# Kerberos V5 UNIX User's Guide

Release: 1.9
Document Edition: 1.0
Last updated: 5 May 2011

**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 network security.

The Kerberos V5 package is designed to be easy to use. Most of the commands are nearly identical to UNIX network programs you already use. 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. Principals can have an arbitrary number of components. Each component is separated by a component separator, generally '/'. The last component is the realm, separated from the rest of the principal by the realm separator, generally '`@`'. If there is no realm component in the principal, then it will be assumed that the principal is in the default realm for the context in which it is being used.

Traditionally, 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.example.com` would be in the realm `EXAMPLE.COM`.

# 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 `example.com`. 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 Kerberos Ticket Properties

There are various properties that Kerberos tickets can have:

If a ticket is *forwardable*, then the KDC can issue a new ticket with a different network address based on the forwardable ticket. This allows for authentication forwarding without requiring a password to be typed in again. For example, if a user with a forwardable TGT logs into a remote system, the KDC could issue a new TGT for that user with the network address of the remote system, allowing authentication on that host to work as though the user were logged in locally.

When the KDC creates a new ticket based on a forwardable ticket, it sets the *forwarded* flag on that new ticket. Any tickets that are created based on a ticket with the forwarded flag set will also have their forwarded flags set.

A *proxiable* ticket is similar to a forwardable ticket in that it allows a service to take on the identity of the client. Unlike a forwardable ticket, however, a proxiable ticket is only issued for specific services. In other words, a ticket-granting ticket cannot be issued based on a ticket that is proxiable but not forwardable.

A *proxy* ticket is one that was issued based on a proxiable ticket.

A *postdated* ticket is issued with the *invalid* flag set. After the starting time listed on the ticket, it can be presented to the KDC to obtain valid tickets.

Tickets with the *postdateable* flag set can be used to issue postdated tickets.

*Renewable* tickets can be used to obtain new session keys without the user entering their password again. A renewable ticket has two expiration times. The first is the time at which this particular ticket expires. The second is the latest possible expiration time for any ticket issued based on this renewable ticket.

A ticket with the *initial* flag set was issued based on the authentication protocol, and not on a ticket-granting ticket. Clients that wish to ensure that the user's key has been recently presented for verification could specify that this flag must be set to accept the ticket.

An *invalid* ticket must be rejected by application servers. Postdated tickets are usually issued with this flag set, and must be validated by the KDC before they can be used.

A *preauthenticated* ticket is one that was only issued after the client requesting the ticket had authenticated itself to the KDC.

The *hardware authentication* flag is set on a ticket which required the use of hardware for authentication. The hardware is expected to be possessed only by the client which requested the tickets.

If a ticket has the *transit policy checked* flag set, then the KDC that issued this ticket implements the transited-realm check policy and checked the transited-realms list on the ticket. The transited-realms list contains a list of all intermediate realms between the realm of the KDC that issued the first ticket and that of the one that issued the current ticket. If this flag is not set, then the application server must check the transited realms itself or else reject the ticket.

The *okay as delegate* flag indicates that the server specified in the ticket is suitable as a delegate as determined by the policy of that realm. A server that is acting as a delegate has been granted a proxy or a forwarded TGT. This flag is a new addition to the Kerberos V5 protocol and is not yet implemented on MIT servers.

An *anonymous* ticket is one in which the named principal is a generic principal for that realm; it does not actually specify the individual that will be using the ticket. This ticket is meant only to securely distribute a session key. This is a new addition to the Kerberos V5 protocol and is not yet implemented on MIT servers.

### 2.2.2 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, EXAMPLE.COM.[1] He would type:

```
shell% kinit david@EXAMPLE.COM
Password for david@EXAMPLE.COM: <-- [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 ⟨undefined⟩ [Viewing Your Tickets with klist], page ⟨undefined⟩).

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.

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

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

```
shell% kinit -f -l 3h david@EXAMPLE.COM
Password for david@EXAMPLE.COM: <-- [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.

