NAME

rpc_svc_create, **svc_control**, **svc_create**, **svc_destroy**, **svc_dg_create**, **svc_fd_create**, **svc_raw_create**, **svc_tli_create**, **svc_tp_create**, **svc_vc_create** - library routines for the creation of server handles

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <rpc/rpc.h>

bool_t
svc_control(*SVCXPRT *svc, const u_int req, void *info*);

int
svc_create(*void (*dispatch)(struct svc_req *, SVCXPRT *), const rpcprog_t prognum, const rpcvers_t versnum, const char *nettype*);

*SVCXPRT **
svc_dg_create(*const int fildes, const u_int sendsz, const u_int recvsz*);

void
svc_destroy(*SVCXPRT *xprt*);

*SVCXPRT **
svc_fd_create(*const int fildes, const u_int sendsz, const u_int recvsz*);

*SVCXPRT **
svc_raw_create(*void*);

*SVCXPRT **
svc_tli_create(*const int fildes, const struct netconfig *netconf, const struct t_bind *bindaddr, const u_int sendsz, const u_int recvsz*);

*SVCXPRT **
svc_tp_create(*void (*dispatch)(struct svc_req *, SVCXPRT *), const rpcprog_t prognum, const rpcvers_t versnum, const struct netconfig *netconf*);

*SVCXPRT **
svc_vc_create(*const int fildes, const u_int sendsz, const u_int recvsz*);

DESCRIPTION

These routines are part of the RPC library which allows C language programs to make procedure calls on servers across the network. These routines deal with the creation of service handles. Once the handle is created, the server can be invoked by calling **svc_run**().

Routines

See rpc(3) for the definition of the *SVCXPRT* data structure.

svc_control()

A function to change or retrieve various information about a service object. The *req* argument indicates the type of operation and *info* is a pointer to the information. The supported values of *req*, their argument types, and what they do are:

SVCGET_VERSQUIET

If a request is received for a program number served by this server but the version number is outside the range registered with the server, an RPC_PROGVERSMISMATCH error will normally be returned. The *info* argument should be a pointer to an integer. Upon successful completion of the SVCGET_VERSQUIET request, **info* contains an integer which describes the server's current behavior: 0 indicates normal server behavior (that is, an RPC_PROGVERSMISMATCH error will be returned); 1 indicates that the out of range request will be silently ignored.

SVCSET_VERSQUIET

If a request is received for a program number served by this server but the version number is outside the range registered with the server, an RPC_PROGVERSMISMATCH error will normally be returned. It is sometimes desirable to change this behavior. The *info* argument should be a pointer to an integer which is either 0 (indicating normal server behavior - an RPC_PROGVERSMISMATCH error will be returned), or 1 (indicating that the out of range request should be silently ignored).

svc_create()

The **svc_create**() function creates server handles for all the transports belonging to the class *nettype*. The *nettype* argument defines a class of transports which can be used for a particular application. The transports are tried in left to right order in NETPATH variable or in top to bottom order in the netconfig database. If *nettype* is NULL, it defaults to "netpath".

The **svc_create**() function registers itself with the rpcbind service (see rpcbind(8)). The *dispatch* function is called when there is a remote procedure call for the given *prognum* and

versnum; this requires calling **svc_run**() (see **svc_run**() in rpc_svc_reg(3)). If **svc_create**() succeeds, it returns the number of server handles it created, otherwise it returns 0 and an error message is logged.

svc_destroy()

A function macro that destroys the RPC service handle *xprt*. Destruction usually involves deallocation of private data structures, including *xprt* itself. Use of *xprt* is undefined after calling this routine.

svc_dg_create()

This routine creates a connectionless RPC service handle, and returns a pointer to it. This routine returns NULL if it fails, and an error message is logged. The *sendsz* and *recvsz* arguments are arguments used to specify the size of the buffers. If they are 0, suitable defaults are chosen. The file descriptor *fildes* should be open and bound. The server is not registered with rpcbind(8).

Warning: since connectionless-based RPC messages can only hold limited amount of encoded data, this transport cannot be used for procedures that take large arguments or return huge results.

svc_fd_create()

This routine creates a service on top of an open and bound file descriptor, and returns the handle to it. Typically, this descriptor is a connected file descriptor for a connection-oriented transport. The *sendsz* and *recvsz* arguments indicate sizes for the send and receive buffers. If they are 0, reasonable defaults are chosen. This routine returns NULL if it fails, and an error message is logged.

svc_raw_create()

This routine creates an RPC service handle and returns a pointer to it. The transport is really a buffer within the process's address space, so the corresponding RPC client should live in the same address space; (see **clnt_raw_create**() in rpc_clnt_create(3)). This routine allows simulation of RPC and acquisition of RPC overheads (such as round trip times), without any kernel and networking interference. This routine returns NULL if it fails, and an error message is logged.

Note: **svc_run**() should not be called when the raw interface is being used.

svc_tli_create()

This routine creates an RPC server handle, and returns a pointer to it. The *fildes* argument is the file descriptor on which the service is listening. If *fildes* is RPC_ANYFD, it opens a file

descriptor on the transport specified by *netconf*. If the file descriptor is unbound and *bindaddr* is not NULL, *fildes* is bound to the address specified by *bindaddr*, otherwise *fildes* is bound to a default address chosen by the transport.

Note: the *t_bind* structure comes from the TLI/XTI SysV interface, which NetBSD does not use. The structure is defined in <*rpc/types.h*> for compatibility as:

```
struct t_bind {
   struct netbuf addr;          /* network address, see rpc(3) */
   unsigned int  qlen;          /* queue length (for listen(2)) */
};
```

In the case where the default address is chosen, the number of outstanding connect requests is set to 8 for connection-oriented transports. The user may specify the size of the send and receive buffers with the arguments *sendsz* and *recvsz*; values of 0 choose suitable defaults. This routine returns NULL if it fails, and an error message is logged. The server is not registered with the rpcbind(8) service.

svc_tp_create()

The **svc_tp_create**() function creates a server handle for the network specified by *netconf*, and registers itself with the rpcbind service. The *dispatch* function is called when there is a remote procedure call for the given *prognum* and *versnum*; this requires calling **svc_run**(). The **svc_tp_create**() function returns the service handle if it succeeds, otherwise a NULL is returned and an error message is logged.

svc_vc_create()

This routine creates a connection-oriented RPC service and returns a pointer to it. This routine returns NULL if it fails, and an error message is logged. The users may specify the size of the send and receive buffers with the arguments *sendsz* and *recvsz*; values of 0 choose suitable defaults. The file descriptor *fildes* should be open and bound. The server is not registered with the rpcbind(8) service.

SEE ALSO

rpc(3), rpc_svc_calls(3), rpc_svc_err(3), rpc_svc_reg(3), rpcbind(8)