geoSHELL

extras

geoSHELL Extras

Table Of Contents

Command descriptions and examples

This manual describes several "extra" commands that were created for geoSHELL 2.2 after geoSHELL was released. These were available for free downloading to geoSHELL users from various online services and are now being supplied with geoSHELL. Previously, these commands were supplied on a separate disk known as the "geoSHELL extras", but they can now be found on the geoSHELL disk (side 2 if 1541 format).

Documentation for these extra commands was always in the form of a separate geoWrite file for each command, but this manual now takes the place of those files for convenience.

ccard (external transient)
pcard

These two commands were created in December, 1993. Merry Christmas from Maurice Randall

These are two handy commands that you can use to create and print greeting cards. No, you won't create the 'entire' card, but with 'ccard', you can create a GeoPaint file with vertical and horizontal guidelines where you would have the folds of the card. Once you have used the command to create the file, you may then open GeoPaint and edit your card. When you have finished creating it, you then use the 'pcard' command to print the file. Even though there are two guidelines in the file, they will not be printed on paper, as long as you use the pcard command to do the printing for you. If you print it directly from GeoPaint, then the guidelines will show up on paper.
The idea is to have four sections on the GeoPaint page, and when folded, it will become a greeting card with a front and back cover and left and right inside pages. The guidelines help you to identify the location of the folds when editing your card. There is a guideline running vertically, halfway across the page and another running horizontally, halfway down the page. You do all of your editing in an upright manner. But when you use the 'pcard' command to print the file, the upper half of the page will be rotated upside-down so it will be properly oriented when folded.

The four editing areas are laid out as follows:
Upper right corner - Front cover

Upper left corner - Back cover
Lower right corner - Inside right
Lower left corner - Inside left

CREATING A CARD FILE

The proper way to create a card is to first use the ccard command to generate a GeoPaint file with the guidelines in it.

```
Example: ccard filename
```

This will create a GeoPaint file on the currently active drive. It will be blank except for the two guidelines.

Once you have done this, you can load the file into GeoPaint and add text, photo scraps, or draw your own graphics as you please. Just keep in mind that anything drawn where the guidelines are will not be printed. The guidelines are where the folds will be. You might want to allow a certain amount of distance from these guidelines as margins for each page of the card when it is folded.

Since some printers can only print at 60 dots per inch horizontally, this layout with four editing areas would not print properly since the right hand 25 percent of the page would not get printed. For that reason, you can add a parameter to tell geoSHELL that you will be using a 60 dpi printer.

```
Example: ccard 6 filename
```

By adding a 6 after the ccard command, the GeoPaint file that is created will have two vertical guidelines instead of one. The one that represents the center fold of the card will be moved slightly to the left. This will be the center when printed. The right hand guideline identifies the right hand edge of the card. Anything from this point over will not be printed.

Load up the two Christmas cards that were included with this package and see how they look on the screen. Then use 'pcard' to print the appropriate one depending on whether your printer is capable of a resolution greater than 60 dpi or not.

PRINTING A GREETING CARD

When you are ready to print your greeting card, use the 'pcard' command to do so. This way, the guidelines will not be printed. The card command uses the GEOS printer drivers when printing.

```
Example:  pcard filename
```

This will proceed to print the desired greeting card. Your standard GEOS printer driver will be used for this purpose.

You can use the STOP key to halt the printing in case the card does not look like you want it to. For some of the multi-pass printer drivers, you may have to hold the key down for several seconds until it responds.

When the card is printed, the upper half of the GeoPaint file is actually flipped upside down as it is printed. This way, the card will be properly oriented when folded. The lower half of the GeoPaint file is printed just as it appears on the GeoPaint screen. The lower half will be the inside of your card, while the upper half will be the front and back covers. When folded, you will have a card that is 5.5 inches tall by 4.25 inches wide.

MAKING MULTIPLE COPIES

By using geoSHELL's 'exec' command, you can effectively create multiple copies of a greeting card. Simply create a GeoWrite file with the following commands in it:

```
pcard filename^  {don't forget the terminator}
exec multicard^
```

Now, assuming that you give this exec file the name of 'multicard', it will just keep repeating itself until you press the STOP key. Of course, a much faster way would probably be to just create one greeting card and take it to the local copy shop to make the additional copies.

convert (external transient)

This command allows you to convert GEOS files to a sequential PRG file format for ease of transferring across the phone lines via existing terminal programs. The command will also convert files that have been converted this way back into the GEOS format. It recognizes the popular conversion format that is widely used known as Convert 2.5..

Just enter the command followed by the desired filename to convert.

Example: convert filename

This makes it quick and easy to convert files one at a time right from geoSHELL.

execp (external transient)

This command is called 'execp', which stands for 'execute with parameters'. What this does is it allows you to pass parameters to your exec files. Previously, an exec file (also known as a script or batch file) could only function as it was written. Now, by using the execp command to invoke your exec files, you can pass from one to six parameters to them, which can make the same exec file perform actions in different ways.

