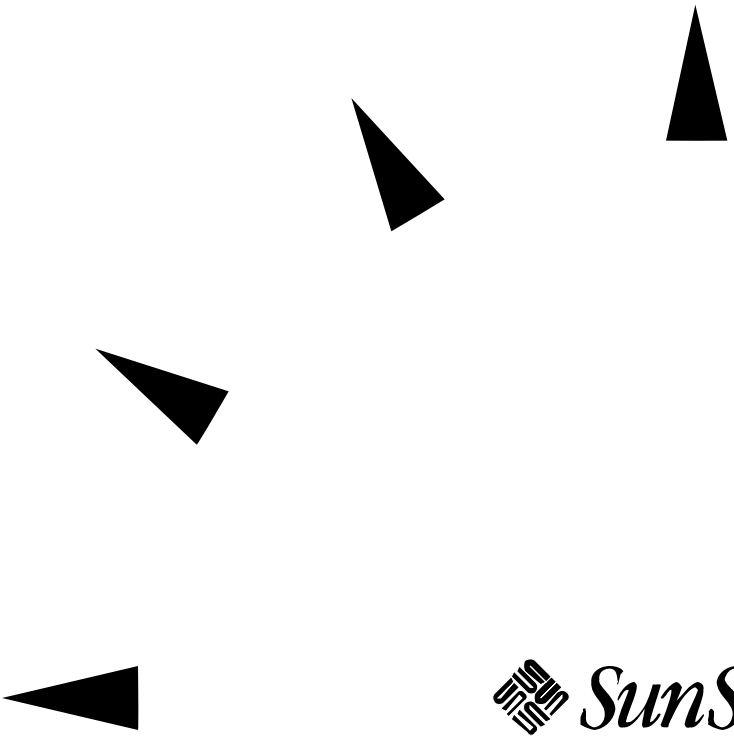
SunOS Reference Manual



Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

© 1994 Sun Microsystems, Inc. 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX® system, licensed from UNIX System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and from the Berkeley 4.3 BSD system, licensed from the University of California. Third-party software, including font technology in this product, is protected by copyright and licensed from Sun's suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS
Sun, Sun Microsystems, the Sun logo, SunSoft, the SunSoft logo, Solaris, SunOS, OpenWindows, DeskSet, ONC, ONC+, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. OPEN LOOK is a registered trademark of Novell, Inc. PostScript and Display PostScript are trademarks of Adobe Systems, Inc.

All SPARC trademarks are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. SPARCcenter, SPARCcluster, SPARCompiler, SPARCdesign, SPARC811, SPARCengine, SPARCprinter, SPARCserver, SPARCstation, SPARCstorage, SPARCworks, microSPARC, microSPARC-II, and UltraSPARC are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK® and Sun™ Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

X Window System is a trademark of the X Consortium.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN. THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAMS(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Portions © AT&T 1983-1990 and reproduced with permission from AT&T.

Preface

OVERVIEW

A man page is provided for both the naive user, and sophisticated user who is familiar with the SunOS operating system and is in need of on-line information. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

The following contains a brief description of each section in the man pages and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.

- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.

- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.

- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2 of this volume.

- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.

- Section 5 contains miscellaneous documentation such as character set tables, etc.

- Section 6 contains available games and demos.

- Section 7 describes various special files that refer to specific hardware peripherals, and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.

- Section 9 provides reference information needed to write device drivers in the kernel operating systems environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver–Kernel Interface (DKI).

- Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer may include in a device driver.

- Section 9F describes the kernel functions available for use by device drivers.

- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the intro pages for more information and detail about each section, and **man**(1) for more information about man pages in general.

NAME

This section gives the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS

This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full pathname is shown. Literal characters (commands and options) are in **bold** font and variables (arguments, parameters and substitution characters) are in *italic* font. Options and

arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

[ ] The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument *must* be specified.

... Ellipses. Several values may be provided for the previous argument, or the previous argument can be specified multiple times, for example, '*filename* ...'.

| Separator. Only one of the arguments separated by this character can be specified at time.

{ } Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

PROTOCOL

This section occurs only in subsection 3R to indicate the protocol description file. The protocol specification pathname is always listed in **bold** font.

AVAILABILITY

This section briefly states any limitations on the availabilty of the command. These limitations could be hardware or software specific.

A specification of a class of hardware platform, such as **x86** or **SPARC**, denotes that the command or interface is applicable for the hardware platform specified.

In Section 1 and Section 1M, **AVAILABILITY** indicates which package contains the command being described on the manual page. In order to use the command, the specified package must have been installed with the operating system. If the package was not installed, see **pkgadd**(1) for information on how to upgrade.

MT-LEVEL

This section lists the **MT-LEVEL** of the library functions described in the Section 3 manual pages. The **MT-LEVEL** defines the libraries' ability to support threads. See **Intro**(3) for more information.

DESCRIPTION

This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, functions and such, are described under USAGE.

IOCTL

This section appears on pages in Section 7 only. Only the device class which supplies appropriate parameters to the **ioctl**(2) system call is called **ioctl** and generates its own heading. **ioctl** calls for a specific device are listed alphabetically (on the man page for that specific device). **ioctl** calls are used for a particular class of devices all of which have an **io** ending, such as **mtio**(7).

OPTIONS

This lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.

OPERANDS

This section lists the command operands and describes how they affect the actions of the command.

OUTPUT

This section describes the output - standard output, standard error, or output files - generated by the command.

RETURN VALUES

If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared as **void** do not return values, so they are not discussed in RETURN VALUES.

