



ChorusOS man pages section 3TELD: Telnet Services

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part No: 806-3336
December 10, 1999

Copyright 1999 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 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. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, ChorusOS, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was 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.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 1999 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, Californie 94303-4900 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, ChorusOS, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Contents

PREFACE 5

Intro(3TELD) 11

inetAccept(3TELD) 13

inetClient(3TELD) 13

inetBind(3TELD) 14

inetClose(3TELD) 14

inetAccept(3TELD) 15

inetClient(3TELD) 15

inetBind(3TELD) 16

inetClose(3TELD) 16

telnetdWrite(3TELD) 17

telnetdFlush(3TELD) 17

telnetdInit(3TELD) 18

telnetdFree(3TELD) 18

telnetdGetTermState(3TELD) 19

telnetdSetTermState(3TELD) 19

telnetdInit(3TELD) 21

telnetdFree(3TELD) 21

telnetdUser(3TELD) 22

telnetdPasswd(3TELD)	22
telnetdRead(3TELD)	23
telnetdReadLine(3TELD)	25
telnetdGetTermState(3TELD)	26
telnetdSetTermState(3TELD)	26
telnetdUser(3TELD)	28
telnetdPasswd(3TELD)	28
telnetdWrite(3TELD)	29
telnetdFlush(3TELD)	29
Index	29

PREFACE

Overview

A man page is provided for both the naive user, and sophisticated user who is familiar with the ChorusOS™ 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 is a list of sections in the ChorusOS man pages and the information it references:

- *Section 1CC: User Utilities; Host and Target Utilities*
- *Section 1M: System Management Utilities*
- *Section 2DL: System Calls; Data Link Services*
- *Section 2K: System Calls; Kernel Services*
- *Section 2MON: System Calls; Monitoring Services*
- *Section 2POSIX: System Calls; POSIX System Calls*
- *Section 2RESTART: System Calls; Hot Restart and Persistent Memory*
- *Section 2SEG: System Calls; Virtual Memory Segment Services*
- *Section 3FTPD: Libraries; FTP Daemon*
- *Section 3M: Libraries; Mathematical Libraries*
- *Section 3POSIX: Libraries; POSIX Library Functions*
- *Section 3RPC: Libraries; RPC Services*
- *Section 3STDC: Libraries; Standard C Library Functions*
- *Section 3TELD: Libraries; Telnet Services*
- *Section 4CC: Files*

- *Section 5FEA: ChorusOS Features and APIs*
- *Section 7P: Protocols*
- *Section 7S: Services*
- *Section 9DDI: Device Driver Interfaces*
- *Section 9DKI: Driver to Kernel Interface*
- *Section 9DRV: Driver Implementations*

ChorusOS man pages are grouped in Reference Manuals, with one reference manual per section.

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	<p>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. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.</p> <p>The following special characters are used in this section:</p> <p>[] The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.</p> <p>. . . Ellipses. Several values may be provided for the previous argument, or the previous argument can be specified multiple times, for example, 'filename . . . '.</p> <p> Separator. Only one of the arguments separated by this character can be specified at time.</p> <p>{ } Braces. The options and/or arguments enclosed within braces are</p>

interdependent, such that everything enclosed must be treated as a unit.

FEATURES

This section provides the list of features which offer an interface. An API may be associated with one or more system features. The interface will be available if one of the associated features has been configured.

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.

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 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	<p>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:</p> <p>Commands Modifiers Variables Expressions Input Grammar</p>
EXAMPLES	<p>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 <code>example%</code> or if the user must be superuser, <code>example#</code>. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS and USAGE sections.</p>
ENVIRONMENT VARIABLES	<p>This section lists any environment variables that the command or function affects, followed by a brief description of the effect.</p>
EXIT STATUS	<p>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 other than zero for various error conditions.</p>
FILES	<p>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.</p>
SEE ALSO	<p>This section lists references to other man pages, in-house documentation and outside publications.</p>
DIAGNOSTICS	<p>This section lists diagnostic messages with a brief explanation of the condition causing the error.</p>
WARNINGS	<p>This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.</p>
NOTES	<p>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.</p>

BUGS

This section describes known bugs and wherever possible, suggests workarounds.

TELNETD Library

NAME Intro – introduction to the TELNETD library

SYNOPSIS `#include <arpa/telnetd.h>`

DESCRIPTION The TELNETD library provides a set of functions to control and access the server side of a TELNET session. The library functions manage the TELNET protocol, filtering all TELNET commands from the stream of characters sent by the TELNET client.

For each active TELNET session, the library maintains the state associated to the virtual terminal. This state can be modified either by the TELNET client using the TELNET protocol or by the TELNET server using the *telnetdSetTermState* function of the TELNETD library. The *telnetdSetTermState* function can be used to switch the input mode of the TELNET client from line mode to one character at a time mode.

API The TELNETD library API is summarized in the following table.

