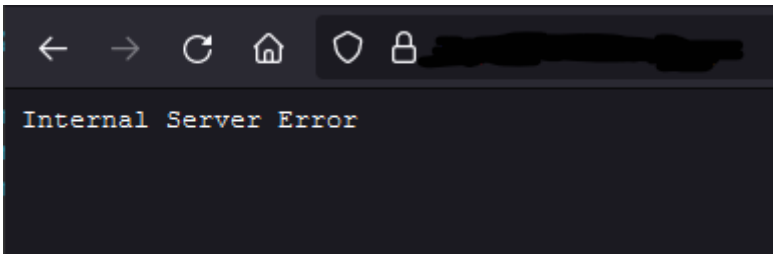
Searx

The ultimate meta-search engine. Hide your search history from Google and others.

- Fixing Searx after Debian 10 to Debian 11 upgrade
- Unix Socket

Fixing Searx after Debian 10 to Debian 11 upgrade

I have recently upgraded my Searx server instance from Debian 10 to Debian 11. Apart from not being able to boot on the first try due to a backported kernel, all of my services also stopped working. Today we will look at Searx which currently responds with the following message:



TLDR; It was a Python version issue, reinstalled Searx while keeping settings.

Troubleshooting

Just like always when we have a problem, we need to start troubleshooting somewhere. Each person approahes troubleshooting differently, here's what I will do in this case.

> By the way – it may be written and sound like a tutorial, but it's mostly me listing out things I did to fix it, even though some of them did not work or were completely useless + my setup is sometimes a bit specific and the same things may not apply for your environment. Just keep that in mind.

Restart Searx

Who doesn't know the phrase "have you tried turning it off and on again?". Most of you probably do. Sometimes a quick restart is all that's needed. However we first need to figure out *what* to restart by listing all services on the system. There are a couple of ways to do this, since Debian uses SystemD as its init system, we can issue the following command:

```
$ systemctl --type=service


or
```

```
$ systemctl list-units --type=service
```

Both should give you the same output, similar to this:

```
  UNIT                                LOAD   ACTIVE     SUB         DESCRIPTION
● apache2.service                     loaded failed     failed      The Apache HTTP Server
● certbot.service                     loaded failed     failed      Certbot
  console-getty.service               loaded active     running     Console Getty
  dbus.service                        loaded active     running     D-Bus System Message Bus
  filtron.service                     loaded active     running     filtron
  ifupdown-pre.service                loaded active     exited      Helper to synchronize boot up for ifupdown
  networking.service                  loaded active     exited      Raise network interfaces
  nginx.service                       loaded active     running     nginx - high performance web server
  ssh.service                         loaded active     running     OpenBSD Secure Shell server
  systemd-journal-flush.service       loaded active     exited      Flush Journal to Persistent Storage
  systemd-journald.service            loaded active     running     Journal Service
  systemd-logind.service              loaded active     running     User Login Management
  systemd-modules-load.service        loaded active     exited      Load Kernel Modules
  systemd-networkd.service            loaded active     running     Network Service
  systemd-remount-fs.service          loaded active     exited      Remount Root and Kernel File Systems
  systemd-resolved.service            loaded active     running     Network Name Resolution
  systemd-sysctl.service              loaded active     exited      Apply Kernel Variables
  systemd-sysusers.service            loaded active     exited      Create System Users
  systemd-tmpfiles-setup-dev.service loaded active     exited      Create Static Device Nodes in /dev
  systemd-tmpfiles-setup.service      loaded active     exited      Create Volatile Files and Directories
  systemd-update-utmp.service         loaded active     exited      Update UTMP about System Boot/Shutdown
  systemd-user-sessions.service       loaded active     exited      Permit User Sessions
  user-runtime-dir@1000.service       loaded active     exited      User Runtime Directory /run/user/1000
  user@1000.service                   loaded active     running     User Manager for UID 1000
  uwsgi.service                       loaded active     running     LSB: Start/stop uWSGI server instance(s)
  whoogle.service                     loaded activating auto-restart Whoogle
```

Searx runs on uWSGI, which is, as its creators call it *"Swiss Army Knife for your network applications"*. I only have one service running on uWSGI, which means I can restart the **entire server** without worrying.

