Discover System Facilities inside Your Android Phone

Jim Huang (黃敬群)

Developer, 0xlab

jserv@0xlab.org

Feb 26, 2012 / Study-Area

Rights to copy

http://0xlab.org/

contact@0xlab.org

creative commons
COMMONS DEED

Attribution – ShareAlike 3.0

You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Corrections, suggestions, contributions and translations are welcome!

Latest update: Feb 26, 2012

Under the following conditions

BY: **Attribution**. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: http://creativecommons.org/licenses/by-sa/3.0/legalcode

Goals of This Presentation

- Pick up one Android phone and discover its internals
- Learn how to select the "weapons" to fight with Android system facilities
- Skipping Java parts, we focus on the native area: dynamic linking, processes, debugger, memory layout, IPC, and interactions with frameworks.
- It is not comprehensive to familarize Android. The goal is to utilize Android platforms, which are the popular and powerful development devices to us.

Agenda

Part I

(0) Environment Setup

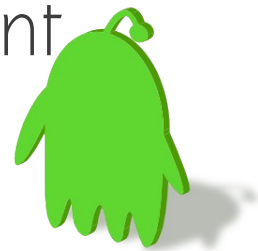
(1) Hello World!

Part II

(2) Case: Angry Birds

(3) Case: Binder driver

(4) Case: Power Management



Environment Setup

Reference Hardware and Host Configurations

- Android Phone: Nexus S
  - http://www.google.com/phone/detail/nexus-s
  - Install CyanogenMod (**CM9**; 4.0)

    http://www.cyanogenmod.com/
- Host: Lenovo x200
  - Ubuntu Linux 11.10+
- Toolchain: Sourcery CodeBench Lite
  - GNU/Linux Release 2011.09-70
  - http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/
- AOSP/CM9 source code: 4.0.3
- Android NDK r5c/r7b

Build CM9 from source

- Follow the instructions in Wiki
  - http://wiki.cyanogenmod.com/wiki/Building_from_source
- Follow the instructrions in AOSP
  - http://source.android.com/source/downloading.html
  - http://source.android.com/source/building.html
  - http://source.android.com/source/building-devices.html
- Obtaining proprietary binaries
  - Starting with ICS, AOSP can't be used from pure source code only, and requires additional hardware-related proprietary libraries to run, specifically for hardware graphics acceleration.
  - Binaries for Nexus Phones and Flagship Devices

    http://code.google.com/android/nexus/drivers.html
- Confirm the exact match
  - between AOSP version and proprietary packages

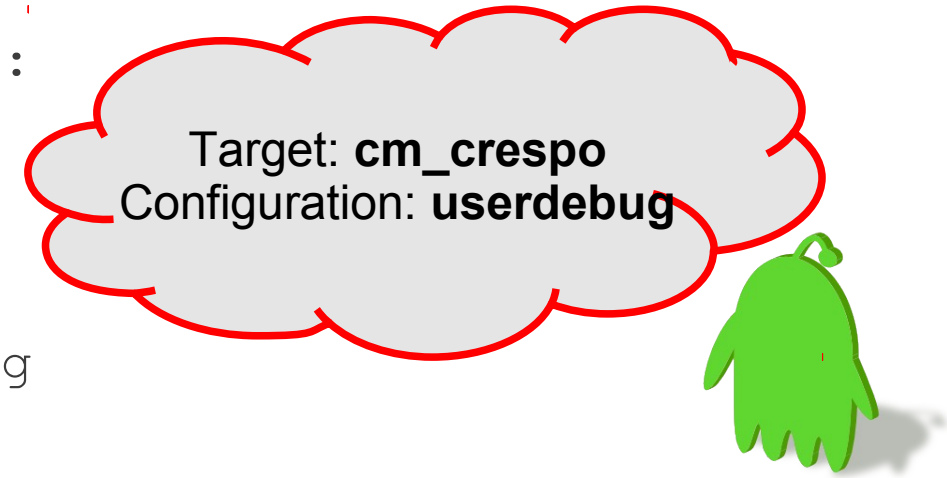
Steps to Build CM (1)

- cyanogen-ics$ **source build/envsetup.sh**

```
including device/moto/stingray/vendorsetup.sh
including device/moto/wingray/vendorsetup.sh
including device/samsung/maguro/vendorsetup.sh
including device/samsung/toro/vendorsetup.sh
including device/ti/panda/vendorsetup.sh
including vendor/cm/vendorsetup.sh
including sdk/bash_completion/adb.bash
```

- cyanogen-ics$ **lunch**

```
You're building on Linux
Lunch menu... pick a combo:
     1. full-eng
…
     8. full_panda-eng
     9. cm_crespo-userdebug
```



Steps to Build CM (2)

- Which would you like? [full-eng] **9**

```
==========================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.0.3
TARGET_PRODUCT=cm_crespo
TARGET_BUILD_VARIANT=userdebug
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a-neon
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=MR1
==========================================
```

hello.c

```
#include <stdio.h>
int main()
{
    printf("Hello World!\n");
    return 0;
}
```

Assume file **`hello.c`** is placed in directory 'test' under AOSP top-level source tree.

Android.mk

```
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)
LOCAL_MODULE_TAGS := optional
LOCAL_MODULE := hello
LOCAL_SRC_FILES := hello.c
include $(BUILD_EXECUTABLE)
```

Assume file **`Android.mk`** is placed in directory 'test' under AOSP top-level source tree.

Hello World!

