IDRIS 3.12 INSTALLATION GUIDE
FOR THE ATARI ST COMPUTERS

CONTENTS

SECTION I

IDRIS 3.12 Installation Guide for the ATARI ST Computers

INTRODUCTION

IDRIS® [1] is a complete replacement operating system for the Atari ST™ [2] series of computers based on the Motorola *MC68000* microprocessor. It provides a *multi-user* and *multi-tasking* environment very similar to that provided by the UNIX® [3] operating system.

You may partition the hard disk drive to allow a single drive to support both *TOS* and *IDRIS*. While running IDRIS support is provided to access many of the computers graphics capabilities.

PREPARING FOR INSTALLATION

Prior to installing IDRIS on your Atari ST you will need the following items:

1. An Atari ST with a minimum of 512K bytes of memory. If you intend to use STX-Windows™ [4], 1-1.5 Mega bytes of memory is suggested.
2. A monochrome or color monitor. Television through RF modulator or composite video monitor is also acceptable.
3. At least one single-sided or double-sided 3 1/2 inch floppy drive.
4. A hard-disk of at least ten (10) megabytes capacity. A five (5) megabyte drive may be workable as long as it is fully compatible with the ATARI disk drive. No five megabyte drives have been tried. If you intend to use STX-Windows, you will need to add another three megabytes of disk storage. Computer Tools International Inc., at the time of this writing, has successfully tested and run IDRIS-ST on ATARI 20 mega-byte (SH 204, 205), ATARI 40 mega-byte, Supra 20 mega-byte, and Supra 60 mega-byte disk drives. Official support for other drives will be upcoming. The full distribution is approximately three 3 megabytes. IDRIS-ST also supports the *Blitter*, if present. IDRIS-ST release 3.14 or later also support multiple drives.

In order to install IDRIS, you must first prepare the hard disk with the utilities provided by the manufacturer of the drive. Format and partition the drive using the utilities that were supplied with the drive. This will be done under the TOS operating system. When setting the hard disk partitions, you must decide how much of the drive will be allocated for TOS and how much will be allocated for IDRIS. Both TOS and IDRIS are able to use multiple partitions on the same drive. A single drive will support up to four partitions. It is recommended to allocate at least 10 mega-bytes for the main, (i.e. the *root partition*) for use by the IDRIS operating system.

INSTALLING IDRIS

The following steps to install IDRIS, can be accomplished with all floppy disks *write-protected*. To insure write-protection, check the write-protect switch of the disks, in the upper left corner. If the switch is open, (i.e. the sliding switch is pushed to the outside of the disk) the disk is write-protected.

[1] Registered Trademark of Whitesmiths Ltd.

[2] Trademark of ATARI Corp.

[3] Registered Trademark of AT&T.

[4] Trademark of Computer Tools International, Inc.

Once the hard disk is formatted and the partitions are set, you are ready to start loading IDRIS onto the hard disk. Please follow these steps.

1. Turn your computer off, and install the **boot cartridge** in the cartridge port of the computer.

2. Place the first floppy in the floppy drive. This floppy will be labeled the **Distribution Disk 1.**

3. Turn the computer on. You will see the IDRIS™bootstrap banner appear which looks like:

   ```
   ====================================
   == IDRIS-68K Loader Program (CTI) ==
   == for ATARI-ST Computers -- V2.0 ==
   ====================================

   device number (? for list):
   ```

   An *auto-boot* mechanism is built into the bootstrap program If you wait more than about twenty seconds at this or the following prompt, the system will try to boot from the hard disk, automatically. If this happens, it would fail and require you to start over again by resetting the system.

4. The bootstrap program is now waiting for you to enter the device number to boot IDRIS from. The only acceptable entries are 0 or 1. Any other value will cause a list of the acceptable values to be displayed. Enter a 0 (numeral zero) followed by a the RETURN key. This tells the bootstrap to use the floppy disk. If a 1 (numeral one) is entered (a legal value, but wrong at this point), you can use Backspace to correct this error. If all is well, you should now have the following prompt

   ```
   ATARI/CTI IDRIS bootstrap
   [cboot]?
   ```

5. Now you can enter the name of the IDRIS kernel that you desire to run. Several other commands such as *ls, cd, ld, od,* and *go* are available also. See the *IDRIS System Administration Manual* pages for more information. The backspace key works from this point on. Type in the name **idrisk1a** (pronounced *IDRIS K ONE A*) followed by the RETURN key. This causes the bootstrap to load the file */idrisk1a* from the floppy file system and then execute it. This will take about thirty seconds, after which time you should see the IDRIS sign-on banners, followed, a few seconds later, by the IDRIS command prompt. The screen should now look something like this:

   ```
   IDRIS/68k V3.12 Mon Nov 09, 1987, 16:17
   copyright (c) 1979, 1986 by Whitesmiths, Ltd.
   copyright (c) 1986, 1987 by Computer Tools International Inc.
   no mmu, system size 135.5KB
   XXX.XKB largest process size,XXX.XKB user space,50 system buffers
   fd0 root filesystem is read only
   no swapping
   IDRIS/68k V3.12 12/08/87 15:37 (Atari ST V 2.0) -fpp -sepid
   #
   ```

   You are now running IDRIS from the floppy file system. Since IDRIS is very disk intensive, things will run somewhat slowly. This will be only until you are running from the hard disk file system. Note that in the display above the values shown as *XXX.X* will be dependent on the memory size of your system.

6. Before building the IDRIS partition on the hard disk, you must flag that partition, so that the IDRIS kernel can find it. A program called show0 is provided to perform this task. At the # prompt type `show0` followed by the RETURN key. In a few seconds the screen should look something like this:

```
+-----------------------------------+----------+---------------------+
| Number of cylinders               |      306 |                     |
| Number of heads                   |        4 |                     |
| (unused)                          |        0 |                     |
| Reduced write current cylinder    |      307 |                     |
| Write-precompensation cylinder    |      307 |                     |
| Landing zone                      |        4 |                     |
| Seek rate code                    |        2 |                     |
| Interleave factor                 |        3 |                     |
| Number of sectors per track       |       34 |                     |
| Total 512-byte blocks             |    41616 |                     |
| Start of bad-sector list          |        0 |                     |
| Number of bad-sectors             |        0 |                     |
| Block checksum                    |        0 |                     |
+-----------------------+-----------+----------+----------+----------+
| Partition number      |  (hdp3)0  | (hdp4)1  | (hdp5)2  | (hdp6)3  |
| Flag byte             |       255 |      255 |        0 |        0 |
| Id string             |       GEM |      GEM |   (none) |   (none) |
| Starting block        |         1 |    20808 |        0 |        0 |
| Partition size        |     20806 |    20806 |        0 |        0 |
| Reserved blocks       |         1 |        1 |        0 |        0 |
| Size for 'mkfs'       |     20805 |    20805 |        0 |        0 |
+-----------------------+-----------+----------+----------+----------+
#
```

The most important part of this screen is the bottom four squares which describe the four possible partitions on the hard disk. What we want to do is change the *ID STRING* to IDR for the main IDRIS partition. The above example shows a 20 mega-byte disk with two 10 mega-byte partitions defined. Since partitions are numbered from zero, to set the second partition (partition number 1) to the IDRIS root, at the # prompt enter `show0 1 IDR` *XXXX* followed by a RETURN. The number 1 refers to partition number 1, and IDR is the new ID STRING to set the partition to. The value shown as *XXX* is the size of the square area, to be reserved on this partition. The following tables shows recommended sizes for different ATARI systems.

Machine	Swap Size (in blocks)	Reserved (in blocks)	Total (in blocks)
520ST, 1040ST	2048	70	2118
MEGA-2	4096	70	4166
MEGA-4	8192	70	8262

For example, if this system is being installed on a 520ST, and the second partition (partition 1), is to be the IDRIS root file system, the `show0` command will be:

```
# show0 1 IDR 2118
```

Upon executing the command, you should see another display of the current settings followed by a line saying that the name is being changed. Please note: **If the size of the swap area is less than the available RAM memory, the kernel will never swap any processes.** This may not be acceptable for smaller systems.

Now enter the `show0` command again, without arguments. You should see that the ID STRING for the partition has been changed. You may select any partition for main IDRIS partition. *IDRIS will use the first partition it finds with the ID STRING of IDR, trying in order from 0 to 3.*

7. Once you have the desired partition named, and the swap size allocated, `show0` will redisplay the current values. The last line shows the values which is the argument for the **mkfs** command. The `mkfs` command is used to initialize a disk partition, and make it look like an IDRIS *filesystem.* Any previous information in that partition is destroyed. The `mkfs` utility is not executed directly, but it is called by the shell script **BuildHard.** If it is desired to use other partitions as *mountable* filesystems under IDRIS, it is not necessary to change their ID STRINGS. The filesystem size for such a partition, is the partition size minus the reserved block(s) size rather than the swap size. If the partition is the last partition on the disk that has space allotted to it, then an extra fifty (50) blocks should be subtracted to allow for bad-blocking alternates.

8. Once you have the size argument written down (from show0 command), you must first reboot the computer. This is needed to get the kernel to read the changed information from the hard disk. Press the reset button on the computer and enter a 0 for the drive prompt and **idrisk1a** at the boot prompt as was done before. Once you have the # prompt again, we are ready to start installing the software on the hard-disk. At the # prompt, enter `BuildHard XXXXXX` followed by the RETURN, where the XXXXXX is the size that was shown by `show0` command (i.e. the partition size minus swap size and reserved blocks). Also note that, if this is the last partition on the disk then, subtract fifty (50) from this value. The command which would then be entered, followed by a RETURN, is:

   ```
   # BuildHard XXXXX
   ```

   As mentioned above the *XXXXX* value is the value shown by `show0` command after subtracting the recommended swap size for your system (and perhaps reserved blocks for bad blocking). You will be asked if you want to continue. After pressing RETURN, this command will create *(mkfs)* a file system on the hard disk, will access *(mount)* it, copys *(cp)* files to it, and then release *(mount -u)* it. This will take several minutes.

9. After the floppies have been copied to the hard disk you should reset the computer. This time, at the `device number` prompt enter the number 1 followed by a RETURN. At the `[cboot]?` prompt you still enter **idrisk1a** followed by a RETURN. Also, after the reset you can wait the twenty or so seconds and *IDRIS* will auto-boot. You must always have the boot cartridge in the cartridge port to boot IDRIS. You should now be running from the hard-disk with the IDRIS sign-on banner on the screen:

   ```
   Idris/68k V3.12 Mon Nov 09, 1987, 16:17
   copyright (c) 1979, 1986 by Whitesmiths, Ltd.
   no mmu, system size 135.5KB
   XXX.XKB largest process size,XXX.XKB user space,50 system buffers
   root filesystem
   YYYY.YKB swap space
   Idris/68k V3.12 12/08/87 (Atari 68K V2.0) -fpp -sepid
   #
   ```

   Note this is somewhat different than when you were running the kernel from the floppy. Also note that number shown as `XXX.X` and `YYYY.Y` depend on the type of the system and the swap size area. ST520™, ST1040™, MEGA-2™, and MEGA-4™ will all show different process size, user space available and swap sizes.

10. Now to load the various distribution floppys enter `Install`, followed by a return:

    ```
    # Install
    ```

    This program will loop over and over asking if you are ready to load in a floppy. First you will be asked if you really want to install the diskettes. If so enter a y followed by a return. You will then be instructed to place each of the diskettes in the drive in turn. Once a diskette is placed in the driver you will be asked to enter a RETURN to let the *Install* script know to continue.

    The order, in which the diskettes will be loaded, is:

    1. Distribution disk 1
    2. Distribution disk 2
    3. Distribution disk 3
    4. Developments utilities disk 1 (C + Pascal + Ctext)
    5. Developments utilities disk 2 (C + Pascal + Ctext)

    This list represents the *Development System*, which may not be what you have. The *Target System* will not have the **Development Utilities** diskettes. The *Professional System* will have the **C and Pascal Compiler** diskettes instead of the **Development Utilities** diskettes.

