# ylog 使用手册

（工具组.天津 ）

| **abbreviation** | **Description** |
|---|---|
| ylog | your log |
| | |
| | |
| | |
| | |
| | |

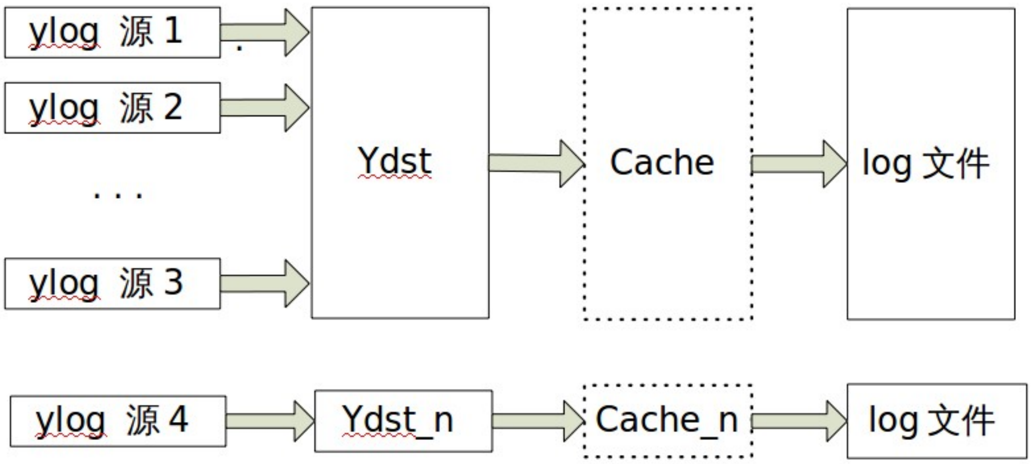
| Revision | Author | Date | Brief |
|---|---|---|---|
| 1.0 | Luther Ge | 2016/02/14 | Initial Draft |
| 1.1 | Yanli Chen | 2016/03/02 | Add ylog use help |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# 1. ylog 设计框架

ylog(your log)是可定制的后台常驻服务程序，用于完成 log 收集、智能解析、行为统计等

## 1.1 ylog 结构设计

ylog 服务程序被设计为 3 部分: ylog 源，ydst 和 cache
ylog 源：数据输入源
ydst： 数据输出
cache：用于缓存输出数据



### 1.1.1 ylog 源

ylog 源重在描述数据输入，比如输入类型， token 和重启时间，当前支持：普通文件、 socket、popen 和 inotify 监测这 4 种类型输入

ylog 参考形式如下：

```
{
.name = "android_main",
.type = FILE_POPEN,
.file = "logcat -v threadtime -b main",
.ydst=&ydst[OS_YDST_TYPE_ANDROID+OS_YDST_TYPE_BASE],
.mode=YLOG_READ_MODE_BLOCK | YLOG_READ_LEN_MIGHT_ZERO | \
YLOG_READ_MODE_BLOCK_RESTART_ALWAYS,
.restart_period = 2000,
.fp_array = NULL,
.id_token = "A0",
.id_token_len = 2,
.id_token_filename = "main.log",
},
```

### 1.1.2 ydst

ydst 专注在数据输出，从 ylog 描述的输入获取到的数据将被传递给 ydst，ydst 将对数据进行 segment 大小和段个数控制，同时如果数据流量比较大，可以考虑为 ydst 挂靠 cache，来缓解大数据、大吞吐下数据因为磁盘短暂效率问题而出现的数据丢失

ydst 参考形式如下：

```
[OS_YDST_TYPE_ANDROID] = {
.file = "android/",
.file_name = "android.log",
.max_segment = 30,
.max_segment_size = 50*1024*1024,
.cache = &os_cacheline[OS_YDST_TYPE_ANDROID],
},
```

### 1.1.3 cache 缓存

cache 专注在 ydst 数据输出如何被缓存，随后择机写入磁盘存储，为了保证尽快回写磁盘， cache 有一个 timeout 属性，用来定义一种行为： cache line 如果在 timeout 内仍未填满，那么 cache line 中数据将被强行 flush 回磁盘

```
[OS_YDST_TYPE_ANDROID] = {
.size = 512 * 1024,
.num = 4,
.timeout = 1000, /* ms */
.debuglevel = CACHELINE_DEBUG_CRITICAL,
}
```

# 2. ylog 模块

ylog 是在 init.rc 中启动的一个用户空间的服务程序，persist.ylog.enabled 属性值可以辅助控制开机后 ylog 默认启动行为。ylog 模块不仅支持多个 ylog 源并行输入，还支持多个 ydst 并行输出。根据 ylog 源数据结构，指定 ydst 输出，并设定 ydst 输出是否挂靠 cache、重启时间间隔、达到段上限值后是否进行覆盖等

## 2.1 ylog 输入源与 ydst 对应列表

| | |
|---|---|
| [ylog] android_main | [ydst] android |
| [ylog] android_radio | |
| [ylog] android_system | |
| [ylog] android_events | |
| [ylog] android_crash | |
| [ylog] kernel | [ydst] kernel |
| [ylog] sys_info | [ydst] sys_info |
| [ylog] tcpdump | [ydst] tcpdump |
| [ylog] hcidump | [ydst] hcidump |
| [ylog] ylog_debug | [ydst] ylog_debug |
| [ylog] info | [ydst] info |
| [ylog] journal | [ydst] journal |
| [ylog] tracer | [ydst] tracer |
| [ylog] traces | [ydst] traces |
| [ylog] socket | [ydst] socket/open/ |
| [ylog] sys_info_manual | [ydst] sys_info_manual |

## 2.2 ylog 输入源

**1、android 类**

包含 android_main、 android_radio、 android_system、 android_events、 android_crash 等五类 log，共同输出到一个 ydst 中，为了提高磁盘效率，此 ydst 默认挂靠 2M 大小的 cache

**2、kenel**

收集 kernel log，kernel ydst 默认挂靠 1M 大小的 cache

**3、sys_info**

定期（2 分钟）收集系统相关 log，具体包括以下信息：

/proc/slabinfo
/proc/buddyinfo
/proc/zoneinfo
/proc/vmstat
/proc/vmallocinfo
/proc/pagetypeinfo
/sys/module/lowmemorykiller/parameters/adj
/sys/module/lowmemorykiller/parameters/minfree
/proc/wakelocks
/d/wakeup_sources
/sys/class/backlight/sprd_backlight/brightness
/sys/kernel/debug/binder/failed_transaction_log
/sys/kernel/debug/binder/transaction_log
/sys/kernel/debug/binder/transactions
/sys/kernel/debug/binder/stats
/sys/kernel/debug/binder/state
/sys/kernel/debug/sprd_debug/cpu/cpu_usage

**4、tcpdump**

用于收集 ap cap log，tcpdump ydst 默认挂靠 1M 大小的 cache

## 5、hcidump

用于收集 hci bt log，hcidump ydst 默认挂靠 1M 大小的 cache

## 6、ylog_debug

用于收集 log 相关的调试信息，包含 log 速率，空间使用、运行状态等信息，每 20 分钟统计一次

/system/bin/ylog_cli ylog<br>
/system/bin/ylog_cli speed<br>
/system/bin/ylog_cli space<br>
getprop ylog.killed

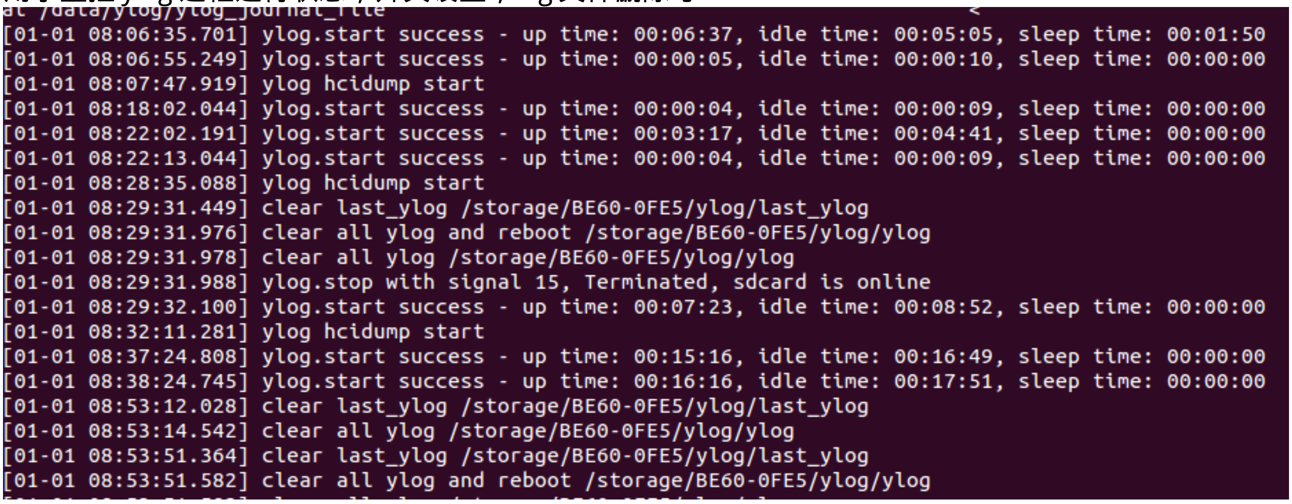
## 7、info

用于收集系统相关静态信息，只会在 ylog 启动时捕获一次

/proc/cmdline
/proc/version
/proc/meminfo
/proc/mounts
/proc/partitions
/proc/diskstats
/proc/modules
/proc/cpuinfo
/default.prop
/data/ylog/ylog.conf
ls -l /
ls -l /dev/block/platform/*/by-name/
ls -l /dev/
getprop
cat /*.rc
ylog 所有线程 pid 和 tid

## 8、journal

用于监控 ylog 进程运行状态，开关设置，log 文件删除等



```
at /data/ylog/ylog_journal_file
[01-01 08:06:35.701] ylog.start success - up time: 00:06:37, idle time: 00:05:05, sleep time: 00:01:50
[01-01 08:06:55.249] ylog.start success - up time: 00:00:05, idle time: 00:00:10, sleep time: 00:00:00
[01-01 08:07:47.919] ylog hcidump start
[01-01 08:18:02.044] ylog.start success - up time: 00:00:04, idle time: 00:00:09, sleep time: 00:00:00
[01-01 08:22:02.191] ylog.start success - up time: 00:03:17, idle time: 00:04:41, sleep time: 00:00:00
[01-01 08:22:13.044] ylog.start success - up time: 00:00:04, idle time: 00:00:09, sleep time: 00:00:00
[01-01 08:28:35.088] ylog hcidump start
[01-01 08:29:31.449] clear last_ylog /storage/BE60-0FE5/ylog/last_ylog
[01-01 08:29:31.976] clear all ylog and reboot /storage/BE60-0FE5/ylog/ylog
[01-01 08:29:31.978] clear all ylog /storage/BE60-0FE5/ylog/ylog
[01-01 08:29:31.988] ylog.stop with signal 15, Terminated, sdcard is online
[01-01 08:29:32.100] ylog.start success - up time: 00:07:23, idle time: 00:08:52, sleep time: 00:00:00
[01-01 08:32:11.281] ylog hcidump start
[01-01 08:37:24.808] ylog.start success - up time: 00:15:16, idle time: 00:16:49, sleep time: 00:00:00
[01-01 08:38:24.745] ylog.start success - up time: 00:16:16, idle time: 00:17:51, sleep time: 00:00:00
[01-01 08:53:12.028] clear last_ylog /storage/BE60-0FE5/ylog/last_ylog
[01-01 08:53:14.542] clear all ylog /storage/BE60-0FE5/ylog/ylog
[01-01 08:53:51.364] clear last_ylog /storage/BE60-0FE5/ylog/last_ylog
[01-01 08:53:51.582] clear all ylog and reboot /storage/BE60-0FE5/ylog/ylog
```

## 9、tracer

tracer 读取/d/tracing/trace_pipe 数据切片存储，允许内核对时间要求苛刻 的驱动模块（比如：usb）

## 10、traces

用于收集发生 native crash 时的 tombstones 信息和发生 anr 时的 traces 堆栈信息

## 11、socket

开放 ylog 网络 socket 端口，接收来自 ylog_benchmark 评测软件发送大数据，用来查找软件和系统瓶颈

## 13、sys_info_manual

手动触发 ylog 抓取 "busybox netstat -ap"和"ps -t"的执行结果

## 2.3 ylog 属性列表

**1、以下属性值表明各类 log 的运行状态**

```
[ylog.svc.android_crash]: [running]
[ylog.svc.android_events]: [running]
[ylog.svc.android_main]: [running]
[ylog.svc.android_radio]: [running]
[ylog.svc.android_system]: [running]
[ylog.svc.kernel]: [running]
[ylog.svc.tcpdump]: [stopped]
[ylog.svc.hcidump]: [stopped]
[ylog.svc.info]: [stopped]
[ylog.svc.journal]: [running]
[ylog.svc.socket]: [running]
[ylog.svc.sys_info]: [running]
[ylog.svc.sys_info_manual]: [stopped]
[ylog.svc.tracer]: [running]
[ylog.svc.traces]: [running]
[ylog.svc.ylog_debug]: [running]
```

