68xxx NEWS

Issue #6 - November 1988

How to Get This Newsletter

Several users have asked how to get their name on our mailing list so they can receive this Newsletter.

If you note the fine print in the box at the bottom, we have decided to make the Newsletter available on a subscription basis. There is now a charge of $10 for a four-issue subscription. We intend to publish one issue every three months or so, so this translates to roughly a one-year subscription. BUT - isn't there always a BUT? - we will send a free four-issue subscription to those Star-K customers who return the User Registration form found at the end of their manuals.

We can't overemphasize the importance of your returning the User Registration form from the back of your SK*DOS manual. We are gradually expanding our 68xxx Newsletter to contain more information, and the User Registrations are our primary mailing list. Remember: no Registration, no Newsletter.

We are making somewhat of an exception with this issue, since Peripheral Technology has been kind enough to send us one set of labels for their SK*DOS and HUMBUG customers. We are thus able to send this issue even to those users who have not returned their Registration forms. But they will not get the next issue unless they return their registration forms.

How to Get Upgrades

We have regretfully decided to change our upgrade policy.

Up until now, upgrades have been free if you send in your original disk along with a self-addressed and stamped mailer for use in returning the disk. We have suddenly realized just how much time it is taking us to make those upgrades - it's not so bad for one or two upgrades, but it is frightening to think of updating the many copies of SK*DOS out there. We have therefore decided that we will have to subcontract some of that work to outside people. That, unfortunately, costs money.

Effective immediately, upgrade copies of SK*DOS will cost $5. You must return your original SK*DOS disk for the upgrade.

We really apologize for taking this step, but it is quite necessary. By way of comparison, here is what some vendors in the IBM PC world are charging for upgrades:

IBM PC-DOS version 3.3: New price $125, upgrade $125.
IBM PC-DOS version 4.0: New price $150, upgrade $95.
Borland Turbo C version 2.0: New price $150, upgrade $50.
Inset Systems' Inset version 2.2: New price $99, upgrade $40.
Prosoft's Fontasy version 3: New price $130, upgrade $50.

There was an interesting letter-to-the-editor of a large PC magazine recently, written by the president of one of the really large multi-million-$ software vendors. He discussed how airlines have gotten themselves into a serious problem by offering mileage clubs. In an effort to woo more customers, airlines essentially offer free trips in the future in exchange for your flying with them now. What this does is to exchange a slight increase in current revenue for a tremendous future obligation which must at some time be paid off. Because of their limited seat capacity, many of those future free flights wind up bumping fare-paying passengers off planes, with the result that these mileage clubs represent a very large future expense which could potentially bankrupt some of the airlines if everyone should decide to cash in on their free flights at the same time.

He then likened this to the plight of software vendors who offer free support and upgrades. While this does help increase present sales, it places a future burden on the vendors which can easily bankrupt them if it represents a greater amount of money than they gain in sales.

I'm not sure that his point of view is entirely valid, but it is an interesting thought.

68xxx NEWS is published and copyright © 1988 by Star-K Software Systems Corp., P. O. Box 209, Mt. Kisco, NY 10549, (914) 241-0287. The subscription price is $10 for four issues; purchasers of Star-K Software 68xxx products are entitled to a free 4-issue subscription upon receipt of their User Registration form. Readers are invited to contribute letters, articles, or other material for publication.

New Fax Number

Several overseas corporate users have asked us to install a Fax machine so they can communicate with us easier and cheaper. It turns out to be a very useful device, and some of you may want to use it to contact us also. Our Fax number is (914) 241-8607. And, of course, don't forget our BBS at (914) 241-3307. Another 68000 BBS which also supports SK*DOS is Mike Evenson's; the number is (817) 488-8398.

COPY Command Improvements

The S option of COPY has been changed. Whereas it used to mean "second copy", now it means "since". It is now used to copy only files generated on or after a date specified after the "from" and "to" parameters. For example, the command `COPY S 0 1 6-28-88` would copy all those files dated June 28th, 1988 or later. Since no directory is specified, only files in the root directory would be copied in this case. The S option can, however, also be combined with several other COPY command features to make more complex commands. The S option can not be combined with the F (copy by file number) option, since the F option will take precedence. (The F and S options are often used for backing up a hard disk to floppy disks. F is used for a complete backup to copy groups of files at a time to separate floppy disks; S is useful in making incremental backups of just those files which have been changed or newly generated in the last few days.)

