fs_hasher工具介绍

工具列表

```
hf-stat  hf-simdex  fs-hasher
```

hf-stat工具

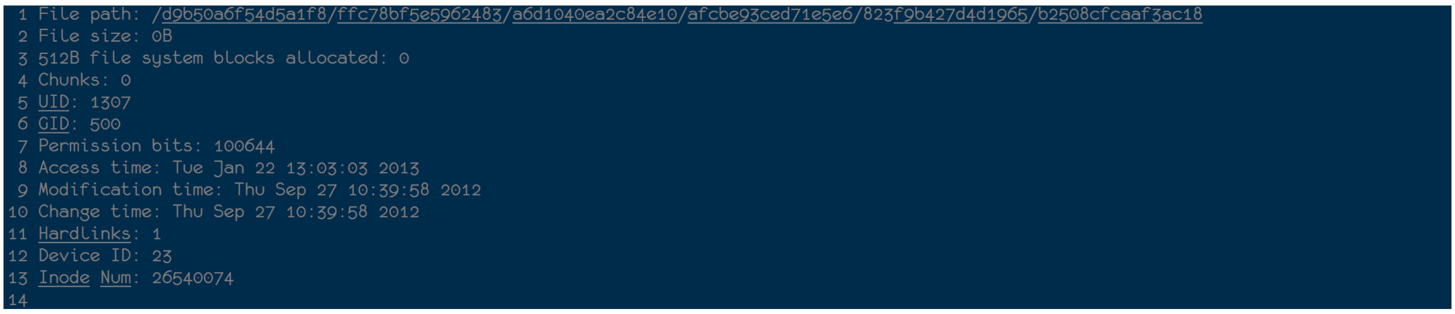
使用方法

以**-f**参数为例

```
Usage: ./hf-stat -f hashfile
```

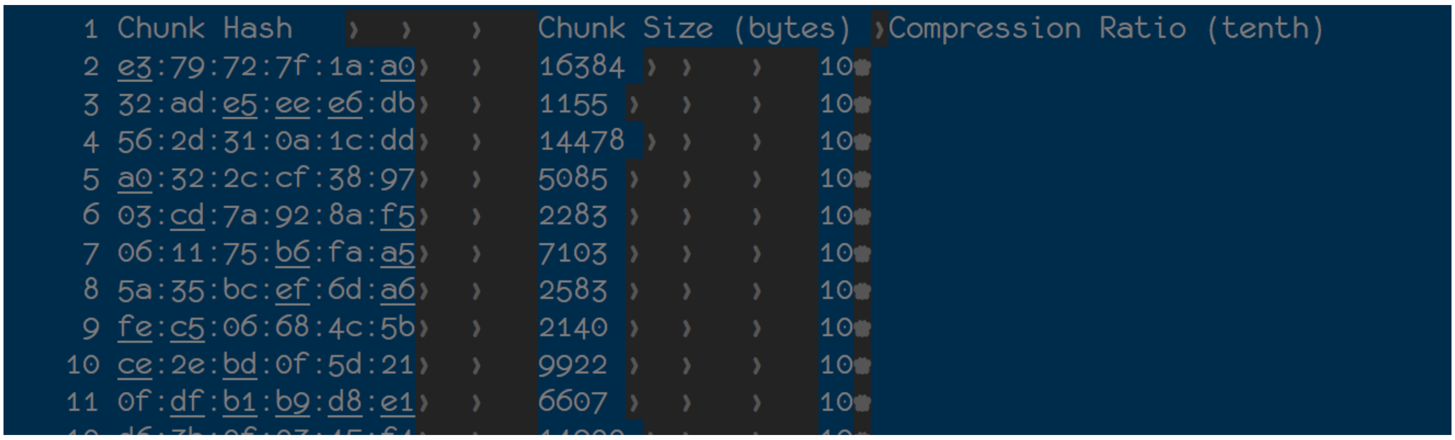
参数用途

-f : 显示**.hash.anon**文件中存储的文件信息列表



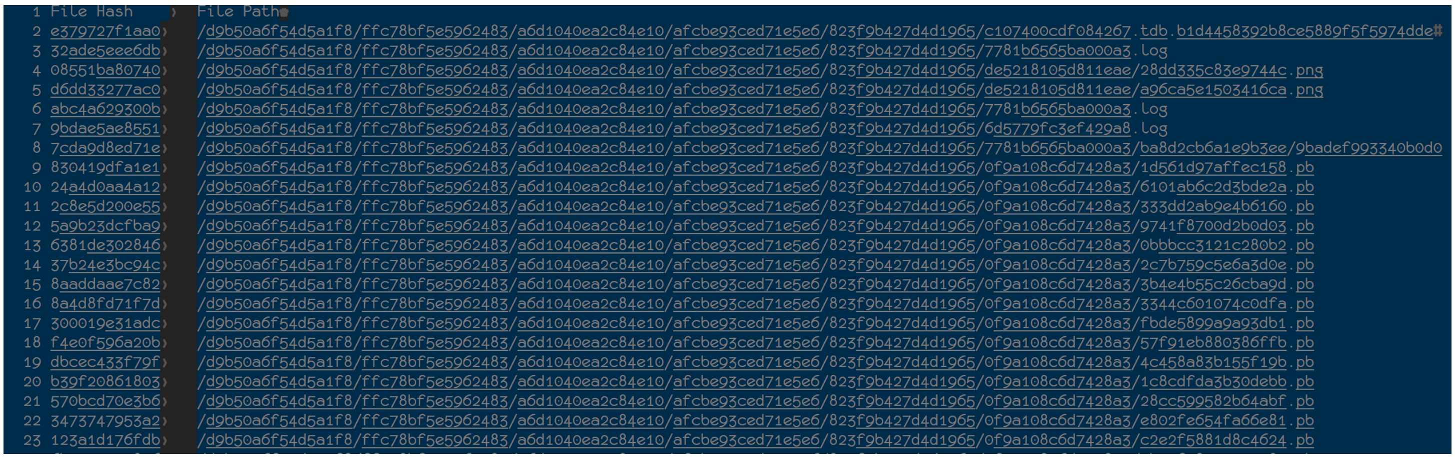
```
File path: /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/b2508cfcaaf3ac18
File size: 0B
512B file system blocks allocated: 0
Chunks: 0
UID: 1307
GID: 500
Permission bits: 100644
Access time: Tue Jan 22 13:03:03 2013
Modification time: Thu Sep 27 10:39:58 2012
Change time: Thu Sep 27 10:39:58 2012
Hardlinks: 1
Device ID: 23
Inode Num: 26540074

```

