NAME

rpc_svc_reg, **rpc_reg**, **svc_reg**, **svc_unreg**, **svc_auth_reg**, **xprt_register**, **xprt_unregister** — library routines for registering servers

SYNOPSIS

#include <rpc/rpc.h>

int
rpc_reg(*rpcprog_t prognum*, *rpcvers_t versnum*, *rpcproc_t procnum*,
    *char *(*procname)()*, *xdrproc_t inproc*, *xdrproc_t outproc*,
    *char *nettype*);

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

void
svc_unreg(*const rpcprog_t prognum*, *const rpcvers_t versnum*);

int
svc_auth_reg(*int cred_flavor*,
    *enum auth_stat (*handler)(struct svc_req *, struct rpc_msg *)*);

void
xprt_register(*SVCXPRT *xprt*);

void
xprt_unregister(*SVCXPRT *xprt*);

DESCRIPTION

These routines are a part of the RPC library which allows the RPC servers to register themselves with rpcbind (see rpcbind(8)), and associate the given program and version number with the dispatch function. When the RPC server receives a RPC request, the library invokes the dispatch routine with the appropriate arguments.

Routines

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

rpc_reg()
: Register program *prognum*, procedure *procname*, and version *versnum* with the RPC service package. If a request arrives for program *prognum*, version *versnum*, and procedure *procnum*, *procname* is called with a pointer to its argument(s); *procname* should return a pointer to its static result(s); *inproc* is the XDR function used to decode the arguments while *outproc* is the XDR function used to encode the results. Procedures are registered on all available transports of the class *nettype*. Seerpc(3). This routine returns 0 if the re gistration succeeded, −1 otherwise.

svc_reg()
: Associates *prognum* and *versnum* with the service dispatch procedure, *dispatch*. If *netconf* is NULL, the service is not registered with the rpcbind(8) service. If *netconf* is non-zero, then a mapping of the triple [*prognum*, *versnum*, *netconf->nc_netid*] to *xprt->xp_ltaddr* is established with the local rpcbind service.

  The **svc_reg**() routine returns 1 if it succeeds, and 0 otherwise.

svc_unreg()

Remove from the rpcbind service, all mappings of the triple [*prognum*, *versnum*, all-transports] to network address and all mappings within the RPC service package of the double [*prognum*, *versnum*] to dispatch routines.

svc_auth_reg()

Registers the service authentication routine *handler* with the dispatch mechanism so that it can be invoked to authenticate RPC requests received with authentication type *cred_flavor*. This interface allows developers to add new authentication types to their RPC applications without needing to modify the libraries. Service implementors usually do not need this routine.

Typical service application would call **svc_auth_reg**() after registering the service and prior to calling **svc_run**(). When needed to process an RPC credential of type*cred_flavor*, the *handler* procedure will be called with two arguments, *struct svc_req ∗rqst* and *struct rpc_msg ∗msg*, and is expected to return a valid *enum auth_stat* value. There is no provision to change or delete an authentication handler once registered.

The **svc_auth_reg**() routine returns 0 if the registration is successful, 1 if *cred_flavor* already has an authentication handler registered for it, and −1 otherwise.

xprt_register()

After RPC service transport handle *xprt* is created, it is registered with the RPC service package. This routine modifies the global variable *svc_fdset* (see rpc_svc_calls(3)). Service implementors usually do not need this routine.

xprt_unregister()

Before an RPC service transport handle *xprt* is destroyed, it unregisters itself with the RPC service package. This routine modifies the global variable *svc_fdset* (see rpc_svc_calls(3)). Service implementors usually do not need this routine.

AVAILABILITY

These functions are part of libtirpc.

SEE ALSO

select(2), rpc(3), rpcbind(3), rpc_svc_calls(3), rpc_svc_create(3), rpc_svc_err(3), rpcbind(8)