systemd: Creating a service

Understanding and Managing Custom Services with systemd

systemd is the standard service and system manager for most Linux distributions, handling system startup, process management, and service control. If you're looking to automate or manage custom scripts or programs as services, systemd makes it easy to create and manage these services. Here's a comprehensive guide on creating, enabling, and managing custom services.

CHEATSHEET

1. **Create service file:** `/etc/systemd/system/my_custom_service.service`
2. **Reload systemd configuration:** `sudo systemctl daemon-reload`
3. **Enable service at startup:** `sudo systemctl enable my_custom_service.service`
4. **Start the service:** `sudo systemctl start my_custom_service.service`

DETAILS

Creating a Custom systemd Service

To create a custom service, start by creating a *service file*. This file will contain essential configurations and is typically stored in `/etc/systemd/system/`. Here's an example of a service file structure for running a custom script as a service:

```
[Unit]
Description=My Custom Service
After=network.target

[Service]
ExecStart=/usr/bin/my_script.sh
Type=simple

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

Breakdown of Service File Sections

- **[Unit]**: This section contains metadata about the service, such as its description and dependencies.

    - *Description*: Provides a brief overview of the service's purpose.
    - *After*: Specifies dependencies and ensures that `my_script.sh` runs only after the network is available.
- **[Service]**: Defines how the service should be executed.
    - *ExecStart*: Specifies the path to the executable or script. In this example, the service runs `/usr/bin/my_script.sh`.
    - *Type*: The service type determines how systemd manages the process. Common types include:
        - `simple`: Default type, used when the process doesn't fork or exit quickly.
        - `forking`: Used if the process forks into the background.
- **[Install]**: Determines how and when the service should be launched.
    - *WantedBy*: Specifies which target (runlevel) the service should start under. Setting this to `multi-user.target` means it will start when the system is in a multi-user, non-graphical environment.

Managing the Service with systemctl

Once you've created the service file, you'll need to use `systemctl`, the command-line tool for managing systemd services, to control it.

Steps to Enable and Start the Service

1. **Load the Service Configuration**
   To ensure systemd reads your new service file, run the following command:
   ```
   sudo systemctl daemon-reload
   ```
2. **Enable the Service at Boot**
   This makes the service start automatically when the system boots:
   ```
   sudo systemctl enable my_custom_service.service
   ```
3. **Start the Service**
   Once enabled, start the service manually for immediate execution:
   ```
   sudo systemctl start my_custom_service.service
   ```

Common systemctl Commands

- **Check Status**: See if the service is running and view recent log entries.
  ```
  sudo systemctl status my_custom_service.service
  ```

- **Stop the Service**: Stop the service manually.
  ```
  sudo systemctl stop my_custom_service.service
  ```
- **Restart the Service**: Restart the service to apply new changes.
  ```
  sudo systemctl restart my_custom_service.service
  ```
- **Disable the Service**: Prevent the service from starting automatically at boot.
  ```
  sudo systemctl disable my_custom_service.service
  ```

With these commands and an understanding of how to create a custom service file, you can effectively manage processes and tasks on your Linux system, enabling better automation and control.

Happy Me! □□

Revision #1
Created 31 October 2024 00:12:31 by Tiffanie BOREUX
Updated 31 October 2024 00:26:31 by Tiffanie BOREUX