The exec command itself is a resident command in geoSHELL. It does not have to be loaded from a disk when used. Normally, when you use the exec command, it will load in your exec file and execute the commands contained within it. If you use the execp command, execp will take care of loading in the file, but it will also plug in any parameters that you have supplied on the command line. Once this is done, it will invoke the exec command to finish the job.

You can use the command exactly like you would the exec command and it will perform exactly the same. However, you can also supply up to six optional parameters. These parameters will be copied into the exec file in place of specific variables that you define when creating the exec file. Here's a simple example that uses one parameter:

```
execp copyram^ b
```
In this example, 'copyram' is the name of the exec file and 'b' is the parameter that is supplied to it.

The exec file might look something like this:

```
fcopy parm1 getshell^
fcopy parm1 geoSHELL^
fcopy parm1 GEOPAINT^
fcopy parm1 MyFile^
```

What would happen here is that since one parameter was supplied, execp will load in the exec file called copyram and replace all occurences of the string 'parm1' with the letter 'b'. When the exec command executes the file, it will call fcopy to copy four different files to the ramdisk which is drive B in this example.

Let's look at another example that requires two parameters:

```
execp paint^ a^ 8
```

By now, you may have noticed that the filename for the exec file must be terminated by the usual up-arrow. Then each parameter is also terminated by an up-arrow. The last parameter on the line need not have an up-arrow since there are no more commands following it. One thing to keep in mind though is that you cannot have any multiple commands on the line with the execp command, since they would look like parameters to it. You must also place exactly one space between each up-arrow and the next parameter. If more than one space is used, the additional spaces will be considered part of the parameter, so this is perfectly legal.

Anyway, let's take a look at an exec file that could use this example:

```
parm1:
@cpparm2^
run GEOPAINT^
```

The first parameter, which is an 'a' would be put in place of 'parm1'. Since parm1 is followed by a colon, this would select drive A as the active drive. Then an '8' would get put in place of 'parm2' and the command to select partition 8 on a CMD device would be formed.

SOME LIMITS

You can use as many as six different parameters named parm1 through parm6, and you can repeat any one of them as often as you wish within an exec file. Of course, depending on the size of your parameters, you may not be able to fit six parameters on one command line, especially in 40 column mode. If you find you need more room, you can define a function key from within an exec file and then use the function key. You can define the maximum allowed 80 characters for a function key if it is done from an exec file. The maximum size that each individual parameter may be is 16 characters. If you find that you need more than this for one parameter, then you would have to combine two parameters together.

If the user supplies more parameters on the command line than exist in the exec file, the extra parameters will simply be ignored. Likewise, if not enough parameters are supplied, then no substitutions will be made for the missing ones. The drawback to this is that when the exec command comes to a location containing the text 'parm2', or 'parm3', or whatever, it will naturally not be a valid command and execution of the file will cease.

USING DEFAULTS

There might be some exec files that you will create and use that can be used exactly the same way most of the time. In that case, the execp command is not needed. You would just do things the way you have always done them. However, there might be those times when you need to make a change here or there, in which case it is handy to be able to send a parameter. The command execp allows you to define some default parameters right within the exec file. You simply put the execp command in the exec file and follow it with the default parameters enclosed within curly braces. A previous example might be written as follows:

```
execp {a^ 8^ GEOPAINT^ }
parm1:
@cpparm2^
run parm3^
```

If this file is named 'paint', you would just enter:

```
execp paint
```

Since you did not supply any parameters on the command line, the file will be loaded and executed as-is. Within the file, however, the execp command will be invoked and will recognize that you wish to use the default parameters. It will proceed to copy the defaults into the desired locations within the file and then exec will continue to execute the remainder of the file. If you had supplied any or all of the required parameters on the command line, then execp would only have used the defaults that you did not supply parameters for.

To use execp from within the exec file like this, you must be sure to place the left curly brace after execp with exactly one space in between. Then put the first default beginning just after the curly brace without any space, unless the space is needed as part of the parameter. Put an up-arrow at the end of each parameter with a space separating each one. You must end the default definitions with a right curly brace. And this must be placed after the last defined default with exactly one space separating it from the last up-arrow. In other words, it would be where the next parameter would be placed if there was one. Up to six default parameters may be used here also. With execp contained in the file, the exec command itself could be used to execute the file since the execp command would set up the file correctly with the defaults anyway. The above example would then load and run GEOPAINT from partition #8 of drive A.

You can place execp with the default definitions anywhere within the exec file as long as they appear before the parameters that are to be changed. Normally, you would put it first in the file, since it is likely that the command is already loaded in memory if it was used to invoke the file.

CALLING ANY PAGE

Execp has another handy feature in that it allows you to execute any page within an exec file. The resident exec command will only read page one. Just put the desired page number between the command name and the filename as in this example:

```
execp 15 menufile^ key1^ key2
```

This would load in page 15 of an exec file called 'menufile' and supply two parameters called 'key1' and key2'. Just like with the exec command, you can use the execp command to call up an exec file from within an exec file. You could conceivably have a 61 page exec file running some sort of presentation demo continuously. You can even put an exec file in a header or footer of your file. In this case, use page 62 or 63. If no page number is supplied, then execp defaults to page one.

