SuperDuper

An Amiga Disk Copier
Version 3.13

by Sebastiano Vigna

1 Introduction

SuperDuper is a disk copier/formatter that tries to be to disk handling what Mostra is to IFF displaying: a fast, compact, system-friendly tool which combines speed, features, and some bells and whistles to make your life easier.

By *fast* I mean exactly what you're hoping—blazingly fast. A disk is usually copied and verified in less than 100s. Without verify, the time drops to 69s. You can buffer a disk in RAM in less than 36s, and then making a verified copy takes 67s, while a non-verified copy takes less than 36s. Adding another destination drive increases verified copy times by 34s, but hardly changes non-verified copy times (the Amiga can write more than one drive at a time; I just need a few tenths of a second in order to measure the drive speed and step the heads). Thus, if you really trust your drives and your media you can make four copies in 38s. These timings can vary with the system configuration, the multitasking overhead, the disposition of the blocks on the surface of the disk, the state of the Date option (which requires a separate write on the root block track for each disk), the DMA access of the custom chips and the display features enabled.

1.1 Changes

SuperDuper 3.x is in many ways a completely new program. This is why I *strongly suggest* to all previous users to *entirely reread* the documentation.

Now the program works only under release 2.04 and beyond of the operating system. Minor maintenance releases of SuperDuper 2.0x will probably be distributed for the users who do not still have Release 2, but you should assume that no programming effort from my part will be ever spent for writing something working under 1.3. Moreover, some features of 3.0 are used if present (for instance, scalable checkmarks and radio buttons gadgets).

The main new features are high density drive support, XPK compression support (for more details, see Section 4.2 [The Buffering System], page 10), a complete graphical display of the copy process status, and much more flexible buffering.

1.2 Main Features

- SuperDuper copies, formats and checks from/to any combination of Amiga drive(s).
- SuperDuper can buffer a disk in RAM, allowing for any number of duplications while reading the source disk only once. The combination of destination drives can be changed at each pass. If you have a hard disk, you can create on it an IFF image file that will act as a buffer. This file can be saved and reused many times. Also, all kinds of virtual disks are supported for buffering (VD0:, RAD:, FMS: ...).

- SuperDuper checksums the RAM buffer. If some badly written program is trashing your memory, you are alerted. Thus, buffered copies are as safe as direct copies.
- SuperDuper also checks its internal DMA buffers at each write.
- SuperDuper can use any XPK library in order to compresss the information it stores in RAM. You can choose your preferred compression algorithm, depending on the speed of your machine and on the available memory. With most compressors, write times are (almost) unaffected.
- SuperDuper will automatically retry tracks which produce a verify error. The number of retries is programmable. A complete graphical rendition of the status of retries and errors is given in the Info Window, and on request detailed error information printing is available.
- SuperDuper is highly system-friendly—the use of CPU time is negligible, so you can multitask efficiently.
- SuperDuper has the option of incrementing the creation date of the copy so AmigaDOS doesn't get confused. If, however, the option is switched on and the disk is not an AmigaDOS disk, SuperDuper won't increment the date.
- SuperDuper is faster than diskcopy—actually it pushes the drives to their limits. At the time of this writing, SuperDuper is the fastest Amiga copier both from a "pure" (physical time) and from a "per-copy" (real time for each copy when a big number of copies of the same disk is produced) point of view.
- SuperDuper alerts the user with sound (and optionally voice) about the operations in progress—so you can really be doing something else!
- SuperDuper can format both OFS, FFS and DCFS disks.
- SuperDuper displays a list of the last disks copied. If you do a lot of copying, you'll find this feature more than a little useful.
- SuperDuper can manage the Amiga drives without help from the trackdisk.device. Through the supplied utility SDBootInstall, you can create a boot disk which will keep the system away from your drives, giving you back more than 30K per unit. This is very useful when doing intensive buffered copying on a 1M machine.
- SuperDuper can automatically start any copy or format operation by monitoring the disks' extraction and insertion.
- SuperDuper's window can be opened on any public screen.
- The start/end cylinder of a copy is programmable.
- Unique numbered names can be automatically generated while formatting.
- SuperDuper has a time indicator.
- SuperDuper has a beautiful name. 8^)
- If this is not enough, an ARexx interface allows any kind of customization. In particular, a startup ARexx script lets you set up a custom configuration. Since SuperDuper can turn off its graphical user interface via a command line switch, it is possible to use SuperDuper as a CLI command by writing a suitable ARexx macro. A switch allows you to shut down ARexx in order to gain memory. ARexx macros can be launched via the ASL file requester.

1.3 First Steps

To use `SuperDuper`, you simply double-click on its icon. You will see a window appearing on the screen: it's the main window, which contains the main controls of the program.

Just under the window title you can see the *progress bar*, which gives you an approximate indication of how much of the copy process has been carried out. Under the progress bar, you can see the *action gadgets* (see Section 3.1 [The Action Gadgets], page 5), which let you control `SuperDuper`'s activities. On the right side, there are the source/destination and copy mode selectors.

To make your first copy, if you have two (or more) drives simply select in the first column of gadgets of the `Src/Dest` box the gadget for the drive which contains the source floppy, and in the second column the gadget(s) for the drive(s) containing the destination(s) (for the time being do not choose the same drive both as source and as destination). Then hit the `Copy` gadget. After a while, the display will flash, a beep will be generated, and the copy will be finished. As each cylinder is copied, the progress bar is updated.

If you have only one drive, select it both as source and as destination. Then select the `Buffer` radio button in the `Mode` box. Now put in the source disk and hit the `Read` gadget: the buffer will be filled with the contents of the disk. If the progress bar reaches its maximum length, then the whole disk has been buffered. Pull out the source disk, put in the destination, and hit the `Write` gadget. The buffer will be written to the disk. If only a part of the source disk was buffered, put it in again, buffer it again (note that now the progress bar starts where it stopped before) and write it again. This process must be repeated until the whole disk has been copied. It is safer to set the write protect tab on the source disk, in order to avoid the unpleasing side-effects of source/destination mismatches.

`SuperDuper` supports both double and high density floppies. Of course, you have to use the same kind of floppy in all drives, or a requester complaining about a type mismatch will be issued. You can change the density of the floppy each time you do a copy, but you have to be careful to not confuse the operating system: See Section 6.1 [SuperDuper and Your System], page 21.

2 Windows

`SuperDuper` 3.13 features several windows. This was made necessary by the growth of information to display, which is now divided among a main window and two additional windows. The left/right arrow keys allow to bring to front cyclically all the opened windows, while `F1` toggles the zoom state of the currently activated window.

