Rc.local File Overview

What is rc.local File and how it is worked?

The `rc.local` file is a script that allows you to execute commands or run scripts automatically during the system boot process, after all other initialization has been completed. It is often used for custom tasks like starting user-defined services, running scripts, or setting specific configurations at boot.

How `rc.local` Works

1. **Location**: The `rc.local` file is typically located at `/etc/rc.local` (or `/etc/rc.d/rc.local` on some systems).
2. **Purpose**:
   - It runs commands **after the system has booted** and all system services have started.
   - It's commonly used for **custom configurations**, **starting non-standard services**, or **executing custom scripts**.
3. **Format**:
   - The `rc.local` script is a standard shell script. Commands should follow shell script syntax.
   - Ensure that the script has a proper shebang (`#!/bin/sh` or `#!/bin/bash`) at the top.
4. **Permissions**:
   - The file must be executable. You can set executable permissions using:

     ```bash
     chmod +x /etc/rc.local
     ```

5. **Systemd and `rc.local`**:
   - On modern systems that use **systemd**, the `rc.local` script is not enabled by default.
   - A service file (`/lib/systemd/system/rc-local.service`) may need to be created or modified to enable and use `rc.local`.
6. **Exit Status**:
   - The script should exit with a status of `0` (success). If it exits with a non-zero status, the boot process may report an error.

Example of an rc.local File

Here is a basic example of an `rc.local` script:

```bash
#!/bin/bash
# rc.local

# Print a message to the console
echo "System booted successfully at $(date)" > /var/log/boot.log

# Start a custom script
/bin/bash /path/to/your/script.sh &

# Enable a specific IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

# Exit with 0 to indicate success
exit 0
```

Steps to Enable `rc.local` on Systemd Systems

1. **Create the `rc.local` File** (if it doesn't exist):

   ```bash
   sudo nano /etc/rc.local
   ```

   Add your commands, then save and exit.

2. **Set Executable Permissions**:

   ```bash
   sudo chmod +x /etc/rc.local
   ```

3. **Create a Systemd Service File (if not present)**:

   ```bash
   sudo nano /etc/systemd/system/rc-local.service
   ```

   Add the following content:

   ```ini
   [Unit]
   Description=/etc/rc.local Compatibility
   ConditionPathExists=/etc/rc.local
   After=network.target

   [Service]
   Type=forking
   ExecStart=/etc/rc.local start
   TimeoutSec=0
   RemainAfterExit=yes
   ```

```
GuessMainPID=no

[Install]
WantedBy=multi-user.target
```

4. **Enable and Start the Service**:

```bash
sudo systemctl enable rc-local
sudo systemctl start rc-local
```

5. **Check the Status**:

```bash
sudo systemctl status rc-local
```

Common Use Cases for `rc.local`

- Setting up **network routes** or **IP tables**.
- Starting custom daemons or background scripts.
- Configuring **hardware-specific settings**.
- Running **diagnostic commands** at boot.
- Mounting network drives or other filesystems.

Modern Alternatives

Although `rc.local` is still supported in many distributions, modern Linux systems encourage using **systemd service units** for similar tasks, as it offers better control, logging, and dependency management.

Ok then, If create rc.local file then i explicitly have the systemd service file then only work is it correct or not?

Yes, you are correct! On modern Linux systems that use **systemd** (most distributions today), if you create an `rc.local` file, it **requires an associated systemd service file** to ensure it is executed during the boot process.

Here's why:

- By default, modern distributions no longer rely on `rc.local` directly, as they use **systemd** for service management.

- To make the `rc.local` file work, you need to explicitly enable and configure the `rc-local.service`.