In the "since" date, the month, day, and year must appear in that order, and may be separated by hyphens or slashes, as in 6-28-88 or 6/28/88. Note that the date must appear after the "to where" parameter, but before the "match-list", if one is used.

Reading Foreign Disks

Several readers have written programs which greatly improve on the TOMSDOS and FROMSDOS commands which come with SK*DOS. But to be able to read both sides of an MS-DOS disk, we have had to make the following change (which, incidentally, will also make it possible to read and write other foreign disks).

Variable FOTHER, which is at DOSORG+$151, tells SK*DOS what format disks it is using. If FOTHER is 0, then it is using SK*DOS disks; a non-zero number tells it that a different format disk is being used, and the exact number tells it how many sectors there are per side. For example, MS-DOS disks use nine 512-byte sectors per side, so FOTHER should be set to 09 for those disks. Programs which translated one format disk to another should change FOTHER back and forth between the two values. Upon return to SK*DOS, FOTHER is automatically reset to 0.

A set of MS-DOS read and write utilities written by Mike Evenson is available for downloading from our BBS; if you live closer to Texas, you may also get it from Mike Evenson's BBS (see above for number).

Turning off Type-Ahead

Under some conditions you may want to turn off keyboard typeahead. For example, some older programs may do their own keyboard access and do not therefore empty out the typeahead buffer. When you exit that program, you then return to SK*DOS with the full buffer, and SK*DOS goes wild trying to execute whatever garbage is in the buffer. No harm done, but you sure get a lot of error messages!

Typeahead is controlled by a flag called TPDFLG, located at 5645(A6). In a typical implementation this will work out to be location $2A0D (since A6 will usually point to $1400 and 5645 is equal to $160d.) You can disable typeahead with a POKE 2A0D 1 command. Typeahed is turned off whenever the flag is non-zero.

The flag can be set manually with the POKE command, or else you can automate the process one of two ways. One way would be to use a batch file; the other is to append a flag-set command to the program itself.

For example, suppose you have a program called HHHH.COM which does not like to work with typeahead. Do a LOCATE on it with the - option. Suppose LOCATE says "Absolute Address: 0000-1235" and "Absolute Execution Address: 0000". That tells us that the program starts at 0000, and that memory above 1235 is free. Then edit and assemble the following program:

```
        ORG $1236
START   MOVE.B #1,$2A0D      Set Flag
        BRA.L $0000          Go to HHHH.COM
        END START
```

Now append this short program to the end of HHHH.COM. The resulting program will now set the flag and then jump to HHHH automatically.

Which Way is Forward?

A recent letter from Bob Jones (the author of RBASIC) brings up an interesting topic. Let me quote part of it:

"I feel you're making a mistake by moving toward multi-tasking/multi-user (or even larger sized sectors) with all the attendant problems... If users need this they already have OS-9/68K together with a wealth of supporting programs. After all, the whole appeal of [SK*DOS] is that it's so friendly to use, and I see the latest implementation of subdirectories as an early step away from this friendliness...." Now the mere fact that Bob has written RBASIC shows that he is not a beginner to computers. He is not pushing for simplicity because SK*DOS is too complex for him - his comments are based strictly on his belief that simpler is better.

On the other hand, we have other users who have since the beginning pushed to add more complex features to SK*DOS - things like environment strings, more complex .BAT file structures, more powerful device drivers, typeahead buffers, subdirectories, true pipes, error handling within .BAT files, different file structures, relocating assemblers and loaders, etc. In fact, we've had one user who even resorted to personal insults and four-letter words in his frustration that SK*DOS wasn't more like Unix, and that he couldn't get the source code so he could add Unix-like functions to it.

There's a lot to be said for both points of view. What do you think we should do???

Speaking of RBASIC...

RBASIC has now been officially released. It is a full Basic interpreter, and quite fast. We have decided not to get involved with selling and supporting it; hence it is available directly from the vendor, Micronics Research Corp., 33383 Lynn Ave., Abbotsford BC, Canada V2S 1E2. The price is $99.95 (or $125 Canadian).

and C...

The C compiler has also been officially released by Computer Systems Consultants, 1454 Latta Lane, Conyers GA 30207, (404) 483-4570 or -1717. It is priced at $99, and also available from us or from Peripheral Technology.

