NAME

gkrellm - The GNU Krell Monitors

SYNOPSIS

gkrellm [**--help**] [**-t | --theme** dir] [**-g | --geometry** +x+y] [**-wm**] [**-w | --withdrawn**] [**-c | --config** suffix] [**-nc**] [**-f | --force-host-config**] [**-demo**] [**-p | --plugin** plugin.so] [**-s | --server** hostname] [**-P | --port** server_port] [**-l | --logfile** path]

DESCRIPTION

With a single process, **gkrellm** manages multiple stacked monitors and supports applying themes to match the monitors appearance to your window manager, Gtk, or any other theme.

FEATURES

- SMP CPU, Disk, Proc, and active net interface monitors with LEDs.
- Internet monitor that displays current and charts historical port hits.
- Memory and swap space usage meters and a system uptime monitor.
- File system meters show capacity/free space and can mount/umount.
- A mbox/maildir/MH/POP3/IMAP mail monitor which can launch a mail reader or remote mail fetch program.
- Clock/calendar and hostname display.
- Laptop Battery monitor.
- CPU/motherboard temperature/fan/voltages display with warnings and alarms. Linux requires a sensor configured sysfs, lm_sensors modules or a running mbmon daemon. FreeBSD can also read the mbmon daemon. Windows requires MBM.
- Disk temperatures if there's a running hddtemp daemon.
- Multiple monitors managed by a single process to reduce system load.
- A timer button that can execute PPP or ISDN logon/logoff scripts.
- Charts are autoscaling with configurable grid line resolution, or
- can be set to a fixed scale mode.
- Separate colors for in and out data. The in color is used for CPU user time, disk read, forks, and net receive data. The out color is used for CPU sys time, disk write, load, and net transmit data.
- Commands can be configured to run when monitor labels are clicked.
- Data can be collected from a **gkrellmd** server running on a remote machine.
- **gkrellm** is plugin capable so special interest monitors can be created.
- Many themes are available.

USER INTERFACE

- **Top frame**

  Btn 1 Press and drag to move **gkrellm** window.

  Btn 3 Popup main menu.

- **Side frames**

  Btn 2 Slide **gkrellm** window shut (Btn1 if -m2 option).

  Btn 3 Popup main menu.

- **All charts**

  Btn 1 Toggle draw of extra info on the chart.

  Btn 3 Brings up a chart configuration window.

- **Inet charts**

  Btn 2 Toggle between port hits per minute and hour.

- **Most panels**

  Btn 3 Opens the configuration window directly to a monitor's configuration page.

- **File System meter panels**

  Btn 1,2
  Toggle display of label and fs capacity scrolling display. The mount button runs mount/umount commands. If ejectable, left click the eject button to open tray, right click to close.

- **Mem and Swap meter panels**

  Btn 1,2
  Toggle display of label and memory or swap capacity scrolling display.

- **Mailbox monitor message count button**

  Btn 1 Launch a mail reader program. If options permit, also stop animations and reset remote message counts.

  Btn 2 Toggle mail check mute mode which inhibits the sound notify program, and optionally inhibits all mail checking.

- **Mailbox monitor envelope decal**

  Btn 1 Force a mail check regardless of mute or timeout state.

- **Battery monitor panel**

  Btn 1 On the charging state decal toggles battery minutes left, percent level, and charge rate display.

  Btn 2 Anywhere on the panel also toggles the display.

- **Keyboard shortcuts**

  F1 popup the user config window.

  F2 popup the main menu.

  Page_Up
  previous theme or theme alternative.

  Page_Down
  next theme or theme alternative.

  <Ctl>Page_Up
  previous theme, skipping any theme alternatives.

  <Ctl>Page_Down
  next theme, skipping any theme alternatives.

If a command has been configured to be launched for a monitor, then a button will appear when the mouse enters the panel of that monitor. Clicking the button will launch the command.

