Federal Communications Commission may see fit to order the system shut down until the problem is resolved. Due to this and the speculation that long term exposure to the EMR may cause health problems, Digital has employed a staff of domain experts to test and subsequently design systems to operate not only with a minimum of EMR emission, but to remain stable and fire retardant as well.

With regard to EMR emission in particular, but applicable to other areas as well, Digital requires that its products are restrained to emit levels well below those allowed by the present U.S. laws. (Regulations in Europe, and specifically Germany, hold standards that, like DIGITAL's, fall well below the maximum allowed in the U.S.)

With regard to the ergonomic considerations employed by Digital, Amann informed the audience that human engineering is strongly considered when Digital products are designed. This is demonstrated by DIGITAL's systems with movable keyboards and adjustable screen brightness.

A brief discussion concerning proposed laws governing VDTs concluded the session with DIGITAL's position on many of the laws which are seen to be unfair to users. Mentioned was one such law whereby a user would be limited to use a VDT for five hours a day, due to the possible harmful effects of the EMR emitted.

This was followed with a presentation of nine examples of experimentation in this area, all nine of which concluded that the use of VDTs present negligible health hazards. *

DECUS Library Report

Ardoth A. Hassler
The Catholic University of America
Washington, DC

Reported by Phil Beene
DECUS Scribe Service

During the recent DECUS Symposium held in St. Louis, Ardoth Hassler, library coordinator for the DECUS Library Board, led an working session designed to update interested users on DECUS Library activities.

Following her brief status report on how the Library has been operating since the previous DECUS Symposium, SIG representatives from the U.S. Program Library Committee delivered short reports on how their individual groups are progressing.

Ardoth began her report by explaining the DECUS Library's new incentive/reward program is designed to encourage program contributions from DECUS users. Any member contributing a program will receive a plaque from the Library acknowledging their efforts. Although the announcement brought a general sense of approval from members of the audience, many thought the program would be more successful if the contributor were offered the alternative of receiving credit towards one of the Library's existing listed programs. This suggestion will be referred to the Library Committee.

Following Ardoth's announcements, Larry Hicks gave a brief report on his work with the Library catalog. Since the last Symposium, the previous three existing versions of the catalog have been compiled together into one document. A free copy of this new version should be mailed to all DECUS members around June.

The Library will begin treating this catalog as its main marketing tool. It will be given away in order to make more DECUS members and others aware of the wide selection of programs available through the Library System.

In addition to the catalog distribution the Library will continue to improve user awareness through posters, buttons, booth displays and the implementation of advertising.

A discussion of the Library's current taping programs and strategies elicited considerable input from members of the audience. One suggestion which came up during this portion of the discussion was the Library's need to make a better general abstract listing of available tapes and the information contained within them. The point brought up, was that many potential Library customers can't justify the tape purchase price without a better knowledge of what they are buying. Ardoth said the group would consider this in the future and try to come up with a better list.

Another taping problem currently being experienced by the Library is their need for a better copying system. A new mass-producing copying unit is one of the Library's main objectives.

For those users and SIG members unable to attend the Symposium, or just wishing to obtain copies of SIG sessions, master tapes will be available. SIGs having copies made at the St. Louis Symposium included: RSX, RT-11, DECsystem-1Ø/2Ø, VAX, and RSTS.

Copies of these tapes are available to users for about $112. The RSX and VAX tapes are quite lengthy, and are expected to include two tapes.

The last portion of the formal half of the session included a discussion of long range planning goals. A goal of the Library is to acquire a method of determining what items and interests users will have in the future. According to committee members, this knowledge will enable the Library to better plan and prepare for these needs, saving everybody time and money in the future.

Following the formal first half of the session, Ardoth and other committee members took questions from the audience. Many of the questions concerned the alternative methods available to create Library donor incentives. Most DECUS members present agreed a credit system would be more effective than the current plaque proposal. Citing other successful library programs, even Ardoth agreed, but said she was not certain such a program would work in the DECUS system. She feels the Library could easily become overwhelmed with nonusable programs offered by users wanting free programs. Although the DECUS Library is a non-profit organization, she said such results could destroy the system's financial stability through increased operating costs.

One alternative to the plaque system already being discussed, is the possibility of offering a RAINBOW 1ØØ system through a lottery which would include all program contributors.

