Exercise 2.6: Create and customize a systemd service

This exercise is going to explore the various configuration directories for a **systemd** service. The application **stress** will be used, install the package **stress** with the appropriate package installer and examine the files installed. Notice the absolute path name of the binary **stress** and the absence of a **stress.service** file in the **systemd** file structure.

Very Important

Some distributions have the **stress** package; others have the newer **stress-ng**. Either package will work as **stress-ng** supports the same command line options as **stress**.

On CentOS

```
# yum install stress
# rpm -ql stress
```

On openSUSE

```
# zypper install stress-ng
# rpm -ql stress-ng
```

On Ubuntu

```
# apt-get install stress
# dpkg -L stress
```

The **stress** package does not include a **systemd** unit configuration, so one must be created. The package installed on the test system has the binary for **stress** as `/usr/bin/stress`. Create a **systemd** vendor unit file as `/usr/lib/systemd/system/foo.service`. You will require **root** level access to create a file in this directory.

```
$ sudo  vim /usr/lib/systemd/system/foo.service
```

/usr/lib/systemd/system/foo.service

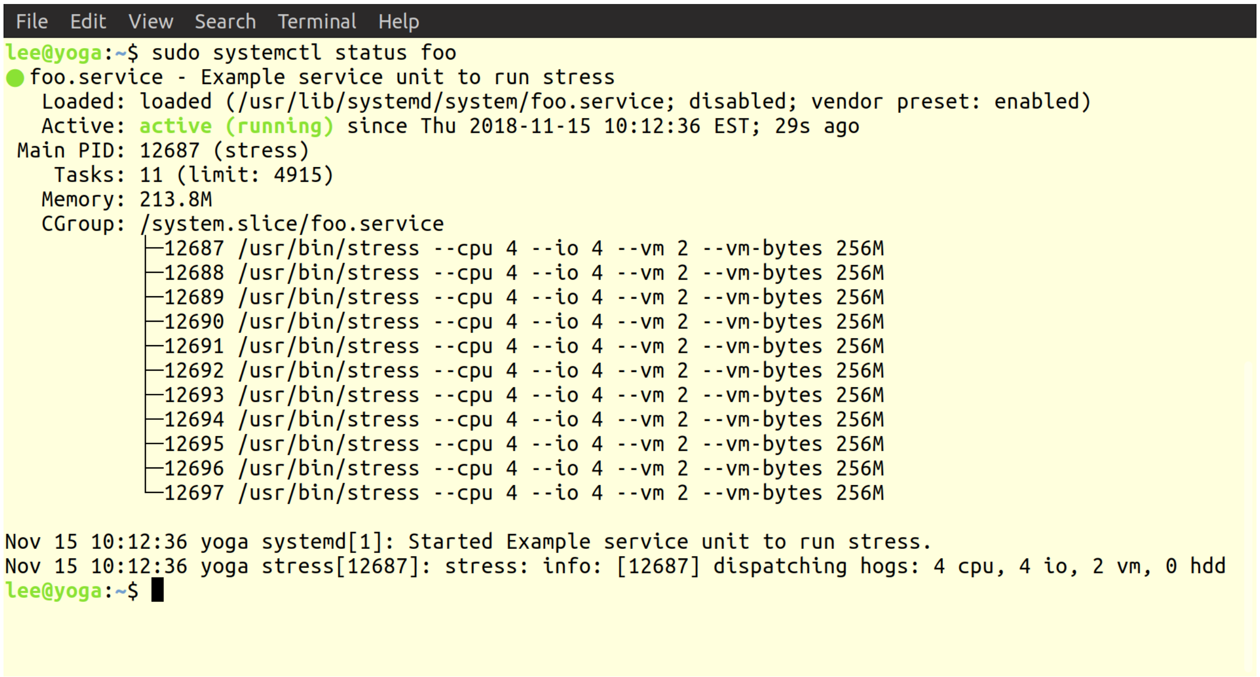
```
[Unit]
Description=Example service unit to run stress
[Service]
ExecStart=/usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
[Install]
WantedBy=multi-user.target
```

A copy of this file (`foo1.service`) can be found in the tarball in the `LFS211/SOLUTIONS/s_02/` directory.

Once the unit file is created **systemd** will be able to start and stop the service. Use the **top** command to verify that **stress** is working. The following commands may be useful:

```
# systemctl daemon-reload
# systemctl start foo
# systemctl status foo -l
# systemd-delta
# systemctl stop foo
```

