GetDevInfo Documentation

Release 1.0.9

Hamish McIntyre-Bhatty

Mar 13, 2020

CONTENTS

Contents:

CHAPTER

ONE

DOCUMENTATION FOR THE OUTPUT FORMAT

This module outputs data in a precisely-formatted dictionary object. In order for it to be useful, this format, and the information that is provided in it, needs to be explained precisely.

This format is the same on both Linux and macOS, but the macOS version of this library currently has less functionality, so some of the information isn't present on that version. Instead, placeholders like "N/A" or "Unknown" are used. Those instances will be pointed out here.

1.1 For each device and partition:

A sub-dictionary is created with the name of that disk as its key.

For example: To access the info for /dev/disk1s1, use:

```
>>> DISKINFO['/dev/disk1s1']
```

1.2 Inside this sub-dictionary (standard devices):

Various information is collected and organised here.

'Name': The disk's name, stored as a string.

'Type': Whether the disk is a "Device" or "Partition", stored as a string.

For example:

```
>>> DISKINFO['/dev/sda']['Type']
>>> "Device"
```

'HostDevice': The "parent" or "host" device of a partition, stored as a string. For a device, this is always set to "N/A". For an LVM disk, this is the host device of the containing partition. eg: /dev/sdb.

Example 1:

```
>>> DISKINFO['/dev/sda']['HostDevice']
>>> "N/A"
```

Example 2:

```
>>> DISKINFO['/dev/sde5']['HostDevice']
>>> "/dev/sde"
```

Example 3:

```
>>> DISKINFO['/dev/disk1s3']['HostDevice']
>>> "/dev/disk1"
```

'Partitions': All the partitions a device contains, stored as a list. For partitions, this is always set to [].

Example 1:

```
>>> DISKINFO['/dev/sda1']['Partitions']
>>> []
```

Example 2:

```
>>> DISKINFO['/dev/sda']['Partitions']
>>> ["/dev/sda1", "/dev/sda2", "/dev/sda3"]
```

Example 3:

```
>>> DISKINFO['/dev/disk0']['Partitions']
>>> ["/dev/disk0s1", "/dev/disk0s2"]
```

'Vendor': The device's/partition's vendor. For a device, this is often the brand. For partitions this is more random, but often has something to do with the file system type, or the OS that created the partition.

Example 1:

```
>>> DISKINFO['/dev/sda']['Vendor']
>>> "VBOX"
```

Example 2:

```
>>> DISKINFO['/dev/sda1']['Vendor']
>>> "Linux"
```

Example 3:

```
>>> DISKINFO['/dev/disk0s1']['Vendor']
>>> "VBOX"
```

'Product': The device's product information. Often model information such as a model name/number. For a partition, this is always the same as it's host device's product information, prefixed by "Host Device: ".

Example 1:

```
>>> DISKINFO['/dev/sda']['Product']
>>> "ST1000DM003-1CH1"
```

Example 2:

```
>>> DISKINFO['/dev/sda1']['Product']
>>> "Host Device: ST1000DM003-1CH1"
```

Example 3:

```
>>> DISKINFO['/dev/disk0']['Product']
>>> "HARDDISK"
```

'Capacity', and 'RawCapacity': The disk's capacity, in both human-readable form, and program-friendly form. Ignored for some types of disks, like optical drives. The human-readable capacity is rounded to make it a 3 digit number. The machine-readable size is measured in bytes, and it is not rounded.

Example:

```
>>> DISKINFO['/dev/sda']['Capacity']
>>> "500 GB"
```

```
>>> DISKINFO['/dev/sda']['RawCapacity']
>>> "500107862016"
```

'Description': A human-readable description of the disk. Simply here to make it easier for a human to identify a disk. On Linux, these are the descriptions provided by lshw (except for logical volumes), and they are fairly basic. On macOS, these are generated using information from diskutil.

Example 1:

```
>>> DISKINFO['/dev/sda']['Description']
>>> "ATA Disk"
```

Example 2:

```
>>> DISKINFO['/dev/disk1']['Description']
>>> "Internal Hard Disk Drive (Connected through SATA)"
```

'Flags': The disk's capabilities, stored as a list.

Note: Not yet available on macOS, or for logical volumes.

For example:

```
>>> DISKINFO['/dev/cdrom']['Flags']
>>> ['removable', 'audio', 'cd-r', 'cd-rw', 'dvd', 'dvd-r', 'dvd-ram']
```

'Partitioning': The disk's partition scheme. N/A for partitions and logical volumes.

Note: Not yet available on macOS.