Here's one more example of a way to switch between 1581 partitions and native partitions on a CMD hard drive. To invoke the file, a command line would look like one of the following two forms:

```
execp hd^ a^ NT
```
- or -
```
execp hd^ c^ 81
```

The first example would make the hard drive as drive A begin working with native partitions, while the second example would make it use 1581 partitions if it is located as drive C. Here's what the exec file would look like:

```
remove parm1
installHDparm2.128 parm1
```

If you imagine the parameters copied in place of parm1 and parm2, you can figure out quite easily how the commands will work. Remember that the command such as installHD81.128 is created by the 'learn' command.

The execp command should open up many possibilities for anything from a simple exec file to a very elaborate one. Just use your imagination and put this command to work for you.

ifont (external transient)

This command, ifont, stands for 'install font', and will install a new font into your copy of geoSHELL. The font that is to be installed must naturally be a GEOS font, but there are some other rules that should also be followed. geoSHELL requires a non-proportional font in order to work properly. This means that every character must be the same width. Also, the width must be 7 pixels and a point size of 7 must be used. The baseline must be set at 5 pixels from the top of the character. Here's an example:

```
ifont CourierGS
```

This will take the 7-point font contained in the CourierGS font file and install it into the geoSHELL file that is on the currently active drive. The font need not be on the same drive.

To see the new font in action now, just enter 'run geoSHELL' and the geoSHELL file on the current drive will be loaded along with the new font that you just installed. If you like the looks of it, you can then copy this version of geoSHELL to your other workdisks, or just run the ifont command again on the other disks that contain geoSHELL.

By using the supplied font, CourierGS, as your starting point, you can use any available font editor and alter this font to suit your taste and then use ifont to install it. CourierGS is a nice looking font for 40 column mode. With a 128 in 80 column mode, it may not be as clear at you would like on some monitors. It will look good on a sharp monochrome monitor or a good-quality color monitor. If it does not suit you, just make a copy of the font and edit it with a font editor to fatten up portions of the characters for legibility.

Another good use for this command is to be able to use a font with different looking characters for use in foreign countries. Just edit the CourierGS font to suit your needs.

laser (external transient)

This command is intended for sending a file to a printer, 'as is', without any translation. As it's name implies, it's original purpose was for sending PostScript documents to a PostScript compatible laser printer. However, you will find that it will work with any dot-matrix printer also. Just be sure that the intended file contains information that your printer understands. If not, the only thing you will hurt is the fact that you might waste some paper.

geoSHELL's '@p' and '@g' commands will send a file to the printer in combination with the 'type' command also, except that the type command performs a certain amount of translation before sending the data to the printer. The reason for the translation is so that only characters that are recognized by GEOS are displayed to the screen. Some characters or byte values will cause GEOS to do strange things or even crash. For this reason, the laser command will not echo it's output to the screen while it is sending the data to the printer.

'laser' requires a destination parameter and a filename parameter. The destination can be either the serial port or the user port via a geoCable. For serial port use, the default printer device number as set with the 'pconf' command will be used. No other pconf settings are used. Remember the contents of the file is sent, as is.

```
Example: laser p filename
```
This will send the file to a serial printer or a parallel printer controlled by an interface connected to the serial port.

```
Example: laser g filename
```
This will send the file to a parallel printer connected to the user port with a geoCable.

The laser command checks the keyboard after every 256 bytes that it sends to the printer. This allows you to hold the CONTOL key to pause the printing or to hit the STOP key to halt the printing altogether. By adding one more parameter to the command, you can cause the command to not check the keyboard. By doing so, the data is sent to the printer a little faster, but with the disadvantage of not being able to stop the printer. So, only use this parameter if you know for sure that the data is correct.

```
Example: laser gn filename
```
By adding an 'n' after the destination parameter, geoSHELL will ignore the keyboard until all of the data has been sent to the printer.

The laser command is intended mainly for sending non-GEOS text files to the printer, however some GEOS files may work OK, as long as it is a sequentially structured file. This command also works on files that have been 'printed' to a disk file.

mem (external command)

This geoSHELL command can do what BASIC's PEEK and POKE commands do. You can use it to peek a value that is in a specific memory location or you can also use it to poke a value to a memory location. Let's first see how to peek a memory location:

```
mem 1025
```
This performs the same function as PRINT PEEK 1025 would from BASIC. The mem command will display the value that is currently located at 1025.

You can only view locations of memory that are currently visible to the system. In GEOS 64, this would be most of the ram areas, including all of the GEOS kernal and all of geoSHELL and it's variables. On the 128, you are only looking at about half of the ram in the machine. This is BANK 1, or in GEOS terminology, FrontRam as well as a good share of the GEOS kernal, but not all of it.

To use the mem command to poke a value into memory, use it like this:

```
mem 1025=9
```

As you can see, the syntax is quite simple. In fact, that particular memory location is where geoSHELL keeps track of the currently active drive that the user is seeing. By poking a 9 to this location we can simulate the same thing as if we entered a 'b' followed by a colon. Try putting anything from an 8 to an 11 here. The advantage to using a drive letter followed by a colon instead of just poking the value at 1025 is that geoSHELL will check to make sure that the device is really there.