A right button mouse click on the side or top frames of the **gkrellm** window will pop up a user configuration window where you can configure all the builtin and plugin monitors. Chart appearance may be configured by right clicking on a chart, and right clicking on many panels will open the configuration window directly to the corresponding monitor's configuration page.

OPTIONS

--help Displays this manual page.

-t, --theme dir
gkrellm will load all theme image files it finds in *dir* and parse the *gkrellmrc* file if one exists. This option overrides the loading of the last theme you configured to be loaded in the Themes configuration window. Theme changes are not saved when **gkrellm** is run with this option.

-g, --geometry +x+y
Makes **gkrellm** move to an *(x,y)* position on the screen at startup. Standard X window geometry position (not size) formats are parsed, ie +x+y -x+y +x-y -x-y. Except, negative geometry positions are not recognized (ie +-x--y).

-wm Forces **gkrellm** to start up with window manager decorations. The default is no decorations because there are themed borders.

-w, --withdrawn
gkrellm starts up in withdrawn mode so it can go into the Blackbox slit (and maybe WindowMaker dock).

-c, --config suffix
Use alternate config files generated by appending *suffix* to config file names. This overrides any previous host config which may have been setup with the below option.

-f, --force-host-config
If **gkrellm** is run once with this option and then the configuration or theme is changed, the config files that are written will have a *-hostname* appended to them. Subsequent runs will detect the *user-config-hostname* and *gkrellm_theme.cfg-hostname* files and use them instead of the normal configuration files (unless the **--config** option is specified). This is a convenience for allowing remote **gkrellm** independent config files in a shared home directory, and for the hostname to show up in the X title for window management. This option has no effect in client mode.

-s, --server hostname
Run in client mode by connecting to and collecting data from a **gkrellmd** server on *hostname*

-P, --port server_port
Use *server_port* for the **gkrellmd** server connection.

-l, --logfile path
Enable sending error and debugging messages to a log file.

-nc No config mode. The config menu is blocked so no config changes can be made. Useful in certain environments, or maybe for running on a **xdm(1)** login screen or during a screensaver mode?

-demo Force enabling of many monitors so themers can see everything. All config saving is inhibited.

-p, --plugin plugin.so
For plugin development, load the command line specified plugin so you can avoid repeated install steps in the development cycle.

BUILTIN MONITORS

Charts

The default for most charts is to automatically adjust the number of grid lines drawn and the resolution per grid so drawn data will be nicely visible. You may change this to fixed grids of 1-5 and/or fixed grid resolutions in the chart configuration windows. However, some combination of the auto scaling modes may give best results.

Auto grid resolution has the following behavior.

Auto mode sticks at peak value is not set:

1) If using auto number of grids, set the resolution per grid and the number of grids to optimize the visibility of data drawn on the chart. Try to keep the number of grids between 1 and 7.

2) If using a fixed number of grids, set the resolution per grid to the smallest value that draws data without clipping.

Auto mode sticks at peak value is set:

1) If using auto number of grids, set the resolution per grid such that drawing the peak value encountered would require at least 5 grids.

2) If using a fixed number of grids, set the resolution per grid such that the peak value encountered could be drawn without clipping. This means the resolution per grid never decreases.

All resolution per grid values are constrained to a set of values in either a 1, 2, 5 sequence or a 1, 1.5, 2, 3, 5, 7 sequence. If you set **Auto mode sticks at peak value** a manual **Auto mode recalibrate** may occasionally be required if the chart data has a wide dynamic range.

CPU Monitor

Data is plotted as a percentage. In auto number of grids mode, resolution is a fixed 20% per grid. In fixed number of grids mode, grid resolution is 100% divided by the number of grids.

Proc Monitor

The krell shows process forks with a full scale value of 10 forks. The chart has a resolution of 10 forks/sec per grid in auto number of grids mode and 50 forks/second maximum on the chart in fixed number of grids mode. The process load resolution per grid is best left at 1.0 for auto number of grids, but can be set as high as 5 if you configure the chart to have only 1 or 2 fixed grids.