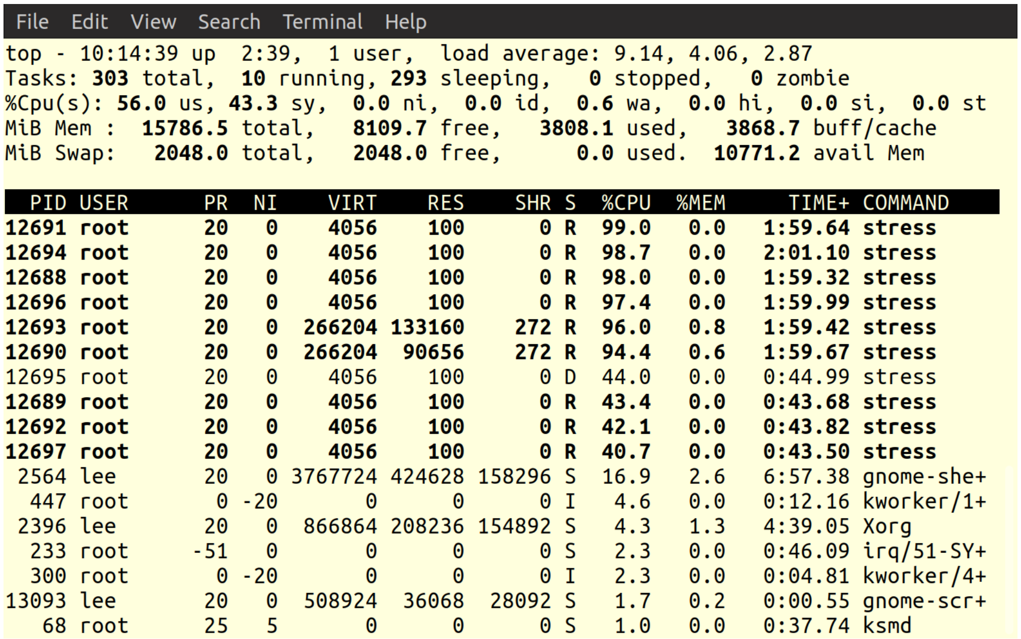
The example program **stress** which is now a service, does not display much feedback as to what it is doing. The **systemctl status** of the service can be checked, the output would look something like this:



```
lee@yoga:~$ sudo systemctl status foo
● foo.service - Example service unit to run stress
   Loaded: loaded (/usr/lib/systemd/system/foo.service; disabled; vendor preset: enabled)
   Active: active (running) since Thu 2018-11-15 10:12:36 EST; 29s ago
 Main PID: 12687 (stress)
    Tasks: 11 (limit: 4915)
   Memory: 213.8M
   CGroup: /system.slice/foo.service
           ├─12687 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           ├─12688 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           ├─12689 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           ├─12690 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           ├─12691 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           ├─12692 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           ├─12693 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           ├─12694 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           ├─12695 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           ├─12696 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M
           └─12697 /usr/bin/stress --cpu 4 --io 4 --vm 2 --vm-bytes 256M

Nov 15 10:12:36 yoga systemd[1]: Started Example service unit to run stress.
Nov 15 10:12:36 yoga stress[12687]: stress: info: [12687] dispatching hogs: 4 cpu, 4 io, 2 vm, 0 hdd
lee@yoga:~$ █
```

Figure 2.18: **systemctl status foo**

Examining the output of the **top** command will show two processes of **stress** with more than the specified 256M of memory, four processes of **stress** with nearly 100% CPU usage and four processes with neither high CPU or high memory usage. They would be the memory hogs, CPU hogs, and io hogs, respectively. See the example below:



```
top - 10:14:39 up  2:39,  1 user,  load average: 9.14, 4.06, 2.87
Tasks: 303 total,  10 running, 293 sleeping,   0 stopped,   0 zombie
%Cpu(s): 56.0 us, 43.3 sy,  0.0 ni,  0.0 id,  0.6 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  15786.5 total,   8109.7 free,   3808.1 used,   3868.7 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.  10771.2 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
12691 root      20   0    4056    100      0 R  99.0   0.0   1:59.64 stress
12694 root      20   0    4056    100      0 R  98.7   0.0   2:01.10 stress
12688 root      20   0    4056    100      0 R  98.0   0.0   1:59.32 stress
12696 root      20   0    4056    100      0 R  97.4   0.0   1:59.99 stress
12693 root      20   0  266204 133160    272 R  96.0   0.8   1:59.42 stress
12690 root      20   0  266204  90656    272 R  94.4   0.6   1:59.67 stress
12695 root      20   0    4056    100      0 D  44.0   0.0   0:44.99 stress
12689 root      20   0    4056    100      0 R  43.4   0.0   0:43.68 stress
12692 root      20   0    4056    100      0 R  42.1   0.0   0:43.82 stress
12697 root      20   0    4056    100      0 R  40.7   0.0   0:43.50 stress
 2564 lee       20   0 3767724 424628 158296 S  16.9   2.6   6:57.38 gnome-she+
  447 root       0 -20       0      0      0 I   4.6   0.0   0:12.16 kworker/1+
 2396 lee       20   0  866864 208236 154892 S   4.3   1.3   4:39.05 Xorg
  233 root     -51   0       0      0      0 S   2.3   0.0   0:46.09 irq/51-SY+
  300 root       0 -20       0      0      0 I   2.3   0.0   0:04.81 kworker/4+
13093 lee       20   0  508924  36068  28092 S   1.7   0.2   0:00.55 gnome-scr+
   68 root      25   5       0      0      0 S   1.0   0.0   0:37.74 ksmd
```

Figure 2.19: **top**

As we are interested specifically in the **stress** service and its child processes, we provide a script for monitoring the service processes running.

track-STRESS.sh looks for a **stress** process with a PPID of 1, then all of the related child processes. Once the processes are located some data is extracted with the **ps** command.