Steps to Make `rc.local` Work with Systemd

1. **Create the `rc.local` File**: Create the file if it doesn't already exist.

```
bash


  sudo nano /etc/rc.local
```

   Add the necessary commands to the file, ensuring it starts with a shebang (`#!/bin/bash`) and ends with `exit 0`:

```
bash


  #!/bin/bash
  echo "Hello, World!" > /var/log/rc.local.log
  exit 0
```

2. **Set Permissions**: Ensure the file is executable.

```
bash


  sudo chmod +x /etc/rc.local
```

3. **Create a Systemd Service File**: If it doesn't already exist, create the service file at `/etc/systemd/system/rc-local.service`:

```
bash


  sudo nano /etc/systemd/system/rc-local.service
```

   Add the following content:

```
ini


  [Unit]
  Description=/etc/rc.local Compatibility
  ConditionPathExists=/etc/rc.local
  After=network.target

  [Service]
  Type=forking
  ExecStart=/etc/rc.local start
  TimeoutSec=0
  RemainAfterExit=yes
  GuessMainPID=no
```

```
[Install]
WantedBy=multi-user.target
```

4. **Reload Systemd Daemon**: After creating or modifying the service file, reload the systemd daemon so it recognizes the changes:

```
bash

  sudo systemctl daemon-reload
```

5. **Enable the Service**: Enable the `rc-local.service` so it runs automatically at boot:

```
bash

  sudo systemctl enable rc-local
```

6. **Start the Service**: Start the service manually to test it:

```
bash

  sudo systemctl start rc-local
```

7. **Verify the Status**: Check whether the service is active and running:

```
bash

  sudo systemctl status rc-local
```

Troubleshooting

- If the service fails, check the logs for details:

```
bash

  sudo journalctl -xe
```

- Ensure the `/etc/rc.local` file has a valid shebang (`#!/bin/bash`) and ends with `exit 0`.

Key Points

- Without the `rc-local.service` file, the `rc.local` file **will not work** on a modern system using systemd.
- This setup ensures compatibility between your `rc.local` file and the systemd-based boot process.
- For complex tasks, consider creating dedicated systemd service files instead of relying on `rc.local`.

In open Wrt have also same systemd file need for rc.local run every boot?

No, **OpenWRT** does not use **systemd**, so you do not need a systemd service file to make `rc.local` run at every boot. OpenWRT uses a different initialization system called **procd** (or sometimes `init.d` scripts), which handles boot-time configuration differently.

How `rc.local` Works in OpenWRT

In OpenWRT, the `rc.local` file is already integrated into the boot process. You simply need to edit the `/etc/rc.local` file, and it will execute the commands in it at the end of the boot process.

Steps to Use `rc.local` in OpenWRT

1. **Locate and Edit the rc.local File**: Open the `rc.local` file using an editor like `vi` or `nano`:

   ```
   bash

     vi /etc/rc.local
   ```

   Or if you're using the LuCI web interface, go to **System > Startup > Local Startup** to edit it.

2. **Add Your Commands**: Add any commands or scripts you want to run at boot. Ensure the file ends with `exit 0`. For example:

   ```
   bash

     #!/bin/sh
     echo "Custom startup commands running" >> /tmp/rc.local.log
     /etc/init.d/my-custom-service start
     exit 0
   ```

3. **Save and Make Executable**: Ensure the file has the correct permissions:

   ```
   bash

     chmod +x /etc/rc.local
   ```

4. **Reboot to Test**: Reboot the device to verify that your commands in `rc.local` execute:

   ```
   bash

     reboot
   ```

5. **Verify Execution**: Check any logs or outputs generated by your `rc.local` commands to ensure they ran correctly. For example:

```bash
cat /tmp/rc.local.log
```

Key Differences Between OpenWRT and Systemd Systems

- **OpenWRT**: No need for a `systemd` service file. The `/etc/rc.local` file is executed automatically at the end of the boot process.
- **Systemd-Based Systems**: You need a `systemd` service file to enable and execute `/etc/rc.local`.

Best Practices for rc.local in OpenWRT

1. **Debugging**: Redirect the output of commands to a log file for troubleshooting:

```bash
echo "Command output" >> /tmp/rc.local.log
```

2. **Avoid Long-Running Tasks**: If a command takes too long to execute, it might delay the boot process. For long-running tasks, consider running them in the background using `&`.
3. **Check Syntax**: Ensure there are no syntax errors in your `rc.local` script, as they can cause the script to fail.

In summary, OpenWRT simplifies `rc.local` usage. You only need to edit the file, add your commands, and ensure it ends with `exit 0`. It will be executed automatically on every boot. No additional systemd setup is required.