Net Monitor

gkrellm is designed to display a chart for net interfaces which are up, which means they are listed in the routing table (however, it is possible in some cases to monitor unrouted interfaces). One net interface may be linked to a timer button which can be used to connect and disconnect from an ISP.

The timer button shows an off, standby, or on state by a distinctive (color or shape) icon.

ppp Standby state is while the modem phone line is locked while ppp is connecting, and the on state is the ppp link connected. The phone line lock is determined by the existence of the modem lock file */var/lock/LCK..modem,* which assumes pppd is using */dev/modem.* However, if your pppd setup does not use */dev/modem,* then you can configure an alternative with:

ln -s /var/lock/LCK..ttySx ~/.gkrellm2/LCK..modem

where ttySx is the tty device your modem does use. The ppp on state is detected by the existence of */var/run/pppX.pid* and the time stamp of this file is the base for the on line time.

ippp The timer button standby state is not applicable to ISDN interfaces that are always routed. The on state is ISDN on line while the ippp interface is routed. The on line timer is reset at transitions from ISDN hangup state to on line state.

For both ppp and ippp timer button links, the panel area of the interface is always shown and the chart appears when the interface is routed with the phone link connected or on line.

If the timer button is not linked to a net interface, then it can be used as a push on / push off timer

Net monitors can have a label so that the interface can be associated with the identity of the other end of the connection. This is useful if you have several net connections or run multiple remote **gkrellm** programs. It can be easier to keep track of who is connected to who.

Mem and Swap Monitor

Here you are reading a ratio of total used to total available. The amount of memory used indicated by the memory monitor is actually a calculated used memory. If you enter the free command, you will see that most of your memory is almost always used because the kernel uses large amounts for buffers and cache. Since the kernel can free a lot of this memory as user process demand for memory goes up, a more realistic reading of memory in use is obtained by subtracting the buffers and cached memory from the kernel reported used. This is shown in the free command output in the -/+ buffers/cache line where a calculated used amount has buffers and cached memory subtracted from the kernel reported used memory, and a calculated free amount has the buffers and cached memory added in.

While the memory meter always shows the calculated used memory, the raw memory values total, shared, buffered, and cached may be optionally displayed in the memory panel by entering an appropriate format display string in the config.

Units: All memory values have units of binary megabytes (MiB). Memory sizes have historically been reported in these units because memory arrays on silicon have always increased in size by multiples of 2. Add an address line to a memory chip and you double or quadruple (a multiplexed address) the memory size. A binary megabyte is 2^20 or 1048576. Contrast this with units for other stats such as disk capacities or net transfer rates where the proper units are decimal megabytes or kilobytes. Disk drive capacities do not increase by powers of 2 and manufacturers do not use binary units when reporting their sizes. However, some of you may prefer to see a binary disk drive capacity reported, so it is available as an option.

Internet Monitor

Displays TCP port connections and records historical port hits on a minute or hourly chart. Middle button click on an inet chart to toggle between the minute and hourly displays. There is a strip below the minute or hour charts where marks are drawn for port hits in second intervals. Each inet krell also shows port hits with a full scale range of 5 hits. The left button toggle of extra info displays current port connections.

For each internet monitor you can specify two labeled datasets with one or two ports for each dataset. There are two ports because some internet ports are related and you might want to group them - for example, the standard HTTP port is 80, but there is also a www web caching service on port 8080. So it makes sense to have a HTTP monitor which combines data from both ports. A possible common configuration would be to create one inet monitor that monitors HTTP hits plotted in one color and FTP hits in another. To do this, setup in the Internet configuration tab:

HTTP 80 8080 FTP 21

Or you could create separate monitors for HTTP and FTP. Other monitors might be SMTP on port 25 or NNTP on port 119.

If you check the Port0 - Port1 is a range button, then all of the ports between the two entries will be monitored. Clicking the small button on the Inet panels will pop up a window listing the currently connected port numbers and the host that is connected to it.

gkrellm samples TCP port activity once per second, so it is possible for port hits lasting less than a second to be missed.