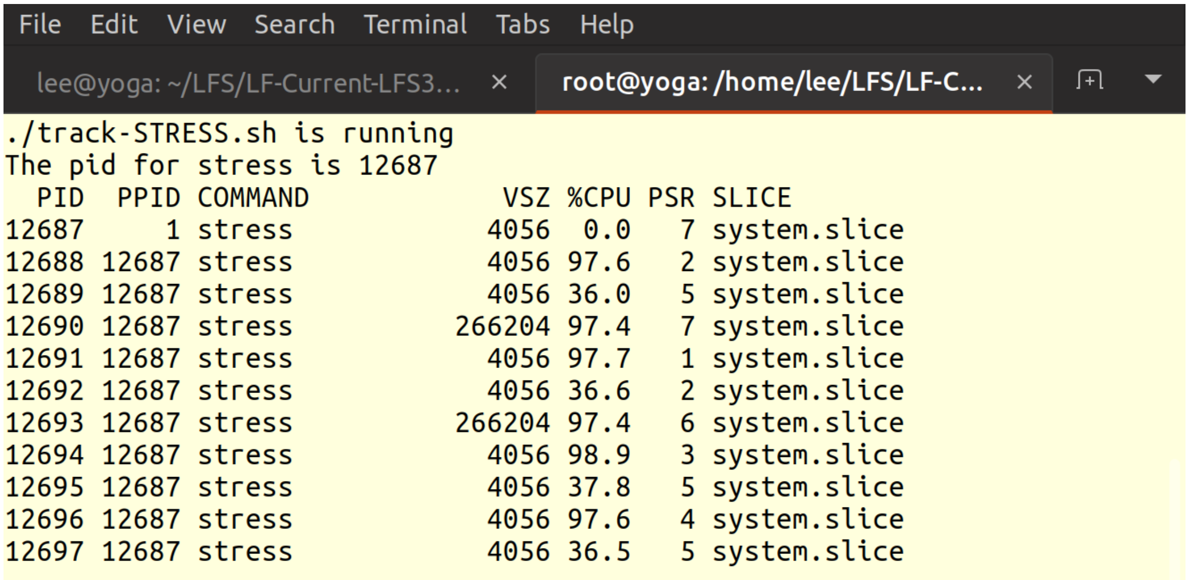
A copy of this script (`track-STRESS.sh`) can be found in the tarball in the `LFS211/SOLUTIONS/s_02/` directory.

SH track-STRESS.sh

```
#!/bin/bash
#
# This little script is used with the LFS311 lab exercise
# on systemd startup files and their affect on a background
# service. The "foo" service.
#
# The script looks for "stress" launched with a PPID of 1
# and it's chidren processes then uses the "ps" command
# to collect some information.

while [ true ]
do

# reset the variables to make sure we are picking up
# current information
PID=""
PID1=""
clear

echo "$0 is running"

# sift and sort the pid's
PID1=` ps -ef  | grep stress | grep -v grep | awk '{print $1,$2,$3,$6}'`
PID=`echo $PID1 |  grep "1 "| awk '{print $2}' `
echo "The pid for stress is $PID"
```

```
# using our list of pid's, grab some information
ps  --ppid $PID  --pid $PID -o pid,ppid,comm,vsz,pcpu,psr,slice 2>/dev/null

sleep 5
done
exit
```

An example of the script running:



File Edit View Search Terminal Tabs Help
lee@yoga: ~/LFS/LF-Current-LFS3...
root@yoga: /home/lee/LFS/LF-C...

```
./track-STRESS.sh is running
The pid for stress is 12687
  PID  PPID COMMAND            VSZ %CPU PSR SLICE
12687     1 stress            4056  0.0   7 system.slice
12688 12687 stress            4056 97.6   2 system.slice
12689 12687 stress            4056 36.0   5 system.slice
12690 12687 stress          266204 97.4   7 system.slice
12691 12687 stress            4056 97.7   1 system.slice
12692 12687 stress            4056 36.6   2 system.slice
12693 12687 stress          266204 97.4   6 system.slice
12694 12687 stress            4056 98.9   3 system.slice
12695 12687 stress            4056 37.8   5 system.slice
12696 12687 stress            4056 97.6   4 system.slice
12697 12687 stress            4056 36.5   5 system.slice
```

Figure 2.20: **track-STRESS.sh**

Since /usr/lib/systemd/system/foo.service is the default configuration supplied by the packager of the service and may be altered by the vendor at any time, create a custom unit file in /etc/systemd/system/foo.service for the **stress** service. This file is not usually overwritten by the vendor so local customizations can go here. Change the parameters slightly for the **foo** service using this directory. It is common practice to copy the vendor unit file into the /etc/systemd/system/ directory and make appropriate customizations.

