NAME

nis_subr, nis_leaf_of, nis_name_of, nis_domain_of, nis_getnames, nis_freenames, nis_dir_cmp, nis_clone_object, nis_destroy_object, nis_print_object – NIS+ subroutines

SYNOPSIS

cc [*flag* ...] *file*... **-lnsl** [*library*...]

#include <rpcsvc/nis.h>

nis_name nis_leaf_of(const nis_name *name***);**

nis_name nis_name_of(const nis_name *name***);**

nis_name nis_domain_of(const nis_name *name***);**

nis_name *nis_getnames(const nis_name *name***);**

void nis_freenames(nis_name **namelist***);**

name_pos nis_dir_cmp(const nis_name *n1***, const nis_name** *n2***);**

nis_object *nis_clone_object(const nis_object **src***, nis_object** **dest***);**

void nis_destroy_object(nis_object **obj***);**

void nis_print_object(const nis_object **obj***);**

MT-LEVEL

Safe

DESCRIPTION

These subroutines are provided to assist in the development of NIS+ applications. They provide several useful operations on both NIS+ names and objects.

The first group, **nis_leaf_of( )**, **nis_domain_of( )**, and **nis_name_of( )** provide the functions for parsing NIS+ names. **nis_leaf_of( )** will return the first label in an NIS+ name. It takes into account the double quote character '"' which can be used to protect embedded '.' (dot) characters in object names. Note that the name returned will never have a trailing dot character. If passed the global root directory name ".", it will return the null string.

nis_domain_of( ) returns the name of the NIS+ domain in which an object resides. This name will always be a fully qualified NIS+ name and ends with a dot. By iteratively calling **nis_leaf_of( )** and **nis_domain_of( )** it is possible to break a NIS+ name into its individual components.

nis_name_of( ) is used to extract the unique part of a NIS+ name. This function removes from the tail portion of the name all labels that are in common with the local domain. Thus if a machine were in domain **foo.bar.baz.** and **nis_name_of( )** were passed a name **bob.friends.foo.bar.baz**, then **nis_name_of( )** would return the unique part, **bob.friends**. If the name passed to this function is not in either the local domain or one of its children, this function will return null.

nis_getnames( ) will return a list of candidate names for the name passed in as *name*. If this name is not fully qualified, **nis_getnames( )** will generate a list of names using the default NIS+ directory search path, or the environment variable **NIS_PATH** if it is set. The returned array of pointers is terminated by a NULL pointer, and the memory associated with this array should be freed by calling **nis_freenames( ).**

Though **nis_dir_cmp( )** can be used to compare any two NIS+ names, it is used primarily to compare domain names. This comparison is done in a case independent fashion, and the results are an enum of type **name_pos**. When the names passed to this function are identical, the function returns a value of SAME_NAME. If the name *n1* is a direct ancestor of name *n2*, then this function returns the result HIGHER_NAME. Similarly, if the name *n1* is a direct descendant of name *n2*, then this function returns the result LOWER_NAME. When the name *n1* is neither a direct ancestor nor a direct descendant of *n2*, as it would be if the two names were siblings in separate portions of the namespace, then this function returns the result NOT_SEQUENTIAL. Finally, if either name cannot be parsed as a legitimate name then this function returns the value BAD_NAME.

The second set of functions, consisting of **nis_clone_object( )** and **nis_destroy_object( )**, are used for manipulating objects. **nis_clone_object( )** creates an exact duplicate of the NIS+ object *src*. If the value of *dest* is non-null, it creates the clone of the object into this object structure and allocate the necessary memory for the variable length arrays. If this parameter is null, a pointer to the cloned object is returned. Refer to **nis_objects**(3N) for a description of the **nis_object** structure.

nis_destroy_object( ) can be used to destroy an object created by **nis_clone_object( )**. This will free up all memory associated with the object and free the pointer passed. If the object was cloned into an array (using the *dest* parameter to **nis_clone_object( )**) then the object *cannot* be freed with this function. Instead, the function **xdr_free(xdr_nis_object,** *dest***)** must be used.

nis_print_object( ) prints out the contents of a NIS+ object structure on the standard output. Its primary use is for debugging NIS+ programs.

ENVIRONMENT

NIS_PATH This variable overrides the default NIS+ directory search path used by **nis_getnames( ).** It contains an ordered list of directories separated by ':' (colon) characters. The '$' (dollar sign) character is treated specially. Directory names that end in '$' have the default domain appended to them, and a '$' by itself is replaced by the list of directories between the default domain and the global root that are at least two levels deep. The default NIS+ directory search path is '$'.

SEE ALSO

nis_names(3N), **nis_objects**(3N), **nis_tables**(3N)

NOTES

nis_leaf_of( ), **nis_name_of( )**and **nis_clone_object( )** return their results as thread-specific data in multithreaded applications.