An Offer of Help

Karl Lunt of Peoria, Arizona, recently dropped us a note that he has ported SK*DOS to his machine with success, and likes the result. He ended his note with the comment "If you know of anyone attempting to bring SK*DOS up on a Convergent Technologies mini-frame that needs some help, please give them my address." Let us know if you need his help.

Atari Anyone?

Speaking of porting to other machines, several potential users have asked whether we know of anyone who has ported SK*DOS to an ATARI ST. We don't - do you?

About Languages

Fairly often, the question comes up of whether MS-DOS programs can run on a 68000 system under SK*DOS. We thought that this might be a good topic for discussion.

Computer programs can be written in one of three types of languages:

a. Machine language is the only language a normal computer understands. A single machine language instruction does very little, so thousands may be needed for even simple programs. Each different microprocessor has its own machine language, and a program written for one will generally not run on another (unless the processors are intentionally similar. For example, a 68008 program will run on a 68000 processor, but an 8088 program will not.)

b. Assembly language is very similar to machine language, but uses words instead of machine language's numbers; it too differs from one processor to another. Because it uses words, it is easier for us humans to use; a translator program called an assembler translates it to machine language. Assemblers are fairly simple programs since assembly and machine languages are quite similar in structure.

c. Higher level languages such as Basic, Pascal, or C, are very different from machine or assembly language. Rather than being tailored to the hardware characteristics of the computer, they are tailored to the job that needs to be done. A quite complex translator, called an interpreter or compiler, is needed to translate to machine language; some compilers translate first to assembly language and let an assembler finish the job by translating the rest of the way to machine language.

A program written in machine or assembly language will only run on the computer it is written for; a program in a higher level language will run on many different computers (with some changes), provided that an appropriate translator is available. Commercial software vendors prefer to write in a higher level language such as C or Pascal, since they can then sell that program

for more than one computer. Writing in such a language is also easier, since higher level language statements are more powerful, and so fewer are needed to do a specific job. They then translate the program for the specific machine, and sell only the machine-language version. But there is a disadvantage - programs initially written in machine or assembly language are substantially faster, and require less memory. As more and more programs are written in a higher level language, computers have to get bigger and faster to keep up.

In SK*DOS, we currently have available two assemblers, a Basic interpreter, and two C compilers; compilers for Modula and Whimsical should be available soon. Thus it is possible to take a Basic or C program from another computer, and translate it to work under SK*DOS. Many of the programs available for SK*DOS have gone through just this route - for example, the User's Group has translated many programs originally written in C for Unix, so they now run under SK*DOS.

Although MS-DOS programs are also often written in C or other high level languages, they are not sold in their original language; only the machine language forms are released for sale. Once translated into machine language, programs designed for an 8088 will not run on the 68000. So, though we can read and write MS-DOS disks under SK*DOS, we can move programs over from MS-DOS but we cannot run them.

But there are two ways of running an MS-DOS program on a 68000 system after all. One way is to add 8088 circuitry to the 68000 system; in essence, this combines two computers into one, sharing memory or I/O equipment, but otherwise maintaining a strict separation of the two systems. For example, Peripheral Technology has an 8088 card which plugs into their PT68K-2 computer and allows just this. MS-DOS programs then run at exactly the same speed they would on an 8088 PC.

The other approach is through software. It is possible to write a program which makes one processor 'understand' the language of another (though at greatly reduced speed.) For example, our SK*DOS09 program allows 6809 programs to run on the 68000. SK*DOS09 is essentially an interpreter program, which takes the 6809 program apart, one instruction at a time, and uses anywhere from ten to twenty 68000 instructions to simulate each 6809 instruction. This explains why it is so much slower.

It is possible to execute 8088 MS-DOS code on the 68000 in exactly the same way, but it is much more complicated for two reasons: the two processors are so different from each other, and also the two operating systems are so different from each other. In essence, it is necessary not just to simulate the 8088 on the 68000, but also to adapt the DOS to run on a different computer. Since we were intimately familiar with the 6809 - and had already written a version of SK*DOS for it - we were able to write SK*DOS09 in just a few weeks. But simulating both the 8088 and MS-DOS on the 68000 might take several man years to do from scratch. It is far beyond our resources.

We therefore contacted one company which specializes in such programs, and which had already written such a simulator for the Atari, another 68000 computer. We felt that they might be able to adapt their Atari program with fairly minimal work. They informed us that they would not even consider the job unless we

paid them $250,000 up front, and also guaranteed sales of at least $500,000 the first year. That was the end of that...