Function	Comment
<i>inetAccept</i>	Accept new connections
<i>inetBind</i>	Create and bind sockets to IP ports
<i>inetClient</i>	Get the client IP address
<i>inetClose</i>	Close sockets created by <i>inetBind</i>
<i>telnetdFlush</i>	Write out buffered data of a TELNET session
<i>telnetdFree</i>	Free resources associated to a TELNET session
<i>telnetdGetTermState</i>	Get the terminal state of a TELNET session
<i>telnetdInit</i>	Initialize a TELNET session on a new connection
<i>telnetdPasswd</i>	Get the password of the user of a TELNET session
<i>telnetdRead</i>	Read characters from a TELNET session
<i>telnetdReadLine</i>	Read a line of characters from a TELNET session
<i>telnetdSetTermState</i>	Set the terminal state of a TELNET session
<i>telnetdUser</i>	Get the name of the user of a TELNET session
<i>telnetdWrite</i>	Write characters on a TELNET session

As these functions rely on the POSIX socket API, their use should be restricted to a ChorusOS system configured with the enabled POSIX_SOCKETS feature.

ATTRIBUTESSee `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

Name	Description
<code>inetAccept(3TELD)</code>	wait for a new INET connection
<code>inetBind(3TELD)</code>	bind, close INET sockets
<code>inetClient(3TELD)</code>	See <code>inetAccept(3TELD)</code>
<code>inetClose(3TELD)</code>	See <code>inetBind(3TELD)</code>
<code>telnetdFlush(3TELD)</code>	See <code>telnetdWrite(3TELD)</code>
<code>telnetdFree(3TELD)</code>	See <code>telnetdInit(3TELD)</code>
<code>telnetdGetTermState(3TELD)</code>	get or set TELNET terminal state
<code>telnetdInit(3TELD)</code>	initialize or free a TELNET session
<code>telnetdPasswd(3TELD)</code>	See <code>telnetdUser(3TELD)</code>
<code>telnetdRead(3TELD)</code>	read from a TELNET session
<code>telnetdReadLine(3TELD)</code>	read a line of characters from a TELNET session
<code>telnetdSetTermState(3TELD)</code>	See <code>telnetdGetTermState(3TELD)</code>
<code>telnetdUser(3TELD)</code>	TELNET session authentication
<code>telnetdWrite(3TELD)</code>	write or flush a TELNET session

NAME	inetAccept, inetClient – wait for a new INET connection				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int inetAccept(inetSocket * <i>sockets</i>, int <i>count</i>, int * <i>connectionSocket</i>, int * <i>connectionPort</i>); int inetClient(int <i>connectionSocket</i>, struct in_addr * <i>clientInAddr</i>);</pre>				
DESCRIPTION	<p>The <i>inetAccept</i> function waits for a new INET connection.</p> <p>The <i>sockets</i> argument is an array of <i>count</i> <i>inetSocket</i> data structures describing sockets previously created and bound to IP ports by <i>inetBind</i> .</p> <p>The <i>connectionSocket</i> argument is a result parameter filled with the file descriptor of the new socket created for the new INET connection.</p> <p>The <i>connectionPort</i> argument is also a result parameter that is filled with the IP port number on which the client initiated the connection.</p> <p>The <i>telnetdInit</i> function may be invoked later on <i>connectionSocket</i> to initialize a TELNET session. This session can be controlled by other functions of the TELNETD library such as <i>telnetdUser</i> , <i>telnetdRead</i> or <i>telnetdSetTermState</i> .</p> <p>However, the application may also manage the new connection directly using the low level POSIX socket API. In particular, it can close the socket if it elects to reject the connection (see <i>close</i> (2POSIX)), or read all data received on the connection including TELNET commands (see <i>read</i> (2POSIX)).</p> <p>The <i>inetClient</i> function fills the <i>clientInAddr</i> argument with the Internet address of the TELNET client connected to the <i>connectionSocket</i> socket.</p>				
RETURN VALUE	Upon successful completion, <i>inetAccept</i> returns a value of zero. Otherwise it returns -1 and sets <i>errno</i> to indicate the error.				
ERRORS	<p><i>errno</i> is set to EINVAL if one of the <i>socket</i> members of the <i>inetSocket</i> data structures is still initialized to <i>TD_INVALID_SOCKET</i> .</p> <p><i>inetAccept</i> may also fail and set <i>errno</i> to any of the errors specified for the <i>accept</i> (2POSIX) and <i>select</i> (2POSIX) functions.</p> <p><i>inetClient</i> may fail and set <i>errno</i> to any of the errors specified for the <i>getpeername</i> (2POSIX) function.</p>				
ATTRIBUTES	<p>See <i>attributes</i>(5) for descriptions of the following attributes:</p> <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	<i>inetBind</i> (3TELD) , <i>inetClient</i> (3TELD) , <i>telnetdInit</i> (3TELD)				