ERRORS

On failure, most functions place an error code in the global variable **errno** indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.

USAGE

This section is provided as a *guidance* on use. This section lists special rules, features and commands that require in-depth explanations. The subsections listed below are used to explain built-in functionality:

Commands
Modifiers
Variables
Expressions
Input Grammar

EXAMPLES

This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command line entry and machine response is shown. Whenever an example is given, the prompt is shown as

example%

or if the user must be super-user,

example#

Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS and USAGE sections.

ENVIRONMENT

This section lists any environment variables that the command or function affects, followed by a brief description of the effect.

EXIT STATUS

This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion and values greater than zero for various error conditions.

FILES

This section lists all filenames referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.

SEE ALSO

This section lists references to other man pages, in-house documentation and outside publications.

DIAGNOSTICS

This section lists diagnostic messages with a brief explanation of the condition causing the error. Messages appear in **bold** font with the exception of variables, which are in *italic* font.

WARNINGS

This section lists warnings about special conditions which could seriously affect your working conditions — this is not a list of diagnostics.

NOTES

This section lists additional information that does not belong anywhere else on the page. It takes the form of an *aside* to the user, covering points of special interest. Critical information is never covered here.

BUGS

This section describes known bugs and wherever possible suggests workarounds.

NAME

Intro, intro – introduction to games and demos

DESCRIPTION

This section describes available games and demos.

Name	Description
gaintool(6)	audio control panel
radio(6)	Radio Free Ethernet receiver
radio_recv(6)	radio receive utility
radio_xmit(6)	radio broadcast utility
soundtool(6)	audio waveform display demo
x_buttontest(6)	Xview demonstration and test program for SunButtons
x_dialtest(6)	Xview demonstration and test program for SunDials
xmit(6)	Radio Free Ethernet transmitter

NAME

gaintool – audio control panel

SYNOPSIS

/usr/demo/SOUND/bin/gaintool [**–d** *device*]

DISCLAIMER

This program is furnished on an AS IS basis as a demonstration of audio applications programming.

DESCRIPTION

gaintool is an XView demonstration program that controls various characteristics of the system audio device. By default, **gaintool** operates on **/dev/audio**, though an alternative audio device may be specified by using the **–d** option. Operations performed by the **gaintool** control panel affect all audio programs using the specified device; for instance, adjusting the **Play Volume** instantly changes the output gain, regardless of which program is playing audio data. Further, **gaintool** detects audio state changes made by other programs, and updates its display accordingly, so that it keeps in sync with the current device configuration.

gaintool demonstrates an important principle involved in the integration of audio in the desktop environment: by enabling global control of important audio characteristics, it is not necessary for every application to provide an interface for these parameters. For instance, since audio output may be paused from the control panel, it is not strictly necessary that output applications display a **Pause** button of their own. However, such applications may detect that audio output has been paused, and take appropriate action.

Control Panel

Play Volume

This slider adjusts the output volume. Volume levels between **0** and **100** may be selected, where **0** represents infinite attenuation and **100** is maximum gain.

Record Volume

This slider adjusts the recording gain level in the range **0** to **100**.

Monitor Volume

Monitor gain controls the amount of audio input signal that is fed through to the output port. For instance, if an audio source (such as a radio or CD-player) is connected directly to the input port, the input signal may be monitored through either the built-in speaker or the headphone jack by adjusting this slider. Note that there may be audible feedback (a high-pitched whine) if a microphone is connected to the workstation and the monitor volume is set greater than zero.

Output

This selector switches the audio output port between the built-in speaker and the external headphone jack.

Pause Play

This button may be used to suspend and resume audio output. If audio output is in progress when **Pause** is pressed, it is stopped immediately and subsequent output data remains queued. The button then switches to a **Resume** button that, when pressed, resumes audio output at the point that it was suspended.

If no process has the device open for output when **Pause** is pressed, **gaintool** holds the device open itself, thereby denying other processes output access. Audio programs that simply open and write to the audio device will typically be suspended when they attempt to open the device. Programs that asynchronously poll the device will discover that it is ''busy'' and may take appropriate action.

Audio Device Status Panel

gaintool also includes an audio status panel that shows the current state of the audio device. This panel is extremely useful for debugging audio applications. Selecting ''Status...'' from the panel menu brings up the status panel (this can also be done on a SPARC system by pressing the **PROPS** (L3) key). Selecting ''Done'' from the panel menu removes the panel (this can also be done on a SPARC system by pressing the **OPEN** (L7) key).

When the **Update Mode** is set to **Status Change**, the audio device status is updated only when a **SIGPOLL** signal is delivered to **gaintool** (see **audio**(7I) on a SPARC system and **sbpro**(7D) on an x86 system). Because of this, the **Active** and **Samples** indicators are not necessarily kept up-to-date. This mode is useful for application debugging in order to see exactly when audio device status changes are being reported. When the **Continuous** mode is selected, the status is continually updated.

SEE ALSO

soundtool(6), **audio**(7I)

See **audiotool** manual page in *OpenWindows Desktop Reference Manual*.

SPARC Only

audioamd(7D), **dbri**(7D)

x86 Only

sbpro(7D)

NAME

radio – Radio Free Ethernet receiver

SYNOPSIS

radio [–**h** *host*] [–**s** *service*] [*generic-tool-arguments*]

DISCLAIMER

This program is furnished on an AS IS basis as a demonstration of audio applications programming.

OVERVIEW

Radio Free Ethernet (RFE) is a network audio broadcasting system. It consists of programs and tools that allow packets of audio data to be transmitted around a network. The system is best understood by using the analogy of traditional radio broadcasting.