    This will complete the the installation of the standard configuration. System should be rebooted at this point so that it will boot in the full configuration. Prior to pressing the reset button, enter the command **sync** followed by a RETURN, then wait for the disk activity light to go out. When running from the hard disk, the *sync* command should always be used, if possible, prior to shutting down. Enter 1 for the drive and idrisk1a at the boot prompt.

11. The system is now ready for any customization you might wish to do. When the IDRIS kernel first comes up, it will be running the system in the single-user mode. If you enter a CONTROL-D (i.e. hold CONTROL key down and press letter D or d) at the root prompt the system will then come up in the multi-user mode. If you wish the rs232 port to be used in multi-user mode, it must be configured in the **/adm/init** file. When you first enter the multi-user mode, you will see a **login:** prompt. You must enter **root** as the login name. This will let you enter an interactive mode which looks much like the single-user mode. You should then add other login names and possibly password protection for these logins. See the system administration manual pages for more information on how to perform this task. You can also change the **options** command line in */adm/init* to suite your own preferences. The *options* command is explained in *IDRIS-ST Special Programs* section.

12. When shutting down a hard disk based IDRIS system, you should run the *sync* command a few (i.e. 2) times before you powerdown or reset the computer. After entering *sync* wait about two seconds to insure the disk is correctly updated before you press reset or turn off the power. See the *System Administration* section of the *IDRIS Users Manual* for more information.

SECTION II

CONTENTS

IDRIS Configuration on Atari ST

1. Introduction

This section describes a few aspects of **IDRIS®**-ST † combination that are noteworthy and would be useful to those who are not familiar with *IDRIS* operating system on *Atari ST*^{TM}* computers. These include a brief description of some utilities for which detailed information could be found in *IDRIS User's Guide.*

2. Memory Considerations

Since the Atari ST has no memory management hardware, each binary program on the system must tell the operating system the maximum memory needed by that program. This is set with the **setb** command. When encountering programs that complain about shortage of memory, setb could be used to determine what the current memory allocation is, and then used again to increase it. See the manuals for more information.

3. Auto Boot

As mentioned in the installation description, the system will automatically boot, upon power up, if there is no activity for twenty seconds after the bootstrap program has been read in from the boot floppy. When automatically booting, the hard disk is selected and the bootstrap loads and runs the kernel file **/idrisk1a** from the first partition with an **IDR** id-string. The twenty second delay is to allow the user time to change devices and/or kernel files. This also, gives the hard disk enough time to spin up from a cold start. By changing the */adm/init* file one can change the behavior of IDRIS after the kernel is booted. This allows one to create a totally auto booting system, from power up to running an application if desired. For unattended systems, this will allow for restarting, if power is dropped and then restored.

4. Core Dumps

The IDRIS operating system is capable of providing **core** files when a machine exception is encountered. IDRIS tries very hard to detect when a user program directs it to do something it should not, such as touch protected memory locations, doing word access on odd boundaries, or overflowing the stack or heap. When one of these *exceptions* occurs the system will print an error message and abort. It will create a file named *core* in the current directory and will write program image in this file. This core file can then be looked at, by using the **db** utility, to determine exactly what caused the exception. The kernel can be modified to create core files only if a file named *core* already exists in the current directory; see the next section on **Kernel Variables.**

5. Kernel Variables

The following variables in the kernel file *idrisk1a* can be configured using the *db* utility. See the *db* manual pages for more information on using this utility.

† Registered trademark of Whitesmiths LTD.

* Trademark of ATARI CORP.

5.1 Variable `_nbufs`

This is a 32-bit value which is *the number of 512-byte disk buffers* to be reserved in memory. The default value is 50. Making this number bigger, while perhaps helping performance, leaves less memory for use by programs. The IDRIS sign-on banner shows the current value.

5.2 Variable `__mdpages`

This variable is *the number of 512-byte shared data pages* to be reserved. They are used by the IDRIS *real-time* extensions.

If the user wants to setup a **RAM-Disk**, this variable needs to be given a value of *500* which is the default value used by `/usr/bin/Memdisk` shell script.

5.3 Variable `_coremode`

This variable is a 32-bit value which *controls creation of core files.* If it is set to 0 then core files are never created. If set to 1 then core files are dumped only if a file named *core* already exists in the current directory. If set to 2, core files are dumped whenever needed. The default value is 2. Note that existence of a *core* file (needed if `_coremode` is set to 1), does not require existence of any data inside the file.

5.4 Variable `_nodename`

This variable is *an 8 byte string space for the system nodename* as reported by the program or system call **uname.** Since it must be null terminated, the name should be seven characters or less. The default name is **atarist.**

6. Changing Kernel Variables

The instructions in this section will try to illustrate the steps needed to change kernel variable to tune system performance. The following example will show how to change the `_nbufs` kernel variable which effects system performance quite heavily.

6.1 An Example

As mentioned above the `_nbufs` kernel variable controls the number of 512 byte disk buffers used by the IDRIS kernel. This value is defaulted to 50, which is a good number for a ST520, or ST1040. On a machine with more memory, bigger load or more users, increasing this value can cause significant performance improvements. The maximum value for this variable is 400. Empirical data, however, has shown that the optimum value for this variable for a MEGA-2 or MEGA-4 is 200.

The following steps will show how to use *db* to change the value of this variable. *Extreme care must be taken when performing these operations.* When using *db* on the kernel file the changes you make are permanent in the kernel file and will become effective once you reboot with the new kernel. You must b logged in as *root* or use the *su* command to become *super-user* in order to perform these operations.

1. First make a copy of the kernel file for editing purposes.

```
# cp /idriskla /newidris
```

2. Now perform the changes on the new copy of the kernel file. Invoke the binary with the update option, giving it the kernel file name.

```
# db -u /newidris
```

3. Once `db` returns with information about the kernel *text* , *data* and *bss* sizes, instruct `db` to show the current value of the variable.

   ```
   _nbufs pi
   ```

   This instruction asks the binary editor to print (*p*) the value of `_nbufs` as an integer (*i*).

4. Instruct `db` to change the value of the variable. In the following instructions the value referred to as *previous value*, within the substitution directive refer to the value returned by the previous print command.

   ```
   _nbufs s/previous value/200/
   ```

   This instruction tells the editor to substitute (*s*) for the previous value, the new value whichs is *200*. Those who have worked with either the UNIX line editor, *ed* or the IDRIS line editor *e* will notice the command syntax similarity.

5. Exit the editor. Note that in the case of the binary editor there is no need to save anything before exiting the editor. All of the updates are made directly to the file, since the –*u* flag was used.

   ```
   q
   ```

   This terminates the session with `db`.

SECTION III

NAME

intro - manual pages specifice to the IDRIS for the ATARI ST.

SYNOPSIS

Introduction

DESCRIPTION

This section provides the manual pages for certain nonstandard IDRIS utilities which are only part of the IDRIS 3.1 for the ST, distribution. These are usually special purpose utilities that are useful on ATARI ST hardware. The following special programs are in */usr/bin* directory.

NAME

call - call out from the ST rs232 port

SYNOPSIS

```
call
```

FUNCTION

This is a small shell script that allows the user to make a call out the rs232 port, using the ATARI ST as a terminal. See the manual pages on *cu* for more information. The call script currently assumes 9600 baud.

NAME

clear - clear the display screen.

SYNOPSIS

clear

FUNCTION

This program uses the *termcap* facilities to clear a terminal screen. It requires that the TERM variable in the environment be correctly set.

AUTHOR

Computer Tools International Inc.

NAME

dir - give a wide directory listing

SYNOPSIS

```
dir
```

FUNCTION

This is a small utility that gives wide directory listings. It marks executable programs with an asterisk * and directories with a slash /. It only works in the current directory. Arguments may be given, but they only qualify the current directory. Perhaps `ls | mc` is as good, except for marking file types.

NOTES

This program does not work on other than current directory.

AUTHOR

David M. Stanhope
Computer Tools International, Inc.

NAME

diskformat - format floppy disks

SYNOPSIS

```
diskformat [interleave <value>] [double] [second] [silent]
```

FUNCTION

This program allows you to format floppy disks on the first or second floppy drive. You can specify an interleave, and format single or double sided. Floppy-disks must be formatted before they can be used by IDRIS. Floppy disks can be formatted TOS also. Valid arguments are:

`interleave <value>`

This argument requires another which is the interleave value to use. This value should be in the range of [1,8]. If interleave is not specified, it is defaulted to 1. The *interleave* value has to do with how sectors are laid out on the disk and playing with this value can effect performance of the floppy disk.

`double`

This argument tells the program to format the drive double-sided, instead single-sided.

`second`

This argument tells the program to use the second drive, instead of the first.

`silent`

Normally the format program will prompt you for a RETURN before doing the actual format operation, also while formatting the floppy it will tell about it's progress. The silent arguments stops the format program from waiting for a RETURN and does not show progress information.

EXAMPLE

To format a double sided floppy with interleave 3 on the second drive:

```
# diskformat second double interleave 3
```

To format a single sided disk in the first drive with interleave of 1 enter:

```
# diskformat
```

This is the default action.

AUTHOR

David M. Stanhope
Computer Tools International, Inc.

NAME

fixtty - reset terminal to a sane state

SYNOPSIS

```
fixtty
```

FUNCTION

This is a simple script that can be used to reset a terminal port back to a reasonable state. Exiting incorrectly out of programs that changes terminal settings can leave echoing and carriage-return/new-line mapping off. If so enter a line-feed, then fixtty and then another line-feed to get things back to normal.

NAME

options - display and/or change console display parameters.

SYNOPSIS

```
options dump
options background <hex value>
options foreground <hex value>
options resolution [low,medium]
options cursor [on,off,steady,blink]
options palette <sixteen (16) three digit hex values>
options color <decimal index> <a three digit hex value>
options kbrate <decimal initial> <decimal repeat>
options [blitter,bell,repeat,click,wrap,kbshift] [on,off]
```

FUNCTION

This utility allows you to display and also change the console display parameters. If run with no arguments it will show the possible command arguments that may be used.

If the dump flag is given a list of the current settings will be given.

```
# options dump
Current Screen Resolution is <1> <medium>!
Palette: <0x000,0x700,0x007,0x070,0x007,0x707,0x077,0x555>!
         <0x333,0x733,0x373,0x773,0x337,0x737,0x377,0x000>!
Term control <0x6> kbshift <off> bell <on> repeat <on> click <off>!
Keyboard repeat parameters: initial <15> repeat <2>!
```

The background and foreground colors may be changed by specifying the index into the color maps. The number of possible values depends on the screen resolutions. For low there are 16 colors [0-f], for medium there are 4 colors [0-3], and for high resolutions there are 2 colors [0-1]. The background color is the color that is written to the screen as the text background, the foreground color is the actual color of the text. The actual colors will depend on the current color-map in effect, since only an index is given.

```
# options foreground 3
# options background f
```

The screen resolutions may be changed with the use of the resolution flag. However, since the high resolution monitor only allows high resolution, this parameter is only effective for the color monitor and can only go back and forth between low and medium resolution.

```
# options resolution low
# options resolution medium
```

The cursor may be turned on, off or set to either steady or blinking by using cursor flag.

```
# options cursor on
# options cursor off
# options cursor steady
# options cursor blink
```

The entire sixteen entry color map may be loaded with the palette argument. You are required to load all 16 values even if the current screen resolution does not support all of them. Each color-map value is given as a three byte hex value. The first digit is the red component, the second is the green, and the third is the blue. Each digit can only range from 0 through 7.

```
# options palette 777 700 070 770 007 707 077 555 \
                  333 733 373 773 337 737 377 000
```

The color argument allows one to change just a single color map value instead of all 16 at once. Again the colors are given as three digit hex numbers with the digits ranging from 0 to 3 (octal). The index to the map value to be changed is given in decimal.

```
# options color 3 070
```

The keyboard repeat parameters may be changed with the kbrate argument. It requires and initial delay value which is the time from the key-press until the key-repeat starts. Also the time between key-repeats must be given. Both values are specified as decimal values which is in system clock ticks, which are 1/50 of a second. The following example sets 25/50 or 1/2 second till the key repeat starts with 1/50 of a second between repeats.

```
# options kbrate 25 1
```

The console bell, key repeat, key click, screen wrap, and keyboard shift reporting can all be turned on and off. Screen wrap controls if the screen output wraps to the next line after writing to the last character position on a line. The keyboard shift reporting only effects input from /dev/ikbd. See the device discussion for more information.

```
# options blitter on
# options blitter off
# options bell on
# options bell off
# options wrap on
# options wrap off
# options click on
# options click off
# options repeat on
# options repeat off
# options kbshift on
# options kbshift off
```

Multiple arguments may be given on the same command line.

```
# options echo on cursor on cursor steady click off
```

NOTES

Changing screen resolution has a timing constraint which precludes changing other parameters at the same time.