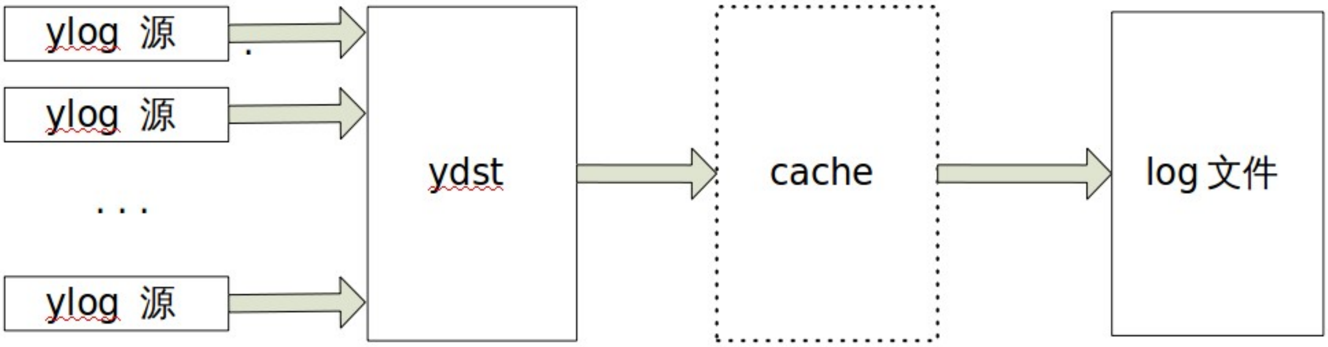
**2、ylog service 总开关控制**

```
[persist.ylog.enabled]: [1]
```

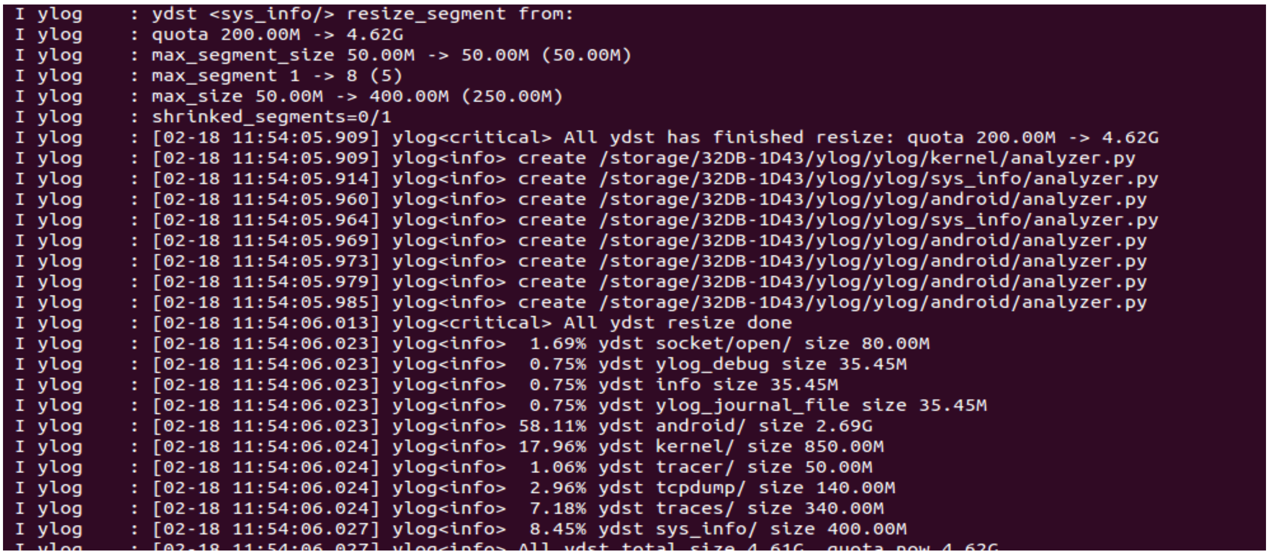
**3、t 卡被卸载次数**

```
[ylog.killed]: [0]
```

## 2.4 ylog 功能

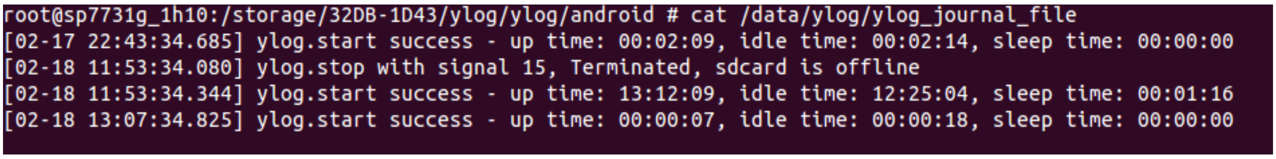


1、易于后期维护，抽象出 3 个数据模型：ylog，ydst 和 cache
2、cache 来缓解常规情况下磁盘压力和大数据吞吐下 log 丢失
3、filter 关键字过滤，能够对接收数据进行检索，并触发相应动作响应：执行 shell 或者其他操作收集更多信息，更有效的辅助问题定位（让 ylog 系统更加 smart 的做实时分析和环境捕获）
4、根据磁盘实际剩余空间分配 log 文件写入上限，动态调整 quota 值



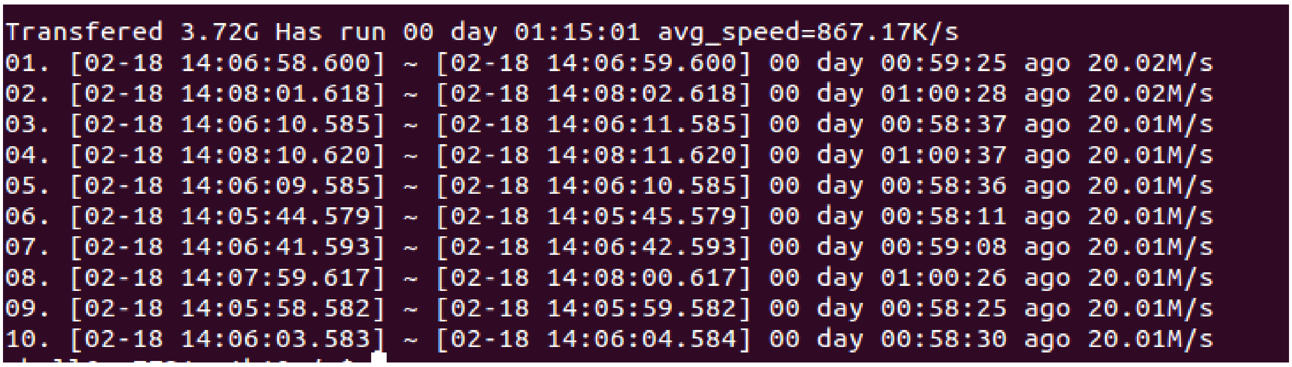
```
 I ylog     : ydst <sys_info/> resize_segment from:
 I ylog     : quota 200.00M -> 4.62G
 I ylog     : max_segment_size 50.00M -> 50.00M (50.00M)
 I ylog     : max_segment 1 -> 8 (5)
 I ylog     : max_size 50.00M -> 400.00M (250.00M)
 I ylog     : shrinked_segments=0/1
 I ylog     : [02-18 11:54:05.909] ylog<critical> All ydst has finished resize: quota 200.00M -> 4.62G
 I ylog     : [02-18 11:54:05.909] ylog<info> create /storage/32DB-1D43/ylog/ylog/kernel/analyzer.py
 I ylog     : [02-18 11:54:05.914] ylog<info> create /storage/32DB-1D43/ylog/ylog/sys_info/analyzer.py
 I ylog     : [02-18 11:54:05.960] ylog<info> create /storage/32DB-1D43/ylog/ylog/android/analyzer.py
 I ylog     : [02-18 11:54:05.964] ylog<info> create /storage/32DB-1D43/ylog/ylog/sys_info/analyzer.py
 I ylog     : [02-18 11:54:05.969] ylog<info> create /storage/32DB-1D43/ylog/ylog/android/analyzer.py
 I ylog     : [02-18 11:54:05.973] ylog<info> create /storage/32DB-1D43/ylog/ylog/android/analyzer.py
 I ylog     : [02-18 11:54:05.979] ylog<info> create /storage/32DB-1D43/ylog/ylog/android/analyzer.py
 I ylog     : [02-18 11:54:05.985] ylog<info> create /storage/32DB-1D43/ylog/ylog/android/analyzer.py
 I ylog     : [02-18 11:54:06.013] ylog<critical> All ydst resize done
 I ylog     : [02-18 11:54:06.023] ylog<info>  1.69% ydst socket/open/ size 80.00M
 I ylog     : [02-18 11:54:06.023] ylog<info>  0.75% ydst ylog_debug size 35.45M
 I ylog     : [02-18 11:54:06.023] ylog<info>  0.75% ydst info size 35.45M
 I ylog     : [02-18 11:54:06.023] ylog<info>  0.75% ydst ylog_journal_file size 35.45M
 I ylog     : [02-18 11:54:06.023] ylog<info> 58.11% ydst android/ size 2.69G
 I ylog     : [02-18 11:54:06.024] ylog<info> 17.96% ydst kernel/ size 850.00M
 I ylog     : [02-18 11:54:06.024] ylog<info>  1.06% ydst tracer/ size 50.00M
 I ylog     : [02-18 11:54:06.024] ylog<info>  2.96% ydst tcpdump/ size 140.00M
 I ylog     : [02-18 11:54:06.024] ylog<info>  7.18% ydst traces/ size 340.00M
 I ylog     : [02-18 11:54:06.027] ylog<info>  8.45% ydst sys_info/ size 400.00M
 I ylog     : [02-18 11:54:06.027] ylog<info> All ydst total size 4.61G, quota now 4.62G
```

5、对 ylog 进程状态监控，journal 日志永久性存储到/data/ylog /ylog_journal_file 来辅助分析系统行为



```
root@sp7731g_1h10:/storage/32DB-1D43/ylog/ylog/android # cat /data/ylog/ylog_journal_file
[02-17 22:43:34.685] ylog.start success - up time: 00:02:09, idle time: 00:02:14, sleep time: 00:00:00
[02-18 11:53:34.080] ylog.stop with signal 15, Terminated, sdcard is offline
[02-18 11:53:34.344] ylog.start success - up time: 13:12:09, idle time: 12:25:04, sleep time: 00:01:16
[02-18 13:07:34.825] ylog.start success - up time: 00:00:07, idle time: 00:00:18, sleep time: 00:00:00
```

6、ylog_cli speed 查询自运行以来最快的 10 次吞吐信息：log 产生速率

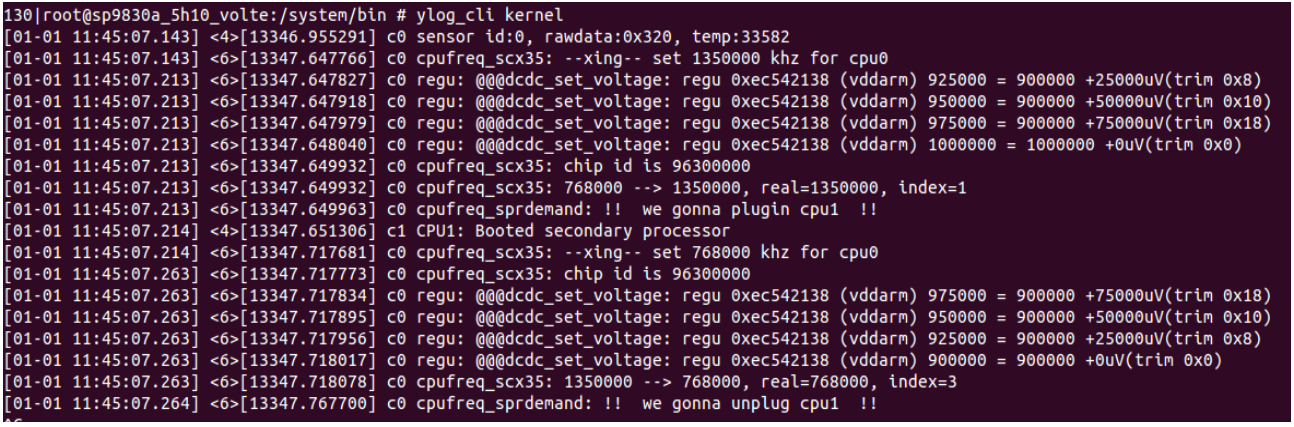


```
Transfered 3.72G Has run 00 day 01:15:01 avg_speed=867.17K/s
01. [02-18 14:06:58.600] ~ [02-18 14:06:59.600] 00 day 00:59:25 ago 20.02M/s
02. [02-18 14:08:01.618] ~ [02-18 14:08:02.618] 00 day 01:00:28 ago 20.02M/s
03. [02-18 14:06:10.585] ~ [02-18 14:06:11.585] 00 day 00:58:37 ago 20.01M/s
04. [02-18 14:08:10.620] ~ [02-18 14:08:11.620] 00 day 01:00:37 ago 20.01M/s
05. [02-18 14:06:09.585] ~ [02-18 14:06:10.585] 00 day 00:58:36 ago 20.01M/s
06. [02-18 14:05:44.579] ~ [02-18 14:05:45.579] 00 day 00:58:11 ago 20.01M/s
07. [02-18 14:06:41.593] ~ [02-18 14:06:42.593] 00 day 00:59:08 ago 20.01M/s
08. [02-18 14:07:59.617] ~ [02-18 14:08:00.617] 00 day 01:00:26 ago 20.01M/s
09. [02-18 14:05:58.582] ~ [02-18 14:05:59.582] 00 day 00:58:25 ago 20.01M/s
10. [02-18 14:06:03.583] ~ [02-18 14:06:04.584] 00 day 00:58:30 ago 20.01M/s
```

