```
/* Disk protection for HP machines.
 *
 * Copyright 2008 Eric Piel
 * Copyright 2009 Pavel Machek <pavel@suse.cz>
 *
 * GPLv2.
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <string.h>
#include <stdint.h>
#include <errno.h>
#include <signal.h>
#include <sys/mman.h>
#include <sched.h>

char unload_heads_path[64];

int set_unload_heads_path(char *device)
{
        char devname[64];

        if (strlen(device) <= 5 || strncmp(device, "/dev/", 5) != 0)
                return -EINVAL;
        strncpy(devname, device + 5, sizeof(devname));

        snprintf(unload_heads_path, sizeof(unload_heads_path),
                                "/sys/block/%s/device/unload_heads", devname);
        return 0;
}
int valid_disk(void)
{
        int fd = open(unload_heads_path, O_RDONLY);
        if (fd < 0) {
                perror(unload_heads_path);
                return 0;
        }

        close(fd);
        return 1;
}

void write_int(char *path, int i)
{
        char buf[1024];
        int fd = open(path, O_RDWR);
        if (fd < 0) {
                perror("open");
                exit(1);
        }
        sprintf(buf, "%d", i);
        if (write(fd, buf, strlen(buf)) != strlen(buf)) {
                perror("write");
                exit(1);
        }
        close(fd);
}

void set_led(int on)
{
        write_int("/sys/class/leds/hp::hddprotect/brightness", on);
}

void protect(int seconds)
{
        write_int(unload_heads_path, seconds*1000);
}

int on_ac(void)
{
//      /sys/class/power_supply/AC0/online
}

int lid_open(void)
{
//      /proc/acpi/button/lid/LID/state
}

void ignore_me(void)
{
        protect(0);
        set_led(0);
}

int main(int argc, char **argv)
{
        int fd, ret;
        struct sched_param param;

        if (argc == 1)
                ret = set_unload_heads_path("/dev/sda");
        else if (argc == 2)
                ret = set_unload_heads_path(argv[1]);
        else
                ret = -EINVAL;

        if (ret || !valid_disk()) {
                fprintf(stderr, "usage: %s <device> (default: /dev/sda)\n",
                                argv[0]);
                exit(1);
        }

        fd = open("/dev/freefall", O_RDONLY);
        if (fd < 0) {
                perror("/dev/freefall");
                return EXIT_FAILURE;
        }
```

```
	daemon(0, 0);
	param.sched_priority = sched_get_priority_max(SCHED_FIFO);
	sched_setscheduler(0, SCHED_FIFO, &param);
	mlockall(MCL_CURRENT|MCL_FUTURE);

	signal(SIGALRM, ignore_me);

	for (;;) {
		unsigned char count;

		ret = read(fd, &count, sizeof(count));
		alarm(0);
		if ((ret == -1) && (errno == EINTR)) {
			/* Alarm expired, time to unpark the heads */
			continue;
		}

		if (ret != sizeof(count)) {
			perror("read");
			break;
		}

		protect(21);
		set_led(1);
		if (1 || on_ac() || lid_open())
			alarm(2);
		else
			alarm(20);
	}

	close(fd);
	return EXIT_SUCCESS;
}
```