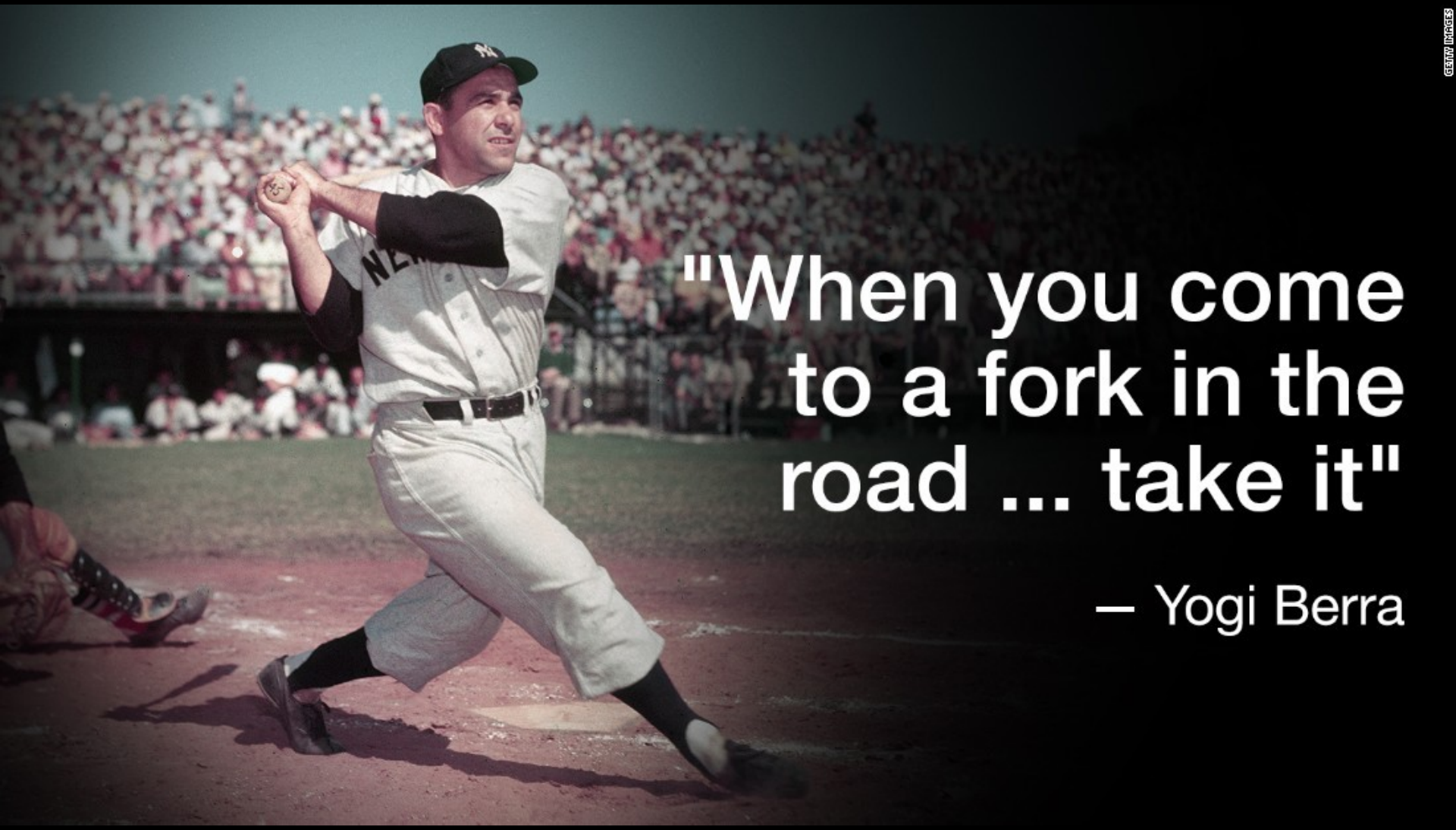

"When you come to a fork in the road ... take it"
— Yogi Berra
GETTY IMAGES

jive5ab

two main developments

Reliability of jive5ab + m5copy

There’s a few issues ...

