NAME

netdir, netdir_getbyname, netdir_getbyaddr, netdir_free, netdir_options, taddr2uaddr, uaddr2taddr, netdir_perror, netdir_sperror, netdir_mergeaddr – generic transport name-to-address translation

SYNOPSIS

#include <netdir.h>

int netdir_getbyname(const struct netconfig **config,*
  const struct nd_hostserv **service,* **struct nd_addrlist** **addrs**);**

int netdir_getbyaddr(const struct netconfig **config***,**
  struct nd_hostservlist ***service***, const struct netbuf** **netaddr***);**

void netdir_free(void **ptr***, const int** *struct_type***);**

int netdir_options(const struct netconfig **config***, const int** *option***,**
  const int *fildes***, char** **point_to_args***);**

char *taddr2uaddr(const struct netconfig **config***, const struct netbuf** **addr***);**

struct netbuf *uaddr2taddr(const struct netconfig **config***, const char** **uaddr***);**

void netdir_perror(char **s***);**

char *netdir_sperror(void);

MT-LEVEL

MT-Safe

DESCRIPTION

These routines provide a generic interface for name-to-address mapping that will work with all transport protocols. This interface provides a generic way for programs to convert transport specific addresses into common structures and back again. The **netconfig** structure, described on the **netconfig**(4) manual page, identifies the transport.

The **netdir_getbyname( )** routine maps the machine name and service name in the **nd_hostserv** structure to a collection of addresses of the type understood by the transport identified in the **netconfig** structure. This routine returns all addresses that are valid for that transport in the **nd_addrlist** structure. The **nd_hostserv** structure contains the following members:

```
char  *h_host;    /* host name */
char  *h_serv;    /* service name */
```

The **nd_addrlist** structure contains the following members:

```
int              n_cnt;        /* number of addresses */
struct netbuf  *n_addrs;
```

netdir_getbyname( ) accepts some special-case host names. The host names are defined in **<netdir.h>**. The currently defined host names are:

HOST_SELF — Represents the address to which local programs will bind their endpoints. **HOST_SELF** differs from the host name provided by **gethostname**(3C), which represents the address to which *remote* programs will bind their endpoints.

HOST_ANY Represents any host accessible by this transport provider. **HOST_ANY** allows applications to specify a required service without specifying a particular host name.

HOST_SELF_CONNECT Represents the host address that can be used to connect to the local host.

HOST_BROADCAST Represents the address for all hosts accessible by this transport provider. Network requests to this address will be received by all machines.

All fields of the **nd_hostserv** structure must be initialized.

To find the address of a given host and service on all available transports, call the **netdir_getbyname( )** routine with each **struct netconfig** structure returned by **getnetconfig**(3N).

The **netdir_getbyaddr( )** routine maps addresses to service names. This routine returns *service*, a list of host and service pairs that would yield this address. If more than one tuple of host and service name is returned, then the first tuple contains the preferred host and service names:

```
struct nd_hostservlist {
        int              *h_cnt;          /* number of hostservs found */
        struct hostserv  *h_hostservs;
}
```

The **netdir_free( )** structure is used to free the structures allocated by the name to address translation routines. *ptr* points to the structure that has to be freed. The **struct_type** identifies the structure:

struct netbuf	**ND_ADDR**
struct nd_addrlist	**ND_ADDRLIST**
struct hostserv	**ND_HOSTSERV**
struct nd_hostservlist	**ND_HOSTSERVLIST**

The universal address returned by **taddr2uaddr( )** should be freed by **free**( ).

The **netdir_options( )** routine is used to do all transport-specific setups and option management. *fildes* is the associated file descriptor. *option*, *fildes*, and *pointer_to_args* are passed to the **netdir_options( )** routine for the transport specified in *config*. Currently four values are defined for *option*:

ND_SET_BROADCAST
ND_SET_RESERVEDPORT
ND_CHECK_RESERVEDPORT
ND_MERGEADDR

The **taddr2uaddr( )** and **uaddr2taddr( )** routines support translation between universal addresses and TLI type **netbufs**. The **taddr2uaddr( )** routine takes a **struct netbuf** data structure and returns a pointer to a string that contains the universal address. It returns **NULL** if the conversion is not possible. This is not a fatal condition as some transports may not suppose a universal address form.

uaddr2taddr( ) is the reverse of **taddr2uaddr( )**. It returns the **struct netbuf** data structure for the given universal address.

If a transport provider does not support an option, **netdir_options** returns **-1** and the error message can be printed through **netdir_perror( )** or **netdir_sperror( )**.

The specific actions of each option follow.

ND_SET_BROADCAST Sets the transport provider up to allow broadcast, if the transport supports broadcast. *fildes* is a file descriptor into the transport (i.e., the result of a **t_open** of **/dev/udp**). *pointer_to_args* is not used. If this completes, broadcast operations may be performed on file descriptor *fildes*.

ND_SET_RESERVEDPORT Allows the application to bind to a reserved port, if that concept exists for the transport provider. *fildes* is an unbound file descriptor into the transport. If *pointer_to_args* is **NULL,** *fildes* will be bound to a reserved port. If *pointer_to_args* is a pointer to a **netbuf** structure, an attempt will be made to bind to any reserved port on the specified address.

ND_CHECK_RESERVEDPORT
Used to verify that the address corresponds to a reserved port, if that concept exists for the transport provider. *fildes* is not used. *pointer_to_args* is a pointer to a **netbuf** structure that contains the address. This option returns **0** only if the address specified in *pointer_to_args* is reserved.

ND_MERGEADDR Used to take a ''local address'' (like the **0.0.0.0** address that TCP uses) and return a ''real address'' that client machines can connect to. *fildes* is not used. *pointer_to_args* is a pointer to a **struct nd_mergearg**, which has the following members:

```
char s_uaddr;   /* server's universal address */
char c_uaddr;   /* client's universal address */
char m_uaddr;   /* the result */
```

If **s_uaddr** is something like **0.0.0.0.1.12**, and, if the call is successful, **m_uaddr** will be set to something like **192.11.109.89.1.12**. For most transports, **m_uaddr** is exactly what **s_uaddr** is.

RETURN VALUES

The **netdir_perror( )** routine prints an error message on the standard output stating why one of the name-to-address mapping routines failed. The error message is preceded by the string given as an argument.

The **netdir_sperror( )** routine returns a string containing an error message stating why one of the name-to-address mapping routines failed.

netdir_sperror( ) returns a pointer to a buffer which contains the error message string. This buffer is overwritten on each call. In multithreaded applications, this buffer is implemented as thread-specific data.

SEE ALSO

gethostname(3C), **getnetconfig**(3N), **getnetpath**(3N), **netconfig**(4)