> **If there are other working services depending on uWSGI, they will be restarted as well.**

Firstly, check status of the uWSGI service.

```
$ systemctl status uwsgi
```

As we can see, it is active and running and has loaded */etc/uwsgi/apps-enabled/searx.ini*

```
● uwsgi.service - LSB: Start/stop uWSGI server instance(s)
     Loaded: loaded (/etc/init.d/uwsgi; generated)
     Active: active (running) since Fri 2021-09-10 10:42:33 CEST; 4 days ago
       Docs: man:systemd-sysv-generator(8)
    Process: 2837 ExecStart=/etc/init.d/uwsgi start (code=exited, status=0/SUCCESS)
       CPU: 48.725s
     CGroup: /system.slice/uwsgi.service
             ├─2938 /usr/bin/uwsgi --ini /usr/share/uwsgi/conf/default.ini --ini /etc/uwsgi/apps-enabled/searx.ini --
daemoni>
             ├─2941 /usr/bin/uwsgi --ini /usr/share/uwsgi/conf/default.ini --ini /etc/uwsgi/apps-enabled/searx.ini --
daemoni>
             ├─2942 /usr/bin/uwsgi --ini /usr/share/uwsgi/conf/default.ini --ini /etc/uwsgi/apps-enabled/searx.ini --
daemoni>
             └─2944 /usr/bin/uwsgi --ini /usr/share/uwsgi/conf/default.ini --ini /etc/uwsgi/apps-enabled/searx.ini --
daemoni>

Warning: some journal files were not opened due to insufficient permissions.
```

To restart, we need to elevate using *sudo.*

```
$ sudo systemctl restart uwsgi
```

Check if the service is running like before with *systemctl status.* Unfortunately in my case, the restart didn't help. Let's finally dig deeper.

Consult Logs

Logs for uWSGI are located in */var/log/uwsgi/app.* We are looking for *searx.log* which should be present within this folder. Open it using your favorite method, I will use *less* and jump to end. Heads up – this file requires elevated privileges to be opened.

```
$ sudo less searx.log
```

The log is fairly verbose and understandable. This part describes the killing and initialization process of uWSGI and also the attempted start of Searx.

```
Tue Sep 14 23:12:10 2021 - SIGINT/SIGQUIT received...killing workers...
Tue Sep 14 23:12:10 2021 - gateway "uWSGI http 1" has been buried (pid: 2944)
Tue Sep 14 23:12:11 2021 - worker 1 buried after 1 seconds
```

