If you want the output of your C executable to be displayed as soon as it runs, you have a few options, including logging, redirection, or using a simple shell script to capture the output and print it. Here's how you can achieve this:

1. \*\*Redirect Output to a Log File\*\*:

You can redirect the standard output (stdout) of your C executable to a log file, which can then be monitored for changes. This log file can be used to check the status of your program.

Modify your systemd unit file (`yourservice.service`) to redirect the output to a log file. Update the `ExecStart` line like this:

```plaintext

[Service]

ExecStart=/path/to/your/executable > /var/log/yourservice.log 2>&1

WorkingDirectory=/path/to/working/directory

Restart=always

User=yourusername

```

After making this change, you can view the log file at `/var/log/yourservice.log` to see the output of your C program:

```shell

tail -f /var/log/yourservice.log

```

2. \*\*Capture Output in a Shell Script\*\*:

You can create a simple shell script to run your C executable and print its output. Create a shell script, for example, `run\_yourservice.sh`, with the following content:

```bash

#!/bin/bash

/path/to/your/executable

```

Make the script executable:

```shell

chmod +x run\_yourservice.sh

```

Now, you can run the script, and it will execute your C program and display the output in the terminal:

```shell

./run\_yourservice.sh

```

The script will print the output to the terminal as soon as it's generated.

Choose the method that best suits your needs and integrates with your system's monitoring and management processes. Redirecting output to a log file is a common practice for long-running services, while using a shell script might be more suitable for interactive or debugging scenarios.

If you want to see the output of your C executable when you boot up the system, you can configure your systemd service to log its output during the boot process and display it on the console. To do this, you can use the `StandardOutput` and `StandardError` options in your systemd unit file.

1. Edit your systemd unit file (`yourservice.service`) and modify the `ExecStart` line and add the `StandardOutput` and `StandardError` options as follows:

```plaintext

[Service]

ExecStart=/path/to/your/executable

WorkingDirectory=/path/to/working/directory

StandardOutput=console

StandardError=console

User=yourusername

```

The `StandardOutput` and `StandardError` options are set to "console," which means the output and error messages will be displayed on the system console during boot.

2. Save the changes and exit the text editor.

3. Reload the systemd manager configuration:

```shell

sudo systemctl daemon-reload

```

4. Ensure your service is enabled to start at boot:

```shell

sudo systemctl enable yourservice

```

5. Reboot your system:

```shell

sudo reboot

```

After your system reboots, the output of your C executable will be displayed on the system console as part of the boot process.

Please replace `yourservice`, `/path/to/your/executable`, `/path/to/working/directory`, and `yourusername` with your actual service name, executable path, working directory, and the appropriate username.

If you want to customize the boot process to give you the option to run your program separately during startup, you can create a custom systemd target and service. This approach allows you to choose whether to run your program as part of the boot sequence.

Here's how you can achieve this:

1. \*\*Create a Custom systemd Target\*\*:

Create a custom systemd target that represents the option to run your program. This will allow you to control when your program runs during startup.

Create a new systemd target file, for example, `yourservice.target`, in the `/etc/systemd/system/` directory:

```shell

sudo nano /etc/systemd/system/yourservice.target

```

Add the following content to your `yourservice.target` file:

```plaintext

[Unit]

Description=Custom target for Your Service

Documentation=https://example.com/your-service-documentation

[Install]

WantedBy=multi-user.target

```

Save the file and exit the text editor.

2. \*\*Create a Custom systemd Service\*\*:

Next, create a systemd service unit file for your program. This service will be associated with your custom target.

Create a systemd service file, for example, `yourservice.service`, in the same directory:

```shell

sudo nano /etc/systemd/system/yourservice.service

```

Add the following content to your `yourservice.service` file, replacing `/path/to/your/executable` with the actual path to your C executable:

```plaintext

[Unit]

Description=Your Service

Documentation=https://example.com/your-service-documentation

Wants=yourservice.target

After=yourservice.target

[Service]

ExecStart=/path/to/your/executable

Restart=always

User=yourusername

[Install]

WantedBy=multi-user.target

```

Save the file and exit the text editor.

