USUS NewsLetter

Jan/Feb 1990

All the News that Fits, We Print

Volume 4
Number 1

IN THIS ISSUE

The USUS NewsLetter is published 6 times per year by USUS, the UCSD Pascal System User's Society, P.O. Box 1148 La Jolla, California 92038. The NewsLetter is a direct benefit of membership in USUS.

Tom Cattrall Editor
Robert Geeslin, Ed.D. Publisher

Modula 2 Draft Proposal
by Randy Bush

The following article was posted by Randy. I didn't include the details on the January meeting in Virginia as it will have taken place by the time you read this. If you have an interest in the proposed Modula 2 standard, be sure to order a copy of the draft proposal and submit your comments. (See last minute news on page 2)

Although the Draft Proposal (DP) was registered (SC22 Secretariat in Canada) on November 2, the offical copy had not arrived in the US as of Friday, December 15. Once the official copy is received in the US, it will be shipped to the Computer Society Standards office in Washington DC, from where it may be ordered (see order form below).

To speed distribution, Modus, in the persons of Kim King (king@gatech.edu) and Stan Osborne (stan@dbi.uucp), is trying to reproduce my copy. As the document is 550 pages, this is no mean feat. Details of this effort should be announced next week. It is hoped that this will get you the document quickly.

Come to the January meeting. It is where we will vote on the DP. Jan 29-31, Contel Technology Center, Chantilly, Virginia. The DP will go through DIS all the way to becoming an ISO standard with no formal public comment. WG13 will send it to SC22, the national TAGs will pass their recommendations to their SC22 representatives, and SC22 will approve it. It will then be blessed by the ISO hierarchy, and become an ISO standard. The only review processes are within the national TAGs and WG13.

Similarly, it can become an IEEE standard with no public comment. P1151 would review and pass it, give it to the MSC for approval, and so on up the IEEE chain, with the draft never having been formally commented upon by the general Modula-2

public. But for it to become a US standard, it must go through ANSI's cycle of formal public review and comment within the US, and similar processes are likely in other countries. Having the US public review and comment after passage of an ISO and/or IEEE standard could very well result in a US standard different from that of ISO and/or IEEE, which is at direct odds with the US committee's desires, intent, and original understanding. We urge you to get this draft and submit comments. We want your input.

We are currently attempting to keep the following schedule:

- 15 Dec - Receive DP, get it disseminated as quickly as possible. Everyone needs time to get and review the draft.
- 29 Jan - US meeting to formulate P1151 and US TAG comments on the DP.
- 15 Feb - Send formal vote and comments to IEEE to go up the chain.
- 30 Mar - Receive collection of all formal comments, and disseminate. Committee members need time to review the comments.
- April and May - US internal communication, likely by email, as time is too short, and we have been unsuccessful holding meetings only three months apart.
- ?? May - WG13 meeting.

The P1151 and US TAG's mailing list is available on uucp/Internet, BITNET, FidoNet, CompuServe, and MCImail. To join the list, please send your email address to

p1151-list-request@m2xenix.uucp

To send mail to the list, send to

p1151-list@m2xenix.uucp
or
..!uunet!m2xenix!p1151-list
or via non-electronic mail
Randy Bush
Pacific Systems Group
9501 SW Westhaven
Portland, OR 97225

Thanks to Tom Reid for volunteering to host the January meeting. We very much hope to see you there. And thanks to Kim King, Stan Osborne, and IEEE for helping to get the DP published.

To order a copy of the draft proposal, see the order form on the next page.

Last Minute News

Due to the inordinate delay in getting the DP in the US and in other countries, the US TAG will be requesting a delay of the ballot by SC22. Other countries may be doing the same, but I am not sure.

The US TAG / P1151 meeting will be delayed until the week of 5 March. New notices will be sent soon.

The WG13 meeting seems to be shaping up for the end of May or early June in the UK.

Within the US, the DP is still only available from Stan Osborne acting for Modus.

My apologies for this embarrassing situation, and my thanks to Kim King, Stan Osborne, and Tom Reid for their support and assistance.

Randy

TO ORDER DP10514 FROM THE IEEE COMPUTER SOCIETY

COST: $35.00
Mail or Fax this form to:
IEEE Computer Society
Standards Office
1730 Massechusets Avenue NW
Washington, DC 20036
Attn: Lisa Granoien
Fax (preferred): +1 (201) 562-1571
Order ISO/JTC1/SC22/WG13 Draft of DP10514 - P1151 Modula-2
Name: _____
Company: _____
Address: _____

City/State: _____
Phone: _____
I have enclosed $ _____ for ____ copies
(make checks payable to IEEE Computer Society P1151) OR

Please charge __ Visa __ M/C __ AmEx
Number: _____
Expires: _____
Bank No: _____
Signature: _____

Using Insight Window Designer and KSAM

By Felix Bearden

When I started the second of my major UCSD projects I was not completely happy with the outcome of the first. Even though I had several customers successfully using the product ToteTrac which is a Work-In-Process control system, I was not really happy with the operator interface using the screen, keyboard and bar code reader. Taking the easy route, I forced the operator to select between keyboard and bar code reader. However, once the reader was selected the operator could operate the on-line part of the system completely from the reader. I would have preferred to have both the reader and the keyboard on-line at the same time.

Because ToteTrac is a product intended to increase productivity, the number of key-strokes and disk access time become very important. To reduce the number of keystrokes, several features such as automatic field advance, were incorporated. To assure that data base accesses would not take excessive time I implemented a special purpose data base unit to handle the disk related problems. The detail records in the data base were ordered sequentially by location either on the production floor or in buffer storage. Each work station on the production floor is assigned an address including transporter number, station number, and position within the station. Buffer storage addresses included system number, bin number, shelf number, and compartment number. Knowing these addresses ToteTrac computed a record number for the detailed entry. Product, operation, or container identification number was

via indexes, accessing the data base by files where the entries are hashed into two fixed length files. Each entry contains the key, detail record address, and the collision links and flags.

The best testimony that the product is successful is that in one installation a material handler with no typing or computer experience moves as many as 600 30 lb. containers a shift, sometimes up to four containers per minute through the dispatchers station. The dispatcher must enter the data into the system, wait for the storage and retrieval system to deliver the container, move the container from storage to a conveyor and signal that the operation is complete.

I have never developed a product with which I am completely satisfied. The system even though accomplishing all of its design goals did not have the "color" and the market appeal that I had hoped for.

One of the problems I had with an installed system was that after about 60,000 transactions the customer experienced a degradation of performance and errors accessing data via the key files. Fortunately, the systems data base was recoverable from the detail data set so the customer did not suffer from large amounts of down-time. However, it did take a couple of weeks, several sleepless nights, and a little help from Knuth ("Art of Computer Programming, Volume 3") to track down the problem.