AUTHOR

David M. Stanhope
Computer Tools International, Inc.

NAME

readbpb - read disk parameter block

SYNOPSIS

```
readbpb <device name>
```

FUNCTION

This program prints out the current *disk parameter block* which is in the first block of a floppy. It requires one argument which must be the device name of a device that can access the first physical block of the disk. This program shows for floppies much of the same information that **show0** does for the hard-disk.

EXAMPLE

```
# readbpb /dev/fd0_all
    #bytes/sector                  512
    #sectors/cluster                 2
    #reserved sectors               18
    #of FATs                         2
    #of root directory entries     112
    #of sectors on media           720
    media descriptor byte          248
    #sectors/FAT                     5
    #sectors/track                   9
    #sides on media                  1
    #hidden sectors                  0
```

AUTHOR

David M. Stanhope
Computer Tools International, Inc.

NAME

setres - change the resolution of the console

SYNOPSIS

```
setlow
setmedium
```

FUNCTION

Setlow is a shell script that changes the console from medium resolution to low resolution. It is a readable text file which you can change as desired.

Setmedium is a script that changes the console from low resolution to medium resolution. It is a readable text file which you can change as desired.

NAME

show0 - show current hard disk partition setup

SYNAPSIS

```
showO [partition-number id-string]
```

FUNCTION

This program allows you to see the current partition setup of the hard-disk. It also allows you to change the 3-character name of a partition. If the command is run without any arguments it will just display the current partition information. If it is followed by a partition number [0-3] and a three character string, the *id–string* of the specified partition will be changed to the given string. If the partition *id–string* is followed by a number, the number is taken to be the number of blocks to be reserved for administrative purposes. This program is mainly used to tell IDRIS which partition is the root partition on the hard-disk, and how many reserved blocks blocks it should have (i.e. swap area). IDRIS will use the lowest partition-number that it finds with the *id–string* of IDR.

NOTES

This program can be moved to */usr/bin* or removed after system installation.

AUTHOR

David M. Stanhope
Computer Tools International, Inc.

NAME

showpic - show DEGAS or NeoChrome pictures

SYNOPSIS

```
showpic <filename> [restart] [wait] [##]
```

FUNCTION

Showpic shows DEGAS or NeoChrome pictures which are stored in *filename*. The filename has to have a three letter extension. The first letter of the extension is always **p** for **picture**. The two characters following this character are, the picture *type* and picture *display resolution*. Picture type can be **d** (for DEGAS) or **n** (for NeoChrome). The display resolution is one of the following: **l**, for low resolution picture, **m**, for medium resolution pictures, and **h** for high resolution pictures.

If *restart* is specified, showpic will cycle through the command line arguments starting at the first argument. If showpic sees any of the characters **q** or **ESCAPE** or **RETURN** from the keyboard, it will stop. If *wait* is specified, showpic always waits for keyboard input before proceeding to the next command line argument. If a number ## is specified, showpic will go to sleep for the specified number of seconds, and once it wakes up it proceeds with the next command line argument.

SEE ALSO

tosdir(1), tosget(1).

AUTHOR

David M. Stanhope
Computer Tools International, Inc.

NAME

tosdir - give a directory listing of a TOS directory

SYNOPSIS

```
tosdir [TOS directory specification]
```

FUNCTION

Tosdir displays a listing of a TOS directory. Directory specification is done in the following manner:

TOS directory	Physical Device
a:	First floppy disk
b:	Second floppy disk
c:	First hard-disk partition
d:	Second hard-disk partition
e:	Third hard-disk partition
f:	Fourth hard-disk partition

The default directory is **a:**. If the environment variable **TOSCWD** is set, it serves as a prefix when searching for a TOS file. When specifying the filename, the / character should be used instead of \ character. If the directory specification begins with a drive specification (i.e *a:*), or a forward slash (i.e. /), the *TOSCWD* environment variable is ignored.

SEE ALSO

tosget(1).

EXAMPLE

To see a directory listing in first hard-disk partition:

```
% tosdir c:
```

FILES

/dev/hdp7, /dev/*_all.

AUTHOR

David M. Stanhope
Computer Tools International, Inc.

NAME

tosget - get a file from a TOS directory

SYNOPSIS

```
tosget [TOS directory specification]/<TOS filename>
```

FUNCTION

Tosget reads a TOS file specified by *filename* and outputs it to the *stdout* . Directory specification is done in the following manner:

TOS directory	Physical Device
a:	First floppy disk
b:	Second floppy disk
c:	First hard-disk partition
d:	Second hard-disk partition
e:	Third hard-disk partition
f:	Fourth hard-disk partition

The default directory is a:. If the environment variable **TOSCWD** is set, it serves as a prefix when searching for a TOS file. When specifying the filename, the / character should be used instead of \ character. If the directory specification begins with a drive specification (i.e *a:*), or a forward slash (i.e. /), the *TOSCWD* environment variable is ignored.

SEE ALSO

tosdir(1).

EXAMPLE

To retrieve a TOS file from the first hard-disk partition:

```
% tosget c:xyz > xyz
```

FILES

/dev/hdp7, /dev/*_all.

AUTHOR

David M. Stanhope
Computer Tools International, Inc.

SECTION IV

CONTENTS

IDRIS-ST Special Files

1. Introduction

IDRIS® † operating system provides a coherent interface between user programs and files/devices. The notion of files holds true for physical device by providing *device drivers* for each physical device. Device drivers emulate standard file operations. These operations are *read*, *write* and *seek* to name a few. It must however be noted that not all file operations are legal on all devices. Devices are known to the file system through certain special files which are normally located in the **/dev** directory. The following sections explain the naming convention for these files and how they could be recreated if needed.

2. Device Naming Conventions

The following sections list all devices on the system and their uses. After the name of each device is the *type* which is either the letter **c** or **b**. The letter **c** denotes a character special device while the letter **b** denotes block special devices. Then the *major* and *minor device numbers* are given. The type, major and minor numbers are the arguments needed by the **mkdev** program. The letter **r** is prefixed to certain block device names refers to the same device in the *raw* mode. These devices are always character special.

3. IDRIS-ST Devices

These devices are divided into five sections. Lists of files in tables *1*, *5* and *6* are common to all IDRIS system. Further description could be found in *IDRIS Users Manual* under *Standard File Formats* section. Lists in other tables describe special files which are particular to the ATARI ST computers. It should be noted that special files refer to both *devices* and *pseudo devices*.

3.1 Kernel Devices

These devices provide windows into the IDRIS kernel currently running. They are mostly used in status reporting and performing administrative tasks.

† **Registered trademark of Whitesmiths Ltd.**

Table 1: Kernel Files

Name	*Type*	*Major*	*Minor*	*Description*
NODEV	c	0	0	A generally illegal looking device.
kmem	c	0	1	Access to kernel memory.
null	c	0	2	Empty file for read, bottomless pit for write.
ps	c	0	3	System process list.
myps	c	0	4	User process list.
mount	c	0	5	Kernel mount table.
inode	c	0	6	Kernel inode list.
where	c	0	7	The where string.
bnames	c	0	8	List of block devices.
cnames	c	0	9	List of character devices.
zero	c	0	10	Limitless source of zeros.
tty	c	0	11	Always the controlling tty.
stat	c	0	12	

3.2 IDRIS-ST Character Devices

The following are the *IDRIS-ST* character I/O devices. If 128 or 0x80 is added to the device minor number, as for **/dev/lp**, then the device is set so that only one open, at a time, is allowed on that device. This exclusive open feature is needed by the *lpr* program to keep several people from time-sharing the printer and disrupting each others printouts.

Table 2: IDRIS-ST Character Devices

Name	*Type*	*Major*	*Minor*	*Description*
console	c	1	0	The Atari console.
midi	c	1	1	The Atari midi ports.
mouse	c	1	2	The Atari mouse/joystick port.
ikbd	c	1	3	The Atari keyboard (direct).
rs232	c	2	0	The Atari RS232 port.
lnk0	c	2	0	Another name for the RS232 port, default for cu or kermit.
parallel	c	3	0	The Atari parallel port.
lp	c	3	128	Another name for the parallel port, default for lpr.

Only the *rs232* device allows selectable baud rates. It always holds **RTS** (pin-4) high. When opened, it asserts **DTR** (pin-20), and when closed, it de-asserts DTR. Pin-5 **CTS** must be asserted in order for the system to acknowledge the port.

The *mouse* and *ikbd* devices communicate with a special protocol. Writes to the mouse or ikbd are directed to the intelligent keyboard controller. Since these devices send back multiple characters, it is best to avoid single character read requests, as each read has a significant overhead. It is much easier to stay ahead of the device by setting *O_NDELAY* on *open* and always reading as much as possible.

When the ikbd is open, all key-press data is diverted from the console to the **ikbd** device. Each key-press sends four (4) bytes to the ikbd device. The first byte is always **0xff** to help the reader stay in sync with the device. The second is the standard ASCII character as would normally be seen by reading **/dev/console**. The third byte is the *scan code* for each key. The scan code is different for each key, and may be affected by *control*, *shift*, *caps-lock* and the *alternate* keys. The last byte is a flag byte which has the current state of the shift, control, caps-lock, and the alternate keys. This last byte will always be zero unless the *kbshift* flag is turned on with the **options** command. The main use of */dev/ikbd* is so that keys, such as function keys which normally do not pass character to /dev/console, can be used.

Talking to /**dev**/**mouse** is much more complex. The mouse port has several modes which can be programmed by writing to the mouse device.

3.3 IDRIS-ST Block Devices

The ATARI ST block special devices are the floppy disk and the hard disk. Each block special device also has a file which accesses that device in raw mode (i.e. *character special*) manner.

3.3.1 Floppy Disk Devices For disks, both hard and floppy, the block devices are normally used for reading and writing data to and from the disks. The *raw* devices are used to send **ioctl** signals to the drive.

Table 3: IDRIS-ST Floppy Special Files

Name	*Type*	*Major*	*Minor*	*Description*
fd0	b	4	0	The first floppydisk, starting past reserved sectors to the end of disk.
fd0_all	b	4	128	The first floppydisk, from physical sector 0 to disk end.
fdl	b	4	1	The second floppydisk, starting past reserved sectors to the end of disk.
fdl_all	b	4	129	The second floppydisk, from physical sector 0 to disk end.
fd	b	4	0	Another shorthand name for fd0.
fd_all	b	4	128	Another shorthand name for fd0_all.
rfd0	c	4	0	The first floppydisk, starting past reserved sectors to the end of disk.
rfd0_all	c	4	128	The first floppydisk, from physical sector 0 to disk end.
rfdl	c	4	1	The second floppydisk, starting past reserved sectors to end of disk.
rfdl_all	c	4	129	The second floppydisk, from physical sector 0 to end of disk.
rfd	c	4	0	Another shorthand name for rfd0.
rfd_all	c	4	128	Another shorthand name for rfd0_all.