NAME	inetBind, inetClose – bind, close INET sockets				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int inetBind(inetSocket * <i>sockets</i>, int <i>count</i>); int inetClose(inetSocket * <i>sockets</i>, int <i>count</i>);</pre>				
DESCRIPTION	<p>The <i>inetBind</i> function creates one stream socket and binds it to every IP port from which the server will listen.</p> <p>The <i>sockets</i> parameter is an array of <i>count</i> <i>inetSocket</i> data structures. Each <i>inetSocket</i> structure has the following members:</p> <pre>int port ; /* IP port */ int socket ; /* socket bound to the port */</pre> <p>where <i>port</i> must be a valid IP port number and <i>socket</i> must be initialized to <i>TD_INVALID_SOCKET</i>. Upon successful return from the call, <i>socket</i> will hold the file descriptor of the new socket bound to the IP port.</p> <p>The <i>inetClose</i> function closes all sockets which were created by <i>inetBind</i>. It does not close sockets returned by <i>inetAccept</i> and does not release any memory allocated by <i>telnetdInit</i> for a new TELNET session; this must be done explicitly by closing the socket and calling <i>telnetdFree</i> on active TELNET sessions.</p>				
RETURN VALUE	Upon successful completion, these functions return a value of zero. Otherwise they return -1 and set <i>errno</i> to indicate the error.				
ERRORS	<p><i>errno</i> is set to EINVAL if one of the <i>socket</i> members of the <i>inetSocket</i> data structures passed to <i>inetBind</i> is not initialized to <i>TD_INVALID_SOCKET</i>.</p> <p>The <i>inetBind</i> function may also fail and set <i>errno</i> to any of the errors specified for the <i>socket</i> (2POSIX), and <i>bind</i> (2POSIX) functions.</p>				
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:				
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	inetAccept(3TELD)				

NAME	inetAccept, inetClient – wait for a new INET connection				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int inetAccept(inetSocket * <i>sockets</i>, int <i>count</i>, int * <i>connectionSocket</i>, int * <i>connectionPort</i>); int inetClient(int <i>connectionSocket</i>, struct in_addr * <i>clientInAddr</i>);</pre>				
DESCRIPTION	<p>The <i>inetAccept</i> function waits for a new INET connection.</p> <p>The <i>sockets</i> argument is an array of <i>count</i> <i>inetSocket</i> data structures describing sockets previously created and bound to IP ports by <i>inetBind</i> .</p> <p>The <i>connectionSocket</i> argument is a result parameter filled with the file descriptor of the new socket created for the new INET connection.</p> <p>The <i>connectionPort</i> argument is also a result parameter that is filled with the IP port number on which the client initiated the connection.</p> <p>The <i>telnetdInit</i> function may be invoked later on <i>connectionSocket</i> to initialize a TELNET session. This session can be controlled by other functions of the TELNETD library such as <i>telnetdUser</i> , <i>telnetdRead</i> or <i>telnetdSetTermState</i> .</p> <p>However, the application may also manage the new connection directly using the low level POSIX socket API. In particular, it can close the socket if it elects to reject the connection (see <i>close</i> (2POSIX)), or read all data received on the connection including TELNET commands (see <i>read</i> (2POSIX)).</p> <p>The <i>inetClient</i> function fills the <i>clientInAddr</i> argument with the Internet address of the TELNET client connected to the <i>connectionSocket</i> socket.</p>				
RETURN VALUE	Upon successful completion, <i>inetAccept</i> returns a value of zero. Otherwise it returns -1 and sets <i>errno</i> to indicate the error.				
ERRORS	<p><i>errno</i> is set to EINVAL if one of the <i>socket</i> members of the <i>inetSocket</i> data structures is still initialized to <i>TD_INVALID_SOCKET</i> .</p> <p><i>inetAccept</i> may also fail and set <i>errno</i> to any of the errors specified for the <i>accept</i> (2POSIX) and <i>select</i> (2POSIX) functions.</p> <p><i>inetClient</i> may fail and set <i>errno</i> to any of the errors specified for the <i>getpeername</i> (2POSIX) function.</p>				
ATTRIBUTES	<p>See <i>attributes</i>(5) for descriptions of the following attributes:</p> <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	<i>inetBind</i> (3TELD) , <i>inetClient</i> (3TELD) , <i>telnetdInit</i> (3TELD)				

NAME	inetBind, inetClose – bind, close INET sockets				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int inetBind(inetSocket * <i>sockets</i>, int <i>count</i>); int inetClose(inetSocket * <i>sockets</i>, int <i>count</i>);</pre>				
DESCRIPTION	<p>The <i>inetBind</i> function creates one stream socket and binds it to every IP port from which the server will listen.</p> <p>The <i>sockets</i> parameter is an array of <i>count</i> <i>inetSocket</i> data structures. Each <i>inetSocket</i> structure has the following members:</p> <pre>int port ; /* IP port */ int socket ; /* socket bound to the port */</pre> <p>where <i>port</i> must be a valid IP port number and <i>socket</i> must be initialized to <i>TD_INVALID_SOCKET</i>. Upon successful return from the call, <i>socket</i> will hold the file descriptor of the new socket bound to the IP port.</p> <p>The <i>inetClose</i> function closes all sockets which were created by <i>inetBind</i>. It does not close sockets returned by <i>inetAccept</i> and does not release any memory allocated by <i>telnetdInit</i> for a new TELNET session; this must be done explicitly by closing the socket and calling <i>telnetdFree</i> on active TELNET sessions.</p>				
RETURN VALUE	Upon successful completion, these functions return a value of zero. Otherwise they return -1 and set <i>errno</i> to indicate the error.				
ERRORS	<p><i>errno</i> is set to EINVAL if one of the <i>socket</i> members of the <i>inetSocket</i> data structures passed to <i>inetBind</i> is not initialized to <i>TD_INVALID_SOCKET</i>.</p> <p>The <i>inetBind</i> function may also fail and set <i>errno</i> to any of the errors specified for the <i>socket</i> (2POSIX), and <i>bind</i> (2POSIX) functions.</p>				
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:				
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	inetAccept(3TELD)				

