If you want to run a service that displays output to the user after they log in, it's important to understand that services usually run in the background and don't directly interact with the user's graphical environment. If you want to display output to the user, you can consider using a script or program that's run at startup or login and interacts with the graphical environment or displays a terminal window.

Here are a couple of ways to achieve this:

1. \*\*Desktop Environment Startup Script:\*\*

You can create a script that runs when your desktop environment starts. The steps may vary depending on your desktop environment. For example, in a GNOME-based environment, you can add a `.desktop` file to the `~/.config/autostart/` directory. Here's an example of a simple `.desktop` file that runs a script:

```desktop

[Desktop Entry]

Type=Application

Name=MyStartupScript

Exec=/path/to/your/script.sh

Terminal=true

```

Replace `/path/to/your/script.sh` with the path to your script, and set `Terminal=true` if you want a terminal window to open and display the script's output.

2. \*\*Use a Startup Script:\*\*

You can create a script that runs at user login. The specifics depend on your desktop environment and user preferences. Here's a general example:

a. Create a shell script (e.g., `startup\_script.sh`) that contains your desired commands and output.

b. Make the script executable:

```bash

chmod +x /path/to/your/startup\_script.sh

```

c. Add the script to your user's startup applications or shell configuration file (e.g., `.bashrc` or `.bash\_profile` for bash users). For example, you can add the following line to `~/.bashrc`:

```bash

/path/to/your/startup\_script.sh

```

d. Log out and log back in, and the script will run, displaying its output.

These methods allow you to run a script or program that displays output to the user after they log in, rather than running it as a traditional background service. Be sure to tailor these instructions to your specific environment and requirements.

If you want your executable to display output to the screen after logging in and have it start automatically every time you log in, you can achieve this by configuring it as a user-level systemd service and using the user session environment. Follow these steps:

1. Create a user-level systemd service unit file:

Create a service unit file in your user's home directory. For example, create a file called `my\_service.service` in `~/.config/systemd/user/`:

```

nano ~/.config/systemd/user/my\_service.service

```

2. Add the following content to the `my\_service.service` file, adjusting the path to your executable:

```systemd

[Unit]

Description=My User Service

After=default.target

[Service]

ExecStart=/path/to/your/executable

Restart=always

StandardOutput=tty

StandardInput=tty

TTYPath=/dev/console

[Install]

WantedBy=default.target

```

Replace `/path/to/your/executable` with the actual path to your executable.

3. Save the file and exit the text editor.

4. Enable and start the user service:

```

systemctl --user enable my\_service.service

systemctl --user start my\_service.service

```

5. Reboot your system or log out and log back in.

Upon logging in, your service will start automatically and display the output on your screen.

This configuration ensures that the output of your executable is attached to your user's terminal, so you will see the output after login. If you wish to redirect the output to a specific terminal or have more control over where the output is displayed, additional configuration may be required.

To achieve this, you can create a custom systemd service that checks if a key is present during the boot process, and based on the result, it can run another service to interact with the user for loading or overwriting the key. Here are the steps to implement this:

1. Create a service to check for the key:

Create a service unit file for key checking. You can create a file called `keycheck.service` in `/etc/systemd/system/`.

```ini

[Unit]

Description=Key Checking Service

After=network.target

[Service]

Type=oneshot

ExecStart=/path/to/keychecker

RemainAfterExit=yes

[Install]

WantedBy=multi-user.target

```

Replace `/path/to/keychecker` with the actual path to your `keychecker` executable.

2. Create a service to interact with the user for key loading:

Create a service unit file for the client key loader. You can create a file called `clientkeyloader.service` in `/etc/systemd/system/`.

```ini

[Unit]

Description=Client Key Loader Service

After=network.target keycheck.service

[Service]

Type=simple

ExecStart=/path/to/client\_key\_loader

Restart=always

[Install]

WantedBy=multi-user.target

```

Replace `/path/to/client\_key\_loader` with the actual path to your `client\_key\_loader` executable.