/etc/systemd/system/foo.service

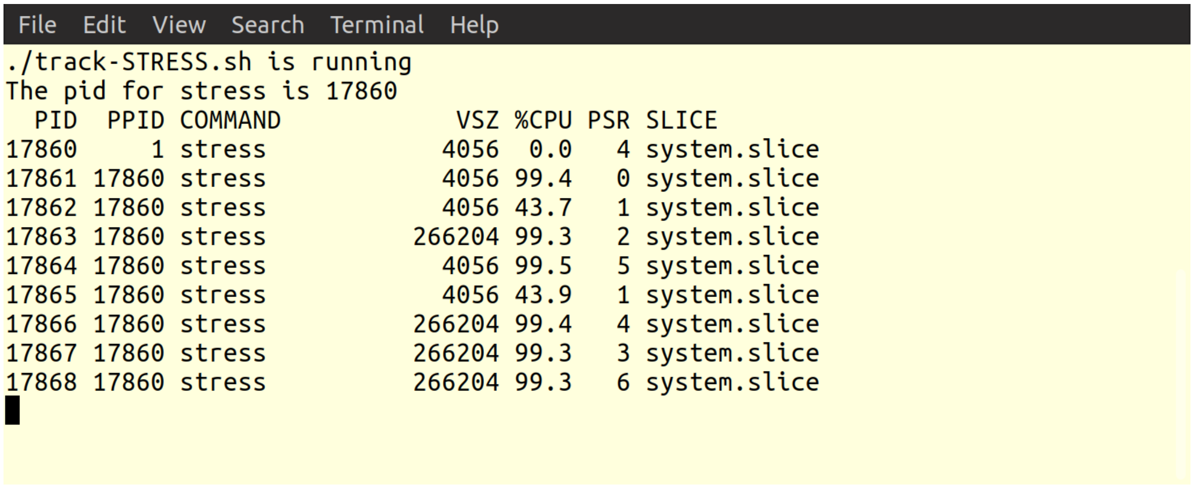
```
[Unit]
Description=Example service unit to run stress
[Service]
ExecStart=/usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
[Install]
WantedBy=multi-user.target
```

A copy of this file (foo2.service) can be found in the tarball in the LFS211/SOLUTIONS/s_02/ directory.

Start or restart the service and examine the differences in the following commands output.

```
# track-STRESS.sh
# systemctl status foo -l
# systemd-delta
```

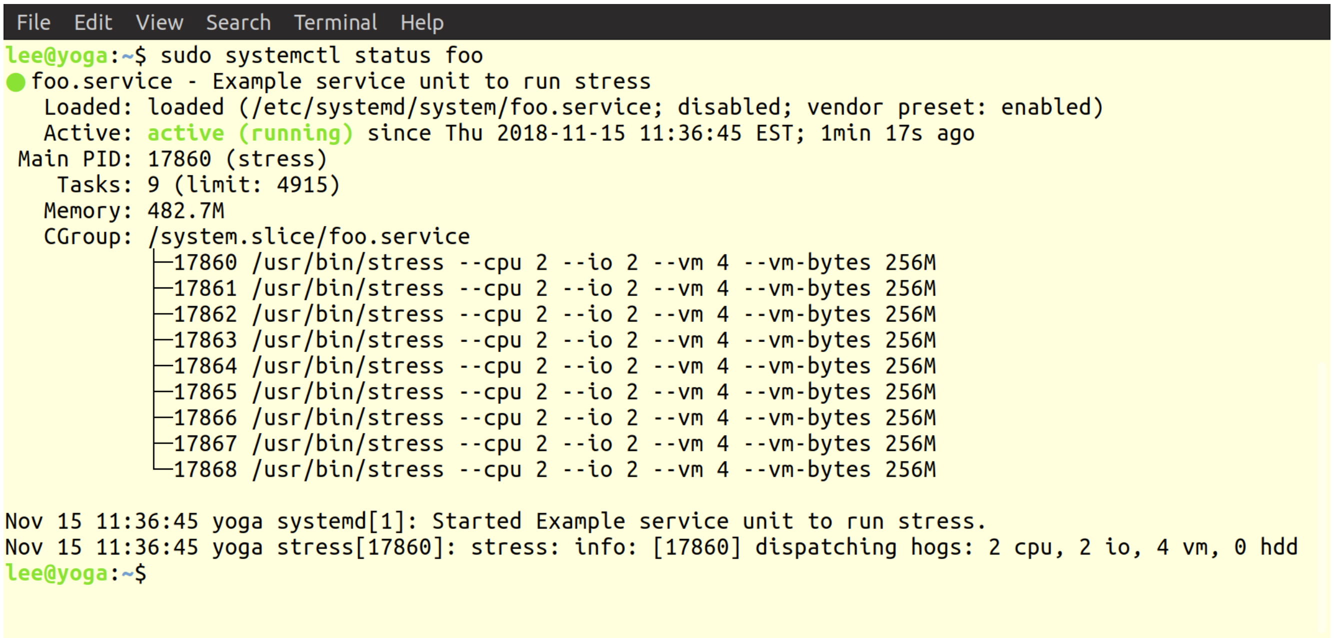
The changes to the configuration file can be seen with the **track-STRESS.sh** script, notice the number of memory hogs is now 4 and the CPU hogs is reduced to 2.



```
./track-STRESS.sh is running
The pid for stress is 17860
  PID  PPID COMMAND              VSZ %CPU PSR SLICE
17860     1 stress              4056  0.0   4 system.slice
17861 17860 stress              4056 99.4   0 system.slice
17862 17860 stress              4056 43.7   1 system.slice
17863 17860 stress            266204 99.3   2 system.slice
17864 17860 stress              4056 99.5   5 system.slice
17865 17860 stress              4056 43.9   1 system.slice
17866 17860 stress            266204 99.4   4 system.slice
17867 17860 stress            266204 99.3   3 system.slice
17868 17860 stress            266204 99.3   6 system.slice
```