### 2.2.3 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 ⟨undefined⟩ [What is a Ticket?], page ⟨undefined⟩.) The listing would look like this:

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

Valid starting     Expires            Service principal
06/07/04 19:49:21  06/08/04 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 ⟨undefined⟩ [What is a Kerberos Principal?], page ⟨undefined⟩)

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
06/07/04 19:49:21  06/08/04 05:49:19  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
06/07/04 20:22:30  06/08/04 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.example.com`, which is also in another Kerberos realm, `EXAMPLE.COM`. If you telnet to this host, you will receive a ticket-granting ticket for the realm `EXAMPLE.COM`, plus the new `host` ticket for `trillium.example.com`. *`klist`* will now show:

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

Valid starting     Expires            Service principal
06/07/04 19:49:21  06/08/04 05:49:19  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
06/07/04 20:22:30  06/08/04 05:49:19  host/daffodil.mit.edu@ATHENA.MIT.EDU
06/07/04 20:24:18  06/08/04 05:49:19  krbtgt/EXAMPLE.COM@ATHENA.MIT.EDU
06/07/04 20:24:18  06/08/04 05:49:19  host/trillium.example.com@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 |
| **H** | **H**ardware authenticated |
| **A** | pre**A**uthenticated |
| **T** | **T**ransit policy checked |
| **O** | **O**kay as delegate |
| **a** | **a**nonymous |

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 ⟨undefined⟩ [kinit Reference], page ⟨undefined⟩, 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
31/07/05 19:06:25  31/07/05 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@EXAMPLE.COM

Valid starting     Expires            Service principal
07/31/05 11:52:29  07/31/05 21:11:23  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        Flags: Ff
07/31/05 12:03:48  07/31/05 21:11:23  host/trillium.example.com@EXAMPLE.COM
        Flags: Ff
shell%
```

### 2.2.4 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
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 ⟨undefined⟩ [Password Advice], page ⟨undefined⟩), and keep it secret. If you need to give access to your account to someone else, you can do so through Kerberos. (See ⟨undefined⟩ [Granting Access to Your Account], page ⟨undefined⟩.) 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
Password for david:     <- Type your old password.
Enter new password:     <- Type your new password.
Enter it again:  <- Type the new password again.
Password changed.
shell%
```

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

```
shell% kpasswd
Password for david:  <- Type the incorrect old password.
kpasswd: Password incorrect while getting initial ticket
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
Password for david:  <- Type the old password.
Enter new password:  <- Type the new password.
Enter it again: <- Type a different new password.
kpasswd: Password mismatch while reading password
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 certain criteria, such as a password dictionary or a minimum length. For example, if the user `jennifer`, who had a policy `"strict"` that required a minimum of 8 characaters, chose a password that was less than 8 characters, Kerberos would give an error message like the following:

```
shell% kpasswd
Password for jennifer:  <- Type your old password here.

jennifer's password is controlled by the policy strict, which
requires a minimum of 8 characters from at least 3 classes (the five classes
are lowercase, uppercase, numbers, punctuation, and all other characters).

Enter new password:  <- Type an insecure new password.
Enter it again:  <- Type it again.

kpasswd: Password is too short while attempting to change password.
Please choose another password.

Enter new password:  <- Type a good password here.
Enter it again:  <- 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 ⟨undefined⟩ [What is a Kerberos Principal?], page ⟨undefined⟩.) 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@EXAMPLE.COM
```

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 ⟨undefined⟩ [Obtaining Tickets with kinit], page ⟨undefined⟩), 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* | forwards a copy of your tickets to the remote host. |
| *-F* | forwards a copy of your tickets to the remote host, and marks them re-forwardable from the remote host. |
| *-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* | turns on 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.example.com
Trying 128.0.0.5 ...
Connected to daffodil.example.com.
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.mit.edu
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.example.com` 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.mit.edu`, 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.mit.edu
Trying 128.0.0.5...
Connected to trillium.mit.edu.
Escape character is '^]'.
[ Kerberos V5 accepts you as "jennifer@mit.edu" ]
[ Kerberos V5 accepted forwarded credentials ]
What you type is protected by encryption.
Last login: Tue Jul 30 18:47:44 from daffodil.example.com
Athena Server (sun4) Version 9.1.11 Tue Jul 30 14:40:08 EDT 2002

shell%
```

