About

Modified and ported to OS/2 by:
Andreas Kaiser
Danziger Str. 4
D-70597 Stuttgart
Germany

Fidonet: 2:246/8506.9

Internet: ak@ananke.s.bawue.de

Fax: 49-7531-18708 - see note below.

If you can use electronic mail, please do so. The fax number is not my own, so sending stuff not related to GTAK gets dropped into the trashcan without reaching me.

In the case of Internet email, please use a valid domain style "From:" address. If you use bang paths, bitnet, the .uucp domain or some weird kind of embedded domain address, chances are good that you won't see any reply. Chances are only little better if you ask for "please reply to x@y.z" in the body instead of the header, especially if this note is not easily noticed at the top of your mail (if I don't notice it until my reply has been returned, it is almost certainly to late to be fixed).

COPYING

GNU GENERAL PUBLIC LICENSE

Version 1, February 1989

Copyright (C) 1989 Free Software Foundation, Inc.
675 Mass Ave, Cambridge, MA 02139, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The license agreements of most software companies try to keep users at the mercy of those companies. By contrast, our General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. The General Public License applies to the Free

Software Foundation's software and to any other program whose authors commit to using it. You can use it for your programs, too.

When we speak of free software, we are referring to freedom, not price. Specifically, the General Public License is designed to make sure that you have the freedom to give away or sell copies of free software, that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of a such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any work containing the Program or a portion of it, either verbatim or with modifications. Each licensee is addressed as "you".

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this General Public License and to the absence of any warranty; and give any other recipients of the Program a copy of this General Public License along with the Program. You may charge a fee for the physical act of transferring a copy.

2. You may modify your copy or copies of the Program or any portion of it, and copy and distribute such modifications under the terms of Paragraph 1 above, provided that you also do the following:

a) cause the modified files to carry prominent notices stating that you changed the files and the date of any change; and

b) cause the whole of any work that you distribute or publish, that in whole or in part contains the Program or any part thereof, either with or without modifications, to be licensed at no charge to all third parties under the terms of this General Public License (except that you may choose to grant warranty protection to some or all third parties, at your option).

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the simplest and most usual way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this General Public License.

d) You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Mere aggregation of another independent work with the Program (or its derivative) on a volume of a storage or distribution medium does not bring the other work under the scope of these terms.

3. You may copy and distribute the Program (or a portion or derivative of it, under Paragraph 2) in object code or executable form under the terms of Paragraphs 1 and 2 above provided that you also do one of the following:

a) accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Paragraphs 1 and 2 above; or,

b) accompany it with a written offer, valid for at least three years, to give any third party free (except for a nominal charge for the cost of distribution) a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Paragraphs 1 and 2 above; or,

c) accompany it with the information you received as to where the corresponding source code may be obtained. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form alone.)

Source code for a work means the preferred form of the work for making modifications to it. For an executable file, complete source code means all the source code for all modules it contains; but, as a special exception, it need not include source code for modules which are standard libraries that accompany the operating system on which the executable file runs, or for standard header files or definitions files that accompany that operating system.

4. You may not copy, modify, sublicense, distribute or transfer the Program except as expressly provided under this General Public License. Any attempt otherwise to copy, modify, sublicense, distribute or transfer the Program is void, and will automatically terminate your rights to use the Program under this License. However, parties who have received copies, or rights to use copies, from you under this General Public License will not have their licenses terminated so long as such parties remain in full compliance.

5. By copying, distributing or modifying the Program (or any work based on the Program) you indicate your acceptance of this license to do so, and all its terms and conditions.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

7. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Program specifies a version number of the license which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the license, you may choose any version ever published by the Free Software Foundation.

8. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

9. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH

YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

10. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to humanity, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 1, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) 19xx name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (a program to direct compilers to make passes at assemblers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

That's all there is to it!

Files

File	Description
tar.exe	Backup & restore program.
gz124-32.zip	Documentation and executable of GZIP compression support. GNU-Zip 1.2.4. This is an unmodified copy of a file available on public access FTP servers. It is just an add-on to GTAR, not part of it. Source code is available on many public access FTP servers carrying OS/2 or GNU packages.
compress-2.zip	Source and executable of COMPRESS compression support. This is an unmodified copy of a file available on public access FTP servers. It is just an add-on to GTAR and is not governed by the GPL.
SRC*	Modified sources of GNU TAR 1.10.
LIB*	Include files and support library source code required to compile TAR.EXE.
TAR.INF	OS/2 hypertext version of the Texinfo document of GNU TAR 1.09. Outdated, but still better than nothing - forget the description of incremental backups.