2.1 The Main Window

When `SuperDuper` is started, it usually opens a window (unless you specify the '`NoGUI`' tooltype; see Section 4.3 [The CLI and WB Options], page 13). This window contains the progress bar, and the main control gadgets. See Section 3.1 [The Action Gadgets], page 5.

The progress bar gives you a rough estimate of the part of the copy process that has been carried out. Also, it tells you when an error occur: in this case, the bar is updated in a lighter (dithered) color. For more precise informations on the error, see Section 2.2 [The Info Window], page 4.

The zoom gadget can be used to reduce the main window to the point that only the main action gadgets and the progress bar are visible. This is most useful if you want to free some space on your screen, yet to be able to govern the copy process.

The main window is opened at startup unless you specify the '`NoGUI`' option, either *via* the CLI or *via* tool types. In this case, no window is opened. Note that it is not possible to close the main window *via* ARexx and have the info or options window opened.

2.2 The Info Window

The Info window contains only informations—there is nothing you can set. All the information about the copy process is gathered here. In order to open it, you have to click on the `Info` gadget of the Main window.

On the left, the `Status` gadget tells you what `SuperDuper` is currently doing. Its normal display is '`Idle`', and it changes to reflect the operation currently in progress.

The `Elapsed` gadget tells you how much time has passed from the start of the current operation. `SuperDuper` uses the system EClock, which is very precise and has a very low overhead.

The '`Copy #`' gadget tells you how many error-free copies of the current buffer have been done (for this to happen, you have to select a buffer, as explained in see Section 4.2 [The Buffering System], page 10).

Just below there is a standard listview: it contains a list of the drives copied, checked or buffered, the topmost being the latest.

On the right, you can see the four disk status displays. For each cylinder read or written, a square is displayed in the respective drive status display. If a retry occurs, a triangle contained in the square is painted in a lighter, dithered color. If the retry are unsuccessful, the triangle is set in the background (lightest) color. Of the two triangles contained in each square, the leftmost represents the upper side of the floppy, while the rightmost represents the lower side.

The info window can be opened and closed at your will. The `ESC` key or the close gadget will make it disappear, but you can open it again *via* the respective gadget, or *via* the '`Window`' ARexx command. See Chapter 5 [ARexx], page 16.

2.3 The Options Window

The options window contains a series of gadgets which allow to control the kind of operations performed by `SuperDuper`, for instance, if buffering should use compression. See Section 3.4 [The Option Gadgets], page 7.

The options window can be opened and closed at your will. The `ESC` key or the close gadget will make it disappear, but you can open it again *via* the respective gadget, or *via* the '`Window`' ARexx command. See Chapter 5 [ARexx], page 16.

3 Gadgets

`SuperDuper` is completely controlled *via* gadgets—there are no menus. Every gadget can be activated *via* the mouse or the keyboard (using the letter which appears underlined in the gadget name). The copy mode gadget in the main window cycles when the `M` key is pressed. The destination drive gadgets can be controled by pressing `SHIFT` together with the underlined number. You can use `Q` or `ESC` to exit, instead of hitting the close gadget. Note however that `ESC` will simply close the currently activated window, and possibly exit `SuperDuper` if the latter is the main window, while `Q` will always exit. Some of the string gadgets have underlined letters which activate them. Moreover, you can use `TAB` and `SHIFT-TAB` to pass from a string gadget to another one.

3.1 The Action Gadgets

These gadgets control the actions that can be performed by `SuperDuper`.

`Stop` stops any operation. If pressed while the multi-pass real-time compression buffer is selected and no operation is in progress, it will empty the buffer and reset the pass count, thus allowing you to buffer another source even if the previous one wasn't finished (see Section 4.2 [The Buffering System], page 10). If you `Stop` immediately after starting a copy operation and nothing has been drawn in the progress bar, nothing has been written to the destinations.

`Copy` can be used only in '`Disk2Disk`' mode; it initiates a disk-to-disk copy operation. The source is copied to the destination(s).

`Read` can be used only when a buffer is selected; it fills the buffer by reading from the source drive.

`Write` can be used only when a buffer is selected; the content of the buffer is written on the destination(s).

`Check` is basically a `Read` without buffering. The source disk is scanned for errors. No buffer is needed to use it. Note that `SuperDuper` will detect trackdisk.device related errors, but it won't find DOS checksum errors (for this purpose, for instance, you can use `DiskSalv`).

`Format` formats the destinations(s).

`Options` opens the options window, or brings it to the front if it has been already opened.

`Info` opens the info window, or brings it to the front if it has been already opened.

`ARexx` opens a file requester, allowing you to choose an ARexx macro to execute.

`NoWB`

`WB` closes the Workbench, flushes the memory and opens a very small screen with only two colors. Moreover, the window is of `SIMPLE_REFRESH` type rather than `SMART_REFRESH`. This way, the maximum amount of memory for your system is at your disposal. If the Workbench can't be closed for some reason, a warning is issued (usually some application has a window opened on the Workbench screen). When you want to get back, hit the gadget again (this time it will be named `WB`). This feature is very powerful if coupled with `SDBootInstall` and with the CLI option '`NoARexx`'. See Section 6.3 [SDBootInstall], page 23, and see Section 4.3 [The CLI and WB Options], page 13.

Warning: If you grab the disk.resource (by selecting a source and/or a destination) just after a disk was inserted, it's likely the Workbench will be locked, waiting for you to unlock the drive in order to load the icon of the disk. If in this moment you hit `NoWB`, you will lock the entire system, since `SuperDuper` will be waiting for the Workbench to close, while the Workbench will be waiting for you to release the disk.

`SaveCon` saves the current configuration to an ARexx file named '`PROGDIR:Startup.supdup`' ('`PROGDIR:`' is the directory the executable program lives in). The file is a standard ARexx macro which can be further edited if necessary. Whenever `SuperDuper` will starts, it will try to execute this file. Note that in order to prevent the main window from being first opened and then moved to the preferred position, which is visually ugly, it is a good thing to use the '`NoGUI`' option. The configuration file contains commands that will open the desired windows even if '`NoGUI`' is specified. See Section 4.3 [The CLI and WB Options], page 13.

`Iconify` closes all windows and creates an icon on the Workbench, using the program icon. As soon as the icon is double-clicked, the windows are reopened again. Iconification does not interfere with copying or ARexx script execution.