In other discussion, some users felt that DECUS should obtain stronger rules applying to users' copying and distribution of programs obtained through the library. Though there is no law which protects the programs, which are public property, members said it would be nice if DECUS created a head or logo to least recognize their contributions to program users. This would not only be self-rewarding to the program authors, but would also create a greater awareness among non-DECUS programmers.

In concluding the meeting, Ardoth promised all suggestions would be carefully discussed among Library managers, and promised to let users know results of these discussions as soon as possible. *

What the New DECUS Will Be -- The OD Task Force

Clair Goldsmith
University of Texas
San Antonio, TX

Reported by Micheal Kintz
DECUS Scribe Service

The Organizational Development Task Force established in 1981 functions "...to evaluate and recommend methods and practices which will benefit DECUS leadership and users in the more effective delivery of DECUS services." The Task force is developing a project designed to give DECUS a new look.

The Organizational Development project stems from the 1981 and 1982 Leadership Interning which indicated the need for:

1. Clarification of DECUS's purpose.

2. Reevaluation of DECUS's structure, control mechanisms and interrelationships.

3. An executive board of managers and policy makers.

4. Emphasis on improved communications.

5. Career development for leadership.

6. More leadership development.

In order to achieve a new look for DECUS, the project developed a strategic plan which consists of several phases: mission, goals, and action plans; organizational structures; transition plans; and human resources plans for volunteers. Clair Goldsmith stated that the function of the strategic plan is "to promote the exchange of information processing related information among users of Digital Equipment Corporation products."

The goals of the OD project are to actively represent the interests of members, the establishment of activities to promote information exchange, the design and implementation of strategies to encourage active membership, effective chapter management, maintenance of the special DECUS relationship with Digital Equipment Corporation, and support of communications between suppliers of products compatible with DIGITAL equipment and users of DIGITAL equipment.

Action plans will be enacted to implement goals of the strategic plan and will become basic activities necessary to continued project progress in the following year.

The organizational structure proposed on May 2Ø, 1983 will not be rigid, but will let elements come and go as appropriate. The currently planned elements of the organizational structure are:

1. An Executive Board with 9 to 11 members (1 DIGITAL representative, 6 voted in by members-at-large, and 2 from the Management Council.) The Executive Board will be responsible for long term planning.

2. A Management Council with 13 or 14 members to include delegates from Functional Groups, SIGs, and LUGs. The Management Council will manage the day-to-day activity of the organization.

3. A Chief of Staff who will manage the DECUS professional staff. The Chief of Staff will serve on the Executive Board as a non-voting member.

4. A five member Recruitment Committee. The Recruitment Committee will head new leadership development.

5. Placement of SIGs and LUGs at the national level of the organization.

6. Several Functional Committees chartered by the Executive Board, to include: Library, Symposium, Publications, Standards, and Special Projects groups.

The transition plan will consist of reviewing and incorporating the data input from the Spring U.S. Symposium, Saint Louis, regarding the June meeting of the Task Force. Goldsmith said that members should watch for feedback in DECUSCOPE, the Pageswapper, and SUGgestions.

After reviewing and incorporating the data input, the Task Force will design a transition plan, revise Bylaws, and submit the revised Bylaws to the membership for approval. Goldsmith said the Task Force would like to have the Transition Plan completed by June 1984.

The final phase of the OD project is the human resources plan for volunteers, which will consist of leadership and career development and volunteer recognition and rewards. *

The "Symposium Report" articles were produced by students under the DECUS Scribe Service, and were not reviewed by the session speakers before publication. *