(use Android toolchain and build rules)

- **`cd tests`**

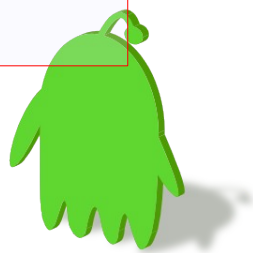
Trigger Android build system to generate 'hello'

- **`mm -B`**

```
No private recovery resources for TARGET_DEVICE crespo
make: Entering directory `/home/jserv/cyanogen-ics'
target thumb C: hello <= tests/hello.c
target Executable: hello
(out/target/product/crespo/obj/EXECUTABLES/hello_intermediates/LINKED/hello)
target Symbolic: hello
(out/target/product/crespo/symbols/system/bin/hello)
target Strip: hello
(out/target/product/crespo/obj/EXECUTABLES/hello_intermediates/hello)
Install: out/target/product/crespo/system/bin/hello
```

- **`adb push \`**
  `../out/target/product/crespo/system/bin/hello \`
  `/data/local/`

For Android 'user' build, only directory `/data/local` is writable and executable.



Hello World!

(deploy to Android device and execute)

- **adb push **
  **../out/target/product/crespo/system/bin/hello **
  /data/local/
- **adb shell /data/local/hello**

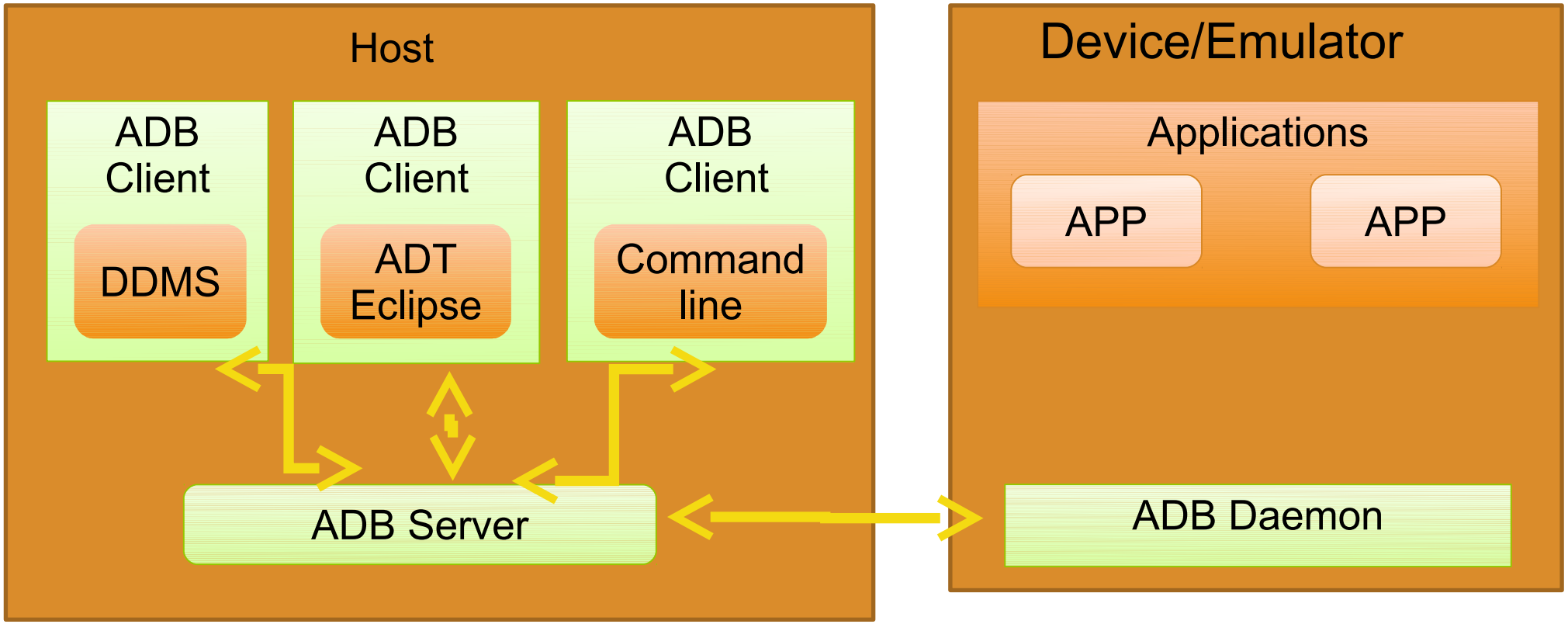
```
Hello World!
```

- **arm-eabi-readelf -a **
  ../out/target/product/crespo/system/bin/hello

```
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
 ...
  Type:                              EXEC (Executable file)
  Machine:                           ARM
...
```

ADB: Android Debug Bridge

- ADB includes three compnenent: Server, Client and Daemon.



Details:
http://developer.android.com/guide/developing/tools/adb.html

Hello World!

(use strace to trace the system calls during execution)

- **adb shell strace /data/local/hello**

```
execve("/data/local/hello", ["/data/local/hello"], [/*
13 vars */]) = 0
...
stat64("/system/lib/libc.so", {st_mode=S_IFREG|0644,
st_size=282248, ...}) = 0
open("/system/lib/libc.so", O_RDONLY|O_LARGEFILE) = 3
ioctl(1, TCGETS or SNDCTL_TMR_TIMEBASE, {B38400 opost
isig icanon echo ...}) = 0
write(1, "Hello World!\n", 13Hello World!
)           = 13
...
```

The above log shows dynamic library **/system/lib/libc.so** is open and manipulated.