3.2 The Disk Gadgets

These gadget allows you to specify the source and the destination(s) of any copy/check/format operation. You should take care of selecting a meaningful set of drives when starting an operation. For instance, you cannot copy if there is a drive which is selected both as source and as destination, but you can read, write, check and format.

3.3 The Copy Mode Gadget

The copy mode gadget defines the copying method that will be used by `SuperDuper`. There are four possibilities:

`Disk2Disk`
: `SuperDuper` will copy the sources onto the destination(s) whenever the `Copy` gadget is hit.

`Buffer`
: `SuperDuper` will read and write alternatively a disk. At each read, part or all of the source disk is copied into RAM, possibly compressing it. At each write, that part of the disk is written down to the destination(s). See Section 4.2 [The Buffering System], page 10.

`HD Buffer`
: works like '`Buffer`', but a file is used instead of the RAM. See Section 4.2 [The Buffering System], page 10.

`VD Buffer`
: works like '`Buffer`', but a physical device is used instead of the RAM. See Section 4.2 [The Buffering System], page 10.

3.4 The Option Gadgets

Several gadgets control various options.

`Verify`
: turns verify on and off (you can also format without verifying). However, turning off verify is not recommended.

`Date`
: toggles on or off the change of the date of an AmigaDOS disk. This change is necessary so AmigaDOS can distinguish otherwise identical disks; if two truly identical disks are inserted in the drives, AmigaDOS gets confused and crashes. However, if for some reason you want a "physical" copy, you would turn off this option. `Date` will be ignored for a non-AmigaDOS disk.

`Comp`
: enables the use of compression when in '`Buffer`' or '`HD Buffer`' mode. It works in conjunction with the `Xpk lib` gadget. See Section 4.2 [The Buffering System], page 10.

`PrintErrors`
: opens/closes `SuperDuper`'s detailed error report window.

`IncName`
: makes easy to format a bunch of disks with different, unique names. If this gadget is selected while formatting, `SuperDuper` will scan the `Label` string gadget searching for a numeric pattern (i.e., one or more digits) and will increment the pattern value for each disk formatted. In case more than one pattern is present, the last one is used. For instance, if you format four disks with label '`Foobar.000`', the disks will be named '`Foobar.001`', '`Foobar.002`', ... and at the end of the copy the label gadget will contain '`Foobar.004`', thus being ready for the next formatting. The more digits, the more unique names. Since you can start from any number, and after 99 ... 9 the numeration wraps around, if you need to start with 00 ... 0 you can put in something like '`Foobar.999`': The first disk will be labeled with '`Foobar.000`'

`FFS`
: enables the formatting of FFS disks; for copying it is ignored.

`Intl`
: enables the formatting of disks with the new international mode; for copying it is ignored. Note that the directory cache file system exists only in the international version.

`DirCache`
: enables the formatting of directory cache disks; for copying it is ignored. This gadget supersedes `FFS` in case both are selected.

`Talk`
: activates `SuperDuper`'s ability to give its status by voice. Currently only English is supported. Note you need '`translator.library`' and '`narrator.device`' for this to work, and they are not distributed any longer with the operating system.

`Auto`
: activates automatic operation starting. `SuperDuper` will monitor disk insertion and ejection. When all sources and/or destination(s) have been ejected and re-inserted, a suitable operation is started.

  If the copy mode is `Disk2Disk`, and source and destination(s) are both selected, a copy is started, but if only destination(s) are selected, a format is started instead.

  If the copy mode is a buffering mode, a source has been selected, and a write-protected disk is inserted in the source drive, a read operation is started. Otherwise, the buffer is written on the destination(s). This allows you to easily copy a series of disks without ever giving a command, just by write-protecting all of your source disks (which is anyway a good idea while using this option). Notice that if you are going to copy several times the same source disk, and you are afraid that some of the destinations could be write-protected, you can inhibit the automatic buffer reading mechanism by simply deselecting the source drive.

  Warning: especially on one-drive-only systems, `Auto` can be extremely dangerous. You'd better write-protect your source disks.

`Name`
`Unit`
: select the name and the unit number of the Exec device that `SuperDuper` will use as a virtual disk when in '`VD Buffer`' mode. Note that this is *not* the name of a DOS device, such as '`RAD:`'. You have to use the Exec name (in this case, '`ramdrive.device`').

`Label`
: lets you choose a name for the disks formatted by `SuperDuper`. The name can be automatically incremented using the `Incname` gadget.

`Filename`
: selects the name of the file `SuperDuper` will use as a buffer when in `HD Buffer` mode.

`Xpk lib`
: selects the name of the XPK library `SuperDuper` will use when compressing a buffer. The name is formed by four letters, denoting the library, and an optional dot followed by a number in the range 0–100, denoting the required efficiency (for instance, '`HUFF.50`' requires Huffman coding with a standard efficiency). Note that not all libraries actually look at the number after the dot.

`Retry`
: selects the number of read/verify retries on each track.

`Start Cylinder`
`End Cylinder`
: select the start and the end cylinders, respectively, for any operation.

4 Reference

This chapter goes into some details about the copy and the buffering process of `SuperDuper`. It is expected that most users will be able to use efficiently most of the features without reading this part, but it is a suggested reading if you want to fully exploit the available features.

4.1 The Copy Process

When `SuperDuper` starts an operation which involves reading a disk, i.e., `Read`, `Check` and `Copy`, it scrolls up the name list and marks the current drive as '`<UNKNOWN>`'. This happens because it can't know if the disk is a DOS disk before reading track 0. After less than a second, the track will be read, and the name will be changed to '`<NDOS>`' if the disk is not a DOS disk. Otherwise, as soon as the track 80 is read (the progress bar is in the middle) the name of the disk will be displayed. However, if for any reason the name is incorrect (wrong format, read error, etc.) `SuperDuper` will name the disk '`<BAD NAME>`'. In this case, it is very likely that the root block is a little bit scrambled, so it's probably a good idea to turn off the `Date` option gadget. Beware: if you are using a multi-pass buffer, the name of the disk could be unavailable at the first pass.

If `SuperDuper` finds an error on read (or verify), it will retry reading (writing and verifying) the track, warning you by painting a half of the square representing the current cylinder in the info window in a lighter, dithered color. See Section 2.2 [The Info Window], page 4. If after a number of retries specified in the gadget `Retry` the error remains, `SuperDuper` will set the half square to the background color and continue. A little lighter, dithered rectangle in the progress bar will point out approximately where the error occured. It will be positioned horizontally proportionally to the track number.

