

- Main
- Events
- Meetings
- Joining
- News

Libparted Handbook

From IlugCal

Libparted is a library for creating, destroying, resizing, checking and copying partitions, and the file systems on them. It is one of the components of the GNU Parted (<http://web.archive.org/web/20080630020636/http://www.gnu.org/software/parted/>) project, and is natively written in C. However C and Python bindings are provided by a couple of external projects-- Libpartedpp and Pyparted respectively.

Libparted is the backbone of a number of disk partitioning utilities. eg., Parted (the commandline front-end component of GNU Parted), GParted (<http://web.archive.org/web/20080630020636/http://gparted.sourceforge.net/>) (a GNOME (<http://web.archive.org/web/20080630020636/http://www.gnome.org/>) package) and QtParted. It is also used by Anaconda, the Fedora (<http://web.archive.org/web/20080630020636/http://fedoraproject.org/wiki/>) and Red Hat (<http://web.archive.org/web/20080630020636/http://www.redhat.com/>) installer, the Gentoo (<http://web.archive.org/web/20080630020636/http://www.gentoo.org/>) installer, and the Debian (<http://web.archive.org/web/20080630020636/http://www.debian.org/>) Installer.

Contents

- 1 Devices
 - 1.1 Probing
 - 1.2 Reading
- 2 Disk Labels
 - 2.1 Creating
 - 2.2 Probing
 - 2.3 Reading
- 3 Note
- 4 External Links

Devices

Devices refer to block storage devices like hard disks, floppy disks, USB sticks or pen

drives, etc..

Devices are represented by instances of *PedDevice*.

Probing

Probing involves detecting all the devices on the system. The *ped_device_probe_all* function creates a list of all the detected devices, which can be traversed using the *ped_device_get_next* function.

The following example lists all available devices on the system. (NB: Run as root.)

```
-----  
#include <error.h>  
#include <stdio.h>  
#include <parted/parted.h>  
int main (void)  
{  
    PedDevice* device = NULL;  
  
    ped_device_probe_all ();  
  
    while ((device = ped_device_get_next(device)))  
        puts (device->path);  
  
    return 0;  
}
```

Reading

To read the properties of device we first need to create a volatile representation of the device in the memory in the form of a *PedDevice* instance. This is done using the *ped_device_get* function which takes the path to the device file representing the block device as argument. eg., /dev/hda, /dev/sda, etc..

The following example displays the details of a particular device that was detected in the previous example. (NB: Run as root.)

```

#include <error.h>
#include <stdio.h>
#include <parted/parted.h>

int main (int argc, char* argv[])
{
    char* size;
    char* transport[13] = {"unknown", "scsi", "ide", "dac960", "cpqarray",
                          "file", "ataraid", "i2o", "ubd", "dasd", "viодasd",
                          "sx8", "dm"};

    PedDevice* device;

    if (argc != 2)
        error (EXIT_FAILURE, 0, "wrong number of arguments");

    device = ped_device_get (argv[1]);
    if (device == NULL)
        goto error;

    size = ped_unit_format_byte (device, device->length * device->sector_size);
    if (size == NULL)
        goto error;

    printf ("Model: %s\n", device->model);
    printf ("Transport: %s\n", transport[device->type]);
    printf ("Size: %s\n", size);
    printf ("Sector size (logical/physical): %lldB/%lldB\n",
            device->sector_size,
            device->phys_sector_size);

    ped_free (size);
    return 0;

error:
    return 1;
}

```

Disk Labels

Disk labels are synonymous with partition tables. Every block storage device, like a hard drive, has a disk label which contains the meta-data about the location and type of the partitions on the device and file systems (if any) on those partitions. It is to be noted that the disk label itself does not belong to any partition.

Libparted can work with a number of different types of disk labels-- aix, amiga, bsd, dvh, gpt, mac, msdos, sun and loop. A loop disk label is a special type used by Libparted to treat partitions as a block device.

Disk labels are represented by instances of *PedDisk* and the types are represented by *PedDiskType*.

Libparted maintains a linked list of the different types of disk labels it supports, and one can walk through these list using the *ped_disk_type_get_next* function. The following example will display a list of the different disk label types supported by Libparted.

```
#include <error.h>
#include <stdio.h>
#include <parted/parted.h>

int main (int argc, char* argv[])
{
    PedDiskType* type;

    for (type = ped_disk_type_get_next (NULL); type;
         type = ped_disk_type_get_next (type))
        puts (type->name);

    return 0;
}
```

Creating

Creation of a disk label involves two broad steps.