Installation

Be sure you have the usual timezone setting in CONFIG.SYS, such as
```
set TZ=CET-1
```
for central europe or
```
set TZ=PST8PDT
```
for US pacific time. Else timestamps may be incorrect when using archives from/on other systems, since the TAR archive contains GMT timestamps. The daylight savings part is optional as it is ignored by the current implementation of the EMX runtime library, so if you want to be precise, you have to take care of the daylight savings time yourself (use set TZ=CEST-2 in summer, or something alike).

Copy the required executables (*.EXE) into a directory mentioned in the environment variable PATH.

Using

This is not an introduction into TAR. Look into TAR.INF or a good UNIX book to learn more about the usage of TAR. Note that TAR.INF is a compilation of the TeX file distributed with GNU TAR and has not been updated since GNU TAR 1.09, so some contents are out of date, most notably the part describing incremental backups.

Get help with
```
tar --help
```

If an option prefixed by `--` expects an argument, you can separate option and argument either by space of by =. If you are lazy, you can use + instead of `--`. Long options can be abbreviated.

New/changed options:

`-0 -1` Alias for `-f disk:a:` and `-f disk:b:`

Writing diskettes in this way wipes out the DOS format. You have to reformat such a diskette before reuse with DOS/OS2 files. Such diskette archives can be used to exchange files with most UNIX systems.

Not recommended for 360/720KB diskettes in 1.2MB/1.44MB drives.

The floppy disks must already be low-level formatted and must not contain bad sectors.

To achieve maximum speed, the TAR blocksize should be the same as the cylinder size of the used floppy disk format: `-b30` for 1200K and `-b36` for 1440K.

`-2` .. `-7` Alias for `-f scsi:+tape$2` .. `-f scsi:+tape$7`

`-E` Similar to `-v`, but prints directory names only as they (and the files within them) are processed. Doesn't work with GNU dumps.

`-D --tape-dir=`*`[option,]filename`*

Use and maintain a tape directory. New data is appended to the (text) file if it already exists.

Option = `abs` (default) uses absolute QFA on restore (default), *option* = `rel` uses relative QFA. Relative QFA requires unique volume names. Choosing between `rel` and `abs` is only required on QFA restore. When archiving in absolute QFA mode, the QFA tape directory is prepared to support both, if the device permits.

This option allows fast random access to files and directories (QFA, Quick File Access), provided the device supports the random access commands of SCSI-2 or Tandberg TDC3600/3800 SCSI-1 series. Some 60-150MB streamers do not support these commands, so there is no way for a fast random access on these models. This is a deficiency of the SCSI command set prior to the SCSI-2 specification.

This option is not usable with compression mode. Buffered mode (`-Q` aka `--buffered`) is only allowed when creating an archive - in this case, a temporary tape directory file is created in the directory pointed to by the environment variable `TMP`, so make sure there is enough space on this particular disk.

Examples:

```
--tape-dir tape-1.dir
-D rel,tape.2.dir
```

`-g --incremental -G --generation`

See the chapter on incremental backups.

`-L` *`n`* `[B|K|M]` You can append `B` or `K` or `M` now, with `B`=blocksize (TAR block size, not tape block size), `K`=KB, `M`=MB.

`-L 145M` splits archive into chunks of 145 MB.

Note that TAR does not account for the current tape position. If there is less space on the remaining tape than specified in `-L`, TAR will run until it hits the physical end of the tape.

`-p --all-files`

Include extended attributes, SYSTEM and HIDDEN files on backup. Account for extended attributes, SYSTEM and HIDDEN file attributes on list and restore. The SYSTEM and HIDDEN attributes are encoded in the group and world execute bits of the UNIX style file mode.

When writing an archive with `-p`, TAR tries to open files in "deny none" share mode only if "deny write" is not allowed. Without `-p`, it fails with "permission denied" (assures consistent contents).

Note: If a file is in use while being archived, extended attributes need not reflect the state of the contents part of the file or directory.

Warning: The archive records created for extended attributes are specific to GTAK. Other TAR implementations may behave strange when encountering these records.

`-pp --all-timestamps`

Archive and restores all time stamps, including access and creation time stamps of HPFS files. This is the default for GNU dumps. `-pp` is the same as twice `-p`.

Posix format archives do not support access and creation timestamps.