Reliability of jive5ab + m5copy

disk2file

disk2file – Transfer data from Mark 5 to file

[command list]

Command syntax: disk2file = <destination filename> : [<start byte#>] : [<end byte#>] : <option> ;
Command response: !disk2file = <return code> ;

Query syntax: disk2file? ;
Query response: !disk2file ? <return code> : <status> : <destination filename> : <start byte#> : <current byte#> : <end byte#> : <option> ;

Purpose: Initiates a data transfer from the Mark 5 data disk to an ordinary file.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<dest filename>	literal ASCII	no spaces allowed	See Comments	Default is scan label as reported by 'scan_set', with 'm5a' suffix attached (e.g. 'grf103_ef_scan035.m5a'). Filename must include path if path is not default.
<start byte#>	int | null		See Comments	Absolute byte#; if null, defaults to <start scan_play> position as set and/or reported by scan_set
<end byte#>	int | null		See Comments>	Absolute end byte#; if preceded by '+', increment from <start byte#> by specified value; if null, defaults to <end scan_play> position as set and/or reported by scan_set.
<option>	char	n | w | a	n	n – create file; error if existing file w –erase existing file, if any; create new file. a – create file if necessary, or append to existing file

Monitor-only parameters:

Parameter	Type	Values	Comments
<dest filename>			Destination filename (returned even if filename was defaulted in corresponding 'disk2file' command)
<status>	char	active | inactive	Current status of transfer
<current byte#>	int		Current byte number being transferred

Notes:

1. To abort data transfer: The 'reset=abort' command may be used to abort an active disk2file data transfer. See 'reset' command for details.
2. When <status> is 'inactive', a 'disk2file?' query returns the <dest filename> of the last transferred scan, if any.

Reliability of jive5ab + m5copy

disk2file

disk2file – Transfer data from Mark 5 to file

[command list]

Command syntax: disk2file = <destination filename> : [<start byte#>] : [<end byte#>] : <option> ;
Command response: !disk2file = <return code> ;

Query syntax: disk2file? ;
Query response: !disk2file ? <return code> : <status> : <destination filename> : <start byte#> : <current byte#> : <end byte#> : <option> ;

Purpose: Initiates a data transfer from the Mark 5 data disk to an ordinary file.

Settable parameters:

Parameter	Type	Allowed values	Default	Comments
<dest filename>	literal ASCII	no spaces allowed	See Comments	Default is scan label as reported by 'scan_set', with 'm5a' suffix attached (e.g. 'grf103_ef_scan035.m5a'). Filename must include path if path is not default.
<start byte#>	int | null		See Comments	Absolute byte#; if null, defaults to <start scan_play> position as set and/or reported by scan_set
<end byte#>	int | null		See Comments>	Absolute end byte#; if preceded by '+', increment from <start byte#> by specified value; if null, defaults to <end scan_play> position as set and/or reported by scan_set.
<option>	char	n | w | a	n	n – create file; error if existing file w –erase existing file, if any; create new file. a – create file if necessary, or append to existing file

Monitor-only parameters:

Parameter	Type	Values	Comments
<dest filename>			Destination filename (returned even if filename was defaulted in corresponding 'disk2file' command)
<status>	char	active | inactive	Current status of transfer
<current byte#>	int		Current byte number being transferred

Notes:

1. To abort data transfer: The 'reset=abort' command may be used to abort an active disk2file data transfer. See 'reset' command for details.
2. When <status> is 'inactive', a 'disk2file?' query returns the <dest filename> of the last transferred scan, if any.

Reliability of jive5ab + m5copy

Many (if not all) transfers will automatically stop

- generally this is A Good Thing
- but never know *why* they stopped:
  - can’t tell succes from #FAIL ...
- `m5copy` cannot tell wether to resume or not

User, FS, ...

disk2file=/path/to/file
!disk2file=1*

⋮

disk2file?
!disk2file? 0 : /path/to/file : 1337

⋮

disk2file?
!disk2file? 0 : inactive : /path/to/file;

‘initiated but not completed’

jive5ab

disk2file ▯ (null)

disk2file ▯

disk2file_status_type

- start byte
- current byte
- ...

(transfer terminates,
frees resources)

disk2file ▯ (null)

Reliability of jive5ab + m5copy

Many (if not all) transfers will automatically stop

- generally this is A Good Thing
- but never know *why* they stopped:
  - can't tell succes from #FAIL ...
- `m5copy` cannot tell wether to resume or not
- last "`<transfer>?`" status should always be available

User, FS, ...

disk2file=/path/to/file
!disk2file=1*

⋮

disk2file?
!disk2file? 0 : /path/to/file : 1337

⋮

disk2file?
!disk2file? 0 : inactive : ... : 2048;

‘initiated but not completed’

jive5ab

disk2file []

disk2file_status_type
- active (false)
- start byte
- ...

disk2file []

disk2file_status_type
- active (true)
- start byte
- ...

