

PROOF OF CONCEPT ON STORAGE MONITORING

ABSTRACT

The purpose of this study is to prove the proof of concept on storage monitoring system of container terminals. Proof of concept is a realization of a certain method or idea in order to demonstrate its feasibility. In this concept we use disk space monitoring system using Linux command. The database of Linux command is to display the total disk space in the file system we use “df”, command which stands for Disk file system, is used to check disk space. It will display available and used storage of file system on your machine. When executing this command, you will see the default columns: File System, Size, Used, Available, Use% and Mounted On. In this command “df -h” it will display the result in a human-readable format. We use python script to execute the output in Linux. As a result of the study, it is proved to select monitoring indicators, define external entities, and define internal elements of the system and present indicator result.

INTRODUCTION

This article takes us through various methods which discuss disk space utilization. The server disk management and reporting in a Linux Environment explains the basic elements of disk and captures growth metrics, and explores how to successfully manage and maintain functionality within a Linux environment. There are many choices when it comes to writing scripts on Linux. The bash shell script has been a popular choice, since the bash shell itself is a programming language and the commands used in the script match the syntax, easy access to Linux system commands, and powerful libraries with which to create reports, for example. In addition Python can be used in other applications and on multiple platforms, so there is also the benefit of learning one language for many programming requirements. In this project to display the disk space of the storage monitoring. The simplest way to find the free disk space on Linux is to use df command. The df command stands for disk-free and quite obviously, it shows you the free and available disk space on Linux systems. We use df -h in which -h option, it shows the disk space in human-readable format (MB and GB).

MODULES:

Io – python built-in I/O library, including both abstract classes and some concrete classes such as file I/O.

subprocess – The subprocess module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. This module intends to replace several older modules and functions

Built-in function open() – The standard way to open files for reading and writing with python.

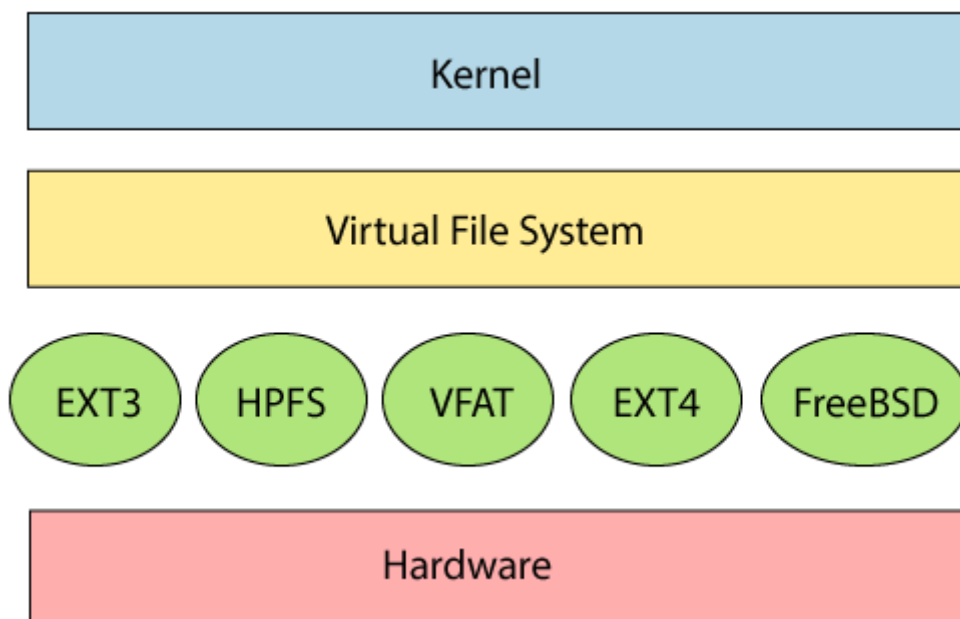
Linux File System Structure

Linux file system has a hierarchal file system structure as it contains a root directory and its subdirectories. All other directories can be accessed from the root directory. A partition usually has only file system, but it may have more than one file system.

The data structure needs to support a hierarchical directory structure; this structure is used to describe the available and used disk space for a particular block. It also has the other details about the files such as file size, date & time of creation, update, and last modified.

Also, it stores advanced information about the section of the disk, such as partitions and volumes.