However, if you use mem to put a 4 into location 1025, you can now fool geoSHELL into sending DOS wedge commands to your printer or interface with the '@' key from geoSHELL. Some interfaces will accept commands via a command channel this way.

So far, we have used decimal numbers with the mem command. You can also enter numbers using the hexidecimal format as in the next two examples:

```
mem $401
mem $401=$9
```

These two examples accomplish the same thing that the previous examples did.

With a little knowledge of the GEOS operating system and some geoSHELL locations, you might be able to do some special stuff with this command. Memory location $0491 is where geoSHELL's pconf command stores the secondary address that is sent whenever geoSHELL sends output to a serial printer. You can use the mem command to also change this. Or if you put zeros at $0750 and $88c5, geoSHELL will then think that you have just booted your system. Just type 'run geoSHELL' and when it restarts itself, any commands following 'onboot' in your startup file will be executed.
For most purposes this command won't be needed, but for some very specific reasons, it might also be one of the most useful. Just be sure that you know where you are poking to.

ppaint (external transient)

This command will print a GeoPaint document without having to load up GeoPaint. You can now print your GeoPaint files right from geoSHELL.

```
Example:   ppaint filename
```
This will print a GeoPaint file from the currently active drive. That's all there is to it.

The standard GEOS printer drivers are used to do the printing. You can use the STOP key to halt the printing in case the document does not look like you want it to. For some of the multi-pass printer drivers, you may have to hold the key down for several seconds until it responds.

MAKING MULTIPLE COPIES

By using geoSHELL's 'exec' command, you can print a number of GeoPaint files all at once and walk away from the computer while the printing is taking place. Just create a GeoWrite file with the necessary commands in it and execute the file with the 'exec' command.

pwrite (external transient)

Once you've used this command a few times, you will love GeoWrite. This gives you the printing capabilites that GeoWrite left out. From the very beginning, GeoWrite was always intended to use your printer's graphic mode. This allows almost any style of font to be used. GeoWrite excels at this type of printing. Some of us still like using our printer's own built-in fonts, however. Yes, GeoWrite has a mode that let's you print in Draft or NLQ. That will use your printer's own font. But here's the catch, all style changes are ignored! This is where pwrite comes in. All bold, italic, underline, subscript and superscript style changes are supported. Only the outline style is ignored since graphic printing must be used in order to support it. So, do not use outline in your document or it will throw the layout off.

This command also supports PAGE, DATE, and TIME in the headers and footers. The title page setting is also supported. Double-spacing is handled as well. But, 1 1/2 spacing is not, since GeoWrite does not format the page properly for this to work on all printers. And not all printers can print at 4 lines per inch. But to do 3 lines per inch only requires an extra linefeed.

Now you can have your cake and eat it too. Use GeoWrite for all your graphic-type printing and use pwrite for all of your NLQ printing. When you create your GeoWrite document, you must use the COMMODORE 10 pt font throughout for proper formatting, since pwrite expects 80 characters per line and 66 lines per page. There will actually be 62 printed lines with 4 lines of nothing in order to skip over the perforation. This is how GeoWrite formats the document, so pwrite does it the same way. In fact, whatever GeoWrite does on the screen, pwrite will do on the printer, as long as the COMMODORE font is used.

Pwrite does not check the font that is in use though. It will ignore any font change. But if you don't use the COMMODORE font, your results may not be what you expect. By using this font, GeoWrite will allocate 12 pixels for each line, giving you 6 lines per inch and so each page on the screen will look just like what comes out on the paper. GeoWrite always uses the number of pixels in the point size plus two extras for each line in the document.

THE CONFIGURE FILE

Here is another plus for pwrite. Before printing begins, it will look for a file called pwconfig. If it finds it, it will use the printer codes it finds in the file. Since pwrite does not use your normal GEOS printer driver, it must know how your printer works. This is what the pwconfig file is for. This contains the codes for turning on the style changes and turning them off. If pwconfig is not found, then all style changes will be ignored. You can also leave out any style change codes that you wish to have ignored when printing your document. Delete the ones that your printer does not support, or change them to any setting that you would like activated.

There is also a series of codes that can be sent at the start of a document and another at the end. Any codes following 'open' will be sent at the beginning of your document. This is useful for setting up your printer into any special configuration you want, such as the desired font selection. Then, when printing is finished, the codes following 'close' will be sent. You can use this to reset your printer back to it's default settings, or whatever you desire. The default codes that are supplied will send the normal reset sequence of 27 64. This is what most Epson mode printers use. You are free to change this any way you want.

All of the other settings should be self-explanatory. For instance 'boldon' will be used whenever the style change for bold is encountered. And likewise, 'boldoff' when non-bold text is once again encountered. The settings that are supplied are used on most Epson mode printers. Each of these settings will accept up to 20 codes. Each string of codes must be terminated with an UP-ARROW character. There must be one space separating each code. The basic rules for any command in geoSHELL is followed here. Just think of these as commands that only pwrite recognizes.