Note that `jennifer`'s machine used Kerberos to authenticate her to `trillium.mit.edu`, 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 ⟨undefined⟩ [telnet Reference], page ⟨undefined⟩)

### 2.4.3 rlogin

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

| | |
|---|---|
| *-f* | forwards a copy of your tickets to the remote host. |
| *-F* | forwards a copy of your tickets to the remote host, and marks them re-forwardable from the remote host. |
| *-k realm* | requests tickets for the remote host in the specified realm, instead of determining the realm itself. |
| *-x* | encrypts the input and output data streams (the username is sent unencrypted) |

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

```
shell% rlogin daffodil.example.com -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.example.com` 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.mit.edu`, 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.mit.edu -f -x
This rlogin session is using DES encryption for all data transmissions.
Last login: Thu Jun 20 16:20:50 from daffodil
Athena Server (sun4) Version 9.1.11 Tue Jul 30 14:40:08 EDT 2002
shell%
```

Note that `jennifer`'s machine used Kerberos to authenticate her to `trillium.mit.edu`, 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 ⟨undefined⟩ [rlogin Reference], page ⟨undefined⟩)

### 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.

`-f` requests that your tickets be forwarded to the remote host. The `-f` 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
200 Data channel protection level set to private.
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 ⟨undefined⟩ [FTP Reference], page ⟨undefined⟩)

### 2.4.5 rsh

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

| | |
|---|---|
| *-f* | forwards a copy of your tickets to the remote host. |
| *-F* | forwards a copy of your tickets to the remote host, and marks them re-forwardable from the remote host. |
| *-k realm* | requests tickets for the remote host in the specified realm, instead of determining the realm itself. |
| *-x* | encrypts the input and output data streams (the command line is not encrypted) |

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

```
shell% rsh trillium.example.com -l root -x date
This rsh session is using DES encryption for all data transmissions.
Tue Jul 30 19:34:21 EDT 2002
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 ⟨undefined⟩ [rsh Reference], page ⟨undefined⟩)

### 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` 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 ⟨undefined⟩ [rcp Reference], page ⟨undefined⟩)

### 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 ⟨undefined⟩ [Granting Access to Your Account], page ⟨undefined⟩) 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
07/31/04 21:53:01  08/01/04 07:52:53  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
07/31/04 21:53:39  08/01/04 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@EXAMPLE.COM
```

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 EXAMPLE.COM 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 EXAMPLE.COM.

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 ⟨undefined⟩ [What is a Ticket?], page ⟨undefined⟩.)

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

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

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

`-l lifetime`
: sets the ticket lifetime. (See ⟨undefined⟩ [Obtaining Tickets with kinit], page ⟨undefined⟩.) 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 ⟨undefined⟩ [ksu Reference], page ⟨undefined⟩)

# 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** [**−V**] [**−l** *lifetime*] [**−s** *start_time*] [**−r** *renewable_life*] [**−p** | **−P**] [**−f** | **−F**] [**−a**] [**−A**] [**−C**] [**−E**] [**−v**] [**−R**] [**−k** [**−t** *keytab_file*]] [**−c** *cache_name*] [**−n**] [**−S** *service_name*][**−T** *armor_ccache*] [**−X** *attribute*[=*value*]] [*principal*]

**DESCRIPTION**

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

**OPTIONS**

**−V** display verbose output.

**−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.

**−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.

**−f** request forwardable tickets.

**−F** do not request forwardable tickets.

**−p** request proxiable tickets.

**−P** do not request proxiable tickets.

**−a** request tickets with the local address[es].

**−A** request address-less tickets.

**−C** requests canonicalization of the principal name.

**−E** treats the principal name as an enterprise name.

**−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.

**−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 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. By default a host ticket is requested but any principal may be specified. On a KDC, the special keytab location **KDB:** can be used to indicate that kinit should open the KDC

## Reference Manual for `kinit`



database and look up the key directly. This permits an administrator to obtain tickets as any principal that supports password-based authentication.

