NAME

tbku - Table-driven backup script

SYNOPSIS

```
tbku allsets | [fileset] ...
```

DESCRIPTION

`tbku` is a utility script for producing “tarball” backups of some- or all of your files. It is useful both for producing incremental backups or for systemwide images or “snapshots”. The script can be run either from the command line or, more typically, as a `cron` job to automate system backup tasks.

`tbku` uses standard utilities common on Unix-like systems, like `tar`, `sed`, and `uname`. It uses no other special or custom tools. For this reason, it is highly portable across many variants of these systems.

The central benefit of using `tbku` over hand written `tar` commands is that `tbku` is “table driven”. You specify the set of files to back up in a table (a separate file). You can have as many of these “filesets” as you wish, corresponding to different kinds of backups you want done. `tbku` will do backups automatically or manually, based on the name of the “fileset”. This considerably simplifies automating backups, keeping backup logs, and generally maintaining an orderly backup environment.

`tbku` was originally developed as a backup tool for FreeBSD servers. Since then, it has been updated to also work with SUSE Linux, both servers and desktops. `tbku` should work with little- or no modification on any other Unix-like system. For example, `tbku` will run without modification (other than default locations) in a `cywgin` environment under MS-Windows.

INSTALLING tbku

To use `tbku`, all you have to do is install the file somewhere in your `$PATH`. Typically, a good place for it is in `/usr/local/bin`. Just make sure its permissions are 755 so all users will be able to use it.

You may optionally want to put `tbku.1.gz` somewhere in your `$MANPATH` so this documentation will be available as a man page.

There is also a `tbku` port for FreeBSD users that automates the installation and deinstallation of `tbku`. This port can be found under `/usr/ports/sysutils/tbku`. Once installed, all of the documentation for `tbku` (in a variety of formats) including the tool itself, the licensing terms, and the instructions for imaging systems with it, will be found in `/usr/local/share/doc/tbku`.

Once you’ve installed the program, you should verify that its default settings are to your liking. If not, you can override them via environment variables (described later in this document). For interactive use, make sure the environment variables you want to set are exported when you log in. If you’re running `tbku` from a `cron` job, be sure to set the environment variables of interest in the `crontab` file.

USING tbku

Using Filesets

`tbku` has to know just *what* you want backed up. You do this by creating a so-called *fileset* in the appropriate directory (default: `$HOME/tbku/`). Filesets are just text files that list all the files and/or directories that are to be backed up together. For instance, suppose you had a fileset called `manual.fileset.homedirs` that contained just these three lines:

```
/root
/home
/usr/home
```

If you now run this command:

```
tbku homedirs
```

The files and/or contents of `/root`, `/home`, and `/usr/home` would be written to a tarball in the backup directory (default: `/bku/`). By default, the resulting tarball's name has a long string of text that includes the machine name, system type, OS type, date, *and* the so-called *set name*. The "set name" is nothing more than the suffix of the name of the fileset used to produce the tarball, in this case, `homedirs`.

Additionally, you'll also find a log of the backup and "dot files" that tell you when the backup began and when it ended. Here's part of what you might see if you did an `ls -a /bku`:

```
.mach.fake.org-FreeBSD-6.3-STABLE-i386-homedirs-20080319-begin
.mach.fake.org-FreeBSD-6.3-STABLE-i386-homedirs-20080319-end
mach.fake.org-FreeBSD-6.3-STABLE-i386-homedirs-20080319.tar.gz
mach.fake.org-FreeBSD-6.3-STABLE-i386-homedirs-20080319.log
```

The "dot files" don't actually contain any information, but their date/time stamps (you can see this with `ls -al /bku`) will tell you when the backup began and ended.

The log file contains a list of all the files that actually made it into the tarball. The log file also captures *the errors* encountered during a backup. This means that `tbku` is generally pretty quiet during a backup run. It scribbles any complaints it has into the log. So... you should check your logs regularly to make sure everything is working as expected.

You can create as many different filesets as you like (for as many different kinds of backups as you need). So, for example, you may have one for the files you want backed up daily, another for weekly backups, another for taking a snapshot of the entire system, and so on.

The *name* of a fileset can be used to change `tbku` behavior (described below). The *content* of a fileset file must conform to only a few rules:

1) Each line may contain the name of a *single* file or directory. You cannot place multiples of these on a single line.
2) Each entry should be *an absolute path.* That way, `tar` will be able to figure out what it is you want to back up. By default, most modern `tar` implementations will strip the leading `/` so your backup tarball will be relative to wherever you are when you restore from it.
3) There is no support for comments or other metadata inside a fileset. File- and directory names are the *only* thing that should ever be there.

Fileset Naming

`tbku` semantics (behavior) depend on how you've named your filesets. In general, a fileset should be named as follows:

```
auto.fileset.setname

OR

manual.fileset.setname
```

Any fileset name that begins with "auto." will automatically be backed up when you run the script without arguments:

```
tbku
```

If a fileset begins with something other than "auto.", you have to explicitly name the set on the command line for it to be backed up. Say we have only two filesets, `manual.fileset.music` and `manual.fileset.docs`. Then:

```
tbku                # Does nothing
tbku music          # Only backs up the manual.fileset.music file-
set
tbku music docs     # Backs up both filesets
```

The "setname" is used to uniquely name each backup tarball.

Strictly speaking, `tbku` only cares about the "auto" string. Anything other than "auto" as a prefix in the fileset name, will cause the file to be seen as requiring manual invocation. Using "manual" is just a helpful convention.