Example 1:

```
>>> DISKINFO['/dev/sda']['Partitioning']
>>> "gpt"
```

Example 2:

```
>>> DISKINFO['/dev/sdb']['Partitioning']
>>> "mbr"
```

'FileSystem': The disk's file system. N/A for devices.

Note: Not yet available on macOS.

Example:

```
>>> DISKINFO['/dev/sda']['FileSystem']
>>> "ext4"
```

'UUID': This disk's UUID. N/A for devices. Length changes based on filesystem type. For example, vfat UUIDs are shorter.

Note: Not yet available on macOS.

Example:

```
>>> DISKINFO['/dev/sda1']['UUID']
>>> XXXX-XXXX
```

'ID': The disk's ID.

Note: Not yet available on macOS.

Example:

```
>>> DISKINFO['/dev/sda']['ID']
>>> "usb-Generic_STORAGE_DEVICE_000000001206-0:1"
```

'BootRecord', 'BootRecordStrings': The MBR/PBR of the disk. Can be useful in identifying the bootloader that resides there, if any.

Note: Not yet available on macOS.

1.3 Inside this sub-dictionary (specifics for LVM disks):

These are keys that are only present for LVM disks (where "Product" is "LVM Partition").

'Aliases': Any aliases the disk has. LVM disks can often be accessed using multiple different names. This is a list of those names.

Example:

```
>>> DISKINFO['/dev/mapper/fedora/root']['Aliases']
>>> ['/dev/mapper/fedora/root', '/dev/fedora--localhost-root']
```

'LVName': The name of the logical volume.

Example:

```
>>> DISKINFO['/dev/mapper/fedora/root']['LVName']
>>> "root"
```

'VGName': The name of the volume group the logical volume belongs to.

Example:

```
>>> DISKINFO['/dev/mapper/fedora/root']['VGName']
>>> "fedora"
```

'HostPartition': The partition that contains this logical volume.

Example:

```
>>> DISKINFO['/dev/mapper/fedora/root']['HostPartition']
>>> "/dev/sda"
```

Note: Not always available depending on disk configuration.

Warning: "UUID" may or may not be available for certain disks.

Warning: "Capacity" and "RawCapacity" may not be available for certain disks.

Warning: "HostPartition" and "HostDevice" may not be available for certain disks.

1.4 Inside this sub-dictionary (NVME disks):

Warning: Various standard keys are not available for NVME disks as they aren't supported by lshw.

CHAPTER

TWO

DOCUMENTATION FOR THE GETDEVINFO MODULE

This is the part of the package that you would normally import and use. It detects your platform (Linux or macOS), and runs the correct tools for that platform.

For example:

```
>>> import getdevinfo
>>> getdevinfo.getdevinfo.get_info()
```

Or, more concisely:

```
>>> import getdevinfo.getdevinfo as getdevinfo
>>> getdevinfo.get_info()
```

Will run the correct tools for your platform and return the collected disk information as a dictionary.

Note: You can import the submodules directly, but this might result in strange behaviour, or not work on your platform if you import the wrong one. That is not how the package is intended to be used, except if you want to use the get_block_size() function to get a block size, as documented for each platform later.

`getdevinfo.getdevinfo.`**`get_info`**`()`

This function is used to determine the platform you're using (Linux or macOS) and run the relevant tools. Then, it returns the disk information dictionary to the caller.

Returns: dict, the disk info dictionary.

Raises: Hopefully nothing, but if there is an unhandled error or bug elsewhere, there's a small chance it could propagate to here. If this concerns you, you can wrap this code in a try:, except: clause:

```
>>> try:
>>>     get_info()
>>> except:
>>>     #Handle the error.
```

Usage:

```
>>> disk_info = get_info()
```

CHAPTER

THREE

DOCUMENTATION FOR THE LINUX MODULE

This is the part of the package that contains the tools and information getters for Linux. This would normally be called from the getdevinfo module, but you can call it directly if you like.

Note: You can import this submodule directly, but it might result in strange behaviour, or not work on your platform if you import the wrong one. That is not how the package is intended to be used, except if you want to use the get_block_size() function to get a block size, as documented below.

Warning: Feel free to experiment, but be aware that you may be able to cause crashes, exceptions, and generally weird situations by calling these methods directly if you get it wrong. A good place to look if you're interested in this is the unit tests (in tests/).

Warning: This module won't work properly unless it is executed as root.

`getdevinfo.linux.`**`assemble_lvm_disk_info`**(*line_counter*, *testing=False*)

Private, implementation detail.

This function is used to assemble LVM disk info into the dictionary.

Like get_device_info(), and get_partition_info(), it uses some of the helper functions here.

