Knowledge Base

An Explanation of CHKDSK and the New /C and /I Switches

PSS ID Number: 187941

Article Last Modified on 5/12/2003

The information in this article applies to:

- Microsoft Windows NT Server 4.0 Terminal Server Edition
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Professional
- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows NT Workstation 4.0 SP4
- Microsoft Windows NT Server 4.0 SP4

This article was previously published under Q187941
For a Microsoft Windows XP version of this article, see 314835.

SUMMARY

In Windows NT version 4.0 Service Pack 4 (SP4) and Windows 2000, two new switches have been added to Chkdsk.exe. These switches enable users to better manage downtime incurred by running CHKDSK or AUTOCHK.

The switches that are added in Windows NT 4.0 SP4 and Windows 2000 are /C and /I, and are only valid when the target drive has the NTFS format. Each switch directs the CHKDSK routine to bypass certain actions it would otherwise take to validate the integrity of NTFS data structures.

MORE INFORMATION

Chkdsk.exe is the command-line interface for a program that verifies the logical integrity of a file system on Windows. When CHKDSK encounters logical inconsistencies it takes actions to repair file system data, provided it is not in read-only mode.

The code that actually performs the verification when CHKDSK is run online resides in utility DLLs such as Untfs.dll and Ufat.dll. The verification routines invoked by Chkdsk.exe are the same ones invoked when a volume is verified through the graphical user interface provided by the Windows Explorer or Disk Administrator. When CHKDSK is scheduled to run at reboot, on the other hand, the binary module that contains the verification code is Autochk.exe. Autochk.exe is a native Windows application that runs early enough in the system boot sequence that it does not have the benefit of Virtual Memory or other Win32 services. Autochk.exe generates the same kind of textual output that the utility DLLs invoked by Chkdsk.exe does. But in addition to displaying this output on the screen during the boot process, Autochk.exe also logs an event to the Application Event Log for the system containing as much of the textual output as can fit into the event log's data buffer.

Because Autochk.exe and the verification code in the utility DLLs used by Chkdsk.exe are based on the same source code, both will be referred to generically as "CHKDSK" throughout the remainder of this article. Likewise, as this article is concerned only with changes in CHKDSK behavior with respect to NTFS volumes, it should be understood that, by saying, "CHKDSK does such-and-such," the following is meant: "CHKDSK does such-and-such when run on an NTFS volume".

Because the use of the /C and /I switches can result in a volume remaining corrupted even after CHKDSK completes, the use of these switches is not recommended except in situations where system downtime must be kept to a minimum. These switches are intended to be used by users with exceptionally large volumes and who require flexibility in managing the downtime that is incurred when CHKDSK must be run on such volumes.

To understand when it might be appropriate to use these switches, it is important to have a basic understanding of some of the internal NTFS data structures, the kinds of corruption that can take place, what actions CHKDSK takes when it verifies a volume, and what the potential consequences are in circumventing CHKDSK's usual verification steps.

CHKDSK's activity is split into three major "passes" during which it examines all the "metadata" on the volume and an optional fourth pass. Metadata is "data about data." It is the file system overhead, so to speak, that is used to keep track of everything about all of the files on the volume. Metadata tells what allocation units make up the data for a given file, what allocation units are free, what allocation units contain bad sectors, and so on. The "contents" of a file, on the other hand, is termed "user data." NTFS protects its metadata through the use of a transaction log. User data is not so protected.

During its first pass, CHKDSK displays a message on the screen saying that it is verifying files and counts from 0 to 100 percent complete. During this phase, CHKDSK examines each file record segment (FRS) in the volume's master file table (MFT). Every file and directory on an NTFS volume is uniquely identified by a specific FRS in the MFT and the percent complete that CHKDSK displays during this phase is the percent of the MFT that has been verified. During this pass, CHKDSK examines each FRS for internal consistency and builds two bitmaps, one representing what FRSs are in use, and the other representing what clusters on the volume are in use. At the end of this phase, CHKDSK knows what space is in use and what space is available both within the MFT and on the volume as a whole. NTFS keeps track of this information in bitmaps of its own that are stored on the disk allowing CHKDSK to compare its results with NTFS's stored bitmaps. If there are discrepancies, they are noted in CHKDSK's output. For example, if an FRS that had been in use is found to be corrupted, the disk clusters formerly associated with that FRS will end up being marked as available in CHKDSK's bitmap, but will be marked as being "in use" according to NTFS's bitmap.

During its second pass, CHKDSK displays a message on the screen saying that it is verifying indexes and counts from 0 to 100 percent complete a second time. During this phase, CHKDSK examines each of the indexes on the volume. Indexes are essentially NTFS directories and the percent complete that CHKDSK displays during this phase is the percent of the total number of directories on the volume that have to be checked. During this pass, CHKDSK examines each directory on the volume for internal consistency and also verifies that every file and directory represented by an FRS in the MFT is referenced by at least one directory. It also confirms that every file or subdirectory referenced in each directory actually exists as a valid FRS in the MFT and checks for circular directory references. Finally, it confirms that the various time stamps and file size information associated with files are all up-to-date in the directory listings for those files. At the end of this phase, CHKDSK has ensured that there are no "orphaned" files and that all the directory listings are for legitimate files. An orphaned file is one for which a legitimate FRS exists, but which is not listed in any directory. When an orphaned file is found, it can often be restored to its rightful directory, provided that directory is still around. If the directory that should hold the file no longer exists, CHKDSK will create a directory in the root directory and place the file there. If

directory listings are found that reference FRSs that are no longer in use or that are in use but do not correspond to the file listed in the directory, the directory entry is simply removed.