New DECUS Library Offerings for RSTS Users

DECUS#	DESCRIPTION
11-560	LST: A Paging Utility for Non-Form Feed Devices (in BASIC+2). A system utility for printing out text files in a paged format on non-form feed devices. Includes options to supress page numbering, set special form lengths, print a title, etc.
11-601	KBSET: System Start-up for RSTS/E Program for speeding up the setting and changing of keyboard characteristics.
11-602	RSTS Libraries for Swedish PASCAL Used with the Swedish PASCAL compiler (DECUS No. 11-346). 28 external procedures for access to some RSTS operating system facilities, such as block I/O, date and time access, etc. Also, 13 procedures for input and string processing and 18 procedures for 32 bit integer arithmetic.
11-607	MEMO: Computerized "note box" A quick, convenient way of storing notes about good ideas and/or things to do. Features include listing memos by subject material, appending to previously entered memos, and output to a file.
11-610	DCW Menu for RSTS/E Systems Supports the creation, interactive editing and use of menus. Includes program to initialize menu files, add, delete, and edit menus, and change menu control parameters. Each menu can contain up to 36 items. Longer menus may be subdivided into 2 or more linked menus, or nested sub menus.
11-622	MONITR: A Display Program for RSTS/E Dynamically monitors statistics for a specified job. Useful for debugging and monitoring of suspicious activities.
11-624	DIBOL Subroutines Includes: Gregorian/Julian date conversions, accept data from CRT, state abbreviations, state/city tax calculations, etc.
11-SP-47	PORTACALC: A Portable Spreadsheet Program (in FORTRAN IV-PLUS) Although this program is not specifically written for use under RSTS, those who have FORTRAN might be able to use it.
11-SP-50	Preliminary 'C' Language with Floating Point and Other Software Requires floating point support hardware.

*

RSTS Newsletter Article Index

Articles through Spring 1983

The following index can be used as a guide in determining where to find an article in a previous newsletter. Realize that many of the old articles are outdated, in that they refer to previous versions of the software. You may expend great effort in obtaining an article only to discover that it only applies to an old version of the software. For those who do not have the old newsletters, there is now available 2 sets of microfiche of old RSTS newsletters through the Fall 1982 issues. The newer set has the issues from 1977 through 1982. This set was sold at the DECUS booth at the St. Louis symposium and it will be available also at Las Vegas this fall. Look for information in the future as to how you can obtain this microfiche set if you are not able to go to the Fall symposium. The older set covers 1974 through 1977. There are just a few of these left and can be requested through Maureen Levine at the DECUS office for no charge. There will be a charge for the newer set. (It was $5.ØØ at the Spring Symposium.)

--- A ---

AT.BAS and INDIRE.BAS--patches........................Vol. 6 #3, May 1979

--- B ---

BACCON--patch for automated backup...................Vol. 6 #3, May 1979
BASIC Language--Statement of desired functionality....Vol. 7 #1, February 198Ø
BASIC User's Survey--Questionaire......................Vol. 6 #4, August 1979
BASIC-PLUS Patch--Garbage Collector...................Vol. 9 #3, Spring 1983
BASIC-PLUS Software Conventions......................Vol. 4 #3, July 1977
BASIC-PLUS Standards--DEC Internal Document..........Vol. 4 #1, February 1977
BASIC-PLUS Standards--tentative draft................Vol. 5 #2, April 1978
BASIC-PLUS--Odds and Ends--routines..................Vol. 4 #2, May 1977
BASIC-PLUS--proposed programming standards...........Vol. 1 #3, December 1974
BASIC-PLUS-2--Making Programs Sharable...............Vol. 8 #1, April 1981
BASIC-PLUS-2/SORT11 sort subroutines.................Vol. 6 #2, March 1979
BMDP Statistical Package--product description........Vol. 4 #4, September 1977

--- C ---

CCLMAN -- CCL Manager for RSTS/E.....................Vol. 9 #2, Fall 1982
CHNGPK--program to change disk MFD label entry........Vol. 6 #5, October 1979
CKPROT--Check Protect String Routine.................Vol. 6 #5, October 1979
COBOL--Sort Calls sample program.....................Vol. 5 #4, July 1978
COMENT--program to produce Basic comment lines........Vol. 5 #1, February 1978
COMMERCIAL SUB-SIG--discussion of Decus Meeting.......Vol. 3 #1, February 1976
Channel 13--answers to users' questions..............Vol. 3 #1, February 1976

--- D ---

DATE compression/expansion function..................Vol. 3 #3
DEC Programs--unsupported--user access discussion.....Vol. 4 #3, July 1977
DECAL--comments and wishes...........................Vol. 4 #2, May 1977
DECUS Library--RSTS-11 Status Report.................Vol. 9 #2, Fall 1982

