$> vbs_ls

$> vbs_rm

$> vbs_fs

FlexBuff stuff

release `r45`: utilities that actually are

Historically ...

`vbs_ls, vbs_rm`

- original bash shell scripts from early 2013
- ‘delivered’ with FlexBuff
- never actually pulled their weight

`vbs_fs`

- FUSE virtual file system
- first release Apr 2015
- compiled C++ code
- presents scattered FlexBuff recording as normal file

And then ...

1. FlexBuff system gains traction

2. more systems deployed in the EVN

3. more users

4. rightful complaints about FlexBuff tools
   - operators are confused wether recording works or not
   - cannot usefully see what's on the system or how much
   - scattered recordings are a bugger to deal with (>30 disks!)

Laura Barbas/Yebes writes prototype `vbs_ls` with useful output

Rewrite, release of “vbs_fs r45”

`vbs_ls, vbs_rm`

- both rewritten from scratch in Python

`vbs_fs`

- >80% code deleted and replaced
- improved based on experience with prev. version
- new features

Common to ***all three*** utilities

- transparent support of FlexBuff and Mark6 recording formats
  - `vbs_ls` catalogues all recognized recordings
  - `vbs_rm` can remove any recognized recordings
  - `vbs_fs` presents any scattered recording as single file
- coherent set of command line flags/arguments:
  - `--version`
  - `--help`
  - `-6` search Mark6 mountpoints
  - `-v` search FlexBuff mountpoints (default)
  - `-R <path>` manually add search path `<path>`

vbs_ls, vbs_rm

Remodelled after their illustrious UNIX command line equivalents `ls(1)` and `rm(1)`

- `vbs_ls` gets owner, group, permissions and mtime from on-disk files
- many familiar command line arguments
  - `vbs_ls -lrth` produces output similar to `ls -lrth`

```
exbuf      3.58G Jul 04 21:23 eg088_mp_163600
exbuf    913.08M Jul 04 21:24 eg088_mp_163700
exbuf     68.52G Jul 05 07:52 eg088_ho_no0088
          drw-r--r-- jops      flexbuf     68.52G Jul 05 08:19 eg088_ho_no0089
```

  - `vbs_rm` only removes recognized recordings
    - `-i`, `-f` command line flags from `rm(1)`

vbs_ls specific option

```
$> vbs_ls ... -T ... <pattern> [<pattern>]
```

"accumulate by <pattern>"

```
verkout@flexbuf3:~$ vbs_ls -lTh em117e* em117f* eg088_ys*

Found 3 recordings in 13426 chunks,    3.11T
drw-r--r-- jops      flexbuf      2.36T Aug 05 13:24 eg088_ys*
drw-r--r-- jops      flexbuf    394.59G Jun 06 20:34 em117e*
drw-r--r-- jops      flexbuf    372.59G Jun 06 18:09 em117f*
```

vbs_fs

Heavily optimized, multithreaded virtual file system

- reconstructs MIT Haystack Mark6 recordings as well as FlexBuff recordings
- user, group, permissions, modification time now reflect actual status of on-disk files
- I/O scheduling done in `vbs_fs`
  - serving multiple files and/or multiple users guaranteed optimal
  - `vbs_fs` can be run multiple times but may degrade I/O performance depending on usage pattern

vbs_fs performance

```
flexbuf3:~$ ls -lh gp054_ys_no0150
-rw-r--r-- 0 jops flexbuf 202G Jun 14 01:15 gp054_ys_no0150
```

```
flexbuf3:~$ dd if=/tmp/vbs/gp054_ys_no0150 of=/dev/null
13493+1 records in
13493+1 records out
215895983360 bytes (216 GB) copied, 224.642 s, 961 MB/s
```

vbs_fs performance

```
flexbuf3:~$ ls -lh gp054_ys_no0150
-rw-r--r-- 0 jops flexbuf 202G Jun 14 01:15 gp054_ys_no015
```

```
flexbuf3:~$ dd if=/tmp/vbs/gp054_ys_no0150 of=/dev/null
13493+1 records in
13493+1 records out
215895983360 bytes (216 GB) copied, 224.642 s, 961 MB/s
```

vbs_fs performance

```
flexbuf3:~$ ls -lh gp054_ys_no0150
-rw-r--r-- 0 jops flexbuf 202G Jun 14 01:15 gp054_ys_no0150
```

```
flexbuf3:~$ dd if=/tmp/vbs/gp054_ys_no0150 of=/dev/null
13493+1 records in
13493+1 records out
215895983360 bytes (216 GB) copied, 224.642 s, 961 MB/s
```

vbs_ls, vbs_fs, vbs_rm performance

Indexing a significantly filled FlexBuff takes time

vbs_ls, vbs_fs, vbs_rm performance

Indexing a significantly filled FlexBuff takes time

```
2016-09-16 07:26:17.11: vbs_init: start indexing metadata of 2982 recordings ...
2016-09-16 07:26:30.05: vbs_init: indexing metadata finished
```

vbs_ls, vbs_fs, vbs_rm performance

Indexing a significantly filled FlexBuff takes time

```
2016-09-16 07:26:17.11: vbs_init: start indexing metadata of 2982 recordings ...
2016-09-16 07:26:30.05: vbs_init: indexing metadata finished
```

Takes ~13 seconds on a small FlexBuff, 36% filled of ~100TB

vbs_ls, vbs_fs, vbs_rm performance

Indexing a significantly filled FlexBuff takes time

- Linux kernel caches inode/directory entry information
  - but flushes that cache very easily
  - reloading file meta data from disk is expensive
- if cache was flushed, indexing can take a few minutes
  - ≥ 4 minutes on Onsala's MonsterBuff (>50 HDDs!)
- if cache is 'fresh' expect `vbs_ls -l` to run in ≤ 1 minute
- even compiled C++ code (`vbs_fs`) suffers heavily from this
  - the time is really spent in the O/S getting the meta data

vbs_fs specific option

```
$> vbs_fs ... --quick ...
```

- Only finds the recording names and types, size set to 0 (zero)
  - recording is indexed when it is opened the first time
- starts up much faster
- useful if you know beforehand which scan you want to access
  - e.g. correlator, transfer, off-hand inspection
- drawback: scan must be opened twice
  - first time O/S will see it has zero size (i.e. can't read data)
  - can use good old `touch(1)` to trigger indexing

Thanks for your attention!

Resources

`http://www.jive.eu/~verkout/flexbuff/`

Main index of FlexBuff documents & downloads

`http://www.jive.eu/~verkout/flexbuff/README.vbs`

Explanation, examples, installation instructions – frankly, worth a read

`http://www.jive.eu/~verkout/flexbuff/vbsfs-r45.deb`

`http://www.jive.eu/~verkout/flexbuff/vbs_fs-r45.tar.gz`

Two distribution formats:

1. self-compiling `.deb` package
2. DIY compilation from sources