A radio station takes audio data from a variety of sources (e.g., CDs, tapes, a microphone in front of a disc jockey, the telephone, etc.) and broadcasts it in the atmosphere. Similarly, a workstation may take audio data from an audio file or input device (e.g., a microphone or CD plugged into the audio cable of a workstation) and broadcast it over the local area network. Each such workstation becomes a radio station, and must broadcast on a distinct 'frequency' in order to be differentiated from other active radio stations.

Individual radios function as receivers for radio broadcasts. To listen to a radio station, you must tune your radio to its frequency. At that point, the broadcast data is converted to an audio signal and may be played through a speaker. In the same fashion, a workstation may scan the network for RFE broadcasts and 'tune in' a particular station by routing its audio data to the audio output device (e.g., the speaker). Multiple stations may broadcast on the same network; it is up to the receiver program to select the radio station of interest and discard any other broadcast data.

As with conventional radios, a receiver may be requested to scan the network and tune in to the next detected radio station. One of the advantages of the computer implementation is the ability to display a list of all the active radio stations and allow the user to select from among them.

In traditional radio, a regulation agency allocates broadcast frequencies, and associates a station identification name with each frequency. In Radio Free Ethernet, the broadcast frequency is directly derived from the station call letters; that is, the ASCII code for a four-character station name becomes a 32-bit broadcast frequency. Each network radio packet contains a 12-byte header that identifies the frequency, packet type, sequence number, and the low order 3 bytes of the broadcaster's network id. This information is used to differentiate radio broadcasts and to detect multiple stations (workstations) that may be broadcasting on the same frequency (call letters). At the network level, each packet also contains a port number corresponding to the Radio Free Ethernet port. Ordinarily, this port is identified by looking up the service named **radio** in the NIS *services* map.

In addition to its audio data packets, each station transmits a Station Identification packet once every five seconds. This packet contains detailed information about the station and disc-jockey (i.e., the host and user ids). Station Identification is also used to recognize stations that are On-The-Air, but are not currently broadcasting audio data.

This can occur when a transmitter detects silent audio input and squelches it (see **radio_xmit**(6)). For instance, if a user is broadcasting from a microphone, the station may be squelched when the microphone is switched off.

When a radio station goes Off-The-Air, it broadcasts a final sign-off message to alert the receiver programs that broadcasting has ceased.

One of the pitfalls of the radio analogy is that users seem to expect that they can always tune in some radio station when they start up a receiver program. *Do not be deceived.* If nobody is broadcasting, there are no stations to tune in. As with radio in the early 20th century, the usefulness of the radio is only as good as the quantity (and quality) of broadcasters.

Another common mistake is to assume that traditional radio broadcasts will somehow be available on the network. Keep in mind that the transmitter only broadcasts data from its audio input source. If you connect the line-level or headphone output of a radio to the workstation audio input, then in fact you can broadcast traditional radio over the network. (Interestingly, the 8khz data rate provides roughly the same audio quality as AM radio.) But if you connect no audio input source, then you will not broadcast anything.

INSTALLATION

In order for Radio Free Ethernet to function properly, the following entry should be present in the NIS *services* map:

radio 5002/udp # Radio Free Ethernet

This entry specifies the port number that is used to identify RFE broadcast data. If this entry is not present, the port number may be specified as a command-line option.

NETWORK IMPLEMENTATION

Radio Free Ethernet can be configured to broadcast data either in UDP Broadcast packets or using IP Multicasting. These techniques differ in subtle but important ways.

UDP Broadcast packets are broadcast only within the local subnetwork. Network gateway routers do not forward these packets to other networks. When a UDP Broadcast packet is issued, every machine on the subnet receives the packet and discards it, unless a program is specifically registered to listen for that particular packet type. Though the overhead of processing such packets is small, it is normally considered unfriendly to issue many UDP Broadcast packets on a network (RFE normally broadcasts approximately eight packets per second, each containing around 1000 bytes). UDP Broadcast is available for the time being only because older versions of the operating system do not support IP Multicast.

IP Multicasting is an improvement over broadcast techniques. By sending network packets to a particular well-known multicast address, only machines that have registered interest in that address will receive the data (packet filtering is usually performed in the network interface hardware). Some experimental IP routers exist that will forward multicast packets to other networks. Such forwarding is only performed when there is a listener on the destination network, and when the packet itself is identified as forwardable.

In order for Radio Free Ethernet to function properly, the following entry should be present in the NIS *hosts* map:

RadioFreeEthernet 224.0.3.255 # IP Multicast address

This entry specifies the base address used for RFE station broadcasts. For more information on how IP Multicasting is used, please refer to the **radio_xmit**(6) manual page.

CONFERENCING

The Radio Free Ethernet tools have been designed to provide a limited kind of audio conferencing capability. Since a workstation may broadcast on one frequency while receiving a different one, clever assignment of radio stations can be constructed to implement 2-way conferencing. Multi-party conferencing may be enabled by modifying the radio receiver to sum the audio data from multiple radio stations together, thus achieving a simple audio mix.

DESCRIPTION

radio is the window-based Radio Free Ethernet receiver. It functions as a graphical front-end to the **radio_recv**(6) program, which it uses to scan the network for active radio stations and to play audio data from the network on the audio device. It also provides six preset buttons, similar to the presets on a car radio, that may be programmed to specific radio station names.