Tue Sep 14 23:12:11 2021 - worker 2 buried after 1 seconds
Tue Sep 14 23:12:11 2021 - goodbye to uWSGI.
Tue Sep 14 23:12:12 2021 - *** Starting uWSGI 2.0.19.1-debian (64bit) on [Tue Sep 14 23:12:12 2021] ***
Tue Sep 14 23:12:12 2021 - compiled with version: 10.2.1 20210110 on 11 June 2021 09:08:33
Tue Sep 14 23:12:12 2021 - os: Linux-5.4.0-74-generic #83~18.04.1-Ubuntu SMP Tue May 11 16:01:00 UTC 2021
Tue Sep 14 23:12:12 2021 - nodename: hostname
Tue Sep 14 23:12:12 2021 - machine: x86_64
Tue Sep 14 23:12:12 2021 - clock source: unix
Tue Sep 14 23:12:12 2021 - pcre jit disabled
Tue Sep 14 23:12:12 2021 - detected number of CPU cores: 8
Tue Sep 14 23:12:12 2021 - current working directory: /
Tue Sep 14 23:12:12 2021 - writing pidfile to /run/uwsgi/app/searx/pid
Tue Sep 14 23:12:12 2021 - detected binary path: /usr/bin/uwsgi-core
Tue Sep 14 23:12:12 2021 - chdir() to /var/www/searx/searx
Tue Sep 14 23:12:12 2021 - your memory page size is 4096 bytes
Tue Sep 14 23:12:12 2021 - detected max file descriptor number: 1024
Tue Sep 14 23:12:12 2021 - lock engine: pthread robust mutexes
Tue Sep 14 23:12:12 2021 - thunder lock: disabled (you can enable it with --thunder-lock)
Tue Sep 14 23:12:12 2021 - *** Cache "searxcache" initialized: 11MB (key: 2136 bytes, keys: 4272000 bytes, data: 8192000 bytes, bitmap: 250 bytes) preallocated ***
Tue Sep 14 23:12:12 2021 - uWSGI http bound on 127.0.0.1:8888 fd 5
Tue Sep 14 23:12:12 2021 - uwsgi socket 0 bound to UNIX address /run/uwsgi/app/searx/socket fd 8
Tue Sep 14 23:12:12 2021 - setgid() to 1002
Tue Sep 14 23:12:12 2021 - setuid() to 1002
Tue Sep 14 23:12:12 2021 - Python version: 3.9.2 (default, Feb 28 2021, 17:03:44) [GCC 10.2.1 20210110]
Tue Sep 14 23:12:12 2021 - PEP 405 virtualenv detected: /var/www/searx/searx-pyenv
Tue Sep 14 23:12:12 2021 - Set PythonHome to /var/www/searx/searx-pyenv
Tue Sep 14 23:12:12 2021 - Python main interpreter initialized at 0x564252505c10
Tue Sep 14 23:12:12 2021 - python threads support enabled
Tue Sep 14 23:12:12 2021 - your server socket listen backlog is limited to 100 connections
Tue Sep 14 23:12:12 2021 - your mercy for graceful operations on workers is 60 seconds
Tue Sep 14 23:12:12 2021 - mapped 218808 bytes (213 KB) for 2 cores
Tue Sep 14 23:12:12 2021 - *** Operational MODE: preforking ***
Tue Sep 14 23:12:12 2021 - added /var/www/searx/searx/ to pythonpath.
Tue Sep 14 23:12:12 2021 - spawned uWSGI master process (pid: 136045)
Tue Sep 14 23:12:12 2021 - spawned uWSGI worker 1 (pid: 136048, cores: 1)
Tue Sep 14 23:12:12 2021 - spawned uWSGI worker 2 (pid: 136049, cores: 1)
Tue Sep 14 23:12:12 2021 - cache sweeper thread enabled
Tue Sep 14 23:12:12 2021 - spawned uWSGI http 1 (pid: 136051)
ModuleNotFoundError: No module named 'searx'

```
Tue Sep 14 23:12:12 2021 - unable to load app 0 (mountpoint='') (callable not found or import error)
Tue Sep 14 23:12:12 2021 - *** no app loaded. going in full dynamic mode ***
ModuleNotFoundError: No module named 'searx'
Tue Sep 14 23:12:12 2021 - unable to load app 0 (mountpoint='') (callable not found or import error)
Tue Sep 14 23:12:12 2021 - *** no app loaded. going in full dynamic mode ***
Tue Sep 14 23:12:24 2021 - --- no python application found, check your startup logs for errors ---
```

It is all looking fine until these lines:

```
ModuleNotFoundError: No module named 'searx'
Tue Sep 14 23:12:12 2021 - unable to load app 0 (mountpoint='') (callable not found or import error)
Tue Sep 14 23:12:12 2021 - *** no app loaded. going in full dynamic mode ***
```