7、内置 analyzer.py，用来离线数据处理

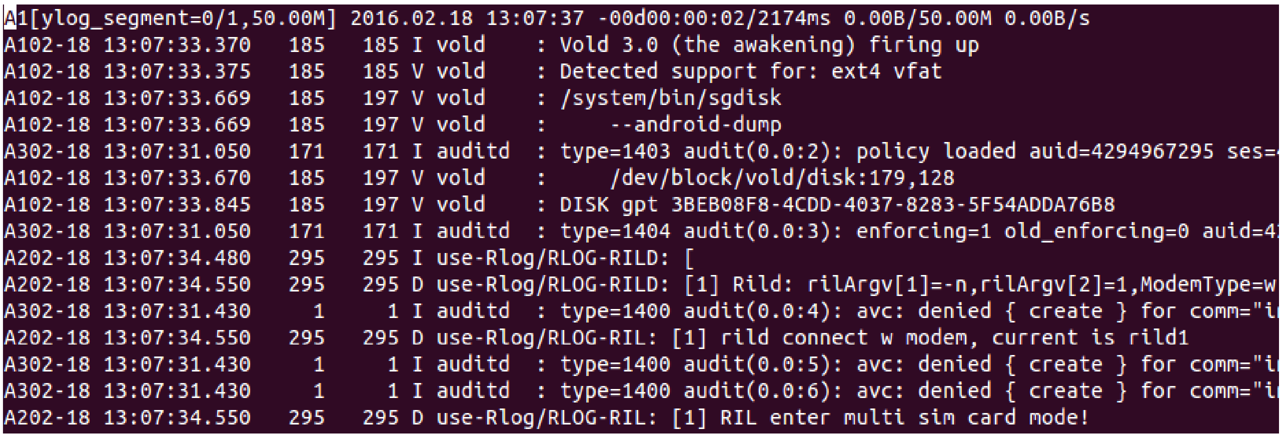
8、多个数据可以同时输出到一个 ydst 中，保证所有 log 时间戳都在同一时间窗体内

9、通过 ylog_cli kernel 可以实时获取 kernel log，最多支持 4 个独立 ylog_cli 同时操作



```
130|root@sp9830a_5h10_volte:/system/bin # ylog_cli kernel
[01-01 11:45:07.143] <4>[13346.955291] c0 sensor id:0, rawdata:0x320, temp:33582
[01-01 11:45:07.143] <6>[13347.647766] c0 cpufreq_scx35: --xing-- set 1350000 khz for cpu0
[01-01 11:45:07.213] <6>[13347.647827] c0 regu: @@@dcdc_set_voltage: regu 0xec542138 (vddarm) 925000 = 900000 +25000uV(trim 0x8)
[01-01 11:45:07.213] <6>[13347.647918] c0 regu: @@@dcdc_set_voltage: regu 0xec542138 (vddarm) 950000 = 900000 +50000uV(trim 0x10)
[01-01 11:45:07.213] <6>[13347.647979] c0 regu: @@@dcdc_set_voltage: regu 0xec542138 (vddarm) 975000 = 900000 +75000uV(trim 0x18)
[01-01 11:45:07.213] <6>[13347.648040] c0 regu: @@@dcdc_set_voltage: regu 0xec542138 (vddarm) 1000000 = 1000000 +0uV(trim 0x0)
[01-01 11:45:07.213] <6>[13347.649932] c0 cpufreq_scx35: chip id is 96300000
[01-01 11:45:07.213] <6>[13347.649932] c0 cpufreq_scx35: 768000 --> 1350000, real=1350000, index=1
[01-01 11:45:07.213] <6>[13347.649963] c0 cpufreq_sprdemand: !!  we gonna plugin cpu1  !!
[01-01 11:45:07.214] <4>[13347.651306] c1 CPU1: Booted secondary processor
[01-01 11:45:07.214] <6>[13347.717681] c0 cpufreq_scx35: --xing-- set 768000 khz for cpu0
[01-01 11:45:07.263] <6>[13347.717773] c0 cpufreq_scx35: chip id is 96300000
[01-01 11:45:07.263] <6>[13347.717834] c0 regu: @@@dcdc_set_voltage: regu 0xec542138 (vddarm) 975000 = 900000 +75000uV(trim 0x18)
[01-01 11:45:07.263] <6>[13347.717895] c0 regu: @@@dcdc_set_voltage: regu 0xec542138 (vddarm) 950000 = 900000 +50000uV(trim 0x10)
[01-01 11:45:07.263] <6>[13347.717956] c0 regu: @@@dcdc_set_voltage: regu 0xec542138 (vddarm) 925000 = 900000 +25000uV(trim 0x8)
[01-01 11:45:07.263] <6>[13347.718017] c0 regu: @@@dcdc_set_voltage: regu 0xec542138 (vddarm) 900000 = 900000 +0uV(trim 0x0)
[01-01 11:45:07.263] <6>[13347.718078] c0 cpufreq_scx35: 1350000 --> 768000, real=768000, index=3
[01-01 11:45:07.264] <6>[13347.767700] c0 cpufreq_sprdemand: !!  we gonna unplug cpu1  !!
```

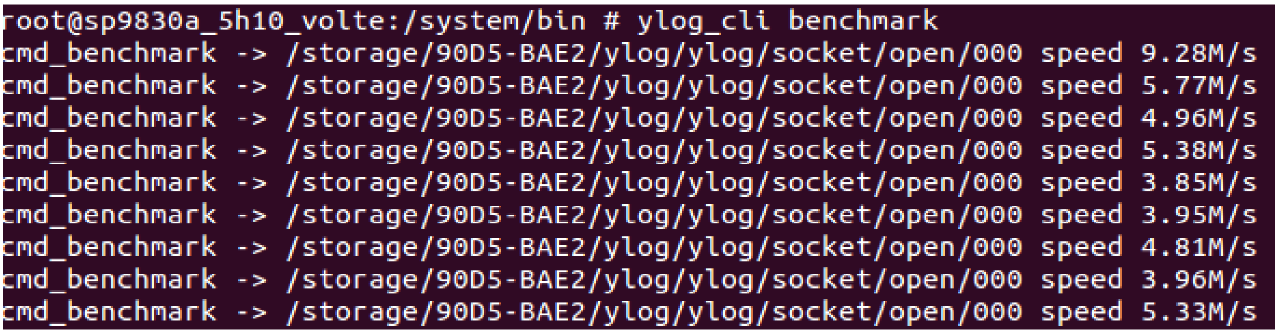
10、可通过段数信息查询 log 是否达到段数上限，是否发生了 log 覆盖



```
A1[ylog_segment=0/1,50.00M] 2016.02.18 13:07:37 -00d00:00:02/2174ms 0.00B/50.00M 0.00B/s
A102-18 13:07:33.370   185   185 I vold     : Vold 3.0 (the awakening) firing up
A102-18 13:07:33.375   185   185 V vold     : Detected support for: ext4 vfat
A102-18 13:07:33.669   185   197 V vold     : /system/bin/sgdisk
A102-18 13:07:33.669   185   197 V vold     :     --android-dump
A302-18 13:07:31.050   171   171 I auditd   : type=1403 audit(0.0:2): policy loaded auid=4294967295 ses=
A102-18 13:07:33.670   185   197 V vold     :     /dev/block/vold/disk:179,128
A102-18 13:07:33.845   185   197 V vold     : DISK gpt 3BEB08F8-4CDD-4037-8283-5F54ADDA76B8
A302-18 13:07:31.050   171   171 I auditd   : type=1404 audit(0.0:3): enforcing=1 old_enforcing=0 auid=4
A202-18 13:07:34.480   295   295 I use-Rlog/RLOG-RILD: [
A202-18 13:07:34.550   295   295 D use-Rlog/RLOG-RILD: [1] Rild: rilArgv[1]=-n,rilArgv[2]=1,ModemType=w
A302-18 13:07:31.430     1     1 I auditd   : type=1400 audit(0.0:4): avc: denied { create } for comm="i
A202-18 13:07:34.550   295   295 D use-Rlog/RLOG-RIL: [1] rild connect w modem, current is rild1
A302-18 13:07:31.430     1     1 I auditd   : type=1400 audit(0.0:5): avc: denied { create } for comm="i
A302-18 13:07:31.430     1     1 I auditd   : type=1400 audit(0.0:6): avc: denied { create } for comm="i
A202-18 13:07:34.550   295   295 D use-Rlog/RLOG-RIL: [1] RIL enter multi sim card mode!
```

11、sys_info 每 2 分钟执行系统信息统计收集操作，数据被切片存储

12、评测磁盘读写速率和 ylog 可读写数据速度瓶颈上限：ylog_cli benchmark



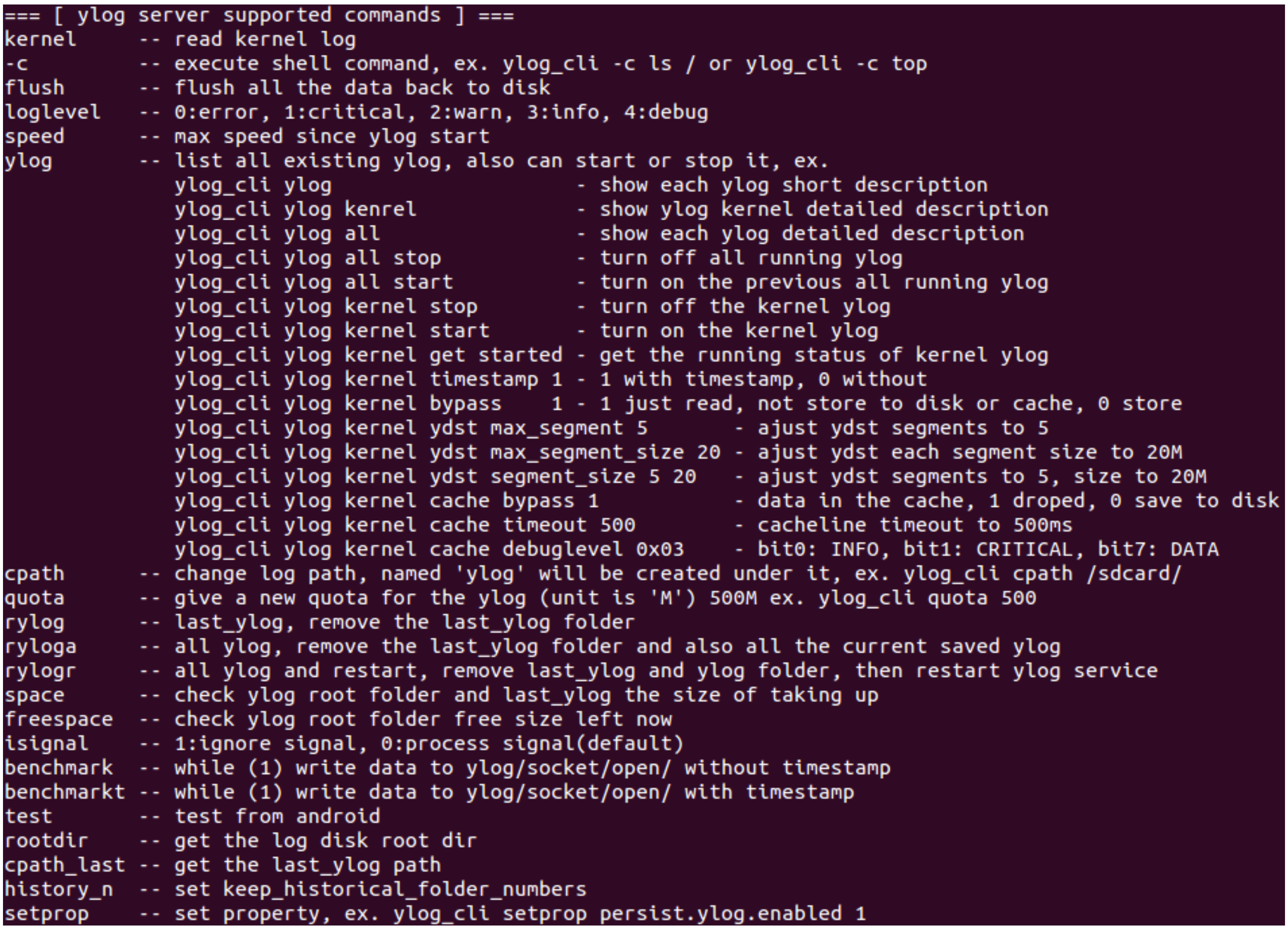
```
root@sp9830a_5h10_volte:/system/bin # ylog_cli benchmark
cmd_benchmark -> /storage/90D5-BAE2/ylog/ylog/socket/open/000 speed 9.28M/s
cmd_benchmark -> /storage/90D5-BAE2/ylog/ylog/socket/open/000 speed 5.77M/s
cmd_benchmark -> /storage/90D5-BAE2/ylog/ylog/socket/open/000 speed 4.96M/s
cmd_benchmark -> /storage/90D5-BAE2/ylog/ylog/socket/open/000 speed 5.38M/s
cmd_benchmark -> /storage/90D5-BAE2/ylog/ylog/socket/open/000 speed 3.85M/s
cmd_benchmark -> /storage/90D5-BAE2/ylog/ylog/socket/open/000 speed 3.95M/s
cmd_benchmark -> /storage/90D5-BAE2/ylog/ylog/socket/open/000 speed 4.81M/s
cmd_benchmark -> /storage/90D5-BAE2/ylog/ylog/socket/open/000 speed 3.96M/s
cmd_benchmark -> /storage/90D5-BAE2/ylog/ylog/socket/open/000 speed 5.33M/s
```

13、tracer 读取/d/tracing/trace_pipe 数据切片存储，允许内核对时间要求苛刻 的驱动模块（比如：usb）添加调试 log 调整 quota 值