The only *ioctl* call currently supported, is used to format floppy disks. It should only be used on the device that covers the entire disk. You should note that if 128 is added to the minor device, it means access to the entire floppy, not after the reserved sectors, as defined by the disk parameter block in the first physical sector of the disk.

Normally after formatting a floppy it only has one reserved sector. **Single-sided** floppy disks have a total of *720* 512-byte blocks, while **double-sided** floppies have *1440* 512-byte blocks. These are the values to pass to *mkfs* when using the device that specifies the entire disk. For normal disk formats, subtract one for the devices that starts after the reserved sectors.

You only need to *make a filesystem* by using mkfs on a floppy (after formatting), if you want to mount the disk as another filesystem. You can save and restore data without doing the mkfs, if you use programs such as *tp*, *tar*, *dump*, and

Each format ioctl call, formats a single track with the specified format. The description of the floppy-format ioctl is:

```
#define FMT_MAGIC 0x87654321
#define FMT_IOCTL 0x2000
struct fmt_req          /* both longs and ints are 32-bits here */
    {
    int side        ; /* side to format (0 or 1)              */
    int track       ; /* track to format (0 to 79 or ??)      */
    int nsects      ; /* number of sectors per track (9 or ?) */
    int interleave  ; /* interleave value for the track       */
    int virgin      ; /* 16 bit value to init. sectors to     */
    long magic      ; /* set to FMT_MAGIC, for cmd verify     */
    } fmt_cmd       ;

ioctl(fd, FMT_IOCTL, &fmt_cmd) ;
```

3.3.2 Hard Disk Devices The physical hard disk device is usually partitioned into several *logical* hard disk devices. Device files are provided to access these device as whole or separate devices. The following tables describe these devices:

Table 4: IDRIS-ST Hard Disk Special Files

Name	*Type*	*Major*	*Minor*	*Description*
rhdp0	c	5	0	The boot blocks reserved in the IDR partition.
rhdp1	c	5	1	The swap space reserved in the IDR partition.
rhdp2	c	5	2	The actual root logical drive in the IDR partition.
rhdp3	c	5	3	The first hard-disk partition.
rhdp4	c	5	4	The second hard-disk partition.
rhdp5	c	5	5	The third hard-disk partition.
rhdp6	c	5	6	The forth hard-disk partition.
rhdp7	c	5	7	This device covers the entire hard disk.
hdp0	b	5	0	The boot blocks reserved in the IDR partition.
hdp1	b	5	1	The swap space reserved in the IDR partition.
hdp2	b	5	2	The actual root logical drive in the IDR partition.
hdp3	b	5	3	The first hard-disk partition.
hdp4	b	5	4	The second hard-disk partition.
hdp5	b	5	5	The third hard-disk partition.
hdp6	b	5	6	The forth hard-disk partition.
hdp7	b	5	7	This device covers the entire hard disk.
hdboot	b	5	0	Another name for hdp0.
hdswap	b	5	1	Another name for hdp1.
hdroot	b	5	2	Another name for hdp2.
hdp3root	b	5	131	The first hard-disk partition, if it was built as a root partition.
hdp4root	b	5	132	The second hard-disk partition, if it was built as a root partition.
hdp5root	b	5	133	The third hard-disk partition, if it was built as a root partition.
hdp6root	b	5	134	The forth hard-disk partition, if it was built as a root partition.

The high bit (0x80 or decimal 128) is used to mark a hard disk partition as having been built as a *root* type partition. As above *hdp3root* has a minor number of 131 which is 128 + 3. Normally there is only one root partition, but in special cases, it might be desired to make more than one. To boot from a different root partition, the show0 utility must be used to relabel the disk. As mentioned in the *IDRIS 3.12 Installation Manual*, auto boot will always choose the first partition with an *IDR* id-string.

The hard disk will almost always be accessed using the block device names. The partitions *hdp3*, *hdp4*, *hdp5*, and *hdp6* map to the four partitions defined on the hard disk when the drive is first setup under *TOS*†. These partitions will have the sizes given to them, when the disk was initialized. Some may be of zero size. When IDRIS is first loaded onto the drive, one of these partitions is labeled the IDR partition. IDRIS then further divides that partition into three (3) partitions, the **boot**, **swap**, and **root** devices. The non IDR partitions may be accessed by using hdp3, hdp4, hdp5 and hdp6 devices.

Partitions hdp3 through hdp6 may be reserved for TOS or IDRIS. Under IDRIS, filesystems may be created on these partitions and mounted from the root device.

3.4 IDRIS Pseudo Devices

These files are used in accessing the *shared memory* area or the *message queues*.

Table 5: IDRIS Pseudo Devices

Name	*Type*	*Major*	*Minor*	*Description*
rmd	c	6	0	Memory disk/shared data device.
rsd0	c	6	1	Memory disk/shared data device.
rsd1	c	6	2	Memory disk/shared data device.
rsd2	c	6	3	Memory disk/shared data device.
rsd3	c	6	4	Memory disk/shared data device.
rsd4	c	6	5	Memory disk/shared data device.
riopage	c	6	6	Memory disk/shared data device.
md	b	6	0	Memory disk/shared data device.
sd0	b	6	1	Memory disk/shared data device.
sd1	b	6	2	Memory disk/shared data device.
sd2	b	6	3	Memory disk/shared data device.
sd3	b	6	4	Memory disk/shared data device.
sd4	b	6	5	Memory disk/shared data device.
iopage	b	6	6	Memory disk/shared data device.
rmsg	c	7	0	Message system device.
rmsg2	c	7	1	Message system device.
rmsg3	c	7	2	Message system device.
rmsg4	c	7	3	Message system device.
msg	b	7	0	Message system device.
msg2	b	7	1	Message system device.
msg3	b	7	2	Message system device.
msg4	b	7	3	Message system device.

3.5 IDRIS Floating Point Device

Although not used in the current models of ATARI ST computers this file is provided for floating point units where available.

Table 6: IDRIS Floating Point Device

Name	*Type*	*Major*	*Minor*	*Description*
rfpp	c	8	0	Floating point processor device.

† Trademark of ATARI CORP.

4. Further Descriptions

For a description of standard IDRIS devices, please refer to *IDRIS User's Manual* and *IDRIS Programmer's Manual.*

SECTION V

CONTENTS

IDRIS–ST Interface Specification

1. Introduction

IDRIS operating system is a complete replacement operating system for the **Atari ST** series computers. IDRIS provides *multi-tasking* and *multi-user* capabilities on a wide range of computers and microprocessors. This document explains certain features of the ST firmware and hardware and their use in conjunction with IDRIS operating system. A generality which could be made is that *almost all hardware and firmware features can be used except when their use violates the integrity of the operating system and user processes.* The following sections will explain some of these finer points in more detail.

2. Supported TOS Memory Locations

The following are the reserved low memory locations that are known to be valid, and are considered worthwhile and reasonably safe to use. Many others are valid but are controlled by the ROMs and are either dangerous to play with, and/or the same effect can be had using the **XBIOS** or **VDI** calls. You must either use */dev/kmem* or the XBIOS call to put you in the supervisor mode. **Do not try to issue IDRIS system calls while in the supervisor state, your system will crash.**

Location	*Description*
`0x440`	(Word) Is seek rate for floppy. 0 = 6ms, 1 = 12ms, 2 = 2ms, 3 = 3ms.
`0x484`	(Byte)
`bit 0`	If it is 0 then key-click is disabled, otherwise if set to 1 then key-click is enabled.
`bit 1`	If it is 0 then key-repeat is disabled, otherwise if it is set to 1 then key-repeat is enabled.
`bit 2`	If it is 0 then bell is disabled, otherwise if it is set to 1 then bell is enabled.
`bit 3`	If it is 0 then kbshift is disabled, otherwise if it is set to 1 then kbshift is enabled.
`0x4bc`	(Long) System counter that increments at 200hz.

3. Using ROM System Calls

When using **ROM** system calls, *do not* have two processes calling the screen routines at the same time. Both the *IDRIS* and *GEM*™† output functions call the ROM screen routines and at that level, there are no mechanisms to insure that they do not call non reentrant code at the same time. It is perfectly legal and safe for two or more programs to output to the console as long as they all use the standard IDRIS output calls. However, one can not mix non-IDRIS calls with either other non-IDRIS calls or IDRIS calls. To do so can cause an IDRIS *panic trap.* This is usually not all that destructive, but will require that IDRIS be rebooted. One way this can happen is if you have terminal echo on while calling the GEM functions to draw on the screen. It is usually best to disable echo before you start using the GEM functions.

† **Trademark of Digital Research Inc.**

4. Support for GEMDOS Calls

There is no support for **GEMDOS** (i.e. *trap #1*) calls. They will cause a system crash.

5. Support for BIOS Calls

There is no support for **BIOS** (i.e. *trap #13*) calls. These also will cause a system crash.

6. Support for XBIOS Calls

The following **XBIOS** calls, which are accessed through *trap #14* should be safe to use from within IDRIS. Not all have completely been tested, although, they should have no ill effects if used wisely. When using these functions you are bypassing all of the IDRIS protection mechanisms, so sending bad data to the function may crash the machine. Other, yet unknown, effects may appear which may cause unpredictable behavior. These functions should be used with care.

To use these calls, include the file *gembind.h* and link to the library file *libgem.o*. Both are in */usr/lib* directory.

Function	*Description*
`Physbase()`	Returns physical base address of the screen.
`Logbase()`	Returns logical base address of the screen.
`Getrez()`	Returns resolution of the screen.
`Setscreen(log_adr, phy_adr, rez)`	Set logical base address, physical base address, and resolution of the screen.
`Setpallete(map_adr)`	Install new sixteen word color map.
`Setcolor(map_index, map_value)`	Change a single color map value.
`Random()`	Return a 24-bit random value.
`Cursconf(func, blink_time)`	Set cursor configuration.
`Giaccess(value, register)`	Read/write sound chip register.
`Offgibit(bit)`	Clear sound chip port-A bit.
`Ongibit(bit)`	Set sound chip port-A bit.
`Dosound(sound_ptr)`	Setup for interrupt driven sound output.
`Kbrate(initial_dly, repeat_dly)`	Set keyboard-repeat parameters.
`Vsync()`	Wait for the next vertical-blank interrupt.
`Supexec(function_address)`	Execute code at given address in supervisor mode, do not use IDRIS system calls while in supervisor mode.

7. Support for VDI Calls

The following VDI calls are tested and are safe to use from within IDRIS. Cautions and warnings given for XBIOS calls, also apply here.

7.1 VDI Graphic routines

To use these calls include the file *gembind.h* and link to the library file *libgem.o*. Both are in */usr/lib* directory.