-h : 显示哈希文件中各文件的数据块的信息：数据块哈希值(Chunk Hash)、数据块的长度(Chunk Size)、压缩率(Compression Ratio)



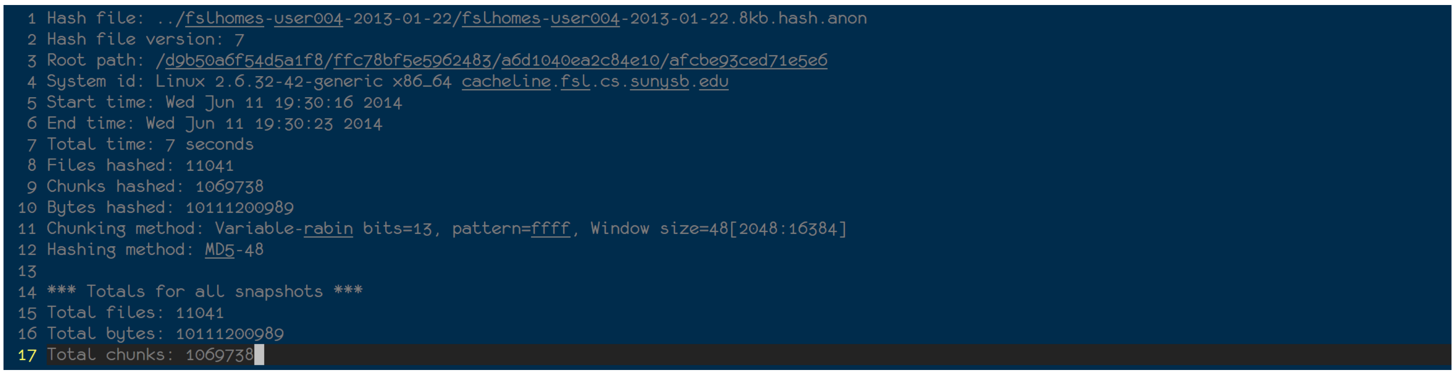
```
Chunk Hash              Chunk Size (bytes)  Compression Ratio (tenth)
e3:79:72:7f:1a:a0       16384               10
32:ad:e5:ee:e6:db       1155                10
56:2d:31:0a:1c:dd       14478               10
a0:32:2c:cf:38:97       5085                10
03:cd:7a:92:8a:f5       2283                10
06:11:75:b6:fa:a5       7103                10
5a:35:bc:ef:6d:a6       2583                10
fe:c5:06:68:4c:5b       2140                10
ce:2e:bd:0f:5d:21       9922                10
0f:df:b1:b9:d8:e1       6607                10
```

-w：显示所包含的文件哈希(File Hash)和相应的文件路径(File Path)



```
File Hash       File Path
e379727f1aa0    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/c107400cdf084267.tdb.b1d4458392b8ce5889f5f5974dde#
32ade5eee6db    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/7781b6565ba000a3.log
08551ba80740    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/de5218105d811eae/28dd335c83e9744c.png
d6dd33277ac0    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/de5218105d811eae/a96ca5e1503416ca.png
abc4a629300b    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/7781b6565ba000a3.log
9bdae5ae8551    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/6d5779fc3ef429a8.log
7cda9d8ed71e    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/7781b6565ba000a3/ba8d2cb6a1e9b3ee/9badef993340b0d0
830419dfa1e1    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/1d561d97affec158.pb
24a4d0aa4a12    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/6101ab6c2d3bde2a.pb
2c8e5d200e55    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/333dd2ab9e4b6160.pb
5a9b23dcfba9    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/9741f8700d2b0d03.pb
6381de302846    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/0bbbcc3121c280b2.pb
37b24e3bc94c    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/2c7b759c5e6a3d0e.pb
8aaddaae7c82    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/3b4e4b55c26cba9d.pb
8a4d8fd71f7d    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/3344c601074c0dfa.pb
300019e31adc    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/fbde5899a9a93db1.pb
f4e0f596a20b    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/57f91eb880386ffb.pb
dbcec433f79f    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/4c458a83b155f19b.pb
b39f20861803    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/1c8cdfda3b30debb.pb
570bcd70e3b6    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/28cc599582b64abf.pb
3473747953a2    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/e802fe654fa66e81.pb
123a1d176fdb    /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6/823f9b427d4d1965/0f9a108c6d7428a3/c2e2f5881d8c4624.pb
```

-t：显示**.hash.anon**文件的总体信息



```
Hash file: ../fslhomes-user004-2013-01-22/fslhomes-user004-2013-01-22.8kb.hash.anon
Hash file version: 7
Root path: /d9b50a6f54d5a1f8/ffc78bf5e5962483/a6d1040ea2c84e10/afcbe93ced71e5e6
System id: Linux 2.6.32-42-generic x86_64 cacheline.fsl.cs.sunysb.edu
Start time: Wed Jun 11 19:30:16 2014
End time: Wed Jun 11 19:30:23 2014
Total time: 7 seconds
Files hashed: 11041
Chunks hashed: 1069738
Bytes hashed: 10111200989
Chunking method: Variable-rabin bits=13, pattern=ffff, Window size=48[2048:16384]
Hashing method: MD5-48

*** Totals for all snapshots ***
Total files: 11041
Total bytes: 10111200989
Total chunks: 1069738
```

hf-simdex

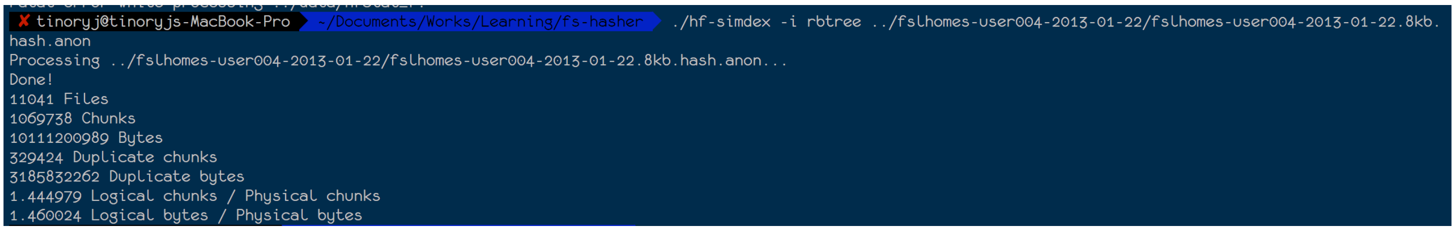
使用方法：

```
Usage: ./hf-simdex [-i <list|rbtree>] [-s <hash>] hashfile ...
```