**-n** Requests anonymous processing. Two types of anonymous principals are supported. For fully anonymous Kerberos, configure pkinit on the KDC and configure *pkinit_anchors* in the client's krb5.conf. Then use the **-n** option with a principal of the form *@REALM* (an empty principal name followed by the at-sign and a realm name). If permitted by the KDC, an anonymous ticket will be returned. A second form of anonymous tickets is supported; these realm-exposed tickets hide the identity of the client but not the client's realm. For this mode, use **kinit -n** with a normal principal name. If supported by the KDC, the principal (but not realm) will be replaced by the anonymous principal. As of release 1.8, the MIT Kerberos KDC only supports fully anonymous operation.

**−T** *armor_ccache*
Specifies the name of a credential cache that already contains a ticket. If supported by the KDC, This ccache will be used to armor the request so that an attacker would have to know both the key of the armor ticket and the key of the principal used for authentication in order to attack the request. Armoring also makes sure that the response from the KDC is not modified in transit.

**−c** *cache_name*
use *cache_name* as the Kerberos 5 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.

**−X** *attribute*[=*value*]
specify a pre−authentication attribute and value to be passed to pre−authentication plugins. The acceptable *attribute* and *value* values vary from pre−authentication plugin to plugin. This option may be specified multiple times to specify multiple attributes. If no *value* is specified, it is assumed to be "yes".

The following attributes are recognized by the OpenSSL pkinit pre-authentication mechanism:

**X509_user_identity**=*value*
specify where to find user's X509 identity information

**X509_anchors**=*value*
specify where to find trusted X509 anchor information

**flag_RSA_PROTOCOL**[=yes]
specify use of RSA, rather than the default Diffie-Hellman protocol

**ENVIRONMENT**

**Kinit** uses the following environment variables:

KRB5CCNAME Location of the Kerberos 5 credentials (ticket) cache.

**FILES**

/tmp/krb5cc_[uid] default location of Kerberos 5 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), kerberos(1)

## 3.2 klist Reference

Reference Manual for `klist`

KLIST(1) KLIST(1)

**NAME**

klist − list cached Kerberos tickets

**SYNOPSIS**

**klist** [**−e**] [[**−c**] [**−f**] [**−s**] [**−a** [**−n**]]] [**−k** [**−t**] [**−K**]] [*cache_name* | *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 |
| H | **H**ardware authenticated |
| A | pre**A**uthenticated |
| T | **T**ransit policy checked |
| O | **O**kay as delegate |
| a | **a**nonymous |

**−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 or if the tickets are
expired.

**−a** display list of addresses in credentials.

**−n** show numeric addresses instead of reverse-resolving addresses.

**−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 variables:

KRB5CCNAME Location of the Kerberos 5 credentials (ticket) cache.

**FILES**

/tmp/krb5cc_[uid] default location of Kerberos 5 credentials cache ([uid] is the decimal UID of the user).

/etc/krb5.keytab default location for the local host's **keytab** file.

**SEE ALSO**

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

Reference Manual for `klist`

## 3.3 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* ] [ **−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, and 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, the 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, except 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 the **−k option).** 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 name and first part of the principal name equal to the prefix of the candidate. For example if

Reference Manual for `ksu`



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>. The target cache name is automatically 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.

**−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 the -Z option.

**−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 the -z option.

**−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 ˜target_user/.k5users file does not exist, authorization fails. Otherwise, ˜target_user/.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

## Reference Manual for `ksu`

KSU(1) KSU(1)

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.** The **−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 4 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"

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

Reference Manual for `ksu`

KSU(1) KSU(1)

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**

## 3.4 kdestroy Reference

Reference Manual for `kdestroy`

KDESTROY(1) KDESTROY(1)

**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 variables:

KRB5CCNAME Location of the Kerberos 5 credentials (ticket) cache.