Note that while retrying `SuperDuper` can't be stopped: don't set the `Retry` gadget to 99 unless you really know that's what you want to do. If you want to get a very detailed error report, you can activate the `PrintErrors` switch. A console window will appear (or the original console will be used if `SuperDuper` was started from the CLI), and every wrong read, write or retry will generate a message explaining what doesn't work. Usually you will get bad checksums, but if a track is really scrambled `SuperDuper` could be unable to get the first sector after a gap, in which case nothing at all is recovered.

The squares in the info window are drawn in a different color if you're doing a read, a format or a copy operation—so you can be sure you read the new chunk in the buffer, and so you can avoid formatting your floppies when you think you're copying something to them. The `Status` gadget will be set to the operation currently executed. Note also that the progress bar and the elapsed time indicator are not updated if something locks the screen (like using menus). The update is delayed until the screen is unlocked (thus `SuperDuper` won't get stuck as will almost all programs which do any rendering to their windows).

If you specify start/end cylinders different from 0/79 in the `Start Cylinder` and `End Cylinder` gadgets, only the part of the disk specified will be copied. The main use of this option is for retrying some lazy disk (usually on the last tracks) if you're not satisfied with the number of retries issued by `SuperDuper`. Please refer to the section on the buffering system for some subtle interactions between the RAM/HD/VDisk buffer and the start/end cylinder selectors.

While doing buffered copies, at each successful copy (that is, without errors) the `Copy #` indicator will be incremented. Thus you can know precisely how many disks you copied. Moreover, the counter will be incremented only if the operation ended on the last track of the disk and started from the first track of the buffer. This allows you to manually retry spare tracks by changing the `Start Cylinder/End Cylinder` gadgets without getting spurious increments, and if a multi-pass copy is in progress only the last pass will actually increment the counter.

4.2 The Buffering System

The buffering system of `SuperDuper` 3.x has been completely rewritten compared to previous versions. Now it is much more orthogonal and powerful. Moreover, it relies on the XPK compression standard, which allows to use a plethora of different compression methods, just by choosing the suitable library.

Buffering is useful when you have to do a lot of copies: you read a disk only once, and then you can make as many copies as you want without rereading it. It also has other uses: if you have to create distribution disks (for instance for a commercial package) you can create them using high speed virtual floppies, such as Commodore's `RAD:` or Matt Dillon/Jim Cooper's `FMS:` disk. `SuperDuper` can then read from those virtual disks and make many copies on floppies at high speed.

Since data integrity is a primary issue, `SuperDuper` checksums the buffers. The possibility of writing a munged track is very low. Strict control is also kept on the validity of the buffer—you can't write random data on your disks inadvertantly.

In order to do a buffered copy, you have to choose the suitable copy mode using the copy mode gadget in the main window. While `Disk2Disk` tells `SuperDuper` to do a disk-to-disk copy, the other three options offer three different buffering techniques.

`Buffer`
: `SuperDuper` will use the RAM for buffering a disk. The memory is allocated while the disk is read and, in case it is not enough for buffering a whole disk, many passes can be necessary. Beware of the fact that many programs tend to crash under low-memory conditions, so if you have 1MB or less you should close everything you can before trying to do a RAM-buffered copy. You should possibly use `NoWB` (see Section 3.1 [The Action Gadgets], page 5).

  If you foresee that a disk won't fit into the available RAM, you can activate compression using the '`Comp`' gadget (see Section 3.4 [The Option Gadgets], page 7). `SuperDuper` uses the XPK standard, which means that you must have the XPK system completely installed in order to use compression. There is a wide choice of compression algorithms available, and you can try out until you find the one with the best compression/speed ratio for your purposes. The compression algorithm can be selected by typing its name into the `Xpk lib` gadget of the options window. The name is specified by four upper case letters, optionally followed by a dot and a number between 0 and 100, included, which specifies the required degree of efficiency. See Section 3.4 [The Option Gadgets], page 7.

  Beware of the fact that while doing compression `SuperDuper` always fully uses the CPU. Even moving the mouse can slow down the operation in progress. Anyway, if you have all of your memory allocated for the buffer, it is definitely not a good idea to do anything besides waiting for the copy to finish.

  A little side-effect of the allocation of all of the available RAM is that some requester could be turned into an alert, or could even disappear without waiting for the user to acknowledge it.

`HD Buffer`
: `SuperDuper` will create a file that will be used as a buffer, exactly like `Buffer` does with RAM. The file name can be changed using the `Filename` gadget (see Section 3.4 [The Option Gadgets], page 7). Of course you should use it only if you have a hard disk. The file is an IFF file, documented in Section 4.6 [The Buffer File], page 16. The `Read` operation will be a little slower, but if you have a good hard disk you should be able to make copies as fast as with a RAM buffer.

  Note that you can give the name of an already existing file. In this case, the file will be considered a ready-to-use buffer file, and you will be able to `Write` immediately. This allows to use `SuperDuper` as a disk compression system.

`VD Buffer`
: This is probably `SuperDuper`'s most esoteric feature. By typing a device name in the string gadget named `Name`, you can select any device (`SuperDuper` needs the Exec device name, e.g., '`ramdrive.device`' for the RAD: AmigaDOS device). The unit number is taken from the gadget with the label `Unit`. The device you specified will be used as a buffer for your disks. `SuperDuper` expects the device to behave like the trackdisk.device,

namely it must be able to write data at specific offsets. The main devices you can use, with their respective names, are:

RAD: the recoverable RAM drive. Configure it in your mountlist as a floppy, and you can use it as a buffer (Exec name: '`ramdrive.device`').

FMS: Matt Dillon/Jim Cooper's virtual floppy-on-hard disk (Exec name: '`fmsdisk.device`').

VD0:, etc. other recoverable, sector-oriented RAM drives.

A RAM buffer is considered non-valid as soon as allocated, because it will contain random info. To make it valid, you must read in a floppy. File buffers and virtual device buffers are instead always assumed to be valid, because they could be externally fed. This mechanism allows you to prepare, for instance, a distribution disk at high speed in RAD: or in your hard disk using FMS:, and then to copy it to floppies directly.

In the same vein, `SuperDuper` will act slightly differently when determining if a buffer contains a DOS disk (if not, the incrementing of the date is inhibited even if selected). At read time, the information is recorded, but if at write time the pass starts from track 0, `SuperDuper` will re-fetch the DOS mark from the buffer and check it again. This way if for instance you externally feed a ramdrive.device with a diskcopy command `SuperDuper` will be aware of it and will increment the date if requested to do so.