`-Y --no-recursion`

Save only the files specified on the command line. Do not save contents of specified directories.

`--fat` Convert long (i.e OS/2 HPFS or UNIX) filenames to FAT 8.3 filenames if required. Use this option if you extract archives with filenames that exceed the 8.3 FAT restrictions onto HPFS file systems but want to convert to 8.3 names. If you extract onto FAT file systems, names which do not fit into 8.3 are always converted, no matter if `--fat` is used or not.

`--compress --uncompress --zip=`...

Compress the archive. By default, the external program `compress` is used for writing archives and `gzip` is used when reading archives.

These defaults can be overridden by `--zip=...` or by defining the environment variable `TAR_COMPRESS`. Use `-zip=gzip` or `set TAR_COMPRESS=gzip` to use gzip both when reading and when writing. You can specify options, so `--zip=gzip,-9` (you can use commas instead of spaces) or `set TAR_COMPRESS=gzip -9` forces maximum (slow) compression.

Any filter program can be used instead of `compress` and `gzip`, provided it can be run with binary pipes on both ends and it accepts the option `-d` for decompression.

`-Q --buffered`=[...]
: Pass the archive data stream through an external buffer program. The optional argument controls the buffer program and its arguments. Spaces can be replaced by commas. If there is no option argument specified, it defaults to the contents of the environment variable TAR_BUFFER, or to `buffer -i90 -o0` when reading and `buffer -i0 -o90` when writing.

  For multi-volume support in buffered mode, you have to use `-L` to avoid running to the end of the backup device.

  Any filter program can be used instead of `buffer`, provided it can be run with binary pipes on both ends.

  Buffered mode can significantly reduce the stop-and-go operations, which are often seen when archiving lots of small files.

`-Z`
: Shorthand for `--compress --buffered`.

`--old-ea`
: Archive extended attributes in 16-bit representation, which is the way the OS/2 program EAUTIL handles them. The default is the 32-bit representation. Versions of GTAK 2 prior to 2.33 cannot handle the 16-bit representation. This option does not imply that extended attributes are archived at all, it just sets the way they are archived if at all.

`--signed-checksum`
: Compute TAR header checksum treating bytes above 127 as negative values. Only different if file or volume names contain non-ASCII characters. Some implementations of TAR require signed checksums. Note that non-ASCII characters are not portable, GTAK does not convert between different code pages.

`--substitute=`*`old`*`,`*`new`*
: When reading an archive, change *old* to *new* at beginning of filename. Allows redirection of LAN Server UNC names. Only one `--substitute` option is allowed.

  Example: `--subst=//code1a/h$/,//code2a/g$/`

`--acl`
: Archive/restore IBM LAN Server access control lists with each file or directory having an ACL attached.

  Warning: The archive records created for access control lists are specific to GTAK. Other TAR implementations (as well as GTAK prior to 2.34) may behave strange when encountering these records.

`--dump-acl`
: When restoring an archive with ACLs, restore the ACLs for file *file* to *file*.ACL. If both `--acl` and `--dump-acl` are specified, both the server ACL and the disk file are created. The ACL file format is ASCII readable.

`--dump-acl=`*`file`*
: When restoring an archive with ACLs, restore the all ACLs into *file*. If both `--acl` and `--dump-acl` are specified, both the server ACL and the disk file are created. The ACL file format is ASCII readable.

`--posix`
: Use Posix archive format when archiving. Posix TAR supports long filenames without mangling. When reading archives, TAR detects the archive format automatically.

  The Posix archive format supports long filenames up to 256 bytes, but only if there is a path separator (`/`) in the right place of the filename, in such a way that the part before the separator is at most 155 bytes long and the part beyond at most 100 bytes. If a name does not comply to this restriction, it is truncated.

  Note: The Posix format does not support accessed/created timestamps.

  Warning: While this version supports GNU style multi-volume archives in Posix archive format, the archive format used for a file split across multiple volumes is incompatible both to plain GNU TAR and to standard Posix TAR due to a confict between GNU and Posix archive formats.

`--case-sensitive`
: When reading an archive, compare filenames case sensitive.

  You won't need this option unless the archive contains two files with the same path and name, distinguished only by upper/lower case.

Notes

This is software designed for (and ported from) UNIX systems, so don't expect all options and features to work. And don't be too much confused by some warnings, which may occur when TAR tries to set attributes or filetimes at times when this is not allowed under OS/2.