Ordinarily, the program scans for radio stations that are broadcasting to the IP Multicast address identified by the host name *RadioFreeEthernet* found in the NIS *hosts* map (stations using UDP Broadcast are also received). The –**h** *host* command-line option may be used to specify an alternate host address or name to use for the default IP Multicast address.

The Radio Free Ethernet tools use the port number identified by the service name *radio* found in the NIS *services* map. The –**s** *service* command-line option may be used to specify an alternate service name or port number to use.

The following sections describe the individual panel controls. In addition, online help is available by positioning the pointer over the item in question and pressing the **<Help>** key (<F1> on Type-3 keyboards).

Power toggles the state of the receiver on and off. Since this involves communicating with a running copy of the **radio_recv** program, this operation may take a few seconds to complete.

Scan initiates a scan for the next active radio station. By pressing this button repeatedly, you can browse all of the active stations.

Station brings up a menu of the radio stations that have been detected. Selecting a station from this menu will tune in that station. Station names are enclosed in square brackets if the station is determined to be on-the-air but has not broadcast any data for some time. This can occur when the transmitter squelches silent audio input.

DJ displays the username and hostname of the disc jockey for the current radio station.

Set establishes a station preset button. It uses the name in the **Preset Station** item or, if that is blank, the name of the current station. Once a preset is programmed, pressing the button forces the radio tuner to scan for that particular station. **Set** also causes the current program parameters and presets to be written out to the initialization file.

Clear clears a station preset button. It uses the name in the **Preset Station** item or, if that is blank, the name of the current station. **Clear** also causes the current program parameters and presets to be written out to the initialization file.

Volume starts up an Audio Control Panel (see **gaintool**(6)).

Props brings up a property sheet for the program that allows various operating parameters to be set. Changes to the tool properties do not take effect until the **Apply** button is pressed. **Apply** also causes the current program parameters and presets to be written out to the initialization file. The **Reset** button on the property sheet restores the controls to their current state.

The following sections describe the property sheet controls:

Auto Scan When this feature is enabled, the radio tuner will scan for a new radio station when no broadcast has been received from the current station for a certain amount of time (specified, in seconds, by a numeric value).

Allow Device Release
When this feature is enabled, the radio tuner will relinquish the audio output device whenever another process tries to use it. This allows other applications, such as AudioTool, the opportunity to interrupt the playing of a radio station in order to play what is presumed to be more important audio data.

Audio Output This property sheet item indicates the audio output device that is used to play a radio broadcast.

ERRORS

If **radio** (or **radio_recv**) are run more than once on a single workstation, subsequent invocations may display the error message: Radio receiver socket is busy. This is because there may only be one program listening to a particular network port at a time.

FILES

~/.radiorc startup initialization file for **radio** and **xmit**

SEE ALSO

gaintool(6), **radio_recv**(6), **radio_xmit**(6), **xmit**(6)

RFC 1256, Network Information Center, SRI International, Menlo Park, Calif., September 1991.

HISTORICAL NOTE

The Radio Free Ethernet tools were inspired by similar programs originally developed by Stephen Uhler and Peter Langston at Bellcore.

NAME

radio_recv – radio receive utility

SYNOPSIS

radio_recv [**–C**] [*OPTION=value*] ...

DISCLAIMER

This program is furnished on an AS IS basis as a demonstration of audio applications programming.

DESCRIPTION

radio_recv is the Radio Free Ethernet (RFE) receiver. (For an overview of Radio Free Ethernet, refer to the **radio**(6) manual page.) Though **radio_recv** is normally initiated directly by the window-based **radio**(6) program, it may also be used directly as a command-line program, or in conjunction with a user-written window-based tool.

In Normal Mode (the default case), **radio_recv** is initiated with a set of receiver options specified on the command line. The program immediately begins scanning the network for radio broadcasts according to the currently selected options. The program will terminate on receipt of a SIGINT (control-C). Once the receiver tunes in a broadcasting station, the program will exit when the station signs off.

If **–C** is specified on the command line, **radio_recv** operates in Command Mode. In this mode, the program reads and processes command options from *stdin* and writes error and status information to *stderr*. Window-based tools use Command Mode to control the receiver program and receive status information for display. When operating in Command Mode, **radio_recv** requires that an output device be specified explicitly, and the START command must be issued to initiate radio reception.

COMMAND OPTIONS

The receiver options are described below. If an option requires a value, the option name must be followed by an equal sign (=) and the value. If an option takes a ON / OFF value, the default is ON if the value is missing or poorly constructed.

Station=[*call letters*]

This option specifies the station call letters of a radio station to tune in. A maximum of four characters may be specified. If the receiver is enabled without specifying a station, the receiver will tune in the first station that is transmitting audio data.

Output=[*audio device name*]

This option specifies the audio output device. In Normal Mode, the audio output defaults to **/dev/audio**.

Release=[*ON/OFF*]

When the receiver tunes in a radio station, it assumes exclusive control of the audio output device. If another process attempts to claim the audio device for output, it will fail (or suspend until the device is released). In such cases, the **radio_recv** program may be notified that another process is competing for the audio output channel.

If the Release option is enabled, the receiver will automatically release the audio device when such notification occurs, allowing other processes the opportunity to claim the device. As long as the device remains unavailable, the receiver will

throw away any radio broadcast packets that are detected. When the device is available again, the receiver will reclaim it and continue playing the incoming audio data. If the Release option is disabled, **radio_recv** will maintain exclusive control of the audio device until it is explicitly stopped or the incoming radio station signs off.

The Release option is enabled by default.