# 3. ylog_cli 命令行

可通过 adb shell ylog_cli 命令获取 ylog_cli 命令行帮助信息。

## 3.1 ylog_cli 命令行概览



```
=== [ ylog server supported commands ] ===
kernel      -- read kernel log
-c          -- execute shell command, ex. ylog_cli -c ls / or ylog_cli -c top
flush       -- flush all the data back to disk
loglevel    -- 0:error, 1:critical, 2:warn, 3:info, 4:debug
speed       -- max speed since ylog start
ylog        -- list all existing ylog, also can start or stop it, ex.
               ylog_cli ylog                   - show each ylog short description
               ylog_cli ylog kenrel            - show ylog kernel detailed description
               ylog_cli ylog all               - show each ylog detailed description
               ylog_cli ylog all stop          - turn off all running ylog
               ylog_cli ylog all start         - turn on the previous all running ylog
               ylog_cli ylog kernel stop       - turn off the kernel ylog
               ylog_cli ylog kernel start      - turn on the kernel ylog
               ylog_cli ylog kernel get started - get the running status of kernel ylog
               ylog_cli ylog kernel timestamp 1 - 1 with timestamp, 0 without
               ylog_cli ylog kernel bypass     1 - 1 just read, not store to disk or cache, 0 store
               ylog_cli ylog kernel ydst max_segment 5        - ajust ydst segments to 5
               ylog_cli ylog kernel ydst max_segment_size 20 - ajust ydst each segment size to 20M
               ylog_cli ylog kernel ydst segment_size 5 20   - ajust ydst segments to 5, size to 20M
               ylog_cli ylog kernel cache bypass 1           - data in the cache, 1 droped, 0 save to disk
               ylog_cli ylog kernel cache timeout 500        - cacheline timeout to 500ms
               ylog_cli ylog kernel cache debuglevel 0x03    - bit0: INFO, bit1: CRITICAL, bit7: DATA
cpath       -- change log path, named 'ylog' will be created under it, ex. ylog_cli cpath /sdcard/
quota       -- give a new quota for the ylog (unit is 'M') 500M ex. ylog_cli quota 500
rylog       -- last_ylog, remove the last_ylog folder
ryloga      -- all ylog, remove the last_ylog folder and also all the current saved ylog
rylogr      -- all ylog and restart, remove last_ylog and ylog folder, then restart ylog service
space       -- check ylog root folder and last_ylog the size of taking up
freespace   -- check ylog root folder free size left now
isignal     -- 1:ignore signal, 0:process signal(default)
benchmark   -- while (1) write data to ylog/socket/open/ without timestamp
benchmarkt  -- while (1) write data to ylog/socket/open/ with timestamp
test        -- test from android
rootdir     -- get the log disk root dir
cpath_last  -- get the last_ylog path
history_n   -- set keep_historical_folder_numbers
setprop     -- set property, ex. ylog_cli setprop persist.ylog.enabled 1
```

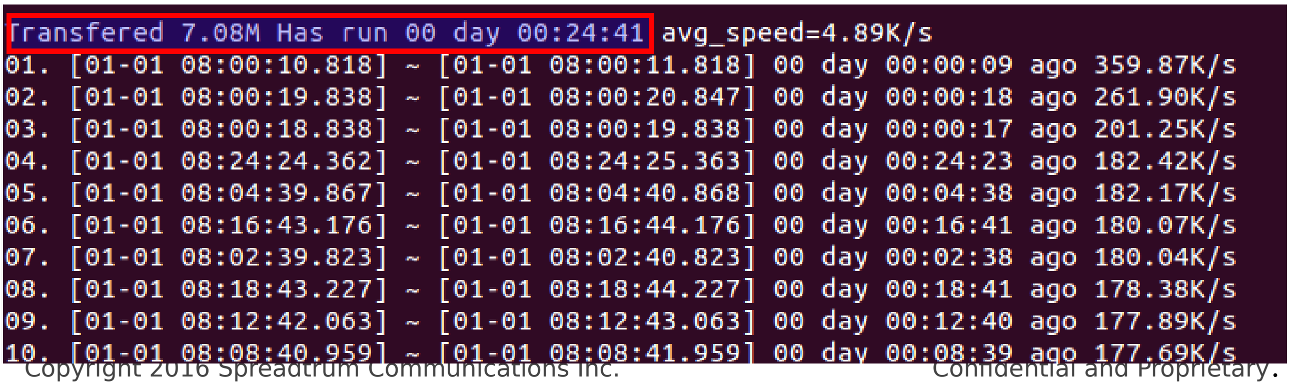
## 3.2 ylog_cli 命令行说明

ylog_cli 通过 socket 方式与 ylog 进程进行通信，支持多类命令行，可分为：查询类、开关类、设置类、删除类和其他。

### 3.2.1 查询类命令行

**1、查询 ylog 进程从启动以来的速度前 10 名**

**2、查询各类 log 开关状态，段上限值、单个段大小上限值、log 传输量**



```
Transfered 7.08M Has run 00 day 00:24:41 avg_speed=4.89K/s
01. [01-01 08:00:10.818] ~ [01-01 08:00:11.818] 00 day 00:00:09 ago 359.87K/s
02. [01-01 08:00:19.838] ~ [01-01 08:00:20.847] 00 day 00:00:18 ago 261.90K/s
03. [01-01 08:00:18.838] ~ [01-01 08:00:19.838] 00 day 00:00:17 ago 201.25K/s
04. [01-01 08:24:24.362] ~ [01-01 08:24:25.363] 00 day 00:24:23 ago 182.42K/s
05. [01-01 08:04:39.867] ~ [01-01 08:04:40.868] 00 day 00:04:38 ago 182.17K/s
06. [01-01 08:16:43.176] ~ [01-01 08:16:44.176] 00 day 00:16:41 ago 180.07K/s
07. [01-01 08:02:39.823] ~ [01-01 08:02:40.823] 00 day 00:02:38 ago 180.04K/s
08. [01-01 08:18:43.227] ~ [01-01 08:18:44.227] 00 day 00:18:41 ago 178.38K/s
09. [01-01 08:12:42.063] ~ [01-01 08:12:43.063] 00 day 00:12:40 ago 177.89K/s
10. [01-01 08:08:40.959] ~ [01-01 08:08:41.959] 00 day 00:08:39 ago 177.69K/s
```

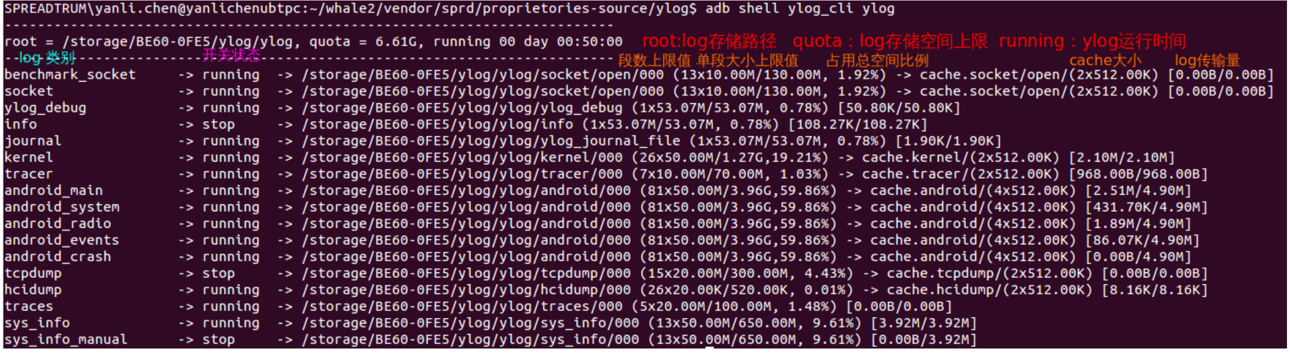
命令格式：adb shell ylog_cli ylog

响应格式：

root:显示当前 log 的存储路径

quota:log 存储空间上限值

running：表示 ylog 运行时间



```
SPREADTRUM\yanli.chen@yanlichenubtpc:~/whale2/vendor/sprd/proprietories-source/ylog$ adb shell ylog_cli ylog
------------------------------------------------------------------------
root = /storage/BE60-0FE5/ylog/ylog, quota = 6.61G, running 00 day 00:50:00
------------------------------------------------------------------------
benchmark_socket     -> running  -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 (13x10.00M/130.00M, 1.92%) -> cache.socket/open/(2x512.00K) [0.00B/0.00B]
socket               -> running  -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 (13x10.00M/130.00M, 1.92%) -> cache.socket/open/(2x512.00K) [0.00B/0.00B]
ylog_debug           -> running  -> /storage/BE60-0FE5/ylog/ylog/ylog_debug (1x53.07M/53.07M, 0.78%) [50.80K/50.80K]
info                 -> stop     -> /storage/BE60-0FE5/ylog/ylog/info (1x53.07M/53.07M, 0.78%) [108.27K/108.27K]
journal              -> running  -> /storage/BE60-0FE5/ylog/ylog/ylog_journal_file (1x53.07M/53.07M, 0.78%) [1.90K/1.90K]
kernel               -> running  -> /storage/BE60-0FE5/ylog/ylog/kernel/000 (26x50.00M/1.27G,19.21%) -> cache.kernel/(2x512.00K) [2.10M/2.10M]
tracer               -> running  -> /storage/BE60-0FE5/ylog/ylog/tracer/000 (7x10.00M/70.00M, 1.03%) -> cache.tracer/(2x512.00K) [968.00B/968.00B]
android_main         -> running  -> /storage/BE60-0FE5/ylog/ylog/android/000 (81x50.00M/3.96G,59.86%) -> cache.android/(4x512.00K) [2.51M/4.90M]
android_system       -> running  -> /storage/BE60-0FE5/ylog/ylog/android/000 (81x50.00M/3.96G,59.86%) -> cache.android/(4x512.00K) [431.70K/4.90M]
android_radio        -> running  -> /storage/BE60-0FE5/ylog/ylog/android/000 (81x50.00M/3.96G,59.86%) -> cache.android/(4x512.00K) [1.89M/4.90M]
android_events       -> running  -> /storage/BE60-0FE5/ylog/ylog/android/000 (81x50.00M/3.96G,59.86%) -> cache.android/(4x512.00K) [86.07K/4.90M]
android_crash        -> running  -> /storage/BE60-0FE5/ylog/ylog/android/000 (81x50.00M/3.96G,59.86%) -> cache.android/(4x512.00K) [0.00B/4.90M]
tcpdump              -> stop     -> /storage/BE60-0FE5/ylog/ylog/tcpdump/000 (15x20.00M/300.00M, 4.43%) -> cache.tcpdump/(2x512.00K) [0.00B/0.00B]
hcidump              -> running  -> /storage/BE60-0FE5/ylog/ylog/hcidump/000 (26x20.00K/520.00K, 0.01%) -> cache.hcidump/(2x512.00K) [8.16K/8.16K]
traces               -> running  -> /storage/BE60-0FE5/ylog/ylog/traces/000 (5x20.00M/100.00M, 1.48%) [0.00B/0.00B]
sys_info             -> running  -> /storage/BE60-0FE5/ylog/ylog/sys_info/000 (13x50.00M/650.00M, 9.61%) [3.92M/3.92M]
sys_info_manual      -> stop     -> /storage/BE60-0FE5/ylog/ylog/sys_info/000 (13x50.00M/650.00M, 9.61%) [0.00B/3.92M]
```

## 3、查询单个 log 开关状态（同 ylog.svc.xxx）

命令格式：ylog_cli ylog xxx get started

响应格式为：

打开状态： 1\n

关闭状态： 0\n

注：xxx 可分别为：android_main android_system android_radio android_events android_crash tcpdump hcidump kernel

## 4、查询 log 存储路径

命令格式：adb shell ylog_cli cpath

响应格式：

不插 t 卡时：

/data/ylog/ylog 内部存储不支持保存历史 log

插 t 卡时：

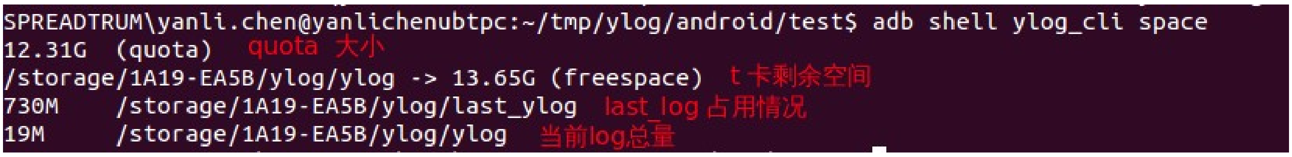
/storage/BE60-0FE5/ylog/ylog

/storage/BE60-0FE5/ylog/last_ylog

## 5、查询 log 空间占用情况

命令格式：adb shell ylog_cli space

相应格式：



```
SPREADTRUM\yanli.chen@yanlichenubtpc:~/tmp/ylog/android/test$ adb shell ylog_cli space
12.31G  (quota)
/storage/1A19-EA5B/ylog/ylog -> 13.65G (freespace)
730M    /storage/1A19-EA5B/ylog/last_ylog
19M     /storage/1A19-EA5B/ylog/ylog
```