Hello World!

(Remote Debugging)

Map host port 12345 to device port 12345

- **`adb forward tcp:12345 tcp:12345`**
- **`adb shell gdbserver :12345 /data/local/hello`**

```
Process /data/local/hello created; pid =
18696
Listening on port 12345
```

Run gdbserver on device to listen port 12345

```
(execute the following in another terminal)
```

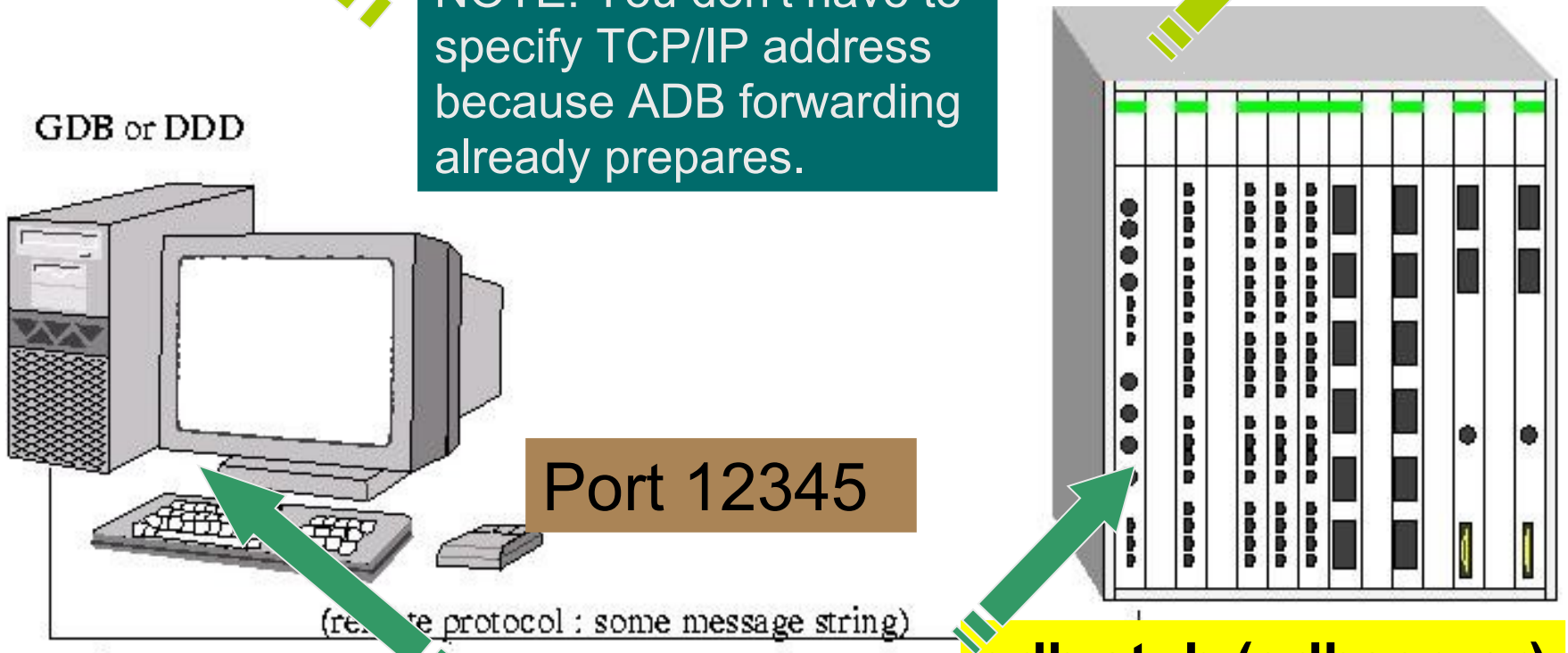
- **`arm-eabi-gdb \`**
  `out/target/product/crespo/symbols/system/bin/hello`

```
(gdb) target remote :12345
Remote debugging using :12345
```

```
arm-eabi-gdb \
    out/target/product/crespo/symbols/system/bin/hello
(gdb) target remote :12345
```

```
adb shell gdbserver :12345 /data/local/hello
Process /data/local/hello created; pid = 18696
Listening on port 12345
```

NOTE: You don't have to specify TCP/IP address because ADB forwarding already prepares.



Linux host running GDB

gdb stub (gdbserver)

Android phone

Hello World!

'hello' is loaded (process created, PID=18696)
But not executed.

- `adb shell cat /proc/18696/maps`

```
00008000-00009000 r-xp 00000000 b3:02 8959       /data/local/hello
00009000-0000a000 rwxp 00001000 b3:02 8959       /data/local/hello
b0001000-b0009000 r-xp 00001000 b3:01 128        /system/bin/linker
b0009000-b000a000 rwxp 00009000 b3:01 128        /system/bin/linker
b000a000-b0015000 rwxp 00000000 00:00 0
beb07000-beb28000 rw-p 00000000 00:00 0          [stack]
ffff0000-ffff1000 r-xp 00000000 00:00 0          [vectors]
```

- (gdb) `b main`
  (gdb) `c`
  Continuing.

Continue exec

Process map changes after execuation.
This means magic in dynamic linking!