Some care must be taken in order to obtain what you really want when mixing the buffering features and the selection of the start/end cylinder. `SuperDuper` implements a reasonable mean of flexibility and reliability for these kinds of operations.

When in `HD Buffer` or in `VD Buffer` mode, the read/write operations start and end exactly where you specify with the start/end cylinder gadgets. Since `SuperDuper` has no control over what you do to the virtual disk while it's not accessing it, it has to assume you made it right. Note that this also means that it is not a good idea to change the start/end cylinder *after* you buffered a disk. A disk buffer keeps no information about the position of the tracks it contains. Thus, if you change the start cylinder `SuperDuper` will start to write the disk buffer at that cylinder, even if the first cylinder of the disk buffer was recorded elsewhere (as a side effect, this allows you to move easily tracks from one part of a disk to another).

When using a RAM buffer, `SuperDuper` can clearly make some assumptions on its validity. In particular, just after allocation or a stopped `Read` it assumes the buffer is not valid. Moreover, it knows exactly were each cylinder was taken from, so that you can rewrite parts of a disk just by changing the start/end cylinders (this is also true of the virtual disk buffer, but only if it is used in one pass).

If you have a valid RAM buffer and you change the start/end cylinders, there are two cases: either the buffer range and the start/end range do not intersect, in which case an error message is issued if you try to write the buffer, or there is a non-empty intersection, in which case the intersection will be written, i.e., the starting track will be the greatest of the start of the buffer and the start cylinder, while the ending track will be the least of the end of the buffer and the end cylinder. Example: if you read something with `Start Cylinder`=20, `End Cylinder`=30, then you set `Start Cylinder`=10, `End Cylinder`=25 and hit `Go`, the range 20−25 will be written.

If all this scares you, don't fear: the buffer/range interaction will simply work just as you intuitively expect. I hope at least `8^)`.

4.3 The CLI and WB Options

`SuperDuper` accepts some arguments, both from the CLI and from the Workbench tool types, in order to select a number of special features.

When you start `SuperDuper` from the CLI, you have the chance to specify an option. The possible options are printed in the standard Amiga template format if you type '`SD ?`'. In this case, the following line

```
PubScreen/K,NoGUI/S,NoARexx/S,NoStartup/S
```

will be displayed. Its meaning is that `NoGUI`, `NoARexx` and `NoStartup` are switches that you can activate, while `PubScreen` must be followed by the name of an existing public screen. For instance, the command line '`SD NoGUI`' will invoke `SuperDuper` in its no-GUI mode. The two flags `NoGUI` and `NoARexx` are mutually exclusive—if both are specified, `SuperDuper` will exit.

`PubScreen`
: tells `SuperDuper` to open its windows on the specified public screen. If this argument is not specified, the windows are opened on the default public screen.

`NoGUI`
: tells `SuperDuper` to not open the main window on startup; you can then control it through the ARexx interface. This makes possible to write an ARexx macro allowing you to use `SuperDuper` from the shell much as the diskcopy command. Note that this option is also useful if you have a startup file (see Section 4.4 [The Startup File], page 14) and you want to avoid the visually unpleasant effect of the main window first appearing and then being moved.

`NoARexx`
: This switch shuts down the ARexx port. `SuperDuper` won't open neither the ARexx port nor rexxsyslib.library. This mode is provided for user with 1M or less who want to have as much free memory as possible (moreover, see Section 6.3 [SDBootInstall], page 23).

`NoStartup`
: tells `SuperDuper` to skip the startup file (see Section 4.4 [The Startup File], page 14). This switch is very useful when controlling `SuperDuper` through an ARexx script, because it allows to start the program in a standard configuration, and without any window opened. Note that `NoARexx` implies `NoStartup`, for obvious reasons.

These options are also available from the Workbench tool types. Just put in `SuperDuper`'s icon the obvious tool types. For instance, '`PubScreen=TURBOTEXT`' will force `SuperDuper` to open the main window on `TurboText`'s screen, while '`NoGUI`' will force the no-GUI mode.

4.4 The Startup File

At startup time, `SuperDuper` checks if ARexx is available, and in this case it tries to start an ARexx macro named '`Startup.supdup`'. This file should contain your usual settings: it is a normal ARexx macro, just like any other one started by the `ARexx` gadget or by the `rx` command. However, a couple of conventions were implemented in order to get a better behaviour on systems without ARexx. In particular, the absence of the ARexx server or the ARexx error message '`Program not found`' will *not* be displayed if caused by the startup file. Notice that the last message can also be caused by the first line of '`Startup.supdup`' not being a comment (every ARexx macro must start with a comment).

The startup file can be automagically generated from the current setup by using the '`SaveCon`' gadget (see Section 3.1 [The Action Gadgets], page 5). Note that if you want to avoid the visually unpleasant effect of the main window first appearing and then being moved by the command in '`Startup.supdup`', you can start `SuperDuper` with the '`NoGUI`' option.

Several tricks are possible by directly editing the startup file: for instance, you can change the order in which the windows are opened, thus deciding which will be the frontmost; or you can start `SuperDuper` in iconified state; you can also set both the normal and the zoomed window position.

If you do not want to run the startup file, for instance because you would like to control `SuperDuper` through ARexx, you can use the `NoStartup` switch. See Section 4.3 [The CLI and WB Options], page 13.

4.5 Special Requesters

When `SuperDuper` needs to inform the user about something, usually a requester with a message appears (if the `Talk` option is on the message is also read out loud). While most of the requesters are self-explanatory, some of them need a more detailed description.

'`Can't get disk.resource`'

The disk.resource is the Exec way of controlling the access to the low-level disk hardware. `SuperDuper` can't access the resource, probably because someone is already using it. If you suspect a particular program, close it and try again to select a disk gadget.

'`Please free disk.resource`'

(See also previous requester). If the disk.resource can't be grabbed, Exec won't give back the message passed by `SuperDuper` until the resource is free. Thus, until that moment `SuperDuper` can't exit.

'`Checksum error: buffer munged.`'

Someone wrote over `SuperDuper`'s RAM buffer. The buffer is no longer valid, and the current copy is probably munged, too. You should probably reboot, because if something writes on someone else's memory it's likely it will do it again.

'`A track buffer has been munged.`'

Someone wrote on one of `SuperDuper`'s track buffers. The same comments of the previous requester apply.

'`ARexx server not active`'

In order to use ARexx macros, the ARexx server has to be activated. Type `RexxMast` at a CLI prompt (if it's not in your path, you should locate it easily).

