NAME

rpld – x86 Network Booting RPL (Remote Program Load) Server

SYNOPSIS

/usr/sbin/rpld [**−fdDMblgz**] *interface*
/usr/sbin/rpld **−a** [**−fdDMblgz**]

AVAILABILITY

x86

SUNWcsu

DESCRIPTION

The RPL server provides network booting functionality to x86 clients by listening to boot requests from them according to the RPL protocol specifications. Boot requests can be generated by clients using the boot floppy supplied in the x86 distribution. Once the request has been received, the server validates the client and adds it to its internal service list. Subsequent requests from the client to download bootfiles will result in the sending of data frames from the server to the client specifying where to load the boot program in memory. When all the bootfiles have been downloaded, the server specifies where to start execution to initiate the boot process.

In the first synopsis, the interface parameter names the network interface upon which **rpld** is to listen for requests. For example:

/usr/sbin/rpld /dev/le0

/usr/sbin/rpld /dev/smc0

In the second synopsis, **rpld** locates all of the network interfaces present on the system and starts a daemon process for each one.

The server starts by reading the default configuration file, or an alternate configuration file if one is specified. If no configuration file can be found, internal default values will be used. Alternatively, command line options are available to override any of the values in the configuration file. After the configuration options are set, it then opens the network interface as specified in the command line and starts listening to RPL boot requests.

Network boot x86 clients have to have information pre-configured on a server for the RPL server to validate and serve them. This involves putting configuration information in both the **ethers**(4) and the **bootparams**(4) databases. The **ethers** database contains a translation from the physical node address to the IP address of the clients and is normally used by the RARP server. The **bootparams** database stores all other information needed for booting off this client, such as the number of bootfiles and the file names of the various boot components. Both databases can be looked up by the RPL server through NIS. See the sub-section **Client Configuration** for information on how to set up these databases.

To assist in the administration and maintenance of the network boot activity, there are two run-time signals that the server will accept to change some run-time parameters and print out useful status information. See the sub-section **Signals** for details.

The RPL server is not limited to the ability to boot only x86 clients. If properly configured, the server should be able to download any bootfiles to the clients.

Client Configuration

The following configuration information is specific to booting x86 clients.

In order to allow clients to boot x86 from across the network, the client's information has to be pre-configured in two databases: **ethers**(4) and **bootparams**(4). Both databases can be accessed through NIS. Refer to *x86: Installing Solaris Software* for information on how to configure a diskless x86 client. The discussion contained in the rest of this section is provided for your information only and should not be performed manually.

The **ethers** database contains a translation table to convert the physical node address to the IP address of the client. Therefore, an IP address must be assigned to the client (if this has not been done already), the node address of the client must be obtained, and then this information needs to be entered in the **ethers** database.

The bulk of the configuration is done in the **bootparams** database. This is a free-format database that essentially contains a number of keyword-value string pairs. A number of keywords have been defined for specific purposes, like the **bootparams** RPC in **bootparamd**(1M). Three more keywords have been defined for the RPL server. They are **numbootfiles**, **bootfile**, and **bootaddr**. All three keywords must be in lowercase letters with no spaces before or after the equals symbol following the keyword.

numbootfiles
: Specifies the number of files to be downloaded to the network boot client. The format of this option is:

  numbootfiles=n

  Always use **numbootfiles=3** to boot x86 across the network.

bootfile
: Specifies the path name of the bootfile to be downloaded and where in memory to start loading the bootfile. A complete path name should be used. For example, assuming the client's IP address is 129.181.32.15:

  bootfile=/rplboot/129.181.32.15.hw.com:45000
  bootfile=/rplboot/129.181.32.15.glue.com:35000
  bootfile=/rplboot/129.181.32.15.inetboot=8000

  The path name following the equals symbol specifies the bootfile to be downloaded, and the hex address following the colon (:) is the absolute address of the memory location to start loading that bootfile. These addresses should be in the range of 7c00 to a0000 (i.e., the base 640K range excluding the interrupt vector and BIOS data areas). Address 45000 for this **hw.com** bootfile is also a suggested value and if possible should not be changed. The address of 35000 for **glue.com** is a suggested value that, if possible, should not be changed. The address of 8000 for **inetboot** is an absolute requirement and should never be changed.

These files, when created following the procedures in the *x86: Installing Solaris Software*, are actually symbolic links to to the real file to be downloaded to the client. **hw.com** is linked to a special driver that corresponds to the network interface card of the client. **glue.com** and **inetboot** are generic to all network boot clients.

