ylog handbook

(Tools. TianJin)

1. design architecture....4
1.1 structure design....4
1.1.1 ylog source....4
1.1.2 ydst....5
1.1.3 cache....5
2. ylog module....5
2.1 corresponding list between ylog source and ydst....7
2.2 ylog source....7
2.3 ylog attribute list....10
2.4 ylog function....11
3. ylog_cli command line....14
3.1 overview of ylog_cli command line....14
3.2 explanation of ylog_cli command line....14
3.2.1 query class....14
3.2.2 switch class....17
3.2.3 setting class....18
3.2.4 delete class....21
3.2.5 other class....22
3.2.6 modem/wcn log rate statistic....24
4. log analysis methods and techniques....26
4.1 ylog directory structure....26
4.2 ylog files analysis off line....28
4.2.1 python analyzer.py....28
4.2.2 sgm log analyzer....31
4.2.3 ylog_verify_pc.sh script use help....33
4.3 analytical method of native crash and anr log....33

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
1.2	Yanli Chen	2016/03/28	Add ylog_cli at/print2android/print2kernel cmd
1.2e	Yue Zhao	2016/03/29	Translate to English
1.3	Yanli Chen	2016/04/01	Add modem/wcn log speed statistics
1.3e	Yue Zhao	2016/04/03	Translate to English
1.4	Yanli Chen	2016/05/06	Add snapshot/mtp/screen cmd,capture logcat info when anr/tombstones, add sgm log

1. design architecture

ylog (your log) is customizable background resident service, which is used for log collection, intelligent analysis and behaviour statistic,etc.

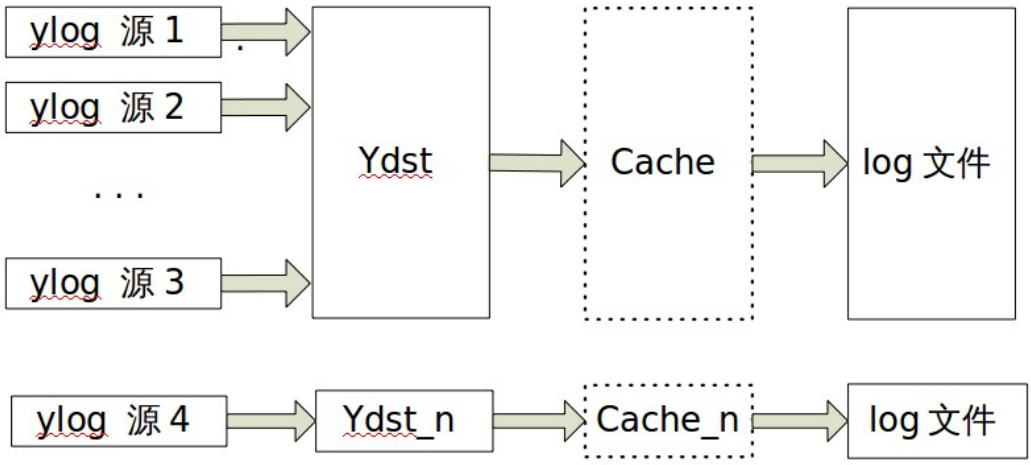
1.1 structure design

ylog service is designed for three parts ：ylog source, ydst and cache.

ylog source ：input data source.

ydst： data output.

cache：cache output data.



1.1.1 ylog source

ylog source focuses on describing input-data, such as input type, token and restart time.

Currently four input types are supported, which are common file, socket, popen and inotify monitor.

ylog source reference form：

```
{
        .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 concentrates on output-data that comes from ylog source. ydst is designed for dividing up data and controlling numbers of segments. When there is a large dataflow, much data lose caused by disk transient efficiency occurred. Under this occasion, ydst can be attached to cache for alleviating big data and high throughput.

ydst reference form：

```
[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

cache focuses on how to cache output-data from ydst, and write them into disk storage at the appropriate time. cache has an attribute named Timeout, which can ensure that output-data can be written to disk as soon as possible. Timeout is used to define an action, where if it doesn't run out of cache line within Timeout, the data in cache line will be flushed to disk with force.

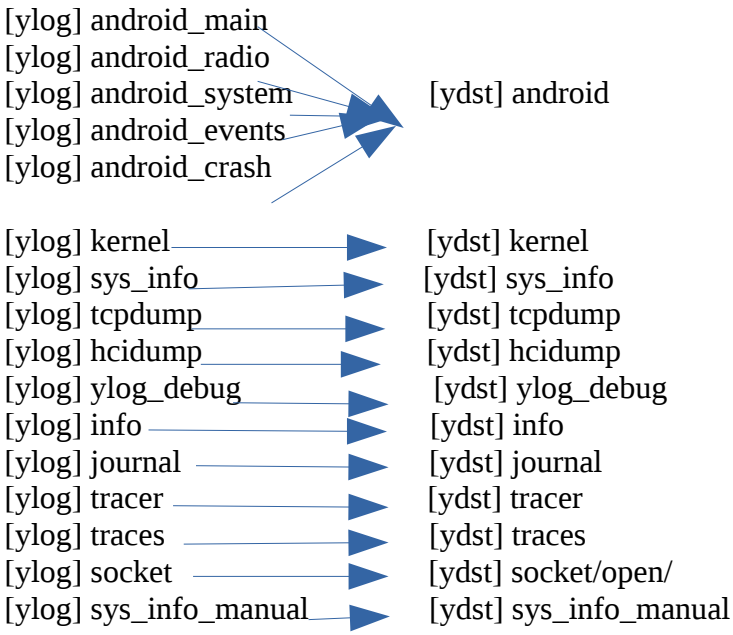
```
[OS_YDST_TYPE_ANDROID] = {
        .size = 512 * 1024,
        .num = 4,
        .timeout = 1000, /* ms */
        .debuglevel = CACHELINE_DEBUG_CRITICAL,
    }
```

2. ylog module

ylog is a a user-space service application started by init.rc, and its default startup behavior after the boot can be aided controlled by attribute value of persist.ylog.enabled . ylog not only supports multiple ylog source parallel inputting, but multiple ydst parallel outputting. It can assign ydst

output, set ydst whether attached to cache, define restart time interval, make up whether overwrite when reach the limit of segments numbers, which is based on data structure of ylog source.

2.1 corresponding list between ylog source and ydst



2.2 ylog source

1. Android

It covers five types log , including android_main, android_radio, android_system, android_events, android_crash, which output into a common ydst. Meanwhile, in order to improve disk efficiency, the ydst is attached to a 1M-sized cache by default.

2. kernel

It collects kernel log, and kernel ydst is attached to a 1M-sized cache by default.

3. sys_info

It collects some system-related log in regular time(two minutes). The included specific information is listed as follows.

/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

It collects ap cap log，and tcpdump ydst is attached to a 1M-sized cache by default.

5. hcidump

It collects hci bt log，and hcidump ydst is attached to a 1M-sized cache by default.

6. ylog_debug

It collects debugging information related with log, includes log rate, space usage, running status, etc, usually takes statistics once every 20 minutes.

/system/bin/ylog_cli ylog
/system/bin/ylog_cli speed
/system/bin/ylog_cli space
getprop ylog.killed

7. info

It collects system-related static information, which is captured only once when ylog starts.

