Kerberos V5 UNIX User's Guide

Release: 1.0
Document Edition: 1.0
Last updated: 10 May 1999

MIT

1 Introduction

Kerberos V5 is an authentication system developed at MIT. Kerberos is named for the three-headed watchdog from Greek mythology, who guarded the entrance to the underworld.

Under Kerberos, a client (generally either a user or a service) sends a request for a ticket to the *Key Distribution Center* (KDC). The KDC creates a *ticket-granting ticket* (TGT) for the client, encrypts it using the client's password as the key, and sends the encrypted TGT back to the client. The client then attempts to decrypt the TGT, using its password. If the client successfully decrypts the TGT (*i.e.*, if the client gave the correct password), it keeps the decrypted TGT, which indicates proof of the client's identity.

The TGT, which expires at a specified time, permits the client to obtain additional tickets, which give permission for specific services. The requesting and granting of these additional tickets is user-transparent.

Since Kerberos negotiates authenticated, and optionally encrypted, communications between two points anywhere on the internet, it provides a layer of security that is not dependent on which side of a firewall either client is on. Since studies have shown that half of the computer security breaches in industry happen from *inside* firewalls, MIT's Kerberos V5 plays a vital role in maintaining your nework security.

The Kerberos V5 package is designed to be easy to use. Most of the commands are nearly identical to UNIX network programs you are already used to. Kerberos V5 is a *single-sign-on* system, which means that you have to type your password only once per session, and Kerberos does the authenticating and encrypting transparently.

1.1 What is a Ticket?

Your Kerberos *credentials*, or "*tickets*", are a set of electronic information that can be used to verify your identity. Your Kerberos tickets may be stored in a file, or they may exist only in memory.

The first ticket you obtain is a *ticket-granting ticket*, which permits you to obtain additional tickets. These additional tickets give you permission for specific services. The requesting and granting of these additional tickets happens transparently.

A good analogy for the ticket-granting ticket is a three-day ski pass that is good at four different resorts. You show the pass at whichever resort you decide to go to (until it expires), and you receive a lift ticket for that resort. Once you have the lift ticket, you can ski all you want at that resort. If you go to another resort the next day, you once again show your pass, and you get an additional lift ticket for the new resort. The difference is that the Kerberos V5 programs notice that you have the weekend ski pass, and get the lift ticket for you, so you don't have to perform the transactions yourself.

1.2 What is a Kerberos Principal?

A Kerberos *principal* is a unique identity to which Kerberos can assign tickets. By convention, a principal is divided into three parts: the *primary*, the *instance*, and the *realm*. The format of a typical Kerberos V5 principal is `primary/instance@REALM`.

- The *primary* is the first part of the principal. In the case of a user, it's the same as your username. For a host, the primary is the word `host`.
- The *instance* is an optional string that qualifies the primary. The instance is separated from the primary by a slash (`/`). In the case of a user, the instance is usually null, but a user might also have an additional principal, with an instance called '`admin`', which he/she uses to administrate a database. The principal `jennifer@ATHENA.MIT.EDU` is completely separate from the principal `jennifer/admin@ATHENA.MIT.EDU`, with a separate password, and separate permissions. In the case of a host, the instance is the fully qualified hostname, e.g., `daffodil.mit.edu`.
- The *realm* is your Kerberos realm. In most cases, your Kerberos realm is your domain name, in upper-case letters. For example, the machine `daffodil.mit.edu` would be in the realm `ATHENA.MIT.EDU`.

2 Kerberos V5 Tutorial

This tutorial is intended to familiarize you with the Kerberos V5 client programs. We will represent your prompt as "`shell%`". So an instruction to type the "`ls`" command would be represented as follows:

shell% `ls`

In these examples, we will use sample usernames, such as `jennifer` and `david`, sample hostnames, such as `daffodil` and `trillium`, and sample domain names, such as `mit.edu` and `fubar.org`. When you see one of these, substitute your username, hostname, or domain name accordingly.

2.1 Setting Up to Use Kerberos V5

Your system administrator will have installed the Kerberos V5 programs in whichever directory makes the most sense for your system. We will use `/usr/local` throughout this guide to refer to the top-level directory Kerberos V5 directory. We will therefor use `/usr/local/bin` to denote the location of the Kerberos V5 user programs. In your installation, the directory name may be different, but whatever the directory name is, you should make sure it is included in your path. You will probably want to put it *ahead of* the directories `/bin` and `/usr/bin` so you will get the Kerberos V5 network programs, rather than the standard UNIX versions, when you type their command names.

2.2 Ticket Management

On many systems, Kerberos is built into the login program, and you get tickets automatically when you log in. Other programs, such as `rsh`, `rcp`, `telnet`, and `rlogin`, can forward copies of your tickets to the remote host. Most of these programs also automatically destroy your tickets when they exit. However, MIT recommends that you explicitly destroy your Kerberos tickets when you are through with them, just to be sure. One way to help ensure that this happens is to add the `kdestroy` command to your `.logout` file. Additionally, if you are going to be away from your machine and are concerned about an intruder using your permissions, it is safest to either destroy all copies of your tickets, or use a screensaver that locks the screen.

2.2.1 Obtaining Tickets with kinit

If your site is using the Kerberos V5 login program, you will get Kerberos tickets automatically when you log in. If your site uses a different login program, you may need to explicitly obtain your Kerberos tickets, using the `kinit` program. Similarly, if your Kerberos tickets expire, use the `kinit` program to obtain new ones.

To use the `kinit` program, simply type `kinit` and then type your password at the prompt. For example, Jennifer (whose username is `jennifer`) works for Bleep, Inc. (a fictitious company with the domain name `mit.edu` and the Kerberos realm `ATHENA.MIT.EDU`). She would type:

```
shell% kinit
Password for jennifer@ATHENA.MIT.EDU:  <-- [Type jennifer's password here.]
shell%
```

If you type your password incorrectly, kinit will give you the following error message:

```
shell% kinit
Password for jennifer@ATHENA.MIT.EDU:  <-- [Type the wrong password here.]
kinit: Password incorrect
shell%
```

and you won't get Kerberos tickets.

Notice that `kinit` assumes you want tickets for your own username in your default realm.

Suppose Jennifer's friend David is visiting, and he wants to borrow a window to check his mail. David needs to get tickets for himself in his own realm, FUBAR.ORG.[1] He would type:

```
shell% kinit david@FUBAR.ORG
Password for david@FUBAR.ORG:  <-- [Type david's password here.]
shell%
```

David would then have tickets which he could use to log onto his own machine. Note that he typed his password locally on Jennifer's machine, but it never went over the network. Kerberos on the local host performed the authentication to the KDC in the other realm.

If you want to be able to forward your tickets to another host, you need to request *forwardable* tickets. You do this by specifying the `-f` option:

```
shell% kinit -f
Password for jennifer@ATHENA.MIT.EDU:  <-- [Type your password here.]
shell%
```

Note that `kinit` does not tell you that it obtained forwardable tickets; you can verify this using the `klist` command (see Section 2.2.2 [Viewing Your Tickets with klist], page 5).

---

[1] Note: the realm FUBAR.ORG must be listed in your computer's Kerberos configuration file, `/etc/krb5.conf`.

Normally, your tickets are good for your system's default ticket lifetime, which is ten hours on many systems. You can specify a different ticket lifetime with the '-l' option. Add the letter 's' to the value for seconds, 'm' for minutes, 'h' for hours, or 'd' for days.

For example, to obtain forwardable tickets for david@FUBAR.ORG that would be good for three hours, you would type:

```
shell% kinit -f -l 3h david@FUBAR.ORG
Password for david@FUBAR.ORG: <-- [Type david's password here.]
shell%
```

You cannot mix units; specifying a lifetime of '3h30m' would result in an error. Note also that most systems specify a maximum ticket lifetime. If you request a longer ticket lifetime, it will be automatically truncated to the maximum lifetime.

2.2.2 Viewing Your Tickets with klist

The klist command shows your tickets. When you first obtain tickets, you will have only the ticket-granting ticket. (See Section 1.1 [What is a Ticket?], page 1.) The listing would look like this:

```
shell% klist
Ticket cache: /tmp/krb5cc_ttypa
Default principal: jennifer@ATHENA.MIT.EDU

Valid starting     Expires            Service principal
06/07/96 19:49:21  06/08/96 05:49:19  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
shell%
```

The ticket cache is the location of your ticket file. In the above example, this file is named /tmp/krb5cc_ttypa. The default principal is your kerberos *principal*. (see Section 1.2 [What is a Kerberos Principal?], page 2)

The "valid starting" and "expires" fields describe the period of time during which the ticket is valid. The *service principal* describes each ticket. The ticket-granting ticket has the primary krbtgt, and the instance is the realm name.

Now, if jennifer connected to the machine daffodil.mit.edu, and then typed klist again, she would have gotten the following result:

```
shell% klist
Ticket cache: /tmp/krb5cc_ttypa
Default principal: jennifer@ATHENA.MIT.EDU

Valid starting     Expires            Service principal
06/07/96 19:49:21  06/08/96 05:49:19  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
06/07/96 20:22:30  06/08/96 05:49:19  host/daffodil.mit.edu@ATHENA.MIT.EDU
shell%
```

Here's what happened: when jennifer used telnet to connect to the host `daffodil.mit.edu`, the telnet program presented her ticket-granting ticket to the KDC and requested a host ticket for the host `daffodil.mit.edu`. The KDC sent the host ticket, which telnet then presented to the host `daffodil.mit.edu`, and she was allowed to log in without typing her password.

Suppose your Kerberos tickets allow you to log into a host in another domain, such as `trillium.fubar.org`, which is also in another Kerberos realm, `FUBAR.ORG`. If you telnet to this host, you will receive a ticket-granting ticket for the realm `FUBAR.ORG`, plus the new `host` ticket for `trillium.fubar.org`. `klist` will now show:

```
shell% klist
Ticket cache: /tmp/krb5cc_ttypa
Default principal: jennifer@ATHENA.MIT.EDU

Valid starting     Expires            Service principal
06/07/96 19:49:21  06/08/96 05:49:19  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
06/07/96 20:22:30  06/08/96 05:49:19  host/daffodil.mit.edu@ATHENA.MIT.EDU
06/07/96 20:24:18  06/08/96 05:49:19  krbtgt/FUBAR.ORG@ATHENA.MIT.EDU
06/07/96 20:24:18  06/08/96 05:49:19  host/trillium.fubar.org@ATHENA.MIT.EDU
shell%
```

You can use the `-f` option to view the *flags* that apply to your tickets. The flags are:

F **F**orwardable
f **f**orwarded
P **P**roxiable
p **p**roxy
D post**D**ateable
d post**d**ated
R **R**enewable
I **I**nitial
i **i**nvalid

Here is a sample listing. In this example, the user jennifer obtained her initial tickets (`'I'`), which are forwardable (`'F'`) and postdated (`'d'`) but not yet validated (`'i'`). (See Section 3.1 [kinit Reference], page 22 for more information about postdated tickets.)

```
shell% klist -f
Ticket cache: /tmp/krb5cc_320
Default principal: jennifer@ATHENA.MIT.EDU

Valid starting     Expires            Service principal
31 Jul 96 19:06:25  31 Jul 96 19:16:25  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
        Flags: FdiI
shell%
```

In the following example, the user david's tickets were forwarded ('f') to this host from another host. The tickets are reforwardable ('F').

```
shell% klist -f
Ticket cache: /tmp/krb5cc_p11795
Default principal: david@FUBAR.ORG

Valid starting     Expires            Service principal
07/31/96 11:52:29  07/31/96 21:11:23  krbtgt/FUBAR.ORG@FUBAR.ORG
        Flags: Ff
07/31/96 12:03:48  07/31/96 21:11:23  host/trillium.fubar.org@FUBAR.ORG
        Flags: Ff
shell%
```

2.2.3 Destroying Your Tickets with kdestroy

Your Kerberos tickets are proof that you are indeed yourself, and tickets can be stolen. If this happens, the person who has them can masquerade as you until they expire. For this reason, you should destroy your Kerberos tickets when you are away from your computer.

Destroying your tickets is easy. Simply type `kdestroy`.

```
shell% kdestroy
shell%
```

If `kdestroy` fails to destroy your tickets, it will beep and give an error message. For example, if `kdestroy` can't find any tickets to destroy, it will give the following message:

```
shell% kdestroy
kdestroy: No credentials cache file found while destroying cache
Ticket cache NOT destroyed!
shell%
```

2.3 Password Management

Your password is the only way Kerberos has of verifying your identity. If someone finds out your password, that person can masquerade as you—send email that comes from you, read, edit, or delete your files, or log into other hosts as you—and no one will be able to tell the difference. For this reason, it is important that you choose a good password (see Section 2.3.2 [Password Advice], page 8), and keep it secret. If you need to give access to your account to someone else, you can do so through Kerberos. (See Section 2.3.3 [Granting Access to Your Account], page 9.) You should *never* tell your password to anyone, including your system administrator, for any reason. You should change your password frequently, particularly any time you think someone may have found out what it is.