If you are using a serial printer or serial interface, pwrite will use the parameters that are set by the pconf command. These are the device number, secondary address, whether or not you need linefeeds added and the character set being used. Be sure to use pconf to make these settings. This also applies to some of the other printing functions in geoSHELL. For geoCable users, the device number and secondary address are not important, since only serial devices use these.

PRINTING YOUR DOCUMENTS

When you are ready to print, you must decide which pages you desire to print, and how your printer is connected. Here is one example:

```
pwrite ap filename
```

If you are familiar with geoSHELL, it's obvious what the filename does. But the parameter before the filename, 'ap', tells pwrite to print 'all' pages and send them to a printer or interface connected to the serial port.

You can see that the 'p' is used just like with any other print command in geoSHELL. Likewise if you are using a geoCable, substitute a 'g' for the 'p'. Instead of using the 'a' to print all the pages, you can use an 'o' or an 'e'. These would select either 'odd' or 'even' pages. This makes it handy for printing double-sided sheets. The only thing you must remember is how your pages are numbered. The first page can start with an even number if you told GeoWrite to do so. Keep this in mind.

For those times when you need to print just one or a few pages out of your document, you can tell pwrite to do so with a slightly different parameter arrangement. You still use the ap, ag, ep, or whatever, but you can also include a range of pages just before this. Here is an example:

```
pwrite 12 24 ap filename
```

This will print all pages from 12 through 24 of your document. If you used an 'o' instead of the 'a', then only the odd pages between those two would get printed. The pages that get printed are determined by how you have them numbered in the document. If your first page starts out as page number 36, then the previous example would print nothing. Let's say you wanted to print just page 10. The following example would do this:

```
pwrite 10 10 ap filename
```

You always select the starting page and the ending page when you use this parameter. Just remember that this is optional and you can leave it out if you wish to print the entire document.

There might be an occasion when you need to print your pages in the other direction. If you were to put a higher number first and a lower number second in your parameter, then the higher numbered page would get printed first, followed by the next page before it and so on until the second page number specified is printed. Here's an example:

```
pwrite 35 30 ap filename
```
This will print the pages from 35 through 30 in reverse order.

HINTS AND TIPS

As with anything, the best way to learn anything is to just start doing it. But, here's some things you might want to remember when you are creating your document. Unless you are experimenting with something a little different, be sure that the COMMODORE font is selected throughout your document. It is very easy for the BSW font to be mistaken here or there when viewing it on the screen. Watch your headers and footers. Make sure the correct font is chosen here also. It is more critical here. If the wrong font is used in the headers and/or footers, some of the data might not print out because of the space allowed. If your headers or footers are not coming out right, double check the font there. The BSW font does not require as much vertical space, and so all of the lines contained within might not print out.

Speaking of headers and footers, these can be very useful. GeoWrite does not provide any means of controlling the top and bottom margins unless you use page breaks. But by using a header or footer with some carriage returns, GeoWrite will allocate a blank line for each carriage return. This let's you control the amount of blank space as each page skips over the perforation.

Headers and footers work just a little different here. When using a header, the content of your page will begin at the very next line following your header. But when you use a footer, GeoWrite inserts one blank line before printing the footer. So, this will cause printing to go to the 63rd line instead of stopping at the 62nd line. So, if you want printing to begin two lines down, just hit two carriage returns in a blank header. Or if you want a bigger margin at the bottom of the page, add carriage returns to a footer. Just count one extra space for the blank one that is inserted by GeoWrite. Pwrite will follow the same format that GeoWrite does here.

Don't worry if this sounds confusing, pwrite will format your page properly. Whatever you see on the screen will be what comes out on paper. But use that COMMODORE font or it might not. Even if you put just carriage returns in a header or footer, make sure that those carriage returns have the COMMODORE font assigned to them for proper line spacing.

Actually, if you don't like the looks of the COMMODORE font on screen, you can substitute any other font as long as it's dimensions in height and width are exactly the same.

When you set up or modify the pwconfig file, you can put anything you want on any page except page one. Only page one is used by pwrite. This way, you can store different setups on other pages and just cut and paste them to page one when needed. You can also keep notes on the additional pages for your own use. If you use a printer with limited capabilites, such as a 1525, check out page 3 of the supplied pwconfig file. If you are using a CMD device and have assigned a partition with the path command, pwrite will also search the path partition for the pwconfig file. When it searches for your document, it will search all drives except the path partition. You don't have to have the document on the currently active drive. This allows large works that are spread across multiple files to be more easily printed without having to select the correct drive first. Just make sure that pwrite doesn't find the wrong one if you have more than one copy of the same name visible on your system.

Since pwrite doesn't care what codes you use for each of the style changes, you might be able to do some special stuff. Let's say for example that you don't really need underlining in your document, but you want to use two different fonts. Just change the underlineon and underlineoff to select a different font. Then whenever you want to use this font, underline your text in GeoWrite. You can have up to 20 codes in each string. This should be more than enough to do some pretty fancy stuff. Use your imagination.