**FILES**

/tmp/krb5cc_[uid] default location of Kerberos 5 credentials cache ([uid] is the decimal UID of the user).

**SEE ALSO**

kinit(1), klist(1), krb5(3)

**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.

## 3.5 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, *kpasswd* uses the principal name from an existing ccache if there is one; if not, the principal is derived from the identity of the user invoking the *kpasswd* command.

**PORTS**

**kpasswd** looks first for kpasswd_server = host:port in the [realms] section of the krb5.conf file under the current realm. If that is missing, **kpasswd** looks for the admin_server entry, but substitutes 464 for the port.

**SEE ALSO**

kadmin(8), kadmind(8)

**BUGS**

**kpasswd** may not work with multi-homed hosts running on the Solaris platform.

## 3.6 telnet Reference

Reference Manual for `telnet`

Reference Manual for `telnet`

Reference Manual for `telnet`

Reference Manual for `telnet`

Reference Manual for `telnet`

Reference Manual for `telnet`

Reference Manual for `telnet`

Reference Manual for `telnet`

Reference Manual for `telnet`

## 3.7 FTP Reference

Reference Manual for `FTP`

Reference Manual for `FTP`

Reference Manual for `FTP`

Reference Manual for `FTP`

Reference Manual for `FTP`

Reference Manual for `FTP`

Reference Manual for `FTP`

Reference Manual for `FTP`

Reference Manual for `FTP`

## 3.8 rlogin Reference

Reference Manual for `rlogin`

Reference Manual for `rlogin`

## 3.9 rsh Reference

Reference Manual for `rsh`

Reference Manual for `rsh`

## 3.10 rcp Reference

Reference Manual for `rcp`

Reference Manual for `rcp`

# 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 can have an arbitrary number of components, but generally has three:

- **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.

# Appendix B Copyright

---

Portions contributed by Matt Crawford <crawdad@fnal.gov> were work performed at Fermi National Accelerator Laboratory, which is operated by Universities Research Association, Inc., under contract DE-AC02-76CHO3000 with the U.S. Department of Energy.

---

Portions of src/lib/crypto have the following copyright:



---

The implementation of the Yarrow pseudo-random number generator in src/lib/crypto/krb/prng/yarrow has the following copyright:



---

The implementation of the AES encryption algorithm in src/lib/crypto/builtin/aes has the following copyright:

The free distribution and use of this software in both source and binary form is allowed (with or without changes) provided that:

1. distributions of this source code include the above copyright notice, this list of conditions and the following disclaimer;
2. distributions in binary form include the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other associated materials;
3. the copyright holder's name is not used to endorse products built using this software without specific written permission.

DISCLAIMER

This software is provided 'as is' with no explcit or implied warranties in respect of any properties, including, but not limited to, correctness and fitness for purpose.

---

Portions contributed by Red Hat, including the pre-authentication plug-in framework and the NSS crypto implementation, contain the following copyright:



Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Red Hat, Inc., nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