File System Monitor

File system mount points can be selected to be monitored with a meter that shows the ratio of blocks used to total blocks available. Mounting commands can be enabled for mount points in one of two ways:

If a mount point is in your */etc/fstab* and you have mount permission then mount(8) and umount(8) commands can be enabled and executed for that mount point simply by checking the Enable /etc/fstab mounting option. Mount table entries in */etc/fstab* must have the user or owner option set to grant this permission unless **gkrellm** is run as root. For example, if you run **gkrellm** as a normal user and you want to be able to mount your floppy, your */etc/fstab* could have either of:

```
/dev/fd0 /mnt/floppy ext2 user,noauto,rw,exec 0 0
/dev/fd0 /mnt/floppy ext2 user,defaults 0 0
```

If **gkrellm** is run as root or if you have **sudo(1)** permission to run the mount(8) commands, then a custom mount command can be entered into the mount command entry box. A umount(8) command must also be entered if you choose this method. Example mount and umount entries using sudo:

```
sudo /bin/mount -t msdos /dev/fd0 /mnt/A
sudo /bin/umount /mnt/A
```

Notes: the mount point specified in a custom mount command (/mnt/A in this example) must be the same as entered in the Mount Point entry. Also, you should have the NOPASSWD option set in */etc/sudoers* for this.

File system monitors can be created as primary (always visible) or secondary which can be hidden and then shown when they are of interest. For example, you might make primary file system monitors for root, home, or user so they will be always visible, but make secondary monitors for less frequently used mount points such as floppy, zip, backup partitions, foreign file system types, etc. Secondary FS monitors can also be configured to always be visible if they are mounted by checking the Show if mounted option. Using this feature you can show the secondary group, mount a file system, and have that FS monitor remain visible even when the secondary group is hidden. A standard cdrom mount will show as 100% full but a monitor for it could be created with mounting enabled just to have the mount/umount convenience.

When the Ejectable option is selected for a file system, an eject button will appear when the mouse enters the file system panel. If you are not using /etc/fstab mounting, a device file to eject will also need to be entered. Systems may have varying levels of support for this feature ranging from none or basic using an ioctl() to full support using an eject command to eject all its supported devices. Linux and NetBSD use the eject command while FreeBSD uses the cdcontrol command, so be sure these commands are installed. Most eject commands will also support closing a CDROM tray. If they do, you will be able to access this function by right clicking the eject button.

Mail Monitor

Checks your mailboxes for unread mail. A mail reading program (MUA) can be executed with a left mouse click on the mail monitor panel button, and a mail notify (play a sound) program such as esdplay or artsplay can be executed whenever the new mail count increases. The mail panel envelope decal may also be clicked to force an immediate mail check at any time.

gkrellm is capable of checking mail from local mailbox types mbox, MH, and maildir, and from remote mailbox types POP3 and IMAP.

POP3 and IMAP checking can use non-standard port numbers and password authentication protocols APOP (for POP3 only) or CRAM-MD5. If supported by the mail server, emote checking may be done over an SSL connection if the Use SSL option is selected.

Before internal POP3 and IMAP checking was added, an external mail fetch/check program could be set up to be executed periodically to download or check remote POP3 or IMAP mail. This method is still available and must be used if you want **gkrellm** to be able to download remote mail to local mailboxes because the builtin checking functions cannot download.

Battery Monitor

This meter will be available if a battery exists and will show battery percentage life remaining. A decal indicates if AC line is connected or if the battery is in use. If the data is available, time remaining may be displayed as well as the percentage battery level. If the time remaining is not available or is inaccurate, the Estimate Time option may be selected to display a battery time to run or time to charge which is calculated based on the current battery percent level, user supplied typical battery times, and a default linear extrapolation model. For charging, an exponential charge model may be selected.

A battery low level warning and alarm alert may be set. If battery time is not available from the OS and the estimate time mode is not set, the alert units will be battery percent level. Otherwise the alert units will be battery time left in minutes. If OS battery time is not available and the estimate time mode is set when the alert is created, the alert will have units of time left in minutes and the alert will automatically be destroyed if the estimate time option is subsequently turned off.

If the OS reports multiple batteries, the alert will be a master alert which is duplicated for each battery.

CPU/Motherboard Sensors - Temperature, Voltages, and Fan RPM

Linux:

Sensor monitoring on Linux requires that either lm_sensors modules are installed in your running kernel, that you run a kernel >= 2.6 with sysfs sensors configured, or, for i386 architectures, that you have the mbmon daemon running when **gkrellm** is started (as long as mbmon supports reporting sensor values for your motherboard).

For lm_sensors to be used, **gkrellm** must be compiled with libsensors support. It will be if the libsensors development package is installed when **gkrellm** is compiled. Using libsensors is the preferred interface on Linux since it is the only interface that will be up to date on supporting correct voltage scaling factors and offsets for recent sensor chips.

If the mbmon daemon is used, it must be started before **gkrellm** like so:

```
mbmon -r -P port-number
```

where the given port-number must be configured to match in the **gkrellm** Sensors->Options config. If you have mbmon installed from a distribution package, you can probably easily set up for mbmon to be started at boot. With Debian, for example, you would edit the file /etc/default/mbmon to set:

```
START_MBMON=1
```

and you would need to set in the **gkrellm** Sensors->Option config the mbmon port to be 411 to match the default in the /etc/default/mbmon file.

Sensor temperatures can also be read from /proc/acpi/thermal_zone, /proc/acpi/thermal, /proc/acpi/ibm, the PowerMac Windfarm /sysfs interface, and PowerMac PMU /sysfs based sensors.

When using lm_sensors, libsensors will be used if available, but if libsensors is not linked into the program, the sensor data will be read directly from the /sysfs or /proc file systems. If running a newer Linux kernel sensor module not yet supported by libsensors and libsensors is linked, there will also be an automatic fallback to using /sysfs as long as libsensors doesn't detect any sensors. But if it does detect some sensors which does not include the new sensors you need, you can force getting /sysfs sensor data either by running:

```
gkrellm --without-libsensors
```

or by rebuilding with:

```
make without-libsensors=yes
```

Disk temperatures may also be monitored if you have the hddtemp daemon running when

gkrellm is started. **gkrellm** uses the default hddtemp port of 7634. Like mbmon, hddtemp is best started in a boot script to guarantee it will be running when **gkrellm** is started.

NVIDIA graphics card GPU temperatures may also be monitored if the nvidia-settings command is installed and your Nvidia card supports the temperature reporting. If nvidia-settings is not installed or does not report temperatures for your card, an option for using the nvclock program will appear in the Sensors config. Nvclock use is not automatically enabled as is nvidia-settings because nvclock can add seconds of **gkrellm** startup time when used on a NVIDIA GPU chipset it does not support. GKrellM must be restarted to recognize changes for the nvclock option.

Windows:
Requires a MBM install: http://mbm.livewiredev.com/.

FreeBSD:
Builtin sensor reporting is available for some sensor chips. FreeBSD systems can also read sensor data from the mbmon daemon as described in the Linux section above.

NetBSD:
Builtin sensor reporting is available for some sensor chips. NetBSD uses the envsys(4) interface and sensors reading is automatically enabled if you have either a lm(4) or viaenv(4) chip configured in your kernel.

General Setup:
Temperature and fan sensor displays may be optionally located on the CPU or Proc panels to save some vertical space while voltages are always displayed on their own panel. If you set up to monitor both a temperature and a fan on a single CPU or Proc panel, they can be displayed optionally as an alternating single display or as separate displays. If separate, the fan display will replace the panel label. The configuration for this is under the CPU and Proc config pages.

If not using libsensors, in the Setup page for the Sensors config enter any correction factors and offsets for each of the sensors you are monitoring (see below and lm_sensor documentation). For Linux, default values are automatically provided for many sensor chips.

But if using libsenors, it is not possible to enter correction factors and offsets on the Sensors config page because libsensors configuration is done in the /etc/sensors.conf file. To get sensor debug output and to find out the sensor data source, run:

```
gkrellm -d 0x80
```

Note for NetBSD users:

The current implementation of the sensor reading under NetBSD opens /dev/sysmon and never closes it. Since that device does not support concurrent accesses, you won't be able to run other apps such as envstat(8) while GKrellM is running. This might change if this happens to be an issue.

The reasons for this choice are a) efficiency (though it might be possible to open/close /dev/sysmon each time a reading is needed without major performance issue) and b) as of October 2001, there's a bug in the envsys(4) driver which sometimes causes deadlocks when processes try to access simultaneously /dev/sysmon (see NetBSD PR#14368). A (quick and dirty) workaround for this is to monopolize the driver :)

CPU/Motherboard Temperatures

Most modern motherboards will not require setting temperature correction factors and offsets other than the defaults. However, for lm_sensors it is necessary to have a correct set sensor line in */etc/sensors.conf* if the temperature sensor type is other than the default thermistor. If using Linux sysfs sensors, this sensor type would be set by writing to a sysfs file. For example, you might at boot set a sysfs temperature sensor type with:

```
echo 2 > /sys/bus/i2c/devices/0-0290/sensor2
```

On the other hand, some older motherboards may need temperature calibration by setting a correction factor and offset for each temperature sensor because of factors such as variations in

physical thermistor contact with the CPU. Unfortunately, this calibration may not be practical or physically possible because it requires that somehow you can get a real CPU temperature reading. So, the calibration discussion which follows should probably be considered an academic exercise that might give you some good (or bad) ideas. If you have a recent motherboard, skip the following.

Anyway, to do this calibration, take two real CPU temperature readings corresponding to two sensor reported readings. To get the real readings, you can trust that your motherboard manufacturer has done this calibration and is reporting accurate temperatures in the bios, or you can put a temperature probe directly on your CPU case (and this is where things get impractical).

Here is a hypothetical CPU calibration procedure. Make sure **gkrellm** is configured with default factors of 1.0 and offsets of 0 and is reporting temperatures in centigrade:

1 • Power on the machine and read a real temperature T1 from the bios or a temperature probe. If reading from the bios, proceed with booting the OS. Now record a sensor temperature S1 as reported by **gkrellm.**

2 • Change the room temperature environment (turn off your AC or change computer fan exhaust speed). Now repeat step 1, this time recording a real temperature T2 and **gkrellm** reported sensor temperature S2.

3 • Now you can calculate the correction factor and offset you need to enter into the Sensor configuration tab:

From:

$$\frac{s - S1}{S2 - S1} = \frac{t - T1}{T2 - T1}$$

$$t = s * \frac{T2 - T1}{S2 - S1} + \frac{S2*T1 - S1*T2}{S2 - S1}$$

So:

$$factor = \frac{T2 - T1}{S2 - S1} \quad offset = \frac{S2*T1 - S1*T2}{S2 - S1}$$

Voltage Sensor Corrections

You need to read this section only if you think the default voltage correction factors and offsets are incorrect. For Linux and lm_sensors and sysfs sensors this would be if **gkrellm** does not know about your particular sensor chip. For MBM with Windows, the default values should be correct.

Motherboard voltage measurements are made by a variety of sensor chips which are capable of measuring a small positive voltage. GKrellM can display these voltage values and can apply a correction factor, offset, and for the negative voltages of some chips (lm80), a level shifting reference voltage to the displayed voltage. There are four cases to consider:

1 • Low valued positive voltages may be directly connected to the input pins of the sensor chip and therefore need no correction. For these, the correction factor should be 1.0 and the offset should be 0.

2 • Higher valued positive voltages will be connected to the input pins of the sensor chip through a 2 resistor attenuation circuit. For these, the correction factor will be a ratio of the resistor values and the offset will be 0.

3 • Negative voltages will be connected to the input pins of the sensor through a 2 resistor attenuation circuit with one of the resistors connected to a positive voltage to effect a voltage level shift. For these (lm80), the correction factor and offset will be ratios of the

resistor values, and a reference voltage must be used.

4 • Some sensor chips (w83782, lm78) are designed to handle negative inputs without requiring an input resistor connected to a voltage reference. For these, there will be a correction factor and a possible offset.

For cases 2 and 3, the sensor chip input network looks like:

```
Vs o----///---o------------o Vin
R1 |
o--///--o Vref
R2
```

where,

Vs is the motherboard voltage under measurement

Vin is the voltage at the input pin of the sensor chip and therefore is the voltage reading that will need correction.

Vref is a level shifting voltage reference. For case 2, Vref is ground or zero. For case 3, Vref will be one of the positive motherboard voltages.

The problem then is to compute correction factors and offsets as a function of R1 and R2 so that GKrellM can display a computed motherboard voltage Vs as a function of a measured voltage Vin.

Since sensor chip input pins are high impedance, current into the pins may be assumed to be zero. In that case, the current through R1 equals current through R2, and we have:

(Vs - Vin)/R1 = (Vin - Vref)/R2

Solving for Vs as a function of Vin:

Vs = Vin * (1 + R1/R2) - (R1/R2) * Vref

So, the correction factor is: 1 + R1/R2
the correction offset is: - (R1/R2)
Vref is specified in the config separately from
the offset (for chips that need it).

Fortunately there seems to be a standard set of resistor values used for the various sensor chips which are documented in the lm_sensor documentation. The GKrellM sensor corrections are similar to the compute lines you find with lm_sensors, with the difference that lm_sensors has an expression evaluator which does not require that compute lines be simplified to the single factor and offset required by GKrellM. But you can easily calculate the factor and offset. For example, this lm_sensor compute line for a case 2 voltage:

compute in3 ((6.8/10)+1)*@ , @/((6.8/10)+1)

yields a correction factor of ((6.8/10)+1) = 1.68 and an offset of zero.

Note that the second compute line expression is not relevant in GKrellM because there is never any need to invert the voltage reading calculation. Also, the compute line '@' symbol represents the Vin voltage.

A more complicated compute line for a case 3 voltage:

compute in5 (160/35.7)*(@ - in0) + @, ...

can be rewritten:

compute in5 (1 + 160/35.7)*@ - (160/35.7)*in0, ...

so the correction factor is (1 + 160/35.7) = 5.48
and the correction offset is -(160/35.7) = -4.48

and the voltage reference Vref is in0

Here is a table of correction factors and offsets based on some typical compute line entries from /etc/sensors.conf:

```
Compute line Factor Offset Vref
-----------------------------------------------
lm80 in0 (24/14.7 + 1) * @ 2.633 0 -
in2 (22.1/30 + 1) * @ 1.737 0 -
in3 (2.8/1.9) * @ 1.474 0 -
in4 (160/30.1 + 1) * @ 6.316 0 -
in5 (160/35.7)*(@-in0) + @ 5.482 -4.482 in0
in6 (36/16.2)*(@-in0) + @ 3.222 -2.222 in0

LM78 in3 ((6.8/10)+1)*@ 1.68 0 -
in4 ((28/10)+1)*@ 3.8 0 -
in5 -(210/60.4)*@ -3.477 0 -
in6 -(90.9/60.4)*@ -1.505 0 -

w83782 in5 (5.14 * @) - 14.91 5.14 -14.91 -
in6 (3.14 * @) - 7.71 3.14 -7.71 -
```