During its third pass, CHKDSK displays a message on the screen saying that it is verifying security descriptors and counts from 0 to 100 percent complete a third time. During this phase, CHKDSK examines each of the security descriptors associated with each of the files and directories on the volume. Security descriptors contain information regarding the owner of the file or directory, NTFS permission for the file or directory, and auditing information for the file or directory. The percent complete in this case is the percent of the number of files and directories on the volume. CHKDSK verifies that each security descriptor structure is well formed and internally consistent. It does not verify that the listed users or groups actually exist or that the permissions granted are in any way appropriate.

The fourth pass of CHKDSK is only invoked if the /R switch is used. /R is used to locate bad sectors in the volume's free space. When /R is used, CHKDSK attempts to read every sector on the volume to confirm that the sector is usable. Sectors associated with metadata are read during the natural course of running CHKDSK even when /R is not used. Sectors associated with user data are read during earlier phases of CHKDSK provided /R is specified. When an unreadable sector is located, NTFS will add the cluster containing that sector to its list of bad clusters and, if the cluster was in use, allocate a new cluster to do the job of the old. If a fault tolerant disk driver is being used, data is recovered and written to the newly allocated cluster. Otherwise, the new cluster is filled with a pattern of 0xFF bytes. When NTFS encounters unreadable sectors during the course of normal operation, it will also remap them in the same way. Thus, the /R switch is usually not essential, but it can be used as a convenient mechanism for scanning the entire volume if a disk is suspected of having bad sectors.

The preceding paragraphs give only the broadest outline of what CHKDSK is actually doing to verify the integrity of an NTFS volume. There are many specific checks made during each pass and several quick checks between passes that have not been mentioned. Instead, this is simply an outline to the more important facets of CHKDSK activity as a basis for the following discussion regarding the time required to run CHKDSK and the impact of the new switches provided in SP4.

During the first and third phases of CHKDSK, the percent complete indicator advances relatively smoothly. There can be some unevenness in the rate at which these phases progress. FRSs that are not in use require less time to process than do those that are in use. Larger security descriptors take more time to process than do smaller ones, and so on. But, overall, the percent complete displayed is a fairly accurate representation of the actual time required for that phase.

The same is not necessarily true for the second phase of CHKDSK. The amount of time required to process a directory is closely tied to the number of files or subdirectories listed in that directory. But the percent complete listed during this phase is the percent of the number of directories to be examined without regard for the fact that some directories might take much longer than others to process. For example, on a volume with many small directories and one very large one, the percent complete might progress rapidly from 0 to 10 percent complete and then appear to get stuck for a long period of time before rapidly progressing from 10 to 100 percent complete. Therefore, unless you know for certain that the directories on a volume are highly uniform with respect to the number of files they contain, the displayed "percent complete" during this phase cannot be considered a reliable representation of the actual time remaining for this phase.

To make matters worse for anyone caught in the middle of an unexpected CHKDSK, the second phase of CHKDSK is the one that typically takes the longest to run.

By now, it should be clear that many factors having to do with the state of a volume play a roll in how long CHKDSK will take to run. A formula to predict the time required to run CHKDSK on a given volume would have to take into account such factors as the number of files and directories, the degree of fragmentation of the volume in general as well as of the master file table in particular, whether files have both long names and 8.3 formatted names, and how much corruption actually needs to be fixed. And that is to say nothing of hardware issues such as amount of system memory, the speed of the CPU, the speed of the disk or disks, and so on.

Rather than attempt to predict how long CHKDSK will take to run for a given volume on a given hardware platform, suffice to say that it can take anywhere from a few seconds to several days -- depending on your specific situation. Unless /R is used, for a given hardware platform the biggest concern is the number of files and directories rather than the absolute size of the volume. That is, a 50 GB volume with one or two large database files will only take seconds for CHKDSK to run provided /R is not specified. If /R is specified, CHKDSK will have to read verify every sector on the volume, and that clearly adds significantly for large volumes. On the other hand, even a relatively small volume might take hours to run CHKDSK if it has hundreds of thousands or millions of small files -- whether or not /R is specified.