Address=[*hostname* | *address*]

This option is used to configure the IP Multicast address that the **radio_recv** program uses to locate radio broadcasts. The value may either be a hostname (which is translated into an IP Multicast address by looking up the name in the NIS *hosts* map) or a specific numeric IP Multicast address. The special hostname *BROADCAST* may be used to force the program to scan only for UDP Broadcast packets.

The default IP Multicast address is designated by the hostname *RadioFreeEthernet*. The **radio**(6) manual page contains more information on the IP Multicast implementation.

Service=[*service* | *port number*]

This option is used to configure the IP protocol port number used to identify Radio Free Ethernet network packets. The value may be either a service name (which is translated into a port number by looking up the name in the NIS *services* map) or a specific numeric port number.

The default port number is designated by the service name *radio*. The **radio**(6) manual page contains more information on configuring the port number.

Report=[*ON/OFF*]

This option enables the reporting, to *stderr*, of all changes in the receiver status. This option also enables the reporting of the active station list. The station list details the state of all active radio stations, and is output whenever a state change is observed. The Report option is normally used only in conjunction with Command Mode to allow the controlling program to maintain state.

Scan

This command is used to initiate scanning for a new radio station. The receiver tunes in to the next radio station whose broadcast is detected. When multiple stations are actively broadcasting, the scan cycles through them all before repeating a station. Note that stations whose broadcast is being squelched due to a silent input signal are not tuned in.

Start

This command is used to initiate radio reception according to the current parameter settings. It is provided for Command Mode; the receiver is enabled by default in Normal Mode.

Stop

This command is used to turn off the receiver. Queued audio data is flushed so that audio output stops immediately. The Stop command is provided for the Command Mode operation.

Quit

This command causes the **radio_recv** program to exit. It is provided for the Command Mode operation.

SEE ALSO **radio**(6), **radio_xmit**(6), **xmit**(6)

NAME

radio_xmit – radio broadcast utility

SYNOPSIS

radio_xmit [–**C**] [*OPTION=value*] ...

DISCLAIMER

This program is furnished on an AS IS basis as a demonstration of audio applications programming.

DESCRIPTION

radio_xmit is the Radio Free Ethernet (RFE) broadcast utility. (For an overview of Radio Free Ethernet, refer to the **radio**(6) manual page.) Though **radio_xmit** is normally initiated directly by the window-based **xmit**(6) program, it may also be used directly as a command-line program, or in conjunction with a user-written window-based tool.

In Normal Mode (the default case), **radio_xmit** is initiated with a set of broadcast options specified on the command line. The program immediately begins transmitting audio data over the network according to the currently selected options. The program will terminate on receipt of a SIGINT (control-C). If an input file is specified, the program will exit when the entire file has been broadcast.

If –**C** is specified on the command line, **radio_xmit** operates in Command Mode. In this mode, the program reads and processes command options from *stdin* and writes error and status information to *stderr*. Window-based tools use Command Mode to control the broadcast program and receive status information for display. When operating in Command Mode, **radio_xmit** requires that a station name and input file be specified explicitly, and the START command must be issued to initiate broadcasting.

COMMAND OPTIONS

The broadcast options are described below. If an option requires a value, the option name must be followed by an equal sign (=) and the value. If an option takes a ON / OFF value, the default is ON if the value is missing or poorly constructed.

Station=[*call letters*]

This option specifies the station call letters of your radio station. A maximum of four characters may be specified. In Normal Mode, the station name defaults to the first four letters of the current workstation hostname. However, it is preferable to supply a Station name on the command line.

Input=[*audio file or device name*]

This option specifies the audio input source for radio broadcast. If a file is specified, it must be a legitimate audio file (i.e., with a standard Sun audio file header), sampled at 8 kHz. In Normal Mode, the audio input defaults to **/dev/audio**.

Squelch=[*ON/OFF*]

This option controls the behavior of the transmitter when the audio input data is determined to be silent. When Squelch is enabled, silent data is not transmitted over the network. This reduces the network traffic for stations that are broadcasting with no audio input. Note that the Station Identification packet continues to be transmitted every 5 seconds, even if the data broadcast is squelched.

When Squelch is disabled, all audio input data is always transmitted. Squelching only occurs for audio device input; audio files are not squelched, even if they contain silence. The Squelch option is enabled by default.

Format=[*COMPRESSED/UNCOMPRESSED*]

This option controls the format for audio data that is broadcast. The uncompressed format broadcasts 8000 bytes of audio data each second. The compressed format sends only 4000 bytes per second, but requires more computation on both the transmitter and receiver.

Data format conversions only occur for audio device input; audio files are transmitted in the format in which they were stored. By default, audio data is broadcast uncompressed.

Agc=[*ON/OFF*]

This option controls the behavior of an Automatic Gain Control feature. When Agc is enabled, the audio recording volume is adjusted automatically to keep the input signal within reasonable bounds. If the input signal is too loud, the record volume is decreased. If the input signal is determined to be present but too soft, the record volume is increased.

Automatic Gain Control adjustments only occur for audio device input; audio files are broadcasted at the level at which they were recorded. The Agc option is enabled by default.

Autostop=[*ON/OFF*]

This option controls the behavior of the transmitter when the audio input data has been silent for a length of time. When Autostop is enabled, the transmitter will sign off the station and stop broadcasting if the audio input has been silent for 60 seconds. When Autostop is disabled, the station continues to remain active. This option does not interact with the Squelch option, although the same criteria are used for determining whether or not the audio input is silent.

Autostop is only processed for audio device input; broadcasting always stops when an end-of-file has been reached on audio files. The Autostop option is disabled by default.