RS-232 Wiring and CMODEM
by Sidney Thompson

(Editor's note: Though Sidney's article was written specifically for the PT68K-2 computer, the basic ideas really apply to most computers.)

The world is made up of two kinds of machines, talkers and listeners. Since the beginning these have been known as Data Terminal Equipment (DTE) or talkers and Data Computer Equipment (DCE) or listeners. Examples of DTE equipment are Video Terminals, Teletypes, and Serial Printers. Computer mainframes and mini computers are examples of DCE.

The PT68K-2 is a DCE type device. After all it is a computer. What this means is that you use a terminal to talk to it. This can be either a normal ascii terminal or the IBM keyboard and Hercules compatible card. The RS232 connectors on the back of the PT68K-2 are wired as DCE so that a "straight thru" cable can be used to connect a terminal to the computer. This cable would normally be a cable with a male connector on both ends. This wiring also allows the use of a standard serial printer connected with the same cable to be used without wiring changes.

When it is decided that we are going to use a modem to talk with the outside world we begin to have problems. The modem is wired as a DCE device. This is because the modem was meant to be connected to a terminal or serial printer. The modem looks like a remote connector from a distant mainframe. This connection then satisfies the requirement that the DTE terminal talk to the DCE device since the modem is simply an extension of the mainframe connector. Unfortunately when we try to have the PT68K-2 talk to our modem we quickly find out that we have a problem.

The PT68K-2 has the connectors on the back wired from the six (6) pin "berg" plug to the 25 pin RS-232 connector (also known as a DB-25 connector) as follows:

Berg	DESCRIPTION	DB-25
1	Request to Send (RTS)	8
2	Index pin	
3	Signal Ground	7
4	Data Terminal Ready (DTR)	20
5	Transmit Data	3
6	Receive Data	2

As seen from the front of the board, the 6 pin connector counts its pins as:

(1) (6)
(2) (5)
(3) (4)

In most cases pin 1 will not be connected on the connectors built by Peripheral Technology, since it is not actually needed for the operation of a terminal or printer with the system. It is needed for use with a modem since it acts as an indicator to the modem that the system is operational and ready to accept data.

There are two methods of wiring the connector of the PT68K-2 that will work with a modem. If you never plan to use the port for any purpose other than as a modem port you can wire a male RS232 connector as follows:

Berg	DESCRIPTION	DB-25
1	Request to Send (RTS)	20
2	Index pin	
3	Signal Ground	7
4	Data Terminal Ready (DTR)	8
5	Transmit Data	2
6	Receive Data	3

You can then use a modem cable like that used by the IBM PC. This is a straight thru cable with a female connector on the computer end and a male connector on the modem end. It must be remembered that with this wiring you can not attach a terminal or printer to the connector. The reason is that all of the data and control lines are reversed.

The other option is to leave the PT68K-2 connectors as they are and modify your cable that is connected between the computer and the modem. This modification will produce what is known as a "null modem" cable. That means that the data and control lines are crossed or transposed within the cable. This type of cable is required when you wish to connect two DCE devices together. The "null modem" cable has pins 2 & 3 and 8 & 20 transposed in the cable connectors. By this it means that pin 2 on one end is connected to pin 3 on the other end of the cable. The other pins are swapped in the same manner. Only pin 7, the signal ground, is not changed.

If you desire to set your modem up to always have "Terminal Ready" high then you can build a connector and cable without the RTS line being connected. If you choose to let the computer assert the "Terminal Ready" line you can hang up the phone line and modem from the keyboard when you are using "CMODEM". This feature allows you to drop the connection to one system and then call another one without having to manually drop the modem line. This may not be a problem on some Hayes compatible modems but other modems do not always allow for software hangup. This is a problem if your modem is not located next to you and you need to hang up the modem remotely.

CMODEM will default to 1200 baud upon execution. If you desire some other baud rate, such as 2400 or 9600, you may change the baud rate manually using the "M" option of the main menu and option 1 of the Maintenance menu. An easier method is to build a "profile.txt" file which has an entry "baud 2400". CMODEM will look on your current work disk for a "profile.txt" and set up all of the parameters as specified in the file. See the CMODEM documentation for a complete list of options that can be set using the "profile.txt".