2.3.1 Changing Your Password

To change your Kerberos password, use the `kpasswd` command. It will ask you for your old password (to prevent someone else from walking up to your computer when you're not there and changing your password), and then prompt you for the new one twice. (The reason you have to type it twice is to make sure you have typed it correctly.) For example, user `david` would do the following:

shell% `kpasswd`
Old password for david: <- *Type your old password.*
New Password for david: <- *Type your new password.*
Verifying, please re-enter New Password for david: <- *Type the new password again.*
Password changed.
shell%

If david typed the incorrect old password, he would get the following message:

shell% `kpasswd`
Old password for david: <- *Type the incorrect old password.*
Incorrect old password.
shell%

If you make a mistake and don't type the new password the same way twice, `kpasswd` will ask you to try again:

shell% `kpasswd`
Old password for david: <- *Type the old password.*
New Password for david: <- *Type the new password.*
Verifying, please re-enter New Password for david: <- *Type a different new password.*
Mismatch - try again
New Password for david: <- *Type the new password.*
Verifying, please re-enter New Password for david: <- *Type the same new password.*
Password changed.
shell%

Once you change your password, it takes some time for the change to propagate through the system. Depending on how your system is set up, this might be anywhere from a few minutes to an hour or more. If you need to get new Kerberos tickets shortly after changing your password, try the new password. If the new password doesn't work, try again using the old one.

2.3.2 Password Advice

Your password can include almost any character you can type (except control keys and the "enter" key). A good password is one you can remember, but that no one else can easily guess. Examples of *bad* passwords are words that can be found in a dictionary, any common or popular name, especially a famous person (or cartoon character), your name or username in any form (*e.g.*, forward, backward, repeated twice, *etc.*), your spouse's, child's, or pet's name, your birth date, your social security number, and any sample password that appears in this (or any other) manual.

MIT recommends that your password be at least 6 characters long, and contain UPPER- and lower-case letters, numbers, and/or punctuation marks. Some passwords that would be good if they weren't listed in this manual include:

- some initials, like "GykoR-66." for "Get your kicks on Route 66."
- an easy-to-pronounce nonsense word, like "slaRooBey" or "krang-its"
- a misspelled phrase, like "2HotPeetzas!" or "ItzAGurl!!!"

Note: don't actually use any of the above passwords. They're only meant to show you how to make up a good password. Passwords that appear in a manual are the first ones intruders will try.

Kerberos V5 allows your system administrators to automatically reject bad passwords, based on whatever criteria they choose. For example, if the user `jennifer` chose a bad password, Kerberos would give an error message like the following:

```
shell% kpasswd
Old password for jennifer:   <- Type your old password here.
New Password for jennifer:   <- Type an insecure new password.
Verifying, please re-enter New Password for jennifer:   <- Type it again.

ERROR: Insecure password not accepted.  Please choose another.

kpasswd: Insecure password rejected while attempting to change password.
Please choose another password.

New Password for jennifer:   <- Type a good password here.
Verifying, please re-enter New Password for david:   <- Type it again.
Password changed.
shell%
```

Your system administrators can choose the message that is displayed if you choose a bad password, so the message you see may be different from the above example.

2.3.3 Granting Access to Your Account

If you need to give someone access to log into your account, you can do so through Kerberos, without telling the person your password. Simply create a file called `.k5login` in your home directory. This file should contain the Kerberos principal (See Section 1.2 [What is a Kerberos Principal?], page 2) of each person to whom you wish to give access. Each principal must be on a separate line. Here is a sample `.k5login` file:

```
jennifer@ATHENA.MIT.EDU
david@FUBAR.ORG
```

This file would allow the users `jennifer` and `david` to use your user ID, provided that they had Kerberos tickets in their respective realms. If you will be logging into other hosts across a network, you will want to include your own Kerberos principal in your `.k5login` file on each of these hosts.

Using a `.k5login` file is much safer than giving out your password, because:

- You can take access away any time simply by removing the principal from your `.k5login` file.
- Although the user has full access to your account on one particular host (or set of hosts if your `.k5login` file is shared, *e.g.*, over NFS), that user does not inherit your network privileges.
- Kerberos keeps a log of who obtains tickets, so a system administrator could find out, if necessary, who was capable of using your user ID at a particular time.

One common application is to have a `.k5login` file in `root`'s home directory, giving root access to that machine to the Kerberos principals listed. This allows system administrators to allow users to become root locally, or to log in remotely as `root`, without their having to give out the root password, and without anyone having to type the root password over the network.

2.4 Kerberos V5 Applications

Kerberos V5 is a *single-sign-on* system. This means that you only have to type your password once, and the Kerberos V5 programs do the authenticating (and optionally encrypting) for you. The way this works is that Kerberos has been built into each of a suite of network programs. For example, when you use a Kerberos V5 program to connect to a remote host, the program, the KDC, and the remote host perform a set of rapid negotiations. When these negotiations are completed, your program has proven your identity on your behalf to the remote host, and the remote host has granted you access, all in the space of a few seconds.

The Kerberos V5 applications are versions of existing UNIX network programs with the Kerberos features added.

2.4.1 Overview of Additional Features

The Kerberos V5 *network programs* are those programs that connect to another host somewhere on the internet. These programs include `rlogin`, `telnet`, `ftp`, `rsh`, `rcp`, and `ksu`. These programs have all of the original features of the corresponding non-Kerberos `rlogin`, `telnet`, `ftp`, `rsh`, `rcp`, and `su` programs, plus additional features that transparently use your Kerberos tickets for negotiating authentication and optional encryption with the remote host. In most cases, all you'll notice is that you no longer have to type your password, because Kerberos has already proven your identity.

The Kerberos V5 network programs allow you the options of forwarding your tickets to the remote host (if you obtained forwardable tickets with the `kinit` program; see Section 2.2.1 [Obtaining Tickets with kinit], page 4), and encrypting data transmitted between you and the remote host.

This section of the tutorial assumes you are familiar with the non-Kerberos versions of these programs, and highlights the Kerberos functions added in the Kerberos V5 package.

2.4.2 telnet

The Kerberos V5 `telnet` command works exactly like the standard UNIX telnet program, with the following Kerberos options added:

`-f, --forward`
: forwards a copy of your tickets to the remote host.

`--noforward`
: turns off forwarding of tickets to the remote host. (This option overrides any forwarding specified in your machine's configuration files.)

`-F, --forwardable`
: forwards a copy of your tickets to the remote host, and marks them re-forwardable from the remote host.

`--noforwardable`
: makes any forwarded tickets nonforwardable. (This option overrides any forwardability specified in your machine's configuration files.)

`-k` *realm*
: requests tickets for the remote host in the specified realm, instead of determining the realm itself.

`-K`
: uses your tickets to authenticate to the remote host, but does not log you in.

`-a`
: attempt automatic login using your tickets. `telnet` will assume the same username unless you explicitly specify another.

`-x, --encrypt`
: turns on encryption.

`--noencrypt`
: turns off encryption.

For example, if `david` wanted to use the standard UNIX telnet to connect to the machine `daffodil.mit.edu`, he would type:

```
shell% telnet daffodil.mit.edu
Trying 128.0.0.5 ...
Connected to daffodil.mit.edu.
Escape character is '^]'.

NetBSD/i386 (daffodil) (ttyp3)

login: david
Password:      <- david types his password here
Last login: Fri Jun 21 17:13:11 from trillium.fubar.org
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.   All rights reserved.

NetBSD 1.1: Tue May 21 00:31:42 EDT 1996

Welcome to NetBSD!
shell%
```

Note that the machine `daffodil.mit.edu` asked for `david`'s password. When he typed it, his password was sent over the network unencrypted. If an intruder were watching network traffic at the time, that intruder would know `david`'s password.

If, on the other hand, `jennifer` wanted to use the Kerberos V5 telnet to connect to the machine `trillium.fubar.org`, she could forward a copy of her tickets, request an encrypted session, and log on as herself as follows:

```
shell% telnet -a -f -x trillium.fubar.org
Trying 128.0.0.5...
Connected to trillium.fubar.org.
Escape character is '^]'.
[ Kerberos V5 accepts you as "jennifer@fubar.org" ]
[ Kerberos V5 accepted forwarded credentials ]
NetBSD 1.1: Tue May 21 00:31:42 EDT 1996

Welcome to NetBSD!
shell%
```

Note that `jennifer`'s machine used Kerberos to authenticate her to `trillium.fubar.org`, and logged her in automatically as herself. She had an encrypted session, a copy of her tickets already waiting for her, and she never typed her password.

If you forwarded your Kerberos tickets, `telnet` automatically destroys them when it exits. The full set of options to Kerberos V5 `telnet` are discussed in the Reference section of this manual. (see Section 3.5 [telnet Reference], page 27)

2.4.3 rlogin

The Kerberos V5 `rlogin` command works exactly like the standard UNIX rlogin program, with the following Kerberos options added:

`-f, --forward`
: forwards a copy of your tickets to the remote host.

`--noforward`
: turns off forwarding of tickets to the remote host. (This option overrides any forwarding specified in your machine's configuration files.)

`-F, --forwardable`
: forwards a copy of your tickets to the remote host, and marks them re-forwardable from the remote host.

`--noforwardable`
: makes any forwarded tickets nonforwardable. (This option overrides any forwardability specified in your machine's configuration files.)

`-k` *realm*
: requests tickets for the remote host in the specified realm, instead of determining the realm itself.

`-x, --encrypt`
: turns on encryption.

`--noencrypt`
: turns off encryption.

For example, if `david` wanted to use the standard UNIX rlogin to connect to the machine `daffodil.mit.edu`, he would type:

```
shell% rlogin daffodil.mit.edu -l david
Password:  <- david types his password here
Last login: Fri Jun 21 10:36:32 from :0.0
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.   All rights reserved.

NetBSD 1.1: Tue May 21 00:31:42 EDT 1996

Welcome to NetBSD!
shell%
```

Note that the machine `daffodil.mit.edu` asked for `david`'s password. When he typed it, his password was sent over the network unencrypted. If an intruder were watching network traffic at the time, that intruder would know `david`'s password.

If, on the other hand, `jennifer` wanted to use Kerberos V5 rlogin to connect to the machine `trillium.fubar.org`, she could forward a copy of her tickets, mark them as not forwardable from the remote host, and request an encrypted session as follows:

```
shell% rlogin trillium.fubar.org -f -x
This rlogin session is using DES encryption for all data transmissions.
Last login: Thu Jun 20 16:20:50 from daffodil
SunOS Release 4.1.4 (GENERIC) #2: Tue Nov 14 18:09:31 EST 1995
Not checking quotas. Try quota.real if you need them.
shell%
```

Note that `jennifer`'s machine used Kerberos to authenticate her to `trillium.fubar.org`, and logged her in automatically as herself. She had an encrypted session, a copy of her tickets were waiting for her, and she never typed her password.

If you forwarded your Kerberos tickets, `rlogin` automatically destroys them when it exits. The full set of options to Kerberos V5 `rlogin` are discussed in the Reference section of this manual. (see Section 3.6 [rlogin Reference], page 36)

2.4.4 FTP

The Kerberos V5 `FTP` program works exactly like the standard UNIX FTP program, with the following Kerberos features added:

`-k` *realm* requests tickets for the remote host in the specified realm, instead of determining the realm itself.

`-forward` requests that your tickets be forwarded to the remote host. The `-forward` argument must be the last argument on the command line.

`protect` *level*
(issued at the `ftp>` prompt) sets the protection level. "Clear" is no protection; "safe" ensures data integrity by verifying the checksum, and "private" encrypts the data. Encryption also ensures data integrity.

For example, suppose `jennifer` wants to get her `RMAIL` file from the directory `~jennifer/Mail`, on the host `daffodil.mit.edu`. She wants to encrypt the file transfer. The exchange would look like the following:

```
shell% ftp daffodil.mit.edu
Connected to daffodil.mit.edu.
220 daffodil.mit.edu FTP server (Version 5.60) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded
Name (daffodil.mit.edu:jennifer):
232 GSSAPI user jennifer@ATHENA.MIT.EDU is authorized as jennifer
230 User jennifer logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> protect private
200 Protection level set to Private.
ftp> cd ~jennifer/MAIL
250 CWD command successful.
ftp> get RMAIL
227 Entering Passive Mode (128,0,0,5,16,49)
150 Opening BINARY mode data connection for RMAIL (361662 bytes).
226 Transfer complete.
361662 bytes received in 2.5 seconds (1.4e+02 Kbytes/s)
ftp> quit
shell%
```

The full set of options to Kerberos V5 `FTP` are discussed in the Reference section of this manual. (see Section 3.7 [FTP Reference], page 38)

2.4.5 rsh

The Kerberos V5 `rsh` program works exactly like the standard UNIX rlogin program, with the following Kerberos features added:

`-f, --forward`
forwards a copy of your tickets to the remote host.

`--noforward`
turns off forwarding of tickets to the remote host. (This option overrides any forwarding specified in your machine's configuration files.)