- `adb shell cat /proc/18696/maps`

```
00008000-00009000 r-xp 00000000 b3:02 8959       /data/local/hello
00009000-0000a000 rwxp 00001000 b3:02 8959       /data/local/hello
40061000-40062000 r-xp 00000000 00:00 0
40079000-40081000 r-xs 00000000 00:0b 392        /dev/__properties__ (deleted)
40087000-400c9000 r-xp 00000000 b3:01 548        /system/lib/libc.so
400c9000-400cc000 rwxp 00042000 b3:01 548        /system/lib/libc.so
400cc000-400d7000 rwxp 00000000 00:00 0
400d7000-400ec000 r-xp 00000000 b3:01 597        /system/lib/libm.so
400ec000-400ed000 rwxp 00015000 b3:01 597        /system/lib/libm.so
40101000-40102000 r-xp 00000000 b3:01 644        /system/lib/libstdc++.so
40102000-40103000 rwxp 00001000 b3:01 644        /system/lib/libstdc++.so
b0001000-b0009000 r-xp 00001000 b3:01 128        /system/bin/linker
b0009000-b000a000 rwxp 00009000 b3:01 128        /system/bin/linker
b000a000-b0015000 rwxp 00000000 00:00 0
beb07000-beb28000 rw-p 00000000 00:00 0          [stack]
ffff0000-ffff1000 r-xp 00000000 00:00 0          [vectors]
```

Everything starts from
Hello World!

Execution flow of Hello World! (GNU/Linux)

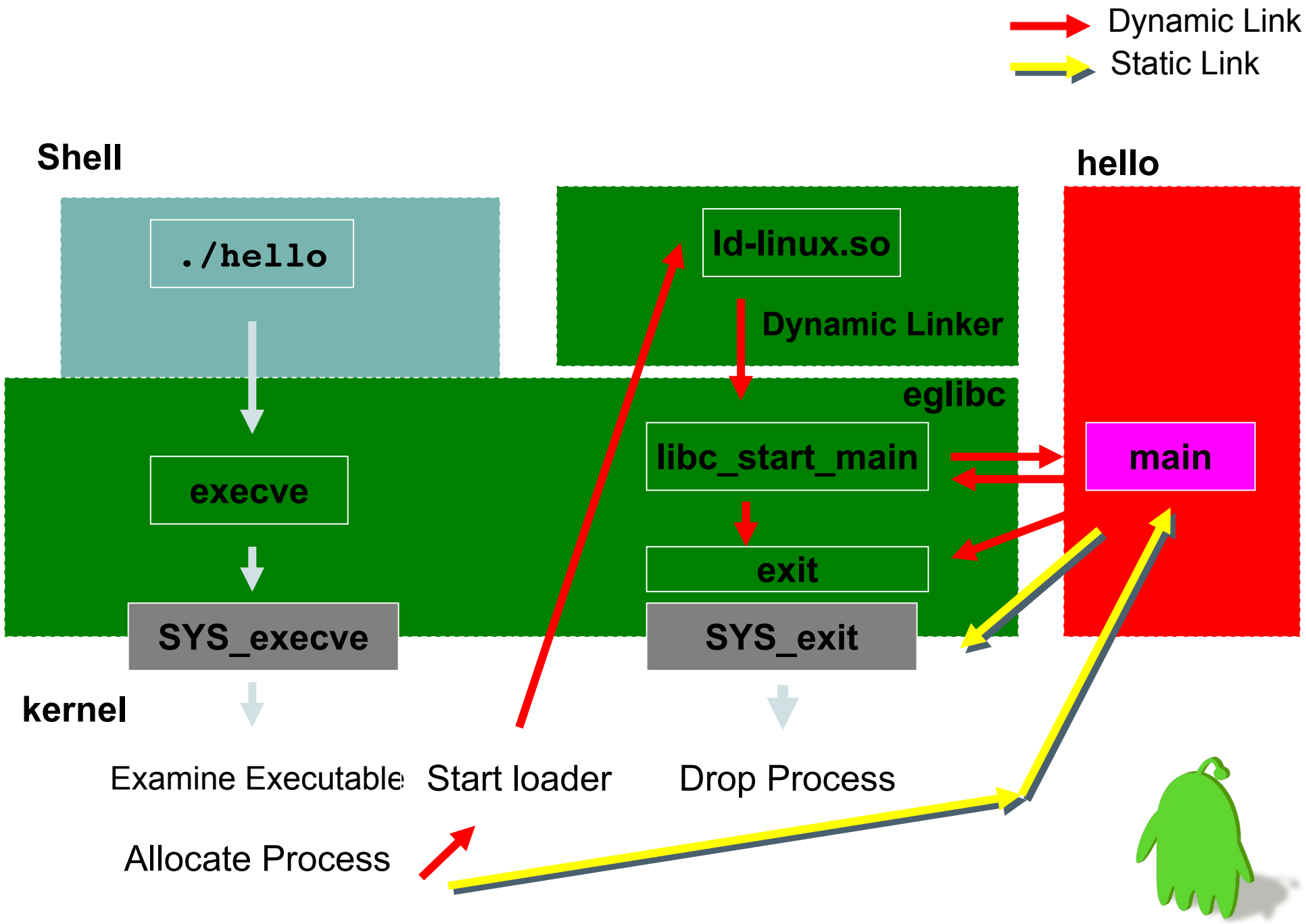


Figure out ELF information (glibc/x86)

