

Xoper

COLLABORATORS

	<i>TITLE :</i> Xoper		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		February 24, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Xoper	1
1.1	Xoper Documentation - Contents	1
1.2	Xoper Documentation - Introduction	1
1.3	Xoper Documentation - Requirements	2
1.4	Xoper Documentation - Installation	2
1.5	Xoper Documentation - Usage	2
1.6	Xoper Documentation - Keys	3
1.7	Xoper Documentation - Commands	5
1.8	Xoper Documentation - Viewing Exec lists	5
1.9	Xoper Documentation - Viewing other system lists	8
1.10	Xoper Documentation - Commands with parameter	11
1.11	Xoper Documentation - Commands without parameter	16
1.12	Xoper Documentation - Toggles and Options	17
1.13	Xoper Documentation - Dangerous Commands	20
1.14	Xoper Documentation - Task Types	22
1.15	Xoper Documentation - Task States	22
1.16	Xoper Documentation - Task Names	23
1.17	Xoper Documentation - CLI Infos	23
1.18	Xoper Documentation - Base Address	23
1.19	Xoper Documentation - Open Count	24
1.20	Xoper Documentation - Version and Revision	24
1.21	Xoper Documentation - Flags	24
1.22	Xoper Documentation - Name	24
1.23	Xoper Documentation - The Amiga Guru Book	24
1.24	Xoper Documentation - Bugs and Limitations	25
1.25	Xoper Documentation - Technical Info	26
1.26	Xoper Documentation - History	27
1.27	Xoper Documentation - Acknowledgments	30
1.28	Xoper Documentation - Disclaimer	31
1.29	Xoper Documentation - Copyright	31
1.30	Xoper Documentation - Author	31

Chapter 1

Xoper

1.1 Xoper Documentation - Contents

Xoper 2.5

A powerful system monitor
Copyright © 1988/95 by Werner Günther and Gunther Nikl
All Rights Reserved.

Introduction	What is Xoper?
Requirements	System requirements
Installation	Getting started
Usage	How to use Xoper
Some Basics	Keyboard Control
Command Overview	Description of all commands
Bugs/Limitations	Existing problems
Technical Info	How it was done
History	What is new?
Acknowledgments	Whom I want to thank.
Disclaimer	No Warrantries
Copyright	About legal issues.
Author	Who did it?

1.2 Xoper Documentation - Introduction

Introduction

Xoper is a freeware program to display and to control system activity.
Take a deep look inside your amiga and investigate what yor amiga is
actually is doing. Xoper gives you interesting system information and
can display various system lists.

1.3 Xoper Documentation - Requirements

Requirements

Xoper has no special requirements to run. To use Xoper an Amiga with at least 512KB RAM and Kickstart 1.2 (V33) is sufficient. It works with any amiga system upto Kickstart 3.1 (V40). No problems should arise with 68020, 68030, 68040 and 68060 (?) processors.

1.4 Xoper Documentation - Installation

Installation

Installing Xoper is fairly easy. Copy the program including the info file to a location on your harddisk. If you want you can then add this location to the global search path using the 'path' command, but this is not required. Thats all :-)

1.5 Xoper Documentation - Usage

Usage

Xoper can be started either from CLI/Shell or Workbench. To start Xoper from the Workbench doubleclick on its icon. To start Xoper from the shell simply type:

```
1> Xoper
```

followed by a return. This should start Xoper provided that it could be located somewhere in your search path. There is no need to 'run' the program because Xoper 'detaches' itself from the shell.

Xoper can be started with several options. If you run it from shell with a questionmark (?) as argument you'll get the argument template:

```
"CMD/K,NOSCRIPT/S,CX_POPUP/K,CX_POPKEY/K,CX_PRIORITY/N/K"
```

- CMD
 - used to specify which system list Xoper shall show by default
 - or
 - used to remove Xoper from memory -> CMD=kill
 - NOSCRIPT
 - Specifying this option disables the execution of Xopers startup script
 - CX_POPUP
 - This switch decides whether to open Xopers window at startup or not by specifying ON or OFF. It replaces the '-b' switch prior to v2.4.
-

The default is to open Xopers main window. Please note, this argument will be used even with Kickstart 1.2/1.3!

- CX_POPKEY

This specifies which hotkey Xoper shall use. The format of this string has to be a valid commodity string description. Its necessary to quote the description, eg.

```
CX_POPKEY="lalt lshift numpad *"
```

This argument affects only the *commodity* hotkey not the inputhandler hotkey. That one is hard-coded into Xoper and its always "Amiga-Amiga-x ↵".

Because the commodity.library is only available with Kickstart 2.0 or higher CX_POPKEY will be ignored with older Kickstart revisions.

- CX_PRIORITY

This argument is used to specify the priority for Xopers commodity ↵ broker.

It has to be between -128 and +127. Here applies the same as with ↵ CX_POPKEY:

this switch will be ignored under Kickstart 1.2/1.3.

All those arguments can also be specified when running Xoper from Workbench ↵ by

adding tooltypes to Xopers icon. If you place Xoper in the WBStartup-drawer there is no need to add the tooltype "DONOTWAIT". Xoper *always* creates its own process.

At startup Xoper loads and executes a file named 'Xoper.Startup'. That file ↵ can

be placed in various locations. First Xoper tries to open the script in the current directory, then in env: and last in s:. This startup-script is used ↵ to

set default values, the window size or the initial default list to be ↵ displayed.

Xopers window will be opened after the startup-script has been processed (if ↵ one was found of course).

Xoper can be controlled via 'Exchange' - the commodity master control ↵ program.

With Exchange its possible to force Xoper to show/hide its window or to ↵ remove

Xoper from memory. Removing can also be done by sending Xoper a Ctrl-C. Its ↵ not

possible to deactivate Xopers commodity broker. It stays always active - I ↵ feel

that a hotkey has always to be present.

1.6 Xoper Documentation - Keys

Keys

Xoper has to be controlled over the keyboard. The only exceptions there the mouse can be used are:

- the slider gadget to scroll in the output window up or down
- the dragbar to move Xopers window around
- the various gadgets to control the window size, to depth arrange the window or the screen.

All commands, however, have to be entered via the keyboard. This may seem to be a little bit ancient, but its in no way a restriction. There are not many keys that need to be remembered. :-)

Xopers window (or screen, but even then its a window :) is divided into two sections: an very small input line on the bottom and a large output area on the top. if the output exceeds the size of the window you may scroll or 'page' through the text using the num-pad keys:

A1000	others	
7	Home	Top of display
1	End	Bottom
9	Pg Up	one page up
3	Pg Dn	one page down
8	Up Arrow	one line up
2	Down Arrow	one line down
4	Left Arrow	one page left
6	Right Arrow	one page right

For up and down scrolling you can also use the slider gadget, but this should be obvious.

The input section has some line editing facilities and a history buffer controllable with the following keys:

Cursor left	cursor one character left
Shift left	cursor to the start of the next word
Cursor right	one character right
Shift right	previous word
Cursor Up	previous line in history buffer
Shift Up	top of buffer
Cursor Down	next line in history buffer
Shift Down	bottom of buffer
DEL	delete char under the cursor
BS	delete char left from the cursor
Ctrl X	delete the entire line
Ctrl K	delete EOL
Ins (numpad '0')	toggles insert mode (default is 'on')
Del (numpad '.')	same as DEL
Tab	Command line completion
	Example: press <c>, hit <tab> several times.

ESC moves the input area from the bottom into the ↵
 output section. Entering a command here (i.e overwiting ↵
 an output line), will cause Xoper to add the address ↵
 of that node to your command line.
 Example: Show the hunks of a 'File.System' process ↵
 :
 type 'T' to get a list of all tasks, press <esc>, ↵
 move the cursor to the line displaying the File.System
 process and type 'Hunks <enter>'. Press <enter> ↵
 once more to return to the usual Xoper display.

HELP displays the help panel (may be pressed at any ↵
 time)

1.7 Xoper Documentation - Commands

Commands

Commands have to be entered in the input line at the bottom of the window.
 These are devided into several groups:

Exec lists
 Other system lists
 Commands with parameter
 Commands without parameter
 Toggles and options
 Dangerous commands

1.8 Xoper Documentation - Viewing Exec lists

- EXEC lists

These are always single character entries and may be specified in any ↵
 order,
 upper or lower case, after the prompt or from CLI/WB as parameters. If ↵
 more
 than one list has to be displayed, you may enter several commands in one
 line. Separating them with blanks displays the list one by one instead of
 showing them all at once.

T = Tasks (default if Xoper is started without parameters)

Shows some essential values for tasks (this includes dos processes). ↵
 The
 display contains the address of the tasknode, task type, task state,

task priority in decimal , CPU usage in percent, the process number (←
only
dos processes started from CLI), and task name.

F = Task Flags

Shows also the task list but with some other information than the 'T' ←
command.
Lists the task node in HEX, task state, allocated signals, signals the ←
task is
waiting for, so far received signals, the address of the next ←
instruction to be
executed - usually referred as the program counter (pc) -, for amigas ←
equipped with
a FPU the fpu-state and task name.

U = CPU Usage

Shows again the task list but also with other information than the 'T' ←
or 'F'
command.
Lists the task node in HEX, task state, a tasks run time and its total ←
amount
of cpu time it used so far (both values since Xoper was started!), the ←
process
number and task name.

C = CLI Commands

Shows a list of all CLI commands.
Displays the task node in HEX, task state, type, mode, cli number, cli ←
name
(the process name) and command (the actual executed command).

L = Libraries

D = Devices

R = Resources

Shows a list of all currently loaded libraries, devices or resources. ←
They are
explained together because all have the same basic structure.
Displays the base address, open count, version and revision, flags and ←
name

E = Resident

Shows a list of all resident modules of the system.
Displays the base address, priority, flags, version, type and name for ←
every
resident module found.

M = Memory

Shows all available memory types
Displays the lower and upper bound, free bytes, memory attributes, ←
priority and
hunkname for all memory regions

P = Ports

Shows a list of all public ports in the system.
Displays the port address, port name, port flags, signal bit, queue ←
length (number
of messages and taskname of the owner.

I = Interrupts

Shows a lists of all interrupts.
Displays the interrupts node address, pointer to interrupt data, ←
pointer to
interrupt code, priority, interrupt type ([S]erver/[H]andler), ←
interrupt
interrupt state ([E]nabled/[D]isabled), another type (interrupt queue ←
the
interrupt belongs to) and interrupt name.

S = Stack

Shows information about task stacks.
Displays the lower limit, the actual stack size, the currently used ←
stacksize
and taskname. If the current stackpointer is not within the tasks ←
stackbounds
this function displays "----" as used stacksize (possible candidates ←
for this
are programs launched with ixemuls vfork() call)

CAUTION: if you want to minimize your stack using this utility, please
note that dos-functions prior Kickstart 2.0 use 1500 bytes at
the bottom of the stack frame for their own purpose and note
that the stacksize is only checked *one* time per second by
this command.

A = Semaphores

Shows information about public semaphores.
Displays the semaphores base address, current owner, queuecount, ←
nesting count
and semaphore name.

Example: Entering TPM would display Tasks, Ports and Memory. The display ←
would

be updated after a few seconds (5 by default, may be changed with ↵
the
'time' command)

1.9 Xoper Documentation - Viewing other system lists

- Other system lists -----

Windows

Here you get a list of all existing windows. For every window will be displayed its address, its ownertask, its location relative to its screen, its size and finally the windowtitle.

Screens

This lists all existing screens. The information displayed will contain for every screen the screen address, the screen size, its depth (how many bitplanes) and the screen title. If Kickstart 2.0 or higher is available the next line shows the screens modeid and if possible also the mode name. If a mode name couldn't be retrieved '???' will be displayed.

PubScreens

Only useful with Kickstart 2.0 or higher. Shows for all pubscreen nodes the node address, the screen name, the actual state (public or private) ↵
,
the use count (how many visitors), the attached signal and the ↵
ownertask.

WindowFonts

This retrieves information which font is used in which window. You will see the window address, the ownertask of a window, the window location on its screen and the windowtitle. The next line will show the node address of the window font, the use count of this font, its X and Y size, its type (rom or diskfont, fixed or proportional), the lo and hi character and finally the font name.

Fonts

This command displays a list of all currently loaded fonts (its not a ↵
list
of all *available* fonts). The display contains (as always) the node address, the font use count, its X and Y dimension, its type, the lo ↵
and
hi character, the font name.

Capture

Displays the values for some entries in the ExecBase structure.

-> Cold capture, Cool Capture, Warm Capture and KickMemPtr

These entries can be used to install recoverable programs that survive resets. Using Cold or Cool capture is somewhat a 'dirty' method (in ancient times often used by viri). As far as I know Warm Capture has never and can never be used. The system conform way of installing programs that survive a reset is the usage of KickMemPtr. The display for those entries will contain the start and end location for an entry and its size.

TimerIO

Displays pending timer requests. It shows the address of the IORequest structure, the unit number (MICROHZ or VBLANK), the time to complete and the task submitting the request. Now fixed to work with Kickstart 2.0 and higher (not totally accurate though). Its somewhat "magic".

DiskChange

Shows all installed diskchange interrupts. It displays the node, data and code field of the interrupt, the device it is attached to (df0:-df3:) and the task that added the interrupt (if available). The main purpose of this command was to check for a virus, as the diskchange interrupt is a wonderful place to install them. To retrieve this information in a legal way a diskchange interrupt should be installed. Unfortunately the trackdisk.device prior Kickstart 2.0 had a bug that prevented installed interrupts from being removed. Therefore this function used some internal knowledge of the trackdisk.device. Thats the reason why its not working with OS2.0 anymore and has been disabled. I was to lazy to correct this for OS2.0, sorry.

InputHandler

Displays the node, the priority and name (if any) of all input handlers currently installed. This was the usual way of implementing hotkeys or intercepting the input stream prior Kickstart 2.0. Xopers hotkey facility itself was based on

an inputhandler. Since OS2.0 there exists a commodity.library, that provides functions to deal with input events and to easily add a hotkey to a program. Its much more flexible than the old method.

Devices

Displays the name, heads, sectors, tracks, buffers (if appropriative), the state (loaded or not) and the handler-process of every dos-device
 Note: DOS-Devices are totally different to exec devices!

Files

List the lock, access type, size and the name of all open files.
 CAUTION! This may not work for all devices, but it works for C= ↵
 Handlers
 upto 3.0 ...

Locks

List any lock
 BUG: Trying to lock a Volume "RAM Disk" crashes the machine sometimes. ↵
 If
 a voulume "RAM Disk" is found it will be replaced by the ↵
 devicename
 "RAM:". Make sure you don't have a disk labeled "RAM Disk" or you ↵
 will
 never see its locks :-)
 (The problem affected only V33 of the operating system)
 CAUTION! Here applies the same as with 'Files'. It works for all C= ↵
 Handlers
 but others may fail (eg AmiCDROM) The problem is not a bug in ↵
 the
 handler but of an illegal assumption Xoper makes about a lock!

CurrentDir

List current directory settings of all processes

Low-Memory Handler

Displays the node, the priority and name (if any) of all lowmemory ↵
 handlers
 currently installed.
 Only available with v39 or up.

Frag

Counts free memory hunks by size. Displays the size in hex and decimal, number of hunks and the largest available hunk.

1.10 Xoper Documentation - Commands with parameter

- Commands with parameter

These commands may be entered in upper or lower case. Parameters enclosed in '<>' must be, enclosed '[' may be specified. Names are usually entered as ASCII strings, it can however happen that two or more nodes of the same name exist. On task you may specify the dos-processnumber to sort them out. If everything fails, you can enter the node-address with a leading '\$'. This address will be checked first before the command is being executed. If the check fails you'll get an error message or a warning or a prompt, depending on what went wrong. Names are always the last parameter to enter. This may seem strange, but this is the simplest way to get rid of embedded blanks.

Time <seconds>

Set time between updates. Minimum is 0.1 seconds, maximum 255.9, default is 5 seconds. "Time 0" stops any automatic update and waits for a keypress. Values < 0.5 are not recommended.

Example: Time 1.5

MyPri <priority>

Shortcut for "Taskpri Xoper <priority>"

Example: mypri 2

Taskpri <priority> [processnumber] <taskname>

Change the priority of a task. Values may range from -128 to 127, better use values between -5 and 5

Example: Taskpri 1 Shell Process

Break [processnumber] <taskname>

Set break signals. Useful for tasks running in background or from Workbench.

Example: Break 3 TolleUhr

Freeze [processnumber] <taskname>

Halt a Task. The task should be READY or WAITING. Frozen tasks are ↵
queued
in a new list called FROZEN (this is a private list of Xoper itself!). ↵
When
you leave Xoper, halted Task will be released automatically.

Example: Freeze Killer Graphics Task

Warm [processnum] <taskname>

Restart a halted Task. Task must be FROZEN.

Example: Warm Killer Graphics Task

SnoopMem [processnumber] <taskname>

Track memory allocation/deallocation of a task. Press break (CTRL-C) to
stop. List includes: action (alloc/free), memory requirements (CHIP/ ↵
FAST/
PUBLIC/ etc.), memory size, memory location (start, end) and the ↵
address
from where AllocMem() was called.

TraceOpen/TraceLock

Monitors dos calls to Open() and Lock(). Press break to stop.

Zerotimer [processnumber] <taskname>

Reset the used time counter on the cpu usage display. This is useful if
benchmarking a cli command.

Hide <taskname>

The task-list tends to be longer than the window size. You may inhibit ↵
the
output of some tasks you are not interested in using this command.

Example: Hide trackdisk.device

Pri <priority> <nodename>

Change the priority of any other node. If the specified node has been ↵
found,
the entire list the node belongs to will be resorted.
This command does not work for tasks.

Example: Pri 50 chip memory (try to allocate memory in CHIP first)

Info <librarynode | devicenode>

Show additional information stored in the lib_IdString field. This can be useful for libraries, devices or resources.

NOTE: Some programs dont follow the rules! That can cause an Enforcer hit.

Example: Info arp.library

RemResident <resident module name>

Kicks a resident module out of the ResModules-List. It does not free the module itself, but only makes sure it won't be reactivated during the next reset. Removing a ROM-based module does not have any effect.

Clear [longword]

Fill unused memory chunks with pattern, default is 0. Handy for debuggers.

Example: Clear \$66726565

Hunks [processnumber] <processname>

Show location, BCPL-Pointers and length of memory blocks the process uses.

Note: If the process has only one hunk with length zero the process has usually been created by the dos-function 'CreateNewProc()'

Example: Hunks RAM

Openlib <libraryname>

Open a library. This is useful if you don't want a specified library being 'flushed' out.

Example: Openlib arp.library

Myfont [size] <fontname> | default

This command changes the font for Xopers window. Since version 2.4 Xoper

is no longer restricted to topaz/8. Every fixed width font any size can be

used. For fontname the suffix '.font' can be omitted. This will be added automatically. This command can also be used to reenable the system default font for Xopers window by specifying "default" as fontname.

Examples: myfont 8 macintosh
 or
 myfont default

SetFont [size] <fontname> <window>

Change the default font of a window. To avoid confusion, you should use a font with the same font size as the original font, as many programs rely on the point size.

Example: Windows
 Press <esc>, move the cursor to the Xoper window line and type
 SetFont diamond.font
 (now you know what I mean by 'relying on a font size' :-))

Lockdrive <drivename:>

Prevent DOS, Workbench and Disk-Validator from cluttering on the drive. This command isn't very useful, but I needed it myself. Please note that the drivename is case sensitive and has to end with a ':'.

Freedrive <drivename:>

Re-enable a drive.

Window <leftedge> [toptedge [width [height]]]

Works only in script files. Defines the window to be opened.

Example: Window 0 0 550 190

IconPos <leftedge> [toptedge [width [height]]]

Defines the initial position of Xoper's icon. Used in the startup script.
Only useful prior Kickstart 2.0.

OutputLines <number of lines>

Set the maximum number of lines the output buffer may hold. If the buffer

overflows, a line from the top of the buffer will be deleted for each ↵
new
line.
The default value is 500, using a maximum of $500 * 104 = 52000$ bytes.

HistoryLines <number of lines>

Set the maximum number of input lines the history buffer should hold.
Default is 10 lines.

MinimumChars <number of characters>

Set the minimum number of characters an inputline should have to be ↵
added
to the history buffer.
Default is 2 characters.

SaveOutput <filename>

Write the contents of the output buffer to a file. If the file already
exists, the output will be appended.

PopKey <description>

Change the hotkey description for our commodity. With the hotkey you ↵
can
popup Xoper if it is in sleep mode or simply pop it's window (including
screen) to front. See for instance 'Toolmanger.doc' for a complete key
description (sorry!) The actual hotkey is always displayed in the ↵
window
title.
The default hotkey is 'lcommand - rcommand - X' (means press left amiga ↵
,
right amiga and x at the same time).

Repeat <Command string>

Repeats the command string at the current refresh rate (see 'Time').
However, commands not producing any output won't be repeated, but only
executed once. Press <enter> to stop.

Example: Time 1
Repeat TimerIO

SetFKey <key number> <string>

Assign a string to a funktion key. 'Key number' is a value between ↵
1-20,
10-20 denotes shifted keys. Use '^' to simulate a <return> and '_' for
space (the parser strips leading/trailing blanks).

Example: SetFKey 1 Hunks^
 enter t <return>, press <escape>, move the cursor to a ↵
 process,
 press F1

Alias <AliasName> <CommandName>

Defines a new name to be used along with the original command name. The ↵
 new
 name should not contain any blanks. To delete an existing alias use the
 <AliasName> without a commandname.

Example: Alias ih InputHandler

1.11 Xoper Documentation - Commands without parameter

- Commands without parameter

Alert

Show last Guru Xoper caught.

Lastalert

Show last Guru Meditation code or rubbish. (information obtained by
 exec). Doesn't work when Enforcer is running, as it needs to peek ↵
 memory
 location \$100.

ClrCool,ClrCold,ClrWarm

Clear one of those pointers.

TrapGuru

Activates a trap handler similar to GOMF. It only works with a 68000
 processor, as it relies on a specified stack frame (at least I think so,
 couldn't check it out). If an exception occurs (i.e. GURU) Xoper will
 stop (or popup, if running in background) and display some information
 about what happened (the taskname causing the error, its program ↵
 counter,
 the alert number etc.) and you'll be asked if you want to (K)ill the ↵
 task
 or (I)gnore the exception. Choosing (i)gnore will do nothing at all if ↵
 the
 erroneous program was a process (as it will stop itself displaying a
 'Task held...' requester), but force a task to execute a 'Wait(0L)' (i. ↵
 e.
 wait forever) as tasks do directly display an alert box.

Flush

Clean up memory, flush unused libraries, devices and fonts.

ShowHistory

Show the history buffer. (quite useless, I know)

KillHistory

Delete all lines from the history buffer. (still useless)

ColdReboot

destroys first execbase to force a "coldreboot" and then executes a ↵
normal
reboot

Reboot

Reboot the machine by either the "official reset code" or by Execs ↵
ColdReboot()

[Q]uit or Hold

Exit Xoper but stay in background. When Xoper pops up window settings ↵
and the
selected display are restored.

NOTE: When Xopers window is closed with the WINDOWCLOSE-Gadget, Xoper ↵
doesn't
exit - it goes to sleep. This behavior is *needed* for a ↵
commodity,
because it has to *hide* its window only.

Exit

Clean up and quit.

1.12 Xoper Documentation - Toggles and Options

- Toggles and Options

All options can be entered with either on or off. If a option is entered alone it acts as toggle.

(Exeptions are usescree/usewindow, those are the toggles itself)

Sort

The tasks listing is sorted to avoid 'jumping' of the display. Sort ↔
toggles
this feature on/off. (Stupid command, but was easy to implement).

CLICmd

Toggles between showing the loaded command and the taskname of CLI ↔
processes
in the tasks listing.

TaskInfo

Toggles additional process information (unitnumber, stdio, devicename) ↔
on
and off.

Taskports

Disable / enable a listing of taskports if ports are displayed.

Hidden

Turn those hidden Tasks back on. It is actually a toggle.

Usage

Toggle CPUSE field on the task display between usage relative to all
possible dispatches and usage relative to actually dispatched tasks.
Ahem...not very clear I think. Well, let me try again...
If you add all CPUSE fields together you get 100 % (more or less 1%). ↔
After
entering "Usage" adding the fields together will give you the same ↔
value as
shown in the 'CPU Activity field'. (I HATE having to write docs)

Header

Toggle the (rather long) header on the task display on/off.

UseScreen

Opens Xopers window on a screen. The new screen will take its data (↔
width,
colors, viewmodes, etc.) from the Workbench screen prior to Kickstart ↔
2.0
and will use tags with kick 2.0. Xoper opens a borderless backdrop ↔
window on

the new screen. With V37+ of the operating system the screen will be a public screen, so other programs may open their windows there. Actually, that's a bad idea because Xoper uses only a two colour screen, so other programs' windows may look a bit strange ...

UseWindow

Opens Xoper on a window. The window will appear on the default public screen (usually the Workbench screen, but this can be changed eg. with "ScreenManager" by Bernhard Möllemann)

UseTopaz

Sets Xoper's window font to topaz/8 or to the with 'myfont' selected one. This command has been implemented to quickly flip to topaz/8 because it's fast for the window display.

PropGad

Turns the scrollbar on or off (as you like) The scrollbar will always be adapted to the right border size.

Iconify

Turns the iconify on or off. A small window or appicon will appear if iconify is on. The iconify window is always the fallback if the installation of the appicon fails. The reason for this is in most cases that the workbench has not been started yet (sigh!)

BackDropIcon

Puts the Xoper icon behind all other windows, instead of creating it on top of them.

SmartPatch

Xoper has to patch some system functions. If Xoper is forced to quit it checks that no other program patched the same functions. In this case we

have to wait until the other program(s) end for safe restoring the
 patched
 library vectors. But if you use 'SaferPatches' or 'SetMan' you can quit
 ,
 though the vectors are patched again, because those programs keep track
 of
 the right order functions have been changed. This works only if library
 vectors are changed via Execs SetFunction(). DO NOT alter library
 vector
 by yourself !!!.
 Xoper knows 'SetMan' internally and switches SmartPatch off regardless
 what is entered.

CxHandler

Prior Kickstart 2.0 the hotkey facility was realized with a lowlevel
 inpuhandler. Since Kickstart 2.0 there exist a much more flexible way
 to cope with hotkeys. The commodities.library provides an easy method
 to implement a hotkey facility where the hotkey itself can easily be
 changed. Changing the key combination for the inpuhandler is rather
 difficult (its hard-coded ;-()). But the inpuhandler has one advantage:
 it gets all input events *before* the handler of the commodities.
 library
 in the input stream.
 This command enables you to switch between the lowlevel inpuhandler
 and
 the commodity hotkey.
 With Kickstart 2.0 and above the commodity hotkey facility is on by
 default. The command has no effect with Kickstarts prior 2.0!

1.13 Xoper Documentation - Dangerous Commands

- Dangerous commands (for experienced users only!)

!!!!!! WARNING: The next few commands are dangerous and 'dirty' !!!!!
 !!!!!!!!!!!!!!! don't use them if not strictly necessary !!!!!!!!!!!!!!!

Kill [processnumber] <taskname> (cancel is a synonym or vice versa ...)

Kill a task or a process. If the task has been called from CLI, the
 Task
 itself and the CLI will be killed. Hunks, Windows, Screens and the
 teminal-window will be freed. Simple tasks are just RemTask()'ed. If it
 is
 not a CLI Task you'll be asked if it is a Workbench task, if the answer
 is
 'Yes' unloading will be done by the Workbench. If not, you will be
 prompted
 if Xoper should unload the code. Enter 'No' if you don't know how the
 task
 has been started. A good example for tasks that should NEVER be
 unloaded

are programs started by ARP'S ASyncRun (or ARun).
 Unloading of workbench tasks is no longer possible because the ↵
 Workbench
 port is since OS2.0 no longer a public port!

Closelib <libraryname>

This is exactly the same as CloseLibrary().

Closewindow <title>

Closes a Window. Please, use it only if the corresponding Task has been
 'Cancel'ed. Use the Window-Structure address if the window has no name.

Closescreen <title>

same as above, but for screens. If a screen will be closed first all
 its windows will be removed.

Unlock <lock>

Unlock a file. *** VERY DIRTY *** (not the way how the lock will be ↵
 removed
 but the way Xoper gets the lock)

CD [processnumber] <processname>

Change the current directory of a process. You are prompted if the old
 directory lock should be unlocked.

RemNode <node address>

remove a node from a list.

RemPort <port address>

remove a port from exec's port list.

RemIntServer <interrupt address>

Remove a interrupt server.

Signal <mask> [processnumber] <taskname>

Set any task-signal. Mask is a hexadecimal value with or w/o leading '\$ ↵
 ',.
 See task's SIGWAIT field for sensible values. Tasks normally do not ↵
 wait

for signals only, but for messages, that's why this command may not have
 the desired effect, but it is quite useful for tasks hanging around and
 waiting for events that may never happen. Warning: Using Signal without
 any knowledge about what you are going to signal may cause a system-
 crash!

Example: Signal 10000000 PopCLI III

1.14 Xoper Documentation - Task Types

A so called "task" can be either a simple exec-task or dos-process. A
 process is
 an extension of an exec task. Only processes are allowed - with some
 exceptions -
 to call functions in the dos.library.
 To create an exec-task one can use the Exec function AddTask() or the
 Exec support
 function CreateTask() (that also uses AddTask()). All programs started
 from CLI or
 Workbench are processes. One can create processes manually with the DOS
 functions
 CreateProc() or with the 2.0 version CreateNewProc(). These functions
 cannot be
 used from simple tasks!

For more information about this topic please consult The Amiga Guru Book.

1.15 Xoper Documentation - Task States

The task state indicates what a task is currently doing. The state can be
 :

INVALID	- ???
ADDED	- a newly created task
RUNNING	- a task that has the processor (only one possible!)
READY	- a task that lost the processor and requests it again
WAITING	- a task waiting for an event (better: for signal(s))
EXCEPTION	- a task exception (not processor exception!) has happened
REMOVED	- a task that has ended
FROZEN	- a task has been frozen by Xoper meaning it has been removed either from the ready or waiting queue (this in no way a system state!)

Usually you will only see tasks that are in a running (always Xoper!), a
 ready
 or a waiting state. If a task is permanent in a ready state, this
 indicates that

the task is either heavily working or it makes "busy waiting" (very bad programming habit). To lower the system load you can decrease the taskpriority of this task with the 'taskpri' command. That doesn't hurt the task until no other task requests the processor. To see ADDED or REMOVED is rather unlikely as well as EXCEPTION. Using task exceptions should be avoided due to some severe bugs within exec for handling those exceptions.

1.16 Xoper Documentation - Task Names

The taskname will be the name of the loaded command if the task is a CLI command.

The loaded command name is then enclosed in [].

Additional information displayed can be:

- stdin and stdout enclosed in ()
- devicename enclosed in {} if the process is a handler
- unit number if the process is a device

1.17 Xoper Documentation - CLI Infos

The type field indicates whether the cli is an interactive or a batch one (script files).

The mode for a CLI can be either foreground or background, meaning if the program

was started by Run or Execute().

For more information about this topic please consult The Amiga Guru Book.

1.18 Xoper Documentation - Base Address

This value will be usually the result of a call to OpenLibrary() or OpenResource(),

but this may be wrong for multiple base libraries (means for libraries that provide

every user a fresh data area. Those libraries cannot be SetFunction()ed!)

To get the base address of a device you have to open that device properly by

a call to OpenDevice() and extract the device pointer from the io_Device entry

in the IORequest.

This is a hexadecimal value.

1.19 Xoper Documentation - Open Count

The open count shows how many users a resource has. This value indicates if the resource can removed in a low-memory situation. A non-zero open count means that a resource is still in use. Its displayed as decimal.

1.20 Xoper Documentation - Version and Revision

The version of system libraries is usually a hint which version of the operating system is installed, eg. v33 indicates OS1.2, v37 OS2.04, etc. The revision field shows how often the resource has been changed during the development cycle. Programs that require a certain version of the operating system simply submit the at least needed library version to a call of `OpenLibrary()`. Both values re displayed as decimal.

1.21 Xoper Documentation - Flags

The flags indicate the current state of a resource or what shall happen if a certain state will arise, eg. an alert if the library checksum doesn't match the one stored in the library base. The value will be display as binary.

1.22 Xoper Documentation - Name

This field will be used if a certain library has to be opened. The name is case-sesitiv, so "Name" is not "name". It is also important that for transient resources its diskname matches excatly the internal name otherwise the system refuses loading those resources.

1.23 Xoper Documentation - The Amiga Guru Book

```
##### #   # #####
#       #   #
#       ##### #####
#       #   #
```

```

          #   #   # #####

      ### #   # ### ##### ###
      #   # ## ## #   #   #
      ##### # # # #   # ## #####
      #   # #   # #   #   # #
      #   # #   # ### ##### #   #

      ##### #   # ##### #   #   #####   ###   ### #   #
      #       #   # #   # #   #   #   #   #   #   #   #
      #   ## #   # ##### #   #   ##### #   #   #   #   ###
      #   # #   # #   # #   #   #   #   #   #   #   #   #
      #####   ### #   #   ###   #####   ###   ###   #   #

```

The Amiga Guru Book - a reference manual - by Ralph Babel

1.24 Xoper Documentation - Bugs and Limitations

Bugs and Limitations

There are no severe bugs known at the moment, however Xoper has some limitations and problems.

Enforcer hits may happen due to accesses to the low memory area (\$100.W, \$180.W) but these are not dangerous (read accesses). Other Enforcer hits may be caused by examining system structures that are not initialized properly (eg. the "keymap. resource" with OS3.0 for lib_IdString, but may be it can be nethertheless Xopers fault: if resources are no "real" libraries)

There exists a problem when Xoper runs on graphic cards. It has been reported that the cursor is not displayed on a Picasso-II and a Retina Z3. I was unable to track this down, sorry.

Furthermore, it has been reported that with Kickstart 3.1 (v40) the system freezes when attempting to mount PC0: if Xoper is already running :(I tried to reproduce that behavior on my system, but it didn't happen. My program versions which worked:

- mount 40.3 (27.5.93)
- mfm.device 40.9 (21.5.93)
- CrossDOSFileSystem 40.19 (9.6.93)

Please note, that I fixed bugs with mfm.device (trashed d3 with OpenDevice()) and mount (didn't recognize tooltypes), but that shouldn't be the solution.

Many internal buffers have a fixed length:

- input buffer can only hold a maximum of 120 characters
- task buffer can only handle a maximum of 125 task (should be enough anyway) ←

The trapguru function works only for 68000 processors. It may sometimes work also for higher cpus, but better don't rely on this. Every 680x0 processor has a special stackframe format for exceptions and I do not know much about this. ←

Killing a task is a very dirty and dangerous thing on the amiga. So using this command may led to a system crash, so beware! Also, prior Kickstart 2.0 the workbench replyport was a public port. This is no longer true for OS3.0 or even OS2.0! That makes it impossible to simulate a workbench exit :-(←

The diskchange command works only with Kickstart 1.2/1.3. Please refer to the description of that command for the reasons. ←

And now the very dirty part:

All dos related functions make a special assumptions about one value in a filelock structure (nethertheless it works upto 3.0 ;-()). Those functions are likely to produce strange results, eg. if AmiCDROM is used as handler for a cdrom-drive (only with an inserted medium) ←
I know these a very dirty facts, but I left the functions in the program because they were often useful though (eg. PasTeX MUI-Specialhost did not unlock newly created directories) ←

1.25 Xoper Documentation - Technical Info

Technical Info

Xoper was written entirely in assembler, but that is not a quality sign. In fact, if you look at the source, you will notice, that it doesn't follow the rules. Parameters are not passed in the regular registers and results don't come in "d0". Common non-scratch register are not saved in subroutines or sometimes a subroutine will be entered on a different location than the main entry. All these facts make it very hard to change ←

Xoper due to unpredictable side effects. The source is not commented very well either.

First A68k v2.71 was used to assemble Xoper, later then SNMA (you need at least v1.95).

Other assemblers may have problems with the source code (eg. PhxAss does not like ".")

in labels and has problems with some expressions).

Xoper uses a combined data/bbs segment. This avoids lots of relocations because even

bss entries can be accessed over the base register. Many c-compilers use this to keep

its programs short (GCC,SAS/C,DICE,Aztec). To create such a "short" executable you

need a linker that suppresses all zeros at the end of a hunk. I recommend PhxLnk

because its free and fast. You need at least V4 of that linker!

Xoper v2.4 and up were developed on an A4000/030 with Kickstart 3.0. It has also

been tested with a softkicked 1.2 (V33).

Xoper is a system monitor and offers some information about the cpu usage of tasks.

To get those informations Xoper patches two functions in the Exec library : Switch()

and AddTask(). The latter is well documented and in no way a secret. However, the

Switch() entry in the Exec library seems to be "secret". This functions has never

been documented but until v37 of the os the exec fd-file contained the name of this

function. The only things Xoper "knows" about Switch() is its name, offset within

the Exec library and purpose. Whether this function has arguments or not does in no

way affect Xopers work. All that Xopers is doing in its patch function, is to add

the current task (SysBase->ThisTask) in an internal buffer and to update an internal

timer value for that task. After that the original function will be executed with

the original registers settings. There is no magic how Xoper gets the cpu usage for

each task!

1.26 Xoper Documentation - History

History

versions developed by Gunther Nikl

- 2.5
- cpuse dropped down to 0.0% for all tasks because of incorrect overflow handling in the switch function (Vogt, Bill Best)
 - window and iconpos examined only the first argument caused by not handling trailing blanks (Thomas Schwartz)
 - didn't recognize all floppy drives when one was missing eg DF0: -> DF2:
 - Enforcer hit in iconify() caused by using a null pointer instead of a pointer to a string with len zero - I should buy includes&autodocs (Jörg-Cyril Höhle)
 - crash due to killing Xoper twice or killing Xoper when it was waiting to restore the patched library vectors
 - exec list of libraries, devices and resources now show the correct values for opencount, version && revision (word instead lower byte) (Gary Duncan)
 - zip gadget behavior for kick2.0+ corrected, switches now between two different sizes (Jörg-Cyril Höhle)
 - checks first if a resource to be freed/deallocated is owned by Xoper eg. for windows, screens, ports and interrupt-servers.
a problem is still how to validate a remnode() ?
 - system crash when 'cxhandler' was specified in the startup-script due to trying to set the windowtitle for the noexisting! window (Stefan Becker)
 - new command 'myfont' to change Xopers window font
 - font handling improved. checks that the font returned by OpenFont() has the requested size. otherwise a OpenDiskFont() is performed. When a diskfont couldn't be opened the result of OpenFont() will be used
 - setprop now scans message list correctly, looked one to far - oops (Jörg Cyril Höhle)
 - no more (partial) overwriting of the delimiter line if window size was unsuitable (really fixed?)
 - when not running on an own custom screen adapts progadget size to the window border size (suggested by Jörg-Cyril Höhle)
 - largest value that can be converted to decimal is now 10^9-1 , should be enough for virtual memory :-)) (Jörg-Cyril Höhle)
 - timerio finally works with kick2.0+, converts _eclockval_ to _timeval_! (Jörg-Cyril Höhle)
 - division function uses a 68020 instruction (if appropriate)
 - startup-code revised (better lock handling)
 - 'stack' command rewritten (according to the book),
 - stack now displays "----" if current stackpointer is not within the tasks stack bounds (suggested by Jörg-Cyril Höhle)
 - crash if 'exit' was specified in the startup-script (tried to remove the noninstalled! inputhandler)
 - new commandline option 'noscript'
 - EOL is now con:-default ctrl-k (suggested by Jörg-Cyril Höhle)
 - fatal taskend if GUI-hiding was requested with pubscreen still used
 - new command 'lowmemhandler'
 - if iconify was off a hide-message erroneously caused a 'wakeup'
- 2.4
- major code cleanup and lots of bugs removed
 - non-proportional fonts different than 8x8 pixels are now handled correctly
 - new command 'usetopaz' to force Xoper to use the "default" topaz/8
 - takes care of the screens barheight and window borders width
 - commandline completion now works at every! position of the inputline and cycles instantly
 - when running under Kickstart 2.0+ Xoper clones the default public screen (size, modeid,...)
-

- Xopers own screen is a pulic screen called 'Xoper' (2.0+)
- uses commodity and appicon feature of Kickstart 2.0+
- searches startupscript at various locations (currentdir, env:, s:)
- older commands (cancel, alert, remnode, remintserver) displayed in the help page
- input-handler (checking for Amiga-Amiga-X) now installed permantly
- new command 'pubscreens'
- screen display shows modeid (only when running under 2.0 or higher)
- removed all selfmodifying code
- system conform and safe patching of library vectors
- IconifyOff and PropGadOff replaced with the options 'Iconify' and 'PropGad' that turn the corresponding feature on or off
- startup code completely rewritten

 versions developed by Werner Günther

- 2.3 - general fixes for WB2.0x, FPU and 680x0 processors
 - 'task usage' section completely redesigned, including a new display showing the total amount of cpu-time by task
 - added new fields to the task, fonts, screen and cli dsisplay
 - added a scrollbar (why not use the keyboard :)
 - open files and filelocks can now be logged
 - added a 'frags' like display
 - 'saveoutput' appends its output to a file if it exists
 - toggling commands may be followed by 'on' or 'off' for clearer startup-scripts
 - 68881 code was done by Lothar English
 - 2.2 - 'KillXoper' integrated into Xoper's main program
 - loaded CLI commands are now always displayed, not only in the tasklist
 - 2.1 - KS1.3 dependancy removed
 - fixed a crash with 'usescreen' in startup script if Xoper was started with the '-b' flag
 - s (stack) command wasn't robust enough
 - removed some strangness in the 'kill' routine
 - iconizing routine couldn't distinguish between multiple drags and doubleclicks
 - 'time' now accepts values < 1
 - 'windows' shows owner task (if available)
 - task display shows name of the loaded command (enclosed in[]) instead of the name, if the task is a CLI.
 - 'clcmd' toggles this feature on/off
 - commandline completion using <tab>
 - added a new command (setfkey)
 - a new program 'KillXoper' has been added to remove Xoper from memory in case it loops or freezes (I hope it won't be needed)
 - 2.0 - user interface (what user interface ?) rewritten from scratch
 - added a small iconify routine
 - added again a few new commands to customize the whole thing a little bit
 - (minimumchars, historylines, showhistory, killhistory, outputlines, iconifyoff, backdropicon, usescreen, usewindow)
-

- other new commands
(timerio, remresident, repeat, trapguru, setfont, diskchange, alias, saveoutput)
 - addresses now shown as 32-bit values for 68020 compatibility
 - 'more' command was obsolete and has been removed
 - 'interrupt list' has two new fields
 - 'time' w/o parameters shows current setting
 - 'display commands' separated by blanks will display the list one by one, instead of displaying them all at once
- 1.3b - added 'C' information on CLI-tasks
- 'time 0' stops any update
 - system requesters are now handled correctly (affects 'kill' and 'closewindow')
- 1.3 - added new commands (stack, sort, hide, hidden, header, window, inpuhandler)
- added support for a startup-script
 - added I/O interrupts/second
 - added 'kill' as an alias for 'cancel'
- fixed bug in the port-display that caused a GURU if more than 32 ports did exist
 - the cli Xoper has been started did act like having a priority of 127
 - unlock didn't unlock sometimes
 - interrupt/priority field did contain rubbish
 - currentdir didn't examine all processes
 - some more bugs (hopefully) removed
- 1.2 - added new commands (snoop, capture, clrcold, clrcool, clrwarm)
- added cpu usage by task
 - cancel command rewritten
 - some minor bugs removed

1.27 Xoper Documentation - Acknowledgments

Acknowledgments

Thanks to all who submitted bugreports ;-(or suggestions that kept me ←
working
on Xoper. Without their help there would be no new version.

Werner Günther - for writing the original program :-)

Jörg-Cyril Hühle - for bugreports, useful suggestions and demanding ' ←
fixes' :-)

Stefan Becker - for bugreports and his excellent 'ToolManager'

Ralph Babel - for his great "Amiga Guru Book"

Samu Nuojuua - for adding the dx feature to his assembler

Vogt (france), Thomas Schwarz, Bill Best, Oliver Jeannet,
Gary Duncan - for bugreports

1.28 Xoper Documentation - Disclaimer

Disclaimer

Standard disclaimer:

THERE IS NO WARRANTY FOR THIS PROGRAM TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHERE OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDER AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1.29 Xoper Documentation - Copyright

Copyright

Xoper may be freely distributed for non-commercial purposes, as long as ↵
all the
files in the original archive are present and have not been modified in ↵
any way.

No charge for Xoper may be made, other than a reasonable cost to cover ↵
the media
and copying time.

1.30 Xoper Documentation - Author

Author

Since v2.4 Xoper is maintained and enhanced by Gunther Nikl. If you find ↵
any bugs,
have some suggestions to enhance or improve the program, please contact ↵
me:

email: gnikl@informatik.uni-rostock.de (prefered)

or

smail: Gunther Nikl
Hans-Beimler-Strasse 17
Parchim
19370
GERMANY

The original program was written and developed upto v2.3 by Werner Günther ↵
.

email: x41@sun0.urz.uni-heidelberg.de

Thanks to him that he made the program (some people disagree - I know :)

PLEASE NOTE: He is no longer involved in the further development of Xoper,
so please don't contact him - it would be the wrong address!