Exercise 11.1: Using stress

stress is a **C** language program written by Amos Waterland at the University of Oklahoma, licensed under the **GPL v2**. It is designed to place a configurable amount of stress by generating various kinds of workloads on the system.

If you are lucky you can install **stress** directly from your distribution's packaging system. Otherwise, you can obtain the source from `http://people.seas.harvard.edu/~apw/stress`, and then compile and install by doing:

```
$ tar zxvf stress-1.0.4.tar.gz
$ cd stress-1.0.4
$ ./configure
$ make
$ sudo make install
```

There may exist pre-packaged downloadable binaries in the **.deb** and **.rpm** formats; see the home page for details and locations.

Once installed, you can do:

```
$ stress --help
```

for a quick list of options, or

```
$ info stress
```

for more detailed documentation.

As an example, the command:

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

will:

- Fork off 8 CPU-intensive processes, each spinning on a `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**, you may want to 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**.

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 CPU's and **RAM** size.

For example, doing

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

puts only a memory stressor on the system.

Play with combinations of the switches and see how they impact each other. You may find the **stress** program useful to simulate various high load conditions.