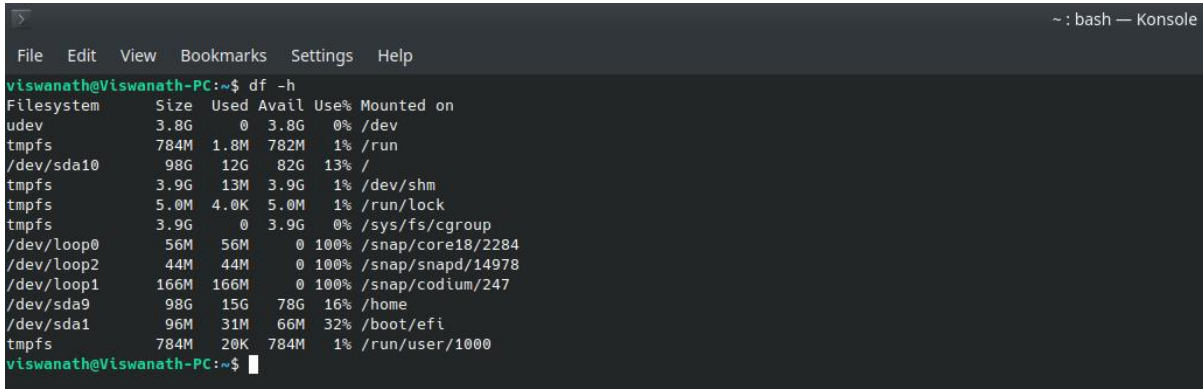
The advanced data and the structures that it represents contain the information about the file system stored on the drive; it is distinct and independent of the file system metadata.



The file system requires an API (Application programming interface) to access the function calls to interact with file system components like files and directories. API facilitates tasks such as creating, deleting and copying the files. It facilitates an algorithm that defines the arrangement of files on a file system.

WORKING PROCEDURE

Use the df command to display file system information about your hard drive. Use the proper options so your output is formatted as follows:



```
viswanath@Viswanath-PC:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            3.8G   0    3.8G   0% /dev
tmpfs           784M  1.8M  782M   1% /run
/dev/sda10      98G   12G   82G   13% /
tmpfs           3.9G  13M   3.9G   1% /dev/shm
tmpfs           5.0M  4.0K   5.0M   1% /run/lock
tmpfs           3.9G   0    3.9G   0% /sys/fs/cgroup
/dev/loop0       56M   56M   0 100% /snap/core18/2284
/dev/loop2       44M   44M   0 100% /snap/snapd/14978
/dev/loop1      166M  166M   0 100% /snap/codium/247
/dev/sda9        98G   15G   78G  16% /home
/dev/sda1        96M   31M   66M  32% /boot/efi
tmpfs           784M  20K   784M   1% /run/user/1000
```

```
>>> import subprocess
```

```
>>> threshold = 90
```

```
>>> child = subprocess.Popen(['df', '-h'], stdout=subprocess.PIPE,
stderr=subprocess.STDOUT)
```

```
>>> output = child.communicate()[0].strip().split(b"\n")
```

```
>>> for x in output[1:]:
```

```
...     if int(x.split()[-2][-1]) >= threshold:
```

```
...         print(x)
```

Output 1:

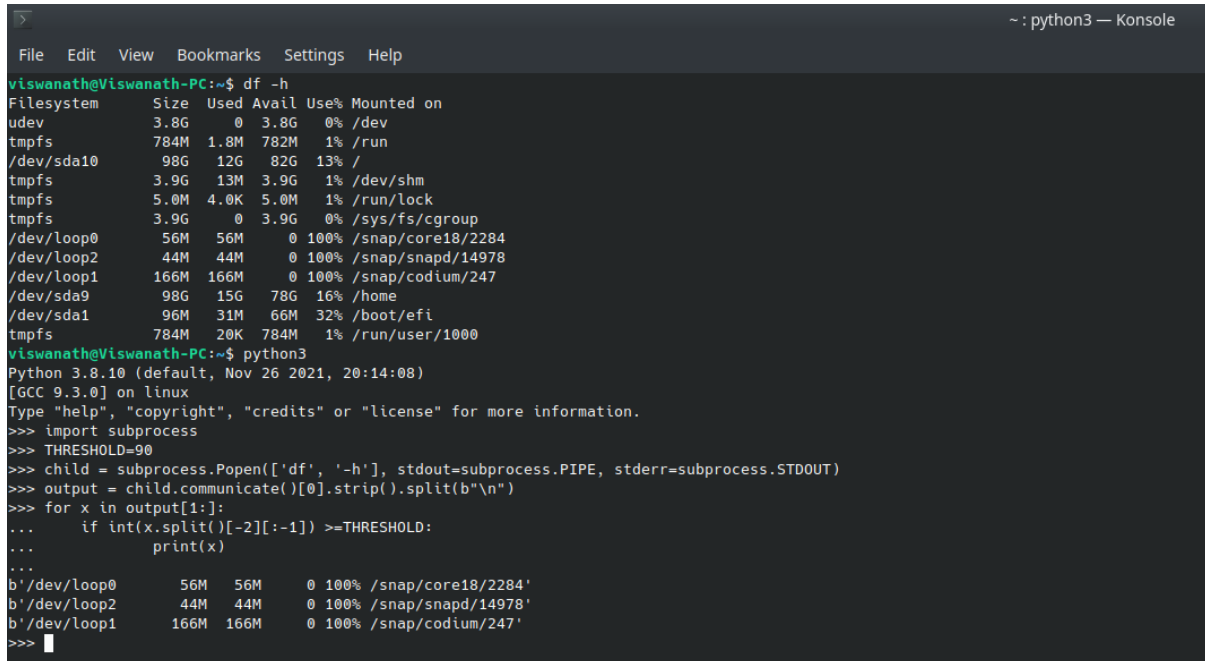
By using this code we will get the output which files has used 90% above threshold value.

```
user3@vm1:~  
login as: user3  
user3@129.153.11.96's password:  
Web console: https://vm1.sub01050807010.vcn1.oraclevcn.com:9090/ or https://10.0.0.168:9090/  
  
Last login: Thu Mar  3 06:52:11 2022 from 117.99.198.152  
[user3@vm1 ~]$ df -h  
Filesystem                Size      Used Avail Use% Mounted on  
devtmpfs                   307M        0   307M   0% /dev  
tmpfs                      342M        0   342M   0% /dev/shm  
tmpfs                      342M      35M   307M  11% /run  
tmpfs                      342M        0   342M   0% /sys/fs/cgroup  
/dev/mapper/ocivolume-root  36G       5.1G    31G  15% /  
/dev/mapper/ocivolume-oled  10G      120M     9.9G   2% /var/oled  
/dev/sda2                  1014M     298M    717M  30% /boot  
/dev/sda1                   100M       5.1M     95M   6% /boot/efi  
tmpfs                      69M        0     69M   0% /run/user/0  
tmpfs                      69M        0     69M   0% /run/user/987  
tmpfs                      69M        0     69M   0% /run/user/1005  
tmpfs                      69M        0     69M   0% /run/user/1001  
tmpfs                      69M        0     69M   0% /run/user/1003  
[user3@vm1 ~]$ python  
Python 3.6.8 (default, Nov 10 2021, 06:50:23)  
[GCC 8.5.0 20210514 (Red Hat 8.5.0-3.0.2)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import subprocess  
>>> threshold = 90  
>>> child = subprocess.Popen(['df', '-h'], stdout=subprocess.PIPE, stderr=subprocess.STDOUT)  
>>> output = child.communicate()[0].strip().split(b"\n")  
>>> for x in output[1:]:  
...     if int(x.split()[-2][:1]) >= threshold:  
...         print(x)  
...  
>>>
```

In our server we didn't have any files that have used 90% above threshold value. So we didn't get any file values.

Output 2:

If we have any files that have used more than 90% threshold value we will get the output as mention in the below screenshot.



```
viswanath@Viswanath-PC:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            3.8G   0    3.8G   0% /dev
tmpfs           784M   1.8M 782M   1% /run
/dev/sda10      98G    12G  82G   13% /
tmpfs           3.9G   13M  3.9G   1% /dev/shm
tmpfs           5.0M   4.0K  5.0M   1% /run/lock
tmpfs           3.9G   0    3.9G   0% /sys/fs/cgroup
/dev/loop0      56M    56M   0 100% /snap/core18/2284
/dev/loop2      44M    44M   0 100% /snap/snapd/14978
/dev/loop1     166M   166M   0 100% /snap/codium/247
/dev/sda9       98G    15G   78G   16% /home
/dev/sda1        96M    31M   66M   32% /boot/efi
tmpfs           784M   20K  784M   1% /run/user/1000

viswanath@Viswanath-PC:~$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import subprocess
>>> THRESHOLD=90
>>> child = subprocess.Popen(['df', '-h'], stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
>>> output = child.communicate()[0].strip().split(b"\n")
>>> for x in output[1:]:
...     if int(x.split()[-2][:-1]) >=THRESHOLD:
...         print(x)
...
b'/dev/loop0      56M    56M   0 100% /snap/core18/2284'
b'/dev/loop2      44M    44M   0 100% /snap/snapd/14978'
b'/dev/loop1     166M   166M   0 100% /snap/codium/247'
```

Conclusion:

We have shown you how to use the df command to get a report of the file system disk space usage. To view all available df command options by typing man df in your terminal.