StressLinux

http://www.stresslinux.org/sl/

Stresslinux is a lightweight Linux distribution designed to test a computer's hardware by running the components at high load while monitoring their health.

It makes use of some utitlities such as stress, cpuburn, hddtemp, lm_sensors, etc. It is dedicated to users who want to test their system(s) entirely on high load and monitor the health of these systems.

StressLinux runs from external bootable media: CD, USB stick, PXE boot, or you can run the VMWare image. There are good instructions for creating your chosen boot medium.

When you boot up, you have the option to allow sl-wizard to probe your system for sensors and then load the appropriate drivers

You can re-run this anytime after boot by deleting /tmp/sensors and then running the sl-wizard.sh script as root, like this:

```
stress@stresslinux:~>;sudo rm /tmp/sensors
stress@stresslinux:~>;sudo sl-wizard.sh
```

This is not an ordinary stripped-down Linux. It was built with SUSE Studio, and is based on OpenSUSE.

StressLinux uses Busybox in place of the usual coreutils, fileutils, and other standard Linux commands. Busybox is a single stripped-down binary containing several dozen commands, and it uses the ash shell, so you may find that some of your favorite options are missing. The Busybox command reference should help you.

Stresslinux uses the Fn keys in an interesting way. There are 6 ordinary `ttys` on F1-F6, and it boots to `tty1` on F1. The rest are normal login `ttys`. On US keyboards you can switch between these with ALT+Fn. (STRG+Fn on German keyboards, which is the same as CTRL+Fn.) F10 displays eth0 throughput (see above), F11 shows hard disk temperatures, and F12 displays `lm-sensors` readings.

What to Do : The Main Task:

Now that Stresslinux is booted up and you have gazed upon your eth0 and sensor outputs, what's next? Let's spend some time with the `stress` command, because that is a good general-purpose workload generator. It operates by siccing a bunch of hogs on your system. This simple invocation puts a light load on the CPU, I/O, memory, and hard drive:

```
stress --cpu 1000 --io 4 --vm 2 --hdd 4 --timeout 15m --verbose
```

When it finishes with "successful run completed" that means there were no errors. If it did detect errors, it would either try to tell you what they were, or tell you to examine the syslog. Let's walk through this so we know what it's doing.

`--cpu 1000` tells it to fork 1000 processes. Each process calculates the square root of a random number (by calling the `sqrt()` and `rand` functions) in a loop that stops at the end of your timeout, or when you stop it with CTRL+c. This is similar, a way to keep the CPU constantly busy. 1000 processes surely put some load !!!

`--io 4` forks 4 processes that call the `sync()` function in a loop. `sync()` flushes any data buffered in memory to disk. Most Linux filesystems use delayed allocation; that is, data are held in memory for a period of time before being written to disk. This speeds up performance because disk I/O is slower than RAM. Running this one by itself and trying out different values will give you an idea of your I/O performance.

`--vm 2` thrashes your RAM by forking 2 processes to allocate and release memory. (Looping `malloc()` and `free()`.) You can control the load on your memory with the `--vm-bytes` option.

```
stress --vm 2 --vm-bytes 3G --vm-hang --timeout 60s
```

Be careful with this because thrashing your memory can make your system hang. The `vm` option allocates and releases memory; the `vm-hang` simulates low memory conditions by having each hog process go to sleep, rather than releasing memory. This above example hogs 3 gigabytes of memory:

`--hdd 4` pummels your hard drive with writes, by calling the `write` function in a loop. The default file size is 1GB, and you can specify any size with the `--hdd-bytes` option, for example `--hdd-bytes 5G` writes a 5 gigabyte file.

```
stress --hdd 1 --hdd-bytes 5G --timeout 5m
```

`--timeout 30s` tells `stress` to stop after 30 seconds. Or whatever time you want, of course, using s,m,h,d,y (seconds, minutes, hours, days, years). Always set a timeout, because this is your protection from the system locking up and becoming inaccessible. `stress` runs in userspace and can't cause any damage, but it would be sad to have to reboot to stop it.

`--verbose` makes it spit out many lines telling what it is doing.

`stress` is tidy and cleans up after itself, so you shouldn't have to worry about leftover hogs running amok. The documentation on `stress` isn't exactly lavish, and the most complete help is in `info stress`.