'`Error while recalibrating unit x.`'

`SuperDuper` found an error while recalibrating a drive head. The head was moved to track 0, but the drive signal `DSKTRACK0` wasn't activated. This means that either your drive has lazy signals, in which case there's nothing to worry about, or that some head step wasn't actually performed (possibly because of power supply reasons) in which case the last copy could be bad, even if `Verify` is on. Better `Check` it. Try also to increase the step and calibrate delays of the drive with `SetTDDelay`. If nothing else works, the `RecalibrateCheck` ARexx command can selectively turn off this requester (see Section 5.2 [Selection Commands], page 18).

'`Better write-protect your sources.`'

This message is issued every time you select the `Auto` gadget on a machine with a single drive (see Section 3.4 [The Option Gadgets], page 7).

'`Can't mix floppy types.`'

If your Amiga is equipped with a high density drive, you should take care of never mixing two floppies of different kind (880K or 1760K), for otherwise `SuperDuper` will be unable to perform the copy.

'`Compression not enabled.`'

You are trying to write a buffer file which has been created using compression, but compression is not currently enabled.

4.6 The Buffer File

When in 'HDBuffer' mode, SuperDuper reads and writes an IFF file. Its format is documented here.

Informally speaking, the file is an SDDD or an SDHD FORM, depending on the density (double or high, respectively) of the disk stored. The allowed chunks are (beside the standard ANNO, AUTH, ... chunks, which are never written, but tolerated while reading) the BODY and XPKF chunks. They contain, respectively, an uncompressed or a compressed track. In the first case, the chunk is always 11K long, for a SDDD FORM, or 22K long, for a SDHD FORM. In the second case, the chunk (header included) can be passed "as it is" to the XPK unpacking functions in order to get the real data, as it is composed exactly by the output of the XPK packing functions (which happens to be an IFF FORM). The same restrictions of a BODY chunk apply to the unpacked data of an XPKF chunk.

The file contains no information about the position of the tracks. SuperDuper takes the first BODY or XPKF chunk of the file and starts to write it onto the first cylinder.

The regular grammar for the SDDD and SDHD FORMs follows:

```
SDDD ::= "FORM" #{ "SDDD" [ANNO] [AUTH] [FVER]
                          [NAME] [(c)] (BODY | XPKF)* }

SDHD ::= "FORM" #{ "SDHD" [ANNO] [AUTH] [FVER]
                          [NAME] [(c)] (BODY | XPKF)* }

BODY ::= "BODY" #{ UBYTE* }

XPKF ::= "XPKF" #{ UBYTE* }
```

5 ARexx

ARexx is the system macro language of the Amiga. It was originally developed by Bill Hawes (to whom every Amiga owner owes much more than he probably realizes) and was then included in the Release 2 of the operating system.

ARexx is a beautiful interpreted language, with unique features such as syntax/semantics collapsing (for instance, you can ask the value of a variable given its name as a string) and, overall, the ability to interface itself with external applications. A single ARexx script can control several different programs and make them interact.

The ARexx interface consists of a port, which is used for communications, and a set of commands that ARexx can issue to the application. For SuperDuper, the port name is 'SUPERDUPER', and the command set is described below. ARexx scripts written for SuperDuper should have extension 'supdup', like 'foobar.supdup'. This is in order to distinguish ARexx scripts written for different applications.

ARexx provides at little or no implementation cost a powerful macro language which substantially increases the performance and the versatility of an application. Maybe some feature you would like to have is not in SuperDuper at this time, but it's very likely you'll be able to put it in *via* a suitable ARexx script.

5.1 General Issues

Besides being able to execute commands issued by an ARexx macro, SuperDuper is also able to start an ARexx macro. This is indeed the purpose of the ARexx gadget (the last one in the last row). The gadget is activated only if the 'rexxsyslib.library' is somewhere in your LIBS: directory. You can start any number of macros at the same time (beware of wild interactions though).

SuperDuper commands generally correspond to gadgets, and are similarly named: for instance, the command Check will check the source drive, while VDUnit 4 will set the virtual disk buffer unit number to 4. Commands are case insensitive. A complete list, specifying each command and its template in AmigaDOS style, follows:

```
Stop                    ,
Copy                    ,
Read                    ,
Write                   ,
Check                   ,
Format                  ,
NoWB                    On/S,Off/S
Iconify                 On/S,Off/S
SaveConf                ,
Verify                  On/S,Off/S
Date                    On/S,Off/S
Comp                    On/S,Off/S
PrintErrors             On/S,Off/S
Incname                 On/S,Off/S
FFS                     On/S,Off/S
Intl                    On/S,Off/S
DirCache                On/S,Off/S
Talk                    On/S,Off/S
Auto                    On/S,Off/S
Label                   /A
Filename                /A
XpkLib                  /A
VDName                  /A
VDUnit                  /N/A
```

```
Retry                   /N/A
SCyl                    /N/A
ECyl                    /N/A
Mode                    Disk2Disk/S,Buffer/S,
                        HDBuffer/S,VDBuffer/S
Quit                    ,
Requesters              On/S,Off/S
RecalibrateCheck        On/S,Off/S
SmartRefresh            On/S,Off/S
Dest                    /M/N,On/S,Off/S
Source                  /N,Off/S
Help                    Command
NOP                     ,
RX                      Command/F
Window                  Names/M/A,Open/S,Close/S,Activate/S,
                        Min/S,Max/S,Front/S,Back/S,
                        LeftEdge/K/N,TopEdge/K/N
```

The same table is printed if you send to `SuperDuper` the `Help` command with no arguments. If you do not know anything about templates, you may want to look at the *Using the System Software* manual.

ARexx needs a console by which it communicates with the user. If you started `SuperDuper` from the CLI, the your original CLI will be used. Otherwise, a console window will be opened. It's always open, but it's an `Auto` console window, so you can close it if you wish: it will be reopened as soon as something is printed into it.

Most commands have absolutely trivial meaning, and will not be discussed in detail.

5.2 Selection Commands

`Source` selects the drive specified by the numeric argument as source; if '`Off`' is specified instead, turns off the source gadget.

`Dest` selects destinations using a list of drive numbers. If neither '`On`' nor '`Off`' are specified, exactly the drives in the list are selected (the remaining ones are deselected); if '`On`' is specified, the drives in the list are selected (and the other ones are left in their current state); if '`Off`' is specified, the drives in the list are deselected (and the other ones are left in their current state).

If you specify no drive in the list, the currently selected drives are assumed as default. Thus, '`Dest Off`' turns off all destinations.

