

**NAME**

**svc\_dg\_enablecache**, **svc\_exit**, **svc\_fdset**, **svc\_freeargs**, **svc\_getargs**, **svc\_getreq\_common**, **svc\_getreq\_poll**, **svc\_getreqset**, **svc\_getrpcaller**, **svc\_pollset**, **svc\_run**, **svc\_sendreply** - library routines for RPC servers

**LIBRARY**

Standard C Library (libc, -lc)

**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 \*rdfs*);

*struct netbuf \**

**svc\_getrpcaller**(*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 \*xp*rt, *xdrproc\_t outproc*, *void \*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.

**SEE ALSO**

poll(2), select(2), rpc(3), rpc\_svc\_create(3), rpc\_svc\_err(3), rpc\_svc\_reg(3)