NAME

rpcgen – an RPC protocol compiler

SYNOPSIS

rpcgen *infile*

rpcgen [**–a**] [**–A**] [**–b**] [**–C**] [**–D** *name* [= *value*]] [**–i** *size*] [**–I** [**–K** *seconds*]] [**–L**] [**–M**] [**–N**] [**–T**] [**–Y** *pathname*] *infile*

rpcgen [**–c** | **–h** | **–l** | **–m** | **–t** | **–Sc** | **–Ss** | **–Sm**] [**–o** *outfile*] [*infile*]

rpcgen [**–s** *nettype*] [**–o** *outfile*] [*infile*]

rpcgen [**–n** *netid*] [**–o** *outfile*] [*infile*]

AVAILABILITY

SUNWcsu

DESCRIPTION

rpcgen is a tool that generates C code to implement an RPC protocol. The input to **rpcgen** is a language similar to C known as RPC Language (Remote Procedure Call Language).

rpcgen is normally used as in the first synopsis where it takes an input file and generates three output files. If the *infile* is named **proto.x**, then **rpcgen** generates a header in **proto.h**, XDR routines in **proto_xdr.c**, server-side stubs in **proto_svc.c**, and client-side stubs in **proto_clnt.c**. With the –**T** option, it also generates the RPC dispatch table in **proto_tbl.i**.

rpcgen can also generate sample client and server files that can be customized to suit a particular application. The –**Sc**, –**Ss** and –**Sm** options generate sample client, server and makefile, respectively. The –**a** option generates all files, including sample files. If the infile is **proto.x**, then the client side sample file is written to **proto_client.c**, the server side sample file to **proto_server.c** and the sample makefile to **makefile.proto**.

The server created can be started both by the port monitors (for example, **inetd** or **listen**) or by itself. When it is started by a port monitor, it creates servers only for the transport for which the file descriptor **0** was passed. The name of the transport must be specified by setting up the environment variable **PM_TRANSPORT**. When the server generated by **rpcgen** is executed, it creates server handles for all the transports specified in **NETPATH** environment variable, or if it is unset, it creates server handles for all the visible transports from **/etc/netconfig** file. Note: the transports are chosen at run time and not at compile time. When the server is self-started, it backgrounds itself by default. A special define symbol **RPC_SVC_FG** can be used to run the server process in foreground.

The second synopsis provides special features which allow for the creation of more sophisticated RPC servers. These features include support for user provided **#defines** and RPC dispatch tables. The entries in the RPC dispatch table contain:

- pointers to the service routine corresponding to that procedure,
- a pointer to the input and output arguments
- the size of these routines

A server can use the dispatch table to check authorization and then to execute the service routine; a client library may use it to deal with the details of storage management and XDR data conversion.

The other three synopses shown above are used when one does not want to generate all the output files, but only a particular one. See the **EXAMPLES** section below for examples of **rpcgen** usage. When **rpcgen** is executed with the –**s** option, it creates servers for that particular class of transports. When executed with the –**n** option, it creates a server for the transport specified by *netid*. If *infile* is not specified, **rpcgen** accepts the standard input.

The C preprocessor, **cc** –**E** is run on the input file before it is actually interpreted by **rpcgen**. For each type of output file, **rpcgen** defines a special preprocessor symbol for use by the **rpcgen** programmer:

RPC_HDR	defined when compiling into headers
RPC_XDR	defined when compiling into XDR routines
RPC_SVC	defined when compiling into server-side stubs
RPC_CLNT	defined when compiling into client-side stubs
RPC_TBL	defined when compiling into RPC dispatch tables

Any line beginning with ''**%**'' is passed directly into the output file, uninterpreted by **rpcgen**. To specify the path name of the C preprocessor use –**Y** flag.

For every data type referred to in *infile*, **rpcgen** assumes that there exists a routine with the string **xdr_** prepended to the name of the data type. If this routine does not exist in the RPC/XDR library, it must be provided. Providing an undefined data type allows customization of XDR routines.

OPTIONS

–**a** Generate all files, including sample files.

–**A** Enable the Automatic MT mode in the server main program. In this mode, the RPC library automatically creates threads to service client requests. This option generates multithread-safe stubs by implicitly turning on the **-M** option. Server multithreading modes and parameters can be set using the **rpc_control**() call. **rpcgen** generated code does not change the default values for the Automatic MT mode.

–**b** Backward compatibility mode. Generate transport specific RPC code for older versions of the operating system.

–**c** Compile into XDR routines.

–**C** Generate header and stub files which can be used with ANSI C compilers. Headers generated with this flag can also be used with C++ programs.

