UEEN 3113 / 3413

SERVER CONFIGURATION AND MANAGEMENT

Storage Management

- To view disk space usage:
  - df -h

- Scenario:
  - Error occurs while creating a file, error message states that the disk is full.
  - However, output from df -h shows that the disk still contain free space!
  - The reason: out of inodes!

Storage Management

- Inode holds all the information about a file except its name and the actual data contents of the file.
- Inode is associated with each file.
- There are limited number of inodes (set when the filesystem is first created), but the number is extremely large and hard to reach.
- To display the usage of inodes:
  - df -i

Storage Management

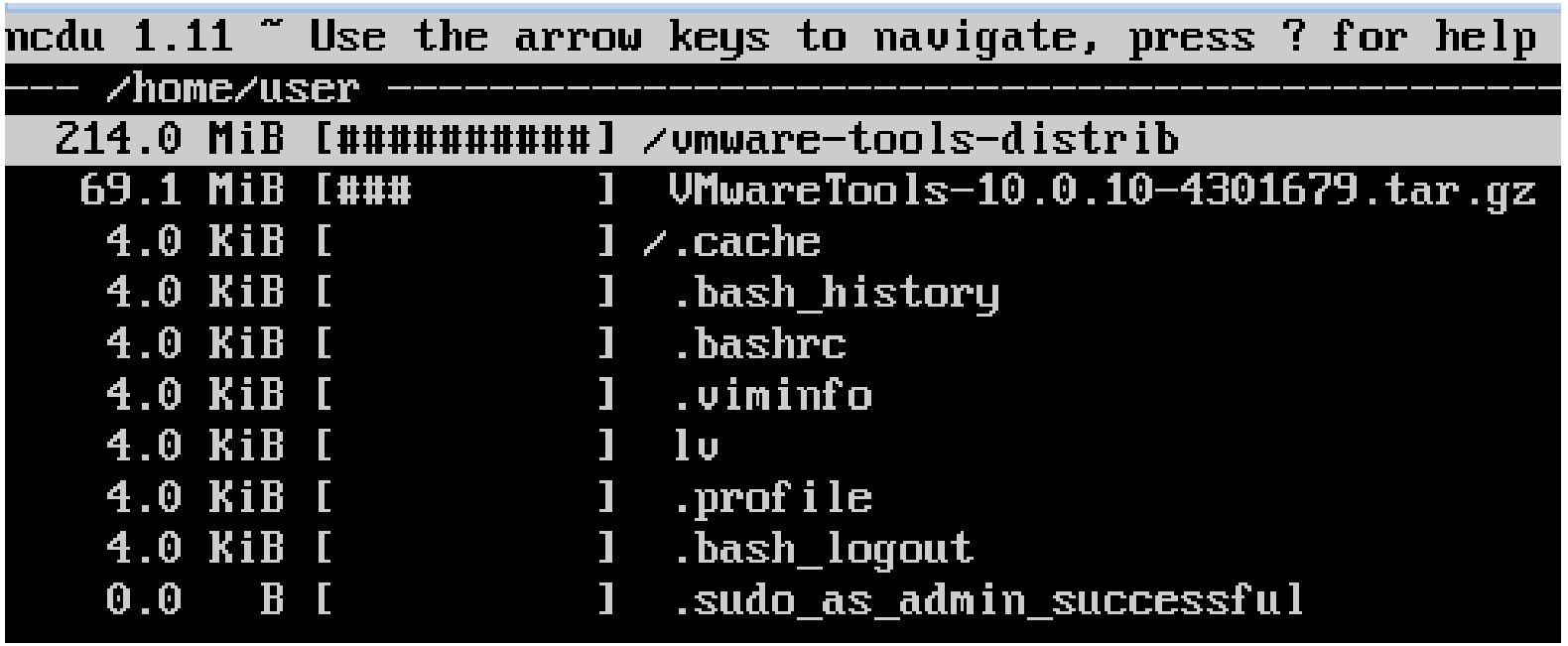
- Created a lots of directories, symlinks, and small files can cause the system runs low on inodes.
- Solutions:
  - Increase the size of volume in LVM (if it is used)
  - Back up and create a new filesystem with higher limit of inodes.
  - Use other filesystem such as btrfs, which uses dynamic inodes.

Storage Management

- In order to know which file or directory that uses a lot of space:
  - du -hsc *
- Run du -hsc * within a directory that's as close as possible to where the problem is.
- The du command is only able to scan directories that its calling user has permission to scan.

Storage Management

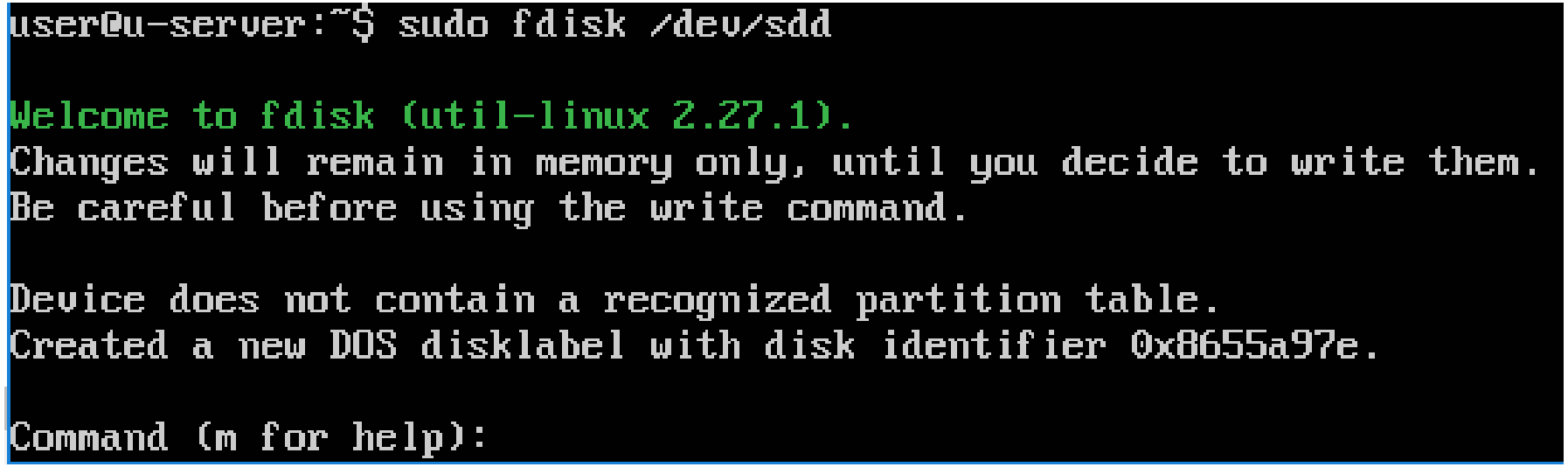
- Another useful application: Ncurses Disk Usage Utility (ncdu), with the ability to traverse the results without having to run a command over and over again.
- Sample output from ncdu



Storage Management

- To view all the disks and the partitions:
  - fdisk -l

- To configure a disk, for example a new disk /dev/sdd:
  - fdisk /dev/sdd



Storage Management

- By default, fdisk will create MBR partition table which has limitations:
  - Maximum 4 primary partitions
  - Maximum disk size can be handled is 2TB

- It’s better to use a newer partition table: GPT partition table.

Storage Management

- In order to create GPT partition table, run command *g* in fdisk prompt

```
  Create a new label
   g   create a new empty GPT partition table
   G   create a new empty SGI (IRIX) partition table
   o   create a new empty DOS partition table
   s   create a new empty Sun partition table


Command (m for help): g
Created a new GPT disklabel (GUID: CD1A620F-AE33-4E15-9959-3519757FCA26).

Command (m for help):
```

Storage Management

- Next, enter *n* in fdisk prompt to add a new partition.

```
Command (m for help): n
Partition number (1-128, default 1): _
```

- Press [Enter] to accept the default partition number.

```
Command (m for help): n
Partition number (1-128, default 1):
First sector (2048-2097118, default 2048):
```

- Press [Enter] again to accept the default first sector.

Storage Management

- fdisk will prompt to enter the last sector.

```
Command (m for help): n
Partition number (1-128, default 1):
First sector (2048-2097118, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097118, default 2097118):
```

- If we press [Enter] to accept the default last sector, the entire free space will used for the new partition.
- If we wish to create a new partition with desired size, enter the size in the prompt.

```
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097118, default 2097118): +500M

Created a new partition 1 of type 'Linux filesystem' and of size 500 MiB.

Command (m for help): _
```

Storage Management

- Enter *w* in fdisk prompt to write the changes and exit fdisk.
- Run fdisk -l to view the new partition.

```
Disk /dev/sdd: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: CD1A620F-AE33-4E15-9959-3519757FCA26

Device     Start     End Sectors  Size Type
/dev/sdd1   2048 1026047 1024000  500M Linux filesystem
```

Storage Management

- After we have created all necessary partitions, we need to format the partitions, using the *mkfs* command.
- Example, to format the partition with ext4 filesystem
  - mkfs.ext4 /dev/sdd1

```
user@u-server:~$ sudo mkfs.ext4 /dev/sdd1
mke2fs 1.42.13 (17-May-2015)
Creating filesystem with 512000 1k blocks and 128016 inodes
Filesystem UUID: b342197a-976e-4b80-97b2-d78a342c908c
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Storage Management

- In order to use the new partition, we need to mount it.
- First, we need decide the mount point. For example, /mnt/data (the data folder in /mnt must be created manually)
  - mount /dev/sdd1 /mnt/data

Storage Management

- To mount disk automatically when the system starts, we need to configure the /etc/fstab file.
- Example of /etc/fstab

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=6665126b-1e10-4b4d-b2f4-7e3b2fc1a873 /               ext4    errors=remount-ro 0       1
# /home was on /dev/sda5 during installation
UUID=da5a7623-50ea-4518-99ec-cf9610918ebc /home           ext4    defaults        0       2
# swap was on /dev/sda6 during installation
UUID=e3e52953-8d32-40a5-b3ec-39df0e03c209 none            swap    sw              0       0

# logical volume for data volume
UUID=e170f8b1-ea13-481f-ae91-e77d3e48acf5 /mnt/lvm/data ext4      errors=remount-ro 0   2
```

Storage Management

- There are 6 columns in /etc/fstab:
  - File system (partition): identified by **UUID** (can be obtained by running *blkid* command)
  - Mount point
  - Type (ext4, ext3, etc.)
  - Options: examples, **errors=remount-ro**, tells the system to remount the filesystem as read-only if an error occurs, **auto**: automatically mount the device at boot time
  - Dump: rarely used, always set to 0 (zero)
  - Pass: the order in which *fsck* will check the file systems
    - 0: never check, 1: check first, 2: check last

Storage Management

- With LVM (Logical Volume Management), we are able to resize filesystems without needing to reboot server.
  - **Volume group** is a namespace given to all the physical and logical volumes on system. Examples: vg-backup
  - **Physical volume** is a physical or virtual hard disk that is a member of a volume group.
  - **Logical volumes** (similar to partitions), can take up a portion of, or an entire disk, but unlike standard partitions, they may span multiple disks. Example, a logical volume include three 100 GB disks and configured to be cumulative total of 300 GB.

Storage Management

- In order to use LVM, make sure the lvm2 package is installed. It can be installed by running
  - apt install lvm2
- We need to configure each disk to be used with lvm, by setting up each one as a physical volume with *pvcreate* command. Examples:
  - pvcreate /dev/sdb
  - pvcreate /dev/sdc

Storage Management

- To list the physical volumes, run the *pvdisplay* command.
- Next, we need to create volume group, with *vgcreate* command.
  - Syntax: vgcreate group_name physical_volume
- Example:
  - vgcreate vg-data /dev/sdb

- Use *vgdisplay* to view all volume groups.

Storage Management

- Then, create logical volume with *lvcreate* command.
  - Syntax: lvcreate -n volume_name -L size volume_group_name
- Example:
  - lvcreate -n datavol -L 3g vg-data

- Use *lvdisplay* command to view all logical volumes.

Storage Management

- In order to format the logical volume, we need to find the full name for logical volume.
  - ls /dev/mapper
  - We should be able to identity the logical volume from the output.
  - Use *mkfs* command to format logical volume.
    - mkfs.ext4 /dev/mapper/logical_volume_name

Storage Management

- Once formatted, it is ready to mount the device.
  - assume that the logical volume is to be mounted at /mnt/lvm/data, which must be created before we mount it
    - mount /dev/mapper/logical_volume_name /mnt/lvm/data

- We might want to edit the /etc/fstab file as well, but we must get the UUID by running blkid.

Storage Management

- To check free space on physical volume, use *pvs* command.
- To extend the size of logical volume, use *lvextend* command with -l or -L option. Examples:
  - extend by taking up the remainder of a physical volume:
    - lvextend -n /dev/mapper/logical_volume_name -l 100%FREE
  - extend 1GB:
    - lvextend -n /dev/mapper/logical_volume_name -L +1g /dev/disk_to_be_used

Storage Management

- If we run the df -h command after *lvextend*, the output shows that the size of the logical volume has not been changed, because the ext4 file system has not been resized.
- We need resize it using *resize2fs* command. Example
  - resize2fs /dev/mapper/logical_volume_name
- Then run the df -h command again to view the updated size.

Storage Management

- We can always add new physical volume to a volume group by using *vgextend* command. Example:
  - vgextend vg-data /dev/sdc

Storage Management

- Another capability of LVM is snapshot.
  - Snapshots allow us to capture a logical volume at certain point in time and preserve it. We can mount a snapshot like other logical volume and even revert the volume group to the snapshot in case something fails.
  - It is useful if you want to test some potentially risky changes to files stored within a volume, but want the insurance that if something goes wrong, you can always undo your changes and go back to how things were.

Storage Management

- Another capability of LVM is snapshot.
  - Snapshots require the volume group to have some unallocated space.
  - We can use snapshots to test how security updates will affect the server if LVM is used for the root file system. If the new updates start to cause problems, we can always revert back. When testing is done, we should merge or remove the snapshot.


OK,
SO WHAT'S
NEXT?