(terminates)

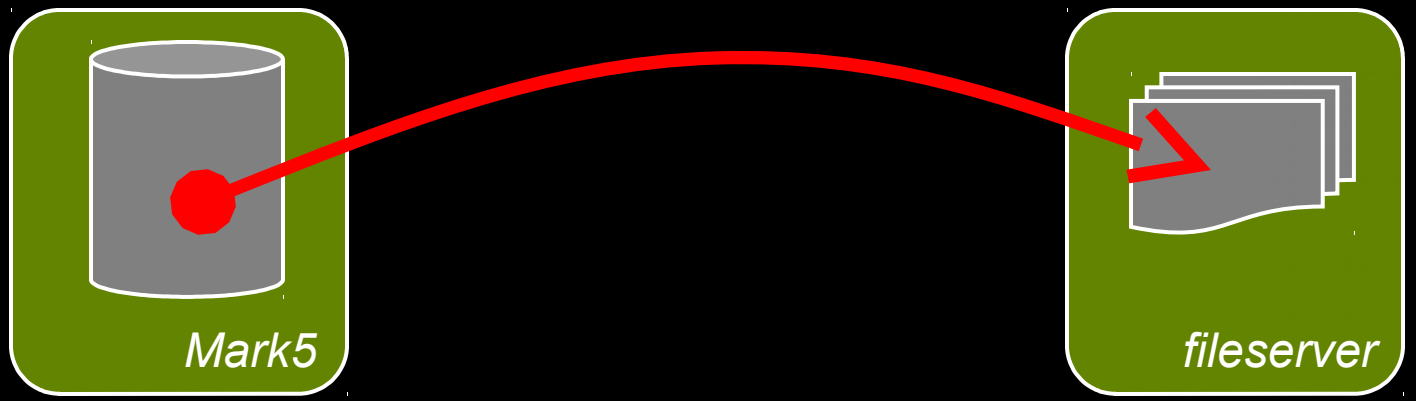
disk2file []

disk2file_status_type
- active (false)
- start byte
- ...

Reliability of jive5ab + m5copy

Many (if not all) transfers will automatically stop

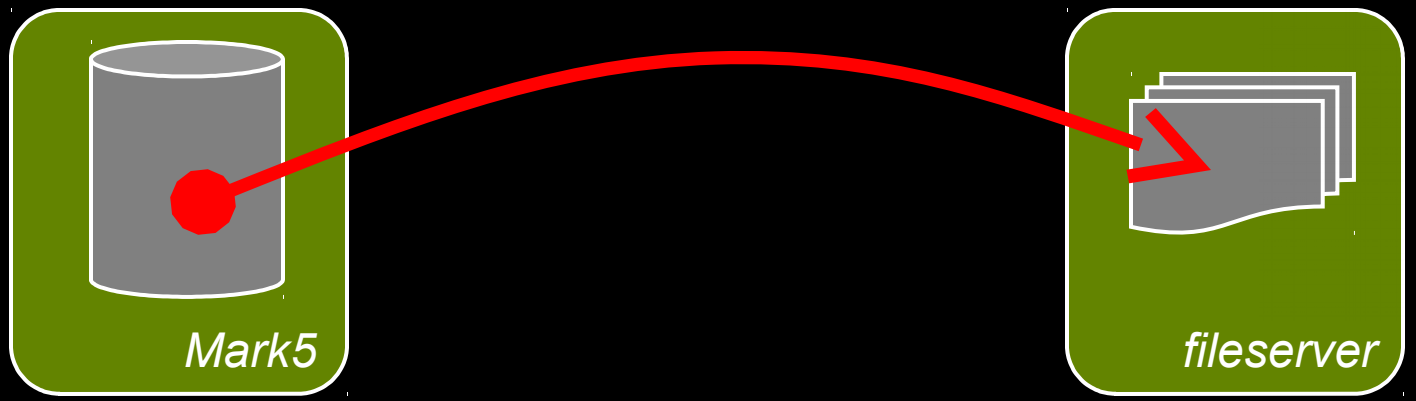
- generally this is A Good Thing
- but never know *why* they stopped:
  - can't tell succes from #FAIL ...
- `m5copy` cannot tell wether to resume or not
- last "`<transfer>?`" status always available
- has to be implemented for almost all transfers
  - work in progress



jive5ab VSI/S commands

```
$> m5copy mk5://... file://...
Progress |===>   | 110MB/s 53.1%
```

control computer



```
$> m5copy mk5://... file://...
Progress |===>   | 110MB/s 53.1%
```

control computer

Reliability of jive5ab + m5copy

Nothing happens if control connection dies ...