In fact, if you are careful in planning, you could have your printer do double-wide and double-high printing. If you do double-wide, limit your line length with carriage returns. If you do double-high or anything else that would print larger than 6 lines per inch, use a page break to limit the length of the page. You might find that you will have to print one page at a time, in some cases when doing special things. Double-wide is handy for headlines.

You can also print columns easily now by using both pcode (or gcode) and pwrite. Start by setting the left margin in your document all the way to the left and the right margin just slightly to the left of the middle on each page. Then begin by printing just the odd pages. When finished, use pcode to send the code to your printer that will move it's left margin over to the center of the page. Put your paper back into the printer and use pwrite to print the even pages. The second column will now print out.

Or, if your printer supports returning to the top of form, you can set up an exec file to do it all in one step. The following example works on a STAR NX-1000 and any other printer that supports the CHRS(27)CHR$(12) sequence:

```
pcode 27 64^ {tell the printer this is the top}
pcode 27 67 70^ {make the page length longer}
pcode 27 108 0^ {left margin to zero}
pwrite 1 1 ap filename^ {first column}
pcode 27 12^ {return to top}
pcode 27 108 40^ {left margin to center}
pwrite 2 2 ap filename^ {second column}
{now do the next page}
pcode 27 64^ {tell the printer this is the top}
pcode 27 67 70^ {make the page length longer}
pcode 27 108 0^ {left margin to zero}
pwrite 3 3 ap filename^ {first column}
pcode 27 12^ {return to top}
pcode 27 108 40^ {left margin to center}
pwrite 4 4 ap filename^ {second column}
```

...and so on....

If you have included any photo scraps in your document, these will be ignored by pwrite. However, the correct amount of blank space will be inserted for the graphic. This allows you to put the paper back in the printer and do one of two things. Using a copy of your document, you can delete all of the text lines on the page and enter carriage returns in their place. And then use GeoWrite to print the graphic on your page. Or you can convert the GeoWrite document to a GeoPaint and then erase all the text and let GeoPaint or the 'ppaint' command print the graphic. With a little thought you will actually have more control here. You can place the graphic wherever you want on the page.

If you have a printer that is wide enough to take a page sideways, you could print left and right pages and then fold them into a 5.5 x 8.5 inch booklet. Just use the codes in the 'open' string to control the size of your font and the lines per inch. A condensed elite setting works pretty good here. Also use margin settings in GeoWrite to help with this.

There is one other hitch whenever you are printing with your printer's own fonts. That occurs when you use 'bold'. Remember that GeoWrite is designed to print in your printer's graphic mode. That was the original idea. NLQ printing was an afterthought. GeoWrite creates bold characters by doubling the width of each horizontal pixel in the characters. This will add one pixel to the overall width of the character. This will in turn produce less than 80 characters per line. Your printer, however, can print bold characters and still get 80 of them on each line. So, now if you use bold characters in your document, what you see on the screen may not be exactly what comes out on paper. Let's say that you change an entire paragraph to bold. GeoWrite might use 6 lines to display it on the screen, but it might only take 5 lines to print it. This is the only problem you will run into. The entire content on the page will still be printed correctly with a blank line being added at the bottom. So, if this doesn't bother you, then OK. But if it does, you could substitute a font that is 7 pixels wide instead of 8 whenever you use bold type. Then what you see on the screen will be exactly what comes out on paper.

If you use bold in a single line that does not cover the full width, then this will not be a problem. By trying it out you will learn to adjust for it. Even though the computer can do a lot of work for us, it still takes a little work on the part of the user. But that's the fun of it. As you use this command, you will discover many tricks that you didn't think were possible.

relabel (external transient)

This is a very simple command. It's purpose is to rename a disk. It will only work on the currently active drive, so that the only parameter it needs is the new name you wish to give the disk. Here's an example:

```
relabel NewPics
```

This will place the name 'NewPics' in the disk's directory header and the new diskname will now show up whenever you list your directory.

If you have ever used 'dcopy' to do a whole disk copy, you will notice that the disk name of the destination is preserved during the copy. You can now use 'relabel' to also change the name of the destination disk to the same as the source disk if desired. Another useful idea for this command is for when you first boot up and configure your ramdisk from within your startup file. Assuming you have used the 'learn' command to create a command for installing your ramdisk, you will notice that the ramdisk is activated, however it is not formatted. The best method here is to have a floppy disk of a similar format containing the files you wish to have in your ramdisk. Use dcopy to copy the disk to your ramdisk. This is the fastest way to fill your ramdisk. At the same time, the ramdisk will effectively be formatted since the bam sector and directory from the source disk will be copied to it. But the diskname of the ramdisk will contain random characters. Just use relabel from within your startup file also to fix this.

Here's an example of how a portion of a startup file could be set up:
(This example assumes that drive B is a 1541 and drive C is a RAM1571 and that the 1541 contains a disk with the desired files)

```
installRM71.64 c
b: dcopy c
c: relabel RAM1571^
```

Of course you could substitute any desired name in place of 'RAM1571'.

sfont (external transient)

This command is useful for programmers that wish to include a font in a program. This command will read in a particular point size of most any font and then proceed to create a GeoWrite file containing the needed .byte statements that can be assembled with GEOASSEMBLER. This way, you can have the font contained right within your program, instead of having to load the font from disk. Here is an example:

```
sfont 12 Roma
```
This will create the source code for the 12-point Roma font. The source code is created on the currently active drive. The font does not need to be located on the same drive, since sfont will search all the drives for the font.