Figure 2.21: **track-STRESS.sh with new unit file**

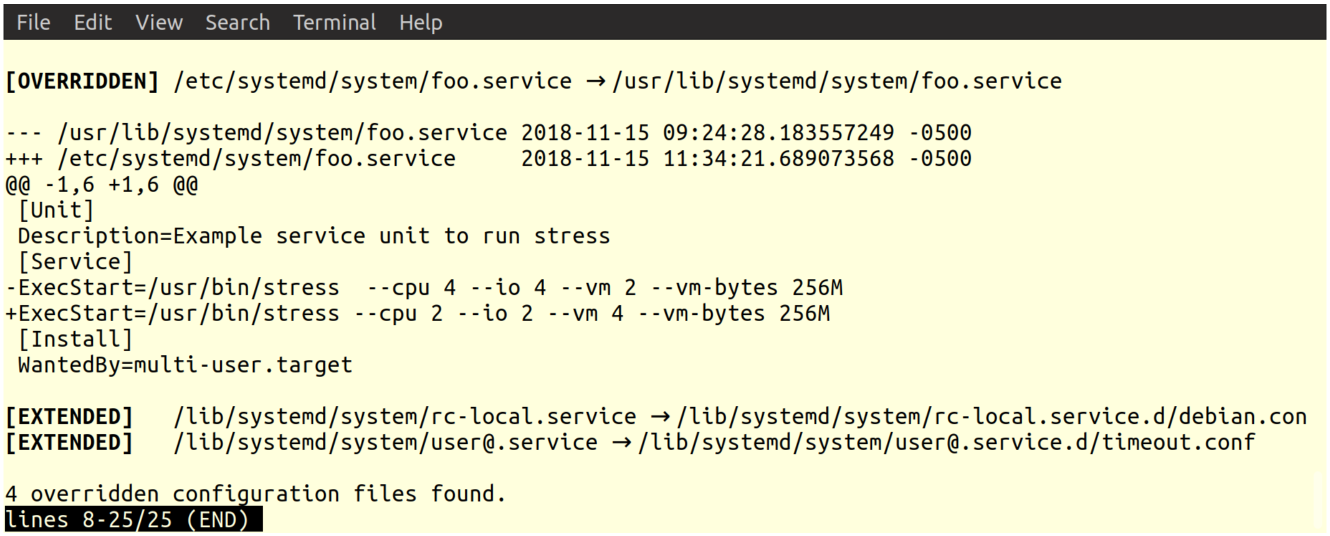
Which configuration (or **unit** file) file is active is not clear in the script. Use `systemctl status foo` to see which unit file is being used. This will show which configuration files are being used but not the differences in the files.



```
lee@yoga:~$ sudo systemctl status foo
● foo.service - Example service unit to run stress
   Loaded: loaded (/etc/systemd/system/foo.service; disabled; vendor preset: enabled)
   Active: active (running) since Thu 2018-11-15 11:36:45 EST; 1min 17s ago
 Main PID: 17860 (stress)
    Tasks: 9 (limit: 4915)
   Memory: 482.7M
   CGroup: /system.slice/foo.service
           ├─17860 /usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
           ├─17861 /usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
           ├─17862 /usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
           ├─17863 /usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
           ├─17864 /usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
           ├─17865 /usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
           ├─17866 /usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
           ├─17867 /usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
           └─17868 /usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M

Nov 15 11:36:45 yoga systemd[1]: Started Example service unit to run stress.
Nov 15 11:36:45 yoga stress[17860]: stress: info: [17860] dispatching hogs: 2 cpu, 2 io, 4 vm, 0 hdd
lee@yoga:~$
```

Figure 2.22: **systemctl status foo changing the unit file**

To see the details of the unit file changes the **systemd-delta** command can be used. The output has the changes in **diff** format, making it easy to see what has changed. See the example below:



```
[OVERRIDDEN] /etc/systemd/system/foo.service → /usr/lib/systemd/system/foo.service

--- /usr/lib/systemd/system/foo.service 2018-11-15 09:24:28.183557249 -0500
+++ /etc/systemd/system/foo.service     2018-11-15 11:34:21.689073568 -0500
@@ -1,6 +1,6 @@
 [Unit]
 Description=Example service unit to run stress
 [Service]
-ExecStart=/usr/bin/stress  --cpu 4 --io 4 --vm 2 --vm-bytes 256M
+ExecStart=/usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
 [Install]
 WantedBy=multi-user.target

[EXTENDED]   /lib/systemd/system/rc-local.service → /lib/systemd/system/rc-local.service.d/debian.con
[EXTENDED]   /lib/systemd/system/user@.service → /lib/systemd/system/user@.service.d/timeout.conf

4 overridden configuration files found.
lines 8-25/25 (END)
```

Figure 2.23: **systemd-delta showing unit file override**

Often times it is desirable to add or change features by program or script control, the drop-in files are convenient for this. One item of caution, if one is changing a previously defined function (like ExecStart) it must be undefined first then added back in. Create a drop-in directory and file for out **stress** service and verify the changes are active. Our example file for foo.service using a drop-in directory (00-foo.conf) can be found in the SOLUTIONS tarball in the LFS211/SOLUTIONS/s_02/ directory and contains:

