The Official Publication of The

PENN-JERSEY COLOR COMPUTER CLUB

MAVERICK BBS

215-760-0456

FEBRUARY 1992

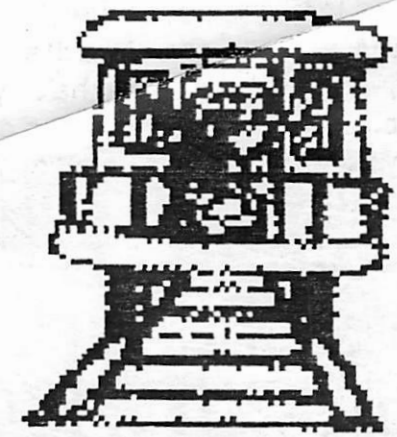
THE PRESIDENT'S REPORT
Rick Hengeveld

The countdown to the March PJCCC meeting continues. So what's so important about the March meeting? Well thats were we will pare down our club membership role to the active members. Currently we are carrying a number of people on our mailing list that are no longer active in the club. A lot of this is due to the fact that the PJCCC was able to forego club dues last year. At the same time some members moved away from the Coco, and are no longer active in the club. This year we've had to reinstate the dues ($15.00) and we will continue shipping the 6809 Express to only those who pay their dues by the March deadline. So if you can't make either the Feb. or March meetings, Let someone in the club know if you intend to remain in the club. Or better yet mail a check to the club. You can address it to the 6809 Editor, Pete Unks. Pete's return address is on the cover of your newsletter. Pete will see that your dues are proccessed.

****

SYSOP REPORT
Rick Hengeveld

The Maverick BBS has now gone over the 500 call mark! Over 100 downloaded files have been accessed from our database, All this from 40 registered users! For a Coco dedicated BBS, these are very good numbers. With almost 300 posted messages there has also been a great deal of information exchanged. For those few members that have a modem but have not yet accessed the Maverick, well ya don't know what your missing. Get online today!



THE LIBRARY CAR

By AL WAGNER

Welcome to the first session of the Library car for 1992. Hope you survived the holiday season in good shape. I'd like to talk a little on OS-9 I/O. OS-9 has a neat feature it inherited from its ancestor UNIX and that is standard input and output. What this means is that data coming into the computer and going out to a device looks the same to the computer regardless of the device from which it is coming or to which it is going. This simplifies writting many program functions. Let's say we wanted to move data from one place to another. The standard I/O feature means we need only one function that moves data from the standard input to the standard output to move data from the keyboard to the monitor or to move data from a disk file to the printer or from anywhere to anywhere. The default for the standard input is the keyboard. The default for the output is the monitor. Well, if these are the defaults, how do we use other devices? This is accomplished through the magic of redirection.

Let's say we want the date and time to show up on the printer. There is a date command in OS-9 that would display the date and time on the monitor. To redirect this to the printer the command would look like this: date t >/p. "Date t" is the command itself and as mentioned before this would put the information on the monitor. The ">" symbol redirects the output of a command to the device or file that follows it, in this case, the printer. Now let's imagine we've written a program where we enter data from the keyboard and it ends up on the monitor. We'll call this program KTOM. Now imagine we've created another program that generates the same data in a disk file called DATAFILE, that we had previously been entering from the keyboard. We'll call this program FILEGEN. KTOM also needs to be run several times a week. Typing all that data in accurately each time is difficult if not tedious. However we can redirect the input of KTOM from the keyboard to DATAFILE by the following command: KTOM <DATAFILE. The "<" symbol redirects the input of the program from the keyboard to the device or file that follows it.

Well that's just great, but now we want to feed KTOM directly from FILEGEN without having to use the disk in between. This calls for a pipeline from FILGEN to KTOM. In OS-9 a pipe is created with the

symbol "!". The command line would look like this: FILGEN ! KTOM. Redirection symbols "<>" are used between a program and a device. Pipes , "!", are used between programs. One caveat about pipes. In order to use them you must boot with Pipeman, Piper, and Pipe in the boot file.

Another use for pipes is in creating filters. A filter can be thought of as a program that modifies data. For instance a sort program. We've been looking at the output of our program KTOM, and have decided that it is getting difficult to find the entries we want for all the data. It would be easier to find them if the data were sorted. We have a sort program but don't want to take the time to manually run the output of the data from FILGEN through the sort but KTOM would work better if the data were sorted.

The easy way to handle this is use SORT as a filter between FILEGEN and KTOM. The command line would look like this: FILEGEN ! SORT ! KTOM. Now the output from FILEGEN is fed to SORT and in turn its output is fed to KTOM and we didn't have to reprogram anything! Isn't OS-9 NEAT!

Now look at another very practical place to use all of this. The command is DSAVE. Dsave allows you to copy large chunks of a disk, if not the whole thing, to another disk. The way it works is it starts at the current data directory and outputs commands to copy its contents and all directories below it to the specified location. The problem is it will output these commands to the monitor where they look impressive, but do nothing. You can handle this two ways, depending on what you need to do with these commands. Let's say you don't want to copy all the files but just most of them. We can redirect the output of dsave to a file that we can later execute as a procedure file. This way we can edit the file so only those files we want copied appear in the file. The command line could look like this: dsave /d0 /d1 >copfile. On the other hand if we wanted to let dsave copy everything it could, we could pipe the output into a shell like this: dsave /d0 /d1 ! shell. This cause each command to be executed by the new shell as dsave generates it.