Address=[*hostname* | *address*]

This option is used to configure the IP Multicast base address that the **radio_xmit** program uses for broadcasting data. The value may either be a hostname (which is translated into an IP Multicast address by looking up the name in the NIS *hosts* map) or a specific numeric IP Multicast address.

Normally, the address should end in 0 or 255, in which case an *id-address* and *data-address* are constructed as follows: the *id-address* consists of the address with the last byte set to 255; the *data-address* consists of the base address with a randomly-selected last byte (0-254). Station identification packets are broadcast to the *id-address*, and normal audio data packets are broadcast to the *data-address*. This convention allows RFE receivers to register interest only in the *id-address*, so that audio data packets need not be replicated over network gateways until a receiver actually tunes in to the station.

If the IP Multicast address ends in a byte that is in the range 1-254, then the address remains unmodified, and both station and identification packets are broadcast to that address. This convention allows a specific multicast address to be used for multi-party audio conferences.

The default IP Multicast address is designated by the hostname *RadioFreeEthernet*. The special hostname *BROADCAST* may be used to force the program to transmit UDP Broadcast packets (which will not be relayed over a network gateway). The **radio**(6) manual page contains more information on the RFE network implementation.

Range=[*hopcount*]

This option is used to configure the number of gateways over which a radio broadcast may pass. If the value is set to one, the broadcast will be restricted to the current subnet. The default Range is 8.

Service=[*service*|*port number*]

This option is used to configure the IP protocol port number used to identify Radio Free Ethernet network packets. The value may be either a service name (which is translated into a port number by looking up the name in the NIS *services* map) or a specific numeric port number.

The default port number is designated by the service name *radio*. The **radio**(6) manual page contains more information on configuring the port number.

Report=[*ON/OFF*]

This option enables the reporting, to *stderr*, of all changes in the transmitter status. It is normally used only in conjunction with Command Mode to allow the controlling program to maintain state.

Buffer=[*input buffer size in bytes*]

In normal operation, the RFE tools are not concerned with the delay between the audio input at the transmitter and the output of audio data at the receiving end. Following the model of radio broadcasting, the delay is not important as long as the transmitted data arrives in sequence and on time. For this reason, and to ensure a consistent, uninterrupted flow of audio data, the **radio_xmit** program normally uses a buffer size of around 8000 bytes to collect audio input data before broadcasting. This results in an end-to-end delay of approximately one second (or longer if the receiver's audio buffer backs up).

For use with real-time audio conferencing, shorter delays may be desired. For experimental purposes, the Buffer command is provided to alter the input delay. The buffer size is specified in bytes. For real-time experimentation, a reasonable value is 256 (corresponding to approximately 1/32 of a second). Note that a side-effect of lowering the input buffer size is to lower the transmitted packet size, resulting in an increased number of broadcast packets.

Start

This command is used to initiate broadcasting according to the current parameter settings. It is provided for Command Mode; the transmitter is enabled by default in Normal Mode.

Stop

This command is used to sign off the current station and cease broadcasting. It is provided for Command Mode operation.

Quit

This command causes the **radio_xmit** program to exit. It is provided for Command Mode operation.

EXAMPLES

Let's say that you have a radio at work that you always keep tuned to your favorite station (though you might shut off your speakers when you don't want to listen to it). If you connect its output to your workstation audio input, you can broadcast the station over the network by using the following command (the *Autostop* option will cause the program to sign-off and exit if you turn your radio off for a minute or longer):

```
example% radio_xmit Station=KPFA Input=/dev/audio Autostop=on
```

The following shell script may be scheduled to run every day at 6:00 AM to record a half-hour of news and rebroadcast it at a more sensible time:

```
#!/bin/sh
# Record from /dev/audio for 30 minutes (around 14 Mbytes).
file=/bigdisk/sound/news
audiorecord -t 30:00 $file
#
# If this runs at 6am, the news is over at 6:30.  Wait 3 hours til 9:30.
sleep 10800
#
# Now broadcast the news on the network and remove the file when finished.
radio_xmit Station=KPFA Input=$file
rm $file
```

SEE ALSO

audiorecord(1), **radio**(6), **radio_recv**(6), **xmit**(6)

NAME

soundtool – audio waveform display demo

SYNOPSIS

/usr/demo/SOUND/bin/soundtool

DISCLAIMER

This program is furnished on an AS IS basis as a demonstration of audio applications programming.

DESCRIPTION

soundtool is an XView demonstration program that allows recording, playing, and simple editing of audio data. The display consists of six regions: a play/record control panel, a function control panel, an oscilloscope, a display control panel, a waveform display panel, and a pop-up audio status panel.

Play/Record Control Panel

Play / Stop
Clicking this button plays the currently selected region of data. While data is playing this button becomes a **Stop** button. If audio output is busy when **Play** is started, this button displays **Waiting**. When the device is available, the button switches to **Stop** and audio output begins. Clicking on the **Waiting** button resets the tool to the idle state.

Record / Stop
Clicking this button starts the recording of data from the audio input port that is wired to the 8-pin mini-DIN connector on the back of the workstation. While recording is in progress, this button becomes a **Stop** button. If audio input is busy when **Record** is selected, an alert pops up and the tool resets to the idle state. A maximum of 5 minutes may be recorded at a time.

Pause Clicking this button while playing or recording suspends the current operation. The button becomes a **Resume** button that may be selected to continue the suspended operation.

Describe
Clicking this button brings up the ''Audio Status Panel''. If the panel was already visible, clicking this button removes it.