`Requesters`

turns on or off the system requesters. Note that you will not receive any explicit error message for missing libraries, *et cetera*.

`RecalibrateCheck`
: turns on or off the recalibration error requester. Many users complained that this requester was appearing often, but the copies were successful. While I know that the only reason for this requester is a drive out of specs, I agreed to patch the situation by allowing to disable selectively the requester. See also Section 4.5 [Special Requesters], page 14.

`SmartRefresh`
: selects the refresh mode of `SuperDuper`'s windows. Usually, the windows are in `SMART_REFRESH`. This method uses more memory, but it is usually faster: you can get `SIMPLE_REFRESH` windows by putting a '`SmartRefresh Off`' line in your '`Startup.supdup`' file, and calling `SuperDuper` with the `NoGUI` option (every window opened after the command has been executed will be in the selected refresh mode). This line is automatically inserted when you generate the startup file using the `SaveCon` gadget, so that you can directly edit it. See Section 4.4 [The Startup File], page 14.

5.3 Miscellaneous Commands

`Help`
: returns in the `result` variable the template of the given command. If no command is specified, prints out a table with all commands and their templates.

`NOP`
: does nothing.

`RX`
: executes an ARexx script or a one-line ARexx command (if its argument is enclosed in quotes).

`Window`
: sets several parameters of `SuperDuper`'s windows. The parameters are applied to all the window listed (the possible window names are '`Main`', '`Info`' and '`Options`'). The syntax is self-explanatory. Note that closing the main window from ARexx will *not* quit `SuperDuper`. Note also that the command is effective even if the specified window is closed. This allows to set up all the position parameters of a window before opening it in the startup file. See Section 4.4 [The Startup File], page 14.

5.4 Return Codes

Commands issued by ARexx to an application should return useful values in order to tell what really happened. Generally, a command which fails returns an error level, while a successful command returns an error level of zero and, upon request of the caller *via* the `OPTIONS RESULTS` command, a result string which can be parsed in order to get useful information.

`SuperDuper` returns an error code of 10 if the syntax of the command was wrong. This will cause ARexx to complain with an error message. An error code of 1 is returned if the syntax was right but the command couldn't be executed, but there is no real failure (for instance, if you send `Go` while a copy is already in progress or if you try to select a ghosted gadget). An error of 30 is returned in extreme cases, for instance when you hit the close gadget and there are still some commands pending. No strings are ever returned, since we have only a few cases to differentiate. Return codes with special meanings are returned by the following commands:

`Source`

`Dest`

2	The selected drive is not connected.
5	The disk.resource is not available.

`Write`

`Read`

`Check`

2	This pass is not the last one.
3	Something is wrong with the chosen source, destination and buffer options. For instance, you're trying to copy from df0: to df0: without a buffer.
4	The buffer is not valid.
5	A unit is empty.
6	A unit is write-protected, or there is a floppy format mismatch.
7	The start/end cylinders chosen are meaningless. This can happen if the numbers are out of range, or (for a RAM-buffered `Write`) if there is no intersection with the current buffer.
8	There were errors.
9	There were errors. Moreover, this pass is not the last one.
20	Someone munged the RAM buffer or the track buffer.

`Talk`

5	The voice system cannot be activated.

`NoWB`

20	The current window has been closed, but it was impossible to open the new one. The program exits in this case.

5.5 What Can I Do with ARexx?

Basically you can expand `SuperDuper`'s capabilities and/or make it interact with other programs. A couple of examples of the first case could be a '`CheckAll.supdup`' macro which checks all drives in sequence. The "native" `SuperDuper` can only check one drive at a time, but if you have two or more drives you can check many drives using a macro like

```
/* CheckAll */
do i = 0 to 3
   source i
   if rc==0 then check
end
```

After checking you should of course look at the return codes in the `rc` variable and decide upon appropriate actions.

Suppose now you have four drives and you want to make a copy of two different floppies. You can put the sources in drives 0 and 2, the destinations in drives 1 and 3, and then

```
/* DoubleCopy */
mode disk2disk
source 0
dest 1
copy
source 2
dest 3
copy
```

This will produce the two copies in a completely unattended way.

6 Performance

`SuperDuper` has been written for performance. In the following sections we will review deeply the relations between `SuperDuper` and the operating system, and how they affect you.

6.1 `SuperDuper` and Your System

`SuperDuper` has been written keeping in mind that a good program doesn't have to eliminate everything from the system in order to work. The Amiga has a very efficient multitasking kernel which allows for resource arbitration.

When `SuperDuper` is started, it won't allocate anything from your system. As soon as a source/destination gadget is clicked, it will inhibit all of the drives (so don't select a gadget while reading or writing to floppies) and then will grab the disk.resource. Until the resource is released, *no one else* can access the Amiga drives. This is necessary in order to avoid unpredictable collisions with the system or other programs. Inhibiting the drives is not enough, since some other file system (like `CrossDOS`) could access them.

If you need to temporarily access your drives, you must simply deselect all `SuperDuper` source/destination gadgets: the disk system will be restarted (it will be re-grabbed on a gadget selection of course).

You have however to be a little bit careful if you change the density type of a drive while `SuperDuper` is active (for instance, if you first copy a double density and then a high density disk). In this case, when the drives are given back to the operating system, the device drivers will get completely confused, and they will still believe they are accessing a double density disk. If you change the density type, you should eject all the floppies before quitting `SuperDuper`.

The CPU use of `SuperDuper` is almost unnoticeable. You can do anything else, and you shouldn't notice any slowdown. In particular, if no source/destination is selected `SuperDuper` is completely asleep.

This however is not true if you use compression. In this case, not only will the system be slowed down (a priority 0 task will almost always be active), but *any* operation (including moving the mouse pointer) will slow down `SuperDuper`.

If you use the utility `ToggleClick` distributed with `SuperDuper` (or any other utility which legally kills drive clicks under Release 2) `SuperDuper` won't click empty drives (drive clicking is necessary for monitoring disk insertion; using `ToggleClick` is good but you must be sure your drives won't try to move past track 0 if asked to do so).

You should avoid running `SuperDuper` while a 16 color hi-res screen (or a 4-color ECS productivity mode screen) is displayed on an old or enhanced chip set. The video DMA access will interfere with the disk/CPU/Blitter access to the point that copy times will rise to incredible values—reading and compressing a disk in the buffer can take more than 100s.

6.2 `SuperDuper` and You

"Well," you could say, "`SuperDuper` is a great copier—but how can I trust it for making my copies? This guy diddles with hardware—maybe I should use the system `DiskCopy` command."

This is not a good idea. First of all, `SuperDuper` is *incredibly* picky about verifying. You will get more verify error messages than with the standard copy commands (for techies: `SuperDuper` verifies also the MFM timing bits, not only the data bits; this means a 200% efficiency improvement in catching verify errors and bad media in general).

Moreover, the 2.0 trackdisk.device has unpleasant side-effects on frequently read/written tracks. These side-effects are cleared when you do a copy of the disk with `SuperDuper` (for techies: trackdisk.device doesn't check for MFM bits being read in correctly, and doesn't re-MFM the track before writing it; it just re-MFMs the changed sector. If a MFM timing bit is read wrong, it will stay wrong forever, possibly causing read errors; but `SuperDuper` re-MFMs every track it copies, thus restoring every MFM timing bit to its correct value).

Finally, if you don't like coffee-breaks during your copies, you'd better use the fastest copier available—namely `SuperDuper`. Note that if you have four drives and you use top-quality disks, so you can skip verify, the buffer system allows you to get a per-copy time of 9 1/2 seconds, which is definitely not bad.

6.3 `SDBootInstall`

When your system boots up (at power on or after a reset), the operating system searches for available drives, and creates some trackdisk.device tasks accordingly. These tasks take a lot of memory for their buffers (>30K), but `SuperDuper` doesn't use them at all, because it has its internal routines.

If you have to do intensive copy work, and you have 1MB of memory or less, you could find it useful to boot up your system in a special configuration that will shut down almost all trackdisk.device tasks, thus freeing a lot of memory.

To accomplish this, do as follows:

1. Make a copy of your usual Workbench disk (from now on we work on the copy).
2. Delete some programs to make room—`preferences`, `diskcopy` and `format` are good candidates. Moreover, delete the file '`Disk.info`'.
3. Copy `SuperDuper` to the disk root directory (by dragging its icon on the disk icon or using the CLI).
4. Edit the startup-sequence of the disk (it's in the '`s`' directory). Delete it entirely, and substitute it with

   ```
   SetPatch >NIL:
   Run >NIL: <NIL: SD NoARexx
   EndCLI >NIL:
   ```

5. Now put the disk in df0:, and run the utility `SDBootInstall`. A special bootblock will be installed on the floppy. When booting from it, the operating system (and you) will be able to access only drive 0—the other ones will be for `SuperDuper`'s use only. To get back to normality, a reboot is necessary. You will gain 30/40K per drive using this method (for techies: it is perfectly legal—the bootblock simply AllocUnit()s the drives with ID>0).

6.4 A Word on Copy Protection

SuperDuper won't copy protected disks (or if it will it's just a coincidence). I do not believe in copy protection. Scrambled tracks will produce random data on the destination. If the read error goes beyond a simple checksum error don't expect anything meaningful to be written on the destination disks.

However, SuperDuper will faithfully reproduce data block checksum errors ('Disk foobar has a read/write error') or DOS checksum errors ('Key 880 checksum error') on the source disk in disk-to-disk copies (header checksum errors are fixed when renumbering the sectors). Thus, if you got the typical 'Key <n> checksum error' you can make a copy of the disk before fixing it. SuperDuper won't do any surgery: use a good tool (such as Dave Haynie's DiskSalv) for this purpose. On the other hand, during buffered copies data block checksums will be silently fixed by recalculating the right checksum.

7 Acknowledgments

The first person I must thank a thousand times is Dirk Reisig. It was by means of his suggestions that I sped up SuperDuper to the current, amazing level. I wrote him a letter which he answered gently with a long explanation of the optimizations performed by PCopy. The first time I read the letter it seemed greek to me, but little by little I learned all the mysteries of MFM encoding and disk direct hardware driving. Moreover, I learned from the source code of TrackSalve the usage of the blitter for MFM encoding and many other subtle things. In other words, without the help of Dirk you would have never seen anything after DFC5 (for version 2.0, a new optimization was introduced; it was suggested by Dan Babcock).

The second guy behind the birth of SuperDuper is Tom Rokicki. He pushed me to write a substitute for TurboBackup, and overall suggested the main thing—that on the Amiga it is possible to write many disks at the same time. Without this trick, you could never do four non-verified copies in 38s. Tom also tested all pre-whatever-greek-letter versions, always giving useful comments ... and risking the life of his drives 8^). Moreover, I had time to work on SuperDuper because the AmigaTeX system is so incredibly efficient I got a lot of spare time while writing math papers ...

Last but not least, Randell Jesup at Commodore drove me through the labyrinth of non-specified-specs, hardware quirks, strange behaviors, and system esoteric features. Without his help SuperDuper could probably work ... but I wouldn't trust it for *my* copies 8^).

The name SuperDuper popped up during a rather intensive BIX discussion. Many other names were proposed, but in the end I chose this one—it has symmetry, correctly defines the product and has a simple shortening (SD). Thus, a thousand thanks to Kent Kalnasy and Dan Barrans for suggesting this name.

Many features were not my ideas. Most of the design of the new GUI is due to Carlo Todeschini. Moreover, an incredible number of BIX users came up with excellent suggestions, many of which were actually implemented. Thanks to them you have support for buffering on any device (I never use RAD: nor FMS:, so I didn't think it could be useful). Nicola Salmoria tracked some of the most hidden bugs, and suggested several relevant new features.

But, as always, the biggest *thanks* goes to the beta-testers of `SuperDuper`: Dennis Atkin, Michele Battilana, Vittorio Calzolari, Jim Cooper, Doug Erdely, Charlie Fair, Blaine Gardner, Robert Jenks, John Jones, Kent Kalnasy, Robert Kesterson, Paul King, Randy Menzer, Linda Munson, Davide Repetto, Tom Rokicki, Sergio Ruocco, Carlo Santagostino, Reinhard Spisser, Jeff Todd, Carlo Todeschini, Michael Scott Velez and Marco Zandonadi. Beta-testing a copier is different from anything else—if it doesn't work you won't get a marginally corrupted picture on your display: rather, the Fish Disks it took an hour to copy could be unusable. A special kind of patience is needed under these conditions `8^)`.

8 Disclaimer and Author Info

Comments, complaints, desiderata are welcome.

```
Sebastiano Vigna
Via California 22
I-20144 Milano MI

BIX: svigna@bix.com
INTERNET: vigna@dsi.unimi.it
UUCP: seba@sebamiga.adsp.sub.org
```

Concept Index

<

A

Gadget Index

X

Program Index

A

C

D

S

T

Table of Contents