The implementations of GSSAPI mechglue in GSSAPI-SPNEGO in `src/lib/gssapi`, including the following files:

```
lib/gssapi/generic/gssapi_err_generic.et
lib/gssapi/mechglue/g_accept_sec_context.c
lib/gssapi/mechglue/g_acquire_cred.c
lib/gssapi/mechglue/g_canon_name.c
lib/gssapi/mechglue/g_compare_name.c
lib/gssapi/mechglue/g_context_time.c
```

```
lib/gssapi/mechglue/g_delete_sec_context.c
lib/gssapi/mechglue/g_dsp_name.c
lib/gssapi/mechglue/g_dsp_status.c
lib/gssapi/mechglue/g_dup_name.c
lib/gssapi/mechglue/g_exp_sec_context.c
lib/gssapi/mechglue/g_export_name.c
lib/gssapi/mechglue/g_glue.c
lib/gssapi/mechglue/g_imp_name.c
lib/gssapi/mechglue/g_imp_sec_context.c
lib/gssapi/mechglue/g_init_sec_context.c
lib/gssapi/mechglue/g_initialize.c
lib/gssapi/mechglue/g_inquire_context.c
lib/gssapi/mechglue/g_inquire_cred.c
lib/gssapi/mechglue/g_inquire_names.c
lib/gssapi/mechglue/g_process_context.c
lib/gssapi/mechglue/g_rel_buffer.c
lib/gssapi/mechglue/g_rel_cred.c
lib/gssapi/mechglue/g_rel_name.c
lib/gssapi/mechglue/g_rel_oid_set.c
lib/gssapi/mechglue/g_seal.c
lib/gssapi/mechglue/g_sign.c
lib/gssapi/mechglue/g_store_cred.c
lib/gssapi/mechglue/g_unseal.c
lib/gssapi/mechglue/g_userok.c
lib/gssapi/mechglue/g_utils.c
lib/gssapi/mechglue/g_verify.c
lib/gssapi/mechglue/gssd_pname_to_uid.c
lib/gssapi/mechglue/mglueP.h
lib/gssapi/mechglue/oid_ops.c
lib/gssapi/spnego/gssapiP_spnego.h
lib/gssapi/spnego/spnego_mech.c
```

and the initial implementation of incremental propagation, including the following new or changed files:

```
include/iprop_hdr.h
kadmin/server/ipropd_svc.c
lib/kdb/iprop.x
lib/kdb/kdb_convert.c
lib/kdb/kdb_log.c
lib/kdb/kdb_log.h
lib/krb5/error_tables/kdb5_err.et
slave/kpropd_rpc.c
slave/kproplog.c
```

are subject to the following license:

BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---

Kerberos V5 includes documentation and software developed at the University of California at Berkeley, which includes this copyright notice:

Copyright © 1983 Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Portions contributed by Novell, Inc., including the LDAP database backend, are subject to the following license:

Copyright © 2004-2005, Novell, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The copyright holder's name is not used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL

---



---



---

permission notice appear in supporting documentation, and that the name of Apple Inc. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Apple Inc. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

---

The implementations of UTF-8 string handling in src/util/support and src/lib/krb5/unicode are subject to the following copyright and permission notice:

The OpenLDAP Public License
Version 2.8, 17 August 2003

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions in source form must retain copyright statements and notices,
2. Redistributions in binary form must reproduce applicable copyright statements and notices, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution, and
3. Redistributions must contain a verbatim copy of this document.

The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use this Software under terms of this license revision or under the terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in advertising or otherwise to promote the sale, use or other dealing in this Software without specific, written prior permission. Title to copyright in this Software shall at all times remain with copyright holders.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved. Permission to copy and distribute verbatim copies of this document is granted.

---

Marked test programs in src/lib/krb5/krb have the following copyright:

Copyright © 2006 Kungliga Tekniska Högskolan
(Royal Institute of Technology, Stockholm, Sweden).
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

---

Portions of the RPC implementation in src/lib/rpc and src/include/gssrpc have the following copyright and permission notice:



---

---



---



---

---

---

The following notice applies to `src/lib/krb5/krb/strptime.c`:



---

The following notice applies to Unicode library files in `src/lib/krb5/unicode`:

---

The following notice applies to `src/util/support/strlcpy.c`:



---

The following notice applies to `src/util/profile/argv_parse.c` and `src/util/profile/argv_parse.h`:



---

The following notice applies to SWIG-generated code in `src/util/profile/profile_tcl.c`:

---

The following notice applies to portiions of `src/lib/rpc` and `src/include/gssrpc`:



---

Implementations of the MD4 algorithm are subject to the following notice:



---

Implementations of the MD5 algorithm are subject to the following notice:

---

The following notice applies to `src/lib/crypto/crypto_tests/t_mddriver.c`:



---

Portions of `src/lib/krb5` are subject to the following notice:



---

# Table of Contents