`-F, --forwardable`
forwards a copy of your tickets to the remote host, and marks them re-forwardable from the remote host.

`--noforwardable`
makes any forwarded tickets nonforwardable. (This option overrides any forwardability specified in your machine's configuration files.)

`-k` *realm* requests tickets for the remote host in the specified realm, instead of determining the realm itself.

`-x, --encrypt`
turns on encryption.

`--noencrypt`
turns off encryption.

For example, if your Kerberos tickets allowed you to run programs on the host `trillium@fubar.org` as root, you could run the '`date`' program as follows:

```
shell% rsh trillium.fubar.org -l root -x date
This rsh session is using DES encryption for all data transmissions.
Fri Jun 21 17:06:12 EDT 1996
shell%
```

If you forwarded your Kerberos tickets, `rsh` automatically destroys them when it exits. The full set of options to Kerberos V5 `rsh` are discussed in the Reference section of this manual. (see Section 3.8 [rsh Reference], page 46)

2.4.6 rcp

The Kerberos V5 `rcp` program works exactly like the standard UNIX rcp program, with the following Kerberos features added:

`-k` *realm* requests tickets for the remote host in the specified realm, instead of determining the realm itself.

`-x, --encrypt`
turns on encryption.

For example, if you wanted to copy the file `/etc/motd` from the host `daffodil.mit.edu` into the current directory, via an encrypted connection, you would simply type:

```
shell% rcp -x daffodil.mit.edu:/etc/motd .
```

The `rcp` program negotiates authentication and encryption transparently. The full set of options to Kerberos V5 `rcp` are discussed in the Reference section of this manual. (see Section 3.9 [rcp Reference], page 48)

2.4.7 ksu

The Kerberos V5 `ksu` program replaces the standard UNIX su program. `ksu` first authenticates you to Kerberos. Depending on the configuration of your system, `ksu` may ask for your Kerberos password if authentication fails. *Note that you should never type your password if you are remotely logged in using an unencrypted connection.*

Once `ksu` has authenticated you, if your Kerberos principal appears in the target's `.k5login` file (see Section 2.3.3 [Granting Access to Your Account], page 9) or in the target's `.k5users` file (see below), it switches your user ID to the target user ID.

For example, `david` has put `jennifer`'s Kerberos principal in his `.k5login` file. If `jennifer` uses `ksu` to become `david`, the exchange would look like this. (To differentiate between the two shells, `jennifer`'s prompt is represented as `jennifer%` and `david`'s prompt is represented as `david%`.)

```
jennifer% ksu david
Account david: authorization for jennifer@ATHENA.MIT.EDU successful
Changing uid to david (3382)
david%
```

Note that the new shell has a copy of `jennifer`'s tickets. The ticket filename contains `david`'s UID with '`.1`' appended to it:

```
david% klist
Ticket cache: /tmp/krb5cc_3382.1
Default principal: jennifer@ATHENA.MIT.EDU

Valid starting       Expires              Service principal
31 Jul 96 21:53:01  01 Aug 96 07:52:53  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
31 Jul 96 21:53:39  01 Aug 96 07:52:53  host/daffodil.mit.edu@ATHENA.MIT.EDU
david%
```

If `jennifer` had not appeared in `david`'s `.k5login` file (and the system was configured to ask for a password), the exchange would have looked like this (assuming `david` has taken appropriate precautions in protecting his password):

> **jennifer%** `ksu david`
> **WARNING: Your password may be exposed if you enter it here and are logged in remotely using an unsecure (non-encrypted) channel.**
> **Kerberos password for david@ATHENA.MIT.EDU:** `<-` `jennifer` *types the wrong password here.*
> **ksu: Password incorrect**
> **Authentication failed.**
> **jennifer%**

Now, suppose `david` did not want to give `jennifer` full access to his account, but wanted to give her permission to list his files and use the "more" command to view them. He could create a `.k5users` file giving her permission to run only those specific commands.

The `.k5users` file is like the `.k5login` file, except that each principal is optionally followed by a list of commands. `ksu` will let those principals execute only the commands listed, using the `-e` option. `david`'s `.k5users` file might look like the following:

```
jennifer@ATHENA.MIT.EDU       /bin/ls /usr/bin/more
joeadmin@ATHENA.MIT.EDU         /bin/ls
joeadmin/admin@ATHENA.MIT.EDU   *
david@FUBAR.ORG
```

The above `.k5users` file would let `jennifer` run only the commands `/bin/ls` and `/usr/bin/more`. It would let `joeadmin` run only the command `/bin/ls` if he had regular tickets, but if he had tickets for his `admin` instance, `joeadmin/admin@ATHENA.MIT.EDU`, he would be able to execute any command. The last line gives `david` in the realm FUBAR.ORG permission to execute any command. (*I.e.*, having only a Kerberos principal on a line is equivalent to giving that principal permission to execute `*`.) This is so that david can allow himself to execute commands when he logs in, using Kerberos, from a machine in the realm FUBAR.ORG.

Then, when `jennifer` wanted to list his home directory, she would type:

> **jennifer%** `ksu david -e ls ~david`
> **Authenticated jennifer@ATHENA.MIT.EDU**
> **Account david: authorization for jennifer@ATHENA.MIT.EDU for execution of /bin/ls successful**
> **Changing uid to david (3382)**
> **Mail News Personal misc bin**
> **jennifer%**

If `jennifer` had tried to give a different command to `ksu`, it would have prompted for a password as with the previous example.

Note that unless the `.k5users` file gives the target permission to run any command, the user must use `ksu` with the `-e` *command* option.

The `ksu` options you are most likely to use are:

`-n` *principal*
: specifies which Kerberos principal you want to use for `ksu`. (*e.g.*, the user `joeadmin` might want to use his `admin` instance. See Section 1.1 [What is a Ticket?], page 1.)

`-c`
: specifies the location of your Kerberos credentials cache (ticket file).

`-C`
: specifies the location you want the Kerberos credentials cache (ticket file) to be for the target user ID.

`-k`
: tells `ksu` not to destroy your Kerberos tickets when `ksu` is finished.

`-f`
: requests forwardable tickets. (See Section 2.2.1 [Obtaining Tickets with kinit], page 4.) This is only applicable if `ksu` needs to obtain tickets.

`-l` *lifetime*
: sets the ticket lifetime. (See Section 2.2.1 [Obtaining Tickets with kinit], page 4.) This is only applicable if `ksu` needs to obtain tickets.

`-z`
: tells `ksu` to copy your Kerberos tickets only if the UID you are switching is the same as the Kerberos primary (either yours or the one specified by the `-n` option).

`-Z`
: tells `ksu` not to copy any Kerberos tickets to the new UID.

`-e` *command*
: tells `ksu` to execute *command* and then exit. See the description of the `.k5users` file above.

`-a` *text*
: (at the end of the command line) tells `ksu` to pass everything after '`-a`' to the target shell.

The full set of options to Kerberos V5 `ksu` are discussed in the Reference section of this manual. (see Section 3.10 [ksu Reference], page 49)

3 Kerberos V5 Reference

This section will include copies of the manual pages for the Kerberos V5 client programs. You can read the manual entry for any command by typing `man` *command*, where *command* is the name of the command for which you want to read the manual entry. For example, to read the `kinit` manual entry, you would type:

```
shell% man kinit
```

Note: To be able to view the Kerberos V5 manual pages on line, you may need to add the directory `/usr/local/man` to your MANPATH environment variable. (Remember to replace `/usr/local` with the top-level directory in which Kerberos V5 is installed.) For example, if you had the the following line in your `.login` file[1]:

```
setenv MANPATH /usr/local/man:/usr/man
```

and the Kerberos V5 man pages were in the directory `/usr/krb5/man`, you would change the line to the following:

```
setenv MANPATH /usr/krb5/man:/usr/local/man:/usr/man
```

[1] The MANPATH variable may be specified in a different initialization file, depending on your operating system. Some of the files in which you might specify environment variables include `.login`, `.profile`, or `.cshrc`.

3.1 kinit Reference

Reference Manual for `kinit`

KINIT(1) KINIT(1)

NAME

kinit − obtain and cache Kerberos ticket-granting ticket

SYNOPSIS

kinit [**−l** *lifetime*] [**−s** *start_time*] [**−v**] [**−p**] [**−f**] [**−k** [**−t** *keytab_file*]] [**−r** *renewable_life*] [**−R**] [**−c** *cache_name*] [**−S** *service_name*] [*principal*]

DESCRIPTION

kinit obtains and caches an initial ticket-granting ticket for *principal*.

OPTIONS

−l *lifetime*

requests a ticket with the lifetime *lifetime*. The value for *lifetime* must be followed immediately by one of the following delimiters:

s seconds
m minutes
h hours
d days

as in "kinit -l 90m". You cannot mix units; a value of '3h30m' will result in an error.

If the **−l** option is not specified, the default ticket lifetime (configured by each site) is used. Specifying a ticket lifetime longer than the maximum ticket lifetime (configured by each site) results in a ticket with the maximum lifetime.

−s *start_time*

requests a postdated ticket, valid starting at *start_time*. Postdated tickets are issued with the *invalid* flag set, and need to be fed back to the kdc before use.

−v requests that the ticket granting ticket in the cache (with the *invalid* flag set) be passed to the kdc for validation. If the ticket is within its requested time range, the cache is replaced with the validated ticket.

−p request proxiable tickets.

−f request forwardable tickets.

−r *renewable_life*

requests renewable tickets, with a total lifetime of *renewable_life*. The duration is in the same format as the **−l** option, with the same delimiters.

−R requests renewal of the ticket-granting ticket. Note that an expired ticket cannot be renewed, even if the ticket is still within its renewable life.

−k [**−t** *keytab_file*]

requests a host ticket, obtained from a key in the local host's *keytab* file. The name and location of the keytab file may be specified with the **−t** *keytab_file* option; otherwise the default name and location will be used.

−c *cache_name*

use *cache_name* as the credentials (ticket) cache name and location; if this option is not used, the default cache name and location are used.

The default credentials cache may vary between systems. If the **KRB5CCNAME** environment variable is set, its value is used to name the default ticket cache. Any existing contents of the cache are destroyed by *kinit*.

−S *service_name*

specify an alternate service name to use when getting initial tickets.

Reference Manual for `kinit`

ENVIRONMENT

Kinit uses the following environment variable:

KRB5CCNAME	Location of the credentials (ticket) cache.

FILES

/tmp/krb5cc_[uid]	default credentials cache ([uid] is the decimal UID of the user).
/etc/krb5.keytab	default location for the local host's **keytab** file.

SEE ALSO

klist(1), kdestroy(1), krb5(3)

3.2 klist Reference

Reference Manual for `klist`

KLIST(1) KLIST(1)

NAME

klist – list cached Kerberos tickets

SYNOPSIS

klist [**–e**] [[**–c**] [**–f**] [**–s**] [*cache_name*]] [**–k** [**–t**] [**–K**] [*keytab_name*]]

DESCRIPTION

Klist lists the Kerberos principal and Kerberos tickets held in a credentials cache, or the keys held in a **keytab** file.

OPTIONS

–e displays the encryption types of the session key and the ticket for each credential in the credential cache, or each key in the keytab file.

–c List tickets held in a credentials cache. This is the default if neither **–c** nor **–k** is specified.

–f shows the flags present in the credentials, using the following abbreviations:

F	**F**orwardable
f	**f**orwarded
P	**P**roxiable
p	**p**roxy
D	post**D**ateable
d	post**d**ated
R	**R**enewable
I	**I**nitial
i	**i**nvalid

–s causes **klist** to run silently (produce no output), but to still set the exit status according to whether it finds the credentials cache. The exit status is '0' if **klist** finds a credentials cache, and '1' if it does not.

–k List keys held in a **keytab** file.

–t display the time entry timestamps for each keytab entry in the keytab file.

–K display the value of the encryption key in each keytab entry in the keytab file.

If *cache_name* or *keytab_name* is not specified, klist will display the credentials in the default credentials cache or keytab file as appropriate. If the **KRB5CCNAME** environment variable is set, its value is used to name the default ticket cache.

ENVIRONMENT

Klist uses the following environment variable:

KRB5CCNAME Location of the credentials (ticket) cache.

FILES

/tmp/krb5cc_[uid] default location of the credentials cache ([uid] is the decimal UID of the user).

/etc/krb5.keytab default location of the **keytab** file.

SEE ALSO

kinit(1), kdestroy(1), krb5(3)

3.3 kdestroy Reference

Reference Manual for `kdestroy`

NAME

kdestroy – destroy Kerberos tickets

SYNOPSIS

kdestroy [**−q**] [**−c** *cache_name]*

DESCRIPTION

The *kdestroy* utility destroys the user's active Kerberos authorization tickets by writing zeros to the specified credentials cache that contains them. If the credentials cache is not specified, the default credentials cache is destroyed.

OPTIONS

−q Run quietly. Normally **kdestroy** beeps if it fails to destroy the user's tickets. The **−q** flag suppresses this behavior.

−c *cache_name*

use *cache_name* as the credentials (ticket) cache name and location; if this option is not used, the default cache name and location are used.