Under SK*DOS, CMODEM has two types of file transfer - text and binary. When your file is ascii text you will use the "S" and "R" functions to transfer the file. If you are trying to move an executable file to your SK*DOS system then you will need to use the "Y" option. To upload a binary executable file you would use the "X" option. If you are transferring a file from an IBM bulletin board you should also change the internal option to add a CR before the NL when the file is sent. This option allows the insertion of both carriage return and new line at the end of each line for data that is being sent to the IBM, and the removal of the new line from data being received from the IBM.

This is required because the end of line character for SK*DOS is a carriage return while the end of line characters for CPM and PC-DOS is a carriage return and a new line. This will be option 5 under the "M" menu and option two (2) under the modify line option (5). If this is your normal operating method you may wish to enter a line "IMC" (Internal Mode CPM) into your "profile.txt" file to set it up each time you execute CMODEM.

From the Users' Group
by Sidney Thompson

Here are some notes on Users' Group Programs:

NRO

Having been involved with UNIX and nroff (its text formatter) for several years, it has become habit to place all my material in nroff format. The C user group has a subset of nroff called nro.

While this program was written for BDS C, I have modified it to run under both SK*DOS and UNIX. I currently use the nro program on my system for SK*DOS, Fortune 32/16, and a VAX running ULTRIX. I have also added most of the common commands used by the TSC formatter so that with the exception of some of the header macros, which do not have an exact nro equivalent, the document can be processed through nro.

One consideration is that the output must be sent directly to the printer if you are using the bold or underline features. Some printers have a problem using the backspace overstrike feature so nro also offers an option of issuing a carriage return and reprinting the affected portion of the line. Under SK*DOS, if the output is redirected to a file, this will generate two separate lines since the carriage return is the end of line character. This feature is due to the end of line character being a line feed in UNIX rather than a carriage return.

CCHK

There are many times while writing a C program that errors can creep into your code. This can be even worse if you receive code from someone who has tab sets that place the end of the lines off the edge of an 80 column screen. An easy error is to have a comment that does not end properly or to put "=" when you meant "= =." CCHK is a program that was a cumulative adaptation by several people on the national "newsnet". I find it useful. After I have made several changes to a C program I run CCHK to ensure that all the braces match, and other syntax appears correct.

The use of this program does not insure that your program will work correctly, but does mean that the compiler will not get upset at you because of missing brackets or that you will not get strange results due to nested comments. A sample program "echo.c" is shown. Admittedly this is an exaggeration (but not by much) of a normal program. Believe it or not this program is correct as far as syntax is concerned. But readability does leave something to be desired.

The CCHK program will compile under SK*DOS with the C compiler.

CB

There are times when you receive a program source from someone else or when you are adding code to an existing program that the structure of the program is less than optimum. The CB (C Beautifier) program will go a long way toward easing this little chore. It was a part of the C user group collection and has been reworked to run on SK*DOS and UNIX. It may seem a waste to have it run under UNIX, except that it is often easier to develop a C program on a UNIX system with its increased throughput and then move it to the micro. Having a consistent set of tools available on all of the systems I use has proven to be very useful in this area. Now if I could just find one screen editor that would work for all systems, it would be even simpler.

The second sample is the echo program after it has been run through CB. At this point it does appear to be a bit more readable than previously. Unfortunately CB does nothing for you

as far as comments or understanding the flow of the program. If these items are missing you are still on your own. It is also unable to break down the code of those programmers trying to impress someone else with how smart they are in the use of all of C's side effects. But then C style is another subject.

CALLS

The calls program can be used to analyze the procedure calls of a C source file. It lists out the function calls made within the program with the appropriate indention to show the program control flow. It also will indicate any function calls that can't be resolved from the source as external functions that would be provided by the runtime library.

I have run echo.c through CALLS and included the output. It does show the basic calling sequence to give a better understanding of how the program gets from beginning to end. It still does not tell you what the program is doing.

XREF

XREF allows you to obtain a cross reference listing of all variables within a program. This program can be used with C, ASM or text files to produce a cross reference listing. It will indicate all lines where a label is referenced within the source. This can be very handy when trying to analyze a program. If you simply type xref without any file name it will give the following usage menu:

```
xref cross-references a c, basic, x assembler, or text file.


use: xref [-option [-option ...]] filename-list
 -a  for non-intel assembler program.
 -ai  for intel format.
 -b  for basic program.
 -bn  for xref by statement, not line number.
 -c  for c program.
 -f  for inclusion of file names in xref.
 -kfilename for kill-list file.
 -l  for listing of input files with line numbers.
 -pxxx  for overriding page depth (xxx = lines).
 -t  for text file (default).
 -u  for forcing upper to lower case.
 -wxxx  for overriding page width (xxx = columns).
 -x  for insertion of line after each xref item.
```

An output based on echo.c is shown. This is a listing of all the variables in the program and where they are referenced.