Command launching

Many monitors can be set up to launch a command when you click on the monitor label. When a command is configured for a monitor, its label is converted into a button which becomes visible when the mouse enters the panel or meter area of the label. If the command is a console command (doesn't have a graphical user interface), then the command must be run in a terminal window such as xterm, eterm, or Gnome terminal. For example running the top command would take:

xterm -e top

You can use the command launching feature to run commands related to monitoring functions, or you may use it to have a convenient launch for any command. Since **gkrellm** is usually made sticky, you can have easy access to several frequently used commands from any desktop. This is intended to be a convenience and a way to maximize utilization of screen real estate and not a replacement for more full featured command launching from desktops such as Gnome or KDE or others. Some launch ideas for some monitors could be:

calendar:
: gnomecal, evolution, or ical

CPU: xterm -e top or gps or gtop

inet: gftp or xterm -e ftpwho

net: mozilla, galeon, skipstone, or xterm -e slrn -C-

And so on... Tooltips can be set up for these commands.

Alerts

Most monitors can have alerts configured to give warnings and alarms for data readings which range outside of configurable limits. Where useful, a delay of the alert trigger can be configured. A warning or alarm consists of an attention grabbing decal appearing and an optional command being executed. For most monitors the command may contain the same substitution variables which are available for display in the chart or panel label format strings and are documented on configuration Info pages. Additionally, the hostname may be embedded in the command with the $H substitution variable.

If you have festival installed, either a warn or alarm command could be configured to speak something. For example a CPU temperature alert warn command could just speak the current

temperature with:

sh -c echo warning C P U is at $s degrees | esddsp festival --tts

Assuming you have esd running.

THEMES

A theme is a directory containing image files and a *gkrellmrc* configuration file. The theme directory may be installed in several locations:

~/.gkrellm2/themes
/usr/local/share/gkrellm2/themes
/usr/share/gkrellm2/themes

For compatibility with Gtk themes, a **gkrellm** theme may also be installed as:

~/.themes/THEME_NAME/gkrellm2
/usr/share/themes/THEME_NAME/gkrellm2

Finally, a theme you simply want to check out can be untarred anywhere and used by running:

gkrellm -t path_to_theme

If you are interested in writing a theme, go to the Themes page at http://www.gkrellm.net and there you will find a Theme making reference.

PLUGINS

gkrellm tries to load all plugins (shared object files ending in .so) it finds in your plugin directory *~/.gkrellm2/plugins.* The directories */usr/local/lib/gkrellm2/plugins* and */usr/lib/gkrellm2/plugins* are also searched for plugins to install.

Some plugins may be available only as source files and they will have to be compiled before installation. There should be instructions for doing this with each plugin that comes in source form.

If you are interested in writing a plugin, go to the Plugins page at http://www.gkrellm.net and there you will find a Plugin programmers reference.

CLIENT/SERVER

When a local **gkrellm** runs in client mode and connects to a remote **gkrellmd** server all builtin monitors collect their data from the server. However, the client **gkrellm** process is running on the local machine, so any enabled plugins will run in the local context (Flynn is an exception to this since it derives its data from the builtin CPU monitor). Also, any command launching will run commands on the local machine.

FILES

~/.gkrellm2
: User gkrellm directory where are located configuration files, user's plugins and user's themes.

~/.gkrellm2/plugins
: User plugin directory.

/usr/lib/gkrellm2/plugins
: System wide plugin directory.

/usr/local/lib/gkrellm2/plugins
: Local plugin directory.

~/.gkrellm2/themes
: User theme directory.

~/.themes/THEME_NAME/gkrellm2
: User theme packaged as part of a user Gtk theme.

/usr/share/gkrellm2/themes
System wide theme directory.

/usr/local/share/gkrellm2/themes
Local theme directory.

/usr/share/themes/THEME_NAME/gkrellm2
System wide theme packaged as part of a system wide Gtk theme.

AUTHORS

This manual page was written by Bill Wilson <billw@gkrellm.net>. http://www.gkrellm.net/

SEE ALSO

fstab(5), **sudo(1),** mount(8), **pppd(8),** umount(8)