The default credentials cache may vary between systems. If the KRB5CCNAME environment variable is set, its value is used to name the default ticket cache.

Most installations recommend that you place the *kdestroy* command in your *.logout* file, so that your tickets are destroyed automatically when you log out.

ENVIRONMENT

Kdestroy uses the following environment variable:

KRB5CCNAME Location of the credentials (ticket) cache.

FILES

/tmp/krb5cc_[uid] default credentials cache ([uid] is the decimal UID of the user).

SEE ALSO

kinit(1), klist(1)

BUGS

Only the tickets in the specified credentials cache are destroyed. Separate ticket caches are used to hold root instance and password changing tickets. These should probably be destroyed too, or all of a user's tickets kept in a single credentials cache.

3.4 kpasswd Reference

Reference Manual for `kpasswd`

KPASSWD(1) KPASSWD(1)

NAME

kpasswd – change a user's Kerberos password

SYNOPSIS

kpasswd [*principal*]

DESCRIPTION

The *kpasswd* command is used to change a Kerberos principal's password. *Kpasswd* prompts for the current Kerberos password, which is used to obtain a **changepw** ticket from the KDC for the user's Kerberos realm. If **kpasswd** successfully obtains the **changepw** ticket, the user is prompted twice for the new password, and the password is changed.

If the principal is governed by a policy that specifies the length and/or number of character classes required in the new password, the new password must conform to the policy. (The five character classes are lower case, upper case, numbers, punctuation, and all other characters.)

OPTIONS

principal
: change the password for the Kerberos principal *principal*. Otherwise, the principal is derived from the identity of the user invoking the *kpasswd* command.

FILES

/tmp/tkt_kadm_[pid]
: temporary credentials cache for the lifetime of the password changing operation. ([pid] is the process-ID of the kpasswd process.)

SEE ALSO

kadmin(8), kadmind(8)

BUGS

If **kpasswd** is suspended, the changepw tickets may not be destroyed.

3.5 telnet Reference

Reference Manual for `telnet`

TELNET(1) TELNET(1)

NAME

telnet − user interface to the TELNET protocol

SYNOPSIS

telnet [**−8**] [**−E**] [**−F**] [**−K**] [**−L**] [**−S** *tos*] [**−X** *authtype*] [**−a**] [**−c**] [**−d**] [**−e** *escapechar*] [**−f**] [**−k** *realm*] [**−l** *user*] [**−n** *tracefile*] [**−r**] [**−x**] [*host* [*port*]]

DESCRIPTION

The **telnet** command is used to communicate with another host using the TELNET protocol. If **telnet** is invoked without the *host* argument, it enters command mode, indicated by its prompt (**telnet>**). In this mode, it accepts and executes the commands listed below. If it is invoked with arguments, it performs an **open** command with those arguments.

OPTIONS

−8 Specify an 8-bit data path. This causes an attempt to negotiate the TELNET BINARY option on both input and output.

−E Stop any character from being recognized as an escape character.

−F forward a *forwardable* copy of the local credentials to the remote system.

−K Specify no automatic login to the remote system.

−L Specify an 8-bit data path on output. This causes the BINARY option to be negotiated on output.

−S *tos* Set the IP type-of-service (TOS) option for the telnet connection to the value *tos,* which can be a numeric TOS value (in decimal, or a hex value preceded by 0x, or an octal value preceded by a leading 0) or, on systems that support it, a symbolic TOS name found in the /etc/iptos file.

−X *atype*

Disable the *atype* type of authentication.

−a Attempt automatic login. This sends the user name via the USER variable of the ENVIRON option, if supported by the remote system. The name used is that of the current user as returned by *getlogin*(2) if it agrees with the current user ID; otherwise it is the name associated with the user ID.

−c Disable the reading of the user's **.telnetrc** file. (See the **toggle skiprc** command on this man page.)

−d Set the initial value of the **debug** flag to TRUE

−e *escape char*

Set the initial **telnet** escape character to *escape char.* If *escape char* is omitted, then there will be no escape character.

−f forward a copy of the local credentials to the remote system.

−k *realm*

If Kerberos authentication is being used, request that telnet obtain tickets for the remote host in realm *realm* instead of the remote host's realm, as determined by *krb_realmofhost*(3).

−l *user* If the remote system understands the ENVIRON option, then *user* will be sent to the remote system as the value for the variable USER. This option implies the **−a** option. This option may also be used with the **open** command.

−n *tracefile*

Open *tracefile* for recording trace information. See the **set tracefile** command below.

−r Specify a user interface similar to *rlogin*(1). In this mode, the escape character is set to the tilde (~) character, unless modified by the **−e** option.

−x Turn on encryption of the data stream. When this option is turned on, **telnet** will exit with an error if authentication cannot be negotiated or if encryption cannot be turned on.

host Indicates the name, alias, or Internet address of the remote host.

Reference Manual for `telnet`

port Indicates a port number (address of an application). If the port is not specified, the default **telnet** port (23) is used.

When in rlogin mode, ˜ is the telnet escape character; a line of the form ˜. disconnects from the remote host. Similarly, the line ˜^Z suspends the telnet session. The line ˜^] escapes to the normal telnet escape prompt.

Once a connection has been opened, **telnet** will attempt to enable the TELNET LINEMODE option. If this fails, then **telnet** will revert to one of two input modes: either ''character at a time'' or ''old line by line,'' depending on what the remote system supports.

When LINEMODE is enabled, character processing is done on the local system, under the control of the remote system. When input editing or character echoing is to be disabled, the remote system will relay that information. The remote system will also relay changes to any special characters that happen on the remote system, so that they can take effect on the local system.

In ''character at a time'' mode, most text typed is immediately sent to the remote host for processing.

In ''old line by line'' mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The ''local echo character'' (initially ''^E'') may be used to turn off and on the local echo. (This would mostly be used to enter passwords without the password being echoed).

If the LINEMODE option is enabled, or if the **localchars** flag is TRUE (the default for ''old line by line''; see below), the user's **quit**, **intr**, and **flush** characters are trapped locally, and sent as TELNET protocol sequences to the remote side. If LINEMODE has ever been enabled, then the user's **susp** and **eof** are also sent as TELNET protocol sequences, and **quit** is sent as a TELNET ABORT instead of BREAK. There are options (see **toggle autoflush** and **toggle autosynch** below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence) and flush previous terminal input (in the case of **quit** and **intr**).

While connected to a remote host, **telnet** command mode may be entered by typing the **telnet** ''escape character'' (initially ''^]''). When in command mode, the normal terminal editing conventions are available.

The following **telnet** commands are available. Only enough of each command to uniquely identify it need be typed (this is also true for arguments to the **mode**, **set**, **toggle**, **unset**, **slc**, **environ**, and **display** commands).

auth *argument ...*
The auth command manipulates the information sent through the TELNET AUTHENTICATE option. Valid arguments for the auth command are as follows:

disable *type*
Disables the specified type of authentication. To obtain a list of available types, use the **auth disable ?** command.

enable *type*
Enables the specified type of authentication. To obtain a list of available types, use the **auth enable ?** command.

status Lists the current status of the various types of authentication.

close Close a TELNET session and return to command mode.

display *argument ...*
Displays some or all of the **set** and **toggle** values (see below).

encrypt *argument ...*
The encrypt command manipulates the information sent through the TELNET ENCRYPT option.

Note: Because of export controls, the TELNET ENCRYPT option is not supported outside of the United States and Canada.

Valid arguments for the encrypt command are as follows:

Reference Manual for `telnet`

disable *type* [**input**|**output**]
Disables the specified type of encryption. If you omit the input and output, both input and output are disabled. To obtain a list of available types, use the **encrypt disable ?** command.

enable *type]fP [***input**|**output***]*
Enables the specified type of encryption. If you omit input and output, both input and output are enabled. To obtain a list of available types, use the **encrypt enable ?** command.

input This is the same as the **encrypt start input** command.

−input This is the same as the **encrypt stop input** command.

output This is the same as the **encrypt start output** command.

−output
This is the same as the **encrypt stop output** command.

start [**input**|**output**]
Attempts to start encryption. If you omit **input** and **output**, both input and output are enabled. To obtain a list of available types, use the **encrypt enable ?** command.

status Lists the current status of encryption.

stop [**input**|**output**]
Stops encryption. If you omit input and output, encryption is on both input and output.

type *type*
Sets the default type of encryption to be used with later **encrypt start** or **encrypt stop** commands.

environ *arguments ...*
The **environ** command is used to manipulate the the variables that my be sent through the TELNET ENVIRON option. The initial set of variables is taken from the users environment, with only the DISPLAY and PRINTER variables being exported by default. The USER variable is also exported if the **−a** or **−l** options are used.

Valid arguments for the **environ** command are:

define *variable value*
Define the variable *variable* to have a value of *value*. Any variables defined by this command are automatically exported. The *value* may be enclosed in single or double quotes so that tabs and spaces may be included.

undefine *variable*
Remove *variable* from the list of environment variables.

export *variable*
Mark the variable *variable* to be exported to the remote side.

unexport *variable*
Mark the variable *variable* to not be exported unless explicitly asked for by the remote side.

list List the current set of environment variables. Those marked with a * will be sent automatically; other variables will only be sent if explicitly requested.

? Prints out help information for the **environ** command.

logout Sends the TELNET LOGOUT option to the remote side. This command is similar to a **close** command; however, if the remote side does not support the LOGOUT option, nothing happens. If, however, the remote side does support the LOGOUT option, this command should cause the remote side to close the TELNET connection. If the remote side also supports the concept of suspending a user's session for later reattachment, the logout argument indicates that you should terminate the

Reference Manual for `telnet`

session immediately.

mode *type*
Type is one of several options, depending on the state of the TELNET session. The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode will be entered.

character
Disable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then enter "character at a time" mode.

line Enable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then attempt to enter "old-line-by-line" mode.

isig (*−isig*)
Attempt to enable (disable) the TRAPSIG mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

edit (**−edit**)
Attempt to enable (disable) the EDIT mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

softtabs (**−softtabs**)
Attempt to enable (disable) the SOFT_TAB mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

litecho (**−litecho**)
Attempt to enable (disable) the LIT_ECHO mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

? Prints out help information for the **mode** command.

open *host* [**-a**] [[**−l**] *user*] [**−***port*]
Open a connection to the named host. If no port number is specified, **telnet** will attempt to contact a TELNET server at the default port. The host specification may be either a host name (see *hosts*(5) or an Internet address specified in the "dot notation" (see *inet*(3). After establishing a connection, the file **.telnetrc** in the user's home directory is opened. Lines beginning with a # are comment lines. Blank lines are ignored. Lines that begin without white space are the start of a machine entry. The first thing on the line is the name of the machine that is being connected to. The rest of the line, and successive lines that begin with white space are assumed to be **telnet** commands and are processed as if they had been typed in manually to the **telnet** command prompt.

−a Attempt automatic login. This sends the user name via the USER variable of the ENVIRON option, if supported by the remote system. The name used is that of the current user as returned by *getlogin*(2) if it agrees with the current user ID; otherwise it is the name associated with the user ID.

[**−l**] *user*
may be used to specify the user name to be passed to the remote system via the ENVIRON option.

−*port* When connecting to a non-standard port, **telnet** omits any automatic initiation of TELNET options. When the port number is preceded by a minus sign, the initial option negotiation is done.

quit Close any open TELNET session and exit **telnet**. An end of file (in command mode) will also close a session and exit.

send *arguments*
Sends one or more special character sequences to the remote host. The following are the arguments which may be specified (more than one argument may be specified at a time):

Reference Manual for `telnet`

abort Sends the TELNET ABORT (Abort processes) sequence.

ao Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output *from* the remote system *to* the user's terminal.

ayt Sends the TELNET AYT (Are You There) sequence, to which the remote system may or may not choose to respond.

brk Sends the TELNET BRK (Break) sequence, which may have significance to the remote system.

ec Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.

el Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

eof Sends the TELNET EOF (End Of File) sequence.

eor Sends the TELNET EOR (End of Record) sequence.

escape Sends the current escape character (initially "^").

ga Sends the TELNET GA (Go Ahead) sequence, which likely has no significance to the remote system.

getstatus
If the remote side supports the TELNET STATUS command, **getstatus** will send the subnegotiation to request that the server send its current option status.

ip Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

nop Sends the TELNET NOP (No OPeration) sequence.

susp Sends the TELNET SUSP (SUSPend process) sequence.

synch Sends the TELNET SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2BSD system -- if it doesn't work, a lower case "r" may be echoed on the terminal).

do *cmd*

dont *cmd*

will *cmd*

wont *cmd*
Sends the TELNET DO *cmd* sequence. *Cmd* can be either a decimal number between 0 and 255, or a symbolic name for a specific TELNET command. *Cmd* can also be either **help** or **?** to print out help information, including a list of known symbolic names.