Each command is displayed on the screen as it is executed, allowing you to follow its progress. But we are busy today! We don't want to wait for dsave to finish doing its task before we start another job. Another OS-9 symbol is the "&". Adding it on to the end of a command, such as either of the dsave command lines shown above, send it into a background mode and allows the screen to be free for other jobs. A typical command line might look like this: dsave /d0 /d1 ! shell&. If you recall, we used this last month to start our wordprocessing window without tying up the window from which we started our procedure file.

Another advantage to OS9 is the languages that are available. If you have a COCO3 and get OS9 level II, BASIC09 comes with it. Pascal, C, and Fourth are available. Since

BASIC09 comes with the level II package, let's talk about it for a while.

If you've done any programing in RSDOS, chances are you've programed in BASIC. The effort required to make step up to BASIC09 is quite small. You say you're not a BASIC programmer? Well BASIC09 fundamentals are no more difficult to learn than RSDOS BASIC fundamentals. The KEY operative here is "FUN"damentals. Besides once you get past the elementry beginnings of programming, you'll quickly appreciate BASIC09's quickness and power. The quickness over RSDOS comes from BASIC09 being a partialy precompiled language rather than a run time compile language as most other implementations of BASIC are. The power comes from its expanded command base. Variable structures are available that just don't exist in RSDOS. Direct access to OS9 commands and system routines is also there for the calling.

Let me explain about compiling. The microprocessor chip in any computer does not understand english or any other language, including BASIC! In order for the CPU to be able to process our instructions a translator, known as a compiler, must change our human readable instructions into machine instructions consisting of binary 1's and 0's. With most BASIC's, and RSDOS is one of them, this compiling takes place as the program is run. Computers can be very ignorant. Imagine yourself in a situation where you are in a foreign country and had to carry an English-Whatever/Whatever-English dictionery with you wherever you went. Imagine also that you couldn't remember a single word of this language even for a split second. Every time someone spoke to you, you had to look up every word. If that word occurred twice in the same sentence you would have to look it up twice! Now after a while you got very good at looking up words. So good in fact that most people didn't even realize you were doing it, but not as good as someone who did not have to look up the words. This is how a run time compiler works! Where the look up time really shows itself is in a loop where a program fragment must be executed many times in rapid succession. Each time through the compiler must re-compile each command in each line. By the way, the computer's act of looking up the commands is called parsing.

In contrast, a fully compiled language will go through the compile routines once at compile time and then when run will not have to look up anything as the translation to machine language has already taken place. This makes for very fast execution. BASIC09 is somewhere in between. Since a single BASIC command usually translates into several if not many machine language commands, a compiler actually looks up a routine rather than a single command. In a fully compiled language these routines are actually written into the compiled program. In BASIC09 however the compiling stops with an index to the machine language

routine being substituted for the command. This means the cpu jumps immediately to the ML routine without having to parse the command first. Its like having an ML program that consists of many subroutines. Where are these subroutines stored? Well if you run the module known as BASIC09, they are in there. If you went a step further and ran a routine known as PACK on the program, you could run the BASIC09 program from an OS9 command line as long as RUNB is present. RUNB is BASIC09's other repository for all the subroutines. You do not actually execute RUNB. You can preload RUNB and that will speed up execution of PACKed programs considerably. To execute a PACKed BASIC09 program, make sure it resides in the current execution directory and that RUNB is either also in that execution directory or loaded into memory, then type the name of the program as it appears in the directory. You don't need or want to type run or enclose the name in quotes.

Next time we will try exercising a few simple programs. If you're interested, get a copy of The BASIC09 Tour Guide by Dale Puckett. It is a excellent tutorial on BASIC09.

Well, that's enough to keep you busy for a little while. Cuddle up to that nice warm COCO power supply and try a few OS-9 commands. Anything worth learning takes a little effort. So far we are just scratching the surface. How would you like to be able to write machine language routines without having to learn machine language or even assembly language? With the OS-9 "C" compiler you can. The programs are even portable to the MM/1 or other 68k machines. They are even protable to mess-dos machines and mainframes! How's that for versatile? Well, you want almost machine language speed, but you've learned BASIC and you really don't want to have to learn another whole new language. BASIC09 will fill the bill. The COCO, particularly the COCO3, was made for OS-9. Come on, warm yourself with the joys of conquering new ground. Don't be intimidated by something new. I'll be seeing you at the meetings to answer questions and you can leave messages on THE MAVERICK or even call me at home. Try it. You'll like!

Happy computing!

****

FEBRUARY 28
FUN WITH RASCAN PIX!!!
AND OTHERS TOO!
Dirty Trix with
CoCo Max III!
Peter Unks will do the DEMO!

6 8 1 0 9 EXPRESS

The Official Publication of The

PENN-JERSEY COLOR COMPUTER CLUB

H. Peter Unks, Editor

FIRST CLASS MAIL