This TAR automatically saves and restores Extended Attributes when `-p` is given.

Path and filename specifications may be sensitive to upper/lower case and forward/backward slash convention, although I tried to remove these dependencies. UNC names are supported.

Since the TAR format is standardized and available on every UNIX system, you can use TAR to exchange data with UNIX systems. Even compressed data is compatible with 32-bit UNIX systems. If you intend to exchange tapes or diskettes with standard UNIX tar programs, avoid the GNU TAR specific options `-V`, `-g` and the GTAK specific options `-p`, `--acl`. Some UNIX tar implementations skip extended attribute and ACL records, some others fail and some put the extended attributes and ACL entries to disk files, which are immediately overwritten by the file itself, except for directories which cannot be created since a file with the same name already exists.

This TAR removes leading absolute path prefixes such as drive letters, leading slashes and UNC prefixes (verbose) on create and extract, unless `-P` or `--absolute-paths` is specified. If you place save several filesystems (partitions) into the same archive without using `-P`, you may not be able to distinguish files with the same path and filename on restore. Use separate archives (run TAR separately) for each filesystem or specify `-P` or `--absolute-paths`. You can still extract files to a relative path even when the archive was written with an absolute path, as the absolute path prefixes are by default removed on restore too (read: the files are restored relative to the current directory unless `-P` or `--absolute-paths` is specified).

On extract, paths and filenames must be specified exactly as shown by `tar t` except for the trailing `/` of directories, which should be omitted.

Hint: Although the output of the `-v` option looks nice, the big amount of screen output may slow down the tape output, forcing the tape into inefficient stop-and-go mode. Use the `-E` option instead, if you like to get feedback about the progress of backups (note that `-E` doesn't work with GNU dumps).

On compressed backups: A 486 33MHz equipped with a 5MB/min streamer provides a compression (compress) rate of about 1.5 on a whole disk with 90MB binaries and an effective disk data throughput of about 85 KB/sec and a tape data throughput of about 55 KB/sec. When storing the backup on a diskette (an easy way to transfer HPFS files), compressed and buffered (`-Z`) data throughput is faster than the diskette data throughput. Therefore compressed mode can be considerably faster than uncompressed mode, depending on the machine's speed and the data throughput of the backup device.

WARNING: If a compressed archive contains a bad block, all data of the archive starting with the bad block is lost. Uncompressed archives are recoverable by manually skipping beyond the bad block, compressed archives are not recoverable. Neither compress nor gzip were written with tape backups in mind.

Examples:

`tar cp c:/`
: Backup the `C:` disk and all subdirectories including extended attributes, hidden and system files. Do not specify `C:/*`, as wildcards given on the command line do not include hidden and system files, so in this case still all hidden and system files are archived, except for those in the root directory.

`tar tpv`
: List all files on the archive.

`tar xpv`
: Restore whole archive. Restore extended attributes and set attributes of hidden and system files.

`tar x path/file`
: Restore a single file or a directory tree. Always use forward slashes.

```
tar -c -v -D c:/tar_maps/tape.dir c:/
```
Backup logical disk `C:` and append a tape directory to the specified file.

```
tar -x -v -D c:/tar_maps/tape.dir this/is/a/single/file
```
Restore `this/is/a/single/file` using absolute QFA.

```
tar -x -v -D rel,c:/tar_maps/tape.dir this/is/directory
```
Restore `this/is/a/directory` using relative QFA.

Quick File Access - QFA

Quick File Access = quick access to a subset of files on the archive.

By default, file access in TAR is strictly sequential, reading the whole tape archive. It may take more than one hour to retrieve a single file from a tape archive. This version of TAR implements quick file access to retrieve files in a few minutes (QIC) or less than one minute (DAT) average.

To be able to use QFA, TAR has to maintain a tape directory. This is not supported by the standard TAR format itself. I did not change the TAR format nor did I add a tape directory to the archive, to remain compatible to other TAR implementations. Instead, a tape directory is written to disk, when `-D` is specified.

BTW: Even though I'm consistently using the term *tape directory*, the (absolute) tape directory mechanism can also be used for quick access to archives written to random access media such as plain files or raw floppy-disks.

TAR cannot recreate a tape directory by reading the archive, since some devices behave differently when reading the archive. For example, the TDC4120 returns different block identifications at the same tape position, depending on whether the tape is read or written. So once a tape directory is lost, you are no longer able to use QFA on the corresponding tape.