3. Create a systemd target:

To control the order in which these services are executed, create a custom systemd target. Create a file called `mytarget.target` in `/etc/systemd/system/`:

```ini

[Unit]

Description=Custom Target to Control Key Checking and Loading

[Install]

WantedBy=multi-user.target

```

4. Configure the key loader service to depend on the key checking service:

Edit the `clientkeyloader.service` file and add a dependency on `keycheck.service` by including it in the `After` line in the `[Unit]` section.

```ini

[Unit]

Description=Client Key Loader Service

After=network.target keycheck.service

```

5. Enable and start the target:

Enable the custom target and the key loader service:

```bash

sudo systemctl enable mytarget.target

sudo systemctl enable clientkeyloader.service

```

6. Reboot your system:

After the system reboots, the `keycheck.service` will run first, checking if the key is present or not. Depending on the result, the `clientkeyloader.service` will run to interact with the user for loading or overwriting the key.

Ensure that your `keychecker` and `client\_key\_loader` executables are properly configured to check for the key presence and handle user interaction as needed.

To achieve the desired behavior where your executable called `keychecker` checks if a key is present during boot-up and then, based on the result, runs another executable called `client\_key\_loader`, you can follow these steps:

1. Create a systemd service for `keychecker`:

Create a systemd service unit file for `keychecker` in the `/etc/systemd/system/` directory. This service will run your `keychecker` executable during boot-up.

Create a file named `keychecker.service`:

```

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

```

Add the following content to the `keychecker.service` file:

```ini

[Unit]

Description=Check for Key Presence

[Service]

Type=oneshot

ExecStart=/path/to/keychecker

[Install]

WantedBy=multi-user.target

```

Replace `/path/to/keychecker` with the actual path to your `keychecker` executable.

2. Create a systemd service for `client\_key\_loader`:

Create another systemd service unit file for `client\_key\_loader` in the same directory. This service will run `client\_key\_loader` if the key is present.

Create a file named `client\_key\_loader.service`:

```

sudo nano /etc/systemd/system/client\_key\_loader.service

```

Add the following content to the `client\_key\_loader.service` file:

```ini

[Unit]

Description=Client Key Loader

[Service]

Type=oneshot

ExecStart=/path/to/client\_key\_loader

RemainAfterExit=true

[Install]

WantedBy=multi-user.target

```

Replace `/path/to/client\_key\_loader` with the actual path to your `client\_key\_loader` executable.

3. Create a systemd target to ensure the order of execution:

Create a custom systemd target unit file, which will ensure that `client\_key\_loader` runs only after `keychecker`.

Create a file named `my\_target.target`:

```

sudo nano /etc/systemd/system/my\_target.target

```

Add the following content to the `my\_target.target` file:

```ini

[Unit]

Description=Custom Target for Key Checking and Loading

[Install]

WantedBy=multi-user.target

```

4. Create dependencies between the services:

Edit the `keychecker.service` and `client\_key\_loader.service` files to specify the `After` and `Wants` directives to ensure the proper execution order. Edit the `[Unit]` section of `keychecker.service` as follows:

```ini

[Unit]

Description=Check for Key Presence

After=local-fs.target

Wants=client\_key\_loader.service

[Service]

Type=oneshot

ExecStart=/path/to/keychecker

[Install]

WantedBy=my\_target.target

```

Edit the `[Unit]` section of `client\_key\_loader.service` as follows:

```ini

[Unit]

Description=Client Key Loader

After=keychecker.service

Wants=my\_target.target

[Service]

Type=oneshot

ExecStart=/path/to/client\_key\_loader

RemainAfterExit=true

[Install]

WantedBy=my\_target.target

```

These changes ensure that `client\_key\_loader` starts only after `keychecker` has completed, and they both belong to the `my\_target.target` unit.

5. Reload systemd and enable the services:

```

sudo systemctl daemon-reload

sudo systemctl enable keychecker.service

sudo systemctl enable client\_key\_loader.service

sudo systemctl enable my\_target.target

```

6. Reboot your system to test the configuration:

```

sudo reboot

```

During boot-up, `keychecker` will check if the key is present, and if it is, it will trigger the `client\_key\_loader` service. The order of execution is controlled by the custom `my\_target.target`.

Make sure to replace `/path/to/keychecker` and `/path/to/client\_key\_loader` with the actual paths to your executables. You should also ensure that your `keychecker` program correctly identifies whether the key is present or not and returns the appropriate exit code or status.