The best way to predict how long CHKDSK will take to run on a given volume is to actually do a trial run in read-only mode during a period of low system usage. Care must be taken using this technique, however, for three reasons:

- Read-only CHKDSK will abort before it completes all three phases if it encounters errors in earlier phases and is prone to falsely reporting errors when in read-only mode. That is, CHKDSK may report that a disk is corrupted even when there is no real corruption present. This can happen if NTFS happens to modify areas of the disk on behalf of some program activity that CHKDSK is examining at the same time. To verify a volume correctly, the volume must be in a static state, and the only way to guarantee that state is to lock the volume. CHKDSK only locks the volume when /F or /R (which implies "F") is specified. Thus, you may need to run CHKDSK more than once to get it to complete all passes in read-only mode.
- System load and whether CHKDSK is running online or during the Windows NT boot sequence can impact the time required to run CHKDSK. CHKDSK is both CPU and disk intensive. Which factor becomes the bottleneck will depend on the specific hardware scenario, but, if heavy disk I/O or high CPU usage is going on concurrent with a read-only CHKDSK, inflated times will result. Also, Autochk.exe runs in a different environment than Chkdsk.exe. While running CHKDSK through Autochk.exe affords exclusive use of CPU and I/O resources to CHKDSK, it also deprives CHKDSK of the benefit of virtual memory. Thus, while Autochk.exe would usually be expected to run faster than Chkdsk.exe, systems with relatively low amounts of RAM may see longer times for Autochk.exe than for Chkdsk.EXE.
- Fixing corruption adds to the time required. A read-only CHKDSK can complete only if no significant corruption is found. If a disk suffers only minor corruption, the time to fix the problems will be only slightly longer than that required for read-only CHKDSK. But if there is major damage, as might result from a serious head-crash or other major hardware failure, the time required to run CHKDSK can increase in proportion to the number of files damaged. In extreme cases, this could more than double the time required for CHKDSK.

Introducing the /C and /I Switches

The /C switch directs CHKDSK to skip the checks that detect cycles in the directory structure. Cycles are a very rare form of corruption in which a subdirectory has itself for an ancestor. Using the /C switch can speed CHKDSK by about 1 to 2 percent. Using /C can also leave directory "loops" on an NTFS volume. Such loops may be inaccessible from the rest of the directory tree and could result in some number of files being orphaned in the sense that they cannot be seen by any Win32 applications -- including backup applications.

The /I switch directs CHKDSK to skip checks that compare directory entries to the FRSs that correspond to those entries. Thus, while the directory entries are still checked to be sure that they are self-consistent, they are not necessarily consistent with the data stored in their corresponding FRSs even after CHKDSK has run with this switch in effect. Using the /I switch typically results in CHKDSK times being reduced by 50 to 70 percent. Exactly how much faster CHKDSK is with this switch will depend on factors such as the ratio of files

to directories, as well as on the relative speed of disk I/O versus CPU speed, and is, therefore, difficult to predict in advance. The use of the /I switch can result in directory entries remaining that refer to incorrect FRSs or in FRSs remaining that are not referenced by any directory entry. The later case is another form of orphaning. The file represented by the FRS may be intact in all ways except for the fact that it is invisible to all Win32 applications-including backup applications. In the former case, files may appear to exist; yet applications encounter errors when attempting to access them.

When disk corruption is detected on a volume, you have three basic choices

- Do nothing. For a mission critical server that is expected to be online 24 hours a day, this is often the choice of necessity. The drawback to this option is that relatively minor corruption can "snowball" into major corruption if it is not repaired as soon as possible after it is detected. Therefore, this option should only be considered when keeping a system up is more important that the integrity of the data stored on the corrupted volume because all data on the corrupted volume should be considered "at risk" until CHKDSK is run.
- Run a full CHKDSK. This option repairs all file system data, restoring all user data that can be recovered by means of an automated process. The drawback to this option is that a full CHKDSK can require several hours of downtime for a mission critical server at an inopportune time.
- Run an abbreviated CHKDSK using some combination of the /C and /I switches. This option repairs the kinds of corruption that can "snowball" into bigger problems in much less time than a full CHKDSK would require, but does not repair all the corruption that might exist. A full CHKDSK will still be required at some future time to guarantee that all the data that can be recovered has been recovered.

It should be pointed out that NTFS does not guarantee the integrity of user data following an instance of disk corruption -- even when a full CHKDSK is run immediately after corruption has been detected. Thus, there may be files that CHKDSK cannot recover. Also, files that are recovered may be internally corrupted even after CHKDSK has been run. It, therefore, remains vitally important that mission critical data be protected by means of a regimen of periodic backups or other robust disaster recovery methodology.

Additional query words: checkdisk cyclic circular

Keywords: kbFEA kbfix kbinfo KB187941
Technology: kbNTTermServ400 kbNTTermServSearch kbwin2000AdvServ kbwin2000AdvServSearch kbwin2000DataServ kbwin2000DataServSearch kbwin2000Pro kbwin2000ProSearch kbwin2000Search kbwin2000Serv kbwin2000ServSearch kbWinAdvServSearch kbWinDataServSearch kbWinNT400search kbWinNTS400search kbWinNTS400sp4 kbWinNTsearch kbWinNTSsearch kbWinNTW400search kbWinNTW400sp4 kbWinNTWsearch

---

Send feedback to Microsoft

© 2004 Microsoft Corporation. All rights reserved.