The QFA support implemented by TAR does not include management of the tape directory, other than appending QFA position records. A tape directory file is not overwritten by TAR, instead TAR appends to existing contents. If you rewrite a tape from the beginning, you should erase the tape directory file first (if you are using the same file as before). TAR does not save the tape directory on tape itself. If you want it on tape, you have to put it there yourself. Or put in on floppy disk - while it may look too large at first glance, it will shrink to ˜20% using file compression programs.

If your device supports partitions (such as Tandbergs SCSI-2 >= 525MB and HP-DAT), you can make a directory partition using the tape utility and put the directory file into it afterwards (either with another TAR run or directly using `tape write`) - a good place for CMD/REXX scripts. I'm not going to explain SCSI-2 partition management right now, sorry.

A short explanation of the contents of the tape directory, if you are interested:

```
        307138+1486,94926       -    237567 Aug 06 19:37 1994 down/unzip511.exe
```

The first number (307138) is the absolute id of the cluster. The second number (+1486) is distance from the beginning of the cluster in TAR records. The third number (94926) is the distance from the beginning of the archive in TAR records. The `-` character is the type of the entry (plain file, directory, extended attributes, ACL, ...).

QFA support is implemented in two variants, described in the following sections. When creating an archive with `-D`, the tape directory is prepared for both variants, provided that the device supports both.

Tape directories created with prior versions of GTAK/GTAR can be used with absolute QFA only.

Absolute QFA

A tape position is identified by an unique id, usually some kind of device specific or format specific block number (but only the device knows what these numbers mean, there need not be a computable relation between the number of written blocks and the block id). To support absolute QFA, the archive device must be able to provide such ids. When using random access media like files or floppies, the file offset within the archive is used as the identification. When using sequential devices, TAR relies on some QFA support to be provided by the device.

There are a few devices which do not support block ids, such as early versions of the Wangtek 150MB streamers, you cannot use absolute QFA with these devices. If a tape directory shows `-1` in the first columns, your device does not support the required SCSI commands, at least not in the SCSI mode you specified.

The SCSI-2 command set supports two different types of block ids. Logical ids (`LB`) and device dependant ids (`DB`). Some devices support only one of these (TDC4120: device dependant block ids only). Other devices support both, but the numbers shown may be different (HP DAT). You can select the type of block id using the

`LB` and `DB` option in the device specification (TAPE environment variable). To remain compatible to prior versions of GTAK/GTAR, `DB` is the default, however `LB` is recommended.

Block ids are obtained every 1MB of archived data. Positions within a 1MB cluster are accessed relative to the beginning of the cluster.

When restoring using absolute QFA, absolute positioning is used whenever the current tape position is unknown or the distance between the current and the requested position is large. For small distances, relative positioning is used even though absolute QFA is requested. This way absolute QFA mode achieves good performance even when a whole subdirectory is restored (a major difference to GTAK <= 2.12 which used absolute positioning for every single file).

Note that, on a HP DAT in `DB` mode, the id of a filemark appears to be the same as the id of the block immediately following it. Therefore absolute QFA may fail for files located within the first cluster of an archive, since the device positions to the filemark instead of the first data block and TAR terminates with error.

Relative QFA

Relative QFA does not use absolute tape positions. Instead the device is scanned for filemarks and volume labels. When the beginning of the requested archive is found, the device is requested to skip logical blocks up to the file's position. Whether relative QFA actually is quicker than standard file access depends on the tape device. While some QIC devices support fast filemark and block skips, this usually has to be enabled explicitly using manufacturer unique commands (not supported by GTAK unless you know the corresponding mode page layout). So, in general, relative QFA is useful with helical scan devices only (DAT, Video8). With most QIC devices, relative QFA operations are not faster that non-QFA operations.

Relative QFA requires unique volume labels to identify the beginning of the requested archive. Each archive on a tape must have its own unique volume label. Remember that volume labels are a nonstandard extension of GNU TAR and may disturb other TAR implementations.

Relative QFA starts at the current tape position and scans up to the end of the used part of the tape. If you are not sure whether a file is located before or after the current tape position, you should explicitly rewind the tape before running TAR.

Incremental Backups

Incremental backups based on OS/2's "archive" bit

`--generation=first`
: First (full) backup. The archive bit is reset after each file is processed.

`--generation=second`
: Incremental backup. Only files with the archive bit set are archived. The archive bit is not reset, so each increment is based on the last full backup.

  To restore using `s` increments, you need the full backup and the last incremental backup. Intermediate increments are not required.