RSTS SIG Newsletter Article Index -- Articles through Spring 1983

DECUS Library--recent submissions	Vol. 6 #6,	November 1979
DETACH SYS CALL--Patch to make non-privileged	Vol. 9 #3,	Spring 1983
DEVICE INPUT/OUTPUT--technical discussion	Vol. 4 #2,	May 1977
DISK OPTIMIZATION	Vol. 5 #4,	July 1978
DISK REBUILDING CHECKLIST	Vol. 9 #3,	Spring 1983
DISKID--program to read and change disk ID	Vol. 8 #1,	April 1981
DOCPKM--Terminal Session Monitor Program	Vol. 6 #1,	January 1979

--- E ---

ECHO CONTROL--Painting with RSTS/E	Vol. 4 #2,	May 1977
EDT V2--My favorite key definitions	Vol. 9 #2,	Fall 1982
EDT--V.1 bugs	Vol. 5 #5,	September 1978
ERRDET--Patch for 'bit' output of registers	Vol. 4 #4,	September 1977
ERROR HANDLING--suggested Basic+ inprovements	Vol. 3 #2,	April 1976

--- F ---

FIELD SERVICE--comments	Vol. 1 #4,	February 1975
FILE ACCESS--bit mapped structure	Vol. 3 #5,	November 1976
FIP Buffering--effect of increasing on 11/7Ø	Vol. 4 #4,	September 1977
FLSTAT--program to alter file status bits in UFD	Vol. 6 #3,	May 1979
FORTRAN Call for proper ASCII collating sequence	Vol. 5 #1,	February 1978
FORTRAN IV under RSTS--product announcement	Vol. 2 #4,	October 1975
FORTRAN to BASIC Chaining--sample programs	Vol. 7 #2,	May 198Ø

--- H ---

High Priority CPU-Bound Job--How to Kill	Vol. 9 #2,	Fall 1982
High Priority CPU-Bound Job--How to Suspend	Vol. 9 #3,	Spring 1983

--- I ---

INDENT and FMS--a comparison	Vol. 9 #2,	Fall 1982
INIT--patch to put date and time in logs	Vol. 3 #1,	February 1976
INPUT AND OUTPUT--technical discussion	Vol. 4 #1,	February 1977

--- J ---

JOB NAME/KB NUMBER Function	Vol. 3 #3	

--- L ---

LABEL--program to read Backup tape labels	Vol. 6 #4,	August 1979
LARGE RSTS USERS SUB-SIG meeting	Vol. 4 #1,	February 1977
LINKER--bugs	Vol. 5 #5,	September 1978
LISTER.BAS--a program listing program	Vol. 4 #4,	September 1977
LOGIN--offer for listing of some patches	Vol. 4 #4,	September 1977
LOGIN--patch for auto baud rate selection	Vol. 2 #3,	August 1975
LOGIN--patches	Vol. 5 #1,	February 1978

