NAME

krb5_initlog, **krb5_openlog**, **krb5_closelog**, **krb5_addlog_dest**, **krb5_addlog_func**, **krb5_log**, **krb5_vlog**, **krb5_log_msg**, **krb5_vlog_msg** - Heimdal logging functions

LIBRARY

Kerberos 5 Library (libkrb5, -lkrb5)

SYNOPSIS

#include <krb5.h>

typedef void
(*krb5_log_log_func_t)(*const char *time, const char *message, void *data*);

typedef void
(*krb5_log_close_func_t)(*void *data*);

krb5_error_code
krb5_addlog_dest(*krb5_context context, krb5_log_facility *facility, const char *destination*);

krb5_error_code
krb5_addlog_func(*krb5_context context, krb5_log_facility *facility, int min, int max,*
*krb5_log_log_func_t log, krb5_log_close_func_t close, void *data*);

krb5_error_code
krb5_closelog(*krb5_context context, krb5_log_facility *facility*);

krb5_error_code
krb5_initlog(*krb5_context context, const char *program, krb5_log_facility **facility*);

krb5_error_code
krb5_log(*krb5_context context, krb5_log_facility *facility, int level, const char *format, ...*);

krb5_error_code
krb5_log_msg(*krb5_context context, krb5_log_facility *facility, char **reply, int level,*
*const char *format, ...*);

krb5_error_code
krb5_openlog(*krb5_context context, const char *program, krb5_log_facility **facility*);

krb5_error_code

krb5_vlog(*krb5_context context, krb5_log_facility *facility, int level, const char *format, va_list arglist*);

krb5_error_code
krb5_vlog_msg(*krb5_context context, krb5_log_facility *facility, char **reply, int level, const char *format, va_list arglist*);

DESCRIPTION

These functions logs messages to one or more destinations.

The **krb5_openlog**() function creates a logging *facility*, that is used to log messages. A facility consists of one or more destinations (which can be files or syslog or some other device). The *program* parameter should be the generic name of the program that is doing the logging. This name is used to lookup which destinations to use. This information is contained in the logging section of the *krb5.conf* configuration file. If no entry is found for *program*, the entry for default is used, or if that is missing too, SYSLOG will be used as destination.

To close a logging facility, use the **krb5_closelog**() function.

To log a message to a facility use one of the functions **krb5_log**(), **krb5_log_msg**(), **krb5_vlog**(), or **krb5_vlog_msg**(). The functions ending in _msg return in *reply* a pointer to the message that just got logged. This string is allocated, and should be freed with **free**(). The *format* is a standard **printf**() style format string (but see the BUGS section).

If you want better control of where things gets logged, you can instead of using **krb5_openlog**() call **krb5_initlog**(), which just initializes a facility, but doesn't define any actual logging destinations. You can then add destinations with the **krb5_addlog_dest**() and **krb5_addlog_func**() functions. The first of these takes a string specifying a logging destination, and adds this to the facility. If you want to do some non-standard logging you can use the **krb5_addlog_func**() function, which takes a function to use when logging. The *log* function is called for each message with *time* being a string specifying the current time, and *message* the message to log. *close* is called when the facility is closed. You can pass application specific data in the *data* parameter. The *min* and *max* parameter are the same as in a destination (defined below). To specify a max of infinity, pass -1.

krb5_openlog() calls **krb5_initlog**() and then calls **krb5_addlog_dest**() for each destination found.

Destinations

The defined destinations (as specified in *krb5.conf*) follows:

STDERR

This logs to the program's stderr.

FILE:*/file*

FILE=*/file*
: Log to the specified file. The form using a colon appends to the file, the form with an equal truncates the file. The truncating form keeps the file open, while the appending form closes it after each log message (which makes it possible to rotate logs). The truncating form is mainly for compatibility with the MIT libkrb5.

DEVICE=*/device*
: This logs to the specified device, at present this is the same as FILE:/device.

CONSOLE
: Log to the console, this is the same as DEVICE=/dev/console.

SYSLOG[:priority[:facility]]
: Send messages to the syslog system, using priority, and facility. To get the name for one of these, you take the name of the macro passed to syslog(3), and remove the leading LOG_ (LOG_NOTICE becomes NOTICE). The default values (as well as the values used for unrecognised values), are ERR, and AUTH, respectively. See syslog(3) for a list of priorities and facilities.

Each destination may optionally be prepended with a range of logging levels, specified as min-max/. If the *level* parameter to **krb5_log**() is within this range (inclusive) the message gets logged to this destination, otherwise not. Either of the min and max valued may be omitted, in this case min is assumed to be zero, and max is assumed to be infinity. If you don't include a dash, both min and max gets set to the specified value. If no range is specified, all messages gets logged.

EXAMPLES

```
[logging]
        kdc = 0/FILE:/var/log/kdc.log
        kdc = 1-/SYSLOG:INFO:USER
        default = STDERR
```

This will log all messages from the **kdc** program with level 0 to */var/log/kdc.log*, other messages will be logged to syslog with priority LOG_INFO, and facility LOG_USER. All other programs will log all messages to their stderr.

SEE ALSO

syslog(3), krb5.conf(5)

BUGS

These functions use **asprintf**() to format the message. If your operating system does not have a working **asprintf**(), a replacement will be used. At present this replacement does not handle some correct conversion specifications (like floating point numbers). Until this is fixed, the use of these conversions should be avoided.

If logging is done to the syslog facility, these functions might not be thread-safe, depending on the implementation of **openlog**(), and **syslog**().