pv allows a user to see the progress of data through a pipeline, by giving information such as time elapsed, percentage completed (with progress bar), current throughput rate, total data transferred, and ETA.

Here's a nice list of cool ways you can use pv

1) Simulate typing

```
echo "You can simulate on-screen typing just like in the movies" | pv -qL 10
```

This will output the characters at 10 per second.

2) Monitor progress of a command

```
pv access.log | gzip > access.log.gz
```

Pipe viewer is a terminal-based tool for monitoring the progress of data through a pipeline. It can be inserted into any normal pipeline between two processes to give a visual indication of how quickly data is passing through, how long it has taken, how near to completion it is, and an estimate of how long it will be until completion.

3) live ssh network throughput test

```
yes | pv | ssh $host "cat > /dev/null"
```

connects to host via ssh and displays the live transfer speed, directing all transferred data to /dev/null

4) copy working directory and compress it on-the-fly while showing progress

```
tar -cf - . | pv -s $(du -sb . | awk '{print $1}') | gzip > out.tgz
```

What happens here is we tell tar to create “-c” an archive of all files in current dir “.” (recursively) and output the data to stdout “-f -“. Next we specify the size “-s” to pv of all files in current dir. The “du -sb . | awk ?{print $1}?” returns number of bytes in current dir, and it gets fed as “-s” parameter to pv. Next we gzip the whole content and output the result to out.tgz file. This way “pv” knows how much data is still left to be processed and shows us that it will take yet another 4 mins 49 secs to finish.

5) Copy a file using pv and watch its progress

```
pv sourcefile > destfile
```

pv allows a user to see the progress of data through a pipeline, by giving information such as time elapsed, percentage completed (with progress bar), current throughput rate, total data transferred, and ETA. (man pv)

6) Another live ssh network throughput test

```
pv /dev/zero|ssh $host 'cat > /dev/null'
```

connects to host via ssh and displays the live transfer speed, directing all transferred data to /dev/null

7) dd with progress bar and statistics

```
sudo dd if=/dev/sdc bs=4096 | pv -s 2G | sudo dd bs=4096 of=~/USB_BLACK_BACKUP.IMG
```

This command utilizes ‘pv’ to show dd’s progress.

Notes on use with dd:

— dd block size (bs=…) is a widely debated command-line switch and should usually be between 1024 and 4096. You won’t see much performance improvements beyond 4096, but regardless of the block size, dd will transfer every bit of data.

— pv's switch, '-s' should be as close to the size of the data source as possible.

— dd's out file, 'of=…' can be anything as the data within that file are the same regardless of the filename / extension.

8) [re]verify a disc with very friendly output

```
dd if=/dev/cdrom | pv -s 700m | md5sum | tee test.md5
```

[re]verify those burned CD's early and often – better safe than sorry –

at a bare minimum you need the good old `dd` and `md5sum` commands,

but why not throw in a super "user-friendly" progress gauge with the `pv` command –

adjust the "-s" "size" argument to your needs – 700 MB in this case,

and capture that checksum in a "test.md5" file with `tee` – just in-case for near-future reference.

uber-bonus ability – positively identify those unlabeled mystery discs –

for extra credit, what disc was used for this sample output?

9) time how fast the computer reads from /dev/zero

```
pv /dev/zero > /dev/null
```

my stats 217GB 0:00:38 [4,36GB/s]