- **gcc -o hello hello.c**
- **readelf -a hello**

```
...
  Type:                              EXEC (Executable file)
  Machine:                           Intel 80386
...
Relocation section '.rel.plt' at offset 0x298 contains 3 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
0804a000  00000107 R_386_JUMP_SLOT   00000000   puts
...
0804a008  00000307 R_386_JUMP_SLOT   00000000   __libc_start_main

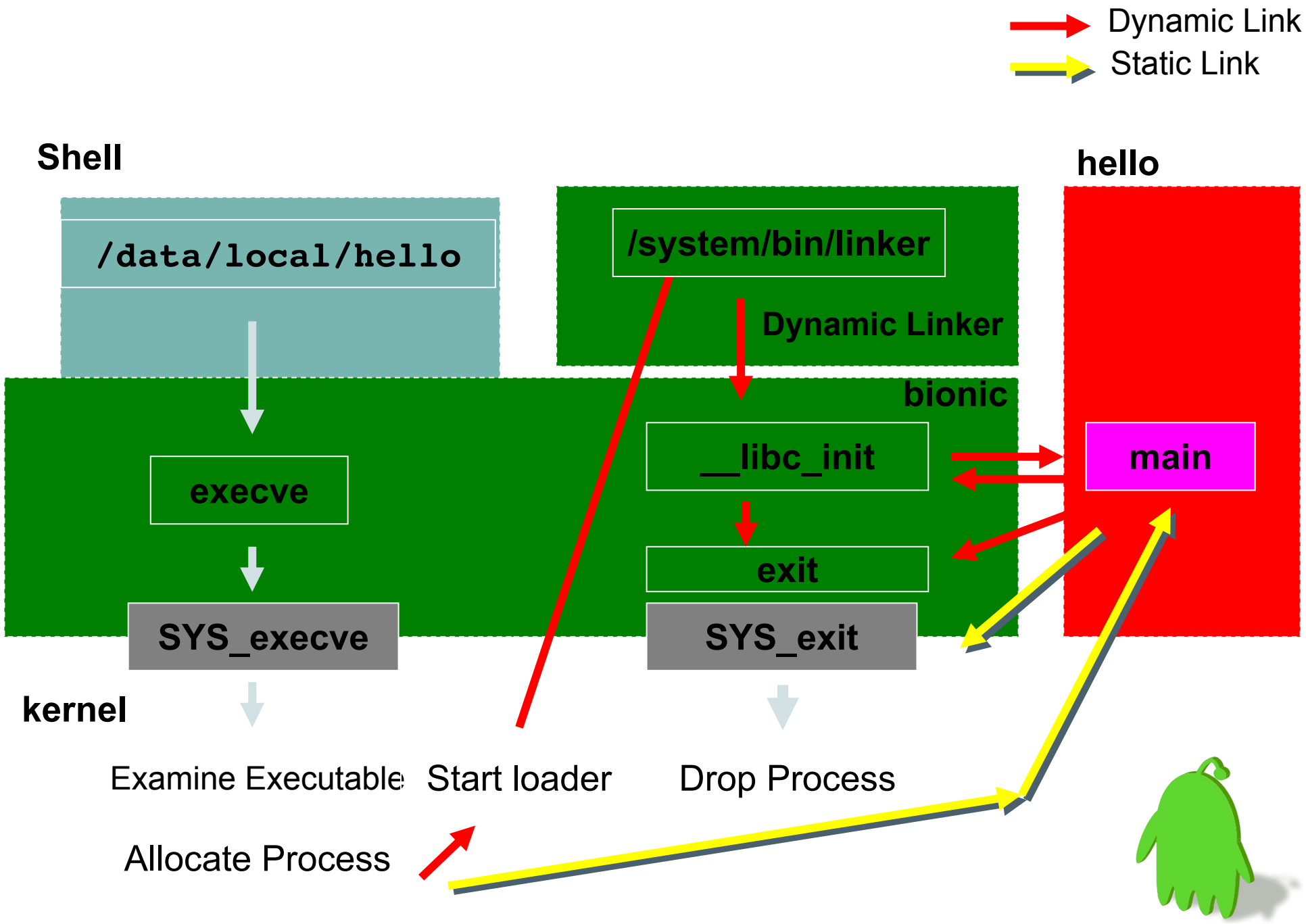
Symbol table '.dynsym' contains 5 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
...
     3: 00000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.0
….
Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
...
  INTERP         0x000154 0x08048154 0x08048154 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
```

Figure out ELF information (Android/ARM)