/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

all the thread pid and tid of ylog

8. journal

It's designed for monitoring ylog running status, monitor switch, and log file deletion,etc.

```
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

It can make data slice storage read from /d/tracing/trace_pipe, which allows to add debug information, and ajust quota values for time critical drive modules, such as usb.

10. traces

It collects tombstones in the case of native crash, and stack trace in the case of anr.

Besides, traces contain logcat info when the anr or natitive crash occure.The log file name contain anr/tombstones number and occure time.

```
root@sp9832a_2h11_volte:/storage/4DD6-1926/ylog/ylog/traces # ls
0001.20120101-104810.339.tombstone
0001.20120101-104810.339.tombstone.logcat
0001.20120101-104832.400.anr
0001.20120101-104832.400.anr.logcat
```

11. socket

It opens ylog socket to receive dataflow from evaluation software called ylog_benchmark, which can search software and system bottleneck.

12. sys_info_manual

It can trigger ylog to capture the execution results of "busybox netstat -ap" and "ps -t" manually.

2.3 ylog attribute list

1.the following attribute values indicate running status of all kinds of log.

```
[init.svc.ylog]: [running]
[ylog.killed]: [0]
[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.benchmark_socket]: [running]
[ylog.svc.cp_5mode]: [stopped]
[ylog.svc.cp_wcn]: [stopped]
[ylog.svc.hcidump]: [running]
[ylog.svc.info]: [stopped]
[ylog.svc.kernel]: [running]
[ylog.svc.sgm.cpu_memory]: [running]
[ylog.svc.snapshot]: [running]
[ylog.svc.socket]: [running]
[ylog.svc.sys_info]: [running]
```

```
[ylog.svc.sys_info_manual]: [stopped]
[ylog.svc.tcpdump]: [stopped]
[ylog.svc.tracer]: [running]
[ylog.svc.traces]: [running]
[ylog.svc.ylog_debug]: [running]
```

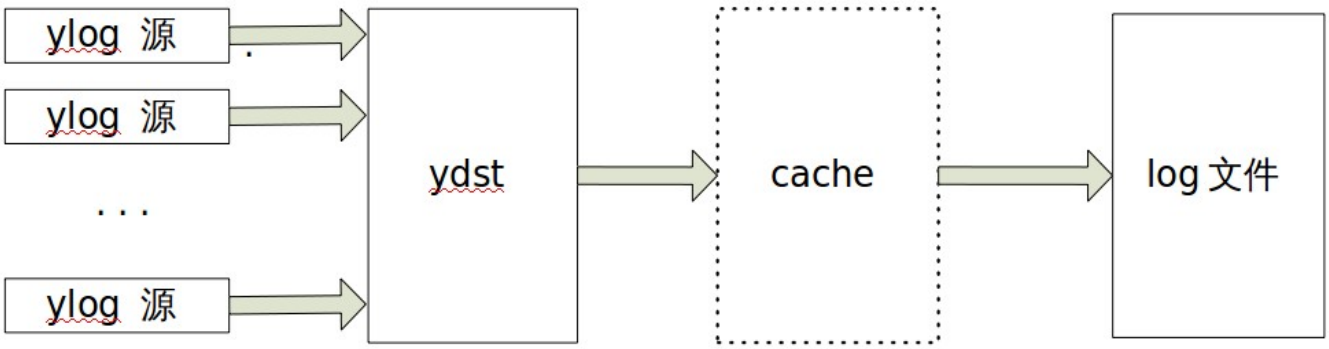
2.ylog service master switch control

[persist.ylog.enabled]: [1]

3. TF card unloaded numbers

[ylog.killed]: [0]

2.4 ylog function



1. For purpose of easy to post-maintenance, it abstracts three data model, including ylog ydst and cache.
2. cache is used for alleviating disk pressure under normal circumstances, and log loss with high data throughput.
3. Keyword filter can retrieve received data, and trigger corresponding action response, which executes shell or other operations to collect much more information, thus to help localize the source of problem. It can make ylog more smart for real-time performance and acquisition for environment.
4. The upper limit for log files is distributed by actual free space, realizing dynamic adjustment of quota values.

```
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
```

5. ylog process can be monitored, and journal is stored into /data/ylog/ylog_journal_file permanently for aided analysis of system behavior.

```
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 can look up the highest ten throughput information since the system started: the productive rate of log.

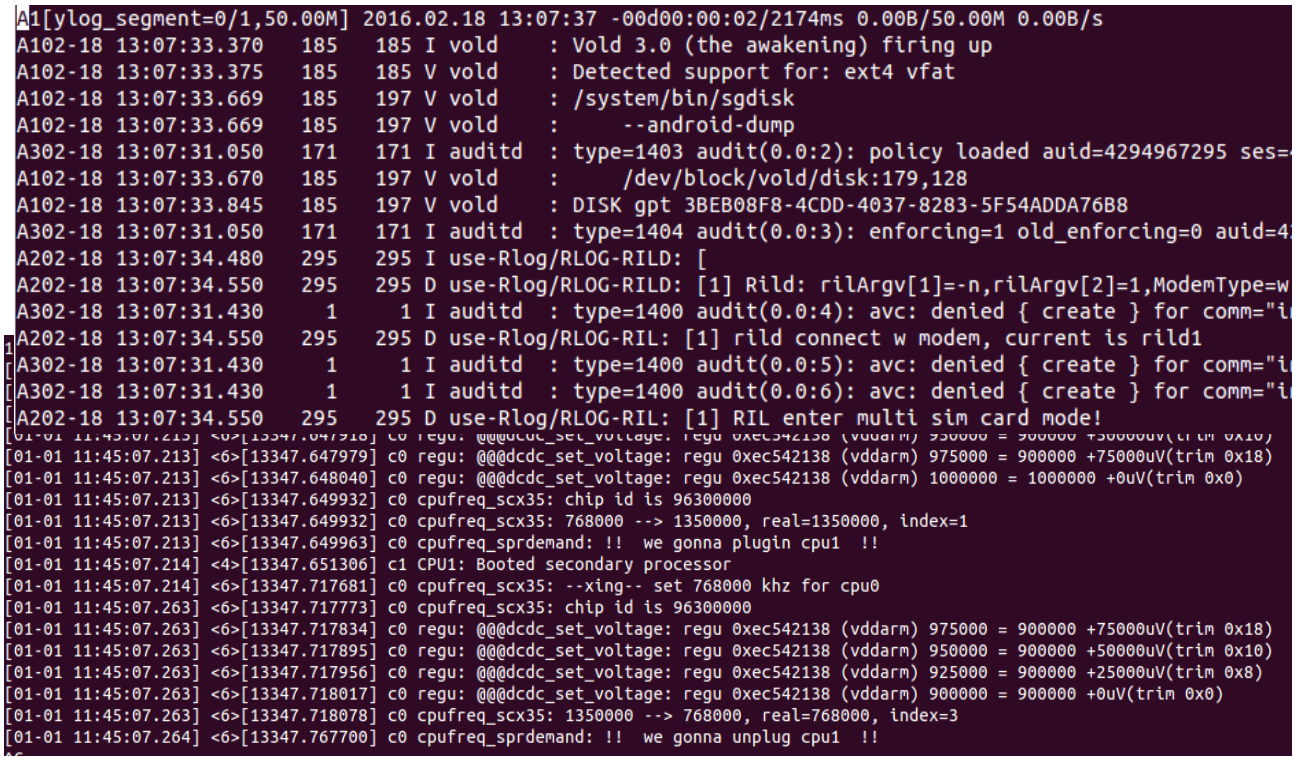
```
Transfered 3.72G Has run 00 day 01:15:01 avg_speed=867.17K/s
01. [02-18 14:06:58.600] ~ [02-18 14:06:59.600] 00 day 00:59:25 ago 20.02M/s
02. [02-18 14:08:01.618] ~ [02-18 14:08:02.618] 00 day 01:00:28 ago 20.02M/s
03. [02-18 14:06:10.585] ~ [02-18 14:06:11.585] 00 day 00:58:37 ago 20.01M/s
04. [02-18 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. Built-in analyzer.py provides for off-line data processing.

8. Multiple data can be output into one ydst, which ensure that log time-stamp are included in the same time form.

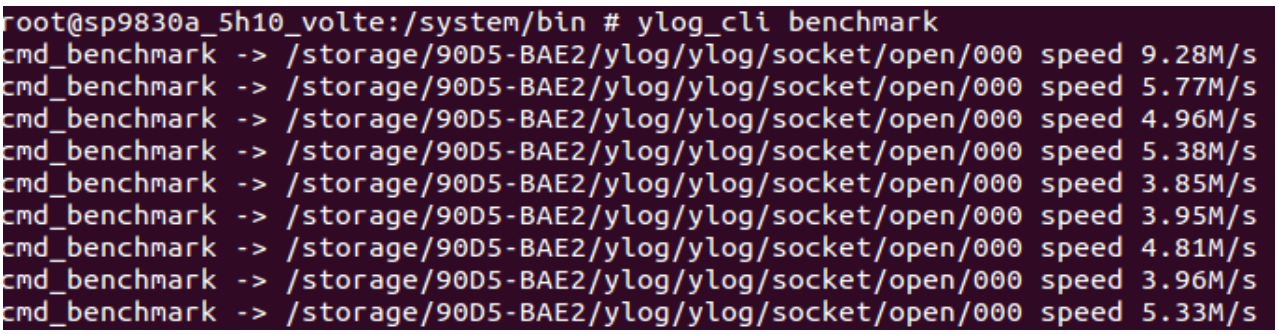
9. ylog_cli kernel can capture kernel log in real time, and simultaneous operation of four independent ylog_cli are supported.

10. It can judge whether log reaches the ceiling by checking the numbers of segments, whether log overwriting has occurred.



11. sys_info performs system information statistics collection operations, and data will be slice storage.

12. ylog_cli benchmark can evaluate upper limit of read-write rate by disks and ylog.



13. It can make data slice storage read from /d/tracing/trace_pipe, which allows adding debug information, and ajust quota values for time critical drive modules, such as usb.

3. ylog_cli command line

Help information of ylog_cli command line can be obtained by “adb shell ylog_cli”.

3.1 overview of ylog_cli command line