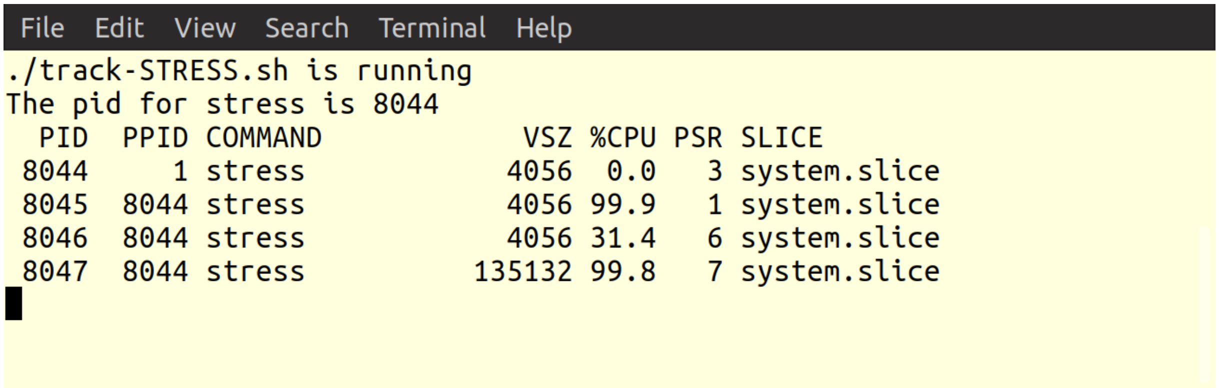
/etc/systemd/system/foo.service.d/00-foo.conf

```
[Service]
ExecStart=
ExecStart=/usr/bin/stress --cpu 1 --vm 1 --io 1 --vm-bytes 128M
```

Start or restart the service and examine the differences in the output of the following commands.

```
# track-STRESS.sh
# systemctl status foo -l
# systemd-delta
```

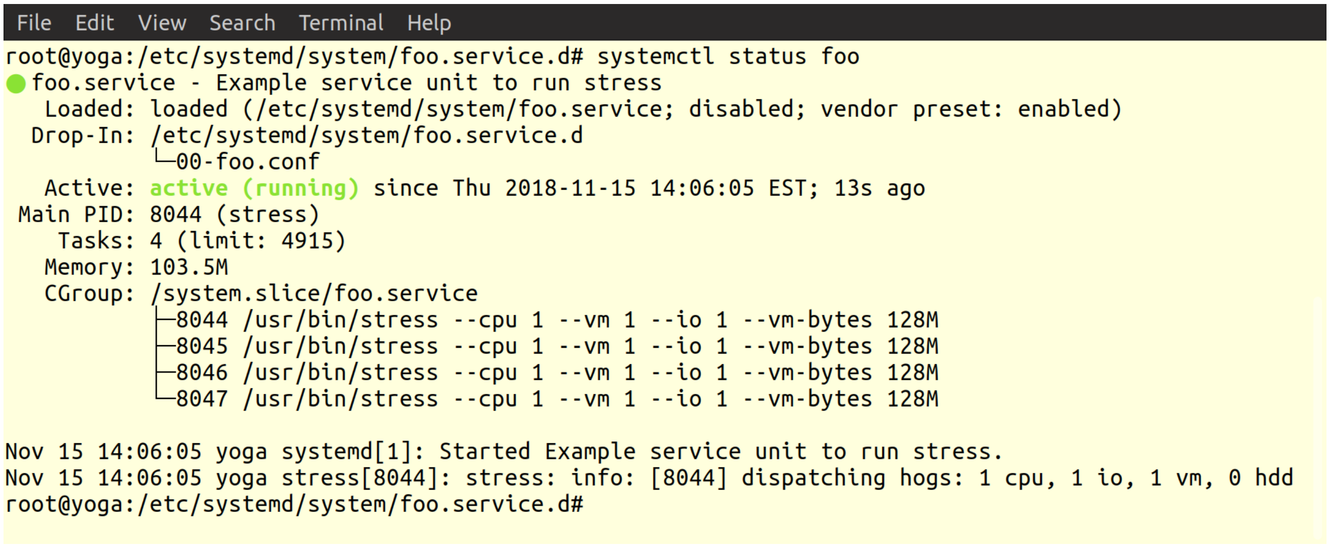
The information in the drop in file over writes the unit file. In this example the number of “hogs” has been greatly reduced.



```
./track-STRESS.sh is running
The pid for stress is 8044
  PID  PPID COMMAND            VSZ %CPU PSR SLICE
 8044     1 stress            4056  0.0   3 system.slice
 8045  8044 stress            4056 99.9   1 system.slice
 8046  8044 stress            4056 31.4   6 system.slice
 8047  8044 stress          135132 99.8   7 system.slice
```