## 6、 查询 log 所在磁盘剩余空间

命令格式：adb shell ylog_cli freespace

响应格式：

/storage/BE60-0FE5/ylog/ylog -> 7.32G

## 7、查询 log 所在磁盘路径

命令格式：adb shell ylog_cli rootdir

响应格式：

/storage/BE60-0FE5

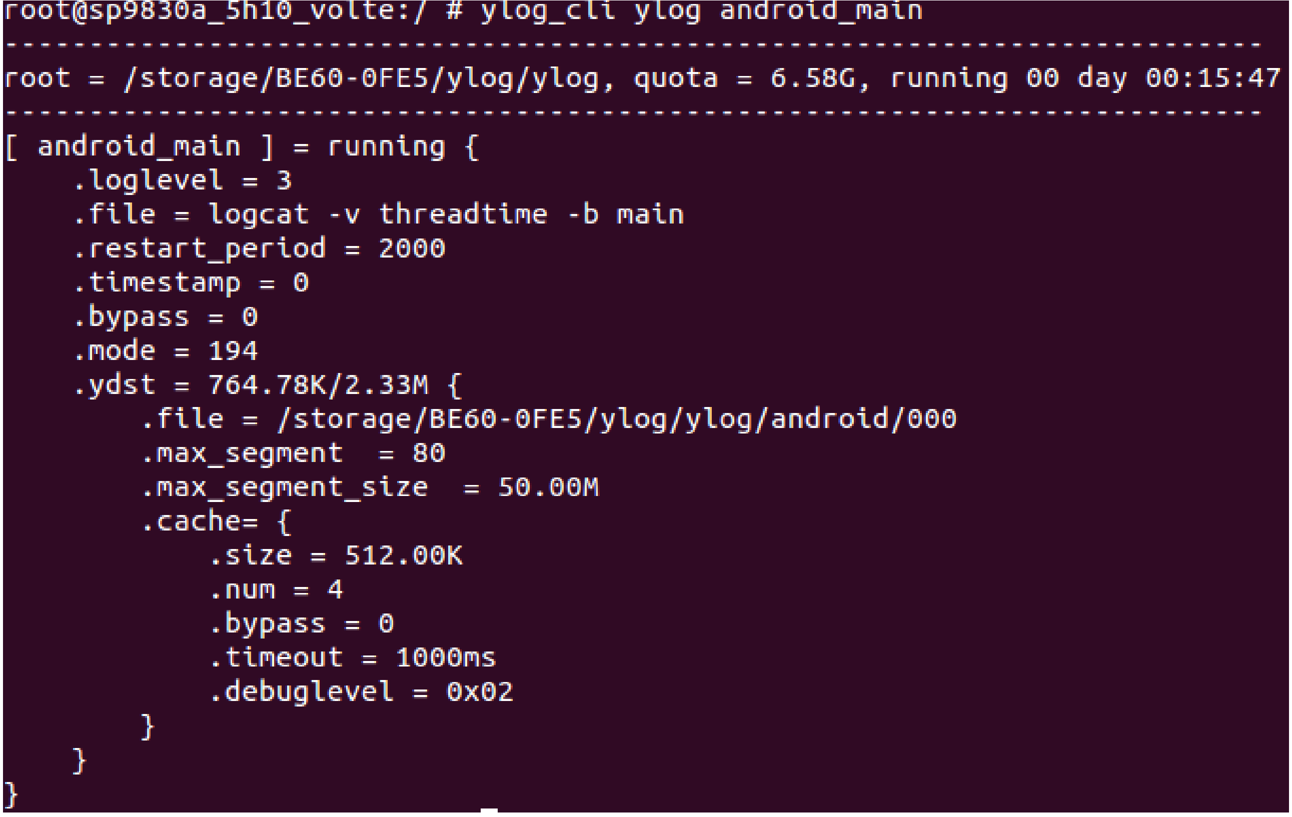
## 8、查询历史 log 保留次数

命令格式：adb shell ylog_cli history_n

响应格式：5\n　默认保留 5 次历史 log

## 9、查询单个 ylog 源的相关信息

命令格式：adb shell ylog_cli ylog xxx



```
root@sp9830a_5h10_volte:/ # ylog_cli ylog android_main
--------------------------------------------------------------------------
root = /storage/BE60-0FE5/ylog/ylog, quota = 6.58G, running 00 day 00:15:47
--------------------------------------------------------------------------
[ android_main ] = running {
    .loglevel = 3
    .file = logcat -v threadtime -b main
    .restart_period = 2000
    .timestamp = 0
    .bypass = 0
    .mode = 194
    .ydst = 764.78K/2.33M {
        .file = /storage/BE60-0FE5/ylog/ylog/android/000
        .max_segment  = 80
        .max_segment_size  = 50.00M
        .cache= {
            .size = 512.00K
            .num = 4
            .bypass = 0
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

响应格式：输出 ylog 源的输入类型、存储路径、cache 大小等信息

注：xxx 可分别为：android_main android_system android_radio android_events android_crash tcpdump hcidump kernel

**10、查询所有 log 相关信息**

命令格式：adb shell ylog_cli ylog all

响应格式：输出所有 log 的输入类型、存储路径、cache 大小等信息

### 3.2.2 开关类命令行

**1、ylog 总开关设置**

命令行格式：

start ylog 进程：adb shell setprop persist.ylog.enabled 1 -----> persist.ylog.enabled 1

stop ylog 进程：adb shell setprop persist.ylog.enabled 0 -----> persist.ylog.enabled 0

**2、所有 log 开关统一设置**

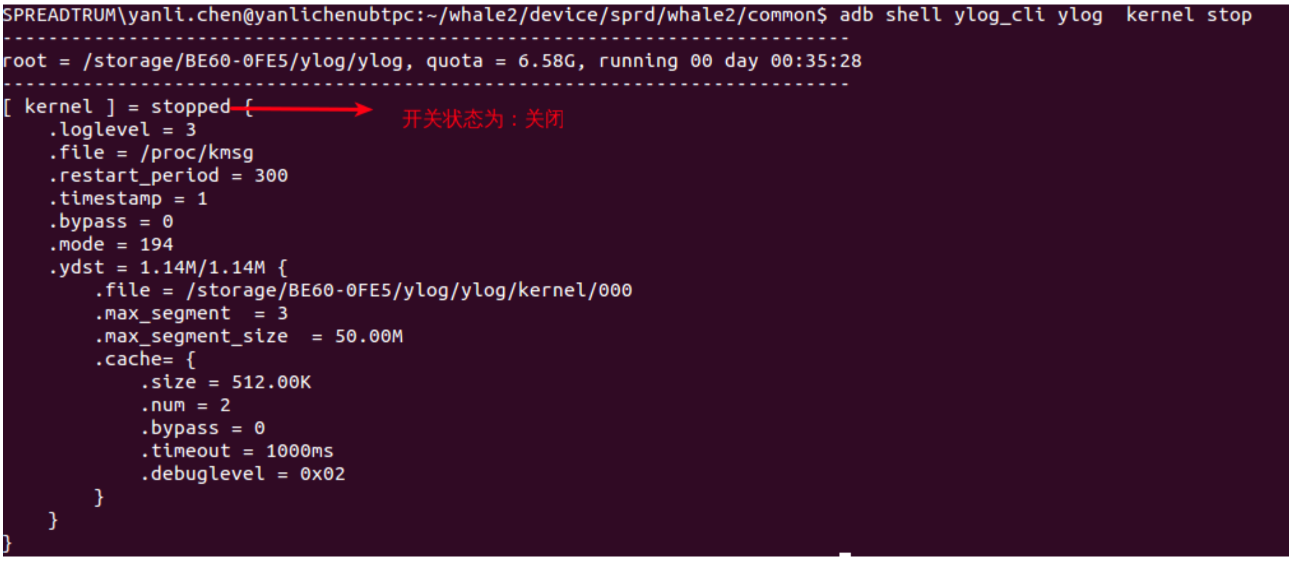
命令格式：adb shell ylog_cli ylog all start/stop

响应格式：输出所有 log 的开关状态、存储路径、cache 大小等信息

**3、单个 log 开关设置**

命令格式：adb shell ylog_cli ylog xxx start/stop

响应格式：输出该 log 的数据结构信息



```
SPREADTRUM\yanli.chen@yanlichenubtpc:~/whale2/device/sprd/whale2/common$ adb shell ylog_cli ylog  kernel stop
--------------------------------------------------------------------------
root = /storage/BE60-0FE5/ylog/ylog, quota = 6.58G, running 00 day 00:35:28
--------------------------------------------------------------------------
[ kernel ] = stopped {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 0
    .mode = 194
    .ydst = 1.14M/1.14M {
        .file = /storage/BE60-0FE5/ylog/ylog/kernel/000
        .max_segment  = 3
        .max_segment_size  = 50.00M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

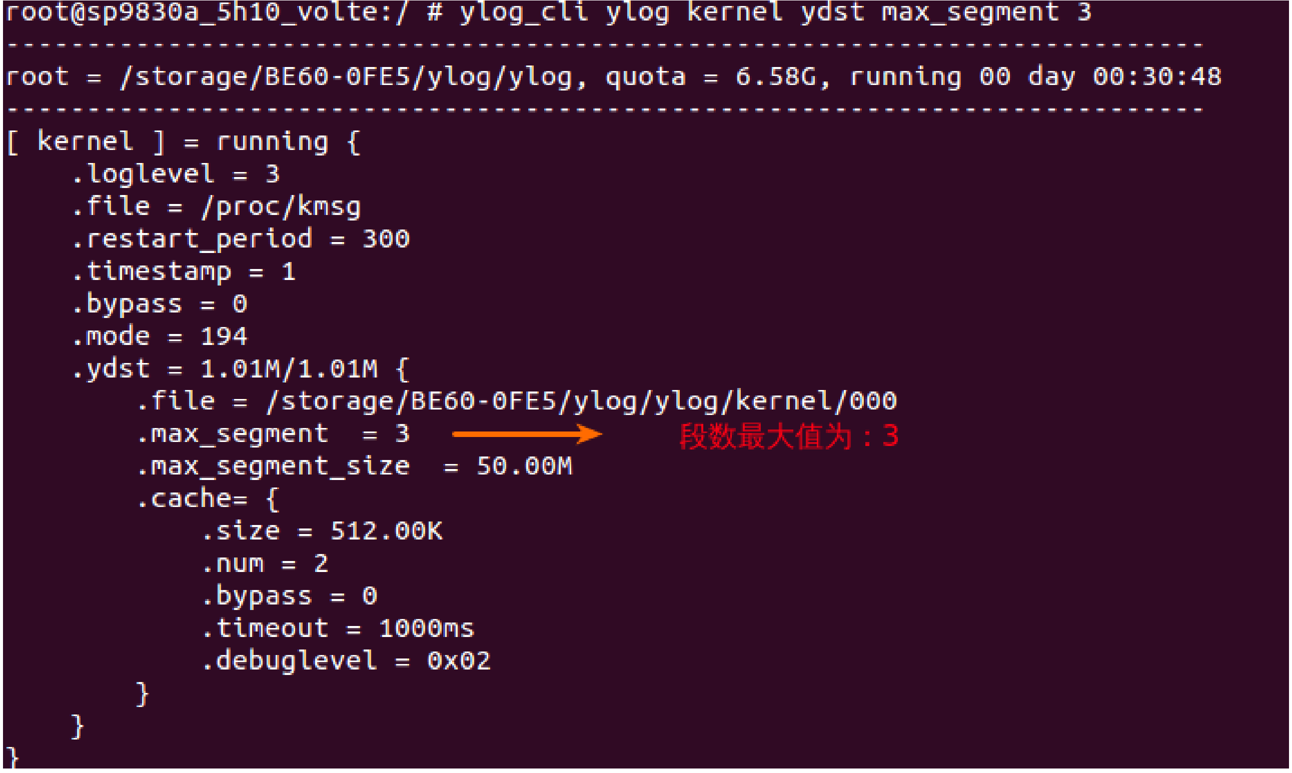
开关状态为：关闭

### 3.2.3 设置类命令行

**1、设置 log 段数最大值**

命令格式：adb shell ylog_cli ylog xxx ydst max_segment n

响应格式：输出 log 的数据结构信息



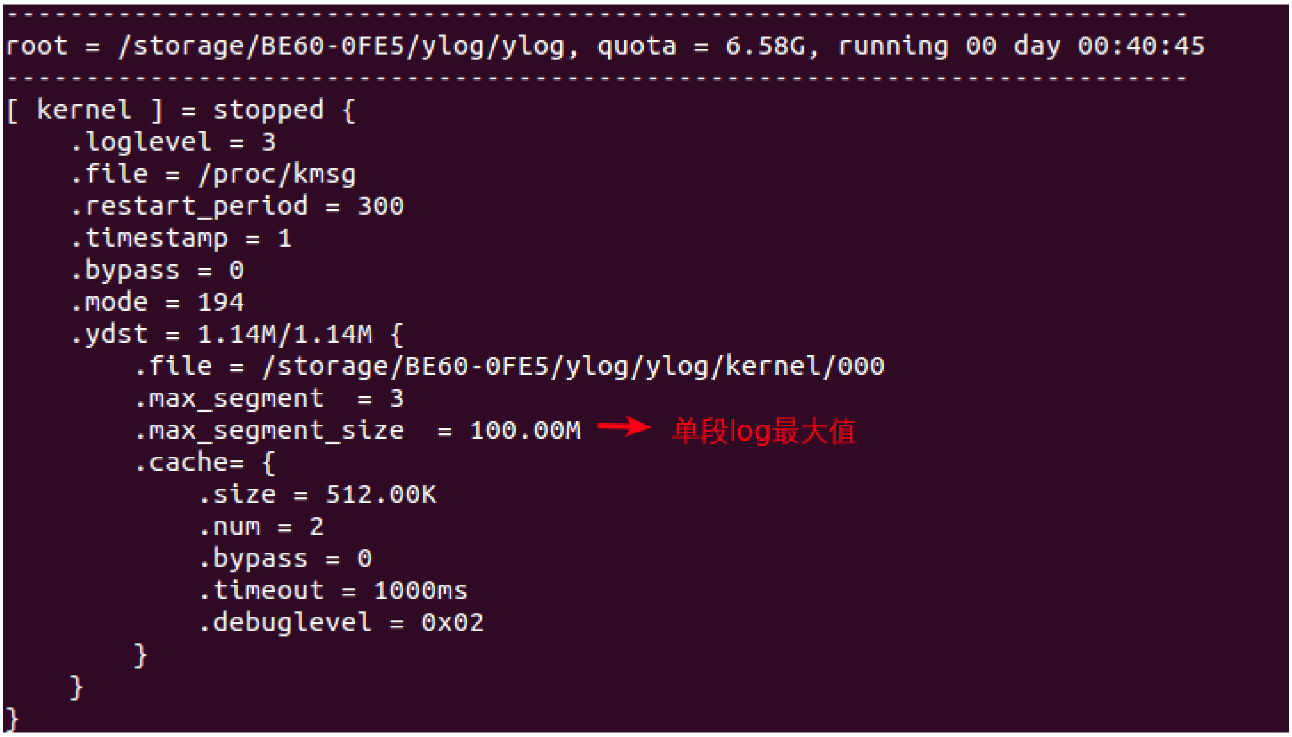
```
root@sp9830a_5h10_volte:/ # ylog_cli ylog kernel ydst max_segment 3
--------------------------------------------------------------------------
root = /storage/BE60-0FE5/ylog/ylog, quota = 6.58G, running 00 day 00:30:48
--------------------------------------------------------------------------
[ kernel ] = running {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 0
    .mode = 194
    .ydst = 1.01M/1.01M {
        .file = /storage/BE60-0FE5/ylog/ylog/kernel/000
        .max_segment  = 3
        .max_segment_size  = 50.00M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

段数最大值为：3

**2、设置 log 单个文件大小,单位为：M**

命令行格式：adb shell ylog_cli ylog xxx ydst max_segment_size 20

响应格式：输出 log 的数据结构信息



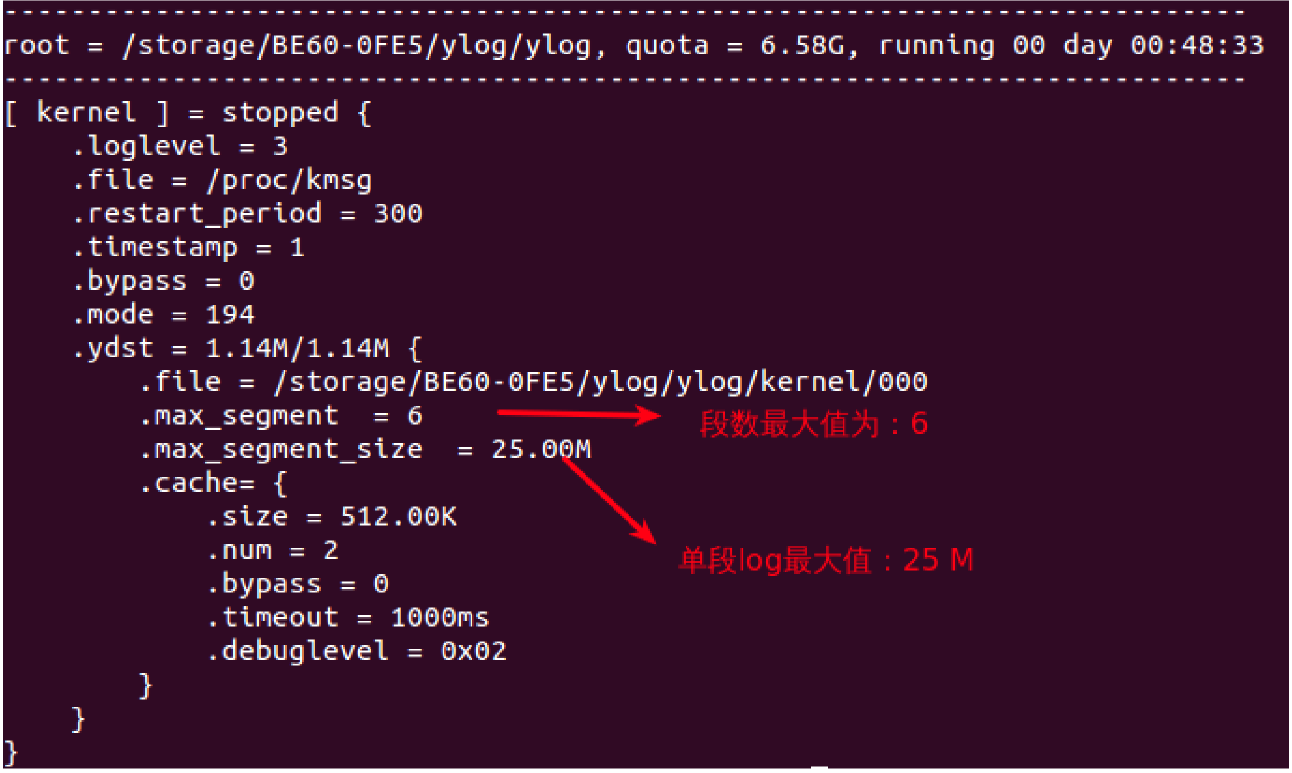
```
------------------------------------------------------------------------
root = /storage/BE60-0FE5/ylog/ylog, quota = 6.58G, running 00 day 00:40:45
------------------------------------------------------------------------
[ kernel ] = stopped {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 0
    .mode = 194
    .ydst = 1.14M/1.14M {
        .file = /storage/BE60-0FE5/ylog/ylog/kernel/000
        .max_segment  = 3
        .max_segment_size  = 100.00M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

单段log最大值

**3、同时设置 log 段数最大值和单段 log 大小上限值**

命令格式：adb shell ylog_cli ylog xxx ydst segment_size 5 20

响应格式：输出 log 的数据结构信息



```
------------------------------------------------------------------------
root = /storage/BE60-0FE5/ylog/ylog, quota = 6.58G, running 00 day 00:48:33
------------------------------------------------------------------------
[ kernel ] = stopped {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 0
    .mode = 194
    .ydst = 1.14M/1.14M {
        .file = /storage/BE60-0FE5/ylog/ylog/kernel/000
        .max_segment  = 6
        .max_segment_size  = 25.00M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

段数最大值为：6

单段log最大值：25 M

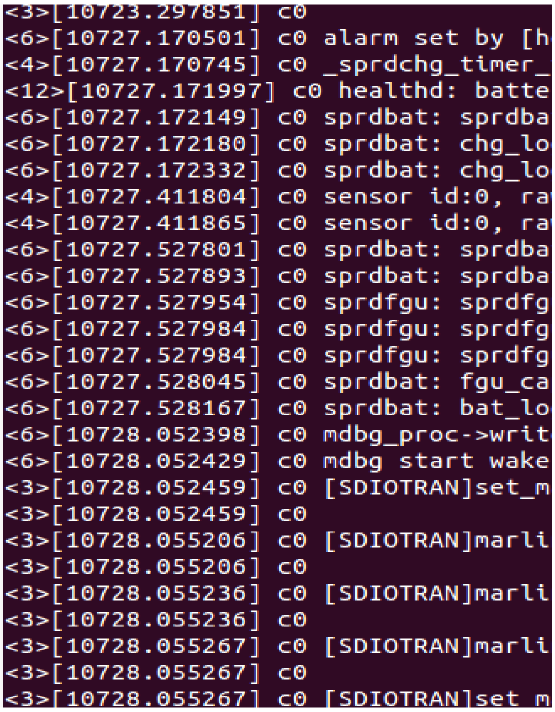
**4、设置通过 ylog_cli 命令行实时打印 kernel log 时是否打印时间戳**

命令格式：adb shell ylog_cli ylog kernel timestamp 1/0

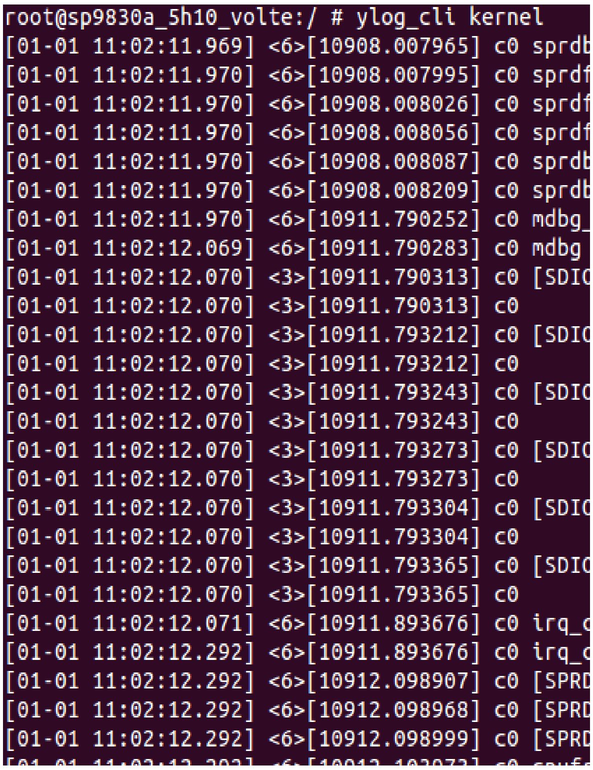
1： 打印时间戳

0：不打印时间戳

响应格式：终端实时打印 kernel log



```
<3>[10723.297851] c0
<6>[10727.170501] c0 alarm set by [h
<4>[10727.170745] c0 _sprdchg_timer_
<12>[10727.171997] c0 healthd: batte
<6>[10727.172149] c0 sprdbat: sprdba
<6>[10727.172180] c0 sprdbat: chg_lo
<6>[10727.172332] c0 sprdbat: chg_lo
<4>[10727.411804] c0 sensor id:0, ra
<4>[10727.411865] c0 sensor id:0, ra
<6>[10727.527801] c0 sprdbat: sprdba
<6>[10727.527893] c0 sprdbat: sprdba
<6>[10727.527954] c0 sprdfgu: sprdfg
<6>[10727.527984] c0 sprdfgu: sprdfg
<6>[10727.527984] c0 sprdfgu: sprdfg
<6>[10727.528045] c0 sprdbat: fgu_ca
<6>[10727.528167] c0 sprdbat: bat_lo
<6>[10728.052398] c0 mdbg_proc->writ
<6>[10728.052429] c0 mdbg start wake
<3>[10728.052459] c0 [SDIOTRAN]set_m
<3>[10728.052459] c0
<3>[10728.055206] c0 [SDIOTRAN]marli
<3>[10728.055206] c0
<3>[10728.055236] c0 [SDIOTRAN]marli
<3>[10728.055236] c0
<3>[10728.055267] c0 [SDIOTRAN]marli
<3>[10728.055267] c0
<3>[10728.055267] c0 [SDIOTRAN]set_m
```



```
root@sp9830a_5h10_volte:/ # ylog_cli kernel
[01-01 11:02:11.969] <6>[10908.007965] c0 sprdb
[01-01 11:02:11.970] <6>[10908.007995] c0 sprdf
[01-01 11:02:11.970] <6>[10908.008026] c0 sprdf
[01-01 11:02:11.970] <6>[10908.008056] c0 sprdf
[01-01 11:02:11.970] <6>[10908.008087] c0 sprdb
[01-01 11:02:11.970] <6>[10908.008209] c0 sprdb
[01-01 11:02:11.970] <6>[10911.790252] c0 mdbg_
[01-01 11:02:12.069] <6>[10911.790283] c0 mdbg
[01-01 11:02:12.070] <3>[10911.790313] c0 [SDIO
[01-01 11:02:12.070] <3>[10911.790313] c0
[01-01 11:02:12.070] <3>[10911.793212] c0 [SDIO
[01-01 11:02:12.070] <3>[10911.793212] c0
[01-01 11:02:12.070] <3>[10911.793243] c0 [SDIO
[01-01 11:02:12.070] <3>[10911.793243] c0
[01-01 11:02:12.070] <3>[10911.793273] c0 [SDIO
[01-01 11:02:12.070] <3>[10911.793273] c0
[01-01 11:02:12.070] <3>[10911.793304] c0 [SDIO
[01-01 11:02:12.070] <3>[10911.793304] c0
[01-01 11:02:12.070] <3>[10911.793365] c0 [SDIO
[01-01 11:02:12.070] <3>[10911.793365] c0
[01-01 11:02:12.071] <6>[10911.893676] c0 irq_c
[01-01 11:02:12.292] <6>[10911.893676] c0 irq_c
[01-01 11:02:12.292] <6>[10912.098907] c0 [SPRD
[01-01 11:02:12.292] <6>[10912.098968] c0 [SPRD
[01-01 11:02:12.292] <6>[10912.098999] c0 [SPRD
```

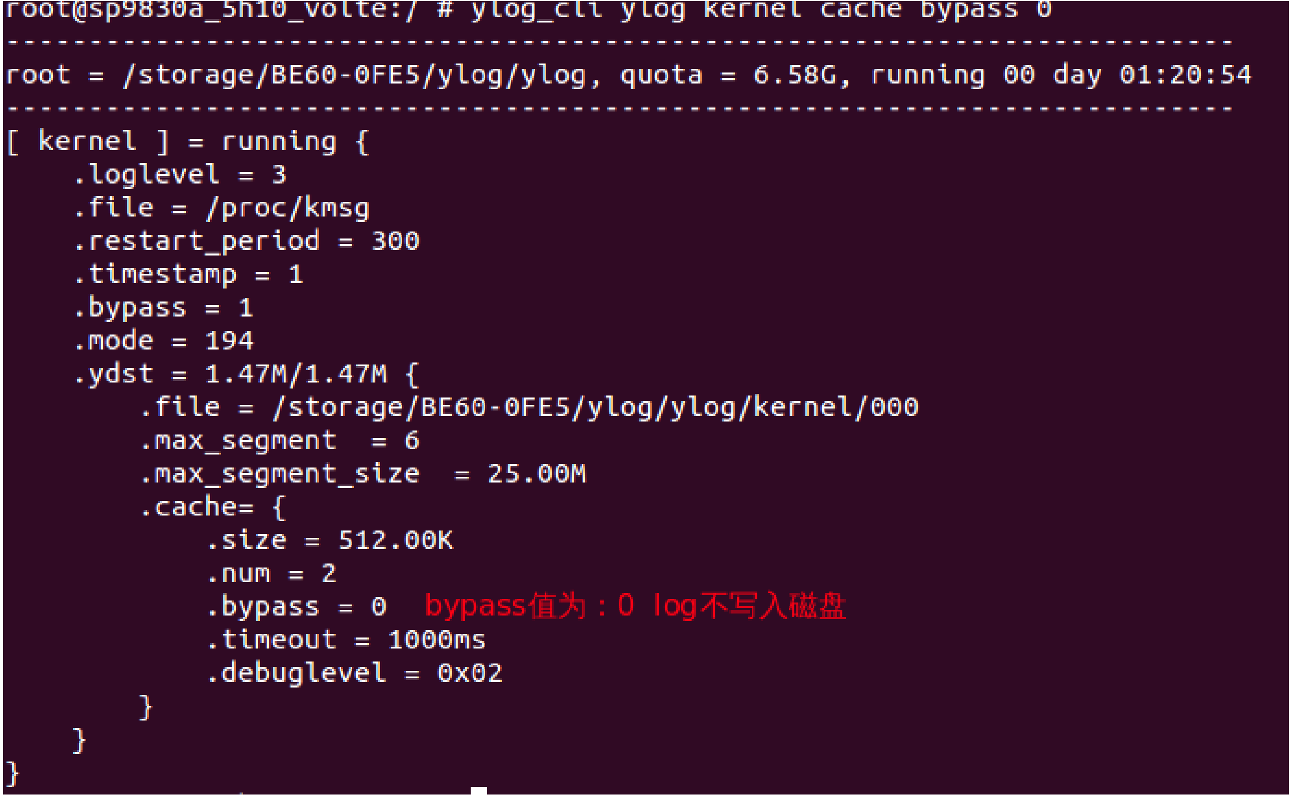
### 5、设置 log 是否回写磁盘

命令格式：adb shell ylog_cli ylog xxx bypass 1/0

1：回写磁盘

0：不写入磁盘

响应格式：输出 log 的数据结构信息