The parameters *xyarray* and *pxyarray* are arrays of shorts which contain *xy* coordinates. Angles are specified in 10th of degrees as short values.

```
v_arc(handle, x, y, radius, begang, endang)
v_bar(handle, pxarray)
v_cellarray(handle, pxyarray, rowlen, elused, numrows, wrtmode, colarray)
v_circle(handle, x, y, radius)
v_clrwk(handle)
v_clswk(handle)
v_contourfl(handle, x, y, index)
v_curdown(handle)
v_curhome(handle)
v_curleft(handle)
v_curright(handle)
v_curtext(handle, &string)
v_curup(handle)
v_dspcur(handle, x, y)
v_eeol(handle)
v_eeos(handle)
v_ellarc(handle, x, y, xrad, yrad, bang, eang)
v_ellipse(handle, x, y, xradius, yradius)
v_ellpie(handle, x, y, xrad, yrad, bang, eang)
v_enter_cur(handle)
v_exit_cur(handle)
v_fillarea(handle, count, pxyarray)
v_get_pixel(handle, x, y, pel, index)
v_gtext(handle, x, y, string)
v_hide_c(handle)
v_justified(handle, x, y, string, length, wordspace, charspace)
v_opnwk(work_in, &handle, work_out)
v_pieslice(handle, x, y, radius, begang, endang)
v_pline(handle, count, pxyarray)
v_pmarker(handle, count, pxyarray)
v_rbox(handle, xyarray)
v_rfbox(handle, xyarray)
v_rmcur(handle)
v_rvoff(handle)
v_rvon(handle)
v_show_c(handle, reset)
v_updwk(handle)
vq_cellarray(handle, xyary, rlen, nrows, &elu, &rused, &status, colarray)
vq_chcells(handle, &rows, &columns)
vq_color(handle, index, setflag, rgb)
vq_curaddr(handle, &row, &column)
vq_extnd(handle, owflag, workout)
vqf_attrib(handle, attrib)
vql_attrib(handle, attrib)
vqm_attrib(handle, attrib)
vqt_attrib(handle, attrib)
vqt_extent(handle, string, extent)
vqt_fnt_info(handle, &minADE, &maxADE, distances, &maxwidth, effects)
vqt_name(handle, number, name)
vqt_width(handle, char, &width, &ldlt, &rdlt)
vr_recfl(handle, pxyarray)
vr_trnfm(handle, srcMFDB, dstMFDB)
vro_cpyfm(handle, wmd, xyarray, sMFDB, dMFDB)
vrt_cpyfm(handle, wmd, xyarray, sMFDB, dMFDB, id)
vs_clip(handle, clip_flag, pxyarray)
vs_color(handle, index, rgbvalue)
vs_curaddr(handle, row, column)
vsc_form(handle, cur_form)
vsf_color(handle, index)
vsf_interior(handle, style)
vsf_perim(handle, perimeter)
vsf_style(handle, styleindex)
vsf_udpat(handle, pattern, planes)
vsl_color(handle, index)
vsl_ends(handle, begstyle, endstyle)
vsl_type(handle, style)
vsl_udsty(handle, pattern)
vsl_width(handle, width)
vsm_color(handle, index)
vsm_height(handle, height)
vsm_type(handle, symbol)
```

```
vst_align(handle, ihor, ivert, ohor, overt)
vst_color(handle, index)
vst_effects(handle, effect)
vst_font(handle, font)
vst_height(handle, height, &charwidth, &charheight, &cellwidth, &cellheight)
vst_point(handle, point, &charwidth, &charheight, &cellwidth, &cellheight)
vst_rotation(handle, angle)
vswr_mode(handle, mode)
```

Most other VDI calls are redefined in *gembind.h* so that they will fail in the link phase of compilation under IDRIS. This is because the version of libgem.o which is supplied with cross-development utilities, supports these calls for execution *TOS* environment.

7.2 Other LIBVDI Functions

The following functions are available in the *libgem.o* library. These are useful ways for dealing with character input/output and opening and closing the graphics display. Some of the following functions expect a valid *fd* be passed to them. The fd is a *file descriptor* to a valid open *character device*.

7.2.1 The Get_Handle Function. Returns an integer handle which can be used by the VDI functions.

```
int Get_Handle(x_width, y_width, max_colors)
    short int *x_width    ;
    short int *y_width    ;
    short int *max_colors ;
```

It returns the *x* and *y* size of the screen in pixels as well as the number of colors supported. It returns the values in the three short integers whose addresses are passed as arguments. The screen is cleared and the cursor is disabled. This routine calls the VDI function *v_opnwk*. It is not possible to use the *v_opnvwk* call under IDRIS as it attempts to call GEMDOS functions which will cause IDRIS to crash. Since v_opnvwk clears and resets the screen to the default color map, the Get_Handle call first save the current color map, calls v_opnwk, then quickly replaces the color map with the saved one. This gives much the same effect as v_opnvwk.

7.2.2 The Close_Handle Function. Requires no arguments and closes the screen that was opened with Get_Handle.

```
void Close_Handle()
```

If you exit a program without closing, you may find that you have no cursor. If so, use the options command to restore it.

7.2.3 The Save_term Function. Returns the current state of a device which can later be restored using *Restore_Term.*

```
int Save_term(fd)
    int fd ;
```

The caller must save the returned value. It is useful to save the device state before making calls to functions such as *Raw_On*, *Raw_Off* , *Echo_On* and *Echo_Off.* By using this process one can restore the device to the state it was in upon entry.

7.2.4 The Restore_Term Function. Restores a character device to the state which was saved by the *Save_term* call.

```
void Restore_Term(fd,sav_mode)
```

```
    int fd       ;
    int sav_mode ;
```

7.2.5 The Raw_On Function. Places a character device in the *raw* mode.

```
void Raw_On(fd)
    int fd ;
```

7.2.6 The Raw_Off Function. Places a character device in the *cooked* (i.e. *non*-raw) mode.

```
void Raw_Off(fd)
    int fd ;
```

7.2.7 The Echo_On Function. Turns *on* the character device input echo.

```
void Echo_On(fd)
    int fd ;
```

7.2.8 The Echo_Off Function. Turns *off* the character device input echo.

```
void Echo_Off(fd)
    int fd ;
```

7.2.9 The Bconstat, Bconin Functions. For these functions to be useful, you should insure the device is in the raw mode (see *Raw_On()* above). These functions work similar to the like-named TOS functions. Instead of passing a device number you must pass a valid open file descriptor.

Bconstat performs a non-blocking test to see if any input characters are available on the device, returns 0 if no characters are there, -1 if characters are ready.

```
int Bconstat(fd)
    int fd ;
```

Bconin returns the next available input character, is blocking unless *O_NDELAY* was set on the device.

```
int Bconin(fd)
    int fd ;
```

7.2.10 The Bconout Function. Outputs the character *c* to the device specified by *fd*.

```
void Bconout(fd, c)
    int fd;
    int c ;
```

7.3 Reading the MOUSE*: An example*

The following is an example that shows the way mouse device can be read. Here goes:

```
/* A sample program to read the mouse */
static int fd_mouse =      -1 ;
OpenMouse()
    {
    static char reset_str[2] =
        {
        0x80, /* reset ikbd, first  byte */
        0x01, /* reset ikbd, second byte */
        } ;
    static char init_str[5] =
        {
        0x08, /* set relative mouse position reporting */
        0x0b, /* set mouse threshold command           */
        0x01, /* x mouse threshold before gives report */
        0x01, /* y mouse threshold before gives report */
        0x11  /* enable the mouse                      */
        } ;
    if ((fd_mouse = open("/dev/mouse", O_RDWR|O_NDELAY)) < 0)
        {
        fprintf(stderr, "Unable to open '/dev/mouse'!0) ;
        exit(1) ;
        }
    Raw_On(fd_mouse)    ; /* put the mouse in the raw mode  */
    Echo_Off(fd_mouse) ; /* turn off echo for the mouse    */

    /*
    ** reset and enable the mouse
    */
    write(fd_mouse, &(reset_str[0]), 2) ;
    sleep(1) ;           /* delay for reset to complete    */
    /*
    ** set the mouse modes
    */
    write(fd_mouse, &( init_str[0]), 5) ;
    }

/* state = 0: looking for command byte */
/* state = 1: skipping rest of command */
/* state = 2: looking for x value      */
/* state = 3: looking for y value      */
CheckMouse(x,y)
    int *x, *y ;
    {
    register int c, len, cx, cy    ;
    static unsigned char mbuf[256] ;
    static short state = 0         ;
    static short cnt               ;
    static char lcount[] = { 7, 5, 2, 2, 2, 2, 6, 2, 1, 1 } ;
    static unsigned char *ptr      ;

    cx = 0 ;
    cy = 0 ;
    /*
    ** Read in large chunks since read system call is expensive ** if doing
    single character I/O.
    */
    while ((len = read(fd_mouse, &(mbuf[0]), 256)) > 0)
        {
        ptr = &(mbuf[0]) ;
        while (len--)
            {
            c = *ptr++ & 0xff ;
            if (state == 0)
                {
                /*
                ** 0xf8,0xf9,0xfa,0xfb are mouse reports, lower
                ** two bits are the two button states, will be
                ** followed by two bytes of x and y delta.
                */
                if ((c >= 0xf8) && (c <= 0xfb))
                    state = 2 ;
                else if (c < 0xf6)
                    continue  ;                 /* skip garbage */
                else
                    {
                    /*
                    ** Skip all but mouse stuff.
                    */
                    cnt = lcount[c - 0xf6] ;
                    state = 1 ;
                    }
                }
            else if (state == 1)
                {
                /*
                ** Skipping non mouse stuff.
                */
                if ((--cnt) == 0) state = 0 ;
                }
            else if (state == 2) /* looking for mouse x change */
                {
                if (c & 0x80) c -= 256 ;
                cx   += c ;
                state = 3 ;
                }
            else if (state == 3) /* looking for mouse y change */
                {
                if (c & 0x80) c -= 256 ;
                cy   += c ;
                state = 0 ;
                }
            }
        }
    /*
    ** set the return values, the accumulated change in position
    */
    *x = cx ;
    *y = cy ;
    }
```

The VDI calls *vsc_form*, *v_dspcur* , and *v_rmcur* can be used to manage a mouse cursor if needed.

8. *Support for AES Calls*

There is no support for AES (*trap #2*) calls. Their use will cause IDRIS to crash.

9. *Support for LINE-A Calls*

The following **Line-A** calls should be safe to use from within IDRIS. Again the same cautions and warnings given for XBIOS calls apply. The VDI calls are built on top of these routines. Line-A calls should use the *Blitter* chip when installed.

To use these functions, include the file *gembind.h* and link to the library file *libgem.o*. Both are in */usr/lib* directory.

Function	*Description*
`Ainit()`	Line-a initialization.
`Appixel()`	Put a pixel.
`Agpixel()`	Get a pixel.
`Aline()`	Draw a line.
`Ahline()`	Draw a horizontal line.
`Afrect()`	Draw a filled rectangle.
`Afpoly()`	Draw a filled polygon.
`Abitblt(control_adr)`	Do a-bitblit transfer.
`Atxtblt()`	Single character text block transfer.
`Ausprit(sav_adr)`	Undraw a sprite.
`Adsprit(x,y,def_adr,sav_adr)`	Draw a sprite.
`Acraste()`	Copy a raster form.
`Aseedfi()`	Do a seed fill.
`Aversio()`	Return version number saved by Ainit().

10. For More Information

For more detailed information than is presented here, the following sources are useful:

- The various IDRIS manuals provided with the *IDRIS 3.12* operating system software.
- The developers kit available from Atari.
- The *Sybex* **Programmers' Guide to Gem** by Phillip Balma and William Fitler
  SYBEX Inc.
  2344 Sixth Street
  Berkeley, Ca 94710
  ISBN 0-89588-297-3
- The *Abacus Software* **AtariST Internals** by K.Gerits, L Englisch, and R. Bruckmann. A *Data Becker Book* published by Abacus Software.
  ABACUS Software, Inc.
  P.O. BOX 7211
  Grand Rapids, Mi. 49510
  ISBN 0-916439-46-1

Abacus also has several other books on various aspects of the Atari ST.

SECTION VI

APPENDIX A

1. ST Console Driver

The *Atari ST** console can, when needed, emulate an standard intelligent terminal. The following control sequences are supported by the ST console screen driver to provide the usual features and functions of a terminal. Esc denotes the *ASCII* escape code (Decimal 27, Hex 0x1b or Octal 033). The ST console is generally a *VT52*† emulation, although in medium or high resolution mode it has 25 instead of 24 lines.