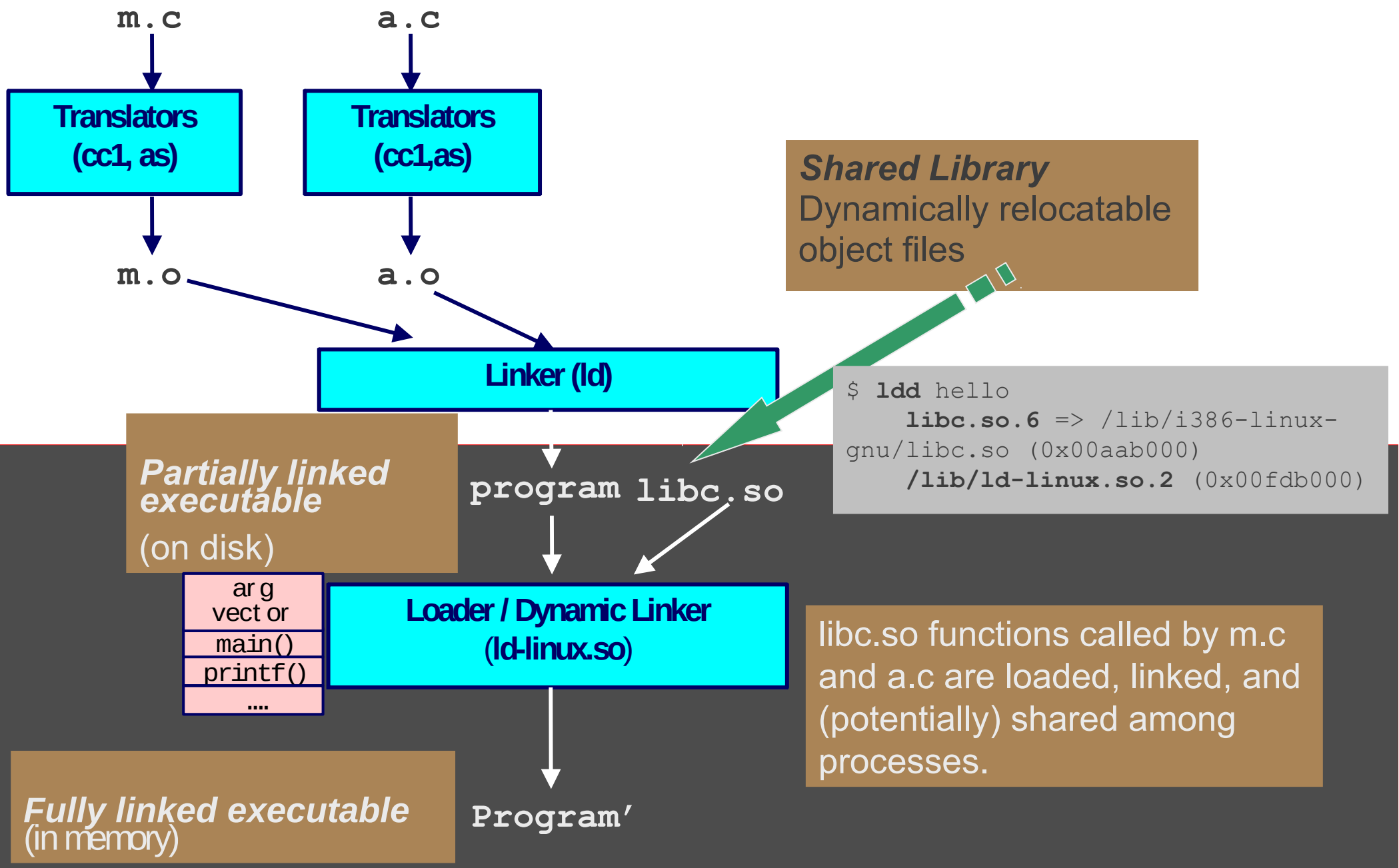
- **arm-eabi-readelf -a **
  ../out/target/product/crespo/system/bin/hello

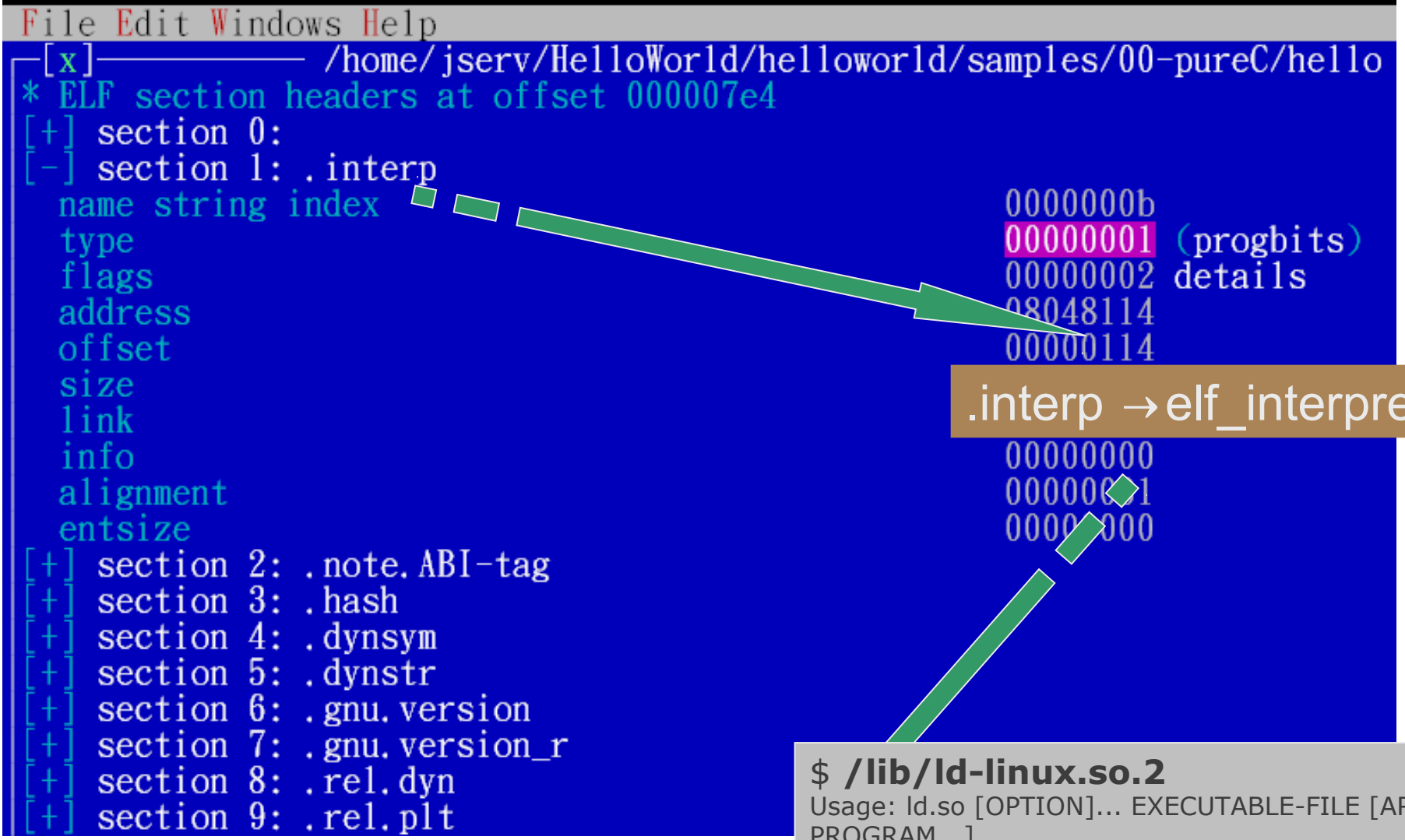
```
...
  Machine:                           ARM
...
Relocation section '.rel.plt' at offset 0x3c0 contains 2 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
000090d4  00000216 R_ARM_JUMP_SLOT   00000000   __libc_init
...
Symbol table '.dynsym' contains 18 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
...
     2: 00000000     0 FUNC    GLOBAL DEFAULT  UND __libc_init
...
Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz
...
  INTERP         0x000114 0x00008114 0x00008114 0x00013
      [Requesting program interpreter: /system/bin/linker]
```