Quit Clicking this button causes **soundtool** to exit.

Play Volume
This slider adjusts the playback volume. Volume levels between **0** and **100** may be selected, where **0** represents infinite attenuation and **100** is maximum gain.

Record Volume
This slider adjusts the recording level in the range **0** to **100**.

Output To
This selector switches the audio output port between the built-in speaker and the external headphone jack.

Looping
When **Looping** is disabled, the current data region (i.e., the data between the two markers in the waveform display) is played once. If **Looping** is enabled, the selected data will be played endlessly until the **Stop** button is pressed.

Function Control Panel

Load Clicking **Load** will read in the audio file specified by the **Directory** and **File** fields. If the named file does not contain a valid audio header, the raw data is copied into the buffer and an alert is displayed. Clicking the **Store** button at that point will rewrite the file with the proper audio file header.

Arbitrarily large audio files may be loaded. However, system swap space resources may be depleted (one minute of audio data consumes roughly **.5** Mbyte of swap space).

Store Clicking **Store** will write the selected data region into the file specified by the **Directory** and **File** fields. If the named file exists, an alert will request confirmation of the operation.

Append
Clicking **Append** will append the selected data region to the file specified by the **Directory** and **File** fields. The named file must contain a valid audio file header.

Directory
The **Directory** field specifies a directory path in which to look for audio files.

File The **File** field designates the file to be loaded from, stored to, or appended to. Holding down the right mouse button on this field presents a menu of audio files in the currently designated directory. All files that contain a valid audio file header, or whose names have the suffix **.au** or **.snd**, are listed.

Oscilloscope

When the program is in the idle state and the cursor is in the waveform display panel, the oscilloscope acts as a magnifying glass, displaying the region of the audio waveform that is currently under the cursor. When the program is playing or recording, the oscilloscope displays the data that is currently being transferred. Note that there is a small time lag in the display of recorded data, due to the fact that the audio device driver buffers input data and delivers it to the application in discrete segments.

Display Control Panel

Zoom The **Zoom** slider adjusts the compression factor used in the display of the waveform. The upper compression limit is chosen so that the entire waveform will fit in the waveform display panel. [The lower limit is restricted by the ability to manipulate large scrolling regions in XView.] Adjustment of the **Zoom** slider ordinarily results in data compression or expansion around the center of the currently displayed waveform. If the waveform display contains one or both data selection markers, an attempt is made to keep at least a portion of the selected data region in the window.

The magnified waveform presented in the oscilloscope display is unaffected by the **Zoom** value. However, cursor movement over the waveform reflects the current compression; i.e., lower **Zoom** values result in finer granularity of mouse movement.

Waveform Display Panel

The waveform display shows all or part of the current waveform, depending on the current **Zoom** value. Scrolling of the waveform may be achieved either by using the scrollbar or by dragging the waveform to the right or left while holding the middle mouse button down. Note that scrolling is disabled when the entire waveform is being displayed (i.e., when the **Zoom** value is at its maximum).

In some cases, it is desirable to identify a subset of the waveform. For instance, the **Play**, **Store**, and **Append** functions operate on a selected region, rather than the entire waveform. The currently selected region of interest is delimited by dashed vertical lines. A new region may be selected by clicking the left or right mouse button and dragging it across the desired region of interest. Alternatively, a single click on the left (or right) mouse buttons adjusts the start (or end) point.

Audio Status Panel

This panel is displayed (or removed) when the **Describe** button is pressed. It contains fields that describe the data in the buffer.

Sample Rate
This field displays the sampling frequency, in samples per second.

Channels
This field denotes the number of interleaved channels of audio data.

Precision
This field identifies the encoding precision, in bits per sample.

Encoding
This field displays the encoding format.

Total Length
This field shows the length of the entire data buffer, in the form *hh*:*mm*:*ss.dd*.

Selection
This field identifies the start and end times of the currently selected region of interest.

Info String
When an audio file is loaded, the first 80 characters of the information field of the audio header are displayed in this field. This string may be edited, though the new information is only written out when the **Store** operation is performed.

BUGS

Currently, **soundtool** is capable of displaying only 8-bit μ-law encoded data.

Audio files should be mapped in order to reduce the swap space requirements. The limit on recording length should also be removed.

There are problems with scrollbars that operate on very large canvases. This constraint is the reason for the lower limit on zooming. Also, there are some problems with the display of very large audio files with a low zoom factor.

Region selections made over the waveform display panel work best when the click and drag paradigm is used. Adjusting the start or end points by a single click is susceptible to error; i.e., if the mouse moves slightly between the button down and up events, the result will be a very small selection.

SEE ALSO

audioconvert(1), **gaintool**(6)

NAME

x_buttontest – Xview demonstration and test program for SunButtons

SYNOPSIS

/usr/demo/BUTTONBOX/x_buttontest

AVAILABILITY

SPARC

DESCRIPTION

X_buttontest is an xview application that displays a window with thirty two buttons, corresponding to those on the SunButtons buttonbox. To determine if the buttonbox has been set up correctly, select the Diagnostic button on the panel. If the buttonbox is functional and correctly interfaced, each of the buttons will light in sequence for about 1 second. Then "OK." is sent to the standard output of the demo program. X_buttontest is now in its (default) interactive mode. Pressing a button on the buttonbox highlights the corresponding button on the screen. Additionally **x_buttontest** sends a BDIOBUTLITE ioctl to the buttonbox in response to each key press and key release, so that the button light is illuminated while the button is held down.