2. Output Escape Sequences

The following table gives a detailed description of character sequences, which when output to console, cause the corresponding action(s) to take place.

Table A: ST Output Escape Sequences

Escape Sequence	*Resulting Action*
`Esc A`	Non destructive cursor up. Does not cause scroll.
`Esc B`	Non destructive cursor down. Does not cause scroll.
`Esc C`	Non destructive cursor right.
`Esc D`	Non destructive cursor left.
`Esc E`	Clear screen and home cursor to top left corner of the display.
`Esc H`	Home cursor to top left corner of the display.
`Esc I`	Non destructive cursor up. Does cause reverse scroll.
`Esc J`	Erase from cursor position to end of page.
`Esc K`	Erase from cursor position to end of line.
`Esc L`	Insert a line. Cursor line and lines below it are moved down a line, cursor is moved to the first column on the inserted line.
`Esc M`	Delete a line. Lines below are moved up one line, cursor is set to the first column.
`Esc Y`	Position the cursor. This must be followed by two more bytes. The first is the row value, the second is the column value. An offset of 32 (or ASCII value of Space) should be added to both the row and the column value. Rows are in the range of 0 to 24, columns in the range 0 to 79.
`Esc b`	Set the character color. Requires one more byte of which the lower 4 bits specify the color map index to be used.
`Esc c`	Set the background color. Requires one more byte of which the lower 4 bits specify the color map index to be used.
`Esc d`	Erase from beginning of the display up to, and including the cursor position.
`Esc e`	Turn the cursor on.
`Esc f`	Turn the cursor off.
`Esc j`	Save cursor position. The current cursor position is memorized for later use by Esc k sequence.

* Trademark of ATARI CORP.

† Trademark of Digital Equipment Corp.

Table A: ST Output Escape Sequences

Escape Sequence	*Resulting Action*
Esc k	Restore cursor position. This will restore the cursor to the position saved by issuing Esc j sequence.
Esc l	Erase the entire line. Clears the line and moves the cursor to the first column.
Esc o	Erase from the beginning of line up to, and including the cursor position.
Esc p	Enter the reverse video character display mode.
Esc q	Exit reverse video character display mode.
Esc v	Turn on end-of-line wrapping.
Esc w	Turn off end-of-line wrapping.

3. Input Escape Sequences

The following table identifies the escape sequences which are produced by the keyboard special keys. In interactive programs, when a user presses one of these special keys the corresponding escape sequence is returned to the process.

Table B: ST Input Escape Sequences

Special Key	*Escape Sequence*
Clr Home	Esc E
Cursor Up	Esc A
Cursor Down	Esc B
Cursor Right	Esc C
Cursor Left	Esc D
Insert	Esc I
Undo	Esc U
Help	Esc H
f1	Esc : ;
f2	Esc : <
f3	Esc : =
f4	Esc : >
f5	Esc : ?
f6	Esc : @
f7	Esc : A
f8	Esc : B
f9	Esc : C
f10	Esc : D
F1	Esc : T
F2	Esc : U
F3	Esc : V
F4	Esc : W
F5	Esc : X
F6	Esc : Y
F7	Esc : Z
F8	Esc : [
F9	Esc : \
F10	Esc :]

Function keys are refered as f1 through f10, while shifted function keys are refered to as F1 through F10.

SECTION VII

APPENDIX B

1. Building A Print Spooler

In order to create a spooling system for the line printer driver *lpr* the following steps should be taken:

1. Login as *root* or change to super user by using the *su* command.
2. Create the spool directory for the lpr:

   ```
   # mkdir /usr/spool
   # mkdir /usr/spool/lpr
   # chmod -0755 /usr/spool /usr/spool/lpr
   ```

3. Move the *lpr* program to *lpr.spool*:

   ```
   # mv /usr/lpr /bin/lpr.spool
   ```

4. Create a new lpr program (shell script) by using the following command:

   ```
   # echo "/bin/enque -dir /usr/spool/lpr -" > /bin/lpr
   ```

5. Make the new lpr shell script executable:

   ```
   # chmod -0755 /bin/lpr
   ```

6. Using your favorite editor (i.e. emacs, e, vi) add the following lines to the end of the */adm/init* file:

   ```
   # emacs /adm/init
   ```

   and add:

   ```
   w /bin/rm /usr/spool/lpr/deque.lock
   M /bin/deque -nw30 -dir /usr/spool/lpr -c "/bin/sh -c '/bin/lpr.spool < %f'"
   ```

7. Reboot the system, and proceed to multi-user, in order for the changes to take effect.

Use *lpr* as usual, command like `lpr < {filename}` and `pr {filename} | lpr` will cause the file to queue into the spool directory */usr/spool/lpr*, and print out correctly.

STX™ WINDOWS

COMPUTER TOOLS INTERNATIONAL

NAME

Install-STX - installing STX-Windows V10.4 on ATARI ST running IDRIS OS

SYNOPSIS

```
mount -r /dev/fd0 /x
cp -rd /x /
mount -u /dev/fd0
```

FUNCTION

These pages describe the installation procedure for installing the STX-Windows version 10.4 on the ATARI-ST/MEGA under IDRIS.

1. For every floppy in the distribution set, execute the following:

   ```
   mount -r /dev/fd0 /x
   cp -rd /x /
   mount -u /dev/fd0
   ```

2. Synchronize the disk and reset the computer. At the cboot prompt enter xidris.314 and press return. This will cause the system to boot with the new kernel.

3. Make the required devices by executing the shell script in root directory called INSTALL.X10.4. This is a listing of this script:

   ```
   :
   : Build special files

   echo "Building special files ..."

   cd /dev

   : character special files
   mkdev -i -c0 < /dev/cnames
   mkdir select socket
   mv sel[0-9][0-9] select
   mv sock[0-9][0-9] socket
   mkdev -cl -u67 /dev/ikbdx

   : block special files
   mkdev -i -b0 < /dev/bnames

   : fix ps
   echo ""
   echo "  Linking /bin/ps to /bin/ps.314"
   echo "  The ps  program for IDRIS 3.12"
   echo "  will be saved  in  /bin/ps.312"
   sync

   ln /bin/ps /bin/ps.312
   rm /bin/ps
   ln /bin/ps.314 /bin/ps

   sync
   ```

```
echo "Done."
```

The program ps is sensitive to changes in the kernel structures which it investigates to find information about processes. This is the reason for the different versions of the ps program.

4. Extend your exectution path to include /usr/X/bin directory. Set up the startup files:

   ```
   cp /usr/X/.Xrc       $H
   cp /usr/X/.Xdefaults $H
   cp /usr/X/.uwmrc     $H
   ```

   These are the files which can be customized by the user.

5. Once you are certain the system is operating correctly, you should rename the new kernel to "idriskla". Be sure to save a copy of the old kernel by renaming it to **idris.old** or some thing similar until you are sure the kernel is working correctly.

6. Use **xstart** to start the STX-Windows. All users should have their .login setup to include /usr/local/bin and /usr/X/bin in their execution search path.

7. Press in sequence the **CONTROL**, **ALTERNATE**, and **ESCAPE** keys to exit STX-Windows. The above order is not important but, all keys must be held down together to cause an exit from the STX-Windows.

NOTES

/usr/X/bin/xterm should be of mode set-user-id and set-group-id in order to be used by logins other than root. In order to do this, you will need to execute the following:

```
chmod -06755 /usr/X/bin/xterm
```

Directories Effected

The following directories are touched or created.

```
/
/bin
/usr/X
/usr/lib
/usr/bin
```

Misc ...

The following table shows the minimum "heap/stack" sizes for the some programs in the STX-Window system.

PROGRAM	"setb" SIZE
Xinit:	4096
Xserver:	98304
bitmap:	24576
gedit:	24576
nxclock:	24576
resize:	0
uwm:	32768
xclock:	8192
xfd:	24576
xnwm:	24576
xrefresh:	24576
xsetroot:	24576
xshell:	16384
xterm:	24576
xwininfo:	24576
xwm:	24576

Contents of Disk 1

```
drwxrwxrwx  3 root          96 May 03 23:27 /x

/x
drwxrwxrwx  3 root          96 May 03 23:27 .
drwxr-xr-x 10 root        1328 May 03 23:35 ..
-rwxrwxrwx  1 root         223 May 03 23:27 MAKEBIG
-rwxrwxrwx  1 root         222 May 03 23:27 MAKESMALL
drwxrwxrwx  5 root          80 May 03 23:27 usr
-rwxr-xr-x  1 root      131940 May 03 23:26 xidris.314

/x/usr
drwxrwxrwx  5 root          80 May 03 23:27 .
drwxrwxrwx  3 root          96 May 03 23:27 ..
drwxrwxrwx  6 root         144 May 03 23:37 X
drwxrwxrwx  2 root         144 May 03 23:22 bin
drwxrwxrwx  2 root          96 May 03 23:24 lib

/x/usr/X
drwxrwxrwx  6 root         144 May 03 23:37 .
drwxrwxrwx  5 root          80 May 03 23:27 ..
-rw-rw-rw-  1 root        1575 May 03 23:28 .Xdefaults
-rwxrwxrwx  1 root         259 May 03 23:28 .Xrc
-rw-rw-rw-  1 root        2308 May 03 23:29 .uwmrc
drwxrwxrwx  2 root         176 May 03 23:40 bin
drwxrwxrwx  2 root         368 May 03 23:34 bitmaps
drwxrwxrwx  2 root         256 May 03 23:32 include
drwxrwxrwx  2 root         128 May 03 23:31 lib

/x/usr/X/bin
drwxrwxrwx  2 root         176 May 03 23:40 .
drwxrwxrwx  6 root         144 May 03 23:37 ..
-rw-r--r--  1 root        1575 May 03 23:37 .Xdefaults
-rw-rw-rw-  1 root           0 May 03 23:37 .Xlog
```

```
-rwxr-xr-x  1 root           226 May 03 23:37 .Xrc
-rw-r--r--  1 root          2295 May 03 23:37 .uwmrc
-rwxr-xr-x  1 root          5135 May 03 23:37 Xinit
-rwxr-xr-x  1 root        107350 May 03 23:38 Xserver
-rwxr-xr-x  1 root         59919 May 03 23:39 bitmap
-rwxr-xr-x  1 root         79648 May 03 23:40 gedit

/x/usr/X/bitmaps
drwxrwxrwx  2 root           368 May 03 23:34 .
drwxrwxrwx  6 root           144 May 03 23:37 ..
-rw-rw-rw-  1 root           212 May 03 23:33 1x1.bitmap
-rw-rw-rw-  1 root           212 May 03 23:33 2x2.bitmap
-rw-rw-rw-  1 root           218 May 03 23:33 boxes.bitmap
-rw-rw-rw-  1 root           221 May 03 23:33 carpet.bitmap
-rw-rw-rw-  1 root           221 May 03 23:33 carpet1.bitmap
-rw-rw-rw-  1 root           236 May 03 23:33 cross_weave.bi
-rw-rw-rw-  1 root           224 May 03 23:33 dimple1.bitmap
-rw-rw-rw-  1 root           224 May 03 23:33 dimple3.bitmap
-rw-rw-rw-  1 root           212 May 03 23:34 dot.bitmap
-rw-rw-rw-  1 root           215 May 03 23:34 floor.bitmap
-rw-rw-rw-  1 root           215 May 03 23:34 floor1.bitmap
-rw-rw-rw-  1 root           218 May 03 23:34 gray1.bitmap
-rw-rw-rw-  1 root           218 May 03 23:34 gray3.bitmap
-rw-rw-rw-  1 root           215 May 03 23:34 icon.bitmap
-rw-rw-rw-  1 root           233 May 03 23:34 root_weave.bit
-rw-rw-rw-  1 root           236 May 03 23:34 root_weave1.bi
-rw-rw-rw-  1 root           221 May 03 23:34 scales.bitmap
-rw-rw-rw-  1 root           221 May 03 23:34 target.bitmap
-rw-rw-rw-  1 root           215 May 03 23:34 wave1.bitmap
-rw-rw-rw-  1 root           215 May 03 23:34 wave2.bitmap
-rw-rw-rw-  1 root           233 May 03 23:34 wide_weave.bit

/x/usr/X/include
drwxrwxrwx  2 root           256 May 03 23:32 .
drwxrwxrwx  6 root           144 May 03 23:37 ..
-rw-rw-rw-  1 root          7944 May 03 23:32 X.h
-rw-rw-rw-  1 root          8534 May 03 23:32 XMenu.h
-rw-rw-rw-  1 root          2772 May 03 23:32 Xdev.h
-rw-rw-rw-  1 root          6671 May 03 23:32 Xint.h
-rw-rw-rw-  1 root          2194 May 03 23:32 Xkeyboard.h
-rw-rw-rw-  1 root          1302 May 03 23:32 Xkeymap.h
-rw-rw-rw-  1 root         11953 May 03 23:32 Xlib.h
-rw-rw-rw-  1 root          2059 May 03 23:32 Xlibinternal.h
-rw-rw-rw-  1 root          3286 May 03 23:32 Xproto.h
-rw-rw-rw-  1 root          1490 May 03 23:32 Xtext.h
-rw-rw-rw-  1 root           312 May 03 23:32 Xtty.h
-rw-rw-rw-  1 root           814 May 03 23:32 mit-copyright.
-rw-rw-rw-  1 root           341 May 03 23:32 rgb.h
-rw-rw-rw-  1 root          1847 May 03 23:33 vsinput.h

/x/usr/X/lib
drwxrwxrwx  2 root           128 May 03 23:31 .
drwxrwxrwx  6 root           144 May 03 23:37 ..
-rw-rw-rw-  1 root             0 May 03 23:30 X0msgs
```

```
-rw-rw-rw-  1 root         17926 May 03 23:30 XMenulib.o
-rw-rw-rw-  1 root         70531 May 03 23:31 Xlib.o
-rw-rw-rw-  1 root          4096 May 03 23:31 rgb.dir
-rw-rw-rw-  1 root          4096 May 03 23:31 rgb.pag
-rw-rw-rw-  1 root          2514 May 03 23:31 termcap

/x/usr/bin
drwxrwxrwx  2 root           144 May 03 23:22 .
drwxrwxrwx  5 root            80 May 03 23:27 ..
-rwxrwxrwx  1 root           129 May 03 23:19 SetClock
-rwxrwxrwx  1 root          5798 May 03 23:19 layers
-rwxrwxrwx  1 root         19618 May 03 23:20 options
-rwxrwxrwx  1 root         14889 May 03 23:20 readclock
-rwxr-xr-x  1 root          7018 May 03 23:21 show0
-rwxr-xr-x  1 root         19597 May 03 23:22 tosdir
-rwxrwxrwx  1 root           217 May 03 23:22 xstart

/x/usr/lib
drwxrwxrwx  2 root            96 May 03 23:24 .
drwxrwxrwx  5 root            80 May 03 23:27 ..
-rw-rw-rw-  1 root          1375 May 03 23:23 dbm.h
-rw-rw-rw-  1 root         18159 May 03 23:23 gembind.h
-rw-rw-rw-  1 root         35553 May 03 23:24 libgem.o
-rw-rw-rw-  1 root          5311 May 03 23:24 socket.h
```