Another disadvantage of ToteTrac was that a sort of the data was required before the reports could be printed. One of my potential customers preferred using a popular data base management system to ToteTrac because the report started printing immediately even though ToteTrac would finish the complete report first.

When I first started KitTrac, a system to assist a warehouseman in preparing orders or kits for production, the problems with ToteTrac weighed heavily on some of the development decisions. An additional factor considered was the short development period scheduled. ToteTrac had been developed primarily for the IBM™ XT and the compatible machines. KitTrac was scheduled to be delivered on a Stride™ 440 and available on the 8088 based machines also. One of the systems purchased from SofTech came with a copy of the Insight Window Designer as a special deal. After running the demonstration and reading the manual I decided that this package would give the appearance that I wanted for KitTrac and solve some of the compatibility problems across machines.

I had been favorably impressed by the products I had received from SofTech and ordered a copy of KSAM for evaluation. After reading the manual and writing the KitTrac data base interface library unit and finding that the required functions were there, I decided that I would use KSAM instead of modifying the ToteTrac database library. (Note: Both the Insight Window Designer and KSAM products are available from Pecan Software Systems, Inc.)

The first system scheduled for delivery was to be implemented on a Stride 440 using the multiuser system and the Stride Wyse50 terminal. After the system arrived I discovered in the documentation that "Screen attributes are not allowed to be embedded in the screen (that is, the attribute should not occupy a character position)". Guess where the screen attributes are in the Stride terminal?

Fortunately, the editing and use of the Insight.Info file enabled me to describe the terminal to Insight programs well enough to effectively use the Stride terminal even though I could not use some of the terminal features which would make KitTrac more attractive on the bi-chromatic terminal. I have since added an interface to the Intercolor CT220 color terminal to KitTrac. My customer who is using the CT220s report a significant reduction in operator errors as a result of the bright red error messages when an error is detected.

Required in KitTrac as well as ToteTrac is the ability to use a standard bar code reader to enter data and make menu selections. Unless I am missing something (which is entirely possible) I cannot easily do this using the Insight Window Designer libraries. Because it is a requirement that KitTrac be able to accept bar coded input for Part Numbers, I implemented a special procedure which would accept data from the device (keyboard or bar-code reader) that sent the first character. I realize that this feature is not generally desired, but it could have made life a lot easier for

me. Some of the systems use a "Wedge" reader that fit between the keyboard and the processor in the IBM compatible systems. Some using the Stride insert the bar-code reader in series with the terminal.

It is difficult to decide which were the most useful features of the Insight Window Designer possibly because it is a very balanced system. The handling of the "pop up" windows is the most visible feature of the system but does not lend itself to the KitTrac application as well as I originally thought. The most windows available at one time in this application number only four (five counting the Help window). The menu selection capabilities are used heavily because of the large number of operator decisions that are required during the course of a days work. The ability to enter and edit data in a consistent manner without having to code the individual operations is definitely high on the list.

Even though the User Manual examples show entry of menu items, error messages, prompt messages, and normal messages by means of literal strings, I would recommend that all messages be included in the errors.info and messages.info files. In addition to saving space in the constant area, this approach simplifies editing messages without recompiling the program and implementing a multilingual application program.

The most difficult feature to use is the Menu Editor. Even though I use the Menu Editor for editing a file, I have to relearn its eccentricities each time. Because I am still not sure of whether the problem is in my code or the Menu Editor at this writing, I will only issue a warning about this procedure.

Most "software engineers" know that a good specification should be written before the software is started. I would suggest that the Help.info file be created before and during the implementation of a program using Insight Window Designer. This technique tends to assure better user interfaces and better references to the help file in the program. Better user interfaces are assured because if an operation is hard to describe, a programmer had rather spend his time programming a simpler interface than writing the description of a difficult one.

The only program bug that caused significant trouble was one in the "mm_push" and "mm_pop" procedures. These procedures bracket the occurence of a specific menu level. Apparently the "mm_push" allocated 5 words on the heap that are not deallocated by the "mm_pop" procedure. It took about two hours of test before the heap overflow message warned me that something was wrong. Preceeding the "mm_push" with a "mark" and following the "mm_pop" with a "release" corrects the problem. This problem may have been fixed in later versions of the product. I haven't removed the "mark"s and "release"s from the product.

A design problem that affects the operation of my product is that with multiple processes I cannot highlight the selected "button" on the menu when I want the user interface part of my program to go to sleep while another process is running and wake up after a button is pressed. The other process in this case is a host computer interface that is on-line to the host at all times.

The KSAM unit performs adequately even though it appears to make more disk accesses than the data base implementation in ToteTrac. However, the reports start printing immediately because the keys are kept in the order needed for the report.

Several characteristics of the package should be considered during the implementation of a program using KSAM.

1. The package requires all field lengths to be even (in bytes).

2. Data records kept by the package are typeless and data transfers are controlled by information in the data files. Therefore, if adequate space is not available in memory to contain the record, adjacent variables will be changed when the record is read from the data base.

3. Buffer size depends on data and key block sizes. The maximum number of records is a function of key block size. Therefore, a large number of records requires large buffers which takes from the available memory for the application. (Note: equations to compute buffer size and record limitations are included in the documentation.)

The problem I had with item 2 above reminded me of why I didn't like most implementations of Fortran and why I like Pascal better. I had been so spoiled by close typing rules and array bounds checking, it took more time than it should have to find the problem associated with a mistake I had made. The mistake being that I had allocated too few locations to hold the record **and its filler**.

I found the documentation to be complete for both the Insight Window Designer and KSAM. The presentation of the Insight Window Designer is very good and assisted in learning the product by supplying the information where it was needed. The products by their very nature are complex and were I to rewrite the documentation, I'm not sure I could do any better. Modern marketing realities do not permit us to publicize the limitations of our products. However, we probably should find a better way than we use now to bring these limitations to the attention of the user.

As far as licensing is concerned, my customers have no problem paying for the licenses after I explain the benefits of using a stable product over re-inventing one. Also it reduces the documentation that I have to write.

Am I happy with KitTrac? Well, happier. Transporting KitTrac from the 440 to the XT took part of one day. The colors are great. My XT/AT/PS2 Model 80 customers like it because it looks like a familiar off-the-shelf spreadsheet package. Post installation maintenance has been nil. Adding features to KitTrac is easier. And development time was reduced significantly.

For these reasons I have no problem recommending either of the packages for use in internal programs or products for sale in the general market place. For development purposes the Insight Window Designer is one of the best deals about to make a program have a professional look with an investment of time and a little bit of money.

Generic Deque Notes

by Peter M. Perchansky
412-1 Springside Drive East, Shillington, PA 19607

Don't let the ARRAY OF BYTE, ADDRESS, or ARRAY OF WORD scare you when you look at generic procedures or modules. Calling a generic procedure can be very simple.