```
=== [ 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 explanation of ylog_cli command line

ylog_cli connects to ylog process through sockets, and support multi-class command lines, which are queries, switches, settings, deleting, etc.

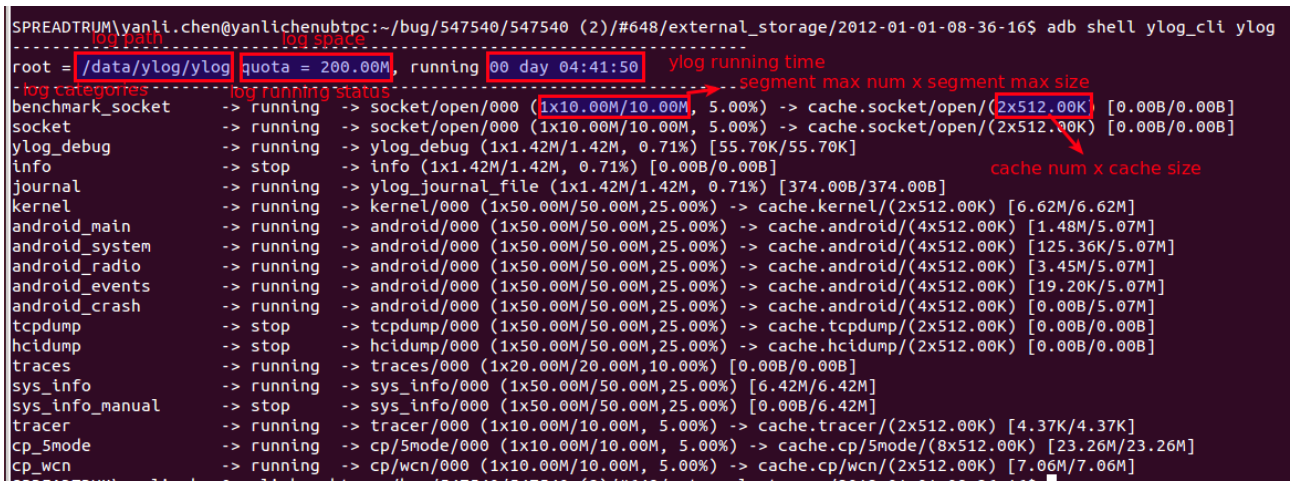
3.2.1 query class

1. Look up the top ten read/write rate since ylog boot.

```
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
```

2. **Look up all kinds of log information, such as switch status, upper limit of segments and single segment's size, log-transfer counter.**

Command format： adb shell ylog_cli ylog Response format： No Root: display the current log storage path. Quota : the upper limit of log storage space. Running： show running time of ylog.




```
SPREADTRUM\yanli.chen@yanlichenubtpc:~/bug/547540/547540 (2)/#648/external_storage/2012-01-01-08-36-16$ adb shell ylog_cli ylog
--------------------------------------------------------------------------------
root = /data/ylog/ylog quota = 200.00M, running 00 day 04:41:50
--------------------------------------------------------------------------------
benchmark_socket      -> running  -> socket/open/000 (1x10.00M/10.00M, 5.00%) -> cache.socket/open/(2x512.00K) [0.00B/0.00B]
socket                -> running  -> socket/open/000 (1x10.00M/10.00M, 5.00%) -> cache.socket/open/(2x512.00K) [0.00B/0.00B]
ylog_debug            -> running  -> ylog_debug (1x1.42M/1.42M, 0.71%) [55.70K/55.70K]
info                  -> stop     -> info (1x1.42M/1.42M, 0.71%) [0.00B/0.00B]
journal               -> running  -> ylog_journal_file (1x1.42M/1.42M, 0.71%) [374.00B/374.00B]
kernel                -> running  -> kernel/000 (1x50.00M/50.00M,25.00%) -> cache.kernel/(2x512.00K) [6.62M/6.62M]
android_main          -> running  -> android/000 (1x50.00M/50.00M,25.00%) -> cache.android/(4x512.00K) [1.48M/5.07M]
android_system        -> running  -> android/000 (1x50.00M/50.00M,25.00%) -> cache.android/(4x512.00K) [125.36K/5.07M]
android_radio         -> running  -> android/000 (1x50.00M/50.00M,25.00%) -> cache.android/(4x512.00K) [3.45M/5.07M]
android_events        -> running  -> android/000 (1x50.00M/50.00M,25.00%) -> cache.android/(4x512.00K) [19.20K/5.07M]
android_crash         -> running  -> android/000 (1x50.00M/50.00M,25.00%) -> cache.android/(4x512.00K) [0.00B/5.07M]
tcpdump               -> stop     -> tcpdump/000 (1x50.00M/50.00M,25.00%) -> cache.tcpdump/(2x512.00K) [0.00B/0.00B]
hcidump               -> stop     -> hcidump/000 (1x50.00M/50.00M,25.00%) -> cache.hcidump/(2x512.00K) [0.00B/0.00B]
traces                -> running  -> traces/000 (1x20.00M/20.00M,10.00%) [0.00B/0.00B]
sys_info              -> running  -> sys_info/000 (1x50.00M/50.00M,25.00%) [6.42M/6.42M]
sys_info_manual       -> stop     -> sys_info/000 (1x50.00M/50.00M,25.00%) [0.00B/6.42M]
tracer                -> running  -> tracer/000 (1x10.00M/10.00M, 5.00%) -> cache.tracer/(2x512.00K) [4.37K/4.37K]
cp_5mode              -> running  -> cp/5mode/000 (1x10.00M/10.00M, 5.00%) -> cache.cp/5mode/(8x512.00K) [23.26M/23.26M]
cp_wcn                -> running  -> cp/wcn/000 (1x10.00M/10.00M, 5.00%) -> cache.cp/wcn/(2x512.00K) [7.06M/7.06M]
```