Figure 2.24: **track-STRESS.sh using dropin file**

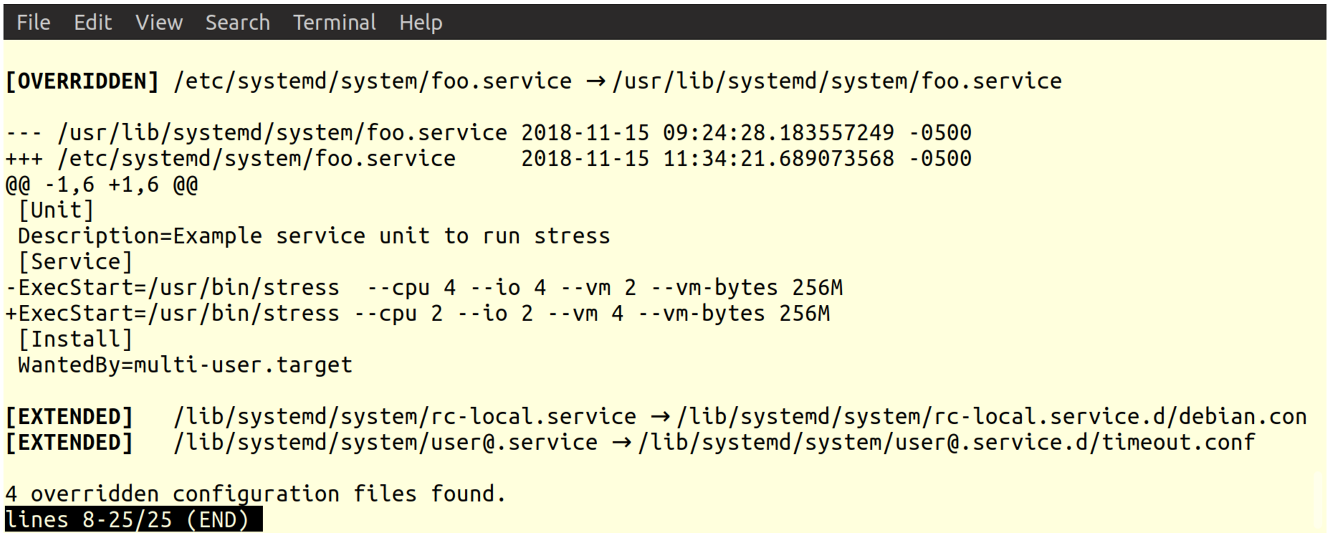
systemctl status shows the dropin file is active. If there was several dropin files it would show the order that were applied to the service.



```
root@yoga:/etc/systemd/system/foo.service.d# systemctl status foo
● foo.service - Example service unit to run stress
   Loaded: loaded (/etc/systemd/system/foo.service; disabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/foo.service.d
           └─00-foo.conf
   Active: active (running) since Thu 2018-11-15 14:06:05 EST; 13s ago
 Main PID: 8044 (stress)
    Tasks: 4 (limit: 4915)
   Memory: 103.5M
   CGroup: /system.slice/foo.service
           ├─8044 /usr/bin/stress --cpu 1 --vm 1 --io 1 --vm-bytes 128M
           ├─8045 /usr/bin/stress --cpu 1 --vm 1 --io 1 --vm-bytes 128M
           ├─8046 /usr/bin/stress --cpu 1 --vm 1 --io 1 --vm-bytes 128M
           └─8047 /usr/bin/stress --cpu 1 --vm 1 --io 1 --vm-bytes 128M

Nov 15 14:06:05 yoga systemd[1]: Started Example service unit to run stress.
Nov 15 14:06:05 yoga stress[8044]: stress: info: [8044] dispatching hogs: 1 cpu, 1 io, 1 vm, 0 hdd
root@yoga:/etc/systemd/system/foo.service.d#
```

Figure 2.25: **systemctl status showing unit file override and dropin file**

Like the other commands, **systemd-delta** shows the files used by the service. In addition to the files used, the details of the changes in the unit file are displayed. Notice the changes to the service made with the drop-in file are not displayed, only the file name.



```
[OVERRIDDEN] /etc/systemd/system/foo.service → /usr/lib/systemd/system/foo.service

--- /usr/lib/systemd/system/foo.service 2018-11-15 09:24:28.183557249 -0500
+++ /etc/systemd/system/foo.service     2018-11-15 11:34:21.689073568 -0500
@@ -1,6 +1,6 @@
 [Unit]
 Description=Example service unit to run stress
 [Service]
-ExecStart=/usr/bin/stress  --cpu 4 --io 4 --vm 2 --vm-bytes 256M
+ExecStart=/usr/bin/stress --cpu 2 --io 2 --vm 4 --vm-bytes 256M
 [Install]
 WantedBy=multi-user.target

[EXTENDED]   /lib/systemd/system/rc-local.service → /lib/systemd/system/rc-local.service.d/debian.con
[EXTENDED]   /lib/systemd/system/user@.service → /lib/systemd/system/user@.service.d/timeout.conf

4 overridden configuration files found.
lines 8-25/25 (END)
```

Figure 2.26: **systemd-delta showing unit file and dropin file**

With **systemd**, additional features and capabilities can be easily added. As an example, **cgroups** controls can be added to our service. Here is an example of adding a **systemd slice** to the example service and adding a resource limit to that slice. The slice is then attached to the service drop-in file. First setup a `<service>.slice` unit file:

/etc/systemd/system/foo.slice

```
[Unit]
Description=stress slice
[Slice]
CPUQuota=30%
```

A copy of this file (foo.slice) can be found in the tarball in the LFS211/SOLUTIONS/s_02/ directory.