NAME	telnetdWrite, telnetdFlush – write or flush a TELNET session				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int telnetdWrite(telnetdSessionHandle * hdl, void * buf, unsigned int nchar); int telnetdFlush(telnetdSessionHandle * hdl);</pre>				
DESCRIPTION	<p>The <i>telnetdWrite</i> function attempts to write <i>nchar</i> characters from the buffer pointed to by <i>buf</i> to the TELNET session designated by the <i>hdl</i> handle.</p> <p>If the TELNET connection cannot accept data immediately and the connection socket has <code>O_NONBLOCK</code> set, <i>telnetdWrite</i> returns -1 and sets <i>errno</i> to <code>EAGAIN</code>. Otherwise, <i>telnetdWrite</i> blocks until data can be accepted.</p> <p>The <code>O_NONBLOCK</code> option may be set for the connection socket by calling <i>fcntl</i> (2POSIX) after the TELNET session has been initialized by <i>telnetdInit</i>.</p> <p>The <i>telnetdFlush</i> function causes any buffered data waiting to be written for the TELNET session designated by the <i>hdl</i> handle to be written to the TELNET connection socket.</p>				
RETURN VALUE	Upon successful completion, <i>telnetdWrite</i> returns the number of characters actually written; this number may be less than <i>nchar</i> . Upon successful completion, <i>telnetdFlush</i> returns 0. In case of failure, both functions return -1 and set <i>errno</i> to indicate the error.				
ERRORS	<p><i>errno</i> is set to <code>EINVAL</code> if <i>telnetdFlush</i> or <i>telnetdWrite</i> is called with an invalid <i>hdl</i> handle.</p> <p>The <i>telnetdWrite</i> and <i>telnetdFlush</i> functions may fail and set <i>errno</i> to any of the errors specified for the <i>write</i> (2POSIX) function.</p>				
ATTRIBUTES	<p>See <i>attributes</i>(5) for descriptions of the following attributes:</p> <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	<i>telnetdGetTermState</i> (3TELD), <i>telnetdRead</i> (3TELD)				

NAME	telnetdInit, telnetdFree – initialize or free a TELNET session				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int telnetdInit(int connectionSocket, telnetdSessionHandle * hdl); int telnetdFree(telnetdSessionHandle * hdl);</pre>				
DESCRIPTION	<p>The <i>telnetdInit</i> function initializes a TELNET session for the <i>connectionSocket</i> socket obtained by <i>inetAccept</i> .</p> <p>The <i>hdl</i> argument is a result parameter filled with the session handle of the new TELNET session. This handle must be used as an argument to other functions of the TELNETD library to designate the new TELNET session.</p> <p>The <i>telnetdInit</i> function allocates memory to maintain the state of the TELNET session. It also sends TELNET options to the client side indicating the ability to do <i>remote echo</i> of characters, and to <i>suppress go ahead</i> . The state of the virtual terminal is configured to operate in line mode (see <i>telnetdSetTermState</i> (3TELD)). The new TELNET session can perform the following functions: <i>echo</i> , <i>binary</i> , <i>suppress go ahead</i> , and <i>timing mark</i> . It allows the remote client to perform the following functions: <i>binary</i> and <i>suppress go ahead</i> .</p> <p>The <i>telnetdFree</i> function frees all memory allocated by <i>telnetdInit</i> for the TELNET session designated by its <i>hdl</i> handle. The TELNET connection socket associated with the TELNET session is not closed by <i>telnetdFree</i> .</p>				
RETURN VALUE	Upon successful completion, these functions return a value of zero. Otherwise they return -1 and set <i>errno</i> to indicate the error.				
ERRORS	<p>The <i>telnetdInit</i> function may fail and set <i>errno</i> to any of the errors specified for the <i>malloc</i> (3STDC), <i>read</i> (2POSIX) and <i>write</i> (2POSIX) functions.</p> <p>The <i>errno</i> file is set to EINVAL if <i>telnetdFree</i> is called with an invalid <i>hdl</i> handle.</p>				
ATTRIBUTES	See <i>attributes</i> (5) for descriptions of the following attributes:				
	<table> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	<i>telnetdFree</i> (3TELD) , <i>telnetdPasswd</i> (3TELD) , <i>telnetdRead</i> (3TELD) , <i>telnetdUser</i> (3TELD) , <i>telnetdWrite</i> (3TELD)				