An example module is given on the opposite page. The same module, or another, could have created a variable called stack of type Deques and used stack operations (push, pop, top, etc.). As you can tell, you don't have to worry about ARRAYs OF BYTE, or ADDRESSes, or ARRAYs OF WORDs. That's why using and writing generic procedures can be very helpful.

The main area to be careful of when writing or using generic procedures or modules is that you, the programmer, are responsible for keeping track of types you are storing or manipulating.

For instance, you could have used Enqueue (or Push) to place variables of type INTEGER (or whatever) in the same queue you placed variables of type PersonRec. That's where the danger lies: keeping careful track of what types you store.

Although you can mix types in a generic dequeue, it's highly advisable that you set up a separate stack, queue, deque, etc. for each data type you plan on manipulating or storing.

Please feel free to contact me via postal mail, CompuServe, FidoNet, or Interlink (language or Pascal Conference, they are working on adding a Modula-2 Conference) if you have any questions or comments.

CompuServe 71301,344
FidoNet 1:273/101
Interlink

The Deque module referred to here was presented in the Nov/Dec 1989 USUS NewsLetter.

```
MODULE Person;

FROM PMPDeque IMPORT Deques (* type *), Enqueue, Dequeue, InQueue;
FROM PMPDeque IMPORT CreateDeque, DestroyDeque;

TYPE
     PersonRec = RECORD
                   lastName      : ARRAY [0..24] OF CHAR;
                   firstName     : ARRAY [0..14] OF CHAR;
                   middleInitial : CHAR;
                   age           : CARDINAL;
                 END;

VAR
     person : PersonRec;
     queue  : Deques;
     ok     : BOOLEAN;  (* used to check for queue errors *)

BEGIN
     CreateDeque (queue); (* make queue NIL *)

     WITH person DO
          lastName := "Fieldings";
          firstName := "Robert;
          middleInitial := "M";
          age := 26;
     END;

     Enqueue (person, queue, ok); (* place Robert in queue *)

     IF NOT ok THEN
        (* handle queue overflow --- not enough memory --- error *)
     END;

     WITH person DO
          lastName := "Yoritomo";
          firstName := "Shakari;
          middleInitial := " ";
          age := 37;
     END;

     Enqueue (person, queue, ok); (* place Shakari in queue *)

     IF NOT ok THEN
        (* handle queue overflow --- not enough memory --- error *)
     END;

     WITH person DO
          lastName := "Fieldings";
          firstName := "Robert;
          middleInitial := "M";
          age := 26;
     END;

     IF InQueue (person, queue) THEN
        (* Robert is in queue *)
     ELSE
        (* Robert is not in queue *)
     END;

     Dequeue (person, queue, ok); (* serve person in front of queue *)
                                  (* Robert in this case            *)

     IF NOT ok THEN
        (* handle queue underflow --- queue is empty ---  *)
        (* possibility also exists that the type being    *)
        (* retrieved is not the same type that was stored *)
     END;
     ....
     (* and so on *)
     ...
     DestroyDeque (queue);    (* remove queue from memory *)
END Person;
```

A Vanilla Pascal Shell for the Macintosh

by David T. Craig
736 Edgewater, Wichita, Kansas 67230
(October 31, 1988)

INTRODUCTION

Converting vanilla Pascal programs which perform only character I/O that run on other machines to run on the Macintosh requires a lot of programming effort. To alleviate this effort I wrote a Macintosh program, MacShell, that allows vanilla Pascal programs to run on the Macintosh with minimal code modifications. To make MacShell and your Pascal program run on your Macintosh you need the source code for MacShell, the source code for your program, and a Macintosh development system that creates standalone applications that have full access to the Macintosh Toolbox.

MACSHELL (Modifications and limitations)

MacShell creates a Macintosh window that handles 24 line by 80 column character output. An Apple menu and a File menu are present. The Apple menu displays MacShell's name and version and the current Macintosh date and time. The File menu supports the Quit command which when selected by the mouse causes MacShell, and therefore the vanilla Pascal program, to terminate. MacShell is written in Lisa Pascal and was compiled using the Lisa WorkShop development environment. Compiling the source code with other development systems should not be difficult and should only require very minor source code changes.

To make a Pascal program run under MacShell requires that you perform four minor source code modifications to the Pascal program:

1) Change the token "PROGRAM" to "PROCEDURE".

2) Change the token "END." to "END;".

3) Replace all calls that clear the screen with a call to MacShell's routine clrScn_MacShell.

4) Add a call to MacShell's routine yield_MacShell within the main program loop.

The fourth modification is optional, but if you want the user of your program to access the Macintosh menu bar then calling yield_MacShell is critical. Within MacShell's source code three modifications must be done:

1) Change the INCLUDE directive to match the name of your Pascal source code file.

2) Place the name of your program in the init_MacShell call.

3) Call your program between MacShell's calls to init_MacShell and term_MacShell.

For a Pascal program in the file named my_Vanilla_Program.TEXT the following code section contains the necessary modifications. It initializes MacShell (i.e., creates the vanilla I/O

```
PROGRAM MacShell;

... MacShell variables and routines ...

{$INCLUDE my_Vanilla_Program.TEXT}

BEGIN { MacShell }
  init_MacShell('my_Vanilla_Program'); { initialize MacShell }
  my_Vanilla_Program;       { run the vanilla Pascal program }
  term_MacShell;                          { terminate MacShell }
END.  { MacShell }
```

window with the name of your program as the title), runs your vanilla program, and terminates when either your program ends or the user selects the Quit command in the File menu.
Two limitations exist within MacShell:

1) Character wrap-around does not function. This is due to my use of Apple's Macintosh runtime library as implemented for the Lisa WorkShop. A fix to this is possible, but is quite complicated and would involve major modifications to vanilla Pascal programs.

2) The screen cursor is never shown. This again involves Apple's Macintosh runtime library.

Once all the necessary modifications have been made to MacShell and your Pascal program source, then compile MacShell. The result will be your Pascal program running on the Macintosh inside a window. Note that the resource file for MacShell must also be compiled and merged with MacShell's compiled object code. This step is specific to the Macintosh development environment that you use.

SUMMARY

If you have a Pascal program that you want running on the Macintosh but don't have the time or ability to rewrite the program for the Macintosh environment, then MacShell is the solution.

Modula 2 Portability Module
by Peter M. Perchansky

Alex Kleider and I are working on creating a PortSys Module to allow for greater portability between Modula-2 Compilers.

If you use a Modula-2 Compiler (especially if you use more than one type), we could use your help.

What we are trying to do is come up with types, constants, and procedures that are compiler / system dependent and encapsulate them into a PortSys Module. By keeping machine / compiler information in one module, porting code from system to system, from compiler to compiler will be neater and less work.

Here is an example of the type of information we need:

JPI's TopSpeed Modula-2 has a type called BYTE, Scenic's Modula-2 does not have this type; instead Scenic can use SHORTCARD.

