Debug 工具

1. **looper 快照**

AMS:

appNotResponding():

if (DEBUG\_AM) {

try {

app.thread.dumpMainLooper();

} catch (Exception ex) {

Slog.e(TAG, "Fail to dump main looper for " + app);

}

}

ActivityThread.java:

public final void dumpMainLooper() {

if (mLooper != null) {

mLooper.dump(new LogPrinter(Log.ERROR, TAG, Log.LOG\_ID\_SYSTEM), "MSG: ");

} else {

Log.d(TAG, "Fail to dump looper because it is null");

}

}

Looper.java:

private MessageHistoryList mMessages = null;

private static class MessageHistory {

int msg\_what;

String msg\_target;

String msg\_callback;

long takenTime = -1;

long startTime = 0;

public MessageHistory(Message \_msg, long start) {

msg\_what = \_msg.what;

msg\_target = \_msg.target == null ? "null" : \_msg.target.toString();

msg\_callback = \_msg.callback == null ? "null" : \_msg.callback.toString();

startTime = start;

}

public void setValue(Message \_msg, long start) {

msg\_what = \_msg.what;

msg\_target = \_msg.target == null ? "null" : \_msg.target.toString();

msg\_callback = \_msg.callback == null ? "null" : \_msg.callback.toString();

takenTime = -1;

startTime = start;

}

}

private static class MessageHistoryList {

private final static int MAX\_NUM\_OF\_MESSAGE = 20;

private final MessageHistory array[] = new MessageHistory[MAX\_NUM\_OF\_MESSAGE];

private int tail = 0;

private int head = 0;

private int size = 0;

public MessageHistory get(int i) {

if (size == 0 || i < 0 || i >= size)

throw new IndexOutOfBoundsException("Invalid index " + i

+ ", size is " + size);

return array[(head + i) % MAX\_NUM\_OF\_MESSAGE];

}

public int size() {

return size;

}

public MessageHistory add(Message msg, long start) {

MessageHistory ret = null;

if (array[tail] == null) {

ret = new MessageHistory(msg, start);

array[tail] = ret;

} else {

ret = array[tail];

ret.setValue(msg, start);

}

tail = (tail + 1) % MAX\_NUM\_OF\_MESSAGE;

if (size < MAX\_NUM\_OF\_MESSAGE) size++;

else head = tail;

return ret;

}

}

public static void prepareMainLooper() {

……

sMainLooper.mMessages = new MessageHistoryList();

}

public static void loop() {

long startTime = SystemClock.uptimeMillis();

MessageHistory headMsg = null;

if (me.mMessages != null) {

synchronized (me.mMessages) {

headMsg = me.mMessages.add(msg, startTime);

}

}

msg.target.dispatchMessage(msg);

if (me.mMessages != null) {

long endTime = SystemClock.uptimeMillis();

long lastTime = endTime - startTime;

synchronized (me.mMessages) {

headMsg.takenTime = lastTime;

}

}

}

public void dump(@NonNull Printer pw, @NonNull String prefix) {

pw.println(prefix + "Dump looper, process: "+Process.myProcessName());

pw.println(prefix + "===== Message History (Most recent to least recent) =====");

if (mMessages != null) {

synchronized (mMessages) {

int numOfSize = mMessages.size();

while (numOfSize > 0) {

numOfSize--;

MessageHistory history = mMessages.get(numOfSize);

if (history.takenTime == -1) {

long lastTime = SystemClock.uptimeMillis() - history.startTime;

pw.println(prefix + history.msg\_target + " " + history.msg\_callback +

": " + history.msg\_what + ", " + "Unfinished(" + lastTime + "ms), start=" + history.startTime);

} else {

pw.println(prefix + history.msg\_target + " " + history.msg\_callback + ": " + history.msg\_what +", "+history.takenTime + "ms, start=" + history.startTime);

}

}

}

}

}

1. **binder\_sample**

1.event log

(1)event log的tag的生成

添加自己的EventLogTags.logtags文件：

Example:

/frameworks/base/packages/SystemUI/src/com/android/systemui/EventLogTags.logtags

格式：

option java\_package com.android.systemui;

# ---------------------------

# PhoneStatusBar.java

# ---------------------------

36000 sysui\_statusbar\_touch (type|1),(x|1),(y|1),(enabled|1)

36001 sysui\_heads\_up\_status (key|3),(visible|1)

添加到mk文件中：

LOCAL\_SRC\_FILES := $(call all-java-files-under, src) \

src/com/android/systemui/EventLogTags.logtags

编译完成后会在out目录中生成EventLogTags.java文件：

格式：

public class EventLogTags {

private EventLogTags() { } // don't instantiate

/\*\* 36000 sysui\_statusbar\_touch (type|1),(x|1),(y|1),(disable1|1),(disable2|1) \*/

public static final int SYSUI\_STATUSBAR\_TOUCH = 36000;

/\*\* 36001 sysui\_heads\_up\_status (key|3),(visible|1) \*/

public static final int SYSUI\_HEADS\_UP\_STATUS = 36001;

public static void writeSysuiStatusbarTouch(int type, int x, int y, int disable1, int disable2) {

android.util.EventLog.writeEvent(SYSUI\_STATUSBAR\_TOUCH, type, x, y, disable1, disable2);

}

public static void writeSysuiHeadsUpStatus(String key, int visible) {

android.util.EventLog.writeEvent(SYSUI\_HEADS\_UP\_STATUS, key, visible);

}

}

将tag加入到/system/etc/event-log-tags文件中：

make event-log-tags 命令生成event-log-tags 文件。

在/build/core/Makefile文件中按照一定的规则通过EventLogTags.logtags文件按将所有模块下的event tag生成TARGET\_OUT)/etc/event-log-tags文件供logd使用。

.PHONY: event-log-tags

所以event-log-tags需要全编才能重新生成然后push到手机中。

**(2)java层和native层对event log的使用**

Java层使用eventlog：

EventLog.writeEvent或者使用EventLogTags.java中生成的方法。

Native层使用event log：

android\_bWriteLog(52004, ……);

**2.Binder\_sample（android 自带的工具） 的实现：**

android\_util\_Binder.cpp

JavaBBinder：：onTransact

android\_os\_BinderProxy\_transact

**3. OOM**

//heap.h

/// [framework] begin, Jim Guo, 2014/12/17, [DEBUG][HPROF] ART: Collecting debug information on OOME occurred.

void DumpHeapInfo() NO\_THREAD\_SAFETY\_ANALYSIS;

/// [framework] end, Jim Guo, 2014/12/17.

/// [framework] begin, Jim Guo, 2014/12/17, [DEBUG][HPROF] ART: Collecting debug information on OOME occurred.

bool is\_hprof\_dumped\_;

/// [framework] end, Jim Guo, 2014/12/17.

//heap.cc

/// [framework] begin, Jim Guo, 2014/12/17, [DEBUG][HPROF] ART: Collecting debug information on OOME occurred.

#include <dirent.h>

#include <fcntl.h>

#include <sys/vfs.h>

#include <sys/stat.h>

#include "hprof/hprof.h"

#include "ScopedFd.h"

/// [framework] end, Jim Guo, 2014/12/17.

Heap::Heap(....... {

/// [framework] begin, Jim Guo, 2014/12/17, [DEBUG][HPROF] ART: Collecting debug information on OOME occurred.

is\_hprof\_dumped\_(false),

/// [framework] end, Jim Guo, 2014/12/17.