Execution flow of Hello World! (Android/ARM)



ELF Image


m.c
a.c
Translators (cc1, as)
Translators (cc1,as)
m.o
a.o
Linker (ld)
Shared Library
Dynamically relocatable object files
$ ldd hello
libc.so.6 => /lib/i386-linux-gnu/libc.so (0x00aab000)
/lib/ld-linux.so.2 (0x00fdb000)
Partially linked executable
(on disk)
program
libc.so
arg vector
main()
printf()
....
Loader / Dynamic Linker (ld-linux.so)
libc.so functions called by m.c and a.c are loaded, linked, and (potentially) shared among processes.
Fully linked executable
(in memory)
Program'


File Edit Windows Help
[x] /home/jserv/HelloWorld/helloworld/samples/00-pureC/hello
* ELF section headers at offset 000007e4
[+] section 0:
[-] section 1: .interp
name string index 0000000b
type 00000001 (progbits)
flags 00000002 details
address 08048114
offset 00000114
size
link
info
alignment
entsize
[+] section 2: .note.ABI-tag
[+] section 3: .hash
[+] section 4: .dynsym
[+] section 5: .dynstr
[+] section 6: .gnu.version
[+] section 7: .gnu.version_r
[+] section 8: .rel.dyn
[+] section 9: .rel.plt
.interp →elf_interpreter
$ /lib/ld-linux.so.2
Usage: ld.so [OPTION]... EXECUTABLE-FILE [ARGS-FOR-PROGRAM...]
You have invoked `ld.so', the helper program for shared library executables. This program usually lives in the file `/lib/ld.so', and special directives in executable files using ELF shared libraries tell the system's program loader to load the helper program from this file. This helper program loads the shared libraries needed by the program executable, prepares the program to run, and runs it.
$ objdump -s -j .interp hello
hello: file format elf32-i38
Contents of section .interp:
8048114 2f6c6962 2f6c642d 6c696e75 782e736f /lib/ld-linux.so
8048124 2e3200 .2.

ELF Interpreter

- **objdump -s -j .interp** hello-x86

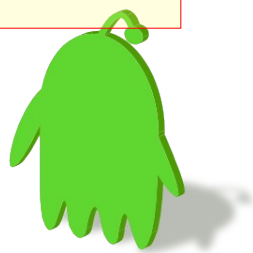
```
hello:     file format elf32-i386

Contents of section .interp:
 8048154 2f6c6962 2f6c642d 6c696e75 782e736f  /lib/ld-linux.so
 8048164 2e3200                               .2.
```

- **arm-eabi-objdump -s -j .interp** \
../out/target/product/crespo/system/bin/hello

```
../out/target/product/crespo/system/bin/hello:     file format elf32-littlearm

Contents of section .interp:
 8114 2f737973 74656d2f 62696e2f 6c696e6b  /system/bin/link
 8124 657200                               er.
```



```
$ /lib/ld-linux.so.2
Usage: ld.so [OPTION]... EXECUTABLE-FILE [ARGS-FOR-PROGRAM...]
```

```
$ file /lib/ld-linux.so.2
/lib/ld-linux.so.2: symbolic link to `i386-linux-gnu/ld-2.13.so'
$ file /lib/i386-linux-gnu/ld-2.13.so
/lib/i386-linux-gnu/ld-2.13.so: ELF 32-bit LSB shared object, Intel
80386, version 1 (SYSV), dynamically linked,
BuildID[sha1]=0x41de5107934017489907fa244bf835ce98feddc1, stripped
$ objdump -f /lib/ld-linux.so.2
/lib/ld-linux.so.2:     file format elf32-i386
architecture: i386, flags 0x00000150:
HAS_SYMS, DYNAMIC, D_PAGED
start address 0x000010e0
```

glibc
sysdeps/generic/dl-sysdep.c
elf/rtld.c

```
$ LD_DEBUG=help /lib/ld-2.13.so
Valid options for the LD_DEBUG environment variable
are:

  libs          display library search paths
  reloc     display relocation processing
  files         display progress for input file
  symbols   display symbol table processing
  bindings    display information about symbol binding
  versions    display version dependencies
  all           all previous options combined
  statistics    display relocation statistics
  unused    determined unused DSOs
  help          display this help message and
```

Hint
Try LD_DEBUG=XXX ./hello

Hint
LD_TRACE_PRELINKING=1 ./hello

Hello World!