If the serial communications become confused as can happen when both the buttonbox and the dialbox are operated at the same time, one or more button lights may remain on after the button is released. Clicking on the Reset button on the panel will unconditionally turn all the button lights off.

SEE ALSO

bdconfig(1M), **x_dialtest**(6), **bd**(7M), **streamio**(7I)

NAME

x_dialtest – Xview demonstration and test program for SunDials

SYNOPSIS

/usr/demo/DIALBOX/x_dialtest

AVAILABILITY

SPARC

DESCRIPTION

X_dialtest is an xview application that displays a window with eight dials, corresponding to the dials on the SunDials dialbox. To determine if the dialbox has been set up correctly, turn a dial on the dialbox. If the dialbox is functional and correctly interfaced, turning a dial by hand will make the corresponding dial in the window turn a similar amount.

The dials do not have any notion of absolute angular position. It is changes in current angular position that are sent to the host application. Thus there is no notion of resetting the position of the dials on the dialbox hardware.

The Diagnostic button on the panel is a demo mode of the **x_dialtest** program. The pointer of each of the dials in the window is rotated one full circle and then disappears in turn. when all eight dials have been rotated, the display dial pointers are reset to their previous rotational positions. The only diagnostic done on the dialbox is a firmware self check. If this self check passes then "OK." is sent to the standard output of the demo program.

The Ram Dump button on the panel arranges to place some firmware data into the file ram_dump.dat in the current directory. It is intended for factory diagnostics use, and is not publicly documented further.

SEE ALSO

bdconfig(1M), **x_buttontest**(6), **bd**(7M), **streamio**(7I)

NAME

xmit – Radio Free Ethernet transmitter

SYNOPSIS

xmit [–**h** *host*] [–**s** *service*] [*generic-tool-arguments*]

DISCLAIMER

This program is furnished on an AS IS basis as a demonstration of audio applications programming.

DESCRIPTION

xmit is the window-based Radio Free Ethernet transmitter. It functions as a graphical front-end to the **radio_xmit**(6) program, which it uses to read audio data from the audio device (or a file) and broadcast it over the network. (For an overview of Radio Free Ethernet, please refer to the **radio**(6) manual page.) Before you can start broadcasting, you must configure a station name. Clicking on the **Station** item brings up a station edit panel that allows you to specify the station name and broadcast characteristics for one or more stations. The **Station** item brings up a menu of the stations that are configured. Once you select a station, you may begin broadcasting by pressing the **Power** button. Ordinarily, the program broadcasts to the IP Multicast address identified by the host name *RadioFreeEthernet* found in the NIS *hosts* map. The –**h** *host* command-line option may be used to specify an alternate host address or name to use for the default IP Multicast address. The Radio Free Ethernet tools use the port number identified by the service name *radio* found in the NIS *services* map. The –**s** *service* command-line option may be used to specify an alternate service name or port number to use. The following sections describe the individual panel controls. In addition, online help is available by positioning the pointer over the item in question and pressing the **<Help>** key.

Power toggles the state of the transmitter on and off. Since this involves communicating with a running copy of the **radio_xmit** program, this operation may take a few seconds to complete. This button is only enabled when a station name has been configured and selected.

Station brings up a menu of the radio stations that have been configured for broadcast. The **Edit...** entry brings up a panel that may be used to add or modify the broadcast station configuration. Changes to the station configuration parameters do not take effect until the **Add** or **Change** button is pressed. If the the configuration parameters are changed for a station that is currently broadcasting, the new configuration will not take effect until the station is powered off. **Add**, **Change**, and **Delete** also cause the current station list and program parameters to be written out to the initialization file.

Input Volume may be used to manually adjust the audio input gain level. It is only enabled when broadcasting data from an audio device (as opposed to a pre-recorded file).

Auto Volume Adjust enables an automatic gain control algorithm that monitors the input volume level and adjusts it when the audio data is too soft or too loud. The algorithm tends to be cautious, lowering the volume quickly when it is too loud, but raising the level slowly to avoid the noise-pumping effects characteristic of cheap tape recorders.

This control is only enabled when broadcasting data from an audio device. The station configuration panel comes up by default in an abbreviated form, displaying only the **Station** option. The plus (+) button in the lower right corner causes the window to expand to display additional transmit parameters for the selected station. The following sections describe all of the station configuration panel controls:

Station is a text field in which a four-letter station name may be entered. The station name is used to identify your broadcast program.

Sign-On File specifies an audio file that is broadcast when the transmitter is initially powered on.

Audio Input specifies the source of audio data for the normal station broadcast.

Sign-Off File specifies an audio file that is broadcast when the transmitter is powered off.

Auto-Shutoff determines the action to be taken when the audio input source is silent. When this option is enabled, the station will automatically sign off if there has been no audio input for a full minute. If this option is disabled, the station will continue to broadcast station identification packets, but will suppress the broadcasting of audio data until some sound is detected.

Audio Format selects the audio data format that will be transmitted. The uncompressed format causes 8000 bytes of audio data to be broadcast each second. Compressed data sends only 4000 bytes per second, but requires more computation on both the transmitter and receivers.

Multicast Addr specifies the network broadcast address (see **radio_xmit**(6) for more information on the IP Multicast implementation). The special string BROADCAST causes the program to transmit UDP Broadcast packets (which will not be relayed over a network gateway).

Multicast Hops specifies the number of network gateways that RFE broadcast packets may traverse.

FILES **~/.radiorc** startup initialization file for **radio** and **xmit**

SEE ALSO **radio**(6), **radio_recv**(6), **radio_xmit**(6)

Index