--- M ---

MACRO-11--Introduction to Macro-11 Programming	Vol. 9 #3,	Spring 1983

RSTS SIG Newsletter Article Index -- Articles through Spring 1983

RSTS SIG Newsletter Article Index -- Articles through Spring 1983

RSTS Marketplace	Vol. 4 #3, July 1977
RSTS Marketplace	Vol. 4 #4, September 1977
RSTS Marketplace	Vol. 5 #1, February 1978
RSTS Marketplace	Vol. 5 #2, April 1978
RSTS Marketplace	Vol. 5 #3, May 1978
RSTS Marketplace	Vol. 5 #4, July 1978
RSTS Marketplace	Vol. 5 #5, September 1978
RSTS Monitor Internals--Disk I/O	Vol. 6 #2, March 1979
RSTS Named Directory Support	Vol. 9 #2, Fall 1982
RSTS New Release Working Group	Vol. 9 #1, May 1982
RSTS Notes (LOGIN and SYSTAT)	Vol. 9 #2, Fall 1982
RSTS Operator Training Course--announcement	Vol. 8 #1, April 1981
RSTS Optimization--Checklist	Vol. 9 #3, Spring 1983
RSTS Performance Evaluation Techniques	Vol. 5 #3, May 1978
RSTS Product Development--summary as of V5B	Vol. 2 #2, May 1975
RSTS Programming Standards Sub-group	Vol. 3 #3
RSTS Programming Standards questionaire	Vol. 3 #3
RSTS Programming Standards--DECUS session summary	Vol. 3 #3
RSTS Questionaire results--Fall 1977 Symposium	Vol. 5 #2, April 1978
RSTS SIG Operating Procedures Final Draft	Vol. 6 #4, August 1979
RSTS SIG Operating Procedures--Amended Final Draft	Vol. 7 #1, February 198Ø
RSTS SIG formation	Vol. 1 #1, February 1974
RSTS SIG future activities	Vol. 1 #1, February 1974
RSTS SIG goals	Vol. 1 #4, February 1975
RSTS SIG--Update and Happenings	Vol. 6 #4, August 1979
RSTS Send/receive--Problems and solutions	Vol. 4 #3, July 1977
RSTS Send/receive--Sample Program (using SYS 18)	Vol. 4 #2, May 1977
RSTS Send/receive--Sample Subroutine	Vol. 4 #4, September 1977
RSTS Software Support--misc. letters	Vol. 2 #4, October 1975
RSTS Standards--Coding Standards and Conventions	Vol. 2 #2, May 1975
RSTS Steering Committee--Meeting Notes, June 198Ø	Vol. 7 #3, September 198Ø
RSTS System Management--slides from Decus session	Vol. 7 #2, May 198Ø
RSTS System Performance inprovement	Vol. 3 #4, September 1976
RSTS Timing Test	Vol. 2 #3, August 1975
RSTS Transaction Processing--system abstract	Vol. 1 #4, February 1975
RSTS User Profile--new version questionaire	Vol. 3 #1, February 1976
RSTS User's Profile--reprint of questionaire	Vol. 2 #1, April 1975
RSTS V4 Coding Tips	Vol. 2 #3, August 1975
RSTS V4 Sub-SIG Newsletter	Vol. 3 #3
RSTS V4 Users speak	Vol. 2 #2, May 1975
RSTS V4--DEC speaks	Vol. 2 #2, May 1975
RSTS V4--Decus meeting, Fall 1975	Vol. 3 #1, February 1976
RSTS V4--open letter concerning DEC support	Vol. 2 #2, May 1975
RSTS V4A Internals document availability	Vol. 1 #2, July 1974
RSTS V4A Questionaire	Vol. 1 #1, February 1974
RSTS V4A Questionaire results	Vol. 1 #2, July 1974
RSTS V5B Discussion	Vol. 1 #3, December 1974
RSTS V5B--description of features	Vol. 1 #2, July 1974
RSTS V6B--features discussion	Vol. 3 #1, February 1976
RSTS V6C Hidden Features	Vol. 5 #1, February 1978
RSTS meeting--Dallas, March 1976--minutes	Vol. 3 #2, April 1976
RSTS-11 and RSTS/E--History	Vol. 9 #3, Spring 1983
RTSLIB--patch to use under V7.Ø	Vol. 7 #2, May 198Ø

RSTS SIG Newsletter Article Index -- Articles through Spring 1983

--- S ---

SCOPE--VT52 or VT1ØØ screen handling functions	Vol. 6 #5,	October 1979
SEND/RECEIVE--How to use (old SYS 18 call)	Vol. 3 #2,	April 1976
SPR's and RSTS Software Support	Vol. 1 #3,	December 1974
SPR's--submitted by user	Vol. 9 #1,	May 1982
SYSTAT--patches	Vol. 6 #4,	August 1979
SYSTEM MONITORING TASK FORCE--Report #2	Vol. 6 #1,	January 1979
SYSTEM MONITORING TASK FORCE--Report Reprint	Vol. 9 #2,	Fall 1982

--- T ---

TECO--change to ampersand processing	Vol. 8 #1,	April 1981
Taxbreak on Software Development	Vol. 9 #2,	Fall 1982

--- U ---

UFD deletion--program to delete empty UFD's	Vol. 3 #4,	September 1976
USER PROFILE QUESTIONAIRE	Vol. 1 #2,	July 1974
UTILITIES AND PLOTTING--Package description	Vol. 4 #4,	September 1977

--- V ---

VAX/VMS vs. RSTS/E	Vol. 9 #1,	May 1982
VT1ØØ's with acoustic couplers--hardware problem	Vol. 7 #1,	February 198Ø
VT1ØØ--A User's Report	Vol. 6 #2,	March 1979

