Using stress (Lab)

coursera.org/learn/linux-for-developers/supplement/E9z9Q/using-stress-lab

Exercise

From time to time, it will be useful to us to stress the system by making the CPU labor, wasting memory, or kicking up I/O activity. The appropriately named **stress** utility is a C language program written by Amos Waterland at the University of Oklahoma under the GPL v2. It is designed to place a configurable amount of stress by generating various kind of workloads on any kind of POSIX system.

A newer enhanced version of **stress** (called **stress-ng**) is also available and has an almost infinite number of options, including over 105 stress tests.

All major distributions now have both of these programs in their repositories, so you should be able to do one of the following:

```
1
2
3
4
$ sudo yum install stress stress-ng
$ sudo dnf install stress stress-ng
$ sudo apt-get install stress stress-ng
$ sudo zypper install stress stress-ng
```

stress-ng is backwards compatible with **stress**, so wherever we use **stress** from here on, you can substitute **stress-ng**:

```
1
$ stress-ng --help
```

for a quick list of options.

As an example, the command:

```
1

$ stress -c 8 -i 4 -m 6 -t 20s
```

will:

- Fork off 8 CPU-intensive processes, each spinning on an **sqrt()** calculation.
- Fork off 4 I/O-intensive processes, each spinning on **sync()**.
- Fork off 6 memory-intensive processes, each spinning on **malloc()**, allocating 256 MB by default. The size can be changed as in **--vm-bytes 128M**.
- Run the stress test for 20 seconds.

After installing **stress**, start up your system's graphical system monitor, which you can find on your application menu, or run from the command line, which is probably **gnome-system-monitor** or **ksysguard**. Later, we will consider these in some detail.

Now, begin to put stress on the system. The exact numbers you use will depend on your system's resources, such as the number of CPUs and RAM size.

For example, doing:

```
1

$ stress -m 4 -t 20s
```

which puts only a memory stressor on the system, is likely to take up all your CPU time.

Play with combinations of the switches and see how they impact each other. In succeeding sessions, we will be able to use the **stress** program to simulate various high load conditions.