NAME

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

MT-LEVEL

MT-Safe

DESCRIPTION

These routines are a part of the RPC library which allows the RPC servers to register themselves with **rpcbind( )** (see **rpcbind**(1M)), 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**(3N) for the definition of the **SVCXPRT** data structure.

#include <rpc/rpc.h>

bool_t rpc_reg(u_long *prognum*, **u_long** *versnum*, **u_long** *procnum*,
char ∗ **const(**∗*procname*) **(char** ∗*arg*), **xdrproc_t** *inproc*, **xdrproc_t** *outproc*,
const char ∗*nettype*);

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 parameter(s); *procname* should return a pointer to its **static** result(s). The *arg* parameter to *procname* is a pointer to the (decoded) procedure argument. *inproc* is the XDR function used to decode the parameters while *outproc* is the XDR function used to encode the results. Procedures are registered on all available transports of the class *nettype*. See **rpc**(3N). This routine returns **0** if the registration succeeded, –**1** otherwise.

int svc_reg(const SVCXPRT ∗*xprt*, **const u_long** *prognum*, **const u_long** *versnum*,
const void (∗*dispatch*), **const struct netconfig** ∗*netconf*);

Associates *prognum* and *versnum* with the service dispatch procedure, *dispatch*. If *netconf* is **NULL**, the service is not registered with the *rpcbind* service. For example, if a service has already been registered using some other means, such as **inetd** (see **inetd**(1M)), it will not need to be registered again. If *netconf* is nonzero, 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.

void svc_unreg(const u_long *prognum*, **const u_long** *versnum*);

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.

int svc_auth_reg(const int *cred_flavor***, const enum auth_stat (***handler***));**

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 parameters **(struct svc_req** **rqst***, 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.

void xprt_register(const SVCXPRT **xprt***);**

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**(3N)). Service implementors usually do not need this routine.

void xprt_unregister(const SVCXPRT **xprt***);**

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**(3N)). Service implementors usually do not need this routine.

SEE ALSO

inetd(1M), **rpcbind**(1M), **rpc**(3N), **rpc_svc_calls**(3N), **rpc_svc_create**(3N), **rpc_svc_err**(3N), **rpcbind**(3N), **select**(3C)