Searx also includes helpful utilities and debug scripts. Try running:

```
$ sudo /var/www/searx/utils/searx.sh inspect service
```

```
service status & log
====================

sourced .config.sh :

  PUBLIC_URL          : http://hostname/searx
  SEARX_URL_PATH      : /searx
  SEARX_INSTANCE_NAME : searx@hostname
  SEARX_INTERNAL_HTTP  : 127.0.0.1:8888

ERROR: Service account searx does not exists!
ERROR: ~searx: python environment is not available!
ERROR: ~searx: Missing searx software!
INFO:  uWSGI app searx.ini is enabled.
INFO:  public URL   --> http://hostname/searx
INFO:  internal URL --> http://127.0.0.1:8888
INFO:  got 500 from http://127.0.0.1:8888
INFO:  got 301 from http://hostname/searx

Enable searx debug mode? [NO/yes]
No

Unit searx.service could not be found.
// use CTRL-C to stop monitoring the log
```

```
Wed Sep 15 00:01:07 2021 - spawned uWSGI http 1 (pid: 137252)
ModuleNotFoundError: No module named 'searx'
Wed Sep 15 00:01:07 2021 - unable to load app 0 (mountpoint='') (callable not found or import error)
Wed Sep 15 00:01:07 2021 - *** no app loaded. going in full dynamic mode ***
ModuleNotFoundError: No module named 'searx'
Wed Sep 15 00:01:07 2021 - unable to load app 0 (mountpoint='') (callable not found or import error)
Wed Sep 15 00:01:07 2021 - *** no app loaded. going in full dynamic mode ***
Wed Sep 15 00:01:21 2021 - --- no python application found, check your startup logs for errors ---
Wed Sep 15 00:01:21 2021 - --- no python application found, check your startup logs for errors ---
Wed Sep 15 00:04:03 2021 - --- no python application found, check your startup logs for errors ---
```

The errors at the bottom are identical to what we got from the uWSGI log, however there is more information at the top of the command output. Apparently, there is not searx service account, which is false, because there is one. Confirm by looking at */etc/passwd* file:

```
$ cat /etc/passwd
...
searx:x:1002:1002:,,,:/home/searx:/usr/sbin/nologin
...
```

You could argue that it is indeed *not* a service account, because it has UID over 1000, but keep in mind that the official install guide works with the idea of having a HOME directory for this user, which itself goes against the general idea of *service account*. What is more important is the fact that the user doesn't have access to shell.

> It is clear to me as of now, that it has something to do with Python. The previous Debian version came with Python 3.7.3, while this comes with 3.9.2. This shouldn't be a problem however, since has Searx supported Python 3.9 for a while now.

Temporarily add shell to Searx user

We will be using the Searx's account to do the next steps, since I don't want to mess up some permissions by running the commands as a different user and because everything is owned by searx account. If you don't want to do that, you can still issue command under searx account, but specify bash manually in each command like this:

```
$ su - searx -s /bin/bash -c 'command'
```

Modify */etc/passwd* with your favorite editor. And replace */usr/sbin/nologin* with */bin/bash* on the searx line.

```
$ sudo vi /etc/passwd
```

```
searx:x:1002:1002:,,,:/home/searx:/usr/sbin/nologin
```

to

```
searx:x:1002:1002:,,,:/home/searx:/bin/bash
```

Update packages and add to PATH

Run the following script to update Searx's packages, just in case.

```
$ sudo -u searx ./manage.sh update_packages
```

It will warn you about some location not being in the path, we can try that and see

```
WARNING: The scripts pip, pip3 and pip3.9 are installed in '/home/searx/.local/bin' which is not on PATH.
 Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
```

Add /home/searx/.local/bin to PATH

Assuming you are still under the searx user, open /home/searx/.bashrc in your favorite editor.

```
$ vi /home/searx/.bashrc
```

and add this at the end of the file:

```
export PATH=/home/searx/.local/bin:$PATH
```

Now exit and sudo su to searx again and confirm that the new location is in PATH

```
$ echo $PATH
/home/searx/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Create Python virtualenv

Switch to searx user.

```
$ sudo su searx
```

Check python version.

```
$ python3 --version
Python 3.9.2
```

Create virtualenv in searx source folder.