Then connect our service to the **slice**. Add the following to the bottom of the unit file in /etc/systemd/system/foo.service.d/00-foo.conf

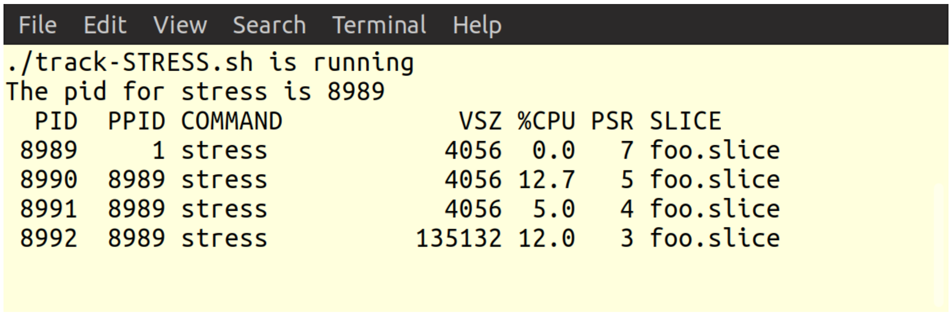
connect slice file to service file

```
Slice=foo.slice
```

Restart the services and examine the differences with:

```
# systemctl stop foo
# systemctl daemon-reload
# systemctl start foo
# systemctl status foo -l
# systemd-delta
# top
# track-STRESS.sh
```

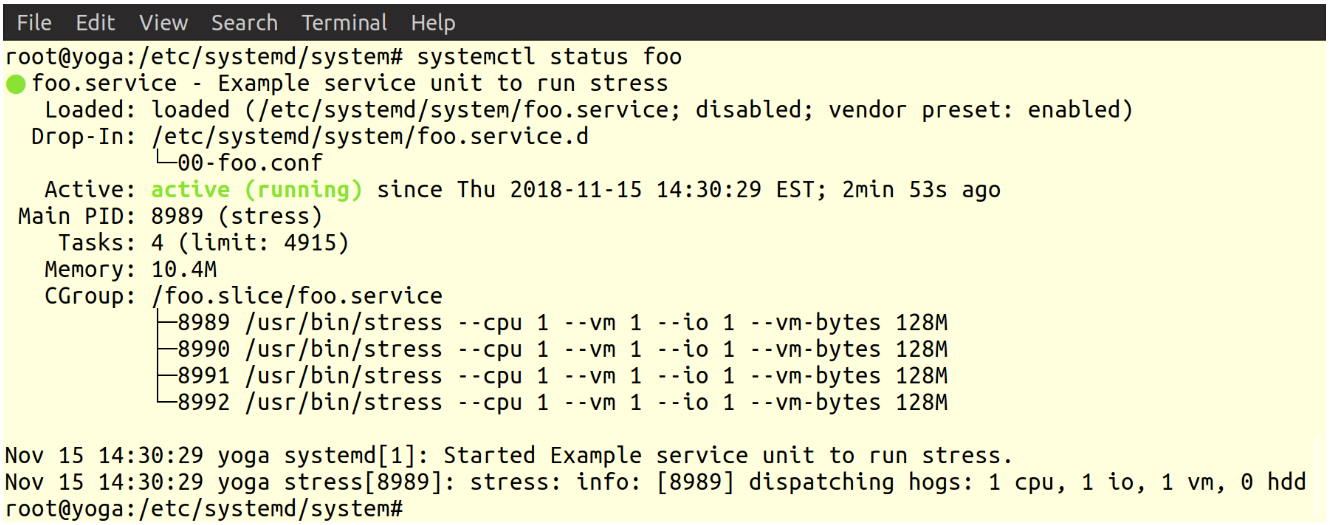
The cgroup information in the **slice** has been applied to the service. Notice the amount of CPU resource consumed. The total is 30% of one processor but it may be spread across multiple CPU's.



```
./track-STRESS.sh is running
The pid for stress is 8989
  PID  PPID COMMAND            VSZ %CPU PSR SLICE
 8989     1 stress            4056  0.0   7 foo.slice
 8990  8989 stress            4056 12.7   5 foo.slice
 8991  8989 stress            4056  5.0   4 foo.slice
 8992  8989 stress          135132 12.0   3 foo.slice
```

Figure 2.27: **track-STRESS.sh using slice attribute**

The **systemctl status** command shows the change in **CGroup** with the **slice** attribute active.



```
root@yoga:/etc/systemd/system# systemctl status foo
● foo.service - Example service unit to run stress
   Loaded: loaded (/etc/systemd/system/foo.service; disabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/foo.service.d
           └─00-foo.conf
   Active: active (running) since Thu 2018-11-15 14:30:29 EST; 2min 53s ago
 Main PID: 8989 (stress)
    Tasks: 4 (limit: 4915)
   Memory: 10.4M
   CGroup: /foo.slice/foo.service
           ├─8989 /usr/bin/stress --cpu 1 --vm 1 --io 1 --vm-bytes 128M
           ├─8990 /usr/bin/stress --cpu 1 --vm 1 --io 1 --vm-bytes 128M
           ├─8991 /usr/bin/stress --cpu 1 --vm 1 --io 1 --vm-bytes 128M
           └─8992 /usr/bin/stress --cpu 1 --vm 1 --io 1 --vm-bytes 128M

Nov 15 14:30:29 yoga systemd[1]: Started Example service unit to run stress.
Nov 15 14:30:29 yoga stress[8989]: stress: info: [8989] dispatching hogs: 1 cpu, 1 io, 1 vm, 0 hdd
root@yoga:/etc/systemd/system#
```

Figure 2.28: **systemctl status showing CGroup and slice**

Bonus step: In our example there are no unique values in the /etc/systemd/system/foo.service file so in this example it is redundant. We can get rid of the extra file.

```
# mv /etc/systemd/system/foo.service /root/
# systemctl daemon-reload
# systemctl restart foo
# systemctl status foo
```

Consult the man pages **systemd.resource-control(5), systemd.service(5), systemd-delta(1)** and other **systemd** man pages for additional information.