`--generation=incremental`
: Incremental backup. Only files with the archive bit set are archived. The archive bit is reset after each file is processed, so each incremental backup is based on the last incremental backup.

  To restore using `i` increments, you need the full backup and each incremental backup in the order they were run.

  If TAR is interrupted in `i` incremental backups, you should no longer run incremental backups. The next backup should be a full backup, since TAR resets the archive bits after each file is processed.

When restoring archive bit based incremental backups, files which were deleted since the last full backup are nevertheless restored to disk.

You do not have to specify `--generation` on restore.

Examples

```
tar -cE --abs --gen=f --label=Full-7.7.1994 c:/ d:/ e:/
tar -cE --abs --gen=i --label=Incr-15.8.1994 c:/ d:/ e:/
tar -xv c:/os2
```

GNU incremental backups

`--incremental=`*`dumpfile`* `--generation=first`
: First (full) backup. The dumpfile is erased before use.

`--incremental=`*`dumpfile`* `--generation=second`
: Incremental backup. Files are only archived if the "modified" timestamp exceeds the timestamp mentioned in the dumpfile or if its directory is not mentioned in the dumpfile. The dumpfile is not updated, so each increment is based on the last full backup using the same dumpfile.

  To restore using `s` increments, you need the full backup and the last increment. Intermediate increments are not required. You do not need the dumpfile.

`--incremental=`*`dumpfile`* `--generation=incremental`
: Incremental backup. Files are only archived if the "modified" timestamp exceeds the timestamp mentioned in the dumpfile or if its directory is not mentioned in the dumpfile. The

dumpfile is updated after all files are processed, so each increment is based on the last incremental backup using the same dumpfile.

To restore using `i` increments, you need the full backup and each incremental backup in the order they were run. You do not need the dumpfile.

If TAR is aborted, the dumpfile it not updated, so the repeated run will start at the same point as the aborted run.

When using GNU incremental backups, archived directories contain info about all files located in this directory. When restoring with `--incremental`, all files which exist in the directory on disk but not in the directory on the archive, are removed from the disk, since they are left over from previous incremental restores. Note that for syntactic reason, you have to pass some filename as argument to `--incremental` even though the file itself is neither required nor touched on restore.

If you do not specify `--incremental` on restore, creation and accessed timestamps are not extracted by default.

You do not have to specify `--generation` on restore.

Since OS/2 does not maintain directory timestamps the way UNIX does, the OS/2 version processes all directories in the way mentioned in the previous paragraphs, not just the ones which were changed in some way since the last update of the dumpfile. So the directories can take up a considerably amount of space in each incremental backup even if not a single file has to be archived, and most likely it causes a lot of stop-and-go when using a QIC type device.

GNU incremental backups implicitly define `--all-timestamps` and `--all-files`.

GNU incremental backup archives are incompatible to standard TAR archives. You might not be able to read such archives with standard TAR programs.

Examples

```
tar -cE --abs --incr=date-1.dat --gen=f --label=Full-7.7.1994 c:/ d:/ e:/
tar -cE --abs --incr-date-1.dat --gen=i --label=Incr-15.8.1994 c:/ d:/ e:/
tar -xv --incr=nul c:/os2
```

Notes on both variants

OS/2 does neither update the "modified" timestamp not does it cause the "archive" bit to be set, when only the extended attribute part of a file or directory have been changed. So incremental backups unavoidably fail to archive files where only extended attributes have been changed. However, directory records are created for all processed directories independent of timestamps or archive bits, so extended attributes of directories are always archived (with `-p`, of course). To include the complete state of the OS/2 desktop even in incremental backups, run TAR non-incremental on critical directories before, creating a disk file with

```
tar cpf c:/desktop.tar "c:/OS!2 2.0 Desktop"
```

(or whatever name your desktop tree has), and include this disk file in the tape backup. Note that there are a few other important extended attributes such as printer settings.

Appendix

Known Bugs

`--tape-length` doesn't know about the current tape head position. If you append to a tape with existing contents, specifying the full tape length in `--tape-length` will not save you from a tape overrun.

-

TAR commands which have to backspace or have to overwrite previous archive contents, are not supported when operating on tapes. Such as `--update`, `--delete`, `--verify`. Since TAR maintains its own end-of-archive indicator, `--append` is likewise unusable with tapes.

-

OS/2 neither updates the "modified" timestamp not does it set the "archive" bit, when only extended attributes of a file have been changed.. So incremental backups unavoidably fail to archive files where only EAs have changed.