JPI's TopSpeed Modula-2's TSIZE returns type CARDINAL, while Scenic's Modula-2 returns type ADDRESS.

My PortSys would contain TYPE tByte = BYTE;, Alex's would contain TYPE tByte = SHORTCARD. My PortSys would contain TYPE tSizeType = CARDINAL;, and Alex's would contain TYPE tSizeType = ADDRESS;.
By then importing and using tByte and tSizeType instead of BYTE (or SHORTCARD) and CARDINAL (or ADDRESS), we now have a more portable foundation.
Please write to:

Peter M. Perchansky,
412-1 Springside Drive East,
Shillington, PA 19607

and let me know what constants, types, and procedures are specific to your compiler. Let me know how many bits per word, bytes per word, and bits per byte are on your system.

The help you provide us with today, may help your future portability needs.

Minutes of the USUS BoDs held on CompuServe
Dec. 11th, 1989

Logged on near the beginning of the proceedings were the following members:

User	User ID	Nod	Area	Name
1	73030,2522	BRK	Rm 1	ronw ron williams
2	76456,416	FLO	Rm 1	JonN
3	71016,1203	VCR	Rm 1	sfbp Stephen Pickett
4	72257,1162	RFI	Rm 1	Babb David R. Babb, Legal
5	73760,3521	SEA	Rm 1	Keith Keith Frederick
6	73447,2754	LIN	Rm 1	Henry Henry Baumgarten
7	75226,3643	DET	Rm 1	BobS A. Robert Spitzer and Felix Bearden
8	73007,173	RBC	Rm 1	Wm William D. Smith, NL Editor (outgoing)
9	71515,447	SMO	Rm 1	AlexK
11	76702,513	WPL	Rm 1	Harry Harry Baya
12	73167,2151	BTM	Rm 1	Wayne Overman from Baltimore
13	70260,306	DEQ	Rm 1	AHB hays busch
14	72747,3126	FLO	Rm 1	BobC Bob Clark, Treasurer
15	72767,622	POO	Rm 1	TomC Tom Cattrall
10	71626,1744	NYJ	Rm 1	don reed
12	73167,2151	BTM	Rm 1	wayne

Frank Lawyer logged on later but before any voting took place. It was really only with his arrival that there was a quorum of BoD members. Nevertheless he did not voice any concern about not knowing about the issues about which the voting was taking place so it is felt that this is a valid meeting.

The Chairman of the BoD was not present; the President was asked to chair the meeting and agreed to do so.

Felix Bearden had posted some proposals on MUSUS just prior to the meeting and they were read into the meeting as follows:

From Felix Bearden;
To the USUS Board 11-Dec-89 07:53:23

In view of the current crisis of USUS I recommend that we enter the meeting with the sole objective of determining the steps to take to have USUS survive and defer discussions of how to dissolve it (if necessary) to a later meeting.

To that end I would like to submit (or resubmit) the following proposals:

1. *That USUS raise the membership fee to $45 a year and make the Journal of Pascal, Ada, and Modula-2 the official journal of USUS. (see discussion of this proposal in USUS newsletter V3 N6)*

2. *That USUS more actively pursue an alliance with, or better still, a merger with MODUS.*

3. *That USUS more fully align itself with JPAM by soliciting members from the ADA community and offering the same services as are currently offered to Pascal and Modula-2 members.*

What can I do? Not as much as I would like! I'm still paying for two college educations and two weddings (girls family). But these things I can do.

1. *Serve on the board if elected.*
2. *Contribute at least one paper to a journal.*
3. *Work with those who are attempting to write an archiving utility in Pascal and Modula 2.*
4. *Contribute a program in M2 to the library.*
5. *Participate in the Stride and M2 sigs.*
6. *Assist members of USUS with technical problems.*
7. *Procure, review and comment on the M2 standard when printed.*
8. *Whatever else time and resources permit.*

These commitments are moderate. However, remembering the past successes of USUS, I believe that these are the kinds of commitments that made USUS of the past. We can't expect people like Hays, William, Harry, Eli, and Sam'l to do it all.

Sincerely,
Felix

Bob Spitzer then went on to "formally move that we increase dues to $45 and include JPAM with membership" with a second from Felix. There then followed some discussion as to who could vote. The consensus was that any member could offer an opinion but only BoD members could vote. Harry Baya suggested that we have to first figure out if we can keep USUS alive and that we need to be sure that someone will do Hay's job. He made a motion that Bob's motion be tabled and this was seconded by Frank Lawyer.

There then followed more discussions as to who should vote and with regard to a procedural matter as to the order in which issues should be subject to voting. There was also a suggestion by Stephen Pickett that the meeting be turned into an open meeting. This was felt by the chairman to be inappropriate and was overruled on the basis that we could not turn a BoD meeting into a general meeting without a BoD decision in this regard.

Stephen expressed the opinion that the time to wind an org up is NOT when we've money in the bank.

The motion to table was defeated by a 3/3 vote of the BoD members present. (Henry Baumgarten, Bob Spitzer and Frank Lawyer)

Next came discussion as to how the professional members would be affected by the new proposal. The consensus was that they would get the Journal for their current membership dues of $100.00.

Then the issue of student members came up. Some felt we didn't have any but William pointed out that this was not so and one attendee declared himself as being a student member.

Henry asked for Bob Clark's comments on the budgetary implications and for some of our visitor's reactions. Bob reported that:

We have $5785 in the bank as of the 12th. Putting out a newsletter costs about 650 and we have been putting out the NL about every two months. We won't make any money with only $25 for USUS. In fact unless we can retain our current memberships we won't survive too long.

BobC felt that the student membership should not include JPAM because past studies indicate that we need at least $45 on general membership to be able to produce our newsletter and pay our bills. Stephen felt that subsidizing student membership might be a good idea.

BobS then clarified that we need to pay JPAM $20 per subscription. $45 leaves $25 for USUS; we could compromise on the student membership - maybe $30?. And he indicated that Felix comments in favor of subsidizing this as well.

BobS further clarified that:

while the subscriptions would begin soon, prices would go up with renewal and presumably renewals would reflect satisfaction, also, new memberships from JPAM itself would pay the higher rate immediately, of course. But, if we do not retain our measly 300 members because of lack of satisfaction, $35 x 0 = $0; so this is a bid to turn around the slide by offering something meaningful, and have MORE members as a result.

William Smith then brought up the issue about how to deal with situations such as his own in that he had already paid two years in advance for JPAM.

In the mean time Frank inquired as to the possibility of raising the membership dues to $50 to cover the $25 to JPAM and $25 to USUS.

Tom Cattrall mentioned that he also subscribes to JPAM and did not see how tying into JPAM will help USUS:

Is it USUS's destiny to be no more than a subscription agent for JPAM? I feel that USUS should offer its own rewards or not bother to continue in existence. I also feel that more important than discussing this is what is to happen after Hays leaves. Perhaps while I was out earlier this was mentioned?