}

/// [framework] begin, Jim Guo, 2014/12/17, [DEBUG][HPROF] ART: Collecting debug information on OOME occurred.

struct FileInfo {

std::string name;

uint64\_t size;

size\_t mtime;

};

void Heap::DumpHeapInfo() {

if (IsShippingRom()) {

return;

}

if (is\_hprof\_dumped\_) {

LOG(INFO) << "HeapInfo: Heap dumped before.";

return;

}

is\_hprof\_dumped\_ = true;

const std::string hprof\_dir\_path("/data/misc/hprof");

if (access(hprof\_dir\_path.c\_str(), F\_OK | R\_OK | W\_OK | X\_OK) != 0) {

LOG(ERROR) << StringPrintf("HeapInfo: access '%s' failed, err: %s",

hprof\_dir\_path.c\_str(), strerror(errno));

return;

}

const uint64\_t bytes\_allocated = GetBytesAllocated();

// The minimum number of bytes that the heap dump needed.

// Reserve some storages (capacity\*3) for the others.

// capacity\_ is the maximum number of heap size.

const uint64\_t bytes\_fileMinNeeded = capacity\_ \* 3 + bytes\_allocated;

struct statfs stats;

if (statfs(hprof\_dir\_path.c\_str(), &stats) != 0) {

LOG(ERROR) << StringPrintf("HeapInfo: statfs '%s' failed, err: %s",

hprof\_dir\_path.c\_str(), strerror(errno));

return;

}

// The number of bytes that are free on the file system and available to applications.

// It is equal to StatFs.getAvailableBytes()

uint64\_t bytes\_available = stats.f\_bavail \* stats.f\_bsize;

uint64\_t bytes\_hprof\_total = 0;

if (bytes\_available < bytes\_fileMinNeeded) {

DIR\* dir = opendir(hprof\_dir\_path.c\_str());

if (dir == nullptr) {

LOG(ERROR) << StringPrintf("HeapInfo: opendir '%s' failed, err: %s",

hprof\_dir\_path.c\_str(), strerror(errno));

return;

}

std::vector<struct FileInfo> file\_list;

// collect ".hprof" files to file\_list.

for (struct dirent\* de = readdir(dir); de != nullptr; de = readdir(dir)) {

const char\* last\_dot = strrchr(de->d\_name, '.');

if (last\_dot == nullptr || strcmp(last\_dot, ".hprof") != 0) {

continue;

}

std::string file\_path(hprof\_dir\_path.c\_str());

file\_path.push\_back('/');

file\_path.append(de->d\_name);

struct stat file;

if (stat(file\_path.c\_str(), &file) == 0) {

struct FileInfo item;

item.name = file\_path;

item.size = file.st\_size;

item.mtime = file.st\_mtime;

file\_list.push\_back(item);

bytes\_hprof\_total += file.st\_size;

} else {

LOG(WARNING) << StringPrintf("HeapInfo: stat '%s' failed, err: %s",

file\_path.c\_str(), strerror(errno));

}

}

LOG(INFO) << "HeapInfo: checking storage, bytes\_hprof\_total: " << bytes\_hprof\_total

<< ", bytes\_available: " << bytes\_available

<< ", bytes\_fileMinNeeded: " << bytes\_fileMinNeeded;

// Here to remove exsisted hprof if need.

if (bytes\_hprof\_total + bytes\_available > bytes\_fileMinNeeded) {

// sort by the time of last modification, the oldest at the end.

std::sort(file\_list.begin(), file\_list.end(),

[](const struct FileInfo& a, const struct FileInfo& b) {

return a.mtime > b.mtime;

});

while (bytes\_available < bytes\_fileMinNeeded && !file\_list.empty()) {

const struct FileInfo oldest = file\_list.back();

if (std::remove(oldest.name.c\_str()) == 0) {

bytes\_available += oldest.size;

LOG(INFO) << "HeapInfo: erase file: " << oldest.name;

}

file\_list.pop\_back();

}

}

}

LOG(INFO) << "HeapInfo: bytes\_available: " << bytes\_available

<< ", bytes\_fileMinNeeded: " << bytes\_fileMinNeeded;

if (bytes\_available < bytes\_fileMinNeeded) {

// no storage

return;

}

Thread\* self = Thread::Current();

std::string hprof\_name(StringPrintf("%s/heap-dump-%ld-pid%d.hprof",

hprof\_dir\_path.c\_str(), time(NULL), getpid()));

//self->TransitionFromRunnableToSuspended(kSuspended);

bool isMutatorSharedHeld = false;

if (Locks::mutator\_lock\_->IsSharedHeld(self)) {

Locks::mutator\_lock\_->SharedUnlock(self);

isMutatorSharedHeld = true;

}

hprof::DumpHeap(hprof\_name.c\_str(), -1, false);

if (isMutatorSharedHeld) {

Locks::mutator\_lock\_->SharedLock(self);

}

//self->TransitionFromSuspendedToRunnable();

// Change hprof file permission for TELL HTC app.

ScopedFd fd(open(hprof\_name.c\_str(), O\_RDONLY));

if (fd.get() == -1) {

LOG(ERROR) << StringPrintf("HeapInfo: failed to open '%s', err: %s",

hprof\_name.c\_str(), strerror(errno));

return;

}

if (fchmod(fd.get(), 0644) != 0) {

LOG(WARNING) << StringPrintf("HeapInfo: change '%s' permission failed, err: %s",

hprof\_name.c\_str(), strerror(errno));

}

}

void Heap::ThrowOutOfMemoryError(Thread\* self, size\_t byte\_count, AllocatorType allocator\_type) {

std::ostringstream oss;

size\_t total\_bytes\_free = GetFreeMemory();

oss << "Failed to allocate a " << byte\_count << " byte allocation with " << total\_bytes\_free

<< " free bytes and " << PrettySize(GetFreeMemoryUntilOOME()) << " until OOM";

// If the allocation failed due to fragmentation, print out the largest continuous allocation.

if (total\_bytes\_free >= byte\_count) {

space::AllocSpace\* space = nullptr;

if (allocator\_type == kAllocatorTypeNonMoving) {

space = non\_moving\_space\_;

} else if (allocator\_type == kAllocatorTypeRosAlloc ||

allocator\_type == kAllocatorTypeDlMalloc) {

space = main\_space\_;

} else if (allocator\_type == kAllocatorTypeBumpPointer ||

allocator\_type == kAllocatorTypeTLAB) {

space = bump\_pointer\_space\_;

} else if (allocator\_type == kAllocatorTypeRegion ||

allocator\_type == kAllocatorTypeRegionTLAB) {

space = region\_space\_;

}

if (space != nullptr) {

space->LogFragmentationAllocFailure(oss, byte\_count);

}

}

/// [framework] begin, Jim Guo, 2014/12/17, [DEBUG][HPROF] ART: Collecting debug information on OOME occurred.

LOG(ERROR) << "Out of memory: Heap Size=" << PrettySize(GetMaxMemory())

<< ", Allocated=" << PrettySize(GetBytesAllocated())

<< ", Capacity=" << PrettySize(capacity\_);

/// [framework] end, Jim Guo, 2014/12/17.

self->ThrowOutOfMemoryError(oss.str().c\_str());

}

//thread.cc

void Thread::ThrowOutOfMemoryError(const char\* msg) {

LOG(WARNING) << StringPrintf("Throwing OutOfMemoryError \"%s\"%s",

msg, (tls32\_.throwing\_OutOfMemoryError ? " (recursive case)" : ""));

if (!tls32\_.throwing\_OutOfMemoryError) {

tls32\_.throwing\_OutOfMemoryError = true;

/// [framework] begin, Jim Guo, 2014/12/17, [DEBUG][HPROF] ART: Collecting debug information on OOME occurred.

Runtime::Current()->GetHeap()->DumpHeapInfo();

/// [framework] end, Jim Guo, 2014/12/17.

ThrowNewException("Ljava/lang/OutOfMemoryError;", msg);

tls32\_.throwing\_OutOfMemoryError = false;

} else {

Dump(LOG(WARNING)); // The pre-allocated OOME has no stack, so help out and log one.

SetException(Runtime::Current()->GetPreAllocatedOutOfMemoryError());

}

}

**4. systrace**

**5. dumpsys window lastscreenshot**