NAME

gunion - control utility for UNION GEOM class

SYNOPSIS

gunion create [**-v**] [**-o** *offset*] [**-s** *size*] [**-S** *secsize*] [**-Z** *gunionname*] *upperdev lowerdev*
gunion destroy [**-fv**] *prov ...*
gunion reset [**-v**] *prov ...*
gunion revert [**-v**] *prov ...*
gunion commit [**-frv**] *prov ...*
gunion list
gunion status
gunion load
gunion unload

DESCRIPTION

The **gunion** utility is used to track changes to a read-only disk on a writable disk. Logically, a writable disk is placed over a read-only disk. Write requests are intercepted and stored on the writable disk. Read requests are first checked to see if they have been written on the top (writable disk) and if found are returned. If they have not been written on the top disk, then they are read from the lower disk.

The **gunion** utility can be especially useful if you have a large disk with a corrupted filesystem that you are unsure of how to repair. You can use **gunion** to place another disk over the corrupted disk and then attempt to repair the filesystem. If the repair fails, you can revert all the changes in the upper disk and be back to the unchanged state of the lower disk thus allowing you to try another approach to repairing it. If the repair is successful you can request that all the writes recorded on the top disk be written to the lower disk.

Another use of the **gunion** utility is to try out upgrades to your system. Place the upper disk over the disk holding your filesystem that is to be upgraded and then run the upgrade on it. If it works, commit it; if it fails, revert the upgrade. An example is given below.

The upper disk must be at least the size of the disk that it covers. The union metadata exists only for the period of time that the union is instantiated, so it is important to commit the updates before destroying the union. If the top disk is about 2.5 percent larger for 512 byte sector disks (or 0.5 percent larger for 4K sector disks) than the disk that it covers, it is posible (thought not currently implemented) to save the union metadata between instantiations of the union device.

If you do not have physical media available to use for the upper layer, the md(4) disk can be used instead. When used in **swap** mode the changes are all held in buffer memory. Pages get pushed out to the swap when the system is under memory pressure, otherwise they stay in the operating memory. If

long-term persistance is desired, **vnode** mode can be used in which a regular file is used as backing store. The disk space used by the file is based on the amount of data that is written to the top device.

The first argument to **gunion** indicates an action to be performed:

create Set up a union provider on the two given devices. The first device given is used as the top device and must be writable. The second device given is used as the bottom device and need only be readable. The second device may be mounted read-only but it is recommended that it be unmounted and accessed only through a mount of the union device. If the operation succeeds, the new provider should appear with name */dev/<upperdev>-<lowerdev>.union*. An alternate name can be specified with the **-Z** flag. The kernel module *geom_union.ko* will be loaded if it is not loaded already.

Additional options include:

-o *offset* Where to begin on the original provider. The default is to start at the beginning of the disk (i.e., at offset 0). This option may be used to skip over partitioning information stored at the beginning of a disk. The offset must be a multiple of the sector size.

-s *size* Size of the transparent provider. The default is to be the same size as the lower disk. Any extra space at the end of the upper disk may be used to store union metadata.

-S *secsize* Sector size of the transparent provider. The default is to be the same sector size as the lower disk.

-v Be more verbose.

-Z *gunionname* The name of the new provider. The suffix ".union" will be appended to the provider name.

destroy Turn off the given union providers.

Additional options include:

-f Force the removal of the specified provider.

-v
Be more verbose.

revert Discard all the changes made in the top layer thus reverting to the original state of the lower device. The union device may not be mounted or otherwise in use when a **revert** operation is being done.

commit

Write all the changes made in the top device to the lower device thus committing the lower device to have the same data as the union.

Additional options include:

-f The **commit** command will not allow the lower device to be mounted or otherwise in use while the **commit** operation is being done. However, the **-f** flag may be specified to allow the lower device to be mounted read-only. To prevent a filesystem panic on the mounted lower-device filesystem, immediately after the **commit** operation finishes the lower-device filesystem should be unmounted and then remounted to update its metadata state. If the lower-device filesystem is currently being used as the root filesystem then the **-r** flag should be specified to reboot the system at the completion of the **commit** operation.

-r Reboot the system at the completion of the **commit** operation.

-v

Be more verbose.

reset Reset statistics for the given union providers.

list See geom(8).

status See geom(8).

load See geom(8).

unload See geom(8).

EXIT STATUS

Exit status is 0 on success, and 1 if the command fails.

EXAMPLES

The following example shows how to create and destroy a union provider with disks */dev/da0p1* as the read-only disk on the bottom and */dev/md0* as the wriable disk on the top.

```
gunion create -v md0 da0p1
mount /dev/md0-da0p1.union /mnt
```

Proceed to make changes in /mnt filesystem. If they are successful and you want to keep them.

```
umount /mnt
gunion commit -v md0-da0p1.union
```

If they are unsuccessful and you want to roll back.

```
umount /mnt
gunion revert -v md0-da0p1.union
```

When done eliminate the union.

```
umount /mnt
gunion destroy -v md0-da0p1.union
```

All uncommitted changes will be discarded when the union is destroyed.

If you use the name of the full disk, for example *da0* and it is labelled, then a union name will appear for the disk as *md0-da0.union* as well as for each partition on the disk as *md0-da0p1.union*, *md0-da0p2.union*, etc. A commit operation can be done only on *md0-da0.union* and will commit changes to all the partitions. If partition level commits are desired, then a union must be created for each partition.

The traffic statistics for the given union providers can be obtained with the **list** command. The example below shows the number of bytes written with newfs(8):

```
gunion create md0 da0p1
newfs /dev/md0-da0p1.union
gunion list
```

SYSCTL VARIABLES

The following sysctl(8) variables can be used to control the behavior of the **UNION** GEOM class. The default value is shown next to each variable.

kern.geom.union.debug: 0

Debug level of the **UNION** GEOM class. This can be set to a number between 0 and 4 inclusive. If set to 0, no debug information is printed. If set to 1, all the verbose messages are logged. If

set to 2, addition error-related information is logged. If set to 3, mapping operations are logged. If set to 4, the maximum amount of debug information is printed.

SEE ALSO

geom(4), geom(8)

HISTORY

The **gunion** utility appeared in FreeBSD 14.0.

AUTHORS

Marshall Kirk McKusick <*mckusick@mckusick.com*>