Args:

line_counter (int): The line in the output that informtion for a particular logical volume begins.

Kwargs: testing (bool): Used during unit tests. Default = False.

Usage:

```
>>> assemble_lvm_disk_info(<anInt>)
```

OR:

```
>>> assemble_lvm_disk_info(<anInt>, testing=<aBool>)
```

`getdevinfo.linux.`**`compute_block_size`**(*stdout*)

Private, implementation detail.

Used to process and tidy up the block size output from blockdev.

Args: stdout (str): blockdev's output.

Returns: int/None: The block size:

- None - Failed!
- int - The block size.

Usage:

```
>>> compute_block_size(<stdoutFromBlockDev>)
```

`getdevinfo.linux.`**`get_block_size`**`(`*disk*`)`

Public

This function uses the blockdev command to get the block size of the given device.

Args:

disk (str): The partition/device/logical volume that we want the block size for.

Returns: int/None. The block size.

- None - Failed!
- int - The block size.

Usage:

```
>>> block_size = get_block_size(<aDeviceName>)
```

`getdevinfo.linux.`**`get_boot_record`**`(`*disk*`)`

Private, implementation detail.

This function gets the MBR/PBR of a given disk.

Args: disk (str): The name of a partition/device.

Returns: tuple (string, string). The boot record (raw, any readable strings):

- (“Unknown”, “Unknown”) - Couldn’t read it.
- Anything else - The PBR/MBR and any readable strings therein.

Usage:

```
>>> boot_record, boot_record_strings = get_boot_record(<aDiskName>)
```

`getdevinfo.linux.`**`get_capabilities`**`(`*node*`)`

Private, implementation detail.

This function gets the capabilities from the structure generated by parsing lshw’s XML output.

Args: node: Represents a device/partition.

Returns: list. The capabilities:

- [] - Couldn’t find them.
- Anything else - The capabilities - as unicode strings.

Usage:

```
>>> capabilities = get_capabilities(<aNode>)
```

`getdevinfo.linux.`**`get_capacity`**`(`*node*`)`

Private, implementation detail.

This function gets the capacity from the structure generated by parsing lshw's XML output. Also rounds it to a human- readable form, and returns both sizes.

Args: node: Represents a device/partition.

Returns: tuple (string, string). The sizes (bytes, human-readable):

- ("Unknown", "Unknown") - Couldn't find them.
- Anything else - The sizes.

Usage:

```
>>> raw_size, human_size = get_capacity(<aNode>)
```

getdevinfo.linux.**get_device_info**(*node*)

Private, implementation detail.

This function gathers and assembles information for devices (whole disks). It employs some simple logic and the other functions defined in this module to do its work.

Args:

node: A "node" representing a device, generated from lshw's XML output.

Returns: string. The name of the device.

Usage:

```
>>> host_disk = get_device_info(<aNode>)
```

getdevinfo.linux.**get_file_system**(*node*)

Private, implementation detail.

This function gets the file system from the structure generated by parsing lshw's XML output.

Args: node: Represents a device/partition.

Returns: string. The file system:

- "Unknown" - Couldn't find it.
- Anything else - The file system.

Usage:

```
>>> file_system = get_file_system(<aNode>)
```

getdevinfo.linux.**get_id**(*disk*)

Private, implementation detail.

This function gets the ID of a given partition or device.

Args: disk (str): The name of a partition/device.

Returns: string. The ID:

- "Unknown" - Couldn't find it.
- Anything else - The ID.

Usage:

```
>>> disk_id = get_id(<aDiskName>)
```

`getdevinfo.linux.`**`get_info`**`()`

This function is the Linux-specific way of getting disk information. It makes use of the lshw, blkid, and lvdisplay commands to gather information.

It uses the other functions in this module to acheive its work, and it **doesn't** return the disk infomation. Instead, it is left as a global attribute in this module (DISKINFO).

Raises: Nothing, hopefully, but errors have a small chance of propagation up to here here. Wrap it in a try:, except: block if you are worried.

Usage:

```
>>> get_info()
```

`getdevinfo.linux.`**`get_lv_aliases`**`(`*line*`)`

Private, implementation detail.

Note: "name" here means path eg /dev/mapper/fedora/root.

This function gets the names of a logical volume. There may be one or more aliases as well as a "default" name. Find and return all of them.

Args: line (int): The line number where the LV name can be found.

Returns: tuple (string, list). The aliases (default_name, all aliases).

Usage:

```
>>> default_name, alias_list = get_lv_aliases(<anLVName>)
```

`getdevinfo.linux.`**`get_lv_and_vg_name`**`(`*volume*`)`

Private, implementation detail.

