NAME

svc_dg_enablecache, **svc_exit**, **svc_fdset**, **svc_freeargs**, **svc_getargs**, **svc_getreq_common**, **svc_getreq_poll**, **svc_getreqset**, **svc_getrpccaller**, **svc_pollset**, **svc_run**, **svc_sendreply** — library routines for RPC servers

SYNOPSIS

#include <rpc/rpc.h>

int
svc_dg_enablecache(*SVCXPRT ∗xprt*, *const unsigned cache_size*);

void
svc_exit(*void*);

bool_t
svc_freeargs(*const SVCXPRT ∗xprt*, *const xdrproc_t inproc*, *caddr_t in*);

bool_t
svc_getargs(*const SVCXPRT ∗xprt*, *const xdrproc_t inproc*, *caddr_t in*);

void
svc_getreq_common(*const int fd*);

void
svc_getreq_poll(*struct pollfd ∗pfdp*, *const int pollretval*);

void
svc_getreqset(*fd_set ∗ rdfds*);

struct netbuf ∗
svc_getrpccaller(*const SVCXPRT ∗xprt*);

struct cmsgcred ∗
__svc_getcallercreds(*const SVCXPRT ∗xprt*);

struct pollfd svc_pollset[FD_SETSIZE];

void
svc_run(*void*);

bool_t
svc_sendreply(*SVCXPRT ∗xprt*, *xdrproc_t outproc*, *char ∗out*);

DESCRIPTION

These routines are part of the RPC library which allows C language programs to make procedure calls on other machines across the network.

These routines are associated with the server side of the RPC mechanism. Some of them are called by the server side dispatch function, while others (such as **svc_run**()) are called when the server is initiated.

Routines

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

svc_dg_enablecache() This function allocates a duplicate request cache for the service endpoint *xprt*, large enough to hold *cache_size* entries. Once enabled, there is no way to disable caching. This routine returns 0 if space necessary for a cache of the given size was successfully allocated, and 1 otherwise.

svc_exit()
: This function, when called by any of the RPC server procedure or otherwise, causes **svc_run**() to return.

  As currently implemented, **svc_exit**() zeroes the *svc_fdset* global variable. If RPC server activity is to be resumed, services must be reregistered with the RPC library either through one of the rpc_svc_create(3) functions, or using **xprt_register**(). The **svc_exit**() function has global scope and ends all RPC server activity.

fd_set svc_fdset
: A global variable reflecting the RPC server's read file descriptor bit mask; it is suitable as an argument to the select(2) system call. This is only of interest if service implementors do not call **svc_run**(), but rather do their own asynchronous event processing. This variable is read-only (do not pass its address to select(2)!), yet it may change after calls to **svc_getreqset**() or any creation routines.

svc_freeargs()
: A function macro that frees any data allocated by the RPC/XDR system when it decoded the arguments to a service procedure using **svc_getargs**(). This routine returns TRUE if the results were successfully freed, and FALSE otherwise.

svc_getargs()
: A function macro that decodes the arguments of an RPC request associated with the RPC service transport handle *xprt*. The *in* argument is the address where the arguments will be placed; *inproc* is the XDR routine used to decode the arguments. This routine returns TRUE if decoding succeeds, and FALSE otherwise.

svc_getreq_common()
: This routine is called to handle a request on the given file descriptor.

svc_getreq_poll()
: This routine is only of interest if a service implementor does not call **svc_run**(), but instead implements custom asynchronous event processing. It is called when poll(2) has determined that an RPC request has arrived on some RPC file descriptors; *pollretval* is the return value from poll(2) and *pfdp* is the array of *pollfd* structures on which the poll(2) was done. It is assumed to be an array large enough to contain the maximal number of descriptors allowed.

svc_getreqset()
: This routine is only of interest if a service implementor does not call **svc_run**(), but instead implements custom asynchronous event processing. It is called when poll(2) has determined that an RPC request has arrived on some RPC file descriptors; *rdfds* is the resultant read file descriptor bit mask. The routine returns when all file descriptors associated with the value of *rdfds* have been serviced.

svc_getrpccaller()
: The approved way of getting the network address of the caller of a procedure associated with the RPC service transport handle *xprt*.

__svc_getcallercreds()
: *Warning*: this macro is specific to FreeBSD and thus not portable. This macro returns a pointer to a *cmsgcred* structure, defined in <sys/socket.h>, identifying the calling client. This only works if the client is calling the server over an AF_LOCAL socket.

struct pollfd svc_pollset[FD_SETSIZE];
: *svc_pollset* is an array of *pollfd* structures derived from *svc_fdset[]*. It is suitable as an argument to the poll(2) system call. The derivation of *svc_pollset* from *svc_fdset* is made in the current implementation in **svc_run**(). Service implementors who do not call **svc_run**() and who

wish to use this array must perform this derivation themselves.

svc_run()

This routine never returns. It waits for RPC requests to arrive, and calls the appropriate service procedure using **svc_getreq_poll**() when one arrives. This procedure is usually waiting for the poll(2) system call to return.

svc_sendreply()

Called by an RPC service's dispatch routine to send the results of a remote procedure call. The *xprt* argument is the request's associated transport handle; *outproc* is the XDR routine which is used to encode the results; and *out* is the address of the results. This routine returns TRUE if it succeeds, FALSE otherwise.

AVAILABILITY

These functions are part of libtirpc.

SEE ALSO

poll(2), select(2), rpc(3), rpc_svc_create(3), rpc_svc_err(3), rpc_svc_reg(3)