--- W ---

WISHLIST additions--letter	Vol. 3 #3	
WISHLIST--Fall 1975, Los Angeles	Vol. 3 #1,	February 1976
WISHLIST--Fall 1975, continued	Vol. 3 #2,	April 1976
WISHLIST--Fall 1977, San Diego	Vol. 5 #2,	April 1978
WISHLIST--Fall 1978, San Francisco	Vol. 6 #6,	November 1979
WISHLIST--Spring 1976 questionaire	Vol. 3 #5,	November 1976
WISHLIST--Spring 1976, Atlanta	Vol. 3 #3	
WISHLIST--Spring 1978, Chicago	Vol. 5 #5,	September 1978
WISHLIST--Spring 1979, New Orleans	Vol. 6 #6,	November 1979
WISHLIST--Spring 198Ø, Chicago	Vol. 7 #3,	September 198Ø
WORD PROCESSING--standards for KB's and characters	Vol. 5 #1,	February 1978

--- X ---

XBUF Allocation--personal observations	Vol. 5 #3,	May 1978

DECUS U.S. Chapter Contacts

FOR INFORMATION ON:

Ordering Material from DECUS:

For general information before placing an order, contact Order Processing	(617) 467-4135
For information about an order you have *placed but have not yet received,* contact Order Processing	(617) 467-4135
For information about an order you have *placed and have received,* contact DECUS Library	(617) 467-4254
For authorization to return defective media, contact the DECUS Library	(617) 467-4254

Library Information:

For information on how to submit a program for inclusion into the DECUS Library, or to check on the status of a newly submitted program, contact the Submissions Coordinator	(617) 467-4177

Membership Information:

To become a member, to check membership status, or to notify DECUS of a change of address or other membership information, contact DECUS Membership Group	(617) 467-4168

DECUS SYMPOSIA, contact DECUS Meetings Planner	(617) 467-7469
DECUSCOPE, Newsletters, and other DECUS Publications, contact Publications Administrator	(617) 467-4143
DECUS Special Interest Groups and Local Users Groups	(617) 467-4889

* * *

A Wishlist item from the first RSTS Newsletter, Vol. 1 #1 February 1974, was a request to increase program size limitation by utilizing I & D space. . .

INPUT/OUTPUT Submissions

INPUT/OUTPUT #1

Caption: SYSTAT Enhancement

Message: A 1981 DECUS tape (San Diego, I think) offered two patches to the SYSTAT.BAS program. One was /Z which showed any open files associated with a job (and where in those files the job was) and /X:n where n was the number of seconds to sleep before running SYSTAT again (useful for tracking job progress). Our source tape for these patches is no longer readable. Help...need a patch for SYSTAT version 8.Ø.

Contact: William S. Ettling
El-Jay, Inc.
P.O. Box 6Ø7
Eugene, OR 9744Ø
(5Ø3) 726-6541

Date: May 17, 1983

INPUT/OUTPUT #2

Caption: CP/M-RSTS Communications

Message: I am in need of programs that will allow a RSTS system to communicate with a CP/M microcomputer. I would like to be able to transfer files between the two systems. I cannot afford to buy any of the commercially available versions.

Contact: Ray Gebbie
Guntert Sales Div., Inc.
P.O. Box 1688
Stockton, CA 952Ø1
(2Ø9) 464-8712

Date: August 5, 1983

Any answers to the above submissions, or any new submissions should be entered on the form at the back of the newsletter. Now that everyone sees how INPUT/OUTPUT works, I expect a large number of submissions. *

INPUT/OUTPUT Submission Form

A SIG Information Interchange

Please reprint in the next issue of The Cache Buffer

Caption: ______________________________

Message: ______________________________

Contact:

Name ______________________________

Address ______________________________

Telephone ______________________

If this is a reply to a previous I/O, which number? ______

Signature ____________________ Date ____________________

Mail this form to: Editor, THE CACHE BUFFER, DECUS, MRO2-1/C11,
One Iron Way, Marlboro, MA Ø1752 USA

Tear out to submit an INPUT/OUTPUT item

Editor, THE CACHE BUFFER
DECUS, MRO2-1/C11
One Iron Way
Marlboro, MA Ø1752
USA

NOTES

NOTES