3. \*\*Reload systemd Manager Configuration\*\*:

After creating the target and service unit files, reload the systemd manager configuration:

```shell

sudo systemctl daemon-reload

```

4. \*\*Enable and Start Your Custom Service\*\*:

Enable the custom service to start at boot:

```shell

sudo systemctl enable yourservice.service

```

5. \*\*Boot and Select the Custom Target\*\*:

When you boot your system, it will start as usual. However, to run your program, you will need to manually select the custom target:

- During the boot process, you can typically press a key (e.g., `ESC`, `F2`, `F8`, or `Shift`) to access the boot menu or enter an interactive boot environment.

- In the boot menu or environment, look for an option to select different systemd targets or services.

- Choose the `yourservice.target` to run your program.

Please replace `yourservice`, `/path/to/your/executable`, `yourusername`, and the documentation URL with your actual values.

This approach allows you to have more control over when your program runs during startup by giving you the option to select the custom target when needed.

If you want to run an executable during system boot-up, you should integrate it as a system service. The exact steps may vary slightly depending on your Linux distribution, but here's a general guide for Red Hat Enterprise Linux (RHEL) or systems that use systemd as the init system.

1. \*\*Create a systemd Service Unit\*\*:

Create a systemd service unit file for your executable. This file defines how the service should be started and managed. Replace `yourservice` with an appropriate name:

```shell

sudo nano /etc/systemd/system/yourservice.service

```

Add the following content to your `yourservice.service` file:

```plaintext

[Unit]

Description=Your Service Description

[Service]

ExecStart=/path/to/your/executable

Restart=always

User=yourusername

[Install]

WantedBy=multi-user.target

```

- `ExecStart`: Specify the full path to your C executable.

- `User`: Specify the user account under which the service will run.

2. \*\*Reload systemd Manager Configuration\*\*:

After creating the service unit file, reload the systemd manager configuration:

```shell

sudo systemctl daemon-reload

```

3. \*\*Enable the Service\*\*:

To have your service start during boot, enable it:

```shell

sudo systemctl enable yourservice

```

4. \*\*Reboot\*\*:

After enabling the service, you can either reboot your system or start the service manually using:

```shell

sudo systemctl start yourservice

```

The service will automatically start during boot once it's enabled.

5. \*\*Check the Status\*\*:

You can check the status of your service to ensure it's running:

```shell

sudo systemctl status yourservice

```

Your C executable will now run as a service during the boot process, performing the necessary tasks.

Make sure to replace `yourservice`, `/path/to/your/executable`, `yourusername`, and the description with your actual service name, executable path, username, and a suitable description for your service.

If you want to run an executable during system boot-up, display its output on the screen, and then prompt the user to continue, you can use the following approach. This method involves using a custom systemd service that shows output on the screen and waits for user input.

Here are the steps:

1. \*\*Create a systemd Service Unit\*\*:

Create a systemd service unit file for your custom service. Replace `yourservice` with a suitable name:

```shell

sudo nano /etc/systemd/system/yourservice.service

```

Add the following content to your `yourservice.service` file:

```plaintext

[Unit]

Description=Your Custom Service

Documentation=https://example.com/your-service-documentation

[Service]

Type=simple

ExecStart=/path/to/your/executable

StandardInput=tty-force

StandardOutput=tty

TTYPath=/dev/console

User=yourusername

[Install]

WantedBy=multi-user.target

```

- `ExecStart`: Specify the full path to your C executable.

- `User`: Specify the user account under which the service will run.

2. \*\*Reload systemd Manager Configuration\*\*:

After creating the service unit file, reload the systemd manager configuration:

```shell

sudo systemctl daemon-reload

```

3. \*\*Enable the Service\*\*:

To have your service start during boot, enable it:

```shell

sudo systemctl enable yourservice

```

4. \*\*Reboot\*\*:

After enabling the service, you can either reboot your system or start the service manually using:

```shell

sudo systemctl start yourservice

```

Your executable will run during boot, and its output will be displayed on the screen. The service will wait for user input to continue.

5. \*\*User Input During Boot\*\*:

When the output of your executable is displayed on the screen, you can enter the necessary input to continue the boot process. The systemd service, configured as described above, will interact with the console.

This approach allows you to run your executable during system boot, display its output on the screen, and prompt the user to continue the boot process. Please make sure to replace `yourservice`, `/path/to/your/executable`, `yourusername`, and the documentation URL with your actual values.