–**D***name*[**=***value*] Define a symbol *name*. Equivalent to the **#define** directive in the source. If no *value* is given, *value* is defined as **1**. This option may be specified more than once.

–**h** Compile into **C** data-definitions (a header). –**T** option can be used in conjunction to produce a header which supports RPC dispatch tables.

–i *size* Size at which to start generating inline code. This option is useful for optimization. The default size is 5.

–I Compile support for **inetd**(1M) in the server side stubs. Such servers can be self-started or can be started by **inetd**. When the server is self-started, it backgrounds itself by default. A special define symbol **RPC_SVC_FG** can be used to run the server process in foreground, or the user may simply compile without the –**I** option.

If there are no pending client requests, the **inetd** servers exit after 120 seconds (default). The default can be changed with the –**K** option. All of the error messages for **inetd** servers are always logged with **syslog**(3).

Note: This option is supported for backward compatibility only. It should always be used in conjunction with the –**b** option which generates backward compatibility code. By default (i.e., when –**b** is not specified), **rpcgen** generates servers that can be invoked through portmonitors.

–K *seconds* By default, services created using **rpcgen** and invoked through port monitors wait **120** seconds after servicing a request before exiting. That interval can be changed using the –**K** flag. To create a server that exits immediately upon servicing a request, use –**K 0**. To create a server that never exits, the appropriate argument is –**K –1**.

When monitoring for a server, some portmonitors, like **listen**(1M), *always* spawn a new process in response to a service request. If it is known that a server will be used with such a monitor, the server should exit immediately on completion. For such servers, **rpcgen** should be used with –**K 0**.

–l Compile into client-side stubs.

–L When the servers are started in foreground, use **syslog**(3) to log the server errors instead of printing them on the standard error.

–m Compile into server-side stubs, but do not generate a "main" routine. This option is useful for doing callback-routines and for users who need to write their own "main" routine to do initialization.

–M Generate multithread-safe stubs for passing arguments and results between rpcgen generated code and user written code. This option is useful for users who want to use threads in their code.

–N This option allows procedures to have multiple arguments. It also uses the style of parameter passing that closely resembles C. So, when passing an argument to a remote procedure, you do not have to pass a pointer to the argument, but can pass the argument itself. This behavior is different from the old style of **rpcgen** generated code. To maintain backward compatibility, this option is not the default.

−n *netid* Compile into server-side stubs for the transport specified by *netid*. There should be an entry for *netid* in the netconfig database. This option may be specified more than once, so as to compile a server that serves multiple transports.

−o *outfile* Specify the name of the output file. If none is specified, standard output is used (−**c**, −**h**, −**l**, −**m**, −**n**, −**s**, −**Sc**, −**Sm**, −**Ss**, and −**t** modes only).

−s *nettype* Compile into server-side stubs for all the transports belonging to the class *nettype*. The supported classes are **netpath**, **visible**, **circuit_n**, **circuit_v**, **datagram_n**, **datagram_v**, **tcp**, and **udp** (see **rpc**(3N) for the meanings associated with these classes). This option may be specified more than once. Note: the transports are chosen at run time and not at compile time.

−Sc Generate sample client code that uses remote procedure calls.

−Sm Generate a sample Makefile which can be used for compiling the application.

−Ss Generate sample server code that uses remote procedure calls.

−t Compile into RPC dispatch table.

−T Generate the code to support RPC dispatch tables.

The options −**c**, −**h**, −**l**, −**m**, −**s**, −**Sc**, −**Sm**, −**Ss**, and −**t** are used exclusively to generate a particular type of file, while the options −**D** and −**T** are global and can be used with the other options.

−Y *pathname* Give the name of the directory where **rpcgen** will start looking for the C-preprocessor.

EXAMPLES

The following example:

example% rpcgen −T prot.x

generates all the five files: **prot.h**, **prot_clnt.c**, **prot_svc.c**, **prot_xdr.c** and **prot_tbl.i**.

The following example sends the C data-definitions (header) to the standard output.

example% rpcgen −h prot.x

To send the test version of the **-DTEST**, server side stubs for all the transport belonging to the class **datagram_n** to standard output, use:

example% rpcgen −s datagram_n −DTEST prot.x

To create the server side stubs for the transport indicated by *netid* **tcp**, use:

example% rpcgen −n tcp −o prot_svc.c

SEE ALSO

cc(1B), **inetd**(1M), **listen**(1M), **syslog**(3), **rpc**(3N), **rpc_svc_calls**(3N)

The **rpcgen** chapter in the *ONC+ Developers Guide* manual.