```
root@sp9830a_5h10_volte:/ # ylog_cli ylog kernel cache bypass 0
-------------------------------------------------------------------------
root = /storage/BE60-0FE5/ylog/ylog, quota = 6.58G, running 00 day 01:20:54
-------------------------------------------------------------------------
[ kernel ] = running {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 1
    .mode = 194
    .ydst = 1.47M/1.47M {
        .file = /storage/BE60-0FE5/ylog/ylog/kernel/000
        .max_segment  = 6
        .max_segment_size  = 25.00M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0   bypass值为：0  log不写入磁盘
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

**6、 设置 cache timeout 时间**

命令格式：adb shell ylog_cli ylog xxx cache timeout 500

响应格式：输出 log 的数据结构信息

**7、 设置历史 log 保留个数**

命令格式：adb shell log_cli history_n N

响应格式：N\n

### 3.2.4 删除类命令行

**1、只删除 last_ylog**

命令格式：adb shell rylog

响应格式： done\n

**2、 删除 ylog 和 last_ylog 文件夹**

命令格式：adb shell ryloga

响应格式：done\n

**3、删除 ylog 和 last_ylog 文件夹，并重启 ylog 进程**

命令格式：adb shell rylogr

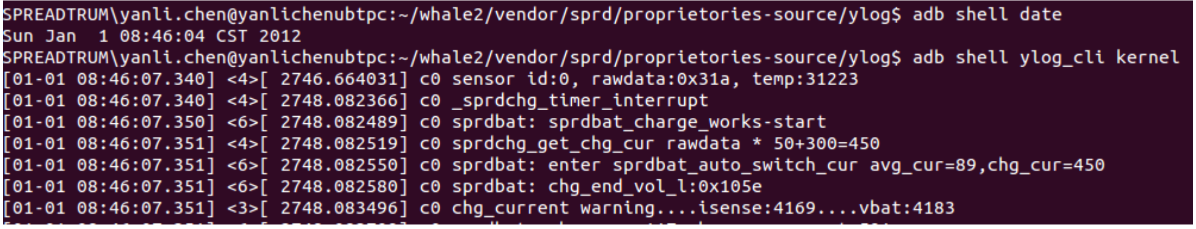
响应格式：done\n

### 3.2.5 其他命令行

**1. 获取实时 kernel log**

命令格式：adb shell ylog_cli kernel

响应格式：打印实时 kernel log

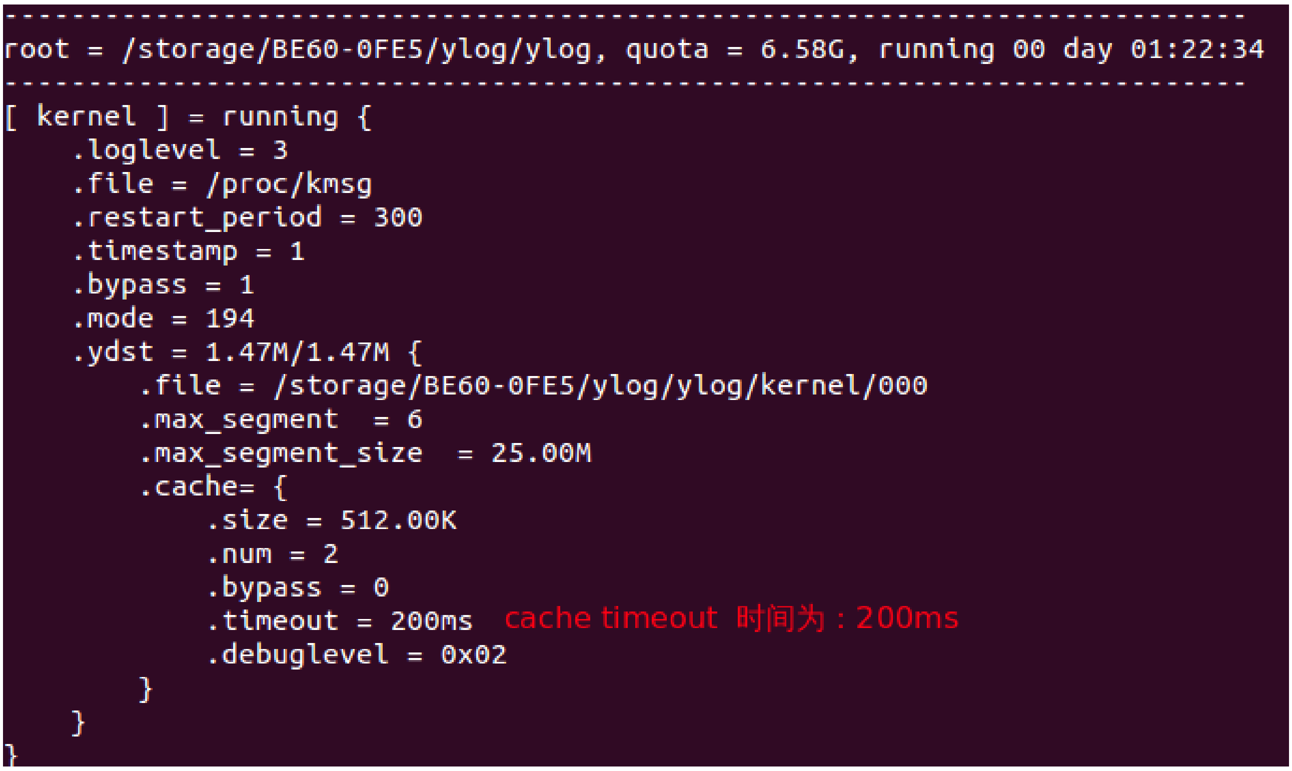


```
SPREADTRUM\yanli.chen@yanlichenubtpc:~/whale2/vendor/sprd/proprietories-source/ylog$ adb shell date
Sun Jan  1 08:46:04 CST 2012
SPREADTRUM\yanli.chen@yanlichenubtpc:~/whale2/vendor/sprd/proprietories-source/ylog$ adb shell ylog_cli kernel
[01-01 08:46:07.340] <4>[ 2746.664031] c0 sensor id:0, rawdata:0x31a, temp:31223
[01-01 08:46:07.340] <4>[ 2748.082366] c0 _sprdchg_timer_interrupt
[01-01 08:46:07.350] <6>[ 2748.082489] c0 sprdbat: sprdbat_charge_works-start
[01-01 08:46:07.351] <4>[ 2748.082519] c0 sprdchg_get_chg_cur rawdata * 50+300=450
[01-01 08:46:07.351] <6>[ 2748.082550] c0 sprdbat: enter sprdbat_auto_switch_cur avg_cur=89,chg_cur=450
[01-01 08:46:07.351] <6>[ 2748.082580] c0 sprdbat: chg_end_vol_l:0x105e
[01-01 08:46:07.351] <3>[ 2748.083496] c0 chg_current warning....isense:4169....vbat:4183
```

**2、将 cache 中内容刷新到磁盘**

命令格式：adb shell ylog_cli flush

响应格式：无



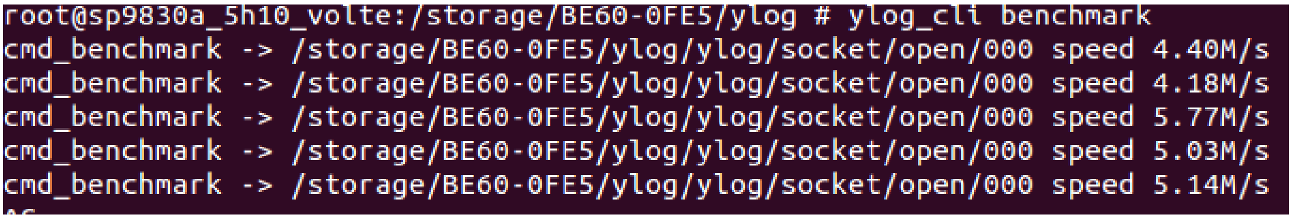
```
--------------------------------------------------------------------------------
root = /storage/BE60-0FE5/ylog/ylog, quota = 6.58G, running 00 day 01:22:34
--------------------------------------------------------------------------------
[ kernel ] = running {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 1
    .mode = 194
    .ydst = 1.47M/1.47M {
        .file = /storage/BE60-0FE5/ylog/ylog/kernel/000
        .max_segment  = 6
        .max_segment_size  = 25.00M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0
            .timeout = 200ms   cache timeout  时间为：200ms
            .debuglevel = 0x02
        }
    }
}
```

**3、评测磁盘读写速率 benchmark 测试**

a. 不加时间戳的 benchmark 测试：

命令格式：adb shell benchmark

相应格式：

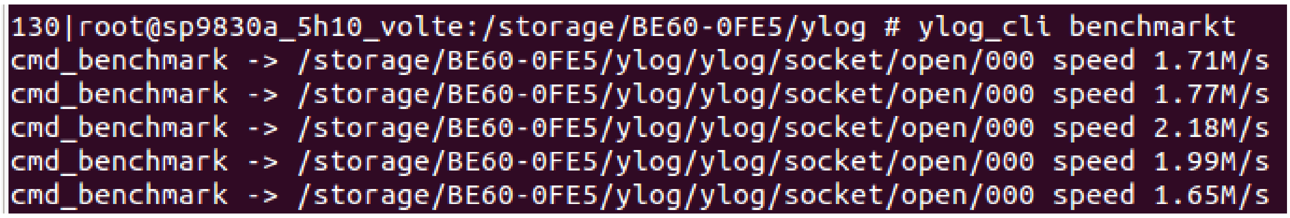


```
root@sp9830a_5h10_volte:/storage/BE60-0FE5/ylog # ylog_cli benchmark
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 4.40M/s
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 4.18M/s
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 5.77M/s
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 5.03M/s
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 5.14M/s
```

b. 加时间戳的 benchmark 测试

命令格式：adb shell benchmarkt

相应格式：



```
130|root@sp9830a_5h10_volte:/storage/BE60-0FE5/ylog # ylog_cli benchmarkt
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 1.71M/s
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 1.77M/s
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 2.18M/s
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 1.99M/s
cmd_benchmark -> /storage/BE60-0FE5/ylog/ylog/socket/open/000 speed 1.65M/s
```

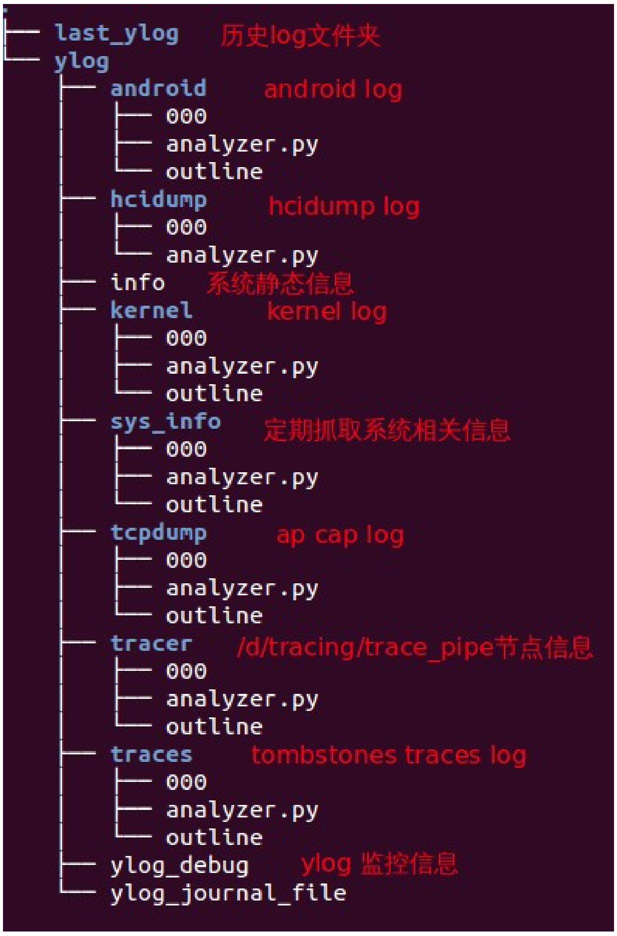
# 4. ylog 后期 log 分析方法和技巧

## 4.1 ylog 目录结构

不插 t 卡时，ylog 存储在 data 分区中，默认 quota 值为 200M，data 分区只保留本次 log，不保留历史 log。当插上 t 卡挂载成功后，会计算 t 卡的可用空间，将可用空间的 90%作为 ylog 的 quota 值，用来动态分配给各个 ydst。同时将 data 分区下的 log move 到 t 卡中，以追加方式存储后续 log。当磁盘空间不足时，ylog 会尝试缩小自身，尽量释放更多空间存储后续 log，或为自己设置新的磁盘配额空间,动态调