3. **Look up switch status of a single log, the same as ylog.svc.**xxx

Command format：ylog_cli ylog xxx get started Response format：No Opened state： 1\n Closed state ： 0\n

Note: xxx can be android_main, android_system, android_radio, android_events, android_crash, tcpdump, hcidump, kernel.

4. **Look up log storage path**

Command format： adb shell ylog_cli cpath Response format： No Without TF card： /data/ylog/ylog Note : Internal storage can't support saving history log.

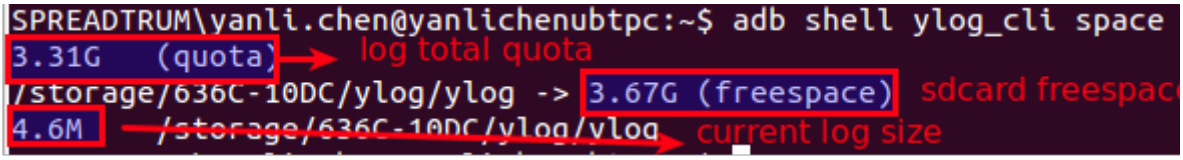
With TF card: /storage/BE60-0FE5/ylog/ylog

/storage/BE60-0FE5/ylog/last_ylog

5. Look up memory space of log

Command format: adb shell ylog_cli space

Response format: No



6. Look up free space of disk that log resides on

Command format: adb shell ylog_cli freespace

Response format: /storage/BE60-0FE5/ylog/ylog -> 7.32G

7. Look up disk path that log resides on

Command format: adb shell ylog_cli rootdir

Response format: /storage/BE60-0FE5

8. Look up history log times

Command format: adb shell ylog_cli history_n

Response format: 5\n keep five history log by default

9. Look up relevant information of single ylog source

Command format: 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
        }
    }
}
```

Response format：Output information, including input types of ylog source, storage path, cache size, ect.

Note：xxx can be android_main, android_system, android_radio, android_events, android_crash, tcpdump, hcidump, kernel.

10. Look up all the relevant log information

Command format：adb shell ylog_cli ylog all

Response format：Output all log information, including input types, storage path, cache size, ect.

3.2.2 switch class

1. Main switch settings of ylog

Command format：

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

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

2. All switch uniform settings of log

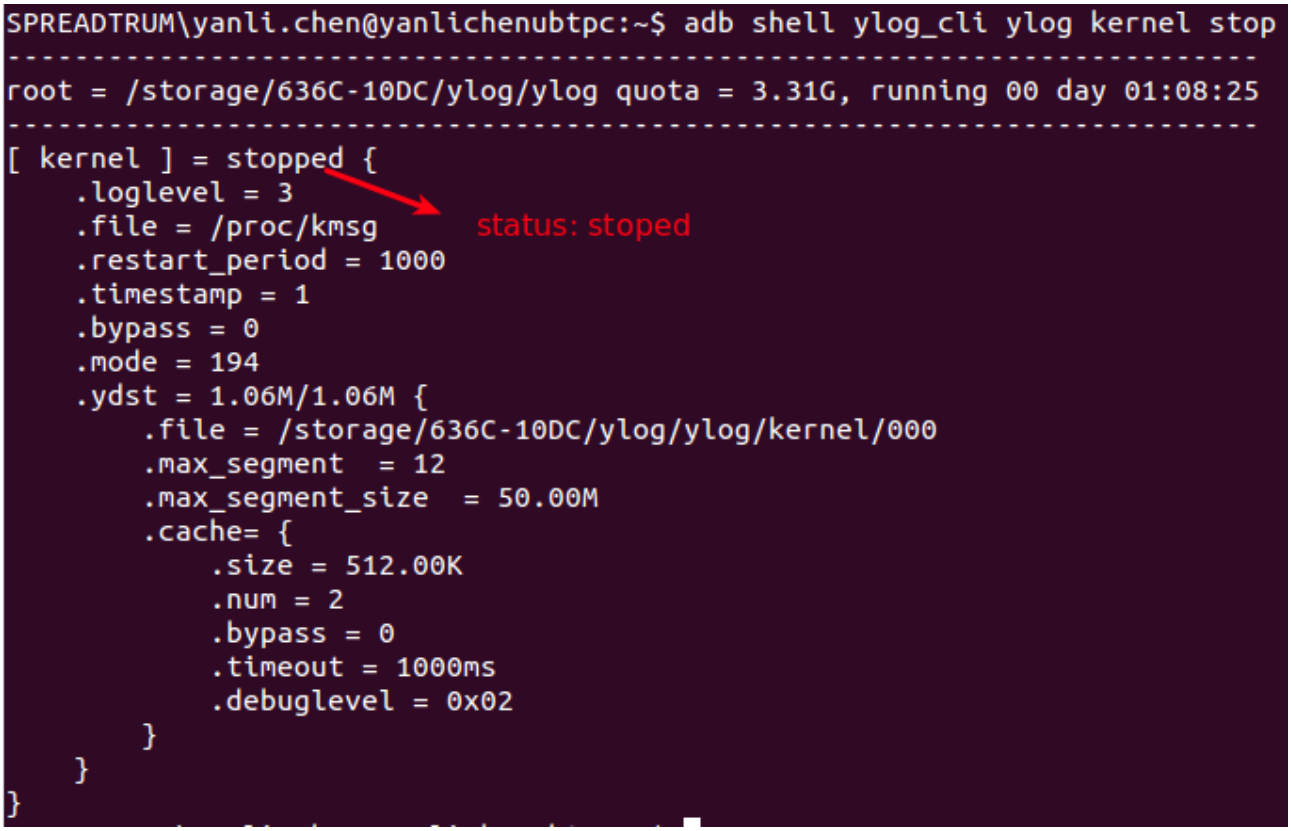
Command format: adb shell ylog_cli ylog all start/stop

Response format: Output all log information, including switch status, storage path, cache size, ect.

3. Switch settings of single log

Command format: adb shell ylog_cli ylog xxx start/stop

Response format: output data structure information of the log



3.2.3 setting class

1. Setting up maximum values of log segments

Command format: adb shell ylog_cli ylog xxx ydst max_segment n

Response format: output data structure information of the log

2. Setting up individual log file size, unit: M

Command format: adb shell ylog_cli ylog xxx ydst max_segment_size 20

Response format: output data structure information of the log

```
SPREADTRUM\yanli.chen@yanlichenubtpc:~$ adb shell ylog_cli ylog kernel ydst max_segment 3
------------------------------------------------------------------------
root = /storage/636C-10DC/ylog/ylog, quota = 1.44G, running 00 day 00:05:31
------------------------------------------------------------------------
[ kernel ] = running {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 0
    .mode = 194
    .ydst = 1.24M/1.24M {
        .file = /storage/636C-10DC/ylog/ylog/kernel/000
        .max_segment   = 3        → change the max_segment to 3
        .max_segment_size  = 50.00M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

```
SPREADTRUM\yanli.chen@yanlichenubtpc:~$ adb shell ylog_cli ylog kernel ydst max_segment_size 20
------------------------------------------------------------------------
root = /storage/636C-10DC/ylog/ylog, quota = 1.44G, running 00 day 00:08:31
------------------------------------------------------------------------
[ kernel ] = running {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 0
    .mode = 194
    .ydst = 1.62M/1.62M {
        .file = /storage/636C-10DC/ylog/ylog/kernel/000
        .max_segment   = 3
        .max_segment_size  = 20.00M   → change max_segment size to 20M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

3. Setting up maximum of log segments, upper limit for log size

Command format：adb shell ylog_cli ylog xxx ydst segment_size 5 20 Response format：output data structure information of the log

4. Setting up whether print time stamps during real kernel log printing by ylog_cli.

Command format：adb shell ylog_cli ylog kernel timestamp 1/0 (yes/no)

```
SPREADTRUM\yanli.chen@yanlichenubtpc:~$ adb shell ylog_cli ylog kernel ydst segment_size 5 20
------------------------------------------------------------------------
root = /storage/636C-10DC/ylog/ylog, quota = 1.44G, running 00 day 00:11:12
------------------------------------------------------------------------
[ kernel ] = running {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 0
    .mode = 194
    .ydst = 1.92M/1.92M {
        .file = /storage/636C-10DC/ylog/ylog/kernel/000
        .max_segment  = 5
        .max_segment_size  = 20.00M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

change max_segment to 5 and max_segment_size to 20 at the same time

Response format：real printing of kernel log on terminal

```
<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. Setting up whether writing log back to disk

Command format：adb shell ylog_cli ylog xxx bypass 1/0 (yes/no)

Response format：output data structure information of the log

```
SPREADTRUM\yanli.chen@yanlichenubtpc:~$ adb shell ylog_cli ylog kernel cache bypass 0
-------------------------------------------------------------------------
root = /storage/636C-10DC/ylog/ylog, quota = 1.44G, running 00 day 00:14:40
-------------------------------------------------------------------------
[ kernel ] = running {
    .loglevel = 3
    .file = /proc/kmsg
    .restart_period = 300
    .timestamp = 1
    .bypass = 0
    .mode = 194
    .ydst = 2.32M/2.32M {
        .file = /storage/636C-10DC/ylog/ylog/kernel/000
        .max_segment  = 5
        .max_segment_size  = 20.00M
        .cache= {
            .size = 512.00K
            .num = 2
            .bypass = 0          --> no write back to disk
            .timeout = 1000ms
            .debuglevel = 0x02
        }
    }
}
```

6. Setting up cache timeout

Command format：adb shell ylog_cli ylog xxx cache timeout 500 Response format：output data structure information of the log

7. Setting up number of history log

Command format：adb shell ylog_cli history_n N Response format：N\n

3.2.4 delete class

1. Deleting last_ylog only

Command format：adb shell rylog Response format：done\n

2. Deleting folders of ylog, last_ylog

Command format：adb shell ryloga Response format：done\n

3. Deleting folders of ylog, last_ylog, and restart ylog

Command format：adb shell rylogr Response format：done\n

3.2.5 other class

1. Get real-time kernel log

Command format：adb shell ylog_cli kernel Response format： print kernel log in real time

```
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. Flush cache to disk

Command format：adb shell ylog_cli flush Response format： None

3. benchmark for evaluating disk read-write rate

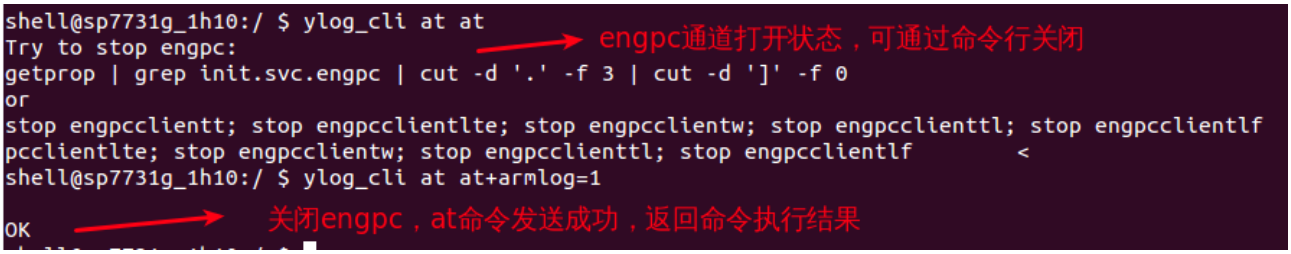
a. benchmark test without time stamps.

Command format：adb shell benchmark Response format：None

```
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. benchmark test with time stamps.

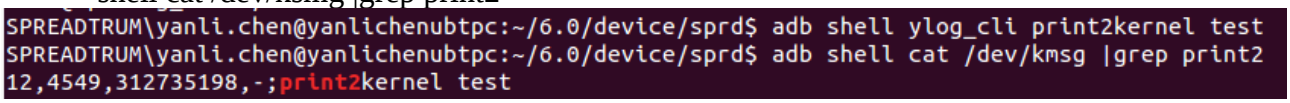
Command format：adb shell benchmarkt Response format：None



4. use ylog_cli command to save kernel log

Command format: adb shell ylog_cli print2kernel + log content
eg, adb shell ylog_cli print2kernel test

Check whether log is written to “/dev/kmsg” successfully, use the command as follows, adb shell cat /dev/ksmg |grep print2



5. use ylog_cli command to save android log

command format: adb shell ylog_cli print2android + log content
eg, adb shell ylog_cli print2android test

Check whether log is saved successfully, use the command as follows,
adb logcat |grep print2


```
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
```

6. capture snapshot info

command format: adb shell ylog_cli snapshot

respond format:

```
root@sp9832a_2h11_volte:/storage/4DD6-1926/ylog/ylog # ylog_cli snapshot
log         -- snapshot current android & kernel log, ex. ylog_cli snapshot log
mtp         -- snapshot current sdcard contents for mtp, ex. ylog_cli snapshot mtp
screen      -- snapshot current screen, ex. ylog_cli snapshot screen
```

a). adb shell ylog_cli snapshot log

capture dmesg and logcat info when the cmd is executed, the log file is named as the current time.

```
root@sp9832a_2h11_volte:/storage/4DD6-1926/ylog/ylog # ylog_cli snapshot log
log /storage/4DD6-1926/ylog/ylog/snapshot/log/20120101-113803.549/
```

b). adb shell ylog_cli snapshot mtp

updata mtp file path, default updata ylog file path

```
root@sp9832a_2h11_volte:/storage/4DD6-1926/ylog/ylog # ylog_cli snapshot mtp
mtp /storage/4DD6-1926/ylog
```

c). adb shell ylog_cli snapshot screen

screen , the picture is saved the screen folder

```
screen /storage/4DD6-1926/ylog/ylog/snapshot/screen/20120101-114656.096.png
```

Also, you can set picture save path and name. For example:adb shell ylog_cli snapshot screen /data/screen.png

named as screen.png picture is generated in /data directory

3.2.6 modem/wcn log rate statistic

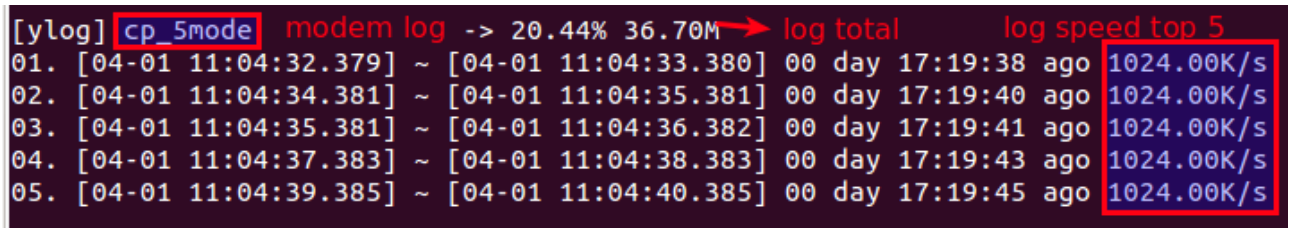
```
cp_5mode   modem log -> running  -> cp/5mode/000 (1x10.00M/10.00M, 5.00%) -> cache.cp/5mode/(8x512.00K) [76.49M/76.49M]
cp_wcn     wcn log   -> running  -> cp/wcn/000 (1x10.00M/10.00M, 5.00%) -> cache.cp/wcn/(2x512.00K) [18.88M/18.88M]
```

The rate of modem/wcn log can be counted by ylog, then the corresponding log is stored in folder of cp.

modem log get by ylog is stored in the cache sizeof 8*512*1024, and the log will be written to disk until timeout or cache runs out of the space. Theoretically, modem log stored by ylog can not be lost, in the case that modem log rate can't exceed 4M/s.

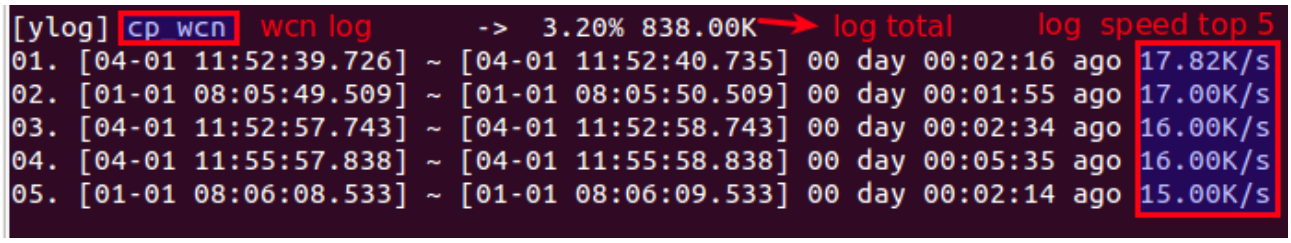
1. modem log rate statistic

Various log can be counted by command, which is “adb shell ylog_cli speed”. modem log shows in the following,



2. wcn log rate statisc

Various log can be counted by command, which is “adb shell ylog_cli speed”. wcn log shows in the following,



3 . ylog can send AT command

Command format： adb shell ylog_cli at at

Response format:

a. engpc channel is open

Try to stop engpc:

getprop | grep init.svc.engpc | cut -d '.' -f 3 | cut -d ']' -f 0

or
stop engpcclientt; stop engpcclientlte; stop engpcclientw; stop engpcclienttl; stop engpcclientlf
pcclientlte; stop engpcclientw; stop engpcclienttl; stop engpcclientlf

Command to close engpc channel:

adb shell getprop | grep init.svc.engpc | cut -d '.' -f 3 | cut -d ']' -f 0 or
adb shell stop engpcclientt; stop engpcclientlte; stop engpcclientw; stop engpcclienttl; stop engpcclientlf pcclientlte; stop engpcclientw; stop engpcclienttl; stop engpcclientlf

b. engpc channel is closed

Return value of AT command

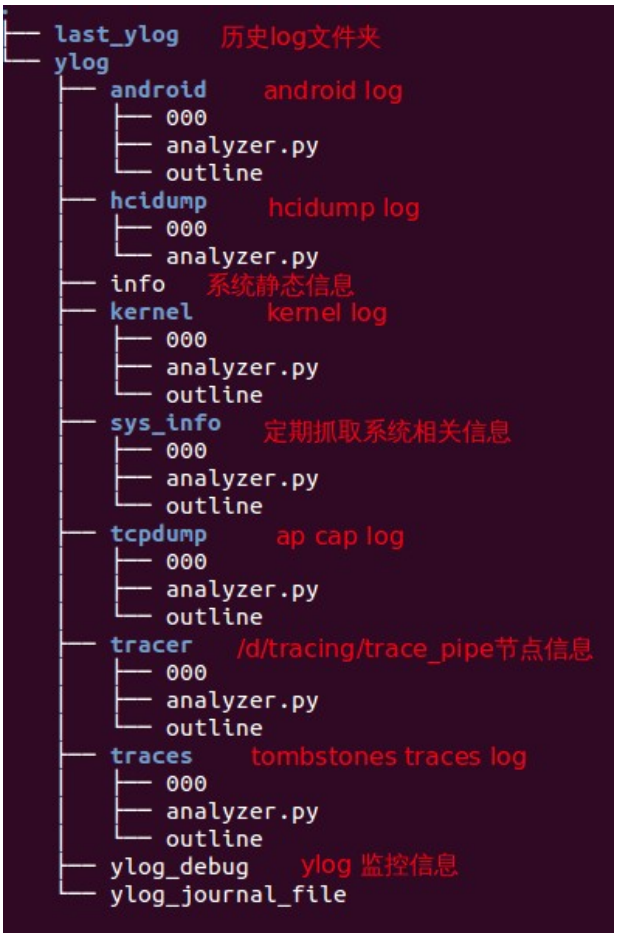
4. log analysis methods and techniques.

4.1 ylog directory structure

In the case of no TF card, ylog is stored in data partition with default quota value of 200M. Once TF card mounts successfully, quota values will be set as 90% free space of TF card, which calculated by ylog. The free space can be dynamically assigned to each ydst. Meanwhile, log stored in data partition will be moved to TF card, and subsequent log will be stored in append mode. In the condition of a disk-out-of space, ylog can try run smaller to free space for following log. In the another way, ylog can modify disk quotas to set segments numbers or size dynamically.

When TF card mounts abnormally or switch of ylog , current ylog folder is moved to last_ylog, and rename ylog to ylog1. Meanwhile, the exsited ylog1 be renamed to ylog2, and so on. The default history log keeps up to 5 times, this number can be set by command, adb shell ylog_cli history_n N.

ylog directory structure stored in TF card is described as following.



000 is current log file, and when its size reaches the upper limit of single file size, ylog can flip it to 001. At the same time, 000 is created to continue saving log. Obvious, the smaller of file name, the newer log is. When log files reach rollover limit, there are two options, one is saving original log , the other is deleting oldest log for free space, these all depend on the property of ydst.

Analyzer.py is used for parsing log files off line, and support multiple platforms, windows/MAC/linux, etc.

Outline files record log time periods and total time used for filling log. In analyzing problems, the corresponding log segments can be addressed according to point time the problem occurred, where outline files can help to open.

The detailed steps as follows.

vim outline 1. place the cursor on 002 2. input gf 3. open file named 002 directly

```
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 information is described as following.