Since the OS/2 workplace shell desktop saves some of its information in extended attributes of directories in the desktop directory tree, you should first save the desktop tree to a backup file on disk, before making an incremental backup of your boot disk, else your desktop may be incomplete or invalid when the backup is restored.

-

If you use compressed backups on tapes despite my warning, note that you may get an error message after restoring the last file in the archive, like "EOF not on block boundary". Don't panic, this is normal.

Reason: On backup, the last written block of compressed data gets padded with nulls to make up a whole tape block. Now when "compress" finds this null data on restore, it feeds it as unwanted garbage data into TAR, but since this data is located beyond TARs own EOF record, the data is actually ignored and there is nothing lost.

-

To avoid left-over extended attributes when extracting an archive, each extracted file is deleted from disk (not recoverable by UNDELETE) before extraction (unless `-k` is given). Except for directories however, so extended attributes of a directory are changed only if the archive contains an EA record for the directory. If you archive a directory without EAs and restore it over a directory with EAs, the existing EAs are not changed.

It is up to you to call this a bug or a feature - while I could explicitly remove EAs from directories, I consider it a little dangerous since directory EAs seem to be far more important than file EAs. Any comments on this subject?

If you want to restore a directory tree (or partition) to the exact state it had at the time of archiving, you should rename or erase the directory tree before.

-

If an archive was created with extended attributes (`-p`), you compare the archive to the disk files (`-dp`) and a disk file (or directory) now has EAs whereas it had no EAs at the time the archive was created (so there is no EA record in the archive) you don't get a "differs" message.

-

Names exceeding 100 chars in length are saved as mangled filenames, unless `--posix` is specified. Mangled names are extracted to your current directory and renamed when TAR hits the GNU-specific demangler record near the end of the archive. Renaming of mangled names most likely fails when `-P` (`--absolute`) is used as files cannot be renamed across filesystems. Demangling does not work with QFA. Comparing archives with mangled names doesn't map mangled names to real names, so you get messages for all mangled names.

-

GTAK does not correctly handle archives with mangled names, created with versions of GTAK prior to 2.3x.

-

When using HPFS386, some files are not archived automatically, since they are not matched by the wildcard pattern `*`, such as all files starting with `#`. This is a bug or feature of the 32-bit filesystem API, not of TAR. However neither the standard HPFS of OS/2 2.1 nor the 16-bit API exhibit this behaviour.

-

Compressed mode conflicts with QFA.

-

Buffered mode conflicts with QFA when reading the archive.

-

Multi-volume support in buffered mode requires `-L` aka `--tape-length`. Hitting the end of the backup device aborts in buffered mode whereas it asks for the next media in unbuffered mode.

-

If TAR is killed (Ctrl-C) while creating an archive in buffered mode with tape directory support, a temporary file is left over in the directory pointed to by the environment variable `TMP`, and the tape-directory file is not updated.

-

Multi-volume support in GNU TAR 1.10 is a real hack and has its problems. If the extended attribute part or the ACL part of a file is split over two volumes, it fails to compare and it gets restored incorrectly.

-

If you archived files with `-P` aka `--absolute` and restore (compare, list) them without, you have to omit absolute path indicators when restoring without QFA and have to specifiy them when restoring with QFA (nevertheless the files are restored relative to the current directory).

Common mistakes

To backup all files, do not use wildcards, such as like `C:/*`. Unquoted wildcards specified on the command line do not include hidden and system files within the specified directory. Backing up the OS/2 boot partition with `C:/*` will **not** backup the OS/2 system files. Use `C:/` instead.

-

To restore selected files, do not use unquoted wildcard patterns like `*.sys`. Use `"*.sys"` instead. Unquoted wildcards are matched against the current disk contents in a very early stage, before argument processing, so if TAR finds a single file like `config.sys` on your disk, it will restore just this single file instead of all archive files matching the pattern. Quoted arguments are not matched, so they behave as expected. To restore whole directory trees, don't use wildcards, just specify the directory name.

-

If you extract a file from an archive and TAR does not seem to find the file, use the option `-v`. By default, TAR does not display an indication of success, so maybe it was successful, but you misinterpreted its silence as failure - best Unix style: if you don't get a message, it's all right.

Notes

Usage info of TAR is output on file handle 2 (error handle). To redirect the text to a file or printer, add `2>file` or `2>prn` to the command line. To redirect the text into an output pipe, add `2>&1` (redirects handle 2 into handle 1 - I was a little surprised that this is supported by CMD).