参数用途

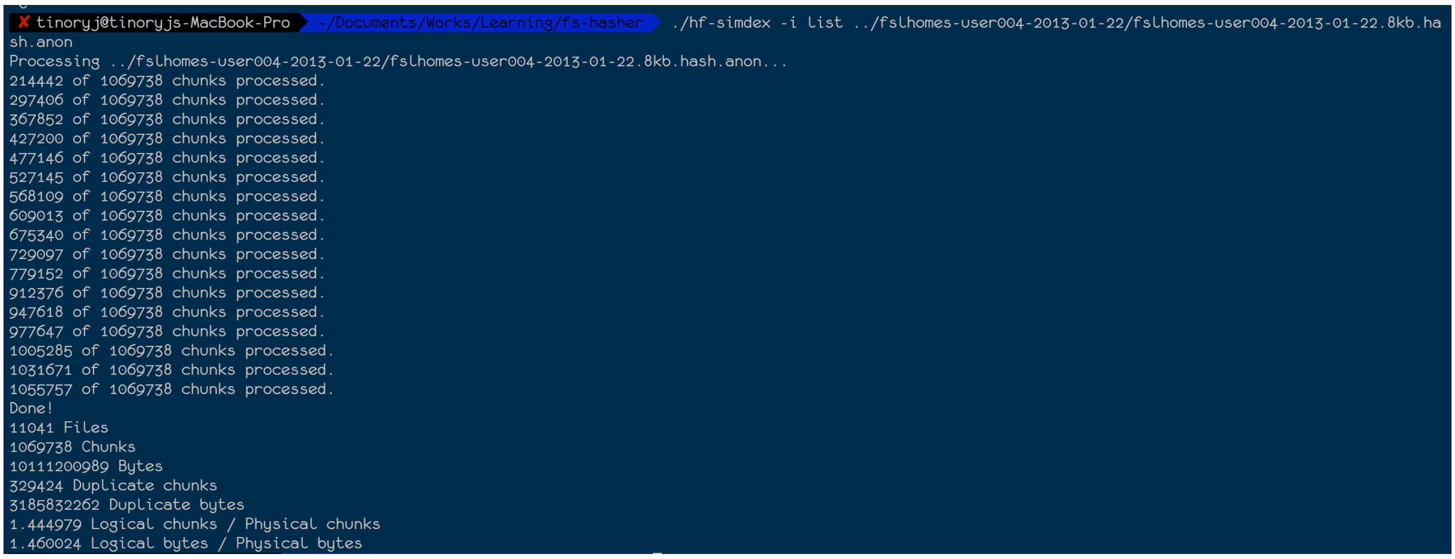
-i：指定操作方法，默认值为**rbtree**。

若添加参数为**rbtree**，效果如下，效果看上去非常像**hf-stat -t**的效果，但多出了重复数据块的统计信息。



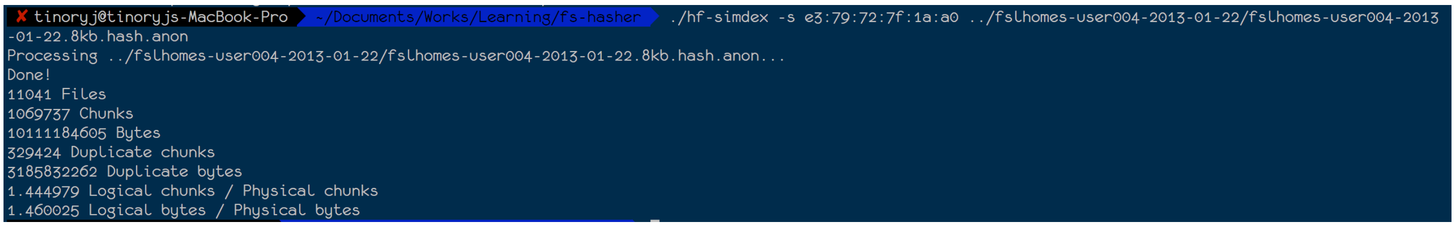
```
✘ tinoryj@tinoryjs-MacBook-Pro  ~/Documents/Works/Learning/fs-hasher  ./hf-simdex -i rbtree ../fslhomes-user004-2013-01-22/fslhomes-user004-2013-01-22.8kb.
hash.anon
Processing ../fslhomes-user004-2013-01-22/fslhomes-user004-2013-01-22.8kb.hash.anon...
Done!
11041 Files
1069738 Chunks
10111200989 Bytes
329424 Duplicate chunks
3185832262 Duplicate bytes
1.444979 Logical chunks / Physical chunks
1.460024 Logical bytes / Physical bytes
```

添加参数为**list**，效果与**rbtree**相同，多输出了中间过程，但运行速度明显较**rbtree**慢10倍以上。



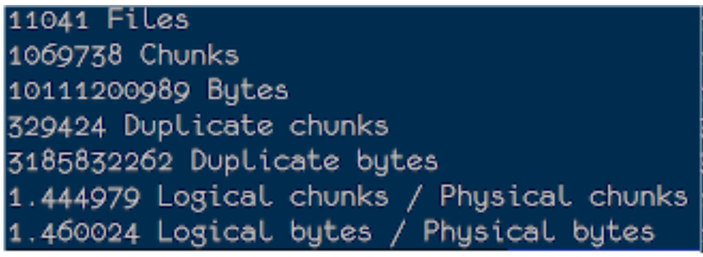
```
✘ tinoryj@tinoryjs-MacBook-Pro  ~/Documents/Works/Learning/fs-hasher  ./hf-simdex -i list ../fslhomes-user004-2013-01-22/fslhomes-user004-2013-01-22.8kb.ha
sh.anon
Processing ../fslhomes-user004-2013-01-22/fslhomes-user004-2013-01-22.8kb.hash.anon...
214442 of 1069738 chunks processed.
297406 of 1069738 chunks processed.
367852 of 1069738 chunks processed.
427200 of 1069738 chunks processed.
477146 of 1069738 chunks processed.
527145 of 1069738 chunks processed.
568109 of 1069738 chunks processed.
609013 of 1069738 chunks processed.
675340 of 1069738 chunks processed.
729097 of 1069738 chunks processed.
779152 of 1069738 chunks processed.
912376 of 1069738 chunks processed.
947618 of 1069738 chunks processed.
977647 of 1069738 chunks processed.
1005285 of 1069738 chunks processed.
1031671 of 1069738 chunks processed.
1055757 of 1069738 chunks processed.
Done!
11041 Files
1069738 Chunks
10111200989 Bytes
329424 Duplicate chunks
3185832262 Duplicate bytes
1.444979 Logical chunks / Physical chunks
1.460024 Logical bytes / Physical bytes
```

-s：从统计中剔除指定的hash值对应的数据块。

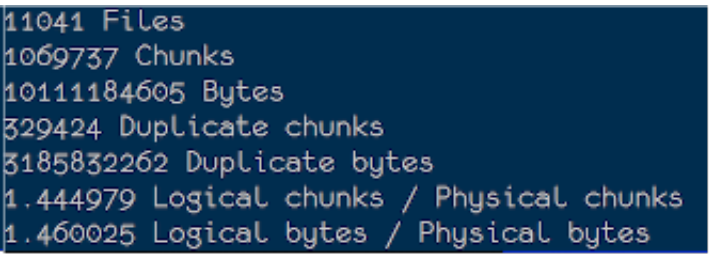


```
✘ tinoryj@tinoryjs-MacBook-Pro  ~/Documents/Works/Learning/fs-hasher  ./hf-simdex -s e3:79:72:7f:1a:a0 ../fslhomes-user004-2013-01-22/fslhomes-user004-2013
-01-22.8kb.hash.anon
Processing ../fslhomes-user004-2013-01-22/fslhomes-user004-2013-01-22.8kb.hash.anon...
Done!
11041 Files
1069737 Chunks
10111184605 Bytes
329424 Duplicate chunks
3185832262 Duplicate bytes
1.444979 Logical chunks / Physical chunks
1.460025 Logical bytes / Physical bytes
```

示例中选择剔除了hash值为**e3:79:72:7f:1a:a0**的数据块，可以看到数据块总数从**1069738**减少到了**1069737**，由于剔除的数据块没有重复出现，因此重复数据块统计结果未变。对比结果如下所示：



```
11041 Files
1069738 Chunks
10111200989 Bytes
329424 Duplicate chunks
3185832262 Duplicate bytes
1.444979 Logical chunks / Physical chunks
1.460024 Logical bytes / Physical bytes
```



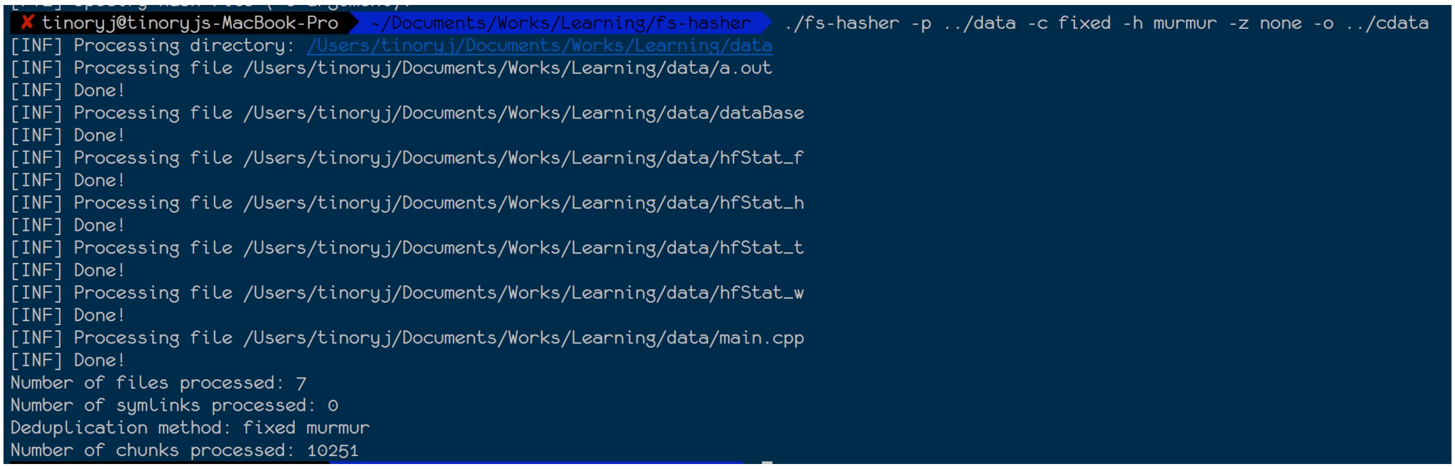
```
11041 Files
1069737 Chunks
10111184605 Bytes
329424 Duplicate chunks
3185832262 Duplicate bytes
1.444979 Logical chunks / Physical chunks
1.460025 Logical bytes / Physical bytes
```

fs-hasher

使用方法

```
Usage: ./fs-hasher [-mdqMeb] -p <path> {-c <fixed|variable> [-C <chunking op
tions>] -h <murmur> -z <none|zlib-def>  -o <hashfile> | -F <configfile>}
```

必须使用参数包括-p，-c，-h，-z，-o；最简单的使用方法如下所示：



```
✗ tinoryj@tinoryjs-MacBook-Pro  ~/Documents/Works/Learning/fs-hasher  ./fs-hasher -p ../data -c fixed -h murmur -z none -o ../cdata
[INF] Processing directory: /Users/tinoryj/Documents/Works/Learning/data
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/a.out
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/dataBase
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_f
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_h
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_t
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_w
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/main.cpp
[INF] Done!
Number of files processed: 7
Number of symlinks processed: 0
Deduplication method: fixed murmur
Number of chunks processed: 10251
```

参数用途

基本参数

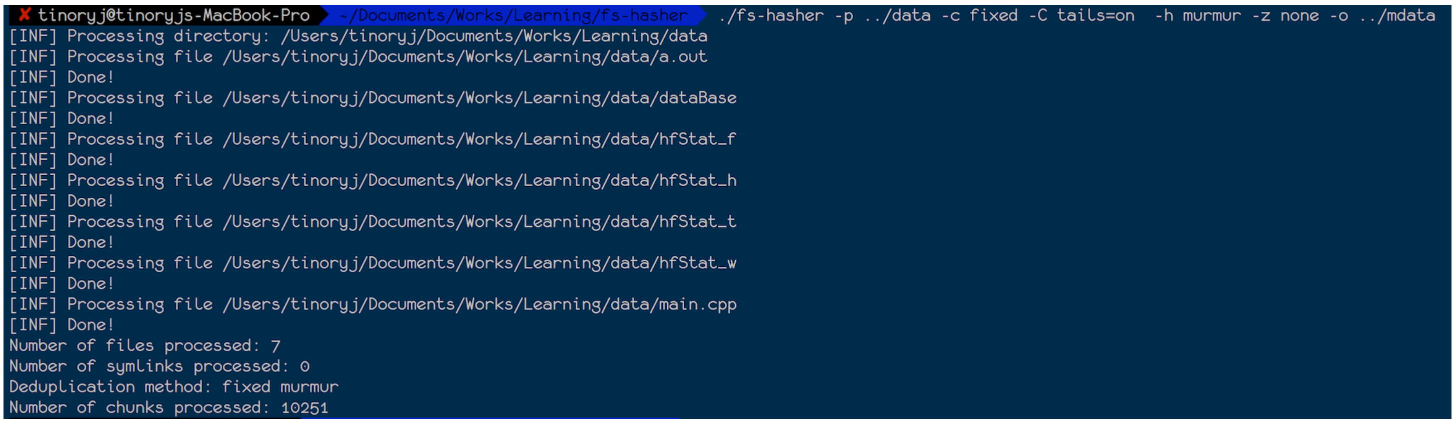
-p：指定要处理的文件目录，相对路径、绝对路径均可。

-c：设置分块选项，有两个参数：fixed、variabe。使用方法为：**-c fixed 或 -c variable**

当使用**-c fixed**时，可以通过附加**-C**参数设定属性：
csize=<csize>
tails=<on|off>
用法如下所示：

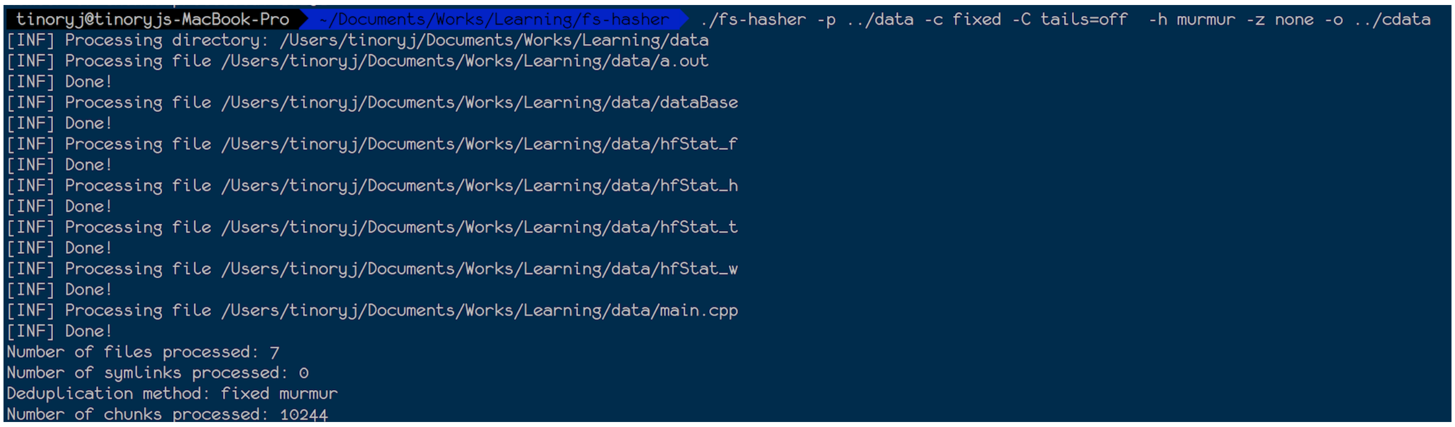
```
-C csize=xxx tails=on 或 tails=off
```

tails开启时，对相同文件目录经行处理时结果如下：



```
✘ tinoryj@tinoryjs-MacBook-Pro  ~/Documents/Works/Learning/fs-hasher  ./fs-hasher -p ../data -c fixed -C tails=on  -h murmur -z none -o ../mdata
[INF] Processing directory: /Users/tinoryj/Documents/Works/Learning/data
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/a.out
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/dataBase
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_f
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_h
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_t
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_w
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/main.cpp
[INF] Done!
Number of files processed: 7
Number of symlinks processed: 0
Deduplication method: fixed murmur
Number of chunks processed: 10251
```

tails关闭时，对相同文件目录经行处理时结果如下：



```
tinoryj@tinoryjs-MacBook-Pro  ~/Documents/Works/Learning/fs-hasher  ./fs-hasher -p ../data -c fixed -C tails=off  -h murmur -z none -o ../cdata
[INF] Processing directory: /Users/tinoryj/Documents/Works/Learning/data
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/a.out
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/dataBase
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_f
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_h
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_t
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_w
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/main.cpp
[INF] Done!
Number of files processed: 7
Number of symlinks processed: 0
Deduplication method: fixed murmur
Number of chunks processed: 10244
```

其中tails=off时处理的数据块较少。

当使用**-c varibale**时，可以通过附加**-C**参数设定属性：
[:algo=<random|simple|rabin]
[:win_size=<window size>]
[:match_bits=<no of bits to check in the fingerprint>]
[:pattern=<pattern of bits to match>]
[:min_csize=<csize>][:max_csize=<csize>]
[:tails=<on|off>]