Contents of Disk 2

```
drwxrwxrwx  3 root            48 May 03 23:47 /x

/x
drwxrwxrwx  3 root            48 May 03 23:47 .
drwxr-xr-x 10 root          1328 May 03 23:35 ..
drwxrwxrwx  3 root            48 May 03 23:47 usr

/x/usr
drwxrwxrwx  3 root            48 May 03 23:47 .
drwxrwxrwx  3 root            48 May 03 23:47 ..
drwxrwxrwx  4 root            64 May 03 23:54 X

/x/usr/X
drwxrwxrwx  4 root            64 May 03 23:54 .
drwxrwxrwx  3 root            48 May 03 23:47 ..
drwxrwxrwx  2 root           240 May 03 23:52 bin
drwxrwxrwx  2 root           512 May 03 23:57 font

/x/usr/X/bin
drwxrwxrwx  2 root           240 May 03 23:52 .
drwxrwxrwx  4 root            64 May 03 23:54 ..
-rwxr-xr-x  1 root         21904 May 03 23:48 resize
-rw-r--r--  1 root           106 May 03 23:48 rs
-rwxr-xr-x  1 root         91221 May 03 23:49 uwm
-rwxr-xr-x  1 root         43230 May 03 23:49 xclock
-rwxr-xr-x  1 root         35709 May 03 23:50 xfd
```

```
-rwxr-xr-x  1 root        54822 May 03 23:50 xnwm
-rwxr-xr-x  1 root        20123 May 03 23:50 xrefresh
-rwxr-xr-x  1 root        33662 May 03 23:50 xsetroot
-rwxr-xr-x  1 root        38634 May 03 23:51 xshell
-rwxr-xr-x  1 root          217 May 03 23:51 xstart
-rwsr-sr-x  1 root       125004 May 03 23:52 xterm
-rwxr-xr-x  1 root        29810 May 03 23:52 xwininfo
-rwxr-xr-x  1 root        48649 May 03 23:53 xwm

/x/usr/X/font
drwxrwxrwx  2 root          512 May 03 23:57 .
drwxrwxrwx  4 root           64 May 03 23:54 ..
-rw-rw-rw-  1 root         1242 May 03 23:54 6x10.onx
-rw-rw-rw-  1 root         1530 May 03 23:54 6x13.onx
-rw-rw-rw-  1 root         1528 May 03 23:54 6x13p.onx
-rw-rw-rw-  1 root         1946 May 03 23:54 8x13.onx
-rw-rw-rw-  1 root         2442 May 03 23:54 9x15.onx
-rw-rw-rw-  1 root         4018 May 03 23:55 accord.onx
-rw-rw-rw-  1 root         1698 May 03 23:55 accord18.onx
-rw-rw-rw-  1 root         4202 May 03 23:55 accordb.onx
-rw-rw-rw-  1 root         5842 May 03 23:55 accordbfx.onx
-rw-rw-rw-  1 root         4402 May 03 23:55 accordbssx.onx
-rw-rw-rw-  1 root         4882 May 03 23:55 accordbsx.onx
-rw-rw-rw-  1 root         5842 May 03 23:55 accordfx.onx
-rw-rw-rw-  1 root         4402 May 03 23:55 accordssx.onx
-rw-rw-rw-  1 root         4882 May 03 23:55 accordsx.onx
-rw-rw-rw-  1 root         8218 May 03 23:55 b28botb.onx
-rw-rw-rw-  1 root         3746 May 03 23:55 cmi5.onx
-rw-rw-rw-  1 root         6642 May 03 23:55 cmi5fx.onx
-rw-rw-rw-  1 root         4338 May 03 23:55 cmi5ssx.onx
-rw-rw-rw-  1 root         5202 May 03 23:56 cmi5sx.onx
-rw-rw-rw-  1 root         5986 May 03 23:56 cmr7.onx
-rw-rw-rw-  1 root        11506 May 03 23:56 cmr7fx.onx
-rw-rw-rw-  1 root         6306 May 03 23:56 cmr7ssx.onx
-rw-rw-rw-  1 root         8306 May 03 23:56 cmr7sx.onx
-rw-rw-rw-  1 root         7586 May 03 23:56 cmsy7.onx
-rw-rw-rw-  1 root         6690 May 03 23:56 cnt57.onx
-rw-rw-rw-  1 root         3706 May 03 23:56 cor.onx
-rw-rw-rw-  1 root         3706 May 03 23:56 cor20.onx
-rw-rw-rw-  1 root         5170 May 03 23:56 cor20fx.onx
-rw-rw-rw-  1 root         5170 May 03 23:57 corfx.onx
```

Contents of Disk 3

```
drwxrwxrwx  3 root           48 May 04 00:09 /x

/x
drwxrwxrwx  3 root           48 May 04 00:09 .
drwxr-xr-x 10 root         1328 May 03 23:35 ..
drwxrwxrwx  3 root           48 May 04 00:09 usr

/x/usr
drwxrwxrwx  3 root           48 May 04 00:09 .
drwxrwxrwx  3 root        48 May 04 00:09 ..
drwxrwxrwx  3 root        48 May 04 00:09 X

/x/usr/X
drwxrwxrwx  3 root        48 May 04 00:09 .
drwxrwxrwx  3 root        48 May 04 00:09 ..
drwxrwxrwx  2 root      1536 May 04 00:21 font

/x/usr/X/font
drwxrwxrwx  2 root      1536 May 04 00:21 .
drwxrwxrwx  3 root        48 May 04 00:09 ..
-rw-rw-rw-  1 root      5746 May 04 00:09 ctl25.onx
-rw-rw-rw-  1 root     10706 May 04 00:10 ctl25fx.onx
-rw-rw-rw-  1 root      6130 May 04 00:10 ctl25ssx.onx
-rw-rw-rw-  1 root      7378 May 04 00:10 ctl25sx.onx
-rw-rw-rw-  1 root      2034 May 04 00:10 ctrlbar.onx
-rw-rw-rw-  1 root      9338 May 04 00:11 cyr30.onx
-rw-rw-rw-  1 root     13826 May 04 00:12 datapad.onx
-rw-rw-rw-  1 root      6738 May 04 00:12 delgat.onx
-rw-rw-rw-  1 root      7106 May 04 00:12 delgatfx.onx
-rw-rw-rw-  1 root       770 May 04 00:12 dolby.onx
-rw-rw-rw-  1 root      2754 May 04 00:12 dotxl.onx
-rw-rw-rw-  1 root      3634 May 04 00:12 dotxlfx.onx
-rw-rw-rw-  1 root      1826 May 04 00:12 doty4.onx
-rw-rw-rw-  1 root      4146 May 04 00:12 doty4fx.onx
-rw-rw-rw-  1 root      2866 May 04 00:12 doty4ssx.onx
-rw-rw-rw-  1 root      3122 May 04 00:12 doty4sx.onx
-rw-rw-rw-  1 root      1826 May 04 00:12 doty5.onx
-rw-rw-rw-  1 root      4146 May 04 00:12 doty5fx.onx
-rw-rw-rw-  1 root      2866 May 04 00:12 doty5ssx.onx
-rw-rw-rw-  1 root      3122 May 04 00:13 doty5sx.onx
-rw-rw-rw-  1 root       730 May 04 00:13 empire.onx
-rw-rw-rw-  1 root      4810 May 04 00:13 esch40.onx
-rw-rw-rw-  1 root     24290 May 04 00:13 esch80.onx
-rw-rw-rw-  1 root      2282 May 04 00:13 helvl0.onx
-rw-rw-rw-  1 root      1754 May 04 00:13 helvl0b.onx
-rw-rw-rw-  1 root      3188 May 04 00:13 helvl0bfx.onx
-rw-rw-rw-  1 root      1940 May 04 00:13 helvl0bssx.onx
-rw-rw-rw-  1 root      2356 May 04 00:13 helvl0bsx.onx
-rw-rw-rw-  1 root      3632 May 04 00:13 helvl0fx.onx
-rw-rw-rw-  1 root      2192 May 04 00:13 helvl0ssx.onx
-rw-rw-rw-  1 root      2672 May 04 00:13 helvl0sx.onx
-rw-rw-rw-  1 root      2042 May 04 00:13 helvl2.onx
-rw-rw-rw-  1 root      2074 May 04 00:13 helvl2b.onx
-rw-rw-rw-  1 root      3410 May 04 00:13 helvl2bfx.onx
-rw-rw-rw-  1 root      2290 May 04 00:13 helvl2bssx.onx
-rw-rw-rw-  1 root      2738 May 04 00:13 helvl2bsx.onx
-rw-rw-rw-  1 root      3410 May 04 00:14 helvl2fx.onx
-rw-rw-rw-  1 root      2122 May 04 00:14 helvl2i.onx
-rw-rw-rw-  1 root      3410 May 04 00:14 helvl2ifx.onx
-rw-rw-rw-  1 root      2290 May 04 00:14 helvl2issx.onx
-rw-rw-rw-  1 root      2738 May 04 00:14 helvl2isx.onx
-rw-rw-rw-  1 root      2290 May 04 00:14 helvl2ssx.onx
-rw-rw-rw-  1 root      2738 May 04 00:14 helvl2sx.onx
```

```
-rw-rw-rw-  1 root        826 May 04 00:14 hisym.onx
-rw-rw-rw-  1 root       1178 May 04 00:14 hisymbol.onx
-rw-rw-rw-  1 root       3586 May 04 00:14 host.onx
-rw-rw-rw-  1 root       7746 May 04 00:14 ice34.onx
-rw-rw-rw-  1 root       1842 May 04 00:14 icons.onx
-rw-rw-rw-  1 root       8218 May 04 00:14 k28bot.onx
-rw-rw-rw-  1 root       8218 May 04 00:14 k28botb.onx
-rw-rw-rw-  1 root       8218 May 04 00:14 k28botbu.onx
-rw-rw-rw-  1 root       8218 May 04 00:15 k28botu.onx
-rw-rw-rw-  1 root       8218 May 04 00:15 k28top.onx
-rw-rw-rw-  1 root       8218 May 04 00:15 k28topb.onx
-rw-rw-rw-  1 root       8218 May 04 00:15 k28topu.onx
-rw-rw-rw-  1 root       7490 May 04 00:15 kata30.onx
-rw-rw-rw-  1 root       9794 May 04 00:15 kbsym.onx
-rw-rw-rw-  1 root       4378 May 04 00:15 kilter.onx
-rw-rw-rw-  1 root       3002 May 04 00:15 kilter10.onx
-rw-rw-rw-  1 root      15898 May 04 00:15 kilter28.onx
-rw-rw-rw-  1 root      15898 May 04 00:16 kilter28b.onx
-rw-rw-rw-  1 root      15898 May 04 00:16 kilter28u.onx
-rw-rw-rw-  1 root       4378 May 04 00:16 kilterb.onx
-rw-rw-rw-  1 root       3450 May 04 00:16 kiltercrn.onx
-rw-rw-rw-  1 root       8218 May 04 00:16 kilterd.onx
-rw-rw-rw-  1 root       8218 May 04 00:16 kilterdb.onx
-rw-rw-rw-  1 root      15898 May 04 00:16 kiltr28bu.onx
-rw-rw-rw-  1 root        146 May 04 00:16 leds.onx
-rw-rw-rw-  1 root       4978 May 04 00:16 math5.onx
-rw-rw-rw-  1 root       6290 May 04 00:17 micr25.onx
-rw-rw-rw-  1 root      11506 May 04 00:17 micr25fx.onx
-rw-rw-rw-  1 root       8306 May 04 00:17 micr25sx.onx
-rw-rw-rw-  1 root      13314 May 04 00:17 monitorbk.onx
-rw-rw-rw-  1 root       9218 May 04 00:17 monitorfr.onx
-rw-rw-rw-  1 root       6658 May 04 00:17 mouse.onx
-rw-rw-rw-  1 root        602 May 04 00:17 nil2.onx
-rw-rw-rw-  1 root       3122 May 04 00:17 noecho.onx
-rw-rw-rw-  1 root       2010 May 04 00:17 noecho14.onx
-rw-rw-rw-  1 root      13314 May 04 00:17 nonb44.onx
-rw-rw-rw-  1 root      30578 May 04 00:18 nonb44fx.onx
-rw-rw-rw-  1 root      18610 May 04 00:18 nonb44ssx.onx
-rw-rw-rw-  1 root      21426 May 04 00:18 nonb44sx.onx
-rw-rw-rw-  1 root      13138 May 04 00:18 noni44.onx
-rw-rw-rw-  1 root      30578 May 04 00:19 noni44fx.onx
-rw-rw-rw-  1 root      18610 May 04 00:19 noni44ssx.onx
-rw-rw-rw-  1 root      21426 May 04 00:19 noni44sx.onx
-rw-rw-rw-  1 root       5286 May 04 00:20 oldeng.onx
-rw-rw-rw-  1 root      12786 May 04 00:20 oldengfx.onx
-rw-rw-rw-  1 root       6546 May 04 00:21 oldengssx.onx
-rw-rw-rw-  1 root       9906 May 04 00:21 oldengsx.onx
-rw-rw-rw-  1 root        682 May 04 00:21 pointers.onx
-rw-rw-rw-  1 root       4650 May 04 00:21 rogue-b.onx
-rw-rw-rw-  1 root       4650 May 04 00:21 rogue-n.onx
-rw-rw-rw-  1 root       8802 May 04 00:21 rune32.onx
```

Contents of Disk 4

```
drwxrwxrwx  4 root        8Ø May 11 21:22 /x

/x
drwxrwxrwx  4 root        8Ø May 11 21:22 .
drwxr-xr-x 1Ø root      1328 May Ø5 23:2Ø ..
-rwxrwxrwx  1 root       49Ø May 11 21:34 INSTALL.X1Ø.4
drwxrwxrwx  2 root        48 May 11 21:23 bin
drwxrwxrwx  3 root        48 May Ø4 ØØ:24 usr

/x/bin
drwxrwxrwx  2 root        48 May 11 21:23 .
drwxrwxrwx  4 root        8Ø May 11 21:22 ..
-rwxr-xr-x  1 root     28213 May 11 21:23 ps.314

/x/usr
drwxrwxrwx  3 root        48 May Ø4 ØØ:24 .
drwxrwxrwx  4 root        8Ø May 11 21:22 ..
drwxrwxrwx  3 root        48 May Ø4 ØØ:24 X

/x/usr/X
drwxrwxrwx  3 root        48 May Ø4 ØØ:24 .
drwxrwxrwx  3 root        48 May Ø4 ØØ:24 ..
drwxrwxrwx  2 root       864 May Ø4 ØØ:32 font

/x/usr/X/font
drwxrwxrwx  2 root       864 May Ø4 ØØ:32 .
drwxrwxrwx  3 root        48 May Ø4 ØØ:24 ..
-rw-rw-rw-  1 root     15578 May Ø4 ØØ:25 sbdi4Ø.onx
-rw-rw-rw-  1 root     34418 May Ø4 ØØ:25 sbdi4Øfx.onx
-rw-rw-rw-  1 root     21938 May Ø4 ØØ:25 sbdi4Øssx.onx
-rw-rw-rw-  1 root     2693Ø May Ø4 ØØ:26 sbdi4Øsx.onx
-rw-rw-rw-  1 root     1433Ø May Ø4 ØØ:26 sbdr4Ø.onx
-rw-rw-rw-  1 root     34418 May Ø4 ØØ:26 sbdr4Øfx.onx
-rw-rw-rw-  1 root     21938 May Ø4 ØØ:26 sbdr4Øssx.onx
-rw-rw-rw-  1 root     2693Ø May Ø4 ØØ:26 sbdr4Øsx.onx
-rw-rw-rw-  1 root     31258 May Ø4 ØØ:27 sbdr66.onx
-rw-rw-rw-  1 root     812Ø2 May Ø4 ØØ:28 sbdr66fx.onx
-rw-rw-rw-  1 root     44546 May Ø4 ØØ:28 sbdr66ssx.onx
-rw-rw-rw-  1 root     635Ø6 May Ø4 ØØ:29 sbdr66sx.onx
-rwxrwxrwx  1 root        34 May Ø4 ØØ:29 showfont
-rw-rw-rw-  1 root     16386 May Ø4 ØØ:29 symbol.onx
-rw-rw-rw-  1 root      2Ø34 May Ø4 ØØ:29 tekctrlbr.onx
-rw-rw-rw-  1 root      2Ø34 May Ø4 ØØ:29 teklwrbar.onx
-rw-rw-rw-  1 root      1658 May Ø4 ØØ:29 timroml Ø.onx
-rw-rw-rw-  1 root      2Ø42 May Ø4 ØØ:29 timroml Øb.onx
-rw-rw-rw-  1 root      2772 May Ø4 ØØ:29 timroml Øbfx.on
-rw-rw-rw-  1 root      194Ø May Ø4 ØØ:29 timroml Øbssx.o
-rw-rw-rw-  1 root      2148 May Ø4 ØØ:29 timroml Øbsx.on
-rw-rw-rw-  1 root      2772 May Ø4 ØØ:3Ø timroml Øfx.onx
-rw-rw-rw-  1 root      1754 May Ø4 ØØ:3Ø timroml Øi.onx
-rw-rw-rw-  1 root      2772 May Ø4 ØØ:3Ø timroml Øifx.on
-rw-rw-rw-  1 root      194Ø May Ø4 ØØ:3Ø timroml Øissx.o
```

```
-rw-rw-rw-   1 root       2148 May 04 ØØ:3Ø timromlØisx.on
-rw-rw-rw-   1 root       194Ø May 04 ØØ:3Ø timromlØssx.on
-rw-rw-rw-   1 root       2148 May 04 ØØ:3Ø timromlØsx.onx
-rw-rw-rw-   1 root       2Ø74 May 04 ØØ:3Ø timroml2.onx
-rw-rw-rw-   1 root       241Ø May 04 ØØ:3Ø timroml2b.onx
-rw-rw-rw-   1 root       341Ø May 04 ØØ:3Ø timroml2bfx.on
-rw-rw-rw-   1 root       2Ø66 May 04 ØØ:3Ø timroml2bssx.o
-rw-rw-rw-   1 root       2738 May 04 ØØ:3Ø timroml2bsx.on
-rw-rw-rw-   1 root       341Ø May 04 ØØ:3Ø timroml2fx.onx
-rw-rw-rw-   1 root       2282 May 04 ØØ:3Ø timroml2i.onx
-rw-rw-rw-   1 root       341Ø May 04 ØØ:3Ø timroml2ifx.on
-rw-rw-rw-   1 root       2Ø66 May 04 ØØ:31 timroml2issx.o
-rw-rw-rw-   1 root       2738 May 04 ØØ:31 timroml2isx.on
-rw-rw-rw-   1 root       2Ø66 May 04 ØØ:31 timroml2ssx.on
-rw-rw-rw-   1 root       2738 May 04 ØØ:31 timroml2sx.onx
-rw-rw-rw-   1 root      19186 May 04 ØØ:31 tng75.onx
-rw-rw-rw-   1 root       1658 May Ø4 ØØ:31 tromanlØ.onx
-rw-rw-rw-   1 root      14338 May 04 ØØ:31 vsl ØØ.onx
-rw-rw-rw-   1 root       4Ø18 May 04 ØØ:31 vstadoc.onx
-rw-rw-rw-   1 root       5842 May 04 ØØ:31 vstadocfx.onx
-rw-rw-rw-   1 root       44Ø2 May 04 ØØ:31 vstadocssx.onx
-rw-rw-rw-   1 root       4882 May 04 ØØ:32 vstadocsx.onx
-rw-rw-rw-   1 root       1946 May 04 ØØ:32 vtbold.onx
-rw-rw-rw-   1 root       361Ø May 04 ØØ:32 vtdhbot.onx
-rw-rw-rw-   1 root       361Ø May 04 ØØ:32 vtdhtop.onx
-rw-rw-rw-   1 root       361Ø May 04 ØØ:32 vtdwidth.onx
-rw-rw-rw-   1 root       1946 May 04 ØØ:32 vtsingle.onx
```