-

When you backup a FAT partition, TAR complains about not being able to backup the file `ea data. sf`. This is perfectly OK, since extended attributes are saved with each file individually. It makes no sense to save it, since the sole reason of this file is to cover the disk space used by extended attributes so that DOS CHKDSK doesn't complain about wasted space.

-

For using QFA with multi-volume backups, you have to create the archive with a volume label (`-V`). Else TAR cannot correctly interpret the tape directory. So if you use multi-volume backups, you either loose standard TAR compatibility (volume labels are non-standard) or loose QFA capability.

-

When creating archives in buffered mode with tape directory support, be sure to have enough space for a whole temporary tape directory file in the directory pointed to by the environment variable `TMP`.

-

Buffered mode (`-Q` aka `--buffered` and `-Z`) allocates 2MB buffer memory by default. On systems with 8MB RAM (well, this is a guess, all I know it that 16MB is enough), buffered mode may lead to heavy swapfile activity due to page trashing, so avoid buffered mode if the system does not have enough memory, or reduce the buffer size.

History

2.13 .. 2.40

The volume label got the wrong tag in the tape directory, which prevented easy QFA with multi-volume archives.

The syntax of the device specification (TAPE environment variable) has changed.

The tape directory format changed. However TAR accepts old tape directories when reading with QFA (in absolute mode only).

QFA operation changed significantly. Optimized, you now can read whole directories efficiently using QFA. Added relative QFA. Previous QFA suffered from 32-bit limitation of the lseek command's byte offset, so tapes greater than 2GB could not use QFA. New QFA limit is 1000 GB (2GB * 512).

Added support for QFA using logical block ids, in SCSI-2.

Bypassed an IBM LAN Server bug. It leads to a large number of files left open on server side, when files are opened in "deny none" mode. Now files are opened in "deny write" mode first. If it fails and TAR is called with `-p` given, it tries again in "deny none" mode. TAR should no longer fail with "out of handles".

Compressed tape archives are no longer buffered by default. Use `-Z` to force buffering. Compressed or buffered mode no longer uses the driver r/w interface.

As previously indicated, I partially switched to the EMX supplied extended attribute support functions. Bugs in these functions prevented a complete switch however. As a result, the EA representation in the archive is now available both in the 16-bit and in the 32-bit representation (see `--old-ea`). When reading archives it accepts both (as a side-effect, it now accepts GTAK 1.x EAs). Comparing archives with EAs compares attributes individually. Versions 2.00..2.29 do not accept the 16-bit EAs, they are flagged as "created with another version of GTAK". Archiving with 16-bit EAs is a little slower, comparing EAs is significantly slower.

Fixed some bugs in handling of mangled names. Now pathnames exceeding 100 chars in length are archived and restored correctly, with several restrictions as mentioned above. Archives created with prior versions of GTAK may as well contain mangled names but the renaming doesn't work and extended attributes are lost.

When reading a tape archive, if TAR hits its end-of-archive marker record (a null record) it skips past the next filemark. This fixes the old bug where the second run of TAR hit the filemark at the end of the previous archive and terminated with error. Consecutive runs of TAR read consecutive tape archives correctly now.

Support for IBM LAN Server access control lists and `--substitute`.

Bugfix: Previous GTAK implementations did not attach extended attributes to a directory which already existed.

Posix archive format. This feature required major changes within TAR, so expect a few bugs related to long filenames. Even without `--posix`.

Single-file ASCII ACL restore (`--dump-acl=`*`file`*).

Remove UNC prefix from filenames unless `--absolute`.

2.41

ACL support failed if relative paths were used.

Found yet another case sensivity. Restores are no longer case sensitive now.

QFA failed when a huge file (at least 100,000,000 bytes) caused tape directory columns to be shifted to the right. Fixed for tape directories created with 2.40, earlier ones may still show this bug.

Moved the wildcard notes to "Common mistakes". Maybe a few more people read it now.

Allow `--buffer` control the buffer program, similar to `--zip`.

2.42

Support buffered mode when archiving with tape-directory (QFA).

Added `-Q` as alias to `--buffered`.

Fixed a bug introduced by the Posix changes: The volume label entry in the the tape directory was incorrect, so relative QFA did not work.

Fixed missing multi-volume compares.

Added optional case sensitive compare. Useful in case of conflicts.