```
$ python3 -m venv /var/www/searx/searx
```

aaand...it did literally nothing. No output and by checking, we can see that the binary isn't sourced from virtualenv...

```
$ command -v python3
/usr/bin/python3
```

After a brief moment, I figured I need to activate the virtualenv, make sure you are still under searx user.

```
$ cd /var/www/searx/searx-pyenv
```

```
$ source bin/activate
```

There should now be (searx-pyenv) before your username@server.

```
(searx-pyenv) searx@hostname:/var/www/searx/searx-pyenv$
```

Now check again

```
$ command -v python3
/var/www/searx/searx-pyenv/bin/python3

$ python3 --version
Python 3.9.2
```

> **Keep the terminal with searx-pyenv open and run the rest of the commands in it!**

Install pip boilerplate

Now that we are at least in the virtualenv, reinstall pip boilerplate packages which are the following:

```
searx$ pip install -U pip
searx$ pip install -U setuptools
searx$ pip install -U wheel
searx$ pip install -U pyyaml
```

Some packages will be reinstalled to the same version while some may be upgraded.

Move to the root directory of searx (*/var/www/searx*) in my case.

```
searx$ cd /var/www/searx
```

and install Searx into the virtualenv (notice the dot at the end)

```
searx$ pip install -e .
```

That should spit out output similar to this:

```
Obtaining file:///var/www/searx
Collecting certifi==2020.12.05
  Using cached certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting babel==2.9.0
  Using cached Babel-2.9.0-py2.py3-none-any.whl (8.8 MB)
Collecting flask-babel==2.0.0
  Using cached Flask_Babel-2.0.0-py3-none-any.whl (9.3 kB)
Collecting flask==1.1.2
  Using cached Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting idna==2.10
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting jinja2==2.11.3
  Using cached Jinja2-2.11.3-py2.py3-none-any.whl (125 kB)
Collecting lxml==4.6.3
  Using cached lxml-4.6.3-cp39-cp39-manylinux2014_x86_64.whl (6.9 MB)
Collecting pygments==2.8.0
  Using cached Pygments-2.8.0-py3-none-any.whl (983 kB)
Collecting python-dateutil==2.8.1
  Using cached python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
Requirement already satisfied: pyyaml==5.4.1 in ./searx-pyenv/lib/python3.9/site-packages (from
searx==1.0.0) (5.4.1)
Collecting requests[socks]==2.25.1
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting langdetect==1.0.8
  Using cached langdetect-1.0.8-py3-none-any.whl
Collecting pytz>=2015.7
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)
Collecting itsdangerous>=0.24
  Using cached itsdangerous-2.0.1-py3-none-any.whl (18 kB)
Collecting Werkzeug>=0.15
  Using cached Werkzeug-2.0.1-py3-none-any.whl (288 kB)
Collecting click>=5.1
  Using cached click-8.0.1-py3-none-any.whl (97 kB)
```

```
Collecting MarkupSafe>=0.23
  Using cached MarkupSafe-2.0.1-cp39-cp39-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (30 kB)
Collecting six
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting chardet<5,>=3.0.2
  Using cached chardet-4.0.0-py2.py3-none-any.whl (178 kB)
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.6-py2.py3-none-any.whl (138 kB)
     |████████████████████████████████| 138 kB 3.6 MB/s
Collecting PySocks!=1.5.7,>=1.5.6
  Using cached PySocks-1.7.1-py3-none-any.whl (16 kB)
Installing collected packages: MarkupSafe, Werkzeug, urllib3, pytz, jinja2, itsdangerous, idna, click, chardet,
certifi, six, requests, PySocks, flask, babel, python-dateutil, pygments, lxml, langdetect, flask-babel, searx
  Running setup.py develop for searx
Successfully installed MarkupSafe-2.0.1 PySocks-1.7.1 Werkzeug-2.0.1 babel-2.9.0 certifi-2020.12.5 chardet-
4.0.0 click-8.0.1 flask-1.1.2 flask-babel-2.0.0 idna-2.10 itsdangerous-2.0.1 jinja2-2.11.3 langdetect-1.0.8 lxml-
4.6.3 pygments-2.8.0 python-dateutil-2.8.1 pytz-2021.1 requests-2.25.1 searx-1.0.0 six-1.16.0 urllib3-1.26.6
```

Test the development server

Make sure you are still in the searx-pyenv terminal and root searx folder /var/www/searx and try running the development server.