The resulting GeoWrite file that is created will be given a filename that represents the font and the desired point size. The above example would create a file called 'Roma12s'. If the font's name is longer than 9 characters, then the name would be truncated so that with the point size and an 's' added would keep the length limited to 12 characters. This is so that when you assemble the file, there is room for '.rel' to be added to the name.

Of course, you could also cut the byte statements out as text scraps and place them into any of your other files. But the easiest method is to just add this filename to your .lnk file.

One other thing that you will need to do before you assemble the font source code is to add the label you wish to have at the start of the code plus some additional imaginary labels. Since the file contains many byte statements, GEOASSEMBLER will give you errors if it doesn't find a label every so often. So, go through the code and about every 20 or thirty lines, add a label. That is the easy part, sfont has already taken care of the tough part for you, eliminating a great deal of typing while you viewing the font in a sector editor.

If you need to find out the point sizes that are contained in a font file, just enter sfont followed by the font name, leaving out any point size in the parameter. This will cause sfont to display all the available point sizes contained in the font. Here is an example:

```
sfont Roma
```
This will display the point sizes contained in the Roma font.

That's all there is to it. It's a simple, yet very useful command.

ucopy (external transient)

Here is the command that we have all been waiting for. This should prove to be the handiest filecopier around. It can actually completely replace the existing fcopy command. This comand started life as the 'update' command. But it grew and obtained many more features. The 'u' in ucopy stands for update, but you might get to thinking that it stands for 'ultimate' after you've used it a few times.

If you are a regular user of fcopy, then you will get to know ucopy real fast. First of all, anything that you can do with fcopy, you can do with ucopy. It is written such that it can be used exactly the same way as fcopy. For information on this method of it's use, refer to your geoSHELL manual. Here's a few examples to refresh your memory:

```
ucopy b filename
ucopy b newname=oldname
ucopy h15 filename
```

The first example copies a file to drive B. As always, the source drive is the currently active drive. The second example will make a duplicate copy of a file on the same drive or a different one while giving it a new name. The third example will copy a file to partition 15 of a CMD hard drive.

Now comes the new stuff. Since this command has updating capabilities, it can copy multiple files. You can give it a filetype parameter like you might do with the 'dir' command. You can give it a time or date parameter similar to the way you would when using the 'cdir' command. Any matches that are found will be copied to the destination drive. In addition to this, you can also supply a filename with wildcards. Every match that is found will be copied. This is one way that ucopy differs from fcopy. If a wildcard is used with fcopy, only the first match that is found is copied. If a filename is supplied with ucopy with no wildcards, then only one file will be copied.

USING THE DIFFERENT PARAMETERS

Let's start out with an example using the filetype parameter:

```
ucopy c 8
```

This will copy all font files to drive C. It's very logical and simple to use. Font files all have a filetype of 8. By using the 'dir' command, you can remind yourself of the different filetype numbers.

Let's see how to use a time or date parameter. If you've used the 'cdir' command, then you should have no problem with this one. When using cdir with a date parameter, you will be presented with a directory of all files that have a time/date stamp from that date on. The same applies here with the ucopy command.

```
ucopy b 011094
```

This will copy all files from January 10, 1994 and later. The time parameter works in a similar manner, except that it only works with files within the last 24 hours.

```
ucopy b 0500p
```
This would copy all files that were created since 5:00pm. Let's say that it is 2:00 in the afternoon. This would copy all files since 5 o'clock of the previous afternoon. If you wanted to copy all the files from today, then you would use a date parameter of today's date. This is the only area where ucopy differs from cdir. If you use cdir without a parameter, it will give you a directory of today's files. Without a time or date parameter with ucopy, then the time/date stamp is not checked, and all files that meet any other desired criteria will be copied. You also cannot use both a time and date parameter, only one or the other.

```
ucopy b *LQ
```
This example uses a filename with wildcards. It will copy all files that end with 'LQ'. That should be fairly easy to understand. You can use wildcards just like you would with most anything else in geoSHELL. You can use one or two asterisks to take the place of one or more characters. Or you can use a question mark to take the place of exactly one character. Up to 16 question marks can be used since the maximum length of a filename is 16. You can combine asterisks and question marks also.

```
ucopy c ???
```
This would copy all files that have three characters in their names.

COMBINING PARAMETERS

Let's see how to combine parameters. Here's an example for copying using a filetype and a filename:

```
ucopy c 8 *LQ
```
This has a filename with wildcards added. This will copy all font files that end with 'LQ' to drive C. If you were to leave out the '8' in this example, then any file of any type that ends with 'LQ' will be copied.

```
ucopy h10 7 012094
```
This would copy all application data files that have been created since January 20, 1994 to partition 10 on the hard drive.

```
ucopy a 7 1000p CHAPTER*
```
This example would copy all data files created since 10 o'clock at night that begin with CHAPTER in their filename.

```
ucopy b 1000p titlepic
```
This would copy the file called 'titlepic' to drive B only if it was created since 10 PM.