? Prints out help information for the **send** command.

set *argument value*

unset *argument value*
The **set** command will set any one of a number of **telnet** variables to a specific value or to TRUE. The special value **off** turns off the function associated with the variable; this is equivalent to using the **unset** command. The **unset** command will disable or set to FALSE any of the specified functions. The values of variables may be interrogated with the **display** command. The variables which may be set or unset, but not toggled, are listed here. In addition, any of the variables for the **toggle** command may be explicitly set or unset using the **set** and **unset** commands.

ayt If **telnet** is in localchars mode, or LINEMODE is enabled, and the status character is typed, a TELNET AYT sequence (see **send ayt** preceding) is sent to the remote host. The initial value for the "Are You There" character is the terminal's status character.

Reference Manual for `telnet`

echo This is the value (initially "^E") which, when in "line by line" mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).

eof If **telnet** is operating in LINEMODE or "old line by line" mode, entering this character as the first character on a line will cause this character to be sent to the remote system. The initial value of the eof character is taken to be the terminal's **eof** character.

erase If **telnet** is in mode (see **toggle localchars** below), *and* if **telnet** is operating in "character at a time" mode, then when this character is typed, a TELNET EC sequence (see **send ec** above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's **erase** character.

escape This is the **telnet** escape character (initially "^[") which causes entry into **telnet** command mode (when connected to a remote system).

flushoutput
If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **flushoutput** character is typed, a TELNET AO sequence (see **send ao** above) is sent to the remote host. The initial value for the flush character is taken to be the terminal's **flush** character.

forw1

forw2 If **telnet** is operating in LINEMODE, these are the characters that, when typed, cause partial lines to be forwarded to the remote system. The initial value for the forwarding characters are taken from the terminal's eol and eol2 characters.

interrupt
If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **interrupt** character is typed, a TELNET IP sequence (see **send ip** above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's **intr** character.

kill If **telnet** is in **localchars** mode (see **toggle localchars** below), *and* if **telnet** is operating in "character at a time" mode, then when this character is typed, a TELNET EL sequence (see **send el** above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's **kill** character.

lnext If **telnet** is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal's **lnext** character. The initial value for the lnext character is taken to be the terminal's **lnext** character.

quit If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **quit** character is typed, a TELNET BRK sequence (see **send brk** above) is sent to the remote host. The initial value for the quit character is taken to be the terminal's **quit** character.

reprint If **telnet** is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal's **reprint** character. The initial value for the reprint character is taken to be the terminal's **reprint** character.

rlogin This is the rlogin escape character. If set, the normal **TELNET** escape character is ignored unless it is preceded by this character at the beginning of a line. This character, at the beginning of a line followed by a "." closes the connection; when followed by a ^Z it suspends the telnet command. The initial state is to disable the rlogin escape character.

start If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is taken to be the terminal's **start** character. The initial value for the kill character is taken to be the terminal's **start** character.

stop If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is taken to be the terminal's **stop** character. The initial value for the kill character is taken to be the terminal's **stop** character.

Reference Manual for `telnet`

susp If **telnet** is in **localchars** mode, or LINEMODE is enabled, and the **suspend** character is typed, a TELNET SUSP sequence (see **send susp** above) is sent to the remote host. The initial value for the suspend character is taken to be the terminal's **suspend** character.

tracefile
This is the file to which the output, caused by **netdata** or **option** tracing being TRUE, will be written. If it is set to "**−**", then tracing information will be written to standard output (the default).

worderase
If **telnet** is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal's **worderase** character. The initial value for the worderase character is taken to be the terminal's **worderase** character.

? Displays the legal **set** (**unset**) commands.

slc *state*
The **slc** command (Set Local Characters) is used to set or change the state of the the special characters when the TELNET LINEMODE option has been enabled. Special characters are characters that get mapped to **telnet** commands sequences (like **ip** or **quit**) or line editing characters (like **erase** and **kill**). By default, the local special characters are exported.

check Verify the current settings for the current special characters. The remote side is requested to send all the current special character settings, and if there are any discrepancies with the local side, the local side will switch to the remote value.

export Switch to the local defaults for the special characters. The local default characters are those of the local terminal at the time when **telnet** was started.

import Switch to the remote defaults for the special characters. The remote default characters are those of the remote system at the time when the TELNET connection was established.

? Prints out help information for the **slc** command.

status Show the current status of **telnet**. This includes the peer one is connected to, as well as the current mode.

toggle *arguments ...*
Toggle (between TRUE and FALSE) various flags that control how **telnet** responds to events. These flags may be set explicitly to TRUE or FALSE using the **set** and **unset** commands listed above. More than one argument may be specified. The state of these flags may be interrogated with the **display** command. Valid arguments are:

authdebug
Turns on debugging information for the authentication code.

autoflush
If **autoflush** and **localchars** are both TRUE , then when the **ao**, or **quit** characters are recognized (and transformed into TELNET sequences; see **set** above for details), **telnet** refuses to display any data on the user's terminal until the remote system acknowledges (via a TELNET TIMING MARK option) that it has processed those TELNET sequences. The initial value for this toggle is TRUE if the terminal user had not done an "stty noflsh", otherwise FALSE (see **stty**(1).

autodecrypt
When the TELNET ENCRYPT option is negotiated, by default the actual encryption (decryption) of the data stream does not start automatically. The autoencrypt (autodecrypt) command states that encryption of the output (input) stream should be enabled as soon as possible.

Note: Because of export controls, the TELNET ENCRYPT option is not supported outside the United States and Canada.

Reference Manual for `telnet`

autologin
If the remote side supports the TELNET AUTHENTICATION option **telnet** attempts to use it to perform automatic authentication. If the AUTHENTICATION option is not supported, the user's login name are propagated through the TELNET ENVIRON option. This command is the same as specifying the **–a** option on the **open** command.

autosynch
If **autosynch** and **localchars** are both TRUE, then when either the **intr** or **quit** characters is typed (see **set** above for descriptions of the **intr** and **quit** characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure *should* cause the remote system to begin throwing away all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is FALSE.

binary Enable or disable the TELNET BINARY option on both input and output.

inbinary
Enable or disable the TELNET BINARY option on input.

outbinary
Enable or disable the TELNET BINARY option on output.

crlf If this is TRUE, then carriage returns will be sent as <CR><LF>. If this is FALSE, then carriage returns will be send as <CR><NUL>. The initial value for this toggle is FALSE.

crmod Toggle carriage return mode. When this mode is enabled, most carriage return characters received from the remote host will be mapped into a carriage return followed by a line feed. This mode does not affect those characters typed by the user, only those received from the remote host. This mode is not very useful unless the remote host only sends carriage return, but never line feed. The initial value for this toggle is FALSE .

debug Toggles socket level debugging (useful only to the **super user**). The initial value for this toggle is FALSE .

encdebug
Turns on debugging information for the encryption code.

localchars
If this is TRUE , then the **flush**, **interrupt**, **quit**, **erase**, and **kill** characters (see **set** above) are recognized locally, and transformed into (hopefully) appropriate TELNET control sequences (respectively **ao**, **ip**, **brk**, **ec**, and **el**; see **send** above). The initial value for this toggle is TRUE in "old line by line" mode, and FALSE in "character at a time" mode. When the LINEMODE option is enabled, the value of **localchars** is ignored, and assumed to always be TRUE. If LINEMODE has ever been enabled, then **quit** is sent as **abort**, and **eof** and **suspend** are sent as **eof** and **susp**, see **send** above).

netdata
Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

options
Toggles the display of some internal **telnet** protocol processing (having to do with TELNET options). The initial value for this flag is FALSE .

prettydump
When the **netdata** flag is enabled, if **prettydump** is enabled the output from the **netdata** command will be formatted in a more user-readable format. Spaces are put between each character in the output, and the beginning of any TELNET escape sequence is preceded by a '*' to aid in locating them.

skiprc When the skiprc flag is TRUE, TELNET skips the reading of the .telnetrc file in the user's home directory when connections are opened. The initial value for this flag is FALSE.

Reference Manual for `telnet`

termdata
Toggles the display of all terminal data (in hexadecimal format). The initial value for this flag is FALSE.

verbose_encrypt
When the **verbose_encrypt** flag is TRUE, TELNET prints out a message each time encryption is enabled or disabled. The initial value for this toggle is FALSE. Note: Because of export controls, data encryption is not supported outside of the United States and Canada.

? Displays the legal **toggle** commands.

z Suspend **telnet**. This command only works when the user's shell is *csh*(1).

! [*command*]
Execute a single command in a subshell on the local system. If **command** is omitted, then an interactive subshell is invoked.

? *command*
Get help. With no arguments, prints a help summary. If a command is specified, will print the help information for just that command.

ENVIRONMENT

Telnet uses at least the HOME, SHELL, DISPLAY, and TERM environment variables. Other environment variables may be propagated to the other side via the TELNET ENVIRON option.

FILES

~/.telnetrc user-customized telnet startup values

~/.k5login
(on remote host) - file containing Kerberos principals that are allowed access.

HISTORY

The **Telnet** command appeared in 4.2BSD.

NOTES

On some remote systems, echo has to be turned off manually when in ‘‘old line by line’’ mode.

In ‘‘old line by line’’ mode or LINEMODE the terminal's **eof** character is only recognized (and sent to the remote system) when it is the first character on a line.

3.6 rlogin Reference

Reference Manual for `rlogin`

RLOGIN(1) RLOGIN(1)

NAME

rlogin – remote login

SYNOPSIS

rlogin *rhost* [**–e** *c*] [**–8**] [**–c**] [**–a**] [**–f**] [**–F**] [**–t** *termtype*] [**–n**] [**–7**] [**–d**] [**–k** *realm*] [**–x**] [**–L**] [**–l** *username*]

DESCRIPTION

Rlogin connects your terminal on the current local host system *lhost* to the remote host system *rhost.*

The version built to use Kerberos authentication is very similar to the standard Berkeley rlogin(1), except that instead of the *rhosts* mechanism, it uses Kerberos authentication to determine the authorization to use a remote account.

Each user may have a private authorization list in a file .k5login in his login directory. Each line in this file should contain a Kerberos principal name of the form *principal/instance@realm*. If the originating user is authenticated to one of the principals named in .k5login, access is granted to the account. If there is no /.k5login file, the principal will be granted access to the account according to the aname–>lname mapping rules. (See *krb5_anadd(8)* for more details.) Otherwise a login and password will be prompted for on the remote machine as in *login*(1). To avoid some security problems, the .k5login file must be owned by the remote user.

If there is some problem in marshaling the Kerberos authentication information, an error message is printed and the standard UCB rlogin is executed in place of the Kerberos rlogin.

A line of the form "~." disconnects from the remote host, where "~" is the escape character. Similarly, the line "~^Z" (where ^Z, control-Z, is the suspend character) will suspend the rlogin session. Substitution of the delayed-suspend character (normally ^Y) for the suspend character suspends the send portion of the rlogin, but allows output from the remote system.

The remote terminal type is the same as your local terminal type (as given in your environment TERM variable), unless the **–t** option is specified (see below). The terminal or window size is also copied to the remote system if the server supports the option, and changes in size are reflected as well.

All echoing takes place at the remote site, so that (except for delays) the rlogin is transparent. Flow control via ^S and ^Q and flushing of input and output on interrupts are handled properly.

OPTIONS

–8 allows an eight-bit input data path at all times; otherwise parity bits are stripped except when the remote side's stop and start characters are other than ^S/^Q. Eight-bit mode is the default.

–L allows the rlogin session to be run in litout mode.

–e*c* sets the escape character to *c*. There is no space separating this option flag and the new escape character.

–c require confirmation before disconnecting via "~."

–a force the remote machine to ask for a password by sending a null local username. This option has no effect unless the standard UCB rlogin is executed in place of the Kerberos rlogin (see above).

–f forward a copy of the local credentials to the remote system.

–F forward a *forwardable* copy of the local credentials to the remote system.

–t *termtype*
replace the terminal type passed to the remote host with *termtype*.

–n prevent suspension of rlogin via "~^Z" or "~^Y".

–7 force seven-bit transmissions.

–d turn on socket debugging (via *setsockopt*(2)) on the TCP sockets used for communication with the remote host.

Reference Manual for `rlogin`

−k request rlogin to obtain tickets for the remote host in realm *realm* instead of the remote host's realm as determined by *krb_realmofhost*(3).

−x turn on DES encryption for all data passed via the rlogin session. This significantly reduces response time and significantly increases CPU utilization.

SEE ALSO

rsh(1), kerberos(3), krb_sendauth(3), krb_realmofhost(3), rlogin(1) [UCB version]

FILES

~/.k5login (on remote host) - file containing Kerberos principals that are allowed access.

BUGS

More of the environment should be propagated.

3.7 FTP Reference

Reference Manual for FTP

FTP(1) FTP(1)

NAME

ftp – ARPANET file transfer program

SYNOPSIS

ftp [**–v**] [**–d**] [**–i**] [**–n**] [**–g**] [**–k** *realm*] [*host*] [**–forward**]

DESCRIPTION

FTP is the user interface to the ARPANET standard File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

OPTIONS

Options may be specified at the command line, or to the command interpreter.