```
searx$ python searx/webapp.py
```

```
* Serving Flask app "webapp" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
Traceback (most recent call last):
  File "/var/www/searx/searx/webapp.py", line 1155, in <module>
    run()
  File "/var/www/searx/searx/webapp.py", line 1104, in run
    app.run(
  File "/var/www/searx/searx-pyenv/lib/python3.9/site-packages/flask/app.py", line 990, in run
    run_simple(host, port, self, **options)
  File "/var/www/searx/searx-pyenv/lib/python3.9/site-packages/werkzeug/serving.py", line 1008, in run_simple
    inner()
  File "/var/www/searx/searx-pyenv/lib/python3.9/site-packages/werkzeug/serving.py", line 948, in inner
```

```
    srv = make_server(
  File "/var/www/searx/searx-pyenv/lib/python3.9/site-packages/werkzeug/serving.py", line 780, in make_server
    return ThreadedWSGIServer(
  File "/var/www/searx/searx-pyenv/lib/python3.9/site-packages/werkzeug/serving.py", line 686, in __init__
    super().__init__(server_address, handler)  # type: ignore
  File "/usr/lib/python3.9/socketserver.py", line 452, in __init__
    self.server_bind()
  File "/usr/lib/python3.9/http/server.py", line 138, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "/usr/lib/python3.9/socketserver.py", line 466, in server_bind
    self.socket.bind(self.server_address)
OSError: [Errno 98] Address already in use
```

Okay, something happened, but we have an error at the end *Address already in use*. Of course! We have uWSGI running at it might have already started Searx in production mode. To test this kill uWSGI.

```
$ sudo systemctl stop uwsgi
```

Start the development server again.

```
searx$ python searx/webapp.py
```

This time, it starts sucessfully on port 8888 and when we navigate to the website, we can see it working and logging:

```
 * Serving Flask app "webapp" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
INFO:werkzeug: * Running on http://127.0.0.1:8888/ (Press CTRL+C to quit)
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET / HTTP/1.0" 200 -
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET /static/themes/oscar/css/bootstrap.min.css HTTP/1.0" 200 -
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET /static/themes/oscar/css/logicodev-dark.min.css HTTP/1.0" 200 -
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET /static/themes/oscar/img/logo_searx_a.png HTTP/1.0" 200 -
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET /static/themes/oscar/js/jquery.min.js HTTP/1.0" 304 -
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET /static/themes/oscar/js/bootstrap.min.js HTTP/1.0" 304 -
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET /static/themes/oscar/js/searx.min.js HTTP/1.0" 304 -
```

```
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET /static/plugins/js/search_on_category_select.js HTTP/1.0" 304 -
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET /static/themes/oscar/fonts/glyphicons-halflings-regular.woff2 HTTP/1.0" 200 -
INFO:werkzeug:your_ip - - [15/Sep/2021 01:22:43] "GET /static/themes/oscar/img/favicon.png HTTP/1.0" 200 -
```

Revert changes to the Searx user

> This is important, before returning Searx into production, change the shell of searx account back to */usr/sbin/nologin* in */etc/passwd*

```
searx:x:1002:1002:,,,:/home/searx:/usr/sbin/nologin
```

Move to production

Now that we have confirmed that it's working, kill the development server with Ctrl+C and restart uWSGI service.

```
$ sudo systemctl restart uWSGI
```

> Navigate to your URL and Searx should be up and running again.

Confirm that everything is working properly

Look into the log file in /var/log/uwsgi/app and if you don't see any errors, it should be okay.

Unix Socket