Note: "name" here means the names of the logical volume and the volume group by themselves. eg volume "root", in volume group "fedora."

This function gets the name of the logical volume (LV), and the name of the volume group (VG) it belongs to.

Args: volume (str): The path for a logical volume.

Returns: tuple (string, string). The VG, and LV name (vg_name, lv_name):

- ("Unknown", "Unknown") - Couldn't find them.
- Anything else - The VG and LV names.

Usage:

```
>>> vg_name, lv_name = get_lv_and_vg_name(<anLVPath>)
```

`getdevinfo.linux.`**`get_lv_file_system`**`(`*disk*`)`

Private, implementation detail.

This function gets the file system of a logical volume.

Args: disk (str): The name of a logical volume.

Returns: string. The file system.

Usage:

```
>>> file_system = get_lv_file_system(<anLVName>)
```

`getdevinfo.linux.`**`get_partition_info`**`(`*subnode*, *host_disk*`)`

Private, implementation detail.

This function gathers and assembles information for partitions. It employs some simple logic and the other functions defined in this module to do its work.

Args:

subnode: A "node" representing a partition, generated from lshw's XML output.

host_disk (str): The "parent" or "host" device. eg: for /dev/sda1, the host disk would be /dev/sda. Used to organise everything nicely in the disk info dictionary.

Returns: string. The name of the partition.

Usage:

```
>>> volume = get_device_info(<aNode>)
```

`getdevinfo.linux.`**`get_partitioning`**`(`*disk*`)`

Private, implementation detail.

This function gets the partition scheme from the structure generated by parsing lshw's XML output.

Args:

disk (str): The name of a device/partition in the disk info dictionary.

Returns: string (str). The partition scheme:

- "Unknown" - Couldn't find it.
- **"mbr" - Old-style MBR partitioning** for BIOS systems.
- "gpt" - New-style GPT partitioning.

Usage:

```
>>> partitioning = get_partitioning(<aDiskName>)
```

`getdevinfo.linux.`**`get_product`**`(`*node*`)`

Private, implementation detail.

This function gets the product from the structure generated by parsing lshw's XML output.

Args: node: Represents a device/partition.

Returns: string. The product:

- "Unknown" - Couldn't find it.
- Anything else - The product.

Usage:

```
>>> product = get_product(<aNode>)
```

`getdevinfo.linux.`**`get_uuid`**`(`*disk*`)`

Private, implementation detail.

This function gets the UUID of a given partition.

Args: disk (str): The name of a **partition**.

Returns: string. The UUID:

- "Unknown" - Couldn't find it.
- Anything else - The UUID.

Usage:

```
>>> uuid = get_uuid(<aPartitionName>)
```

`getdevinfo.linux.`**`get_vendor`**`(`*node*`)`

Private, implementation detail.

This function gets the vendor from the structure generated by parsing lshw's XML output.

Args: node: Represents a device/partition.

Returns: string. The vendor:

- "Unknown" - Couldn't find it.
- Anything else - The vendor.

Usage:

```
>>> vendor = get_vendor(<aNode>)
```

`getdevinfo.linux.`**`parse_lsblk_output`**`()`

Private, implementation detail.

This function is used to get NVME disk information from the output of lsblk.

Note: This will only remain here until lshw adds support for NVME disk detection - this is a temporary fix.

Usage:

```
>>> parse_lsblk_output()
```

`getdevinfo.linux.`**`parse_lvm_output`**`(`*testing=False*`)`

Private, implementation detail.

This function is used to get LVM partition information from the output of lvdisplay –maps.

Kwargs: testing (bool): Used during unit tests. Default = False.

Usage:

```
>>> parse_lvm_output()
```

OR:

```
>>> parse_lvm_output(testing=<aBool>)
```

CHAPTER

FOUR

DOCUMENTATION FOR THE MACOS MODULE

This is the part of the package that contains the tools and information getters for macOS. This would normally be called from the getdevinfo module, but you can call it directly if you like.

Note: You can import this submodule directly, but it might result in strange behaviour, or not work on your platform if you import the wrong one. That is not how the package is intended to be used, except if you want to use the get_block_size() function to get a block size, as documented below.

Warning: Feel free to experiment, but be aware that you may be able to cause crashes, exceptions, and generally weird situations by calling these methods directly if you get it wrong. A good place to look if you're interested in this is the unit tests (in tests/).

Warning: This module won't work properly unless it is executed as root.

`getdevinfo.macos.`**`compute_block_size`**`(`*disk*, *stdout*`)`

Private, implementation detail.