使用方法与fixed中**-C**参数的使用方法一致。

-h : 指定hash方法，观察了一下貌似只支持murmurhash。

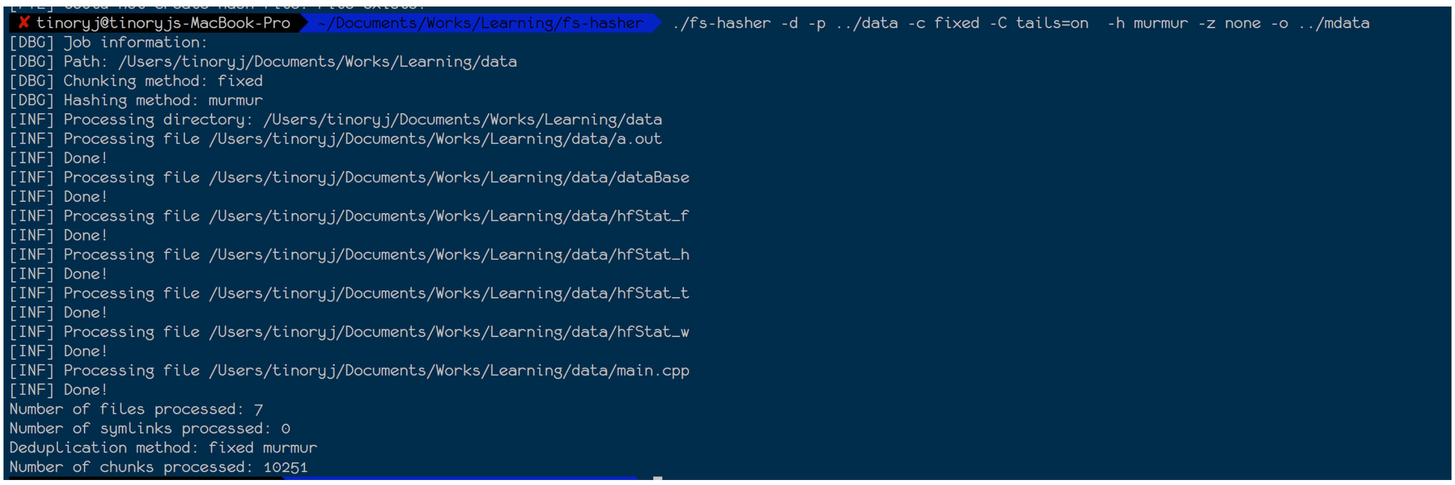
-z : 指定生成文件的压缩方法，可选择不进行压缩(none)或者zlib-def压缩。

-o : 指定所要生成的hash文件的路径、文件名。

其他参数

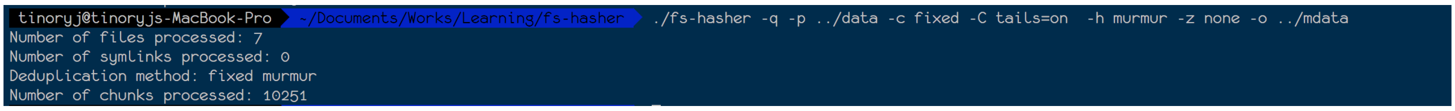
-m : 忽律所有挂载在指定的路径下的文件。

-d : 输出调试信息，效果如下所示，会以[DBG]开头输出。



```
✘ tinoryj@tinoryjs-MacBook-Pro  ~/Documents/Works/Learning/fs-hasher  ./fs-hasher -d -p ../data -c fixed -C tails=on  -h murmur -z none -o ../mdata
[DBG] Job information:
[DBG] Path: /Users/tinoryj/Documents/Works/Learning/data
[DBG] Chunking method: fixed
[DBG] Hashing method: murmur
[INF] Processing directory: /Users/tinoryj/Documents/Works/Learning/data
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/a.out
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/dataBase
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_f
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_h
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_t
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/hfStat_w
[INF] Done!
[INF] Processing file /Users/tinoryj/Documents/Works/Learning/data/main.cpp
[INF] Done!
Number of files processed: 7
Number of symlinks processed: 0
Deduplication method: fixed murmur
Number of chunks processed: 10251
```

-q : 静默执行，避免输出执行过程，仅输出结果，效果如下所示：



```
tinoryj@tinoryjs-MacBook-Pro  ~/Documents/Works/Learning/fs-hasher  ./fs-hasher -q -p ../data -c fixed -C tails=on  -h murmur -z none -o ../mdata
Number of files processed: 7
Number of symlinks processed: 0
Deduplication method: fixed murmur
Number of chunks processed: 10251
```

-M : 使用mmap来读取文件

-e : 加入该参数会使得程序在读取文件失败后直接退出，若不添加该参数，程序将继续运行。

-b : 扫描块设别（默认会跳过）

> block device是块设备 char devicej是字符设备;
> I/O设备大致分为两类：块设备和字符设备。块设备将信息存储在固定大小的块中，每个块都有自己的地址。数据块的大小通常在512字节到32768字节之间。块设备的基本特征是每个块都能独立于其它块而读写。磁盘是最常见的块设备。
> 　　在大多数的UNIX操作系统中，块设备只支持以块为单位的访问方式，如磁盘等.KYLIN支持以字符方式来访问块设备，即支持以字符为单位来读写磁盘等块设备。所以在/dev目录中的块设备，如磁盘等，均以字符设备的外观出现。所以，字符设备和块设备的区别主要体现在KYLIN内核中的管理方式，操作方式和内核/设备驱动接口上。

-u : 输出帮助信息。

-F : 从设置文件中读取数据块和hash文件设置的相关信息。