整自己,以新的配额大小规划 ydst 段数或段大小。

插有 t 卡时，在 t 卡挂载状态不出现异常和不重启 ylog 总开关的情况下，ylog 会将上次手机运行周期内的 log 存储到 last_ylog 文件夹中，并重命名为 ylog1，同时将上上次手机运行周期内的 log 重命名为 ylog2，依次类推，默认最多保留 5 次历史 log，可通过 adb shell ylog_cli history_n N 命令将历史 log 个数设置为 N 次。t 卡中 ylog 存储的 log 目录结构为：



```
├── last_ylog    历史log文件夹
└── ylog
    ├── android      android log
    │   ├── 000
    │   ├── analyzer.py
    │   └── outline
    ├── hcidump      hcidump log
    │   ├── 000
    │   └── analyzer.py
    ├── info    系统静态信息
    ├── kernel       kernel log
    │   ├── 000
    │   ├── analyzer.py
    │   └── outline
    ├── sys_info    定期抓取系统相关信息
    │   ├── 000
    │   ├── analyzer.py
    │   └── outline
    ├── tcpdump      ap cap log
    │   ├── 000
    │   ├── analyzer.py
    │   └── outline
    ├── tracer    /d/tracing/trace_pipe节点信息
    │   ├── 000
    │   ├── analyzer.py
    │   └── outline
    ├── traces     tombstones traces log
    │   ├── 000
    │   ├── analyzer.py
    │   └── outline
    ├── ylog_debug     ylog 监控信息
    └── ylog_journal_file
```

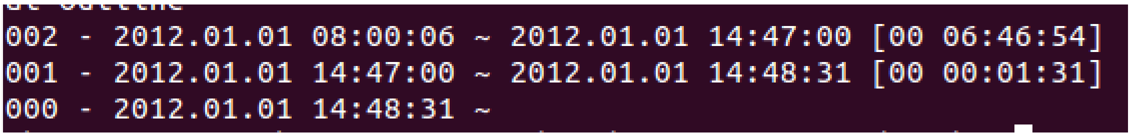
其中 000 为当前 log 存储文件，当 000 大小达到 log 单个文件大小上限值时，会翻转为 001，并新建 000 用来继续存储 log，数字越小，表示 log 越新。当 log 文件达到翻转上限时，会根据 ydst 的属性，判断是停止存储 log 以保留原始 log，还是进行将最老的 log 删除来释放空间

analyzer.py 脚本为 log 离线解析文件，支持 windows/MAC/linux 等多个平台

outline 文件记录每段 log 的起止时间和填满该段 log 所用时间。在分析问题时，可根据问题发生时间点直接定位对应段 log，用 vim 打开 outline 文件，可以快速打开发生问题时间区间内的文件，详细步骤如下：

vim outline

1. 光标定位到 002
2. 然后输入 gf
3. 就可以直接打开对应 002 文件

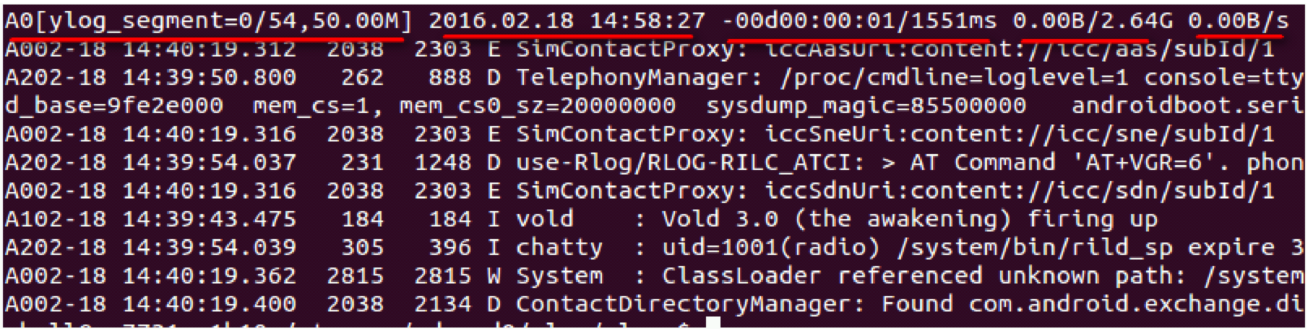


```
002 - 2012.01.01 08:00:06 ~ 2012.01.01 14:47:00 [00 06:46:54]
001 - 2012.01.01 14:47:00 ~ 2012.01.01 14:48:31 [00 00:01:31]
000 - 2012.01.01 14:48:31 ~
```

Log 内容中包含的信息：

其中 0 表示该文件为第 000 文件，54 表示最多可以产生 54 段

50.00M 表示每段文件最多存储 50M 大小数据



```
A0[ylog_segment=0/54,50.00M] 2016.02.18 14:58:27 -00d00:00:01/1551ms 0.00B/2.64G 0.00B/s
A002-18 14:40:19.312  2038  2303 E SimContactProxy: iccAasUri:content://icc/aas/subId/1
A202-18 14:39:50.800   262   888 D TelephonyManager: /proc/cmdline=loglevel=1 console=tty
d_base=9fe2e000  mem_cs=1, mem_cs0_sz=20000000  sysdump_magic=85500000   androidboot.seri
A002-18 14:40:19.316  2038  2303 E SimContactProxy: iccSneUri:content://icc/sne/subId/1
A202-18 14:39:54.037   231  1248 D use-Rlog/RLOG-RILC_ATCI: > AT Command 'AT+VGR=6'. phon
A002-18 14:40:19.316  2038  2303 E SimContactProxy: iccSdnUri:content://icc/sdn/subId/1
A102-18 14:39:43.475   184   184 I vold     : Vold 3.0 (the awakening) firing up
A202-18 14:39:54.039   305   396 I chatty   : uid=1001(radio) /system/bin/rild_sp expire 3
A002-18 14:40:19.362  2815  2815 W System   : ClassLoader referenced unknown path: /system
A002-18 14:40:19.400  2038  2134 D ContactDirectoryManager: Found com.android.exchange.di
```

## 4.2 ylog 文件离线解析

analyzer.py 中包含 token 映射信息

'A0':'main.log',

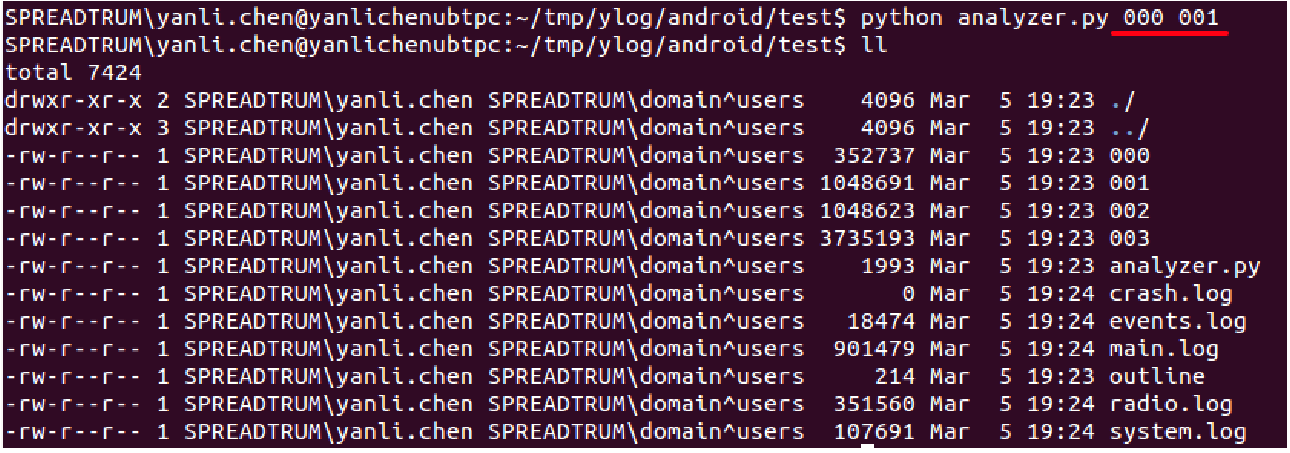
'A1':'system.log',

'A2':'radio.log',

'A3':'events.log',

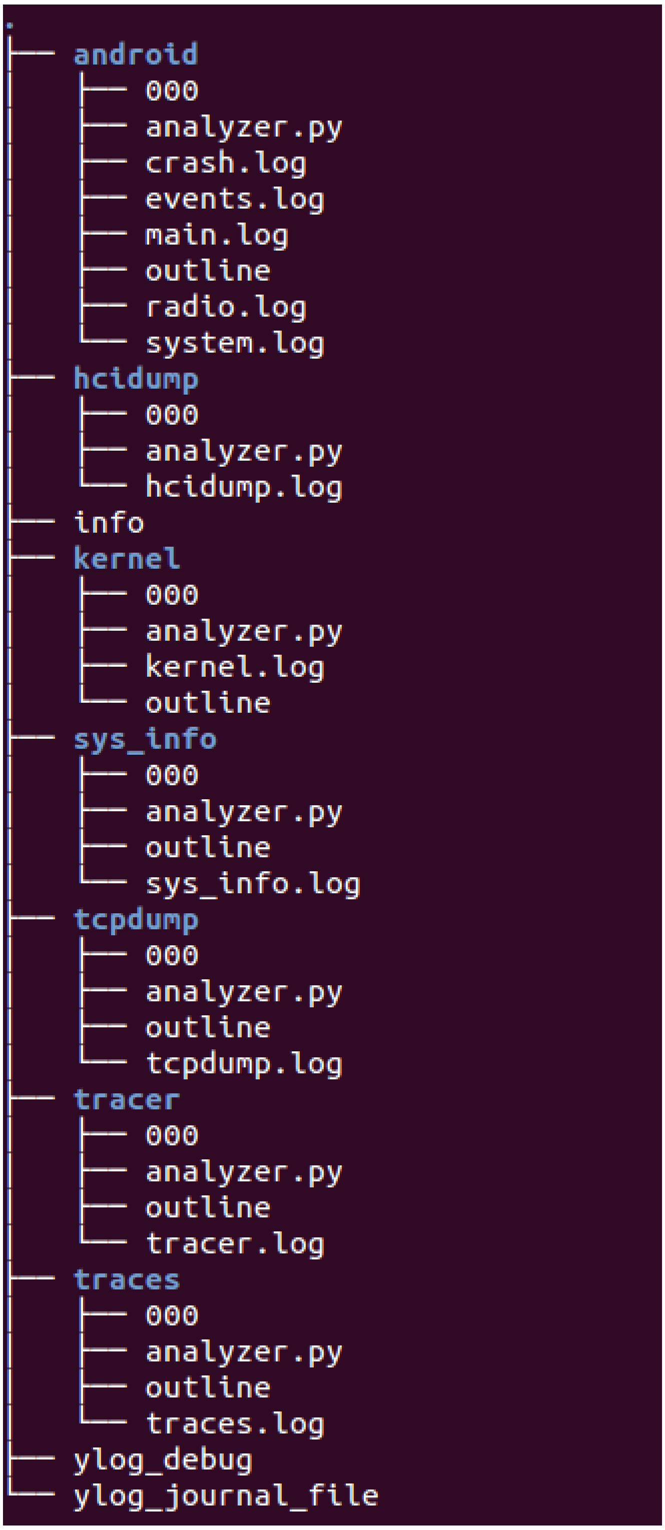
'A4':'crash.log',

将 ylog pull 到本地，在对应文件夹目录中执行 python analyzer.py，就可以拿到解析后的 log，目前解析支持将 log 合并和拆分，可根据自身需求进行定制解析。另外，离线解析还支持指定段解析，例如：执行 python analyzer.py 000 001，就可以单独解析 000 001 这两段 log。



```
SPREADTRUM\yanli.chen@yanlichenubtpc:~/tmp/ylog/android/test$ python analyzer.py 000 001
SPREADTRUM\yanli.chen@yanlichenubtpc:~/tmp/ylog/android/test$ ll
total 7424
drwxr-xr-x 2 SPREADTRUM\yanli.chen SPREADTRUM\domain^users    4096 Mar  5 19:23 ./
drwxr-xr-x 3 SPREADTRUM\yanli.chen SPREADTRUM\domain^users    4096 Mar  5 19:23 ../
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users  352737 Mar  5 19:23 000
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users 1048691 Mar  5 19:23 001
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users 1048623 Mar  5 19:23 002
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users 3735193 Mar  5 19:23 003
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users    1993 Mar  5 19:23 analyzer.py
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users       0 Mar  5 19:24 crash.log
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users   18474 Mar  5 19:24 events.log
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users  901479 Mar  5 19:24 main.log
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users     214 Mar  5 19:23 outline
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users  351560 Mar  5 19:24 radio.log
-rw-r--r-- 1 SPREADTRUM\yanli.chen SPREADTRUM\domain^users  107691 Mar  5 19:24 system.log
```

解析后的 log 目录为：



```
.
├── android
│   ├── 000
│   ├── analyzer.py
│   ├── crash.log
│   ├── events.log
│   ├── main.log
│   ├── outline
│   ├── radio.log
│   └── system.log
├── hcidump
│   ├── 000
│   ├── analyzer.py
│   └── hcidump.log
├── info
├── kernel
│   ├── 000
│   ├── analyzer.py
│   ├── kernel.log
│   └── outline
├── sys_info
│   ├── 000
│   ├── analyzer.py
│   ├── outline
│   └── sys_info.log
├── tcpdump
│   ├── 000
│   ├── analyzer.py
│   ├── outline
│   └── tcpdump.log
├── tracer
│   ├── 000
│   ├── analyzer.py
│   ├── outline
│   └── tracer.log
├── traces
│   ├── 000
│   ├── analyzer.py
│   ├── outline
│   └── traces.log
├── ylog_debug
└── ylog_journal_file
```