Conclusion

These are some of the tools that are in use on a daily basis in my development of C programs. They are all in the public domain and the original author's names have been left in for proper credit. They are currently being used on several machines.

Example 1: Unformatted (Original) Program

```
#include <stdio.h>
#include <ctype.h>

char *expand();
int nl;
char *p;
char *q;
int c, j, i,sp;
main(argc, argv)
int argc;
char
*argv[];
{for (i = 1; i < argc; i++)fputs(expand(argv[i]), stdout);
if (!nl)putc('\n', stdout);fflush(stdout);exit(0);}

char
*expand(a)
char *a;
{char string[256];
if (*a == '-'){nl += ((*(a + 1) | 0x20) == 'n');return "";}
p = string;if (sp++)*p++ = ' ';
while (*a)if ((*p++ = *a++) == '\\'){switch (c = *a++)
                {case 0:--a;break;
case '0':for (c = j = 0; isdigit(*a) && (j < 4); ++j, ++a)
c = (c << 3) + (*a - '0');*(p - 1) = c;break;
case 'B':case 'b':*(p - 1) = '\b';break;
case 'C':case 'c':--p;++nl;break;
case 'F':case 'f':*(p - 1) = '\f';break;
case 'N':case 'n':*(p - 1) = '\n';break;
case 'R':case 'r':*(p - 1) = '\r';break;
case 'T':case 't':*(p - 1) = '\t';break;
case 'V':case 'v':*(p - 1) = '\v';case '\\':break;
default:*p++ = c;}}*p = 0;return string;}
```

Example 2: After CB

```
#include <stdio.h>
#include <ctype.h>

char *expand();
int nl;
char *p;
char *q;


int c, j, i,sp;
main(argc, argv)
int argc;
char
*argv[];
{
   for (i = 1; i < argc; i++)
      fputs(expand(argv[i]), stdout);
   if (!nl)
      putc('\n', stdout);
   fflush(stdout);
   exit(0);
}
char
*expand(a)
char *a;
{
   char string[256];
   if (*a == '-'){
      nl += ((*(a + 1) | 0x20) == 'n');
      return "";
   }
   p = string;
   if (sp++)
      *p++ = ' ';
   while (*a)if ((*p++ = *a++) == '\\'){
      switch (c = *a++){
      case 0:
         --a;
         break;
      case '0':
         for (c = j = 0; isdigit(*a) && (j < 4); ++j, ++a)
            c = (c << 3) + (*a - '0');
         *(p - 1) = c;
         break;
      case 'B':
      case 'b':
```

```
            *(p - 1) = '\b';
            break;
        case 'C':
        case 'c':
            --p;
            ++nl;
            break;
        case 'F':
        case 'f':
            *(p - 1) = '\f';
            break;
        case 'N':
        case 'n':
            *(p - 1) = '\n';
            break;
        case 'R':
        case 'r':
            *(p - 1) = '\r';
            break;
        case 'T':
        case 't':
            *(p - 1) = '\t';
            break;
        case 'V':
        case 'v':
            *(p - 1) = '\v';
        case '\\':
            break;
        default:
            *p++ = c;
        }
    }
    *p = 0;
    return string;
}
```

Example 3: Output From CALLS

```
1   main
2     fputs [ext]
3     expand
4               isdigit [ext]
5     putc [ext]
6     fflush [ext]
7     exit [ext]
```

Example 4: Output From XREF

```
        page 1 Fri Aug 12 13:21:46 1988
a           22 23 26 28 34 35 37 40 43 44
argc        9 10 14
argv        9 12 15
break       41 46 50 55 59 63 67 71 76
c           8 37 43 44 45 78
case        39 42 47 48 51 52 56 57 60 61 64 65 68 69 72 73 75
char        4  6  7 11 21 23 25
ctype       2
default     77
exit        19
expand      4 15 22
fflush      18
for         14 43
fputs       15
h           1  2
i           8 14 15
if          16 26 32 35
include     1  2
int         5  8 10
isdigit     43
j           8 43
main        9
nl          5 16 28 54
p           6 31 33 35 45 49 53 58 62 66 70 74 78 81
putc        17
q           7
return      29 82
sp            8 32
stdio       1
stdout      15 17 18
string      25 31 82
switch      37
while       34
```