NAME	telnetdGetTermState, telnetdSetTermState – get or set TELNET terminal state																				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int telnetdGetTermState(telnetdSessionHandle * hdl, telnetdTermState * termState); int telnetdSetTermState(telnetdSessionHandle * hdl, telnetdTermState * termState);</pre>																				
DESCRIPTION	<p>The <i>telnetdGetTermState</i> function returns to the <i>termState</i> argument the current state of the virtual terminal of the TELNET session designated by the <i>hdl</i> handle.</p> <p>The <i>telnetdSetTermState</i> function sets the state of the virtual terminal of the TELNET session designated by the <i>hdl</i> handle to the value of the <i>termState</i> argument .</p> <p>The <i>termState</i> argument is a pointer to a data structure which has the following member:</p> <pre>int state ; /* terminal state */</pre> <p>where <i>state</i> is constructed by or-ing flags from the following list:</p> <table> <tr> <td>TERMSTATE_LINEMODE</td><td>client input mode is set to line mode (otherwise client input mode is one character at a time)</td></tr> <tr> <td>TERMSTATE_ECHO</td><td>client echoing is enabled (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_EDIT</td><td>client line editing is enabled (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_TRAPSIG</td><td>client signal trapping is enabled (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_SOFTTAB</td><td>client tab expansion is enabled (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_LITECHO</td><td>typed control characters are echoed literally by the client (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_CRNL</td><td>map CR to NL on input (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_BININ</td><td>binary on input.</td></tr> <tr> <td>TERMSTATE_BINOUT</td><td>binary on output.</td></tr> <tr> <td>TERMSTATE_FLOW</td><td>client flow control is enabled.</td></tr> </table>	TERMSTATE_LINEMODE	client input mode is set to line mode (otherwise client input mode is one character at a time)	TERMSTATE_ECHO	client echoing is enabled (valid only in line mode)	TERMSTATE_EDIT	client line editing is enabled (valid only in line mode)	TERMSTATE_TRAPSIG	client signal trapping is enabled (valid only in line mode)	TERMSTATE_SOFTTAB	client tab expansion is enabled (valid only in line mode)	TERMSTATE_LITECHO	typed control characters are echoed literally by the client (valid only in line mode)	TERMSTATE_CRNL	map CR to NL on input (valid only in line mode)	TERMSTATE_BININ	binary on input.	TERMSTATE_BINOUT	binary on output.	TERMSTATE_FLOW	client flow control is enabled.
TERMSTATE_LINEMODE	client input mode is set to line mode (otherwise client input mode is one character at a time)																				
TERMSTATE_ECHO	client echoing is enabled (valid only in line mode)																				
TERMSTATE_EDIT	client line editing is enabled (valid only in line mode)																				
TERMSTATE_TRAPSIG	client signal trapping is enabled (valid only in line mode)																				
TERMSTATE_SOFTTAB	client tab expansion is enabled (valid only in line mode)																				
TERMSTATE_LITECHO	typed control characters are echoed literally by the client (valid only in line mode)																				
TERMSTATE_CRNL	map CR to NL on input (valid only in line mode)																				
TERMSTATE_BININ	binary on input.																				
TERMSTATE_BINOUT	binary on output.																				
TERMSTATE_FLOW	client flow control is enabled.																				

TERMSTATE_RESTARTANY restart flow on any character.

At initialization time, *telnetdInit* negotiates the following terminal state with the TELNET client:

TERMSTATE_FLOW | TERMSTATE_LINEMODE | TERMSTATE_CRNL |
TERMSTATE_ECHO | TERMSTATE_EDIT | TERMSTATE_TRAPSIG

RETURN VALUE

Upon successful completion, the functions return a value of zero. Otherwise they return -1 and set *errno* to indicate the error.

ERRORS

errno is set to EINVAL if one of the functions is called with an invalid *hdl* handle.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

SEE ALSO

`telnetdInit(3TELD)`

NAME	telnetdInit, telnetdFree – initialize or free a TELNET session				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int telnetdInit(int connectionSocket, telnetdSessionHandle * hdl); int telnetdFree(telnetdSessionHandle * hdl);</pre>				
DESCRIPTION	<p>The <i>telnetdInit</i> function initializes a TELNET session for the <i>connectionSocket</i> socket obtained by <i>inetAccept</i> .</p> <p>The <i>hdl</i> argument is a result parameter filled with the session handle of the new TELNET session. This handle must be used as an argument to other functions of the TELNETD library to designate the new TELNET session.</p> <p>The <i>telnetdInit</i> function allocates memory to maintain the state of the TELNET session. It also sends TELNET options to the client side indicating the ability to do <i>remote echo</i> of characters, and to <i>suppress go ahead</i> . The state of the virtual terminal is configured to operate in line mode (see <i>telnetdSetTermState</i> (3TELD)). The new TELNET session can perform the following functions: <i>echo</i> , <i>binary</i> , <i>suppress go ahead</i> , and <i>timing mark</i> . It allows the remote client to perform the following functions: <i>binary</i> and <i>suppress go ahead</i> .</p> <p>The <i>telnetdFree</i> function frees all memory allocated by <i>telnetdInit</i> for the TELNET session designated by its <i>hdl</i> handle. The TELNET connection socket associated with the TELNET session is not closed by <i>telnetdFree</i> .</p>				
RETURN VALUE	Upon successful completion, these functions return a value of zero. Otherwise they return -1 and set <i>errno</i> to indicate the error.				
ERRORS	<p>The <i>telnetdInit</i> function may fail and set <i>errno</i> to any of the errors specified for the <i>malloc</i> (3STDC), <i>read</i> (2POSIX) and <i>write</i> (2POSIX) functions.</p> <p>The <i>errno</i> file is set to EINVAL if <i>telnetdFree</i> is called with an invalid <i>hdl</i> handle.</p>				
ATTRIBUTES	<p>See <i>attributes</i>(5) for descriptions of the following attributes:</p> <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	<i>telnetdFree</i> (3TELD) , <i>telnetdPasswd</i> (3TELD) , <i>telnetdRead</i> (3TELD) , <i>telnetdUser</i> (3TELD) , <i>telnetdWrite</i> (3TELD)				