You can see that the parameters are easy to understand. But you are required to use them in a certain order. The command name is naturally always first. Next comes the destination. This can be a, b, c, or d or a partition destination just like fcopy uses. After the destination, everything else is optional. Take this next example for instance:

```
ucopy b
```

This would copy every single file that is in the directory. Relative files and subdirectories cannot be copied however. To copy the files contained within a subdirectory, you must open the subdirectory and then copy the files. Using ucopy with just a destination for a parameter is a good way to straighten out the files on a disk for faster access. When a lot of deleting, copying and editing takes place on a well used disk, the order of the sectors in each individual file can become quite disorganized. That is why you might hear the drive's head do a lot of moving about, perhaps more than on other disks.

When using multiple parameters, the filetype always comes first, if used, followed by a time or date, if used, and finally a filename, if used. So, if a filename is used, it is always last. And since a filename is optional and the fact that ucopy's parameter/s can be of an unknown length, if you use this command in an exec or startup file, or if you want to use another command on the same line from the keyboard, you must end it with a terminator, as in the following example:

```
ucopy b 9 ^ date
```

This would display the current date after ucopy copies all of the printer drivers to drive B. The terminator in this case takes the place of the filename. When no filename is used, as in this example, the terminator begins where the filename would begin, with a space just prior to it. If a filename is included, then the terminator would be immediately following the filename. Just remember that this is only needed when another command is used on the same line.

When you use ucopy, you will notice that it gives you the opportunity to back out before the copying begins. It will first tell you what you have chosen to do. This gives you the chance to check that you did not mistype the command and it's parameters. However, if you use ucopy either in an exec or startup file, it will proceed without interacting with you. It was done this way since it is expected that if used in any manner other than being typed in from the immediate mode or a function key, you would want it to proceed without stopping and asking questions. But it is easy to realize that you have made a mistake just as you hit the RETURN key. So, ucopy will let you double check yourself here.

You might also enjoy the added error messages contained in this command. It helps to diagnose when you have a problem with a disk.

ONE LAST FEATURE

As with fcopy, ucopy is a destructive filecopier. In other words, if a file of the same name is found on the destination drive, it is replaced by the file from the source drive. The only exception to this rule is when using the 'duplicate' mode.

Now you have a choice. By default, ucopy will still delete the file on the destination drive and then proceed with the copy, unless you add one more parameter. You can tell ucopy to skip over any files that are already on the destination drive, or you can have ucopy ask if you wish to replace the file with the one from the source drive. You do this by adding either an uppercase 'S' or an uppercase 'A' just before the destination parameter as in the following two examples:

```
ucopy Ab 7 C800p
ucopy Sr4
```

In the first example, whenever ucopy is about to copy a data file since 8 o'clock PM that already exists on drive B, you will be asked if you wish to replace it or skip it. In the second example, ucopy will copy all files to partition 4 of the RamLink unless a file of the same name already exists. Notice that no space if used after the A or S. You might also notice that the STOP key is active while the copying is taking place. Just hold down the STOP key to halt the copying process. If this is done while a file is being copied, no harm will be done, the destination disk will be in a state that is no different than if the file never got copied. The one thing to remember is that if a file existed on the destination drive of the same name, it may have already been deleted by the time you press the STOP key. So check your disk any time you do this.

REPLACING FCOPY IN GEOSHELL

Since ucopy is every bit as good as fcopy, you might want to consider replacing fcopy with it. You already have the tools to do this. Start by copying ucopy to the disk that has geoSHELL on it. Then use the 'external' command to remove fcopy from geoSHELL as follows:

```
external fcopy
```

This will remove fcopy from geoSHELL and place it into it's own file on the disk. Then enter the following two lines:

```
del fcopy
ren fcopy=ucopy
```

You should be able to figure out that fcopy is completely deleted and ucopy has been renamed to fcopy. There is only one thing left to do:

```
internal fcopy
```

This now puts your new fcopy (ucopy) into geoSHELL. You may keep on using it just like you have always used fcopy. When you need the added features, just use the needed parameters. Just remember that it is now called fcopy instead of ucopy. It is not really necessary that you rename ucopy to fcopy. You can always just add ucopy to geoSHELL instead of doing it this way. But there is no need to keep the original fcopy around anymore. The fcopy that was supplied with geoSHELL V2.0 was a good copier. In V2.2 it got better. Now 'ucopy' is the one to have. Renaming it to fcopy is purely up to you.

I hope you enjoy this command. Sorry for putting so many different possible parameters in this one, but sometimes to give you added power, this becomes necessary. As far as speed goes, ucopy appears to be about 5 percent faster than fcopy. About the only drawback is it's extra size compared to fcopy. If you do a lot of work with multiple files every day, you should find ucopy to be a great pleasure to have handy.

- NOTES -

Maurice Randall
Click Here Software Co.
P.O. Box 606
Charlotte MI 48813

PH: 517-543-5202