The motion was then called to a vote after further clarification by BobS:

If we can get JPAM for $20 as originally arranged; let me explain that with other journals that I get when I have also gotten a membership, the extra issues have just been tacked on the end; i.e. we would pay for members who already get this to get 12 months more for each year they renew. This should clarify this. If JPAM is › 20, we may need to get $50 as Frank proposed.

The same three BoD members all voted yes so the motion carried. William was interested in knowing how everyone else would have voted.
"no"s included BobC, William, TomC, Harry Baya;
"yes"s included Felix Bearden and Ron Williams;

Frank Lawyer then moved that:

we disband USUS in an orderly manner, Hays has sent an outline to all Bod and officers of the sort of things we should do to wind it up

and this was seconded by BobC.

Keith felt that:

disbanding is a bit hasty; why not just make the operation less of what it is in that we don't have to be with JPAM or anything and just be a small distributor of software for the p-system and related OS's and have a small newsletter.

All I wanted from USUS was access to the software. I could care less about the newsletter being out every other month.

BobS quickly moved to table this motion with a second from Stephen.

Of the three BoD members, Henry and BobS voted to table, Frank voted against.

Others in favour of tabling included Stephen; against were Jon Nevins, William, and BobC.

BobC volunteered that:

Unless we have someone to collect the dues and maintain the membership information before Hays finishes in just a few days we will not have the means to continue as a viable organization.

FrankL's comment was that:

from a practical standpoint, there is no way we can continue once Hays ceases operations, the rest is just a slow depletion.

Henry commented that:

For the benefit of the visitors, JPAM was not just an added journal. We expected to have a column in JPAM for USUS to use and to have some friendly interactions with Richard Weiner, the editor.

Harry mentioned:

Either we find some way to get the jobs done that Hays does, (Membership and dues) or I cannot see what choice we have but to close things down. We could announce that we are looking for volunteers for, say one month and then shut it down, or use the volunteer, or we shut it down now.

Felix Bearden indicated that he would talk to Hays ASAP regarding his duties and mentioned that BobS has volunteered to write the JPAM column. Felix indicated that:

Subject to approval by the BoD I am willing to a) maintain the membership roll, b) send renewal notices as memberships expire, c)collect dues and forward to Bob Clark, d) answer queries about membership, e) send welcome letters to new members and renewing members.

BobS :

If Wm maintains the PO box, and I do the JPAM liaison, and someone does the NL, this is enough to keep going.

Stephen Pickett indicated that he might be able to do the newsletter and the library. His qualifications for this job included possession of a US PO Box and a laser printer.

Tom Cattrall indicated that he had previously volunteered to do the NewsLetter. His position as NewsLetter Editor was approved unanimously by the BoD members present. (HB, BS, FL) It was hoped that Stephen would help Tom in this regard.

Before leaving the meeting, Keith (206/285-1576) volunteered to help out in anyway he could.

Stephen was prepared to take over responsibility for the library and Henry offered to help in this regard. Jon Nevins also expressed a willingness to get involved in the library.

The upcoming BoD elections were then discussed. An arrangement was made to get ballots out to the membership soon. Candidates included Felix Bearden, Tom Cattrall, and Stephen Pickett.

After an exchange of phone numbers the meeting was adjourned.

Letter To JPAM

Richard P. Friedman, Publisher
Journal of Pascal, Ada & Modula-2
310 Madison Avenue, Suite 503
New York, New York 10017

January 3, 1990

Dear Mister Friedman:

On behalf of USUS allow me to thank you for agreeing to have your Publication and our Society form a liaison. As I understand it, our agreement is that a subscription to JPAM is to become a benefit of USUS membership and that your organization is in agreement to provide such a subscription to each of our members upon receipt of notification of membership along with the member's name, address and a fee of $20.00 for each member who renews or joins as of now.

We will also endeavor to keep you informed as to which if any of these members are already subscribers and for those, it is our understanding that their subscription will be extended by a year for each such notification and payment.

If this is also your understanding, then let's plan to implement this agreement immediately and I am hopeful that you will be receiving such notices along with subscription fees in the very near future. I for one will renew at our new rate very soon and look forward to continuing to receive JPAM.

Yours very sincerely,

Alex Kleider, USUS President.

P.S. If you need to reach me directly, please make a note of my address and phone number.

1651 Stone Pine Lane,
Menlo Park, CA 94025
415/327-7916 or 415/780-2286.

Treasurer's Reports
by Robert E. Clark, Treasurer

November 1989

Bank Balance $6,944.85 10-31-89*

Income - November 1989

Dues:		(new/renew)
Student	0.00	0/0
General	485.00	1/12
Professional	0.00	0/0
Institutional	0.00	0/0
Other Income:		
CIS	0.00	
Library fees	55.00	
Publications	0.00	
PowerTools	0.00	
Total Income: $	540.00	

Expenses - November 1989

Administrator:	
CIS (2 months)	162.91
Telephone	19.50
Postage	59.47
Photocopies	5.77
Supplies	43.83
Other:	
Bank charge	3.00
Printing NL (2)	694.28
Mail from La Jolla	7.35
Mailing NL (2)	427.60
Refunds	2.75
Total Expenses	$ 1,426.46

Bank Balance $ 6,058.39 11-30-89

* Includes an adjustment of plus $50 for revised deposit amount.

December 1989

Bank Balance $ 6,058.39 11-30-89

Income - December 1989

Dues:		(new/renew)
Student	25.00	0/1
General	910.00	0/27
Professional	200.00	0/2
Institutional	0.00	0/0
Other Income:		
CIS (2 months)	75.53	
Library fees	16.00	
Publications	20.00	
PowerTools	0.00	
Total Income:	$ 1,246.53	

Expenses - December 1989

Administrator:	
CIS (2 months)	71.21
Telephone	1.04
Postage	97.24
Photocopies	34.59
Supplies	64.12
Other:	
Bank charge	2.00
Printing NL	0.00
Mail from La Jolla	1.50
Mailing NL	0.00
Reimbursements	261.88
Total Expenses	$ 533.58

Bank Balance $ 6,771.34 12-31-89

End of String By Binary Search
by Peter M. Perchansky

Most Modula-2 procedures to return the end of a string (EOS) use a brute force method similar to the following function:

```
CONST
    Null = CHR(0);

PROCEDURE Length (s : ARRAY OF CHAR) : CARDINAL;

VAR
    i, hs    : CARDINAL;

BEGIN
    hs := HIGH (s);
    IF String[hs] # Null THEN
```

```
                    (* string is full *)
        i := hs + 1
    ELSE
        i := 0;
        WHILE (i < hs) AND (s[i] # Null) DO
              INC (i)              (* walk through string looking for *)
        END;                       (* Null character
    END;

    RETURN i;
END Length;
```

Using this method takes n passes where n is the actual length of the string.

In "A better way to Combine Efficient String Length Encoding and Zero-termination," C. Bron and E.J. Dijkstra talk about a truly different alternative. That method being a binary search for the end of a string. Here's what they had to say:

"We postulate that, within the string container, all characters following the last significant character should be null-characters. It is then possible to locate the end-of-string position by a binary search (!!) because the contents of the string container monotonically 'increase' from non-null to null-characters!"

The following is my implementation of the algorithm to perform a binary search to locate the end-of-string position.

The first procedure, InitStr, should be called before any string operations. InitStr initializes all elements of a string to Null. The second procedure, LocateEnd, uses a binary search if the string is not full.

```
CONST
    Null = CHR(0);

PROCEDURE InitStr (VAR String : ARRAY OF CHAR);
(* Initializes all elements of String to Null *)

VAR
    i,
    hs      : CARDINAL;

BEGIN
    hs := HIGH (String);

    FOR i := 0 TO hs DO
        String[i] := Null
    END;
END InitStr;

PROCEDURE LocateEnd (String : ARRAY OF CHAR) : CARDINAL;
(* returns the location of the end of the string *)

VAR
    hs,
    high,
    low,
    mid     : CARDINAL;

BEGIN
    hs   := HIGH (String);
```

```
    IF String[hs] # Null THEN        (* String is full if EOS not Null  *)
       mid := hs + 1
    ELSIF String[0] = Null THEN      (* String is empty                 *)
          mid := 0
       ELSE
          high := hs;
          low  := 0;

          WHILE low < high DO        (* Use binary search to locate EOS *)
                mid  := (low + high + 1) DIV 2;

                IF String[mid] # Null THEN
                   low  := mid + 1
                ELSE
                   high := mid - 1
                END;
          END;

          WHILE String[mid] # Null DO  (* Make sure we reached a Null  *)
               INC (mid)
          END;

          WHILE String[mid - 1] = Null DO
                DEC (mid)              (* Make sure char before Null    *)
          END;                         (* is significant                *)
       END;

    RETURN mid;
END LocateEnd;
```

The following table lists a breakdown of the number of passes to find a given length using strings of TYPE String = ARRAY [0..79] OF CHAR; and TYPE String = ARRAY [0..254] OF CHAR;:

Length	[0..79] Passes	[0..254] Passes	Brute Force Passes
5	7	7	6
10	7	8	11
15	7	8	16
20	7	8	21
30	7	8	31
79	6	7	80

As you can see, strings containing few elements take longer when using a binary search to locate the end-of-string position.

However, strings containing more than 7 or 8 characters break-even in terms of time; ie. both methods are the same.

Using a binary search to locate EOS will produce the best time savings in cases where the string contains a lot of characters.

If you have any comments or suggestions you can send them via U.S. Mail to Peter M. Perchansky, 412-1 Springside Drive East, Shillington, PA 19607. I am also available on CompuServe via 71301,344, FidoNet Node 1:273/101, and Interlink.

A Review of Apple's MPW and MPW Pascal

David Craig (April 9, 1989)
9939 Locust #4013, Kansas City, MO 64131

Introduction

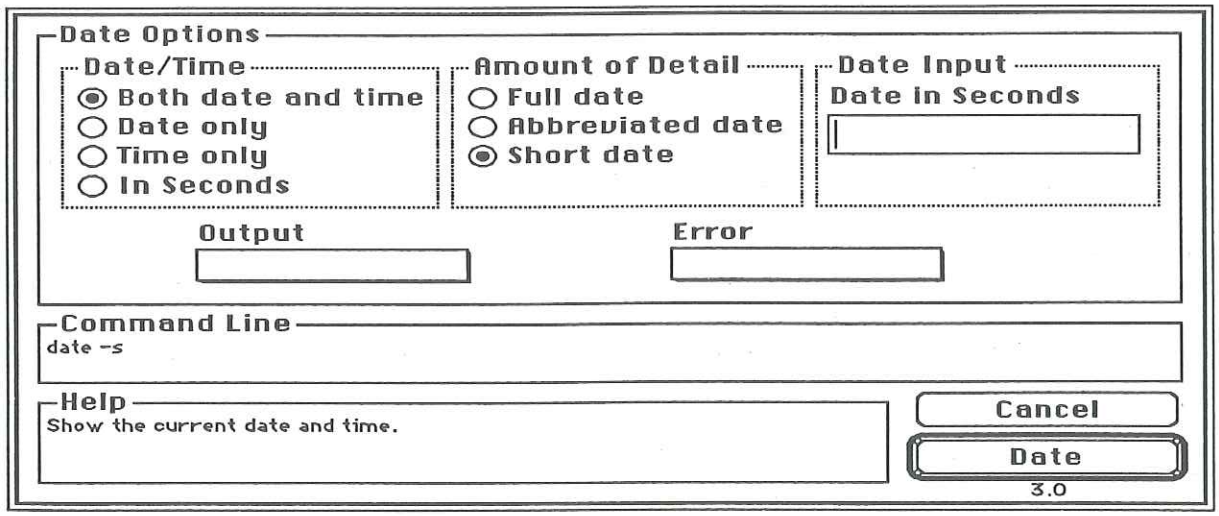
Apple Computer supports Macintosh software development with its Macintosh Programmer's Workshop (MPW). The current version is MPW 3.0. This programming environment is based upon the MPW Shell, a multiwindow text editor with a built-in UNIX-like command language. Commands are entered from within any window and are executed by selecting the command text with the mouse cursor and pressing the Enter key. With MPW programmers create programs for either the Macintosh desktop or for the MPW environment (these MPW-only programs are called "tools" by Apple). MPW combines command and tool input and output using automated scripts, sequences of commands that are treated as a single command group.

MPW Tools

MPW provides about 100 tools to aid in software development. The major tools are:

- Project management (Projector)
- Resource compiler and decompiler (Rez and Derez)
- Linker (Link)
- Make facility (Make)
- Tool dialog interface (Commando)
- Symbolic Application Debugging Environment (SADE)
- Performance-measurements
- Language compilers

Projector is a project-management system that can be customized for either a single programmer or a large programming team. **Rez** takes a text source file containing resource definitions and compiles them into their binary form for inclusion into a Macintosh application. **Link** links object files that originate from different languages. Make tracks source file dependencies so that all the needed files are correctly compiled at compile time. When one component of a program is modified, **Make** automatically updates all other parts of the program that depend on it. These updates consist of compiles, assemblies, links, and resource compiles. **Commando** is a graphical interface for MPW commands. It displays the command's options and parameters in a dialog box making it easier to use MPW tools. For example, the Date command, which displays the current date and time, has the following Commando dialog:



When the Date button is pressed the command "date -s" is output. When executed the date and time like "4/9/89 5:30:37 PM" appears. SADE is an interactive debugger for high-level languages like Pascal and C. It is a stand-alone application that runs under MultiFinder along with the program you want to debug. SADE allows you to stop your executing program at any time to examine variables, set breakpoints, and single-step through any portion while displaying the source code. Performance-measurement tools measure the performance of your programs. MPW supports many Apple and third party language compilers. Apple provides tools for Pascal, C, and 680X0 assembler. TML Pascal, Aztec C, SemperSoft Modula-2, and Language Systems FORTRAN are also available. Extensive on-line help for every tool and command is available with the Help command.

MPW supports a variety of languages that create stand-alone Macintosh desktop applications, desk accessories, drivers, and MPW tools. The languages supported by Apple are Pascal, C, and 680X0 assembler. Complete access to the Macintosh ToolBox ROM is provided by extensive interface libraries.

MPW Pascal

This Pascal compiler evolved from Apple's earlier Lisa Pascal for the Lisa computer. As such, MPW Pascal is very similar to the Lisa Pascal compiler which I reviewed in the **USUS NewsLetter** (January 1989). MPW Pascal produces native 680X0 code for the Macintosh. Four features set it apart from the Lisa Pascal compiler. MPW Pascal identifiers are significant to 63 characters versus the Lisa's 8 character limit. This allows programmers to use names that are more meaningful. MPW Pascal code optimization is extremely good. For example, the following code fragment (z is a 4 byte integer) compiles correctly into one machine instruction:

```
z := 60*60*60;      MOVE.L  #$00034BC0,-$0004(A5)
```

MPW Pascal supports a set of object-oriented extensions, known as Object Pascal, that provide you with the ability to write object-oriented programs. Effective use of Object Pascal is through Apple's MacApp toolkit. This toolkit provides a framework of object classes that implement a skeleton Macintosh application. MPW Pascal supports generation of 68020 machine code and 68881 FPU (Floating Point Unit) code for the Macintosh II series. For machines without FPU MPW Pascal generates SANE (Standard Apple Numeric Environment) code. SANE, a superset of the IEEE Standard 754 numerics, provides accurate, extended-precision, floating-point arithmetic.

680X0 assembler code and C code can also be used by MPW Pascal programs through the EXTERNAL directive and the C directive. The following source fragment declares routine K_R to be a C routine:

```
PROCEDURE K_R (parm: INTEGER); C; EXTERNAL;
```

The INLINE directive allows you to write explicit 680X0 hexadecimal machine instructions within a Pascal program. The following routine has an INLINE directive:

```
PROCEDURE OS_Trap (tos: LONGINT); INLINE $A9ED;
```

Three MPW tools assist in the documentation of Pascal programs. PasMat formats Pascal programs into a standard format. PasRef creates a listing of a program followed by a cross-reference listing of all the identifiers. ProcNames produces a listing of all the routine names within a program.

Documentation

The documentation for the MPW Shell, its tools, the 680X0 Assembler, and the Pascal compiler is extensive. The Shell and tool volumes contain 990 pages, the Assembler manual has 325 pages, and the Pascal manual has 394 pages. All the manuals have similar organization and good indices.

Summary

My experiences with the MPW Shell and MPW Pascal have been positive and productive. The tools provide a very professional development environment that is guaranteed to contain all the latest Macintosh software extensions. For serious Macintosh software developers MPW is hard to beat.

That's all, folks

SOFTWARE DONATION STATEMENT
As of: 4/18/87

The undersigned hereby make the computer software or other software or information identified on the reverse of this page (the "Program") available for inclusion in the USUS Software Library (the "Library") with the representation and on the terms, and subject to the conditions, set forth below. The singular form is used as a matter of convenience and includes the plural where applicable.

1. The undersigned represents that the material furnished, except for those portions identified elsewhere in these documents, was created either:

(a) Personally by the undersigned who is a natural person who does not employ others or -

(b) by the the employees of the undersigned that is a business, whether incorporated or not.

This means that the undersigned is the "author", for copyright law purposes, of the material so donated. The only material excepted from "authorship" is indicated on an additional page (Exhibit A) which is attached hereto as to:

(a) Material believed not copyrighted by anyone, identified by source and with reason for such belief, and

(b) material copied from a program currently available in the USUS Software Library, identified by Volume and name.

IF THERE IS NO EXCEPTED MATERIAL, WRITE "NONE" HERE _____

The undersigned will hold USUS, Incorporated and its representatives harmless from and indemnify them for all loss, cost and expense they incur or suffer as a result of, or in connection with, any claim, suit or other action asserting that any material of any Program, or any Program, or its copying or use, infringes the intellectual property rights of, or is a plagiarism from any person.

2. The undersigned acknowledges that the Programs are subject to review and will not be included in the Library if a review is negative. Reasonable effort will be made to furnish the undersigned a copy of any negative review, upon request, but such furnishing can not be guaranteed. Programs that are not original, are not provided in source and in machine readable form, or that duplicate a function already provided adequately by a Program in the Library are not eligible for inclusion in the Library. Programs accepted are subject to revision by the reviewers to improve their function or utility.

3. Programs in the Library may be made available to USUS members for non-commercial uses in accordance with current Library policy. If such policy is changed, publication of a general notice of such change in a USUS publication will be deemed sufficient notice to all who have submitted Programs. If a Program remains in the Library after such notice, it is presumed that the undersigned accepts such policy change. Programs may be withdrawn from the Library at any time on written notice to the Software Library Distribution Chairman, but such action can not be effective for Programs already released to members.

The undersigned acknowledges that neither USUS, Incorporated nor any of its representatives is liable to the undersigned for any harm suffered or incurred by the undersigned as a result of either the failure of any person to whom a Program was made available by the Library to comply with the provisions set forth in the applicable Software Order Form or any use of a Program.

4. The undersigned is not entitled to any money payment whatever for making the Programs available in the Library and understands that material furnished will not be returned

(Signature)_____ Date_____

(Type full name)_____

(Signature)_____ Date_____

(Type full name)_____

Program Submitted _____ (name)

Program function or description in brief_____

Administrator Sez
by Felix Bearden

I feel a lot like "Klinger" on M.A.S.H when he was designated to take over the job of company clerk. "I'll never be able to do the job Radar did." Hays has done an excellent job as Administrator during his tenure in office (which probably is why no one would step forward to fill his position) and this struggling programmer is going to have difficulty keeping up.

Thanks to all of you for the encouragement and the offers of help. You can be sure I will call when I figure out how you can help.

USUS, as evidenced by the last several issues of the newsletter as well as this, is in the midst of dramatic change. We are moving into a period of challenge and opportunity. I believe we can seize the opportunity and be a part of moving USUS into the 90's, adjust to the needs of our members, and remain an effective organization in furthering the goals of software engineering.

I am excited about our arrangement with JPAM. The BofD's decision to include the Journal as a benefit of USUS membership helps answer the question "What do I get from USUS for my dues?" I am excited about the renewed activity on MUSUS. This is the place to discuss and learn about Pascal, Modula-2, and other languages that can be considered transportable and structured. With the standardization activities now in progress on Modula-2, I am excited about the part that USUS/MUSUS can play in influencing that standard.

And possibly, in time, like "Klinger" I can make this job mine, ... and _____, ... and _____, ...

Felix

A leftover half page is too much temptation to resist so I'm inaugurating the Bug Box. A feature devoted to entertaining, enlightening, and filling up unused page space. Send me your contributions to share with others.

Bug Box by Tom Cattrall

To start out, I'll bring out just about my favorite story. It happened back in the 60's to a friend, co-worker, and college roommate of mine.

He had written a Fortran application for some users but this compiler would issue a message something like "End Of Fortran Execution" whenever a program terminated normally. Feeling that the users wouldn't know what to think of this, he wrote a short (very short) assembly language routine that just called the operating system to terminate. Calling this should bypass the Fortran message. His routine looked like:

```
MYEXIT  SBJP    Exit to operating system
```

That's all. Just one instruction. And it didn't work!

This was on a CDC 3300 which had no stack register. Subroutine return addresses were stored in memory at the entry point and execution started at the following location. His one word program should have been two words long:

```
MYEXIT  BSS  1  Entry point here
        SBJP    Exit to operating system
```

He took a tremendous amount of ribbing over this. Ever since then, when someone tells me that a program (or modification) is so short and simple that they don't need to test it, I tell them this story.

From The Editor
by Tom Cattrall

When I joined USUS in 1984, I sent off my check and application form and then heard nothing. After awhile I asked the MUSUS sysop if I could get access to MUSUS and he said yes. That was the only response for some time. Later the occasional journal appeared but not much else ever happened for quite a few years.

Today things are very different. New members are promptly greeted with letters and a welcome disk. Old and new members receive a regularly published newsletter. For these remarkable changes we owe a lot to the outgoing USUS administrator Hays Busch, and to the outgoing newsletter editor William Smith. Not to slight the efforts of others, but these two have made a big difference in USUS. So I wish to thank them and wish them well with their newly found spare time.

Felix Bearden is the new USUS administrator and I am the new newsletter editor. I hope that we can continue the fine work that our predecessors have accomplished. I also hope that more members continue to get involved with USUS activities. For those that aren't sure what they could contribute, the following list, derived from Felix's list given on page 10, may help inspire you:

1: Serve on board or other position
2: Contribute to this newsletter
3: Work with others on group projects
4: Contribute programs to the library
5: Participate in SIGS (special interest groups)
6: Assist others by answering questions raised by letters and on MUSUS

I would like to see the newsletter become a strong forum for Pascal and Modula 2 articles. USUS remains as probably the only forum for p-System users and information specific to the p-System is always appropriate. But many members now use other systems as well. Articles written in such a way that any of the readers can use the information even if they use a different system or a different language are especially desirable.

If you worked out some new algorithm in your last project, consider writing it up so that the rest of us can get ideas from it for our own bag of tools. Present it with explanations and some code fragments and/or procedures. The ideas will help us all whether or not we use the same language as you.

If you have developed a program that you would like to share, submit it to the USUS library and write up a short article telling what the program does, how to use it, and how it works.

MUSUS discussions these days lean much more towards Modula 2 than Pascal. I have strong interests in Modula 2 as well. I would like to provide more Modula 2 coverage in the newsletter but in such a way that Pascal readers will still benefit.

Besides programming type articles, reviews of software, hardware, and books are of interest too. And how about some more entertaining items such as describing the dumbest bug you ever created? I'd like to start a section where we can share some less serious anecdotes of what has gone wrong. They can be anonymous if you wish. And last but not least, letters are always welcome.

The bottom line is that you, the readers and contributors, will have more say on what appears in

the newsletter than I as the editor.

Group Projects

There has been an ongoing problem on MUSUS concerning how to upload files to the forum. Many forums have mainly MSDOS users and they use arc format files for uploads. (Some newer file formats are gaining ground). Other forums have utilities specific to their users and systems. But the p-System doesn't have an archiving utility.

The characteristics of the archiving utilities are (in roughly decreasing importance):

1: Package files together into one archive file with original file names included.

2: Compress the archive file so that modem transfer time is reduced. This also results in less disk space usage.

3: Original file attributes are saved when a file is packaged, and restored when a file is unpacked.

Several of us would like to implement something for use on MUSUS. It should be written in relatively portable Pascal and Modula 2 versions. In the next issue of this newsletter I hope to write up a design to get this project going. I can provide a design and algorithms but would like to rely on others to fill out the details. Especially in the areas of user interface and operating system interactions. Get in touch with me or leave a message on MUSUS if you wish to contribute in any way.

Submission Guidelines

Submit articles to me at the address shown on the back cover. Electronic mail is probably best, disks next best, and paper copy is last. If your article has figures or diagrams, paper copy is probably the only practical method.

You can send E-Mail to my Compuserve ID: 72767,622, or indirectly from internet: 72767.622@compuserve.com. For disks, I can read Sage/Stride/Pinnacle format disks. Also, any MSDOS 5.25" or 3.5" disks, and 3.5" Amiga disks. If someone wants to send Mac format disks I could probably get someone to translate them into something I can use. Whatever you send, please mark on the disk what format it is. That will save me a lot of guesswork.

Text should be plain ascii rather than a word processor file. It can have carriage returns at the end of all lines or only at the ends of paragraphs. What you send doesn't have to look pretty. I will take care of that. My spelling checker will take care of spelling errors too. If you want special formatting use the following conventions:

1. _Underline_, put an underline character at each end of the section to underline.

2. *Bold*, put a star at each end of the section to **bold**.

3. ^Italics^, put a caret at each end of the section to be set in *italics*.

4. ??Special requests??, such as ??box next paragraph?? should be surrounded with "?? ??".

USUS Membership Information

Student Membership	$ 30 / year
Regular Membership	$ 45 / year
Professional Membership	$100 / year

$15 special handling outside USA, Canada, and Mexico

Write to the La Jolla address to obtain a membership form.

NewsLetter Publication Dates

Issue	Due Date For All Newsletter Material
Mar/Apr 90	March 9, 1990
May/Jun 90	May 11, 1990
Jul/Aug 90	July 6, 1990
Sep/Oct 90	September 7, 1990
Nov/Dec 90	November 9, 1990

USUS Administrator

Felix Bearden
Compuserve : 74076,1715
Internet : 74076.1715@compuserve.com

NewsLetter Editor

Tom Cattrall
Amity Software Inc.
7600 Seawood Road SE
Amity, Oregon 97101
503/835-1613
Compuserve: 72767,622
Internet: 72767.622@compuserve.com

USUS
P.O. BOX 1148
LA JOLLA, CA 92038

ADDRESS CORRECTION REQUESTED
FIRST CLASS MAIL

[redacted]