NAME	telnetdUser, telnetdPasswd – TELNET session authentication				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int telnetdUser(telnetdSessionHandle * hdl, char * buf, unsigned int nchar); int telnetdPasswd(telnetdSessionHandle * hdl, char * buf, unsigned int nchar);</pre>				
DESCRIPTION	<p>The <i>telnetdUser</i> function requests the name of the user connected to the TELNET session designated by the <i>hdl</i> handle. It then copies this name into the buffer pointed to by <i>buf</i>. No more than <i>nchar</i> characters are copied to <i>buf</i>, and the last character is always the null character.</p> <p>The <i>telnetdPasswd</i> function requests the password of the user connected to the TELNET session designated by the <i>hdl</i> handle. It then copies this password into the buffer pointed to by <i>buf</i>. No more than <i>nchar</i> characters are copied to <i>buf</i>, the last character is always the null character. Echoing is turned off while the password is entered by the user.</p>				
RETURN VALUE	Upon successful completion, the functions return a value of zero. Otherwise they return -1 and set <i>errno</i> to indicate the error.				
ERRORS	<p><i>errno</i> is set to EINVAL if one of the functions is called with an invalid <i>hdl</i> handle.</p> <p>The <i>telnetdUser</i> and <i>telnetdPasswd</i> functions may fail and set <i>errno</i> to any of the errors specified for the <i>telnetdReadLine</i> (3TELD) function.</p>				
ATTRIBUTES	<p>See <i>attributes</i>(5) for descriptions of the following attributes:</p> <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	<i>telnetdReadLine</i> (3TELD)				

NAME	telnetdRead – read from a TELNET session
SYNOPSIS	<pre>#include <arpa/telnet.h> int telnetdRead(telnetdSessionHandle *hdl, void *buf, unsigned int nchar);</pre>
DESCRIPTION	<p>The <i>telnetdRead</i> function attempts to read <i>nchar</i> characters from the TELNET session designated by the <i>hdl</i> handle into the buffer pointed to by <i>buf</i>.</p> <p>While reading characters from the TELNET connection socket, <i>telnetdRead</i> filters and executes all TELNET commands sent by the TELNET client. Some of these commands may change the state of the virtual terminal associated with this session (for example, change the client input mode).</p> <p>The session handle <i>hdl</i> is a pointer to a <i>telnetdSessionHandle</i> data structure which has the following members:</p> <pre>int socket ; /* TELNET connection socket */ int newState ; /* virtual terminal state flag */</pre> <p>where <i>socket</i> is the socket associated with the TELNET session. The <i>newState</i> flag is set to 1 by <i>telnetdRead</i> if the state of the virtual terminal changed during the <i>telnetdRead</i> operation.</p> <p>If the TELNET session has no data available and the connection socket has <code>O_NONBLOCK</code> set, <i>telnetdRead</i> returns -1 and sets <i>errno</i> to <code>EAGAIN</code>. Otherwise, <i>telnetdRead</i> blocks until data becomes available or the state of the virtual terminal changes.</p> <p>The <code>O_NONBLOCK</code> option may be set for the connection socket by calling <i>fcntl(2POSIX)</i> after the TELNET session has been initialized using <i>telnetdInit</i>.</p> <p>On return of <i>telnetdRead</i> the application should check whether the state of the virtual terminal has changed, and get the new state by calling <i>telnetdGetTermState</i>. Calling <i>telnetdGetTermState</i> returns the new terminal state; it also clears the <i>newState</i> flags of the session handle and session descriptor kept internally by the TELNETD library.</p>
RETURN VALUE	Upon successful completion, <i>telnetdRead</i> returns the number of characters actually read and placed in the buffer; this number may be less than <i>nchar</i> . Otherwise it returns -1 and sets <i>errno</i> to indicate the error.
ERRORS	<p>The <i>errno</i> file is set to <code>EINVAL</code> if <i>telnetdRead</i> is called with an invalid <i>hdl</i> handle.</p> <p>The <i>telnetdRead</i> function may fail and set <i>errno</i> to any of the errors specified for the <i>read(2POSIX)</i> and <i>write(2POSIX)</i> functions.</p>
ATTRIBUTES	See <i>attributes(5)</i> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

SEE ALSO

telnetdGetTermState(3TELD), telnetdReadLine(3TELD),
telnetdWrite(3TELD)