–v Verbose option forces **ftp** to show all responses from the remote server, as well as report on data transfer statistics.

–n Restrains **ftp** from attempting "auto-login" upon initial connection. If auto-login is enabled, **ftp** attempts to authenticate to the FTP server by sending the AUTH command, using whichever authentication types are locally supported. Once an authentication type is accepted, an authentication protocol will proceed by issuing ADAT commands. **ftp** then will check the *.netrc* (see below) file in the user's home directory for an entry describing an account on the remote machine. If no entry exists, **ftp** will prompt for the remote machine login name (default is the user identity on the local machine), and, if necessary, prompt for a password and an account with which to login.

–i Turns off interactive prompting during multiple file transfers.

–d Enables debugging.

–g Disables file name globbing.

–k *realm*
When using Kerberos authentication, get tickets in *realm*.

–forward
Cause tickets to be forwarded to the remote host.

COMMANDS

The client host with which **ftp** is to communicate may be specified on the command line. If this is done, **ftp** will immediately attempt to establish a connection to an FTP server on that host; otherwise, **ftp** will enter its command interpreter and await instructions from the user. When **ftp** is awaiting commands from the user the prompt "ftp>" is provided to the user. The following commands are recognized by **ftp**:

! [*command*] [*args*]]
Invoke an interactive shell on the local machine. If there are arguments, the first is taken to be a command to execute directly, with the rest of the arguments as its arguments.

$ *macro-name* [*args*]
Execute the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.

account [*passwd*]
Supply a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user will be prompted for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]
Append a local file to a file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file after being altered by any **ntrans** or **nmap** setting. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

ascii Set the file transfer **type** to network ASCII . This is the default type.

bell Arrange that a bell be sounded after each file transfer command is completed.

Reference Manual for FTP

binary Set the file transfer **type** to support binary file transfer.

bye Terminate the FTP session with the remote server and exit **ftp**. An end of file will also terminate the session and exit.

case Toggle remote computer file name case mapping during **mget** commands. When **case** is on (default is off), remote computer file names with all letters in upper case are written in the local directory with the letters mapped to lower case.

cd *remote-directory*
Change the working directory on the remote machine to *remote-directory*.

cdup Change the remote machine working directory to the parent of the current remote machine working directory.

chmod *mode file-name*
Change the permission modes of the file *file-name* on the remote system to *mode*.

clear Set the protection level on data transfers to ‘‘clear’’. If no ADAT command succeeded, then this is the default protection level.

close Terminate the FTP session with the remote server, and return to the command interpreter. Any defined macros are erased.

cr Toggle carriage return stripping during ascii type file retrieval. Records are denoted by a carriage return/linefeed sequence during ascii type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single linefeed record delimiter. Records on non-UNIX remote systems may contain single linefeeds; when an ascii type transfer is made, these linefeeds may be distinguished from a record delimiter only when **cr** is off.

delete *remote-file*
Delete the file *remote-file* on the remote machine.

debug [*debug-value*]
Toggle debugging mode. If an optional *debug-value* is specified it is used to set the debugging level. When debugging is on, **ftp** prints each command sent to the remote machine, preceded by the string ‘−−>’

dir [*remote-directory*] [*local-file*]
Print a listing of the directory contents in the directory, *remote-directory*, and, optionally, placing the output in *local-file*. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **dir** output. If no directory is specified, the current working directory on the remote machine is used. If no local file is specified, or *local-file* is ‘**−**’, output comes to the terminal.

disconnect
A synonym for *close*.

form *format*
Set the file transfer **form** to *format*. The default format is ‘‘file’’.

get *remote-file* [*local-file*]
Retrieve the file *remote-file* and store it on the local machine. If the local file name is not specified, it is given the same name it has on the remote machine, subject to alteration by the current **case**, **ntrans**, and **nmap** settings. The current settings for **type**, **form**, **mode**, and **structure** are used while transferring the file.

glob Toggle filename expansion for **mdelete**, **mget**, and **mput**. If globbing is turned off with **glob**, the file name arguments are taken literally and not expanded. Globbing for **mput** is done as in *csh*(1). For **mdelete** and **mget**, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is likely to be different from expansion of the name of an ordinary file: the exact result depends on the foreign operating system and ftp server, and can be previewed by doing ‘mls remote-files −’ Note: **mget** and **mput** are not meant to

Reference Manual for FTP

transfer entire directory subtrees of files. That can be done by transferring a *tar*(1) archive of the subtree (in binary mode).

hash Toggle hash-sign ("#") printing for each data block transferred. The size of a data block is 1024 bytes.

help [*command*]
Print an informative message about the meaning of *command*. If no argument is given, **ftp** prints a list of the known commands.

idle [*seconds*]
Set the inactivity timer on the remote server to *seconds* seconds. If *seconds* is omitted, the current inactivity timer is printed.

lcd [*directory*]
Change the working directory on the local machine. If no *directory* is specified, the user's home directory is used.

ls [*remote-directory*] [*local-file*]
Print a listing of the contents of a directory on the remote machine. The listing includes any system-dependent information that the server chooses to include; for example, most UNIX systems will produce output from the command 'ls –l'. (See also **nlist**.) If *remote-directory* is left unspecified, the current working directory is used. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **ls** output. If no local file is specified, or if *local-file* is '**–**', the output is sent to the terminal.

macdef*macro-name*
Define a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consecutive newline characters in a file or carriage returns from the terminal) terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macros remain defined until a **close** command is executed. The macro processor interprets '$' and '\' as special characters. A '$' followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A '$' followed by an 'i' signals that macro processor that the executing macro is to be looped. On the first pass '$i' is replaced by the first argument on the macro invocation command line, on the second pass it is replaced by the second argument, and so on. A '\' followed by any character is replaced by that character. Use the '\' to prevent special treatment of the '$'.

mdelete [*remote-files*]
Delete *remote-files* on the remote machine.

mdir *remote-files local-file*
Like **dir**, except multiple remote files may be specified. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **mdir** output.

mget *remote-files*
Expand the *remote-files* on the remote machine and do a **get** for each file name thus produced. See **glob** for details on the filename expansion. Resulting file names will then be processed according to **case**, **ntrans**, and **nmap** settings. Files are transferred into the local working directory, which can be changed with 'lcd directory'; new local directories can be created with '! mkdir directory'.

mkdir *directory-name*
Make a directory on the remote machine.

mls *remote-files local-file*
Like **nlist**, except multiple remote files may be specified, and the *local-file* must be specified. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **mls** output.

Reference Manual for FTP

mode [*mode-name*]
Set the file transfer **mode** to *mode-name*. The default mode is ‘‘stream’’ mode.

modtime *file-name*
Show the last modification time of the file on the remote machine.

mput *local-files*
Expand wild cards in the list of local files given as arguments and do a **put** for each file in the resulting list. See **glob** for details of filename expansion. Resulting file names will then be processed according to **ntrans** and **nmap** settings.

newer *file-name*
Get the file only if the modification time of the remote file is more recent that the file on the current system. If the file does not exist on the current system, the remote file is considered **newer**. Otherwise, this command is identical to **get**.

nlist [*remote-directory*] [*local-file*]
Print a list of the files in a directory on the remote machine. If *remote-directory* is left unspecified, the current working directory is used. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **nlist** output. If no local file is specified, or if *local-file* is ‘**−**’, the output is sent to the terminal.

nmap [*inpattern outpattern*]
Set or unset the filename mapping mechanism. If no arguments are specified, the filename mapping mechanism is unset. If arguments are specified, remote filenames are mapped during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to non−UNIX remote computer with different file naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*. [*Inpattern*] is a template for incoming filenames (which may have already been processed according to the **ntrans** and **case** settings). Variable templating is accomplished by including the sequences ‘$1’, ‘$2’, ..., ‘$9’ in *inpattern*. Use ‘\’ to prevent this special treatment of the ‘$’ character. All other characters are treated literally, and are used to determine the **nmap** [*inpattern*] variable values. For example, given *inpattern* $1.$2 and the remote file name "mydata.data", $1 would have the value "mydata", and $2 would have the value "data". The *outpattern* determines the resulting mapped filename. The sequences ‘$1’, ‘$2’, *inpattern* template. The sequence ‘$0’ is replace by the original filename. Additionally, the sequence ‘[*seq1*, *seq2*]’ is replaced by [*seq1*] if *seq1* is not a null string; otherwise it is replaced by *seq2*. For example, the command

```
nmap $1.$2.$3 [$1,$2].[$2,file]
```

would yield the output filename "myfile.data" for input filenames "myfile.data" and "myfile.data.old", "myfile.file" for the input filename "myfile", and "myfile.myfile" for the input filename ".myfile". Spaces may be included in *outpattern*, as in the example: ‘nmap $1 sed "s/ *$//" > $1’. Use the ‘\’ character to prevent special treatment of the ‘$’,’[’,’]’, and ‘,’ characters.

ntrans [*inchars* [*outchars*]]
Set or unset the filename character translation mechanism. If no arguments are specified, the filename character translation mechanism is unset. If arguments are specified, characters in remote filenames are translated during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non-UNIX remote computer with different file naming conventions or practices. Characters in a filename matching a character in *inchars* are replaced with the corresponding character in *outchars*. If the character’s position in *inchars* is longer than the length of *outchars*, the character is deleted from the file name.

Reference Manual for FTP

open *host* [*port*] [**–forward**]
Establish a connection to the specified *host* FTP server. An optional port number may be supplied, in which case, **ftp** will attempt to contact an FTP server at that port. If the **auto-login** option is on (default), **ftp** will attempt to authenticate to the FTP server by sending the AUTH command, using whichever authentication types which are locally supported. Once an authentication type is accepted, an authentication protocol will proceed by issuing ADAT commands. **ftp** will also attempt to automatically log the user in to the FTP server (see below). If the **–forward** option is specified, **ftp** will forward a copy of the user's Kerberos tickets to the remote host.

private Set the protection level on data transfers to ‘‘private’’. Data transmissions are confidentiality and integrity protected by encryption. If no ADAT command succeeded, then the only possible level is ‘‘clear’’.

prompt
Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. If prompting is turned off (default is on), any **mget** or **mput** will transfer all files, and any **mdelete** will delete all files.

protect [*protection-level*]
Set the protection level on data transfers to *protection-level*. The valid protection levels are ‘‘clear’’ for unprotected data transmissions, ‘‘safe’’ for data transmissions integrity protected by cryptographic checksum, and ‘‘private’’ for data transmissions confidentiality and integrity protected by encryption. If no ADAT command succeeded, then the only possible level is ‘‘clear’’. If no level is specified, the current level is printed. The default protection level is ‘‘clear’’.

proxy *ftp-command*
Execute an ftp command on a secondary control connection. This command allows simultaneous connection to two remote ftp servers for transferring files between the two servers. The first **proxy** command should be an **open ,** to establish the secondary control connection. Enter the command "proxy ?" to see other ftp commands executable on the secondary connection. The following commands behave differently when prefaced by **proxy**: **open** will not define new macros during the auto-login process, **close** will not erase existing macro definitions, **get** and **mget** transfer files from the host on the primary control connection to the host on the secondary control connection, and **put**, **mput**, and **append** transfer files from the host on the secondary control connection to the host on the primary control connection. Third party file transfers depend upon support of the ftp protocol PASV command by the server on the secondary control connection.

put *local-file* [*remote-file*]
Store a local file on the remote machine. If *remote-file* is left unspecified, the local file name is used after processing according to any **ntrans** or **nmap** settings in naming the remote file. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

pwd Print the name of the current working directory on the remote machine.

quit A synonym for **bye**.

quote *arg1* [*arg2*] [...]
The arguments specified are sent, verbatim, to the remote FTP server.

recv *remote-file* [*local-file*]
A synonym for get.

reget *remote-file* [*local-file*]
Reget acts like get, except that if *local-file* exists and is smaller than *remote-file*, *local-file* is presumed to be a partially transferred copy of *remote-file* and the transfer is continued from the apparent point of failure. This command is useful when transferring very large files over networks that are prone to dropping connections.

remotehelp [*command-name*]
Request help from the remote FTP server. If a *command-name* is specified it is supplied to the server as well.

