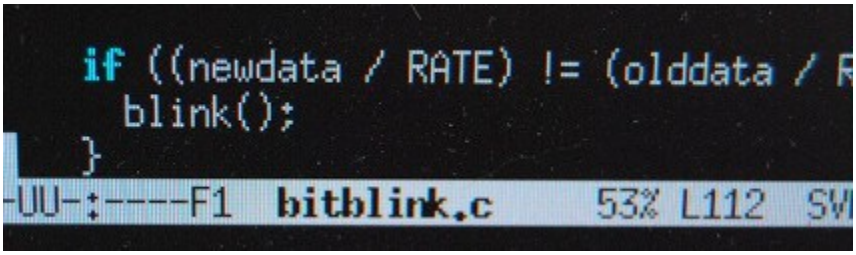
NETWORK ACTIVITY LED WITH LINUX LED SUBSYSTEM

This is a nice userspace application I use on my router to control the Internet connection status LED in “smart” way. The idea is simple, instead of just randomly blink the LED when there is some activity on the network, this application checks for the total bytes transferred on the network interface, and blinks the LED every 100KB of data.



That behavior is borrowed from modern electric counter, which have a LED that blinks every predefined number of Watt/hour.

That’s useful because you can quickly have an idea of the bandwidth utilization of your connection by just checking how often the LED blinks, so you can instantly identify a low-bandwidth constant traffic by a high-bandwidth traffic.

Operation

All the configuration is done in #defines at the beginning of the source. This is not so elegant but simplifies the whole thing.

```
#define RX_FILE "/sys/class/net/ppp0/statistics/rx_bytes"
#define TX_FILE "/sys/class/net/ppp0/statistics/tx_bytes"
#define RATE (100 * 1024)
#define LED_FILE "/sys/class/leds/hdd/brightness"
```

When started, the application open the LED_FILE path, which have to be set to the “brightness” file of the LED to be used. After that, the application daemonize, and start checking the combined traffic on the interface “ppp0″ every 100ms.

From this moment, every time the total traffic grow up to the configured RATE, the led is blinked for 100ms.

That’s it!

Start and Stop

To make some sense, the application have to be started when the PPP connection is established and killed when the connection goes down.

Also, the status LED should be normally ON when the connection is up, and goes OFF when the connection is terminated.

That’s easily obtained tweaking the ip-up and ip-down helper scripts, such as:

/etc/ppp/ip-up.d/01leds

```
#!/bin/sh

echo 1 > "/sys/class/leds/hdd/brightness"

if ! pgrep bitblink > /dev/null 2>&1 ; then
  /usr/local/sbin/bitblink
fi
```

/etc/ppp/ip-down.d/01leds

```
#!/bin/sh

if pgrep bitblink > /dev/null 2>&1 ; then
  killall -9 bitblink
fi

echo 0 > "/sys/class/leds/hdd/brightness"
```

Source code

That's the whole source code, compile it with a command like:

```
gcc -Wall -O2 -o bitblink bitblink.c
```

bitblink.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define RX_FILE "/sys/class/net/ppp0/statistics/rx_bytes"
#define TX_FILE "/sys/class/net/ppp0/statistics/tx_bytes"

#define RATE (100 * 1024)

#define LED_FILE "/sys/class/leds/hdd/brightness"

static void daemonize (void)
{
  int i;
  pid_t pid;

  if ((i = open("/dev/null", O_RDONLY)) != 0) {
    dup2(i, 0);
    close(i);
  }
  if ((i = open("/dev/null", O_WRONLY)) != 1) {
    dup2(i, 1);
    close(i);
  }
  if ((i = open("/dev/null", O_WRONLY)) != 2) {
    dup2(i, 2);
    close(i);
  }

  setsid();

  pid = fork();

  if (pid < 0) {
    perror("fork");
```

```
    exit(1);
  } else if (pid) { /* parent */
    exit(0);
  } else { /* child */
  }
}


static unsigned long long update (void)
{
  char buf[32];
  int fd;
  unsigned long long ret;

  /* RX */
  fd = open(RX_FILE, O_RDONLY);
  if (fd < 0) {
    perror("open");
    return -1;
  }

  read(fd, buf, 32);
  ret = atoll(buf);

  close(fd);

  /* TX */
  fd = open(TX_FILE, O_RDONLY);
  if (fd < 0) {
    perror("open");
    return -1;
  }

  read(fd, buf, 32);
  ret += atoll(buf);

  close(fd);

  return ret;
}

static int led_fd;

static void blink (void)
{
  write(led_fd, "0", 1);

  usleep(100 * 1000);

  write(led_fd, "1", 1);
}

int main (int argc, char ** argv)
{
```

```
80      unsigned long long newdata;
81      unsigned long long olddata;
82
83      led_fd = open(LED_FILE, O_WRONLY);
84      if (led_fd < 0) {
85        perror("open");
86        exit(-1);
87      }
88
89      daemonize();
90
91      olddata = 0;
92      for (;;) {
93        newdata = update();
94
95        /* printf("%lld\n", newdata / RATE); */
96
97        if ((newdata / RATE) != (olddata / RATE)) {
98          blink();
99        }
100
101       olddata = newdata;
102
103       usleep(100 * 1000);
104     }
105
106     close(led_fd);
107 }
108
109
110
111
112
113
114
115
116
117
118
119
120
```

Source : http://fabiobaltieri.com/2011/12/03/network-activity-led-linux/