NAME	telnetdReadLine – read a line of characters from a TELNET session				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int telnetdReadLine(telnetdSessionHandle *hdl, void *buf, unsigned int nchar, int echo, char *prompt);</pre>				
DESCRIPTION	<p>The <i>telnetdReadLine</i> function attempts to read a line of characters from the TELNET session designated by the <i>hdl</i> handle into the buffer pointed to by <i>buf</i>. A line is delimited by a NEWLINE (ASCII LF) character. This means that <i>telnetdReadLine</i> will not complete until an entire line has been typed. However, <i>nchar</i> indicates the maximum number of characters to be read. Any number of characters may be requested in <i>telnetdReadLine</i>, even one, without losing information. Also, no matter how many characters are requested by the call, a maximum of one line will be returned.</p> <p>If the value of <i>echo</i> is not zero, characters read from the TELNET session will be echoed on the user terminal. Otherwise, no echoing will be performed.</p> <p>If <i>prompt</i> is not the NULL pointer, <i>telnetdReadLine</i> first writes the string (terminated by a null character) pointed to by <i>prompt</i> on the TELNET connection.</p> <p>If the TELNET session has no data available and the connection socket has O_NONBLOCK set, <i>telnetdReadLine</i> returns -1 and sets <i>errno</i> to EAGAIN. Otherwise, <i>telnetdReadLine</i> blocks until a line of characters becomes available.</p> <p>As described in <i>telnetdRead</i>, the application should check if on return of <i>telnetdReadLine</i> the state of the virtual terminal has changed, and get the new state by calling <i>telnetdGetTermState</i>.</p>				
RETURN VALUE	<p>Upon successful completion, <i>telnetdReadLine</i> returns the number of characters actually read and placed in the buffer; this number may be less than <i>nchar</i>. Otherwise it returns -1 and sets <i>errno</i> to indicate the error.</p>				
ERRORS	<p><i>errno</i> is set to EINVAL if <i>telnetdReadLine</i> is called with an invalid <i>hdl</i> handle.</p> <p>The <i>telnetdReadLine</i> function may fail and set <i>errno</i> to any of the errors specified for the <i>telnetdRead</i>(3TELD) and <i>telnetdWrite</i>(3TELD) functions.</p>				
ATTRIBUTES	<p>See <i>attributes</i>(5) for descriptions of the following attributes:</p> <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	<p><i>telnetdGetTermState</i>(3TELD), <i>telnetdRead</i>(3TELD), <i>telnetdWrite</i>(3TELD)</p>				

NAME	telnetdGetTermState, telnetdSetTermState – get or set TELNET terminal state																				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int telnetdGetTermState(telnetdSessionHandle * hdl, telnetdTermState * termState); int telnetdSetTermState(telnetdSessionHandle * hdl, telnetdTermState * termState);</pre>																				
DESCRIPTION	<p>The <i>telnetdGetTermState</i> function returns to the <i>termState</i> argument the current state of the virtual terminal of the TELNET session designated by the <i>hdl</i> handle.</p> <p>The <i>telnetdSetTermState</i> function sets the state of the virtual terminal of the TELNET session designated by the <i>hdl</i> handle to the value of the <i>termState</i> argument .</p> <p>The <i>termState</i> argument is a pointer to a data structure which has the following member:</p> <pre>int state ; /* terminal state */</pre> <p>where <i>state</i> is constructed by or-ing flags from the following list:</p> <table> <tr> <td>TERMSTATE_LINEMODE</td><td>client input mode is set to line mode (otherwise client input mode is one character at a time)</td></tr> <tr> <td>TERMSTATE_ECHO</td><td>client echoing is enabled (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_EDIT</td><td>client line editing is enabled (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_TRAPSIG</td><td>client signal trapping is enabled (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_SOFTTAB</td><td>client tab expansion is enabled (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_LITECHO</td><td>typed control characters are echoed literally by the client (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_CRNL</td><td>map CR to NL on input (valid only in line mode)</td></tr> <tr> <td>TERMSTATE_BININ</td><td>binary on input.</td></tr> <tr> <td>TERMSTATE_BINOUT</td><td>binary on output.</td></tr> <tr> <td>TERMSTATE_FLOW</td><td>client flow control is enabled.</td></tr> </table>	TERMSTATE_LINEMODE	client input mode is set to line mode (otherwise client input mode is one character at a time)	TERMSTATE_ECHO	client echoing is enabled (valid only in line mode)	TERMSTATE_EDIT	client line editing is enabled (valid only in line mode)	TERMSTATE_TRAPSIG	client signal trapping is enabled (valid only in line mode)	TERMSTATE_SOFTTAB	client tab expansion is enabled (valid only in line mode)	TERMSTATE_LITECHO	typed control characters are echoed literally by the client (valid only in line mode)	TERMSTATE_CRNL	map CR to NL on input (valid only in line mode)	TERMSTATE_BININ	binary on input.	TERMSTATE_BINOUT	binary on output.	TERMSTATE_FLOW	client flow control is enabled.
TERMSTATE_LINEMODE	client input mode is set to line mode (otherwise client input mode is one character at a time)																				
TERMSTATE_ECHO	client echoing is enabled (valid only in line mode)																				
TERMSTATE_EDIT	client line editing is enabled (valid only in line mode)																				
TERMSTATE_TRAPSIG	client signal trapping is enabled (valid only in line mode)																				
TERMSTATE_SOFTTAB	client tab expansion is enabled (valid only in line mode)																				
TERMSTATE_LITECHO	typed control characters are echoed literally by the client (valid only in line mode)																				
TERMSTATE_CRNL	map CR to NL on input (valid only in line mode)																				
TERMSTATE_BININ	binary on input.																				
TERMSTATE_BINOUT	binary on output.																				
TERMSTATE_FLOW	client flow control is enabled.																				