Reference Manual for FTP

remotestatus [*file-name*]
With no arguments, show status of remote machine. If *file-name* is specified, show status of *file-name* on remote machine.

rename [*from*] [*to*]
Rename the file *from* on the remote machine, to the file *to*.

reset Clear reply queue. This command re-synchronizes command/reply sequencing with the remote ftp server. Resynchronization may be necessary following a violation of the ftp protocol by the remote server.

restart *marker*
Restart the immediately following **get** or **put** at the indicated *marker*. On UNIX systems, marker is usually a byte offset into the file.

rmdir *directory-name*
Delete a directory on the remote machine.

runique
Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a **get** or **mget** command, a ".1" is appended to the name. If the resulting name matches another existing file, a ".2" is appended to the original name. If this process continues up to ".99", an error message is printed, and the transfer does not take place. The generated unique filename will be reported. Note that **runique** will not affect local files generated from a shell command (see below). The default value is off.

safe Set the protection level on data transfers to ‘‘safe’’. Data transmissions are integrity-protected by cryptographic checksum. If no ADAT command succeeded, then the only possible level is ‘‘clear’’.

send *local-file* [*remote-file*]
A synonym for put.

sendport
Toggle the use of PORT commands. By default, **ftp** will attempt to use a PORT command when establishing a connection for each data transfer. The use of PORT commands can prevent delays when performing multiple file transfers. If the PORT command fails, **ftp** will use the default data port. When the use of PORT commands is disabled, no attempt will be made to use PORT commands for each data transfer. This is useful for certain FTP implementations which do ignore PORT commands but, incorrectly, indicate they've been accepted.

site *arg1* [*arg2*] [...]
The arguments specified are sent, verbatim, to the remote FTP server as a SITE command.

size *file-name*
Return size of *file-name* on remote machine.

status Show the current status of **ftp**.

struct *struct-name*
Set the file transfer *structure* to *struct-name*. By default ‘‘stream’’ structure is used.

sunique
Toggle storing of files on remote machine under unique file names. Remote ftp server must support ftp protocol STOU command for successful completion. The remote server will report unique name. Default value is off.

system Show the type of operating system running on the remote machine.

tenex Set the file transfer type to that needed to talk to TENEX machines.

trace Toggle packet tracing.

type [*type-name*]
Set the file transfer **type** to *type-name*. If no type is specified, the current type is printed. The default type is network ASCII.

Reference Manual for FTP

umask [*newmask*]
Set the default umask on the remote server to *newmask*. If *newmask* is omitted, the current umask is printed.

user *user-name* [*password*] [*account*]
Identify yourself to the remote FTP server. If the *password* is not specified and the server requires it, **ftp** will prompt the user for it (after disabling local echo). If an *account* field is not specified, and the FTP server requires it, the user will be prompted for it. If an *account* field is specified, an account command will be relayed to the remote server after the login sequence is completed if the remote server did not require it for logging in. Unless **ftp** is invoked with ‘‘auto-login’’ disabled, this process is done automatically on initial connection to the FTP server.

verbose
Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose is on.

? [*command***]**
A synonym for help.

Command arguments which have embedded spaces may be quoted with quote ‘"’ marks.

ABORTING A FILE TRANSFER

To abort a file transfer, use the terminal interrupt key (usually Ctrl-C). Sending transfers will be immediately halted. Receiving transfers will be halted by sending a FTP protocol ABOR command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server's support for ABOR processing. If the remote server does not support the ABOR command, an ‘ftp>’ prompt will not appear until the remote server has completed sending the requested file.

The terminal interrupt key sequence will be ignored when **ftp** has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the ABOR processing described above, or from unexpected behavior by the remote server, including violations of the ftp protocol. If the delay results from unexpected remote server behavior, the local **ftp** program must be killed by hand.

FILE NAMING CONVENTIONS

Files specified as arguments to **ftp** commands are processed according to the following rules.

1. If the file name ‘**-**’ is specified, *stdin* (for reading) or *stdout* (for writing) is used.
2. If the first character of the file name is ‘|’, the remainder of the argument is interpreted as a shell command. **Ftp** then forks a shell, using *popen*(3) with the argument supplied, and reads from (writes to) stdout (stdin). If the shell command includes spaces, the argument must be quoted; e.g. ‘‘" ls -lt"’’. A particularly useful example of this mechanism is: ‘‘dir more’’.
3. Failing the above checks, if ‘‘globbing’’ is enabled, local file names are expanded according to the rules used in *csh*(1); c.f. the **glob** command. If the **ftp** command expects a single local file (.e.g. **put**), only the first filename generated by the ‘‘globbing’’ operation is used.
4. For **mget** commands and **get** commands with unspecified local file names, the local filename is the remote filename, which may be altered by a **case**, **ntrans**, or **nmap** setting. The resulting filename may then be altered if **runique** is on.
5. For **mput** commands and **put** commands with unspecified remote file names, the remote filename is the local filename, which may be altered by a **ntrans** or **nmap** setting. The resulting filename may then be altered by the remote server if **sunique** is on.

FILE TRANSFER PARAMETERS

The FTP specification specifies many parameters which may affect a file transfer. The **type** may be one of ‘‘ascii’’, ‘‘image’’ (binary), ‘‘ebcdic’’, and ‘‘local byte size’’ (mostly for PDP-10's and PDP-20's). **Ftp** supports the ascii and image types of file transfer, plus local byte size 8 for **tenex** mode transfers.

Ftp supports only the default values for the remaining file transfer parameters: **mode**, **form**, and **struct**.

Reference Manual for FTP

FTP(1) FTP(1)

THE .netrc FILE

The *.netrc* file contains login and initialization information used by the auto-login process. It resides in the user's home directory. The following tokens are recognized; they may be separated by spaces, tabs, or new-lines:

machine *name*
Identify a remote machine *name*. The auto-login process searches the *.netrc* file for a **machine** token that matches the remote machine specified on the **ftp** command line or as an **open** command argument. Once a match is made, the subsequent *.netrc* tokens are processed, stopping when the end of file is reached or another **machine** or a **default** token is encountered.

default This is the same as **machine** *name* except that **default** matches any name. There can be only one **default** token, and it must be after all **machine** tokens. This is normally used as:

    default login anonymous password user@site

thereby giving the user *automatic* anonymous ftp login to machines not specified in *.netrc*. This can be overridden by using the **–n** flag to disable auto-login.

login *name*
Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified *name*.

password *string*
Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note that if this token is present in the *.netrc* file for any user other than *anonymous*, **ftp** will abort the auto-login process if the *.netrc* is readable by anyone besides the user.

account *string*
Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an ACCT command if it does not.

macdef *name*
Define a macro. This token functions like the **ftp macdef** command functions. A macro is defined with the specified name; its contents begin with the next *.netrc* line and continue until a null line (consecutive new-line characters) is encountered. If a macro named **init** is defined, it is automatically executed as the last step in the auto-login process.

ENVIRONMENT

Ftp utilizes the following environment variables.

HOME For default location of a *.netrc* file, if one exists.

SHELL For default shell.

SEE ALSO

ftpd(8)

Lunt, S. J., FTP Security Extensions, Internet Draft, November 1993.

HISTORY

The **ftp** command appeared in 4.2BSD.

BUGS

Correct execution of many commands depends upon proper behavior by the remote server.

An error in the treatment of carriage returns in the 4.2BSD ascii-mode transfer code has been corrected. This correction may result in incorrect transfers of binary files to and from 4.2BSD servers using the ascii type. Avoid this problem by using the binary image type.

3.8 rsh Reference

Reference Manual for `rsh`

RSH() RSH()

NAME

rsh – remote shell

SYNOPSIS

rsh *host* [**–l** *username*] [**–n**] [**–d**] [**–k** *realm*] [**–f** | **–F**] [**–x**] *command*

DESCRIPTION

Rsh connects to the specified *host,* and executes the specified *command*. **Rsh** copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. This implementation of **rsh** will accept any port for the standard error stream. Interrupt, quit and terminate signals are propagated to the remote command; *rsh* normally terminates when the remote command does.

Each user may have a private authorization list in a file .k5login in his login directory. Each line in this file should contain a Kerberos principal name of the form *principal/instance@realm*. If there is a ~/.k5login file, then access is granted to the account if and only if the originater user is authenticated to one of the princiapls named in the ~/.k5login file. Otherwise, the originating user will be granted access to the account if and only if the authenticated principal name of the user can be mapped to the local account name using the aname -> lname mapping rules (see *krb5_anadd*(8) for more details).

OPTIONS

–l *username*
: sets the remote username to *username*. Otherwise, the remote username will be the same as the local username.

–x causes the network session traffic to be encrypted.

–f cause nonforwardable Kerberos credentials to be forwarded to the remote machine for use by the specified *command*. They will be removed when *command* finishes. This option is mutually exclusive with the **–F** option.

–F cause *forwardable* Kerberos credentials to be forwarded to the remote machine for use by the specified *command*. They will be removed when *command* finishes. This option is mutually exclusive with the **–f** option.

–k *realm*
: causes *rsh* to obtain tickets for the remote host in *realm* instead of the remote host's realm as determined by *krb_realmofhost*(3).

–d turns on socket debugging (via *setsockopt*(2)) on the TCP sockets used for communication with the remote host.

–n redirects input from the special device */dev/null* (see the BUGS section below).

If you omit *command*, then instead of executing a single command, you will be logged in on the remote host using *rlogin*(1).

Shell metacharacters which are not quoted are interpreted on the local machine, while quoted metacharacters are interpreted on the remote machine. Thus the command

rsh otherhost cat remotefile >> localfile

appends the remote file *remotefile* to the local file *localfile*, while

rsh otherhost cat remotefile ">>" otherremotefile

appends *remotefile* to *otherremotefile*.

FILES

/etc/hosts

~/.k5login (on remote host) - file containing Kerberos principals that are allowed access.

SEE ALSO

rlogin(1), kerberos(3), krb_sendauth(3), krb_realmofhost(3)

Reference Manual for `rsh`

BUGS

If you are using *csh*(1) and put a *rsh*(1) in the background without redirecting its input away from the terminal, it will block even if no reads are posted by the remote command. If no input is desired you should redirect the input of *rsh* to /dev/null using the **−n** option.

You cannot run an interactive command (like *rogue*(6) or *vi*(1)); use *rlogin*(1).

Stop signals stop the local *rsh* process only; this is arguably wrong, but currently hard to fix for reasons too complicated to explain here.

3.9 rcp Reference

Reference Manual for `rcp`

RCP(1) RCP(1)

NAME

rcp – remote file copy

SYNOPSIS

rcp [**–p**] [**–x**] [**–k** *realm*] [**–D** *port*] [**–N**] *file1 file2*

rcp [**–p**] **[–x] [–k** *realm*] **[–r] [–D** *port*] **[–N]** *file ... directory*

DESCRIPTION

Rcp copies files between machines. Each *file* or *directory* argument is either a remote file name of the form ‘‘rhost:path’’, or a local file name (containing no ‘:’ characters, or a ‘/’ before any ‘:’s).

By default, the mode and owner of *file2* are preserved if it already existed; otherwise the mode of the source file modified by the *umask*(2) on the destination host is used.

If *path* is not a full path name, it is interpreted relative to your login directory on *rhost*. A *path* on a remote host may be quoted (using \, ", or ´) so that the metacharacters are interpreted remotely.

Rcp does not prompt for passwords; it uses Kerberos authentication when connecting to *rhost*. Each user may have a private authorization list in a file .k5login in his login directory. Each line in this file should contain a Kerberos principal name of the form *principal/instance@realm*. If there is a ˜/.k5login file, then access is granted to the account if and only if the originater user is authenticated to one of the principals named in the ˜/.k5login file. Otherwise, the originating user will be granted access to the account if and only if the authenticated principal name of the user can be mapped to the local account name using the aname -> lname mapping rules (see *krb5_anadd*(8) for more details).

OPTIONS

–p attempt to preserve (duplicate) the modification times and modes of the source files in the copies, ignoring the *umask*.

–x encrypt all information transferring between hosts.

–k *realm*
obtain tickets for the remote host in *realm* instead of the remote host's realm as determined by *krb_realmofhost*(3).

–r if any of the source files are directories, copy each subtree rooted at that name; in this case the destination must be a directory.

–D *port*
connect to port *port* on the remote machine.

–N use a network connection, even when copying files on the local machine (used for testing purposes).

Rcp handles third party copies, where neither source nor target files are on the current machine. Hostnames may also take the form ‘‘rname@rhost’’ to use *rname* rather than the current user name on the remote host.

FILES

˜/.k5login (on remote host) - file containing Kerberos principals that are allowed access.

SEE ALSO

cp(1), ftp(1), rsh(1), rlogin(1), kerberos(3), krb_getrealm(3), rcp(1) [UCB version]

BUGS

Rcp doesn't detect all cases where the target of a copy might be a file in cases where only a directory should be legal.

Rcp is confused by any output generated by commands in a .login, .profile, or .cshrc file on the remote host.

Kerberos is only used for the first connection of a third-party copy; the second connection uses the standard Berkeley rcp protocol.

3.10 ksu Reference

Reference Manual for `ksu`

KSU(1) KSU(1)

NAME

ksu − Kerberized super-user

SYNOPSIS

ksu [*target_user*] [**−n** *target_principal_name*] [**−c** *source_cache_name*] [**−C** *target_cache_name*] [**−k**] [**−D**] [**−r** *time*] [**−pf**] [**−l** *lifetime*] [**−zZ**] [**−q**] [**−e** *command* [*args ...*]] [**−a** [*args ...*]]

REQUIREMENTS

Must have Kerberos version 5 installed to compile ksu. Must have a Kerberos version 5 server running to use ksu.

DESCRIPTION

ksu is a Kerberized version of the su program that has two missions: one is to securely change the real and effective user ID to that of the target user, the other is to create a new security context. For the sake of clarity all references to, and attributes of the user invoking the program will start with 'source' (e.g. source user, source cache, etc.). Likewise all references to and attributes of the target account, will start with 'target'.