Used to process and tidy up the block size output from diskutil info.

Args: stdout (str): diskutil info's output.

Returns: int/None: The block size:

- None - Failed!
- int - The block size.

Usage:

```
>>> compute_block_size(<stdoutFromDiskutil>)
```

`getdevinfo.macos.`**`get_block_size`**`(`*disk*`)`

Public

This function uses the diskutil info command to get the block size of the given device.

Args:

disk (str): The partition/device that we want the block size for.

Returns: int/None. The block size.

- None - Failed!

- int - The block size.

Usage:

```
>>> block_size = get_block_size(<aDeviceName>)
```

getdevinfo.macos.**get_boot_record**(*disk*)
Not yet implemented, returns ("Unknown", "Unknown").

getdevinfo.macos.**get_capabilities**(*disk*)
Not yet implemented, returns "Unknown"

getdevinfo.macos.**get_capacity**()
Private, implementation detail.

This function gets the capacity of the disk currently referenced in the diskutil info output we're storing. You can't really use this standalone. Also rounds it to a human-readable form, and returns both sizes.

Returns: tuple (string, string). The sizes (bytes, human-readable):

- ("Unknown", "Unknown") - Couldn't find them.
- Anything else - The sizes.

Usage:

```
>>> raw_size, human_size = get_capacity()
```

getdevinfo.macos.**get_description**(*disk*)
Private, implementation detail.

This function generates a human-readable description of the given disk.

Args: disk (str): Name of a device/partition.

Returns:

string. The description: This may contain various bits of info, or not, depending on what macOS knows about the disk.

Usage:

```
>>> description = get_description(<aDisk>)
```

getdevinfo.macos.**get_device_info**(*disk*)
Private, implementation detail.

This function gathers and assembles information for devices (whole disks). It employs some simple logic and the other functions defined in this module to do its work.

Args: disk (str): The name of a device, without the leading /dev. eg: disk1

Returns: string. The name of the device.

Usage:

```
>>> host_disk = get_device_info(<aNode>)
```

getdevinfo.macos.**get_file_system**(*disk*)
Not yet implemented, returns "Unknown".

getdevinfo.macos.**get_id**(*disk*)
Not yet implemented, returns "Unknown".

`getdevinfo.macos.`**`get_info`**`()`

This function is the macOS-specific way of getting disk information. It makes use of the diskutil list, and diskutil info commands to gather information.

It uses the other functions in this module to acheive its work, and it **doesn't** return the disk infomation. Instead, it is left as a global attribute in this module (DISKINFO).

Raises: Nothing, hopefully, but errors have a small chance of propagation up to here here. Wrap it in a try:, except: block if you are worried.

Usage:

```
>>> get_info()
```

`getdevinfo.macos.`**`get_partition_info`**`(`*disk*, *host_disk*`)`

Private, implementation detail.

This function gathers and assembles information for partitions. It employs some simple logic and the other functions defined in this module to do its work.

Args:

disk (str): The name of a partition, without the leading /dev. eg: disk1s1

host_disk (str): The "parent" or "host" device. eg: for /dev/disk1s1, the host disk would be /dev/disk1. Used to organise everything nicely in the disk info dictionary.

Returns: string. The name of the partition.

Usage:

```
>>> volume = get_device_info(<aDisk>, <aHostDisk>)
```

`getdevinfo.macos.`**`get_partitioning`**`(`*disk*`)`

Not yet implemented, returns "Unknown".

`getdevinfo.macos.`**`get_product`**`(`*disk*`)`

Private, implementation detail.

This function gets the product of the given disk.

Args: disk (str): Name of a device/partition.

Returns: string. The product:

- "Unknown" - Couldn't find it.
- Anything else - The product.

Usage:

```
>>> product = get_product(<aDisk>)
```

`getdevinfo.macos.`**`get_uuid`**`(`*disk*`)`

Not yet implemented, returns "Unknown".

`getdevinfo.macos.`**`get_vendor`**`(`*disk*`)`

Private, implementation detail.

This function gets the vendor of the given disk.

Args: disk (str): Name of a device/partition.

Returns: string. The vendor:

- “Unknown” - Couldn’t find it.
- Anything else - The vendor.

Usage:

```
>>> vendor = get_vendor(<aDisk>)
```

`getdevinfo.macos.`**`is_partition`**`(`*disk*`)`

Private, implementation detail.

This function determines if a disk is a partition or not.

Args: disk (str): Name of a device/partition.

Returns: bool:

- True - Is a partition.
- False - Not a partition.

Usage:

```
>>> is_a_partition = is_partition(<aDisk>)
```

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX