# Standard RAID levels

From Wikipedia, the free encyclopedia

The **standard RAID levels** are a basic set of RAID configurations and employ striping, mirroring, or parity. The standard RAID levels can be nested for other benefits (*see Nested RAID levels*).

## Contents



## Concatenation (SPAN)

The controller treats each drive as a stand-alone disk, therefore each drive is an independent logical drive. Concatenation does not provide data redundancy.

**Concatenation** or Spanning of disks is not one of the numbered RAID levels, but it is a popular method for combining multiple physical disk drives into a single virtual disk. It provides no data redundancy. As the name implies, disks are merely concatenated together, end to beginning, so they appear to be a single large disk.

Concatenation may be thought of as the reverse of partitioning. Whereas partitioning takes

one physical drive and creates two or more logical drives, concatenation uses two or more physical drives to create one logical drive.

In that it consists of an array of independent disks, it can be thought of as a distant relative of RAID. Concatenation is sometimes used to turn several odd-sized drives into one larger useful drive, which cannot be done with RAID 0. For example, JBOD could combine 3 GB, 15 GB, 5.5 GB, and 12 GB drives into a logical drive at 35.5 GB, which is often more useful than the individual drives separately.

In the diagram to the right, data are concatenated from the end of disk 0 (block A63) to the beginning of disk 1 (block A64); end of disk 1 (block A91) to the beginning of disk 2 (block A92). If RAID 0 were used, then disk 0 and disk 2 would be truncated to 28 blocks, the size of the smallest disk in the array (disk 1) for a total size of 84 blocks.

Some RAID controllers use JBOD to refer to configuring drives without RAID features. Each drive shows up separately in the OS. This JBOD is not the same as concatenation.

Many Linux distributions use the terms "linear mode" or "append mode". The Mac OS X 10.4 implementation — called a "Concatenated Disk Set" — does not leave the user with any usable data on the remaining drives if one drive fails in a concatenated disk set, although the disks otherwise operate as described above.

Concatenation is one of the uses of the Logical Volume Manager in Linux, which can be used to create virtual drives spanning multiple physical drives and/or partitions.

## RAID 0

A **RAID 0** (also known as a **stripe set** or **striped volume**) splits data evenly across two or more disks (striped) with no parity information for redundancy. It is important to note that RAID 0 was not one of the original RAID levels and provides zero data redundancy. RAID 0 is normally used to increase performance, although it can also be used as a way to create a small number of large virtual disks out of a large number of small physical ones.

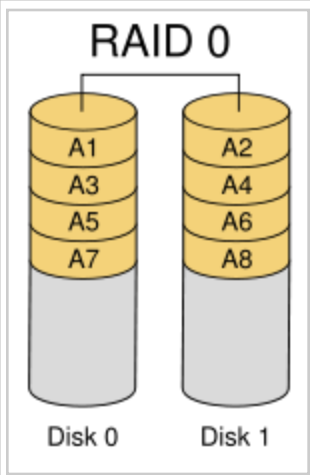

Diagram of a RAID 0 setup.

A RAID 0 can be created with disks of differing sizes, but the storage space added to the array by each disk is limited to the size of the smallest disk. For example, if a 120 GB disk is striped together with a 100 GB disk, the size of the array will be 200 GB.

$$\begin{aligned}\text{Size} &= 2 \cdot \min(120\,\text{GB}, 100\,\text{GB}) \\ &= 2 \cdot 100\,\text{GB} \\ &= 200\,\text{GB}\end{aligned}$$

In the diagram to the right, the odd blocks are written to disk 0 while the even blocks are written to disk 1 such that A1, A2, A3, A4, ... would be the order of blocks read if read sequentially from the beginning.

### RAID 0 failure rate

Although RAID 0 was not specified in the original RAID paper, an idealized implementation

of RAID 0 would split I/O operations into equal-sized blocks and spread them evenly across two disks. RAID 0 implementations with more than two disks are also possible, though the group reliability decreases with member size.

Reliability of a given RAID 0 set is equal to the average reliability of each disk divided by the number of disks in the set:

$$\mathrm{MTTF}_{\mathrm{group}} \approx \frac{\mathrm{MTTF}_{\mathrm{disk}}}{\mathrm{number}}$$

That is, reliability (as measured by mean time to failure (MTTF) or mean time between failures (MTBF) is roughly inversely proportional to the number of members — so a set of two disks is roughly half as reliable as a single disk. If there were a probability of 5% that the disk would fail within three years, in a two disk array, that probability would be upped to $Pr(atleastonefails) = 1 - Pr(neitherfails) = 1 - (1 - 0.05)^2 = 0.0975 = 9.75\,\%$.

The reason for this is that the file system is distributed across all disks. When a drive fails the file system cannot cope with such a large loss of data and coherency since the data is "striped" across all drives (the data cannot be recovered without the missing disk). Data can be recovered using special tools (see *data recovery*), however, this data will be incomplete and most likely corrupt, and recovery of drive data is very costly and not guaranteed.

### RAID 0 performance

While the block size can technically be as small as a byte, it is almost always a multiple of the hard disk sector size of 512 bytes. This lets each drive seek independently when randomly reading or writing data on the disk. How much the drives act independently depends on the access pattern from the file system level. For reads and writes that are larger than the stripe size, such as copying files or video playback, the disks will be seeking to the same position on each disk, so the seek time of the array will be the same as that of a single drive. For reads and writes that are smaller than the stripe size, such as database access, the drives will be able to seek independently. If the sectors accessed are spread evenly between the two drives, the apparent seek time of the array will be half that of a single drive (assuming the disks in the array have identical access time characteristics). The transfer speed of the array will be the transfer speed of all the disks added together, limited only by the speed of the RAID controller. Note that these performance scenarios are in the best case with optimal access patterns.

RAID 0 is useful for setups such as large read-only NFS servers where mounting many disks is time-consuming or impossible and redundancy is irrelevant.

RAID 0 is also used in some gaming systems where performance is desired and data integrity is not very important. However, real-world tests with games have shown that RAID-0 performance gains are minimal, although some desktop applications will benefit. [1][2] Another article examined these claims and concludes that "Striping does not always increase performance (in certain situations it will actually be slower than a non-RAID setup), but in most situations it will yield a significant improvement in performance." [3]

## RAID 1

A **RAID 1** creates an exact copy (or **mirror**) of a set of data on two or more disks. This is useful when read performance or reliability are more important than data storage capacity. Such an array can only be as big as the smallest member disk. A classic RAID 1 mirrored pair contains two disks (see diagram), which increases reliability geometrically over a single disk. Since each member contains a complete copy of the data, and can be addressed independently, ordinary wear-and-tear reliability is raised by the power of the number of self-contained copies.

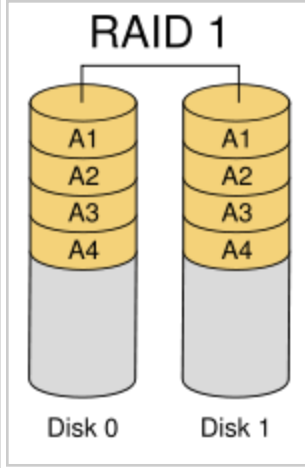

Diagram of a RAID 1 setup.

## RAID 1 failure rate

For trivial example, consider a RAID 1 with 2 identical models of a disk drive with a 5% probability that the disk would fail within three years. Provided that the failures are statistically independent, then the probability of both drives failing (with no replacements for 3 years) is

$$P(bothfail) = (0.05)^2 = 0.0025 = 0.25\,\%.$$

## RAID 1 performance

Additionally, since all the data exists in two or more copies, each with its own hardware, the read performance can go up roughly as a linear multiple of the number of copies. That is, a RAID 1 array of two drives can be reading in two different places at the same time, though not all implementations of RAID 1 do this.[4] To maximize performance benefits of RAID 1, independent disk controllers are recommended, one for each disk. Some refer to this practice as **splitting** or **duplexing.** When reading, both disks can be accessed independently and requested sectors can be split evenly between the disks. For the usual mirror of two disks, this would, in theory, double the transfer rate when reading. The apparent access time of the array would be half that of a single drive. Unlike RAID 0, this would be for all access patterns, as all the data are present on all the disks. In reality, the need to move the drive heads to the next block (to skip unread blocks) can effectively mitigate speed advantages for sequential access. Read performance can be further improved by adding drives to the mirror. Many older IDE RAID 1 controllers read only from one disk in the pair, so their read performance is always that of a single disk. Some older RAID 1 implementations would also read both disks simultaneously and compare the data to detect errors. The error detection and correction on modern disks makes this less useful in environments requiring normal availability. When writing, the array performs like a single disk, as all mirrors must be written with the data. Note that these performance scenarios are in the best case with optimal access patterns.

RAID 1 has many administrative advantages. For instance, in some environments, it is possible to "split the mirror": declare one disk as inactive, do a backup of that disk, and then "rebuild" the mirror. This is useful in situations where the file system must be constantly available. This requires that the application supports recovery from the image of data on the disk at the point of the mirror split. This procedure is less critical in the presence of the "snapshot" feature of some file systems, in which some space is reserved for changes, presenting a static point-in-time view of the file system. Alternatively, a new disk can be substituted so that the inactive disk can be kept in much the same way as traditional backup. To keep redundancy during the backup process, some controllers support adding a

third disk to an active pair. After a rebuild to the third disk completes, it is made inactive and backed up as described above.

## RAID 2

A **RAID 2** stripes data at the bit (rather than block) level, and uses a Hamming code for error correction. The disks are synchronized by the controller to spin in perfect tandem. Extremely high data transfer rates are possible. This is the only original level of RAID that is not currently used.

The use of the Hamming(7,4) code (four data bits plus three parity bits) also permits using 7 disks in RAID 2, with 4 being used for data storage and 3 being used for error correction.

RAID 2 is the only standard RAID level, other than some implementations of RAID-6, which can automatically recover accurate data from single-bit corruption in data. Other RAID levels can detect single-bit corruption in data, or can sometimes reconstruct missing data, but cannot reliably resolve contradictions between parity bits and data bits without human intervention.

(Multiple-bit corruption is possible though extremely rare. RAID 2 can detect but not repair double-bit corruption.)

At the present time, there are no commercial implementations of RAID-2.

## RAID 3

A **RAID 3** uses byte-level striping with a dedicated parity disk. RAID 3 is very rare in practice. One of the side-effects of RAID 3 is that it generally cannot service multiple requests simultaneously. This comes about because any single block of data will, by definition, be spread across all members of the set and will reside in the same location. So, any I/O operation requires activity on every disk and usually requires synchronized spindles.

In our example, a request for block "A" consisting of bytes A1-A6 would require all three data disks to seek to the beginning (A1) and reply with their contents. A simultaneous request for block B would have to wait.

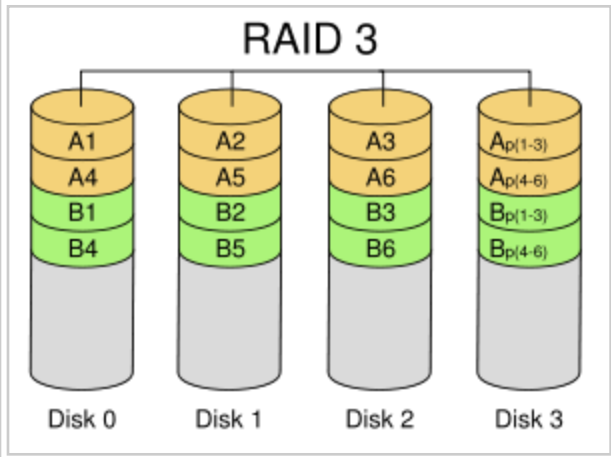


Diagram of a RAID 3 setup of 6-byte blocks and two parity bytes, shown are two blocks of data (orange and green)

## RAID 4

A **RAID 4** uses block-level striping with a dedicated parity disk. This allows each member of the set to act independently when only a single block is requested. If the disk controller allows it, a RAID 4 set can service multiple read requests simultaneously. RAID 4 looks similar to RAID 5 except that it does not use distributed parity, and similar to RAID 3 except that it stripes at the block level, rather than the byte level. Generally, RAID 4 is implemented with hardware support for parity calculations, and a minimum of 3 disks is required for a complete RAID 4 configuration.

In the example on the right, a read request for block "A1" would be serviced by disk 0. A simultaneous read request for block B1 would have to wait, but a read request for B2 could be serviced concurrently by disk 1.

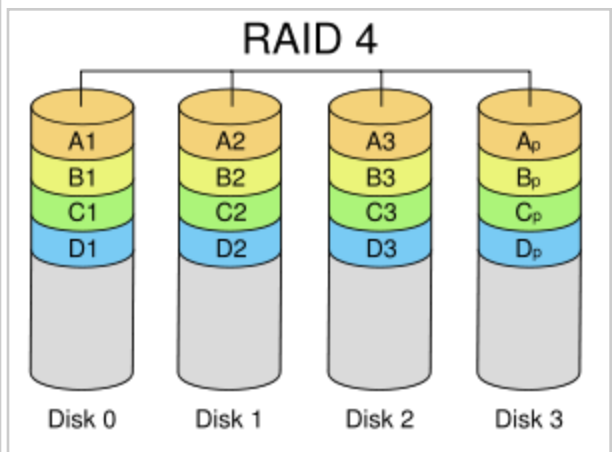


Diagram of a RAID 4 setup with dedicated parity disk with each color representing the group of blocks in the respective parity block (a stripe)

## RAID 5

A **RAID 5** uses block-level striping with parity data distributed across all member disks. RAID 5 has achieved popularity due to its low cost of redundancy. This can be seen by comparing the number of drives needed to achieve a given capacity. RAID 1 or RAID 0+1, which yield redundancy, give only s/2 storage capacity, where s is the sum of the capacities of n drives used. In RAID 5, the yield is s * (n - 1)/n. Using 1 TB drives as an example, four of them can build a 2 TB redundant array under RAID 1 or RAID 1+0, but they can be used to build a 3 TB array under RAID 5. Although RAID 5 is commonly implemented in a disk controller, some with hardware support for parity calculations (hardware raid cards) and some using the main system processor (motherboard based raid controllers), it can also be done at the operating system level, e.g. using Windows "Dynamic Disks" or with mdadm in Linux. A minimum of 3 disks is required for a complete RAID 5 configuration. In some implementations a degraded RAID 5 disk set can be made (3 disk set of which only 2 are online), while mdadm supports a fully-functional (non-degraded) RAID 5 setup with two disks - which function as a slow RAID-1, but can be expanded with further volumes.

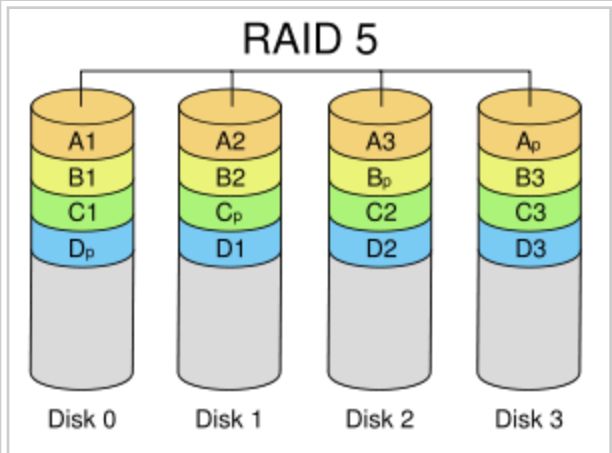


Diagram of a RAID 5 setup with distributed parity with each color representing the group of blocks in the respective parity block (a stripe). This diagram shows left asymmetric algorithm

In the example on the right, a read request for block "A1" would be serviced by disk 0. A simultaneous read request for block B1 would have to wait, but a read request for B2 could be serviced concurrently by disk 1.

## RAID 5 parity handling

A concurrent series of blocks (one on each of the disks in an array) is collectively called a "stripe". If another block, or some portion of a block, is written on that same stripe, the parity block (or some portion of the parity block) is recalculated and rewritten. For small writes, this requires:

- Reading the old data block
- Reading the old parity block
- Comparing the old data block with the write request. For each bit that has flipped (changed from 0 to 1, or from 1 to 0) in the data block, flipping the corresponding bit in the parity block
- Writing the new data block
- Writing the new parity block

The disk used for the parity block is staggered from one stripe to the next, hence the term "distributed parity blocks". RAID 5 writes are expensive in terms of disk operations and traffic between the disks and the controller.

The parity blocks are not read on data reads, since this would be unnecessary overhead and would diminish performance. The parity blocks are read, however, when a read of blocks in the stripe and within the parity block in the stripe are used to reconstruct the errant sector. The CRC error is thus hidden from the main computer. Likewise, should a disk fail in the array, the parity blocks from the surviving disks are combined mathematically with the data blocks from the surviving disks to reconstruct the data on the failed drive "on the fly".

This is sometimes called Interim Data Recovery Mode. The computer knows that a disk drive has failed, but this is only so that the operating system can notify the administrator that a drive needs replacement; applications running on the computer are unaware of the failure. Reading and writing to the drive array continues seamlessly, though with some performance degradation. The difference between RAID 4 and RAID 5 is that in interim data recovery mode, RAID 5 might be slightly faster than RAID 4: When the CRC and parity are in the disk that failed, the calculation does not have to be performed, while with RAID 4, if one of the data disks fails, the calculations have to be performed with each access.

## RAID 5 disk failure rate

The maximum number of drives in a RAID 5 redundancy group is theoretically unlimited, but it is common practice to limit the number of drives. The tradeoffs of larger redundancy groups are greater probability of a simultaneous double disk failure, the increased time to rebuild a redundancy group, and the greater probability of encountering an unrecoverable sector during RAID reconstruction. As the number of disks in a RAID 5 group increases, the Mean Time Between Failures (MTBF, the reciprocal of the failure rate) can become lower than that of a single disk. This happens when the likelihood of a second disk failing out of (N-1) dependent disks, within the time it takes to detect, replace and recreate a first failed disk, becomes larger than the likelihood of a single disk failing. RAID 6 is an alternative that provides dual parity protection thus enabling larger numbers of disks per RAID group. Some implementations have a hot spare disk that can be used to immediately rebuild the failed drive.

Some RAID vendors will avoid placing disks from the same manufacturing lot in a

redundancy group to minimize the odds of simultaneous early life and end of life failures as evidenced by the Bathtub curve.

### RAID 5 performance

RAID 5 implementations suffer from poor performance when faced with a workload which includes many writes which are smaller than the capacity of a single stripe; this is because parity must be updated on each write, requiring read-modify-write sequences for both the data block and the parity block. More complex implementations may include non-volatile write back cache to reduce the performance impact of incremental parity updates.

Random write performance is poor, especially at high concurrency levels common in large multi-user databases. The read-modify-write cycle requirement of RAID 5's parity implementation penalizes random writes by as much as an order of magnitude compared to RAID 0.[5]

Performance problems can be so severe that some database experts have formed a group called BAARF — the Battle Against Any Raid Five.[6]

The read performance of RAID 5 is almost as good as RAID 0 for the same number of disks. Except for the parity blocks, the distribution of data over the drives follows the same pattern as RAID 0. The reason RAID 5 is slightly slower is that the disks must skip over the parity blocks.

In the event of a system failure while there are active writes, the parity of a stripe may become inconsistent with the data. If this is not detected and repaired before a disk or block fails, data loss may ensue as incorrect parity will be used to reconstruct the missing block in that stripe. This potential vulnerability is sometimes known as the "write hole". Battery-backed cache and similar techniques are commonly used to reduce the window of opportunity for this to occur.

### RAID 5 usable size

Parity data use up the capacity of one drive in the array (this can be seen by comparing it with RAID 4: RAID 5 distributes the parity data across the disks, while RAID 4 centralizes it on one disk, but the amount of parity data is the same). In case that the drives vary in capacity, the smallest of them sets the bar. Therefore, the usable capacity of a RAID 5 array is $(N-1) \cdot S_{\min}$, where $N$ is the total number of drives in the array and $S_{\min}$ is the capacity of the smallest drive in the array..

The number of hard drives that can belong to a single array is theoretically unlimited. The time required for initial construction of the array as well as that for reconstruction of a failed disk increases with the number of drives in an array.

## RAID 6

**RAID 6** extends RAID 5 by adding an additional parity block, thus it uses block-level striping with two parity blocks distributed across all member disks. It was not one of the original RAID levels.

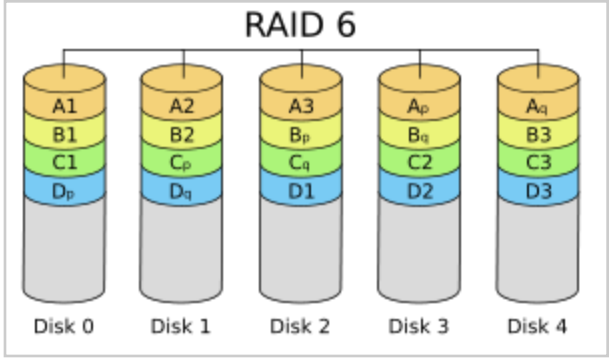

Diagram of a RAID 6 setup, which is identical to RAID 5 other than the addition of a second parity block

RAID 5 can be seen as a special case of a Reed-Solomon code.[7] RAID 5, being a degenerate case, requires only addition in the Galois field. Since we are operating on bits, the field used is a binary galois field **GF**(2). In cyclic representations of binary galois fields, addition is computed by a simple XOR.

After understanding RAID 5 as a special case of a Reed-Solomon code, it is easy to see that it is possible to extend the approach to produce redundancy simply by producing another syndrome; typically a polynomial in **GF**($2^8$) (8 means we are operating on bytes). By adding additional syndromes it is possible to achieve any number of redundant disks, and recover from the failure of that many drives anywhere in the array, but RAID 6 refers to the specific case of two syndromes.

Like RAID 5, the parity is distributed in stripes, with the parity blocks in a different place in each stripe.

## RAID 6 performance

RAID 6 is no more space inefficient than RAID 5 with a hot spare drive when used with a small number of drives, but as arrays become bigger and have more drives the loss in storage capacity becomes less important and the probability of data loss is greater. RAID 6 provides protection against data loss during an array rebuild; when a second drive is lost, a bad block read is encountered, and the more often occurrence of removal and replacement of the wrong disk drive.

The usable capacity of a RAID 6 array is $(N-2)\cdot S_{\min}$, where *N* is the total number of drives in the array and $S_{\min}$ is the capacity of the smallest drive in the array.

RAID 6 does not have a performance penalty for read operations, but it does have a performance penalty on write operations due to the overhead associated with parity calculations. Performance varies greatly depending on how RAID 6 is implemented in the manufacturer's storage architecture - in software, firmware or by using firmware and specialized ASICs for intensive parity calculations. It can be as fast as RAID 5 with one fewer drives (same number of data drives.)

## RAID 6 implementation

According to SNIA (Storage Networking Industry Association), the definition of RAID 6 is: "Any form of RAID that can continue to execute read and write requests to all of a RAID array's virtual disks in the presence of any two concurrent disk failures. Several methods, including dual check data computations (parity and Reed Solomon), orthogonal dual parity check data and diagonal parity have been used to implement RAID Level 6."[8]

## Non-standard RAID levels

There are other RAID levels that are promoted by individual vendors, but not generally standardized. The non-standard RAID levels 5E, 5EE and 6E extend RAID 5 and 6 with hot-spare drives.

Other non-standard RAID levels include: RAID 1.5, RAID 7, RAID-DP, RAID S or Parity RAID, Matrix RAID, RAID-K, RAID-Z, RAIDn, Linux MD RAID 10, IBM ServeRAID 1E, and unRAID.

## JBOD

JBOD stands for Just a Bunch Of Disks (Just a Box Of Drives). Depending on the Host Bus Adapter a JBOD can be used as individual disks or any RAID configuration supported by the HBA.

## See also

- RAID
- Nested RAID levels
- Non-standard RAID levels

## References


1. ^ "Western Digital's Raptors in RAID-0: Are two drives better than one? (http://www.anandtech.com/storage/showdoc.aspx?i=2101) ". AnandTech (July 1, 2004). Retrieved on 2007-11-24.
2. ^ "Hitachi Deskstar 7K1000: Two Terabyte RAID Redux (http://www.anandtech.com/storage/showdoc.aspx?i=2974) ". AnandTech (April 23, 2007). Retrieved on 2007-11-24.
3. ^ "RAID 0: Hype or blessing? (http://tweakers.net/reviews/515/1/raid-0-hype-or-blessing-pagina-1.html) ". Tweakers.net (August 7, 2004). Retrieved on 2008-07-23.
4. ^ "Mac OS X, Mac OS X Server: How to Use Apple-Supplied RAID Software (http://docs.info.apple.com/article.html?artnum=106594) ". Apple.com. Retrieved on 2007-11-24.
5. ^ Cary Millsap (21 August 1996). "*Configuring Oracle Server for VLDB (http://oreilly.com/catalog/oressentials2/chapter/vldb1.pdf)* " (PDF).
6. ^ "BAARF - Battle Against Any Raid Five (http://www.baarf.com) ". Retrieved on 2008-07-11.
7. ^ H. Peter Anvin (24 October 2007). "*The mathematics of RAID-6 (http://kernel.org/pub/linux/kernel/people/hpa/raid6.pdf)* " (PDF).
8. ^ "Dictionary R (http://www.snia.org/education/dictionary/r/) ". Storage Networking Industry Association. Retrieved on 2007-11-24.


## External links

- Tutorial on "RAID 6 Essentials — reduced performace or not?" (http://www.brainshark.com/winchestersystemsinc1/vu?pi=707421860)
- IBM summary on RAID levels (http://www-1.ibm.com/support/docview.wss?uid=swg21149421)
- RAID 5 Parity explanation and checking tool. (http://www.dtidata.com/resourcecenter/2008/05/08/raid-configuration-parity-check/)
- Dell animations and details on RAID levels 0, 1, 5 (http://support.dell.com/support/topics/global.aspx/support/entvideos/raid?c=us&l=en&s=gen)