Hammer LED Control Program

This example program was provided by the good folks at gentoo linux (http://www.gentoo.org/). Gentoo has a web page dedicated for the Hammer/Nail Board located at:
http://www.gentoo.org/proj/en/base/embedded/handbook/?part=4&chap=1

The ledctl program allows you to control the Hammer and Nail Board LEDs from the command line.

Usage

To compile the source use your toolchain command line:

```
arm-linux-uclibc-gcc ledctl.c -o ledctl
```

Then transfer the ledctl binary from the PC to the Hammer file system. Make sure that the permissions on the file are set to executable.

```
chmod 755 ledctl
```

Executing ledctl with no parameters will display the following results:

```
./ledctl

Usage:
       ./ledctl <led> <state>
       ./ledctl <state>
```

```
/*
 * Distributed under the terms of the GNU General Public License v2
 * $Header: $
 *
 * Portable led control interface using sysfs files.
 * Written for the TCT NailKit in 2008 by solar@gentoo.org
 */

/*
 * This program has the same net result as echoing values in the the sys files.
 * We just try to automate some of the logic for controling the items thru a common
 * interface.
 * We have the distant advantage here that embedded vendors simply need to make sure they
 * enable the leds in the correct boardfile. And users don't have to worry about custom
 * led drivers etc..
 *
 * Examples:
 *
 * ledctl 0   (Turns all LEDS off)
 * ledctl 1   (Turns all LEDS on)
 *
 * ledctl 0 0 (Turns off LED 0)
 * ledctl 0 1 (Turns on LED 0)
 *
 * ledctl 1 1 (Turns on LED 1)
 * ledctl 1 0 (Turns off LED 1)
 *
 * ledctl 2 1 (Turns on LED 2)
 * ledctl 2 0 (Turns off LED 2)
 *
 * and so on...
 *
 * Return values:
 *    status: 0 (all good)
 *    status: 1 (usage syntax/help)
 *    status: 255/-1 (no such led)
 */

/* arm-softfloat-linux-uclibc-gcc ledctl.c -Os -fomit-frame-pointer -march=armv4t -mtune=arm920t -s -o ledctl */

#include <stdio.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <dirent.h>

const char *led_path = "/sys/class/leds/";

#define ledctl sysfs_ledctl
signed int sysfs_ledctl(const int led, const int val, const char *trigger) {
      FILE *fp;
      char fname[64];
      int state = -1;

      snprintf(fname, sizeof(fname), "%s/led%d/trigger", led_path, led);
      if ((fp = fopen(fname, "w")) != NULL) {
            fprintf(fp, "%s\n", trigger == NULL ? "none" : trigger);
            fclose(fp);
      }
      snprintf(fname, sizeof(fname), "%s/led%d/brightness", led_path, led);
      if (val >= 0 && val <= 255) {
            if ((fp = fopen(fname, "w")) == NULL)
                  return state;
            fprintf(fp, "%d\n", val);
            fclose(fp);
      }
      if ((fp = fopen(fname, "r")) != NULL) {
            fscanf(fp, "%d", &state);
            fclose(fp);
      }
      return state;
}

int main(int argc, char **argv) {
      struct dirent **dnames;
      int state = 1, ret = 0;
      int count;

      if (argc == 1) {
            printf("Usage:\n\t%s <led> <state>\n\t%s <state>\n", *argv, *argv);
            return 1;
      }
      if (argc > 1)
            state = atoi(argv[1]);
```

```
	if (argc > 2) {
		ret = ledctl(atoi(argv[1]), atoi(argv[2]), argc > 3 ? argv[3] : NULL);
		return (ret < 0 ? ret : 0);
	}
	if ((count = scandir(led_path, &dnames, NULL, alphasort)) < 0) {
		fprintf(stderr, "No leds found in %s\n", led_path);
		return -1;
	}
	while (count--) {
		ret = ledctl(count, state, NULL);
		free(dnames[count]);
	}
	  free(dnames);
	return (ret < 0 ? ret : 0);
```