- generally this is A Good Thing
- except when it isn't ...
- maybe impossible for `m5copy` to clean up
  - leaking file descriptors, memory
  - keeping data port open!

Reliability of jive5ab + m5copy

Nothing happens if control connection dies ...

- `jive5ab` implement *transient* control connections
- if transient connection closes:
  - transfers started from it are terminated
  - resources freed
  - sockets released
  - instant win!
- `m5copy` updated to make use of this feature

Keeping time inside jive5ab

```
#include <time.h>

struct timespec {
   // Integer seconds since epoch
   // and nanosecond within current second
   time_t tv_sec;
   long   tv_nsec;
};

typedef unsigned int samplerate_type;
```

Keeping time inside jive5ab

nano-second resolution is fine for Just Recording (tm) ...

However ... `jive5ab` can do more

- generate data w/ valid time stamps
- decode time stamps (e.g. `scan_check?`)
- convert data format X to VDIF
  - must recode timestamp X to VDIF

Keeping time inside jive5ab

1024 Mbps

$$\delta t_{1\ bit} = 1 / 1024 \times 10^6$$
$$= 0.9765625\ ns$$

Cannot properly represent sample time stamps!

Keeping time inside jive5ab

```
#include           >

struct jive5ab_
   // Integer                epoch
   // and         secon     n current
second
   t       v_sec;
   I       tv_psec;
};
```

$\delta t_{1\ \mathrm{bit}}$ @ 1024 Mbps = 1 / 1024×10^{6} = 976.5625 ps

Keeping time inside jive5ab

Sample rates + clocks are *rationals*

$$\frac{n \text{ ticks}}{\text{second}}$$

$$\frac{n \text{ samples}}{\text{second}}$$

Keeping time inside jive5ab

```
#include <time.h>
#include <boost/rational.hpp>

struct jive5ab_timespec {
   // Integer seconds since epoch
   // plus fraction of second
   time_t       tv_sec;
   rational<uint64_t>  tv_subsec;
};

typedef rational<uint64_t> samplerate_type;
```

Keeping time inside jive5ab

Touching all code that deals with time and/or sample rate and or data frame length (`mode=`, `scan_check?`, . . .):

- mostly done

Return on investment:

- correctly describes all time stamps:
  - “integer seond + frame number within second”
  - even MarkIV/VLBA time codes are rationals
- describe 1-channel 1-bit 128 Gbps data stream
- no loss of precision, other than in conversion to:
  - double
  - text on screen

Keeping time inside jive5ab

Sample rates + clocks are *rationals*

$$\frac{n \text{ ticks}}{period}$$

$$\frac{n \text{ samples}}{period}$$

period is integer number of seconds, not necessarily ‘1’

Keeping time inside jive5ab

Touching all code that deals with time and/or sample rate and or data frame length (`mode=`, `scan_check?`, . . .):

- mostly done

Return on investment:

- correctly describes all time stamps:
  - “integer seond + frame number within second”
  - even MarkIV/VLBA time codes are rationals
- describe 1-channel 1-bit 128 Gbps data stream
- no loss of precision, other than in conversion to:
  - double
  - text on screen
- describe ‘funny’ rates like 1000 frames / 3 seconds

Keeping time inside jive5ab

Experimental version of `jive5ab`:

- under test @ HartRAO
- survived smoke test (record under FS control, `m5copy`, ...)
- used for e-VLBI Feb 2, 2016 (last week)
- generate ‘funny’ VDIF:
  - `mode = VDIF_1000-1/3-1-1`
  - (1 Mbit / 3 seconds, 1 channel, 1 bit)

```
file_check? : : /tmp/vdif-1over3-1ch-1bit.vdif

!file_check? 0 : VDIF : 1 : 2016y033d10h04m45.0000s
               : 185.928s : 0.333333Mbps : 0 : 1032 ;
```

Thanks for your attention!