TERMSTATE_RESTARTANY restart flow on any character.

At initialization time, *telnetdInit* negotiates the following terminal state with the TELNET client:

TERMSTATE_FLOW | TERMSTATE_LINEMODE | TERMSTATE_CRNL |
TERMSTATE_ECHO | TERMSTATE_EDIT | TERMSTATE_TRAPSIG

RETURN VALUE

Upon successful completion, the functions return a value of zero. Otherwise they return -1 and set *errno* to indicate the error.

ERRORS

errno is set to EINVAL if one of the functions is called with an invalid *hdl* handle.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

SEE ALSO

`telnetdInit(3TELD)`

NAME	telnetdUser, telnetdPasswd – TELNET session authentication				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int telnetdUser(telnetdSessionHandle * hdl, char * buf, unsigned int nchar); int telnetdPasswd(telnetdSessionHandle * hdl, char * buf, unsigned int nchar);</pre>				
DESCRIPTION	<p>The <i>telnetdUser</i> function requests the name of the user connected to the TELNET session designated by the <i>hdl</i> handle. It then copies this name into the buffer pointed to by <i>buf</i>. No more than <i>nchar</i> characters are copied to <i>buf</i>, and the last character is always the null character.</p> <p>The <i>telnetdPasswd</i> function requests the password of the user connected to the TELNET session designated by the <i>hdl</i> handle. It then copies this password into the buffer pointed to by <i>buf</i>. No more than <i>nchar</i> characters are copied to <i>buf</i>, the last character is always the null character. Echoing is turned off while the password is entered by the user.</p>				
RETURN VALUE	Upon successful completion, the functions return a value of zero. Otherwise they return -1 and set <i>errno</i> to indicate the error.				
ERRORS	<p><i>errno</i> is set to EINVAL if one of the functions is called with an invalid <i>hdl</i> handle.</p> <p>The <i>telnetdUser</i> and <i>telnetdPasswd</i> functions may fail and set <i>errno</i> to any of the errors specified for the <i>telnetdReadLine</i> (3TELD) function.</p>				
ATTRIBUTES	<p>See <i>attributes</i>(5) for descriptions of the following attributes:</p> <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	telnetdReadLine(3TELD)				

NAME	telnetdWrite, telnetdFlush – write or flush a TELNET session				
SYNOPSIS	<pre>#include <arpa/telnetd.h> int telnetdWrite(telnetdSessionHandle * hdl, void * buf, unsigned int nchar); int telnetdFlush(telnetdSessionHandle * hdl);</pre>				
DESCRIPTION	<p>The <i>telnetdWrite</i> function attempts to write <i>nchar</i> characters from the buffer pointed to by <i>buf</i> to the TELNET session designated by the <i>hdl</i> handle.</p> <p>If the TELNET connection cannot accept data immediately and the connection socket has O_NONBLOCK set, <i>telnetdWrite</i> returns -1 and sets <i>errno</i> to EAGAIN . Otherwise, <i>telnetdWrite</i> blocks until data can be accepted.</p> <p>The O_NONBLOCK option may be set for the connection socket by calling <i>fcntl</i> (2POSIX) after the TELNET session has been initialized by <i>telnetdInit</i> .</p> <p>The <i>telnetdFlush</i> function causes any buffered data waiting to be written for the TELNET session designated by the <i>hdl</i> handle to be written to the TELNET connection socket.</p>				
RETURN VALUE	Upon successful completion, <i>telnetdWrite</i> returns the number of characters actually written; this number may be less than <i>nchar</i> . Upon successful completion, <i>telnetdFlush</i> returns 0. In case of failure, both functions return -1 and set <i>errno</i> to indicate the error.				
ERRORS	<p><i>errno</i> is set to EINVAL if <i>telnetdFlush</i> or <i>telnetdWrite</i> is called with an invalid <i>hdl</i> handle.</p> <p>The <i>telnetdWrite</i> and <i>telnetdFlush</i> functions may fail and set <i>errno</i> to any of the errors specified for the <i>write</i> (2POSIX) function.</p>				
ATTRIBUTES	<p>See <i>attributes</i>(5) for descriptions of the following attributes:</p> <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr> </thead> <tbody> <tr> <td>Interface Stability</td><td>Evolving</td></tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				
SEE ALSO	<i>telnetdGetTermState</i> (3TELD) , <i>telnetdRead</i> (3TELD)				

Index

I

inetAccept — wait for a new INET
connection 13, 15
inetBind — bind, close INET sockets 14, 16
inetClient — wait for a new INET
connection 13, 15
inetClose — bind, close INET sockets 14, 16
intro — introduction to the TELNETD
library 11

T

telnetdFlush — write or flush a TELNET
session 17, 29
telnetdFree — initialize or free a TELNET
session 18, 21

telnetdGetTermState — get or set TELNET
terminal state 19, 26
telnetdInit — initialize or free a TELNET
session 18, 21
telnetdPasswd — TELNET session
authentication 22,
28
telnetdRead — read from a TELNET session 23
telnetdReadLine — read a line of characters
from a TELNET session 25
telnetdSetTermState — get or set TELNET
terminal state 19, 26
telnetdUser — TELNET session authentication
22, 28
telnetdWrite — write or flush a TELNET
session 17, 29