- A representation of the disk label has to be created in the memory. This is done by the *ped_disk_new_fresh* function which takes the block device on which the disk label is to be created and the type of disk label as arguments. The device is untouched at this stage and the change is volatile.
- The freshly created disk label is then committed to the block device using the *ped_disk_commit* function, which takes the disk label created in the previous step as an argument. This modifies the contents on the device and change is made permanent.

The following example creates a disk label of a specific type on a given block device.

```
#include <error.h>
#include <stdio.h>
#include <parted/parted.h>

int main (int argc, char* argv[])
{
    PedDevice* device;
    PedDisk* disk;
    PedDiskType* type;

    if (argc != 3)
        error (EXIT_FAILURE, 0, "wrong number of arguments");

    device = ped_device_get (argv[1]);
    if (device == NULL)
        goto error;

    type = ped_disk_type_get (argv[2]);
    if (type == NULL)
        goto error_destroy_device;

    disk = ped_disk_new_fresh (device, type);
    if (disk == NULL)
        goto error_destroy_device;

    if (!ped_disk_commit (disk))
        goto error_destroy_disk;

    ped_disk_destroy (disk);
    ped_device_destroy (device);
    return 0;

error_destroy_disk:
    ped_disk_destroy (disk);
error_destroy_device:
    ped_device_destroy (device);
error:
    return 1;
}
```

Probing

Probing involves figuring out the type of the existing disk label on a block device. The *ped_disk_probe* function takes the device to be probed as an argument and returns the type of the disk label on it.

The following example probes a given block device and prints the name of the disk label type found on it.

```
#include <error.h>
#include <stdio.h>
#include <parted/parted.h>

int main (int argc, char* argv[])
{
    PedDevice* device;
    PedDiskType* type;

    if (argc != 2)
        error (EXIT_FAILURE, 0, "wrong number of arguments");

    device = ped_device_get (argv[1]);
    if (device == NULL)
        goto error;

    type = ped_disk_probe (device);
    if (type == NULL)
        goto error_destroy_device;

    puts (type->name);

    ped_device_destroy (device);
    return 0;

error_destroy_device:
    ped_device_destroy (device);
error:
    return 1;
}
```

Reading

We can read the disk label of a block device using the *ped_disk_new* function. This creates a volatile representation of the disk label on the device in the memory in the form of a *PedDisk* instance.

The following example displays the partitions present on a specified block device by first reading the disk label on the device. Each partition is an instance of *PedPartition*. For each partition we display the partition number, the starting sector, size in sectors and the file system name (if any).

```

#include <error.h>
#include <stdio.h>

#include <parted/parted.h>

int main (int argc, char* argv[])
{
    PedDevice* device;
    PedDisk* disk;
    PedPartition* part;

    if (argc != 2)
        error (EXIT_FAILURE, 0, "wrong number of arguments");

    device = ped_device_get (argv[1]);
    if (device == NULL)
        goto error;

    disk = ped_disk_new (device);
    if (disk == NULL)
        goto error_destroy_device;

    printf ("%3s  %s  %s  %s\n", "no.", "start", "size", "fs");

    for (part = ped_disk_next_partition (disk, NULL); part;
        part = ped_disk_next_partition (disk, part))
    {
        if (part->num < 0)
            continue;

        printf ("=  %lld  %lld  %s\n", part->num,
                part->geom.start,
                part->geom.length,
                (part->fs_type) ? part->fs_type->name : "");
    }

    ped_disk_destroy (disk);
    ped_device_destroy (device);
    return 0;

error_destroy_device:
    ped_device_destroy (device);
error:
    return 1;
}

```

Note

- All the examples in this page require Libparted to be installed on the system for compilation. Pass *-lparted* to GCC for linking the code against the library.

```
user@box ~$ gcc foo.c -lparted
```

- It is advisable not to experiment with the actual block devices, like */dev/hda*, */dev/sda*, etc.. It is better one uses a *loop device* to try out the examples. A loop device is nothing but a file, and the size of the device is the same as the size of the file. You can create a loop device (say */tmp/devloop*) using the *dd* command.

```
user@box ~$ dd if=/dev/zero of=/tmp/devloop bs=512 count=40960
```

External Links

- GNU Parted home page: <http://www.gnu.org/software/parted/>
- GNU Parted wiki: <http://parted.alioth.debian.org/>

Retrieved from "http://www.ilug-cal.org/wiki/index.php/Libparted_Handbook"

- This page was last modified 13:11, 19 September 2007.
- This page has been accessed 2,053 times.
- Content is available under Attribution-Noncommercial 3.0 .
 - About IlugCal
 - Disclaimers

Design of this site is based on the theme used at the Tango Project, available under the terms and conditions of the Creative Commons Attribution Share-Alike license.