The order of these bootfile lines is not significant, but because problems have been found with certain boot PROMs, it is highly recommended that the bootfile lines be ordered in descending order of the load addresses.

bootaddr
The absolute address in memory to start executing after all the bootfiles have been downloaded. This address should always correspond to the address where **glue.com** is being loaded. If possible, always use:

bootaddr=35000

OPTIONS

–**f** config filename.
Use this to specify a configuration file name other than the system default **/etc/rpld.conf** file.

–**d** debug level.
Specify a level of **0** if you do not want any error or warning messages to be generated, or a level from **1** to **9** for increasing amounts of messages. This option corresponds to the *DebugLevel* setting in the configuration file. The default value is **0**. Note that it is best to limit the level to **8** or below; use of level **9** may generate so many debug messages that the performance of the RPL server may be impacted.

–**D** debug destination.
Specify **0** to send error or warning messages to standard output, **1** to syslogd, and **2** to the log file. This option corresponds to the *DebugDest* setting in the configuration file. The default value is **2**.

–**M** maximum clients.
Specify the maximum number of simultaneous network boot clients to be served. This option corresponds to the *MaxClients* setting in the configuration file. A value of –**1** means unlimited, and the actual number will depend on available system resources. The default value is –**1**.

–**b** background mode.
Specify **1** to run the server in the background and relinquish the controlling terminal, or **0** to run in the foreground without relinquishing the controlling terminal. This option corresponds to the *BackGround* setting in the configuration file. If you have specified that the error or warning messages be sent to standard output in the configuration file or by using the –**D** option above, the server cannot be run in background mode. Doing so will cause the server to exit after announcing the error.

–**l** log filename.
Specify an alternate log file name to hold the error or warning messages in connection with the –**D 2** option or the configuration file *DebugDest = 2* setting. This option corresponds to the *LogFile* setting in the configuration file. The default is **/var/spool/rpld.log**.

–s start delay count.

This option corresponds to the *StartDelay* setting in the configuration file. Specify the number of delay units between outgoing data frames sent to clients to avoid retransmission requests from them. Using the LLC type **1** protocol, data transfer is a one-way, best-effort delivery mechanism. The server, without any type of delay mechanism, can overrun the client by sending data frames too quickly. Therefore, a variable delay is built into the server to limit the speed of sending data to the clients, thus avoiding the clients sending back retransmission requests. This value should be machine environment specific. If you have a fast server machine but slow client machines, you may want to set a large start delay count. If you have comparable server and client machines, the delay count may be set to **1**. The delay is only approximate and should not be taken as an accurate measure of time. There is no specific correlation between the delay unit and the actual time of delay. The default value is **20**.

–g delay granularity.

This corresponds to the *DelayGran* setting in the configuration file. If retransmission requests from clients do occur, the delay granularity factor will be used to adjust the delay count for this client upwards or downwards. If the retransmission request is caused by data overrun, the delay count will be incremented by delay granularity units to increase the delay between data frames. If the retransmission request is caused by sending data too slowly, this will be used to adjust the delay count downwards to shorten the delay. Eventually the server will settle at the delay count value that works best with the speed of the client and no retransmission request will be needed. The default value is **2**.

–z frame size.

This option corresponds to the *FrameSize* setting in the configuration file. This specifies the size of the data frames used to send data to the clients. This is limited by the underlying physical medium. For **ethernet/802.3**, the maximum physical frame size is 1500 octets. The default value is **1500**. Note that the protocol overhead of LLC1 and RPL is 32 octets, resulting in a maximum data length of 1468 octets.

Signals

The RPL server accepts two signals to change run-time parameters and display status information, respectively:

HANGUP

This will cause the RPL server to reread the default configuration file **/etc/rpld.conf** or an alternate configuration file if one is specified when the server is started. New values of certain parameters can be used immediately, such as *DebugLevel*, *DebugDest*, *LogFile*, *DelayGran*, and *FrameSize*. For *MaxClients*, if the server is already serving more than the new value, the server will not accept additional boot requests until the number has fallen below the *MaxClients* parameter. For *StartDelay*, this will only affect new boot requests. All the existing delay counts for the various clients in service will not be affected. Finally, the *BackGround* parameter will have no effect once the server has been running. You cannot change the mode of service without first killing the server and then restarting it.

USR1 This signal will cause the server to dump all the parameter values and the status of each individual boot client to the destination specified by *DebugDest*.

FILES

/usr/sbin/rpld
/etc/rpld.conf
/var/spool/rpld.log
/etc/ethers
/etc/bootparams
/rplboot

SEE ALSO

bootparamd(1M), **in.rarpd**(1M), **bootparams**(4), **ethers**(4), **nsswitch.conf**(4), **rpld.conf**(4)

x86: Installing Solaris Software