AUTHENTICATION

To fulfill the first mission, ksu operates in two phases: authentication and authorization. Resolving the target principal name is the first step in authentication. The user can either specify his principal name with the **−n** option (e.g. **−n** jqpublic@USC.EDU) or a default principal name will be assigned using a heuristic described in the OPTIONS section (see **−n** option). The target user name must be the first argument to ksu, if not specified root is the default. If '.' is specified then the target user will be the source user (e.g. ksu .). If the source user is root or the target user is the source user, no authentication or authorization takes place. Otherwise, ksu looks for an appropriate Kerberos ticket in the source cache.

The ticket can either be for the end-server or a ticket granting ticket (TGT) for the target principal's realm. If the ticket for the end server is already in the cache, it's, decrypted and verified. If it's not in the cache but the TGT is, TGT is used to obtain the ticket for the end-server. The end-server ticket is then verified. If neither ticket is in the cache, but ksu is compiled with the GET_TGT_VIA_PASSWD define, the user will be prompted for a Kerberos password which will then be used to get a TGT. If the user is logged in remotely and does not have a secure channel, the password may be exposed. If neither ticket is in the cache and GET_TGT_VIA_PASSWD is not defined, authentication fails.

AUTHORIZATION

This section describes authorization of the source user when ksu is invoked without the **−e** option. For a description of the **−e** option, see the OPTIONS section.

Upon successful authentication, ksu checks whether the target principal is authorized to access the target account. In the target user's home directory, ksu attempts to access two authorization files: .k5login and .k5users. In the .k5login file each line contains the name of a principal that is authorized to access the account.

For example: jqpublic@USC.EDU
jqpublic/secure@USC.EDU
jqpublic/admin@USC.EDU

The format of .k5users is the same, accept the principal name may be followed by a list of commands that the principal is authorized to execute. (see the **−e** option in the OPTIONS section for details).

Thus if the target principal name is found in the .k5login file the source user is authorized to access the target account. Otherwise ksu looks in the .k5users file. If the target principal name is found without any trailing commands or followed only by '*' then the source user is authorized. If either .k5login or .k5users exist but an appropriate entry for the target principal does not exist then access is denied. If neither file exists then the principal will be granted access to the account according to the aname−>lname mapping rules (see *krb5_anadd(8)* for more details). Otherwise, authorization fails.

EXECUTION OF THE TARGET SHELL

Upon successful authentication and authorization, ksu proceeds in a similar fashion to su. The environment is unmodified with the exception of USER, HOME and SHELL variables. If the target user is not root,

Reference Manual for `ksu`

USER gets set to the target user name. Otherwise USER remains unchanged. Both HOME and SHELL are set to the target login's default values. In addition, the environment variable KRB5CCNAME gets set to the name of the target cache. The real and effective user ID are changed to that of the target user. The target user's shell is then invoked (the shell name is specified in the password file). Upon termination of the shell, ksu deletes the target cache (unless ksu is invoked with **–k**
or ' **–C .' options).** This is implemented by first doing a fork and then an exec, instead of just exec, as done by su.

CREATING A NEW SECURITY CONTEXT

Ksu can be used to create a new security context for the target program (either the target shell, or command specified via the -e option). The target program inherits a set of credentials from the source user. By default, this set includes all of the credentials in the source cache plus any additional credentials obtained during authentication. The source user is able to limit the credentials in this set by using -z or -Z option. -z restricts the copy of tickets from the source cache to the target cache to only the tickets where client == the target principal name. The -Z option provides the target user with a fresh target cache (no creds in the cache). Note that for security reasons, when the source user is root and target user is non-root, -z option is the default mode of operation.

While no authentication takes place if the source user is root or is the same as the target user, additional tickets can still be obtained for the target cache. If -n is specified and no credentials can be copied to the target cache, the source user is prompted for a Kerberos password (unless -Z specified or GET_TGT_VIA_PASSWD is undefined). If successful, a TGT is obtained from the Kerberos server and stored in the target cache. Otherwise, if a password is not provided (user hit return) ksu continues in a normal mode of operation (the target cache will not contain the desired TGT). If the wrong password is typed in, ksu fails.

Side Note: during authentication, only the tickets that could be obtained without providing a password are cached in in the source cache.

OPTIONS

–n *target_principal_name*

Specify a Kerberos target principal name. Used in authentication and authorization phases of ksu.

If ksu is invoked without **–n,** a default principal name is assigned via the following heuristic:

Case 1: source user is non-root.
If the target user is the source user the default principal name is set to the default principal of the source cache. If the cache does not exist then the default principal name is set to target_user@local_realm. If the source and target users are different and neither ~/target_user/.k5users nor ~/target_user/.k5login exist then the default principal name is target_user_login_name@local_realm. Otherwise, starting with the first principal listed below, ksu checks if the principal is authorized to access the target account and whether there is a legitimate ticket for that principal in the source cache. If both conditions are met that principal becomes the default target principal, otherwise go to the next principal.

a) default principal of the source cache
b) target_user@local_realm
c) source_user@local_realm

If a-c fails try any principal for which there is a ticket in the source cache and that is authorized to access the target account. If that fails select the first principal that is authorized to access the target account from the above list. If none are authorized and ksu is configured with PRINC_LOOK_AHEAD turned on, select the default principal as follows:

For each candidate in the above list, select an authorized principal that has the same realm

Reference Manual for ksu

name and first part of the principal name equal to the prefix of the candidate. For example if candidate a) is jqpublic@ISI.EDU and jqpublic/secure@ISI.EDU is authorized to access the target account then the default principal is set to jqpublic/secure@ISI.EDU.

Case 2: source user is root.
If the target user is non-root then the default principal name is target_user@local_realm. Else, if the source cache exists the default principal name is set to the default principal of the source cache. If the source cache does not exist, default principal name is set to root@local_realm.

–c *source_cache_name*
Specify source cache name (e.g. **–c** FILE:/tmp/my_cache). If **–c** option is not used then the name is obtained from KRB5CCNAME environment variable. If KRB5CCNAME is not defined the source cache name is set to krb5cc_<source uid>.

–C *target_cache_name*
Specify the target cache name (e.g. **–C** FILE:/tmp/target_cache). If '.' is specified (e.g. ksu –C .) ksu uses the source cache and does not create a new target cache. Note: this case requires both source and target user to have read and write permissions for the source cache. If **–C** option is not used, the default target cache name is set to krb5cc_<target uid>.(gen_sym()), where gen_sym generates a new number such that the resulting cache does not already exist.
For example: krb5cc_1984.2

–k Do not delete the target cache upon termination of the target shell or a command (**–e** command). Without **–k,** ksu deletes the target cache (unless ksu was invoked with '-C .' option).

–D turn on debug mode.

Ticket granting ticket options: -l lifetime -r time -pf
The ticket granting ticket options only apply to the case where there are no appropriate tickets in the cache to authenticate the source user. In this case if ksu is configured to prompt users for a Kerberos password (GET_TGT_VIA_PASSWD is defined), the ticket granting ticket options that are specified will be used when getting a ticket granting ticket from the Kerberos server.

–l *lifetime* option specifies the lifetime to be requested for the ticket; if this option is not specified, the default ticket lifetime (configured by each site) is used instead.

–r *time* option specifies that the RENEWABLE option should be requested for the ticket, and specifies the desired total lifetime of the ticket.

–p option specifies that the PROXIABLE option should be requested for the ticket.

–f option specifies that the FORWARDABLE option should be requested for the ticket.

–z restrict the copy of tickets from the source cache to the target cache to only the tickets where client == the target principal name. Use the **–n** option if you want the tickets for other then the default principal. Note that the **–z** option is mutually exclusive with '-C .' and -Z options.

–Z Don't copy any tickets from the source cache to the target cache. Just create a fresh target cache, where the default principal name of the cache is initialized to the target principal name. Note that **–Z** option is mutually exclusive with '-C .' and -z options.

–q suppress the printing of status messages.

–e *command [args ...]*
ksu proceeds exactly the same as if it was invoked without the **–e** option, except instead of executing the target shell, ksu executes the specified command (Example of usage: ksu bob **–e** ls **–lag).**

The authorization algorithm for -e is as follows:

If the source user is root or source user == target user, no authorization takes place and the command is executed. If source user id != 0, and .k5users file does not exist, authorization

Reference Manual for `ksu`

fails. Otherwise, .k5users file must have an appropriate entry for target principal to get authorized.

The .k5users file format:

A single principal entry on each line that may be followed by a list of commands that the principal is authorized to execute. A principal name followed by a '*' means that the user is authorized to execute any command. Thus, in the following example:

jqpublic@USC.EDU ls mail /local/kerberos/klist
jqpublic/secure@USC.EDU *
jqpublic/admin@USC.EDU

jqpublic@USC.EDU is only authorized to execute ls, mail and klist commands. jqpublic/secure@USC.EDU is authorized to execute any command. jqpublic/admin@USC.EDU is not authorized to execute any command. Note, that jqpublic/admin@USC.EDU is authorized to execute the target shell (regular ksu, without the **–e** option) but jqpublic@USC.EDU is not.

The commands listed after the principal name must be either a full path names or just the program name. In the second case, CMD_PATH specifying the location of authorized programs must be defined at the compilation time of ksu.

Which command gets executed ?

If the source user is root or the target user is the source user or the user is authorized to execute any command ('*' entry) then command can be either a full or a relative path leading to the target program. Otherwise, the user must specify either a full path or just the program name.

–a *args* specify arguments to be passed to the target shell. Note: that all flags and parameters following -a will be passed to the shell, thus all options intended for ksu must precede **–a.** **–a** option can be used to simulate the **–e** option if used as follows: **–a –c** [command [arguments]]. **–c** is interpreted by the c-shell to execute the command.

INSTALLATION INSTRUCTIONS

ksu can be compiled with the following 5 flags (see the Imakefile):

GET_TGT_VIA_PASSWD
in case no appropriate tickets are found in the source cache, the user will be prompted for a Kerberos password. The password is then used to get a ticket granting ticket from the Kerberos server. The danger of configuring ksu with this macro is if the source user is loged in remotely and does not have a secure channel, the password may get exposed.

PRINC_LOOK_AHEAD
during the resolution of the default principal name, PRINC_LOOK_AHEAD enables ksu to find principal names in the .k5users file as described in the OPTIONS section (see -n option).

CMD_PATH
specifies a list of directories containing programs that users are authorized to execute (via .k5users file).

HAS_GETUSERSHELL
If the source user is non-root, ksu insists that the target user's shell to be invoked is a "legal shell". getusershell(3) is called to obtain the names of "legal shells". Note that the target user's shell is obtained from the passwd file.

SAMPLE CONFIGURATION:
KSU_OPTS = -DGET_TGT_VIA_PASSWD -DPRINC_LOOK_AHEAD -DCMD_PATH='"/bin /usr/ucb /local/bin"

Reference Manual for `ksu`

PERMISSIONS FOR KSU

ksu should be owned by root and have the set user id bit turned on.

END-SERVER ENTRY

ksu attempts to get a ticket for the end server just as Kerberized telnet and rlogin. Thus, there must be an entry for the server in the Kerberos database (e.g. host/nii.isi.edu@ISI.EDU). The keytab file must be in an appropriate location.

SIDE EFFECTS

ksu deletes all expired tickets from the source cache.

AUTHOR OF KSU: GENNADY (ARI) MEDVINSKY

Appendix A Kerberos Glossary

client an entity that can obtain a ticket. This entity is usually either a user or a host.

host a computer that can be accessed over a network.

Kerberos in Greek mythology, the three-headed dog that guards the entrance to the underworld. In the computing world, Kerberos is a network security package that was developed at MIT.

KDC Key Distribution Center. A machine that issues Kerberos tickets.

keytab a **key tab**le file containing one or more keys. A host or service uses a *keytab* file in much the same way as a user uses his/her password.

principal a string that names a specific entity to which a set of credentials may be assigned. It generally has three parts:

- **primary** the first part of a Kerberos *principal*. In the case of a user, it is the username. In the case of a service, it is the name of the service.
- **instance** the second part of a Kerberos *principal*. It gives information that qualifies the primary. The instance may be null. In the case of a user, the instance is often used to describe the intended use of the corresponding credentials. In the case of a host, the instance is the fully qualified hostname.
- **realm** the logical network served by a single Kerberos database and a set of Key Distribution Centers. By convention, realm names are generally all uppercase letters, to differentiate the realm from the internet domain.

The typical format of a typical Kerberos principal is primary/instance@REALM.

service any program or computer you access over a network. Examples of services include "host" (a host, *e.g.*, when you use `telnet` and `rsh`), "ftp" (FTP), "krbtgt" (authentication; cf. *ticket-granting ticket*), and "pop" (email).

ticket a temporary set of electronic credentials that verify the identity of a client for a particular service.

TGT Ticket-Granting Ticket. A special Kerberos ticket that permits the client to obtain additional Kerberos tickets within the same Kerberos realm.

Table of Contents