0 stands for files named 000, 54 is the maximum of segments.

50.00M means the maximum data size of each segment.

```
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 files analysis off line

4.2.1 python analyzer.py

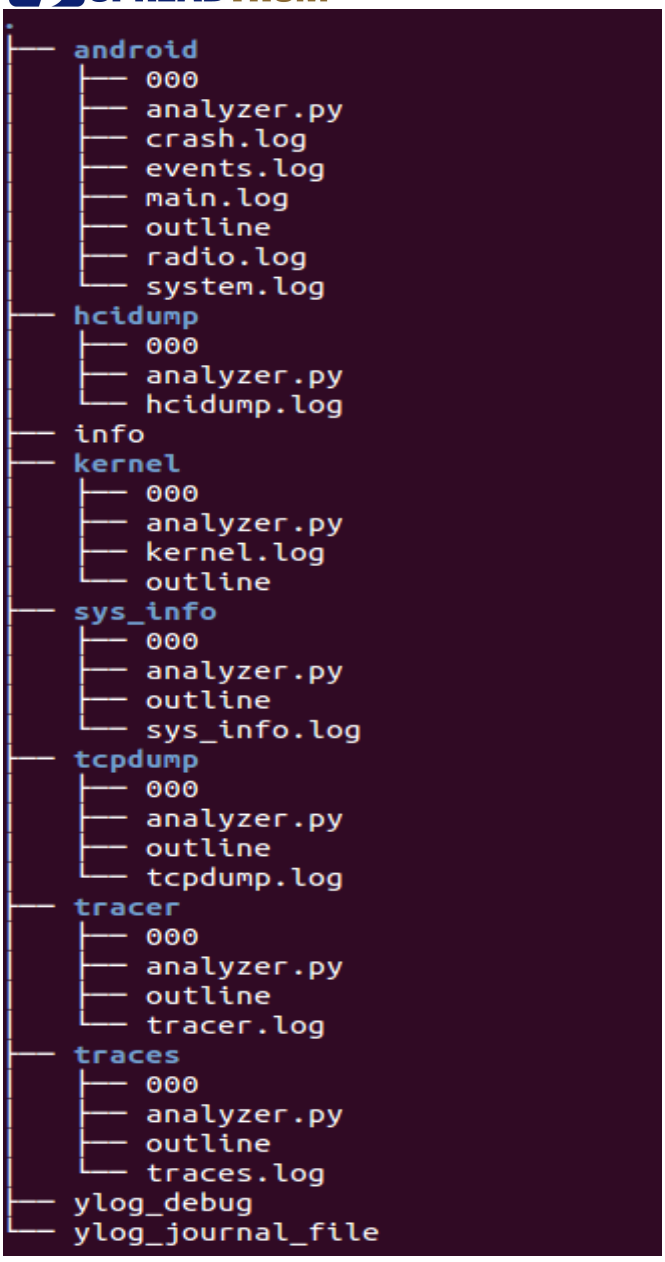
analyzer.py contains token mapping information

```
'A0':'main.log',
'A1':'system.log',
'A2':'radio.log',
'A3':'events.log',
'A4':'crash.log',
```

After pulling ylog to the local, the parsed log can be obtained by executing command “python analyzer.py” in corresponding folder. Analysis supports log resolution and combination, and it can be customized. In addition, it supports for specified segments analysis. For instance, 000 and 001 log can be parsed separately by command that “python analyzer.py 000 001”.

```
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
```

Log parsed directory is shown as follows.



4.2.2 sgm log analyzer

1. obtain sgm.toolkits tool

ubuntu os: git clone git://10.5.2.45/sgm.toolkits

windows os: (make sure adb environment and phone driver is installed)

a. http://10.5.2.45/sync/installer download the matched version: git-for-windows、MobaXterm、Kst2.0

b. git clone git://10.5.2.45/sgm.toolkits

c. operate as steps in 《SGM.toolkits 使用手册.pdf》

2. sgm.toolkits help info

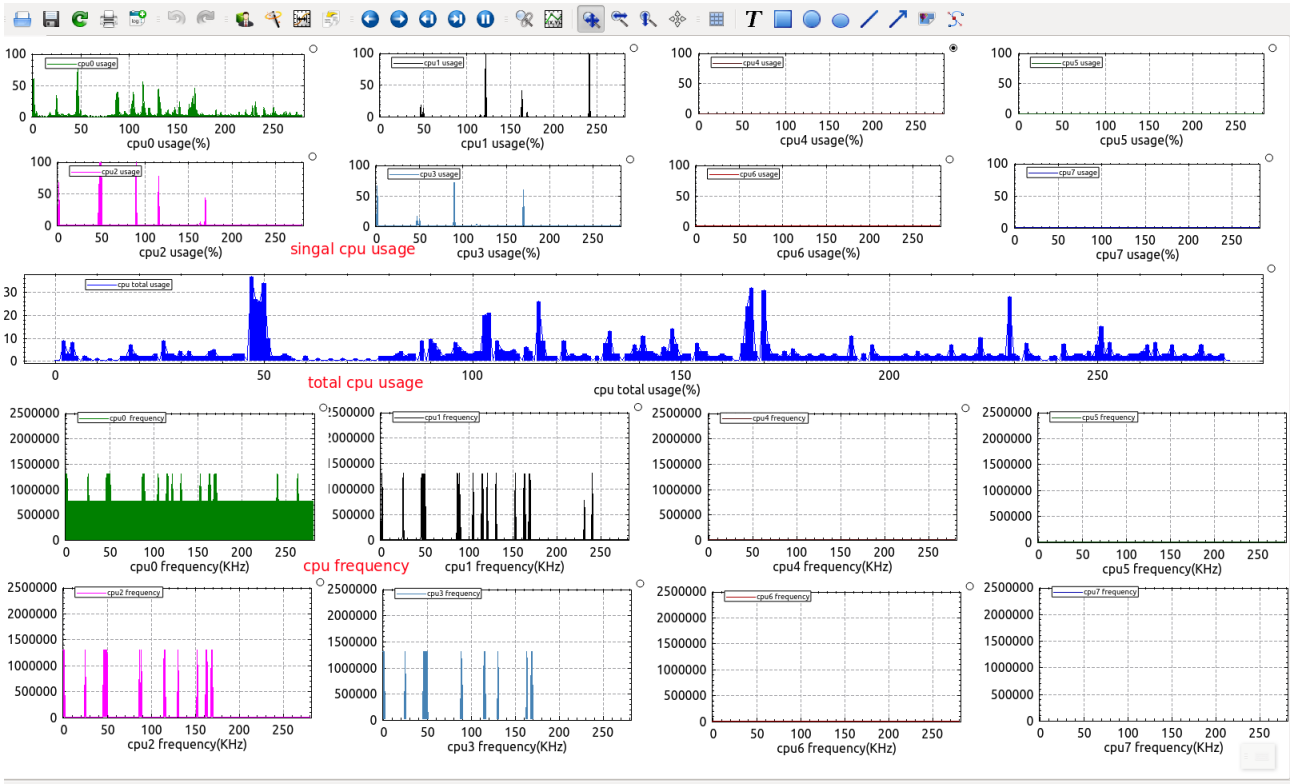
```
SPREADTRUM\yanli.chen@yanlichenubtpc:~/sgm.toolkits$ ./sgm.toolkits
Spreadtrum GUI Monitors
you should chose one of the following charts in ---[ chart name list ]---

sgm.toolkits [-s adb-device-serial-number] [-e event_id:event_id:...]
[-a argv_key:argv_value] [-A argv_key_long:argv_value] [-t interval]
[-c command] [-p pid] [-F /datapath/datasource] [-g /pngfilepath/filename] <chart1-name> <chart2-name> <...>

Example:
sgm.toolkits bus-monitor-bandwidth

---[ chart name list ]---

bus-monitor-bandwidth
cpu-memory-info
cpu-usage-freq
disk-info
perf-pmu-4counter-registers-basic
perf-pmu-4counter-registers-Dcache
perf-pmu-4counter-registers-Dcache-RA-STB-DDRaccess
thermal-info
```

3. how to use sgm.toolkits

pull sgm log to local pc, in sgm folder exec: python analyzer.py, it generage sgm.cpu_memory.log file.

Eg: ./sgm.toolkits -F ~/tmp/sgm/sgm.cpu_memory.log cpu-usage-freq

get cpu usage info and frequency etc.



You can also set parameter -g to make chart-name list info to static picture.

Eg: ./sgm.toolkits -F ~/tmp/sgm/sgm.cpu_memory.log -g 1.png cpu-usage-freq

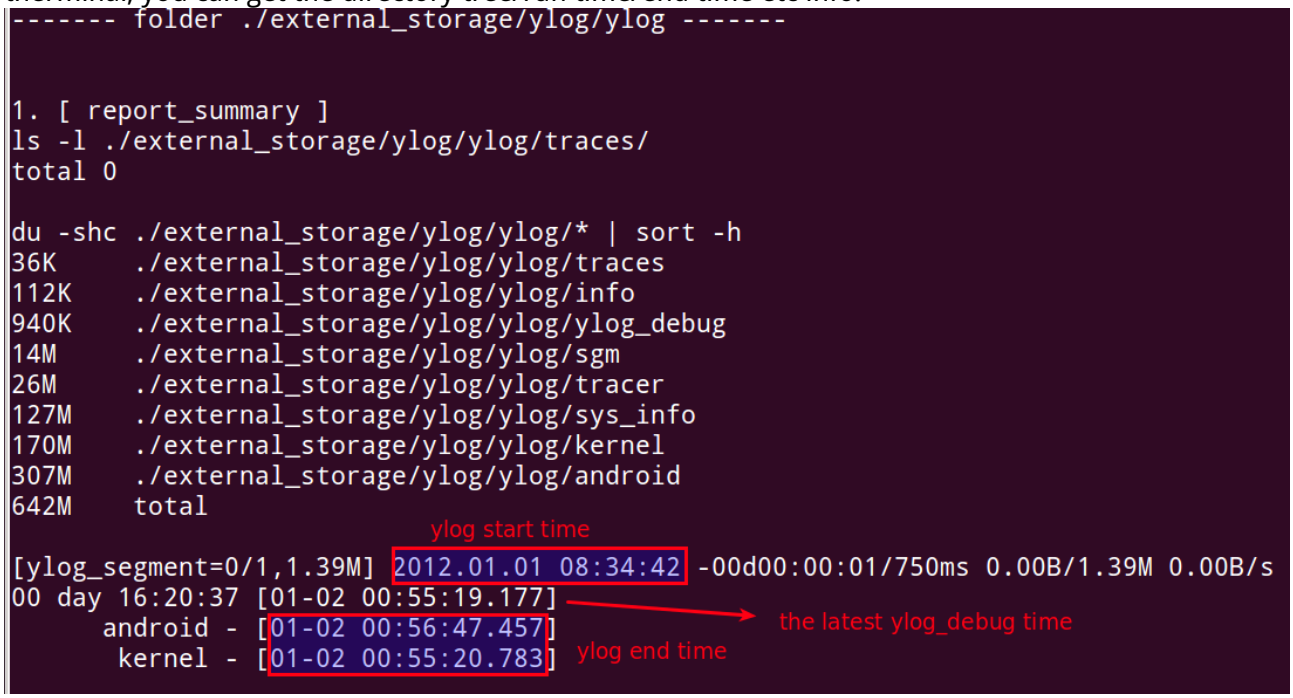
The cpu-usage-freq info is saved as picture named as 1.png

4.2.3 ylog_verify_pc.sh script use help

You can get ylog_verify_pc.sh script frop ylog source code, then copy it to /usr/bin.

cp ylog_verify_pc.sh /usr/bin

Then you can exec ylog_verify_pc.sh directly. In the log folder exec ylog_verify_pc.sh through therminal, you can get the directory tree/run time/end time etc info.



4.3 analytical method of native crash and anr log

1. native crash

a. enter /ylog/ylog/traches, run “python analyzer.py”.

vim traces.log, search “ylog tombstones”.

eg. ylog_tombstones 001 [cat /data/tombstones/tombstone_00] [01-02 02:14:16.785]

time stamp : [01-02 02:14:16.785] presents the time occurred native crash.

b. enter android directory, then vim outline. You can find the segments occurred native crash at that time point.

eg. “01-02 02:14:16.785” is located in the following period, the corresponding log segment is 015.

015 - 2012.01.02 02:12:36 ~ 2012.01.02 02:40:37 [00 00:28:01]

c. obtain log, such as main, system, events, radio, crash contained in this log segment.

command : phthon analyzer.py 015.

generation: crash.log events.log main.log radio.log system.log

d. open corresponding xxx.log, then problems can be analyzed.

2. anr

a. enter /ylog/ylog/traces, run “python analyzer.py”.

vim traces.log, search “ylog traces”.

eg. ylog_traces 001 [cat /data/anr/traces.txt] [01-02 04:07:43.903]

Time stamp : [01-02 04:07:43.903] presents the time occurred anr.

b. enter android directory, vim outline. You can find the segments occurred anr at that time point.

eg. “01-02 04:07:43.903” is located in the following period, the corresponding log segment is 011.

011 - 2012.01.02 04:04:04 ~ 2012.01.02 04:29:18 [00 00:25:14]

c. obtain log, such as main, system, events, radio, crash contained in this log segment.

command: phython analyzer.py 011.

generation: crash.log events.log main.log radio.log system.log

d. open corresponding xxx.log, then problems can be analyzed.

3. log analysis related with system status (system-related log can be collected every 2 minitus)

Method One:

enter “ylog/ylog/sys_info”, run “python anlayzer.py”,

generation: sys_info.log (the total log collected by all the test processing)

Method Two:

a. get time point occurred problems.

b. enter “ylog/ylog/sys_info”, vim outline, you can find the segments occurred anr at that time point.

c. vim the corresponding log segment, analyze the system status about time occurred problems.