Similarly, you don't need the "fileset" in the middle of the filename, it's just a helpful convention. `tbku` only examines the prefix of the filename (up to the ".") to determine whether to do automatic backups. It uses the suffix (from the last "." to the end of the file name) to determine the set name. In fact, you don't even have to fully specify the set name, just any trailing substring:

```
tbku ic             # Backs up manual.fileset.music
```

While these little semantic subtleties may be interesting, you are strongly *discouraged* from using them, as they are not guaranteed to be preserved in future releases of `tbku`. Stick to the conventions described above, and you should be fine.

The allsets Option

As you might guess, you can force *all* backup sets to be done regardless of whether they are marked as "auto" or "manual" by doing this:

```
tbku allsets
```

The "allsets" argument must be the first argument on the command line, and anything following it will be ignored. In other words, only the form shown above is meaningful.

tbku: Nothing to do!

You may see `tbku` grumbling about having nothing to do. This happens under one of several circumstances:

1) You ran `tbku` without arguments, but there are no "auto" filesets defined.
2) You ran `tbku` with arguments, but no filesets with matching set names were found.
3) There are no filesets at all.

Autodeletion Of Old Backups

As shipped, `tbku` uniquely identifies each backup set based on machine name, OS, CPU architecture, set name, and, most importantly, date. If you've set it up to run as a cron job, over time you'll accumulate lots of older copies of backups. That's because each new day, the backup file name will change (since it includes the date).

If you don't like this default behavior, change the `TBKUDEL` environment variable to be "YES". It must be *exactly* this string, all in upper case. Anything else will cause `tbku` to *not* autodelete old backups. This is intentional, to make it hard to accidentally enable this feature.

Enabling this feature forces `tbku` to delete all older files associated with the selected set name. This includes the start/stop "dot" files, the log, and the backup tarball itself. In effect, this option forces `tbku` to only keep the most recent backup of each backup set.

Use this option with caution! If you only keep the most recent copy of your backups in your backup directory, you may never be able to get to changes made days, weeks, or months prior.

IMAGING WITH tbku

It is possible to use `tbku` backups to completely (re)image a machine. The general idea is to have `tbku` produce a tarball of all the (relevant) files on the system you want to "clone". Then, you can dump that onto a newly prepared filesystem on the target machine. This is a handy (and relatively quick) way to recover a system after a hard drive failure or upgrade, for example.

The `tbku` distribution contains separate documents that describe in detail how to image both FreeBSD and SUSE Linux systems. You can also read the documents on line at:

http://www.tundraware.com/Software/tbku

CUSTOMIZING tbku

`tbku` is written to be "smart" enough to figure out where your system keeps needed tools like `tar` or `sed`. The only requirement here is that `tbku` be run in an environment that can find the `which` command somewhere in its `$PATH` - `tbku` uses `which` to figure out just where everything it needs lives on your filesystem. If `tbku` cannot figure out where your system keeps things, it will use the FreeBSD default values.

For most FreeBSD and Linux users, this should work without any customization beyond setting environment variables to override default behavior (described below). In rare circumstances, you may need further customization. All the things you're likely to ever want to change appear first in the actual `tbku` script, and are briefly documented there.

DEFAULTS & ENVIRONMENT VARIABLES

You can override the various `tbku` defaults by setting a corresponding environment variable.

Env. Variable	Default Value	Meaning
TBKUDEL	NO	YES -> Delete old backups
TBKUDIR	/bku	Where to write backups
TBKUNAME	$MACHINE-$OSTYPE-$OSREV-$HWTYPE	Tarball base name
TBKUSETS	$HOME/tbku	Filesets found here
TBKUTAPE	/dev/sa0	Tape device (or file)

Examples:

```
export TBKUDIR=/mnt/backups    # Backups written to /mnt/backups

export TBKUNAME=JoeBackup      # Backups named: JoeBackup-<setname>

export TBKUSETS=/tbku          # Looks for filesets in /tbku
export TBKUTAPE-
/tmp/faketape  # Tape backups actually written to *file*

export TBKUDEL="YES"           # Autodelete old backups when starting a set
```

OTHER

`tbku` was originally designed for use by experienced systems administrators and users. As such, it does little or no error checking. If you define backup or fileset directories that are non-existent, for instance, you will get strange behavior. `tbku` *will* try to create the backup directory you've specified if it does not already exist, but this may not work if you're running as anything other than `root` user.

`tbku` is intended to make it easier/more automatic to to backups. It is not, however, idiot-proof. There are some general backup guidelines you should observe:

> **NEVER, EVER, EVER, EVER, EVER ... EVER**, trust a backup tool until you've confirmed that it is correctly producing backups **and** you can properly restore from them!
>
> Always keep multiple copies of your backups. If `tbku` is writing its backups to the same drive/system it runs on, **make sure you also keep a copy of those backups "off system"**.
>
> It's a pretty good idea to keep **multiple backup copies**, on **different media** (disk, tape, DVD, thumbdrive), in **different locations**.

UPDATES & SUPPORT

To get the latest version of 'tbku', go to:

> http://www.tundraware.com/Software/tbku

For questions, comments, or other feedback, send email to:

> tbku@tundraware.com

AUTHOR

Tim Daneliuk, TundraWare Inc.

COPYRIGHT & LICENSING

DOCUMENT INFORMATION

This document was produced using the very useful `reStructuredText` tools in the `docutils` package. For more information, see:

> http://docutils.sourceforge.net/rst.html

`$Id:  tbku.txt,v 1.111 2008/03/21 05:37:05 tundra Exp $`