Notes on SMALL C
by Sidney Thompson

The Small C compiler from the Users Group is the small C as written by Ron Cain and modified by John Byrns. To use the compiler, the file SC.COM should be on the system disk. To execute simply type "sc". It will then ask several questions. One is whether you wish to have the c code included in your output. I usually answer 'no', but if you wish to see what type of code is being generated then answer 'yes'. The next question is whether you wish quick code. Answer 'yes' to this since it generates shorter object files. The next question is your output file name. Use some name like "edit.txt" if you are building the editor. If you have several modules then they should be included by a master module. An example of this is the ed-68k.c file that comes with the user group disk 1. After it has digested the input file it will again ask for an input. If you are going to include your own library etc., then enter that name; if not, enter a CR.

After you have run the compiler you must assemble all of the output. This will generate a file of type .COM which can be executed. Use a command line like "ASM EDIT -L". Hope this helps.

Call Waiting
by Paul Bervaldi

The following item from Family Computing (Sept. '87 issue) by Henry Beechhold might be of interest: "A reader wanted to know what could be done about loss of a modem carrier when a call-waiting signal hits the phone line. This is an easy one: Dial '1170' before you dial the answering modem. If this newly instituted feature is operational in your calling area (check with your telephone company), you should hear a double beep on the line immediately after entering the code. When you've finished your modem call, the line will automatically return to call-waiting status."

Programming the 68000
by Peter A. Stark

We were planning to start a series of lessons on programming, but in the meantime, the December 1988 issue of Radio-Electronics has come out with some programming info I wrote at the beginning of the year, so we'll refer you to that. We will start with programming in the next issue of this newsletter.

Your Input

Do you have a letter, article, program, or other material you'd like to contribute? Send it to us for the next issue. (One absolute requirement! It must be supplied as a ready-to-go disk file, in either SK*DOS or MS-DOS format. We will edit it to conform with our word processing requirements.)

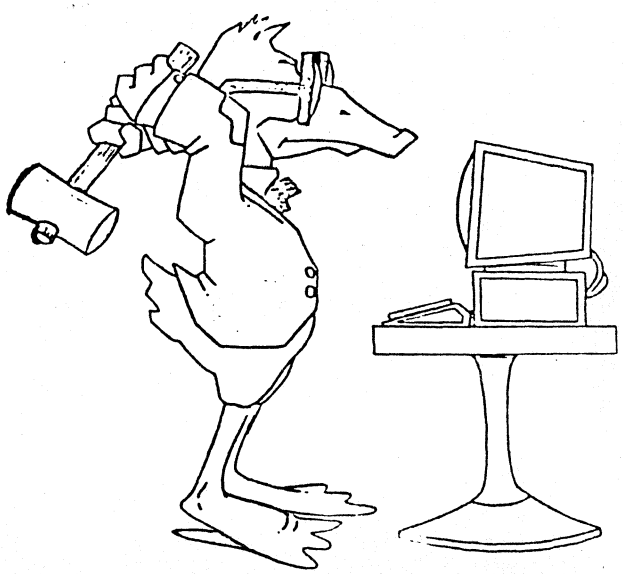
We are also starting a no-charge classified advertising section; feel free to use it to sell or buy computer-related stuff. As before, you must send us the ad as a ready-to-go disk file.

If you have a product related to SK*DOS or any of our other Star-K products, which you would like to sell, you may also place a display ad (see, for example, the Lagrange Instruments ad on page 3, or the Peripheral Technology ad on page 5). Please send camera-ready copy - nothing elaborate is needed - for our next issue. Our display advertising rate is $10 per quarter page, which is calculated to just about cover our expenses.

Look at your mailing label

If your mailing label has a dash above your name, you have not returned your User Registration Form to us and will not receive the next issue of the 68xxx News.

How To "HIT ANY KEY TO CONTINUE"



From: Star-K Software Systems Corp.
P. O. Box 209
Mt. Kisco, NY 10549

Bulk Rate
U.S. Postage
PAID
Permit #195

To: