Event!

COLLABORATORS			
	TITLE : Event!		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 24, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Event!

1.1 Event!.guide

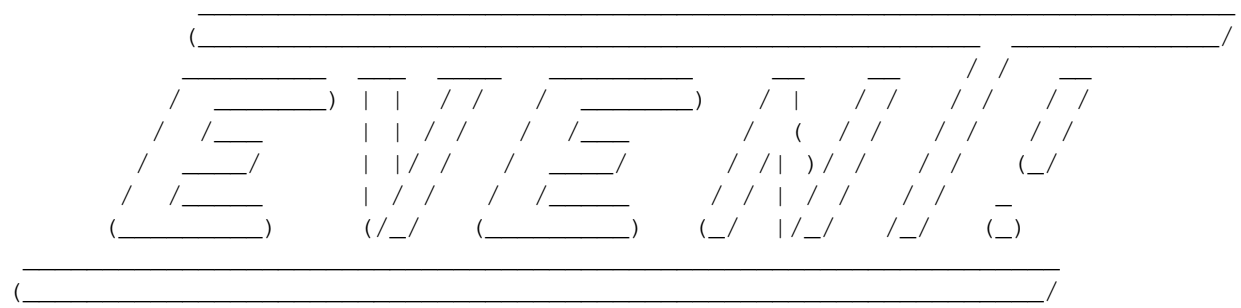
```
Event! - The event scheduler for the Commodore Amiga
----------------------------------------------------

What's it all
About?

Introduction
Installation
Starting Event!

The intuition interface

Event!'s  intuition  interface  is quite self-explanatory.  Anyway
here they are described:

The Main Configuration Window
The Main Configuration Window's Gadgets
The Main Configuration Window's Menus

The Event Edit Window
The Event Edit Window's Gadgets
The Event Edit Window's Menu

The Clock Window
The Clock Window's Gadgets
The Clock Window's Menu

Keyboard shortcuts for Event!'s gadgets

And now for the profi user:

The ARexx Interface

How to optimize the use of Event!:

Hints

If you are a programmer read this even if you don't want to use Event!:
```

For application programmers

Last not least the things that must not miss in any documentation:

1.2 Event!.guide/About

Stefan Hochmuth proudly presents:



====== Version 1.2 ======

An event scheduler for the Commodore Amiga.
copyright (C) 1992-94 by Stefan Hochmuth
Shareware, OS 2.0 and higher only

1.3 Event!.guide/Introduction

Some words about this software package...

1.4 Event!.guide/What can Event! do for me?

What can Event! do for me?

Event! is an intuition based commodity for managing all kinds
of events that need to be executed regularly or on a specific
condition (time, file notification, hotkeys) on your Amiga system.

If you, for example, run a FidoNet(TM) compatible system, you would
schedule your polls with Event!, set your Mailer to ZMH-Mode during ZMH,
generate automatic mail, etc.. You can define an event that looks at your
inbound mail directory for changes and let it start your import tool
(tosser).

You can define hotkeys for all the programs, that you use regularly,
so you won't have to use the workbench anymore.

You may define events for automatically extracting downloaded archives,
playing downloaded modules, show downloaded pictures, update distribution
archives, assemble or compile code, etc..

You may tell Event! to wake you up in the morning with your favourite
sound module, tell you the current time through SPEAK:,
to remind you of the date with your girlfriend, and so on.

1.5 Event!.guide/Why should I use Event! ?

Why should I use Event! ?

Event! offers a full ARexx interface with lots of useful
commands.

Event! has an extensive font-sensitive intuition user interface
following the guidelines of the C= "User Interface Style Guide"
and supporting keyboard shortcuts for gadgets.

Event! can display messages, play sounds and melodies,
open requesters, and execute ARexx and CLI commands.

Event! offers a small clock window/memory with an user
configurable font, that can optionally follow to any active
public screen.

Event! won't skip events if the clockwindow display can't be
updated. The checking for events is done by a separate task!

Event! let's you determine even the second of the time the event
should occur.

Event! is able to determine in how many minutes a specific event
occurs

Event! can execute events when a file or directory changes!

Event! optionally writes event occurrences to a logfile.

Event!'s events support hotkeys.

Event! is written in 100% assembly code.

Event! makes extensive use of OS2.0, WB 2.1 and OS 3.0 functions.

Event! is extremely configurable and flexible.

Event! has been tested by many betatesters on many different Amigas for a long time.

Event!'s evaluation version only differs from the registrated version by a self-disappearing shareware-reminder on startup.

Event! is localized. (currently for English, German and Danish)

Event! opens its AmigaGuide-documentation asynchronously when you press the HELP key.

Event! is a commodity.

Event! is simply the best. :-)

1.6 Event!.guide/What you can do for me

What you can do for me

Do me a favour and register Event!. It took many months of hard work to create this program in this level of perfection you got it in front of you now. Read the file "Registration_Form" for further information. Additionally, if you register you get a keyfile which prevents the "About"-requester to pop up at program start.

1.7 Event!.guide/Out of English

Out of English

I think my English isn't very good anymore. If you find any orthographical or grammatical mistakes in this documentation please don't hesitate to tell me so I can correct them in the next version.

1.8 Event!.guide/System requirements

System requirements

Event! needs OS 2.0 (V37) and ARexx to operate.

For localization WB 2.1 (V38) is required.

For online help you need amigaguide.library.

Event! will make use of some special functions of Kickstart 3.x
if available.

1.9 Event!.guide/Legal stuff

Legal stuff

    Warranty

1.10 Event!.guide/Installation

```
Installation

The most easiest way to install Event!  is by using the provided
Install-Script.  You will need the Installer by Commodore for this purpose
but this should be no problem since Commodore made it freely distributable
for users.  The Install Script will copy Event!  either to your
SYS:WBStartup-directory or any other directory.  If you don't specify
SYS:WBStartup as directory you will be asked if Event!  should be started
from "s:user-startup".  If you answer "Yes" the necessary lines will be
added to your "s:user-startup".  So Event!  will get started every time
your system boots up.

If you can't use Installer you should at least copy the program file
to the desired place and the necessary catalogs to LOCALE: if you own
WB 2.1 or higher and want to use localization.
```

1.11 Event!.guide/Starting Event!

```
Starting Event!

From Workbench
From Shell
Important Note
```

1.12 Event!.guide/From Workbench

```
From Workbench

Simply move Event! into your SYS:WBStartup directory. If you own
the Installer-Utility by Commodore, simply click on the
Install Event!-icon and any necessary files
will be copied.

The ToolTypes for Workbench Start are the same as the
Shell arguments.

Additionally you may use the following tooltypes:

STARTPRI

set  this lower than the STARTPRI of RexxMast if you start RexxMast
from WBStartup.

TOOLPRI

The initial TaskPriority of the Event! main task. May be overwritten
by MAINPRI while reading in the config.

CREATEICONS
```

Create Icons for configuration files. Default=YES
If Event! is started from CLI iconcreation is always turned off.

Example: CREATICONS=NO

DONOTWAIT

You have to specify this tooltype if you put Event! in your WBStartup directory.

To start Event! with another configuration than "s:Event!.cfg" you have to save the configuration with enabled iconcreation and start the configuration just like Event!. This also works with the WBStartup-directory.

Alternatively you may hold down the SHIFT-key, click on the configuration file icon and double-click on Event!'s icon .

1.13 Event!.guide/From Shell

From Shell

It is advised to put a line like "Run >NIL: Event!" in your s:UserStartup after REXXMAST.
If you own the Installer-Utility by Commodore, simply click on the Install Event!-icon and any necessary files will be copied.

Template: FROM,CX__PRIORITY/N/K,CX__POPKEY/K,CX__POPUP/K,GUIDE/K

FROM (CLI only)

Event! configuration, defaults to S:Event!.CFG

CX__PRIORITY/N/K

Priority of Event! within the commodities queue. Default: 0.

CX__POPUP/K

Open configuration window after startup. Default: YES.

Example: CX__POPUP=NO

CX__POPKEY <key>

Commodities Hotkey definition for showing the userinterface. Default: "lalt e".

Example: CX__POPKEY=ctrl lcommand e

GUIDE/K <guidefile>

Path and FileName for the AmigaGuide online help file.

Example: GUIDE=SYS:Utilities/Guides/Event!.guide

1.14 Event!.guide/Important Note

Important Note

Make sure that the ARexx-Servertask ("RexxMast") is already running
when you start Event! at system startup from s:User-Startup or the
WBStartup directory. To ensure this, start the "RexxMast" command BEFORE
Event!, either by placing it before Event! in your user-startup or by
using the ToolType STARTPRI=1 for RexxMast in your WBStartup.

1.15 Event!.guide/The Main Configuration Window

The Main Configuration Window

This window is used to configure Event! and to work on whole
event-definitions, like creating a new editable event or deleting it, etc.
This window opens

-> on startup if you did not specify "CX_POPUP=NO"

-> if you select to show interface on Event! in the Commodities
Exchange utility.

-> if you select "Open Configuration Window" from the clock's
menu

-> if you press the CX_POPKEY sequence. Default is "LALT e"
which means you have to hold down the left alt key and press
"e". The selected popkey is displayed in the title of Event!'s
main configuration window.

-> if you send CTRL-F to the main task.

-> if you start Event! though it is already running.

1.16 Event!.guide/The Main Configuration Window Gadgets

The Main Configuration Window contains the following gadgets:

```
The list of event definitions      : Event Definitions

Create a new event definition      : New
Edit an event definition           : Edit...
Cut an event definition            : Cut
Duplicate an event definition      : Clone
Execute an event definition        : Test
Paste an event definition          : Paste
```

```
Move an event definition to the top: Top
Move an event definiton upwards    : Up
Move an event definition downwards : Down
Move an event definition to the end: Bottom

Specify the log file to use        : Logfile
Toggle small/verbose log files     : Tiny Logs

Set type of clock display          : ClockType
Toggle following to public screens : Follow to Active PubScreen
Use clock's screen's font for clock: Use Screen's Font for Clock
Specify clock's font               : Specify...

Set main task's priority           : Main Task Priority
Set event checking task's priority : Check Task Priority

Save configuration                 : Save
Hide the main configuration window : Hide
Quit Event! -NO!!!                 : Quit
```

1.17 Event!.guide/Event Definitions

```
    Event Definitions
```

This is a listview gadget showing the list of events you can operate on with the surrounding gadgets in the same box. You have to select an event in this gadget before you can operate on it.

1.18 Event!.guide/New

```
   New
```

Use this gadget to create a new event. If you have selected an event in the definitions gadget the new event will be inserted right below it otherwise at the end of the list.

Note: The name of the new event will be set to "New Event". All other string gadgets will be left empty. The new event is set to occur "Never" but if you select to let it occur "On Time" you will find that the occurance date is set to the current time with seconds zero and the day of week and year as wildcards. The only set action flag is to bring the clockwindow to front.

1.19 Event!.guide/Edit...

```
    Edit
```

If you select this gadget you will open The Event Edit Window

Note: If the event-edit window is already open this gadget will first work like pressing the Ok gadget in the event edit window. Then if the definition in the event edit window is o.k. it will allow you to edit the selected event.

1.20 Event!.guide/Cut

Cut

Copies the selected event definition to the clipboard in ASCII format. Afterwards the event definition is erased. You can use the event definition in other clipboard supporting programs like CED, if you use it together with CB2REXX, a freely distributable utility. Note: CygnusEd 3.5 supports the clipboard anyway, so you don't need CB2REXX.

1.21 Event!.guide/Clone

Clone

Duplicates the selected event and inserts it right below the selected one. The copy becomes the currently selected event.

1.22 Event!.guide/Test

Test

Executes the action part of the selected event.

1.23 Event!.guide/Paste

Paste

Paste an event (or even an ARexx Command) from the clipboard.

1.24 Event!.guide/TopUpDownBottom

Top / Up / Down / Bottom

Moves the selected event in the list.

1.25 Event!.guide/Logfile

```
    Logfile
```

In this gadget you can enter a filename where you want the logfile written to. Leave this empty to disable logging.

1.26 Event!.guide/Tiny Logs

```
    Tiny Logs
```

This checkbox determines the amount of information written to the logfile. If you select "Tiny Logs" only date, time, name and cause of the occured events will be written to, otherwise also the _whole_ event definition.

1.27 Event!.guide/ClockType

```
    ClockType
```

Select this to select the kind of clock you want Event! to display. You can select between "No Clock", "Clock Only" and "Clock And Mem" which should be selfexplanatory.

Note: If you select "No Clock" and an event occurs which should display a message the clockwindow will open for the time the message is displayed and disappear afterwards immediately again.

1.28 Event!.guide/Follow to Active PubScreen

```
    Follow to Active PubScreen
```

Since Amiga OS Release 2 programs may open their screens in public mode which means that other programs can open their windows on it in a system conform manner. If you select this gadget, Event!'s clockwindow will follow to the active public screen. Note that some programs only support visitor windows in a dirty way. CED 2.12 is one of this programs which guru sometimes when a visitorwindow is on it, but the screen should be closed. It is not advised to run CED in public screen mode.

1.29 Event!.guide/Use Screen's Font for Clock

```
    Use Screen's Font for Clock
```

If this gadget is selected the clock window uses the font of the screen it is on.

1.30 Event!.guide/Specify...

Specify...

This gadget is mutually exclusive to the Use Screen's Font for Clock gadget. If you select this gadget a fontrequester will pop up and ask you for the font the clock should use. Note that proportional fonts are accepted but not advisable since mostly there'll be some space left in the right part of the clock. Note also that you have to select the "Use Screen's Font for Clock" gadget first if you want to select a new font.

1.31 Event!.guide/Main Task Priority

Main Task Priority

This is the priority Event!'s main task runs with. The main task does everything but checking if an event occured and creating the string for the clockwindow which is done by the event check task. The main task also executes occuring events and displays the time and date in the clockwindow. This is also the priority CLI commands will be executed at.

1.32 Event!.guide/Check Task Priority

Check Task Priority

See Main Task Priority

1.33 Event!.guide/Save

Save

Saves the configuration to the current configuration file which is also displayed in the screentitle of the window.

1.34 Event!.guide/Hide

Hide

Hides the main configuration window user interface. Besides, same as the closegadget.

1.35 Event!.guide/Quit

```
    Quit

    Guess what!
```

1.36 Event!.guide/The Main Configuration Window Menus

```
Menu Items in the Main Configuration Window

Project Menu:

Open a configuration file          : Open
Append a configuration file        : Append...
Execute an ARexx-script            : Execute ARexx-Script
Save configuration                 : Save
Save configuration as...           : Save as
Hide the main configuration window : Hide
Display About-window               : About
No description available           : Quit

Edit Menu:

Remove all event definitions       : Remove All Events
Cut an event definition to clip    : Cut
Copy an event definition to clip   : Copy
Paste an event definition from clip: Paste
Erase an event definition          : Erase

Special Menu:

Snapshot window's position         : Snapshot window
Toggle creation of icons           : Create Icons
```

1.37 Event!.guide/Open...

```
   Open...

Load a new configuration.  Note that the previous configuration will be
lost!

IMPORTANT Note:  Config reading is done asynchronously.  So you should wait
for the config reading to complete before you save it.  If you don't,
you'll save a incomplete configuration.

Note also:  Event!-configuration files are simply ARexx-Scripts.  Due to
limits of ARexx you cannot execute scriptfiles containing a space, so
Event!  configuration filenames mustn't contain spaces, too.  Since many
users will extract their Event!-archives to "Ram Disk:", Event!  converts
this volume name to "Ram:"
```

1.38 Event!.guide/Append...

```
    Append...
```

Same as "Open...", but doesn't erase the eventlist before executing the script.

1.39 Event!.guide/Execute ARexx-Script...

```
    Execute ARexx-Script...
```

Internally nearly the same as "Append..." since configurations are ARexx-Scripts. The only difference is that "Append..." tries to load the corresponding icon for reuse when saving the configuration with icons.

1.40 Event!.guide/MSave

```
   Save
```

Same as the Save-Gadget.

1.41 Event!.guide/Save as

```
   Save as
```

A requester will ask you for the name of the file you want the configuration to be saved to. If you select "Ok" in the requester this name will become the new current configuration name.

1.42 Event!.guide/MHide

```
   Hide
```

Same as the Hide-Gadget.

1.43 Event!.guide/MAbout

```
   About
```

Displays some information about Event!. If you registered and own a keyfile, your name and address will be displayed as well.

1.44 Event!.guide/MQuit

Quit

Don't do it ;-)

1.45 Event!.guide/Remove All Events

Remove All Events

This will erase all eventdefinitions in the list.

1.46 Event!.guide/MCut

Cut

See gadget Cut.

1.47 Event!.guide/MCopy

Copy

Like Cut but the event is not erased.

1.48 Event!.guide/MPaste

Paste

Paste an event (or even an ARexx Command) from the clipboard.

1.49 Event!.guide/Erase

Erase

Delete selected event. No copy to the clipboard is done. An in this way deleted event can't be recalled.

1.50 Event!.guide/Snapshot window

```
   Snapshot window
```

Saves the position of the window to the Text Gadgets. These are the positions that will be saved to the configuration file. Do not mix this up with Event!'s remembering of window positions which is only for the current session.

1.51 Event!.guide/Create Icons

```
   Create Icons
```

The state of this menu items's checkmark determines if Event!'s configuration files are saved with icons.

1.52 Event!.guide/The Event Edit Window

This window allows you to edit an event definition. You can open it by selecting an event and clicking on "Edit..." in the main configuration window.

The Event Edit Window Gadgets are divided into three parts:

The left side of the window contains the gadgets describing the conditions when an event will occur.

The right side of the window contains the gadgets describing the actions executed if an event occurs.

The remaining gadgets have general functions, i.e. naming the event or confirming the event definition.

1.53 Event!.guide/The Event Edit Gadgets

```
   The Event Edit Window Gadgets
```

```
Left Side: Conditions

Specify condition type            : Occur
Define time conditions            : The Time Definition Gadgets
File/directory to supervise       : Change of File
Notification tolerance time       : Tolerance Secs
This event's hotkey               : Action Hotkey

Right Side: Actions

 Audible beep                     : Audible Beep
```

```
 Visible beep                      : Visible Beep
 Clock to front                    : Clockwindow to Front
 Clock's screen to front           : Clockwindows screen to front
 Occur only once                   : Only Once
 Do not log occurance              : Do Not Log
 Erase event when occured          : Kill When Done (NOT saved)
 Display/Request Action            : Display Type
 Run/RX/Melody Action              : Command Type

General gadgets:

Specify the event's name           : Name
Remember changes                   : Ok
Like "Ok" but goes to prev. defin.: Ok & Prev
Like "Ok" but goes to next defin. : Ok & Next
Execute event definition           : Test it!
Forget changes                     : Cancel
```

1.54 Event!.guide/Occur

```
   Occur

This is a cycle gadget. You can select between:

    "Never":    The  event  will  only  occur  when  "Test"-ed  or the
    specified hotkey is pressed.

    "On   Time":   The  event  will occur when the specified time (see
    The Time Definition Gadgets) is matching.

    "On  Notification" The event will occur when the specified file or
    directory is changed (see Change of File).
```

1.55 Event!.guide/The Time Definition Gadgets

```
   The Time Definition Gadgets

An "On Time" event will only occur if all the time gadgets match the
current time.  By choosing the WeekDay / Day / Month / Year" gadgets you
specify the date an event will occur.  The "Hour / Minute / Seconds" you
choose the time an event will occur.  If the checkbox at the right side of
a time gadget isn't checked, Event!  will not compare the current time to
this part of the time definition, it is a "wildcard" and its gadget is
disabled.  With this method you can define events that occur every hour or
every week or the first of every month, etc.

Hint:

If you want several events use the same action part of an event you can
define multiple events using the "TEST" ARexx command to start the action
part of this event.  This can be useful if you want i.e.  an event occuring
```

every Monday, Wednesday and Friday. Thus, if you want to change the action you have to change only one event. But be careful: ARexx commands for Event! are first parsed by the ARexx-parser and then by ReadArgs(). For further information see "If 'Command Type' is ARexx" in the Command Type description.

1.56 Event!.guide/Change of File

Change of File

In this gadget you can enter a file or directory name. The event will occur if "On Notification" is set and the given file or directory changes.

Deeper Information: This uses the notification mechanism of dos.library V37+. This doesn't work on all filesystems. With the OFS and FFS in Kickstart V2.0 and the RAM: Disk V37+ it works fine anyway. I am sorry it is not possible to use wildcards or to determine which file in a directory changed cause there is no mechanism provided for this in the current release of dos.library. ;-(

1.57 Event!.guide/Tolerance Secs

Tolerance Secs

Many programs do more than one file/directory accesses within a very short time, for which only a single notification event is desired.
I.e. if CygnusED has switched on "Safe saves" it first saves to a temporary file, afterwards it deletes the old file and then renames the temporary file. If you had set a notification to this file this would result in 2 notifications. To prevent this you can specify with this gadget, how many seconds Event! tolerates notifications for the notification object without executing the event. As soon as no notifications for the specified time occured, the event will be executed.

This feature is also very useful for notifications to the inbound-directory of your system since TrapDoor also works with temporary files.

1.58 Event!.guide/Action Hotkey

Action Hotkey

This defines a Hotkey for executing the action part of an event using an input description.

You may define a hotkey for every event, so that you can execute them manually, without having to bring up Event!'s intuition interface. A hotkey start overrides even the Occur- "Never"cycle gadget setting.

1.59 Event!.guide/Audible Beep

```
    Audible Beep
```

It beeps when the event occurs. The audio.device is used for this. If no beep can be heard, Event! couldn't get a channel because it was already allocated. Or you've just got to turn up the volume! ;-)

1.60 Event!.guide/Visible Beep

```
    Visible Beep
```

The screen will flash.

Note:

This one uses the intuition function DisplayBeep(). In later releases than OS2.0 of the system software you will be able to choose what happens if DisplayBeep() is called. So there might be no flashing but a sound for example.

1.61 Event!.guide/Clockwindow to Front

```
   Clockwindow to Front
```

Bring the clockwindow to front if it is open. You should set this for events that display messages in the clockwindow. It is also advisable that you bring the window sometimes (i.e. once a minute to front" in case it is obscured by other windows.

1.62 Event!.guide/Clockwindows screen to front

```
   Clockwindow's screen to front
```

Bring the screen of the clockwindow to front if it is open.

1.63 Event!.guide/Only Once

```
   Only Once
```

Let this event occur only once while this Event! is running or the event is changed.

1.64 Event!.guide/Do Not Log

```
   Do Not Log
```

Do not write it to the logfile when this event occured.

1.65 Event!.guide/Kill When Done (NOT saved)

```
   Kill When Done (NOT saved)
```

The event will be erased after it was executed. Events with this flag only exist until they occur or Event! is terminated because they are NOT saved when you save the configuration.

1.66 Event!.guide/Display Type

```
   Display Type
```

This can be either "ClockWindow" which means the "Display String" will be displayed in the clockwindow or "Requester" which means a requester will be opened, display "DisplayString" and then waits to be satisfied.

The Meaning of the Display String...

...is predetermined by 'Display Type'.

If 'Display Type' is 'ClockWindow'...

the given string will be displayed in the clockwindow. If the text fits into the clockwindow, it will simply be displayed for 5 seconds (default) - if it does not, it is scrolled. If the clock is turned off when a text is about to be displayed , a window of "clock only" size is opened for the time necessary.

While displaying messages the user interface is still usable, pending events are queued up and executed right after displaying the text is finished.

You can change the values for the displaytime and the scrolling speed, the template is:

```
    [DisplayTime,[ScrollDelay,]]Your Text
```

Examples (do not use the quotes):

```
    "Hello World!!!!"
```

This one displays the string "Hello World!!!!" for 5 seconds if the string

fits in the clockwindow, otherwise it scrolls it with a delay of 1/50 s between scrolling moves. Note that 5 and 1 are the default values for displaytime and speed.

```
"4,0,Hello World!!!!"
```

This one displays the text for 4 secs if it fits in the clockwindow, otherwise the text is scrolled with no (zero) delay.

```
"4,Hello World!!!!"
```

This one also displays it for 4 secs if it fits in the clockwindow, but otherwise the text is scrolled with the 1/50s default delay, since no option was specified for this...

```
"42 is the answer!"
```

This example again displays the text with the default values since there is no comma right after the numbers.

Note: I like a scrolldelay of zero, but on Amigas being faster than A4000/40s this might be *too* fast, besides there's no CPU-time left for other tasks when scrolling with a zero-delay and a high Event! main task priority.

Technical Information: Displaying messages is done asynchronously. If you switch screens/ windowtypes, fonts, or you do anything else that reopens the window the message will be displayed *completely* again. This avoids that you miss a message or parts of it due to screenswitches and/or font changes.

If 'Display Type' is 'Requester'...

...the given text will be displayed in a requester. You can separate multiple lines by using a semicolon, i.e:

```
"It's not a bug,;it's a feature!"
```

1.67 Event!.guide/Command Type

Command Type

Command Type can be either "CLI Command", "ARexx Command" or "Melody".

The meaning of the 'Command String'...

...is predetermined by 'Command Type':

If 'Command Type' is CLI Command

---> it will be executed. Output will be sent to an Amiga OS Release 2 autoclose window, that means the window will be only opened, if the executed command really does output. You can redirect the output by using ">" i.e.

"run >NIL: copy ram:Disk.info sys:storage/ram.inf"

Multiple CLI commands can be separated by ";"

This way it is possible to set stack and current directory by using the "stack" and "cd" commands. Example:

"stack 11111;cd devs:;dir;stack;wait 5"

Note: The wait command prevents the outputwindow to close immediately after the "stack" command is executed.

Hint: The string is not executed asyncronously so you have to preappend "run " if you don't want to delay Event!'s main task. However if you don't "run " commands you will not loose any occuring events, since the event check task marks those events for later execution.

If 'Command Type' is 'ARexx Command'

---> the given string will either be interpreted as an ARexx script or if it is in quotes as an ARexx command.

Default extension for scripts is .evt
Default port for commands is: EVENT!

NOTE:
You've got to enter ARexx commands for Event! very carefully since they are parsed first by ARexx AND then by the Amiga OS2.0 ReadArgs() function. To TEST an event called "Full Hour Beep-2" for example, you should enter 'TEST "Full Hour Beep-2"'. The outside quotes make ARexx execute the instead of trying to load a script from disk. The quotation marks are necessary for ReadArgs() because the event name contains spaces. If the parameter of the "TEST"-command would contain an * or a " you would have to use * as an escape char. For example, instead of 'TEST "Poll*"' you would have to write 'TEST "Poll**"'

If 'Command Type' is 'Melody'

---> a list of notes to play are accepted as parameters, separated by spaces. Each note consists of a period and a duration, separated by a comma. Sounds difficult? Here is an example (don't type in the quotes:)

"381,10 0,1 381,15 339,15 381,15 285,15 302,30"

```
      D    Pause D      E      D      H      F#
```

The duration must be given in ticks (1/50s), the periods are special amiga values. You can look these up in the following table:

```
    +----+------+
    |Note|Period|
    |  C |  428 |
    | C# |  404 |
    |  D |  381 |
    | D# |  360 |
    |  E |  339 |
    |  F |  320 |
    | F# |  302 |
    |  G |  285 |
    | G# |  269 |
    |  A |  254 |
    | A# |  240 |
    |  H |  226 |
    +----+------+
```

Double the value to let the tone sound one octave lower, halve it to do the opposite.

Don't expect too much from this option. This is not a soundtracker clone ;-) Just to remind you of special events. (You probably know the funny little sounds PCs sometimes let out.)

If you want to play real samples, we recommend the use of 'upd' or similar tools. You can perfectly control them via Event!

1.68 Event!.guide/Name

Name

Nearly forgot this one! ;) Defines the name of an event. This name is displayed in the eventlist and can be used to operate on the event via ARexx.

1.69 Event!.guide/Ok

Ok

Checks the changes made in the event edit window and closes it if everything is fine. If a bad hotkey or notification file is given the display will beep and the stringgadget with the bad content will be activated. If Event! is out of memory for allocating the event definition Event! will also beep and keep the window open. When you are that much out of memory you have to shut down other events or you'll have to get rid

off the event edit window with the Cancel/Close gadget.

1.70 Event!.guide/Ok & Prev

Ok & Prev

Same as Ok, but will not close the event edit window, but edit the previous event in the event list.

1.71 Event!.guide/Ok & Next

Ok & Next

Same as Ok, but will not close the event edit window, but edit the next event in the event list.

1.72 Event!.guide/Test it!

Test it!

Execute the action part of the edited event. Also checks hotkey and notification file like "Ok".

1.73 Event!.guide/Cancel

Cancel

Loose changes on the current edited event and close the event edit window. Same as the closegadget.

1.74 Event!.guide/The Event Edit Windows Menu

The Event Edit Window's Menu

"Ok" & "Cancel" work like the gadgets with the same names.

"Snapshot Window" works like Snapshot window from the main configuration window but for the event edit window of course.

1.75 Event!.guide/The Clock Window

```
    The Clock Window

If  this  window  is  activated  it  will display the current date
instead of the current time. The clock window's appearance depends
on the following gadgets of the The Main Configuration Window:

 ClockType
 Follow to Active PubScreen
 Use Screen's Font for Clock
 Specify...

To set the position of the clock window select "Snapshot Window" from
the Clock Window Menu.
```

1.76 Event!.guide/The Clock Window Gadgets

```
   The Clock Window Gadgets


Closegadget

It is invisible, but works.  It is at the top left corner of the clock.
When it is pressed the clockwindow will disappear for a few seconds from
the current screen.  This allows you to close a public screen where the
Event!-clock is on.  This is also useful when intuition attempts to reset
the Workbench.


The Depth Gadget

    ... is also invisible but works.
```

1.77 Event!.guide/The Clock Window Menu

```
   The Clock Window Menu


Snapshot Window

    Snapshot  the  current  clockwindow  position  to the textgadgets.
    The clockwindow will reopen the next time at this position.

Open Configuration Window

    Open the main configuration window.
```

1.78 Controlling Event!’s gadgets with keys

```
   Controlling Event!’s gadgets with keys
```

Most of the gadgets in Event! can be controlled with keyboard shortcuts. These gadgets have an underlined letter in their name. Press this alphanumeric key instead of the gadget. In the CYCLE, the LISTVIEW- and the SLIDER gadgets you can use the SHIFT key together with the key to make a backward selection. The checkboxes in the time definition part of the event edit window can be toggled with the CTRL key and the shortcutkey of the related gadget. Of course you can’t use keyboard shortcuts for disabled gadgets.

Since Event! V1.2 you can additionally select events in the event listview with the cursor up/down keys and edit them with the return key. To close windows you may now use the Escape key.

1.79 Event!.guide/Hotkeys

```
   Hotkeys
```

Hotkeys are descriptions of keyboard sequences and done via Commodities. A hotkey may look like this: (everything is case insensitive)

```
      [<qualifier> [<qualifier>...]] <key>

      Qualifier keywords

        alt           either Alt key
        ralt          right Alt key
        lalt          left Alt key
        shift         either Shift key
        rshift        right Shift key
        lshift        left Shift key
        capslock      Caps Lock key
        rcommand      right Amiga key
        lcommand      left Amiga key
        Control       Control key
        numericpad    Enables the use of a key on the numeric keypad
        rbutton       Click the right mouse button
        midbutton     Click the middle mouse button
        leftbutton    Click the left mouse button
        newprefs      Preferences changed
        diskremoved   Disk removed
        diskinserted  Disk inserted


      Keys

        a .. z, 0 .. 9, etc.   Normal keys
        f1 .. f10              Function keys
        up, down, left, right  Cursor keys
        help                   Help key
        del                    Delete key
```

```
        return                  Return key
        enter                   Enter key (MUST be combined with 'numericpad'!)
        backspace               Backspace key
        esc                     Escape key
        space                   Space key
        comma                   Comma key
        upstroke                Upstroke key


      Examples

        lalt ralt enter
        ramiga f10
        alt x
```

NOTE: Choose your hot keys CAREFULLY, because commodities.library has a high priority in the input events chain (i.e. will override existing definitions).

1.80 The ARexx Interface

```
   The ARexx Interface.
```

Event! has a complex ARexx interface. You can send all the following commands to Event! while it is running. ARexx is also used for the configuration, which is saved in ASCII and can also be modified by the user. When using an ARexx command also look up the description of the corresponding gadget or menu item.

What a surprise, Event!'s ARexx port is named EVENT!. And, yes, this is case sensitive.

Note: It is advised to use "options results" in your AREXX Scripts since some of the commands return resultstrings. Most of the commands also return resultcodes. If no error occured then rc will be zero.

Note also: ARexx-Commands for Event! are first parsed by the ARexx-parser and then by ReadArgs(). For further information see "If 'Command Type' is ARexx" in the Command Type section.

```
 The TellEvent! command

ARexx Commands

 Defining events

 Window Position Commands

 Clock type specification

 Task Priority Commands

 Log File Commands
```

1.81 Event!.guide/The TellEvent! command

The TellEvent! command

The Event! archive contains a command called TellEvent!. It can be used to send commands to Event!'s ARexx port.

Examples:

```
TellEvent! 'BAN 5'

TellEvent! 'QUIT'

TellEvent! 'REQUEST "Just another test!"'

TellEvent! 'BEEP BLINK DISPLAY "2,0,Just another test!"'

TellEvent! 'EVENT "Poll UUCP" ONTIME WHEN "#3 #24 #6 #92 0 3 30"
 RUN "run mail:scripts/polluucp"'
```

Note that you have to write the whole event definition in a SINGLE line!

Note also: ARexx commands for Event! are first parsed by the ARexx parser and then by ReadArgs(). For further information see "If 'Command Type' is ARexx" in the Command Type section.

1.82 Event!.guide/Defining events

Defining events

Example:

```
'EVENT "Poll UUCP"
 ONTIME
 WHEN "#3 #24 #6 #92 0 3 30"
 NOTIFY ""
 HOTKEY "ctrl lshift u"
 DISPLAY "5,0,Polling UUCP-Mail!"
 RUN "run mail:scripts/polluucp"'
```

This will define an event called "Poll UUCP". It will occur every day at 0:03:30h or if its hotkey "ctrl lshift u" is pressed. When running the script "mail:scripts/polluucp" it will display "Polling UUCP-Mail!" in the clock's window.

```
Keywords for defining events:
-----------------------------

General:

 The event's name        : EVENT/K

Condition part:

 Occurance               : NEVER/S ONTIME/S ONNOTIFY/S
 Time condition          : WHEN/K
 File to supervise       : NOTIFY/K
 Notification Tolerance  : NOTIFYTOLERANCE/K/N
 Hotkey for event        : HOTKEY/K

Action part:

 Audible beep            : BEEP/S
 Visible beep            : BLINK/S
 Clock to front          : WINFRONT/S
 Clock's screen to front : SCREENFRONT/S
 Occur only once         : ONCE/S
 Do not log occurance    : DONOTLOG/S
 Erase event when occured: KILLDONE/S
 Display text in clock   : DISPLAY/K
 Put up requester w/text : REQUEST/K
 Run a CLI command       : RUN/K
 Execute an ARexx command: RX/K
 Play specified melody   : MELODY/K
```

1.83 Event!.guide/EVENT/K

```
  EVENT/K

Defines the name of an event.  You must use this keyword to define an
event.  All the following keywords are optional since they will use default
values.
```

1.84 Event!.guide/NEVER/S ONTIME/S ONNOTIFY/S

```
    NEVER/S

The event is to occur never.  Default when defining a new event.

    ONTIME/S

The event is to occur depending on time.

    ONNOTIFY/S
```

The event is to occur depending on file notification.

1.85 Event!.guide/WHEN/K

```
   WHEN/K
```

Defines the time schedule when an ONTIME event occurs.

The template is (same order as in the user interface):

<weekday> <day> <month> <year> <hour> <minute> <second>

If a '#' preceeds a value, it is saved but not actually used. This has the same meaning as an unset checkmark for a time condition in the event edit window.

1.86 Event!.guide/NOTIFY/K

```
   NOTIFY/K
```

Defines the template for file notification.

Will only have an effect if ONNOTIFY/S is set.

1.87 Event!.guide/NOTIFYTOLERANCE/K/N

NOTIFYTOLERANCE/K/N

Specify the Tolerance Secs for the
Notify-argument.

1.88 Event!.guide/HOTKEY/K

```
   HOTKEY/K
```

Defines a hotkey for the starting this event.

1.89 Event!.guide/BEEP/S

```
   BEEP/S
```

Creates an audible beep. Equals "MELODY 400,20".

1.90 Event!.guide/BLINK/S

```
   BLINK/S
```

Flashes all screens.

1.91 Event!.guide/WINFRONT/S

```
   WINFRONT/S
```

Pops the clock window to front each time the event is executed.

1.92 Event!.guide/SCREENFRONT/S

```
   SCREENFRONT/S
```

Pops the screen the clock window is currently on to front each time the event is executed.

1.93 Event!.guide/ONCE/S

```
   ONCE/S
```

Disables but does not delete an event after its first execution.

1.94 Event!.guide/DONOTLOG/S

```
   DONOTLOG/S
```

Do not write logs for this event.

1.95 Event!.guide/KILLDONE/S

```
   KILLDONE/S
```

Deletes an event after its first execution.

1.96 Event!.guide/DISPLAY/K

```
   DISPLAY/K
```

Displays the given text in the clock window when the event occurs. For further information see "If 'Display Type' is 'ClockWindow'" in the Display Type description.

1.97 Event!.guide/REQUEST/K

```
   REQUEST/K
```

Opens a requester with the given contents. For further information see
"If 'Display Type' is 'Requester'" in the Display Type description.

1.98 Event!.guide/RUN/K

```
   RUN/K
```

Runs an AmigaDOS command when the event actually occurs. For further
information see "If 'Command Type' is CLI Command" in the
Command Type description.

1.99 Event!.guide/RX/K

```
   RX/K
```

Starts an ARexx script when the event actually occurs.

For further information see "If 'Command Type' is ARexx" in the Command Type
description.

1.100 Event!.guide/MELODY/K

```
   MELODY/K
```

Plays a melody.

For further information see "If 'Command Type' is 'Melody'" in the
Command Type description.

1.101 Event!.guide/Window Position Commands

```
Window Position Commands

Set the left edge open position for the main window      : MainX/N/K <x-pos>
Set the top  edge open position for the main window      : MainY/N/K <y-pos>
Set the left edge open position for the event edit window: EventX/N/K <x-pos>
Set the top  edge open position for the event edit window: EventY/N/K <y-pos>
Position the clock window at this x-position             : ClockX/N/K <x-pos>
Position the clock window at this y-position             : ClockY/N/K <y-pos>
```

1.102 Event!.guide/MainX/N/K <x-pos>

```
   MainX/N/K <x-pos>

Set the X position of the main window.
```

1.103 Event!.guide/MainY/N/K <y-pos>

```
   MainY/N/K <y-pos>

Set the Y position of the main window.
```

1.104 Event!.guide/EventX/N/K <x-pos>

```
   EventX/N/K <x-pos>

Set the X position of the window used to edit single events.
```

1.105 Event!.guide/EventY/N/K <y-pos>

```
   EventY/N/K <y-pos>

Set the Y position of the window used to edit single events.

Note:  The main and the event edit windows will only be affected by the
positions when you reopen the window.
```

1.106 Event!.guide/ClockX/N/K <xpos>

```
   ClockX/N/K <xpos>

Set the X position of the clock  window.
```

1.107 Event!.guide/ClockY/N/K <ypos>

```
   ClockY/N/K <ypos>

Set the Y position of the clock  window.

Note: The clockwindow will move immediately to the new position.
```

1.108 Event!.guide/Clock type specification

```
     Clock type specification

No clock                              : ClockOff/S
Display clock                         : ClockOn/S
Display clock and available memory    : ClockAndMem/S

Set clock's font to its screen's font: SCREENFONT/S
Set clock's font to specified font    : USERFONT/S
Specify clock's font                  : FONT/K <fontname.font fontheight>

Do not follow to public screens       : DONTFOLLOW/S
Follow to public screens              : FOLLOWSCREENS/S
```

1.109 Event!.guide/ClockOff/S

```
   ClockOff/S

Switch the clock window off.
```

1.110 Event!.guide/ClockOn/S

```
   ClockOn/S

Switch the clock window on if disabled and set it to display the current
time only.
```

1.111 Event!.guide/ClockAndMem/S

```
   ClockAndMem/S

Switch the clock window on if disabled and set it to display the current
time and free fast and chip memory.
```

1.112 Event!.guide/SCREENFONT/S

```
   SCREENFONT/S

Use the screen default font in the clock.
```

1.113 Event!.guide/USERFONT/S

```
   USERFONT/S

Use the user specified font in the clock.
```

1.114 Event!.guide/FONT/K <fontname.font fontheight>

```
   FONT/K <fontname.font fontheight>

Use  the  font  specified in the clock.  May even be proportional.
You will only have visible results if USERFONT is set.

    Example: FONT "topaz.font,9"
```

1.115 Event!.guide/DONTFOLLOW/S

```
   DONTFOLLOW/S

Event! does not follow the active public screen.
```

1.116 Event!.guide/FOLLOWSCREENS/S

```
   FOLLOWSCREENS/S

Event! follows the active public screen.
```

1.117 Event!.guide/Task Priority Commands

```
   Task Priority Commands

Set main  task's priority : MAINPRI/N/K <priority>
Set check task's priority : EventPri/N/K <priority>
```

1.118 Event!.guide/MAINPRI/N/K <priority>

```
   MAINPRI/N/K <priority>

Set the priority of the main Event!  task.  This value is inherited by
tasks spawned by Event!
```

1.119 Event!.guide/EventPri/N/K <priority>

```
   EventPri/N/K <priority>

Set the priority of the Event!  time control task.  This task checks the
current time for events to be executed.
```

1.120 Event!.guide/Log File Commands

```
       Log File Commands

Specify logfile to use: LogFile/K <filename>
Write small logfiles  : TinyLog/S
Write verbose logfiles: BigLog/S
```

1.121 Event!.guide/LogFile/K <filename>

```
   LogFile/K <filename>

Specifies the filename of the logfile.
```

1.122 Event!.guide/TinyLog/S

```
   TinyLog/S

Writes brief log entries.
```

1.123 Event!.guide/BigLog/S

```
   BigLog/S

Writes verbose log entries:  The complete definition of the executed event
is saved into the logfile.
```

1.124 Event!.guide/Operations on events

```
    Operations on events

Erase an event definition  : REMOVE/K <eventname>
Cut an event definition    : CUT/K <eventname>
Copy an event definition   : COPY/K <eventname>
Paste an event definition  : PASTE/K
Execute an event definition: TEST/K <eventname>
```

1.125 Event!.guide/REMOVE/K <eventname>

```
   REMOVE/K <eventname>

Removes a given event.  Wildcards are accepted.  Event!  will return the
number of deleted events as result.
```

1.126 Event!.guide/CUT/K <eventname>

```
   CUT/K <eventname>

Pastes an event to the clipboard.  The event is deleted.  Wildcards
accepted.  Same result as 'Remove/k'
```

1.127 Event!.guide/COPY/K <eventname>

```
   COPY/K <eventname>

Pastes an event to the clipboard.

Wildcards accepted.  Only the last event matching the pattern will stay in
the clipboard.
```

1.128 Event!.guide/PASTE/K

```
   PASTE/K

Pastes an event or ARexx-Command from the clipboard.
```

1.129 Event!.guide/TEST/K <eventname>

```
   TEST/K <eventname>

Test (start once) a given event.  Wildcards accepted.  As result Event!
will return the number of executed events.
```

1.130 Event!.guide/Configuration File Operations

```
       Configuration File Operations

Load a given configuration file  : OPEN/K <filename>
Append a given configuration file: APPEND/k <filename>
Save configuration               : SAVE/k
Save configuration with new name : SAVEAS/k <filename>
```

1.131 Event!.guide/OPEN/K <filename>

```
   OPEN/K <filename>

Loads the given Configuration.

If the configfile can't be located, an ARexx resultcode of 10 will be
returned.
```

1.132 Event!.guide/APPEND/k <filename>

```
   APPEND/k <filename>

Appends the given configuration to the current one.

If the configfile can't be located, an ARexx resultcode of 10 will be
returned.
```

1.133 Event!.guide/SAVE/k

```
   SAVE/k

Saves the current configuration to disk.

If the configfile can't be saved, an ARexx resultcode of 10 will be
returned.
```

1.134 Event!.guide/SAVEAS/k <filename>

```
   SAVEAS/k <filename>

Saves the current configuration to the given filename.

If the configfile can't be saved, an ARexx resultcode of 10 will be
returned.
```

1.135 Event!.guide/Others

```
       Others

Ask for status information           : Status <Nr>

Calculate when an event will occur   : WHENWILL/k <pattern>
Calculate when an event will occur   : WHENWILLNR/k/n <number of event>
Look up if the specified event exists: EXISTS/k <pattern>

Close the clock window for some secs : BAN/N/K <secs>
Quit Event!                          : QUIT/k
```

1.136 Event!.guide/Status <Nr>

```
   Status <Nr>

        --> STATUS 0  returns filename of current config
        --> STATUS 1  returns number of events
```

1.137 Event!.guide/WHENWILL/k <pattern>

```
   WHENWILL/k <pattern>
```

Determines in how many minutes an event or eventpattern will occur first. Returns either a string in the format "<mins to go> <name of event>" or "-1" if the next occurrence is more than 366 days in the future or the event will never occur again. 'On Notification' and 'Never'-occuring events will return "0 <name of event>". If no event with the specified pattern can be found "0" will be returned. If the pattern matches on more than one event the nearest occuring event will be returned.

1.138 Event!.guide/WHENWILLNR/k/n <number of event>

```
   WHENWILLNR/k/n <number of event>
```

Determines in how many minutes the event with the given number in the config will occur. The results are same as for WHENWILL/k <pattern>

1.139 Event!.guide/EXISTS/k <pattern>

```
   EXISTS/k <pattern>
```

Returns the number of events matching the pattern.

1.140 Event!.guide/BAN/N/K <secs>

```
   BAN/N/K <secs>
```

Bans the clock window temporarily from its current screen. You have to specify the number of seconds to disappear.

```
    Example: Ban 3
```

1.141 Event!.guide/QUIT/k

```
   QUIT/k
```

Terminates Event! immediately. No requester will be displayed.

1.142 Event!.guide/Hints

```
   Hints

If you are using the Follow to Public Screens mode:
Define a BAN-Hotkey

Other hints:
Hints and tricks by Martin Kuhne
How to define Hotkeys
```

1.143 Event!.guide/BAN-Hotkey

```
   BAN-Hotkey

If you like to define a hotkey for banning the
Event!-Clock from its current screen, you'll have to set up an event which
addresses Event!s ARexx-Port with a BAN/N/K <secs> command.  Set "occur" to
"never".  A sample event for this is included in the sample configuration
file.
```

1.144 Event!.guide/For application programmers

```
   For application programmers

If you're programmer of an application which opens a Public Screen, it is
advised that you close all your windows on the screen and then wait up to
1.5 secs for the last visitor window to close before actually closing the
screen or displaying an error message.  That's 'cause Event!'s clock window
disappears off the screen in FOLLOWSCREENS mode when the last window of a
screen is deactivated.  You may also try to send a "BAN" command to
Event!'s ARexx port.  This makes working with the FOLLOWSCREENS mode of
Event!  much more convenient for the user!
```

1.145 Event!.guide/UsersNote

```
                  ------------------------------------------
                          Event! Release version 1.0

                       Notes from a user and beta tester

                               December 28th, 1992

                                 by Martin Kuhne
                  ------------------------------------------


  1. What the heck is this?
```

This is a little collection of notes, tips, tricks and examples that some Event! users might find interesting.

2. Notification

2.1. Problems

You will find out that especially when notification on a directory is active, events will be triggered more than once for one operation.

This happens because some applications change more than one file for one operation. For example. if you have "safe saves" active in CygnusED, the editor writes a temporary file, then deletes the original and renames the temporary file. Or take TrapDoor... when receiving a file, a file Traptemp#? is first created, then renamed when received completely.

So you will encounter situations where notification is less useful than expected. If you spawn a batchfile, make sure it does not run more than once at a time, like this aftersession script:

```
failat 21
if NOT $aftersession EQ true
    setenv aftersession true
    [...]
endif
setenv aftersession false
endcli
```

(you'll find more examples in the TrapDoor release archive)

2.2. Dealing with IPrefs

IPrefs is a smart program lurking in the background waiting if some of the prefs files are changed.

If you change e.g. the screenmode, the notorious "Intuition is attempting to..." requester appears, asking you to close all windows except drawers.

Be smarter! Let Event! wait in the background for you, closely watching env:sys/. As soon as AmigaDOS reports changes in that directory, the clock window suddenly vanishes just to reappear a few seconds later. All it takes is a 'BAN 5' (or similar) ARexx command.

Note: Talking to oneself is generally considered a bad habit. A program talking to its own ARexx port is of course completely different :-) What is more, there are no lockups!

If you are really clever, you can also start a "real" ARexx script that attempts to close other applications. There exists, for example, a program by Olaf Barthel that closes all open shell windows. (I forgot the name, I'm sorry).

3. Cloning events

One thing you can't do as easily with Event! compared to a standard cron type program is quickly configuring an event that its toggled every five minutes for the period of one hour of the day.

You can, of course, clone the event several times and change the time. But if you want to change the event, you'll have to change all copies.

Here's a workaround:

Clone the original event once, and select "ARexx" as action. Then "TEST" the original event....

4. Time for tea?

Imagine you have a BBS running between 22:00h and 06:00h. Imagine you want to have raytracing pictures in the workbench background, and beautiful young ladies at night ?

So you set up several events to switch. What happens if your machine crashes ? The changes are lost... this is one way to do it: First, add an event named "Midnight" that occurs at 00:00h (did you guess it ? :) every day. Then look at this script how to find out about a certain period of the day...

```
/* example script */

options results
address "EVENT!"

EXISTS Midnight
if result==0 then do
   say "WARNING: No event called 'Midnight' exists"
end

WHENWILL Midnight
parse var result OccurMins foo

if OccurMins~==0 then do
    if OccurMins<= 2*60 | OccurMins >= 18*60 then do
        say "It's later than 22:00h or earlier than 06:00h"
        /* start bbs, change configs, etc. */
    end
end

/* example script ends */
```

Again, don't forget that you can toggle events that have already been executed with the TEST command.

5. Acknowledgements

The following names are either trademarks or the efforts of the person and/or company listed:

- Fido and FidoNet are trademarks of Tom Jennings, Fido Software
- TrapDoor by Maximilian Hantsch and Martin Laubach
- CygnusED by CygnusSoft Software

1.146 Event!.guide/Acknowledgements

Acknowledgements

Finally I want to thank the very first betatesters:

Manfred Büsing	For the initial inspiration, good ideas and parts of the documentation.
Martin Kuhne	For debugging Event! with Enforcer, good ideas and and parts of the documentation.
Christian Buchner	For BaudBandit.device, additional debugging, and good ideas.
Alexander Görig	For continuous Enforcer-testing.

Additional thanks to:

Kenneth Fribert	For the Danish translation.

All the people in the Munich Amiga eXchange Net (MAX) who supported me with ideas and hints on programming the Amiga.

1.147 Event!.guide/Registration Form

Dear Stefan,

I want to register the Event! package. I have evaluated this software package and I think that it is worth registering. This is the address to which you should send the keyfile as soon as you receive the money:

Name: _____

Address: _____

City: _____

Country: _____

Telephone Number: _____

The following entries can be filled in optionally
and will be used only for my personal statistics:

Computer: _____

Memory: _____

Harddisk Capacity: _____

Processor: _____

Operating System: _____

AmigaNet Address: _____

FidoNet Address: _____

Z-Net Address: _____

EMAIL-Address: _____

|__| A cheque over DM 30 is included. (Delivery
might take a while)

|__| The payment is enclosed cash.

|__| I let my bank transfer the money to your
bank account:

Bank: Raiffeisenbank Feldkirchen bei München
BLZ: 70169364
Account: 403849
Owner: Stefan Hochmuth

I will receive a keyfile for Event! V1.2 that will stop
it from displaying the shareware reminder.
I know that I am not allowed to copy my Event!
keyfile to anyone else.

Following these rules my Event! keyfile will also
work with any future releases of Event!.

I know that my personal data is stored and processed
electronically as well as the transfer of the money
is at my own risk if I send cash or a cheque.

_____ _____
Place, Date Signature

1.148 Event!.guide/How to reach the author

How to reach the author

If you have any questions about Event! you can reach me in AmigaNet. Just write a matrixmail to Stefan Hochmuth at 39:171/201.2. Event! was really lots of work. There are many features I made not for me but for you! So if you want to use Event! regularly please register.

Send ideas, registrations and bugreports to:

Stefan Hochmuth
Flussaalweg 9
81825 München
Deutschland

==========
Keep the shareware concept alive!
==========

1.149 Event!.guide/Program History

```
Version 1.00
------------

- first public release

Version 1.0a
------------

- fixed harmless Enforcer-Hit at $20 when reading the configuration
  or executing ARexx-commands
- fixed bug with causing sometimes the edit window either not
  to open or causing weird text attributes (everything bold or
  underlined). This bug could also cause program failures, but
  only when a font fallback was necessary.
- did some cleanup of the documentation
- Why are bugs found right after the release of a program? ;-/

Version 1.0b
------------

- If you've got more than 10,000 KB (10 MB) FastMemory Event!
  wouldn't erase the leading digit in the clock's memory display when
  the free memory dropped under the 10 MB limit. This resulted in
  a 10 MB too high free memory display. :-) Thanx to Gerhard Müller
  for reporting. Fixed.

- Small bug: If you selected "Edit..." in the main window while the
  event-edit window was already open, the event-edit window would
```

not allow you to edit the selected event as described in the documentation. Instead it would simply close. Fixed.

- included ARexx-scripts for use of Event! with Spot 1.2b registered or higher

- did some cleanup of the documentation

Version 1.1

- Small Bug: the terminating zero-byte was copied to the clipboard when cutting/copying events. Fixed.

- ASCII-Text documentation replaced by AmigaGuide documentation

- Installer Script included in distribution

- Configuration Save - Routine will now only save keywords as necessary

- Tolerance Secs implemented.

- Event! is now localized (German catalog included)

- Bug in the waitroutine fixed (could occur very seldom when the system was quite overloaded)

- Small bug: The Cut/Copy/Remove menu items weren't disabled when the config window had just been reopened (and thus no event had been selected from the list). Fixed.

Version 1.1a

- internal release

Version 1.2

- small layout changes give more space for translations

- Danish catalog included. Thanks to Kenneth Fribert for the translation!

- Bug: The name of an event in the logfile entry and some keywords of the config could get trashed when running certain patches for dos.library due to a non-preserved-register bug. Fixed.

- maximum search period for ARexx-commands WHENWILL/k <pattern> and WHENWILLNR/k/n <number of event> increased from 28 to 366 days. (User request)

- Performance of WHENWILL/k <pattern> and WHENWILLNR/k/n <number of event> greatly enhanced

- Bug: Enforcer hit(s) could occur when duplicating certain events. Fixed.

- Bug: 3 enforcer hits occured the first time you changed a check
  gadget in a time definition of an event. Thanx to Finn Kettner for reporting
  the enforcer hits!

- Top/Up and Down/Bottom gadgets are now disabled if pressing them would have
  no effect

- When pressing the Help-Key the AmigaGuide-documentation pops now up
  asynchronously

- New ToolType and CLI-Parameter 'GUIDE' specifies the path to the
  online help AmigaGuide-file

- Enhanced keyboard control for the listview:
  - CURSOR UP/DOWN to select an event definition
  - ENTER/RETURN to edit an event definition

- ESC has now the same function as the window closegadget

- On machines with OS 3.0 or higher Event! now uses intuition's
  busy-pointer instead of its own

- German catalog updated

<<<<<<<<<<<<<<<<<< fascism IS bad - log it out! >>>>>>>>>>>>>>>>>>