(use plain Linux toolchain and try to execute in Android)

- **`export PATH=/usr/local/csl/arm-2011.09/bin:$PATH`**
- **`arm-none-linux-gnueabi-gcc -o hello hello.c`**
- **`adb push hello /data/local && \`**
  `adb shell /data/local/hello`

  `/system/bin/sh: /data/local/hello: No such file or directory`

  Why?!
- **`arm-eabi-readelf -a hello | grep interpreter`**

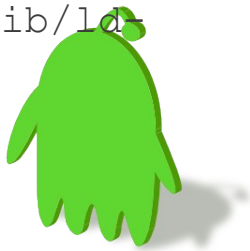
  `[Requesting program interpreter: /lib/ld-linux.so.3]`
- **`find /usr/local/csl/arm-2011.09/ \`**
  `-name ld-linux.so.3`

  `/usr/local/csl/arm-2011.09/arm-none-linux-gnueabi/libc/lib/ld-linux.so.3`

  `/usr/local/csl/arm-2011.09/arm-none-linux-gnueabi/libc/armv4t/lib/ld-linux.so.3`

  `/usr/local/csl/arm-2011.09/arm-none-linux-gnueabi/libc/thumb2/lib/ld-linux.so.3`

  Directory /lib is empty in Android.
- **`adb push /usr/local/csl/arm-2011.09/arm-none-linux-gnueabi/libc/lib/ld-linux.so.3 /data/local/`**



Hello World!

(use GNU/Linux toolchain)

Specify ELF interpreter to /data/local/ld-linux.so.3

- **arm-none-linux-gnueabi-gcc -o hello hello.c **
  -Wl,-dynamic-linker,/data/local/ld-linux.so.3
- **adb push hello /data/local && **
  adb shell /data/local/hello

  ```
  Inconsistency detected by ld.so: dl-deps.c: 622:
  _dl_map_object_deps: Assertion `nlist > 1' failed!
  ```

Why?!

- **adb shell strace /data/local/hello**

  ```
  open("/vendor/lib/tls/v7l/neon/vfp/libgcc_s.so.1", O_RDONLY)
  = -1 ENOENT (No such file or directory)

  ...

  open("/usr/lib/libgcc_s.so.1", O_RDONLY) = -1 ENOENT (No such
  file or directory)
  ```

- **arm-none-linux-gnueabi-gcc -o hello hello.c **
  **-Wl,-dynamic-linker,/data/local/ld-linux.so.3 **
  -static-libgcc

Eliminate libgcc_s.so.1 dependency by static linking

Hello World!

(use GNU/Linux toolchain)

- **arm-none-linux-gnueabi-gcc -o hello hello.c **
  **-Wl,-dynamic-linker,/data/local/ld-linux.so.3 **
  -static-libgcc
- **adb push hello /data/local && adb shell /data/local/hello**

  Inconsistency detected by ld.so: dl-deps.c: 622: _dl_map_object_deps: Assertion `nlist > 1' failed!
- **adb shell strace /data/local/hello**

  open("/usr/lib/libc.so.6", O_RDONLY) = -1 ENOENT (No such file or directory)
- **find /usr/local/csl/arm-2011.09 -name libc.so.6**

  /usr/local/csl/arm-2011.09/arm-none-linux-gnueabi/libc/lib/libc.so.6

  /usr/local/csl/arm-2011.09/arm-none-linux-gnueabi/libc/armv4t/lib/libc.so.6

  /usr/local/csl/arm-2011.09/arm-none-linux-gnueabi/libc/thumb2/lib/libc.so.6

Let's provide libc on Androi device.

- **adb push /usr/local/csl/arm-2011.09/arm-none-linux-gnueabi/libc/lib/libc.so.6 /data/local/**
- **arm-none-linux-gnueabi-gcc -o hello hello.c **
  **-Wl,-dynamic-linker,/data/local/ld-linux.so.3 **
  -static-libgcc -Wl,--rpath -Wl,/data/local

rpath: runtime library search path

- **adb push hello /data/local && adb shell /data/local/hello**

  Hello World!

Options when compiling and linking

- There is no libgcc runtime in Android target device. Build system looks for libgcc.a provided by toolchain and link it statically.
- Two flags have to be passed to linker
  - dynamic-linker
  - rpath
- Source file: `build/core/combo/TARGET_linux-arm.mk`

```
define transform-o-to-executable-inner
$(hide) $(PRIVATE_CXX) -nostdlib -Bdynamic -Wl,-T,$
(BUILD_SYSTEM)/armelf.x \
        -Wl,-dynamic-linker,/system/bin/linker \
        -Wl,--gc-sections \
        -Wl,-z,nocopyreloc \
        -o $@ \
        $(TARGET_GLOBAL_LD_DIRS) \
        -Wl,-rpath-link=$(TARGET_OUT_INTERMEDIATE_LIBRARIES) \
```

hello-crash.c

```
#include <stdio.h>
void hello() { printf("Hello World!\n"); }
void (*ptr)();
int main()
{
    ptr = &hello;
    (*ptr)();
    ptr = NULL;
    (*ptr)();
    return 0;
}
```

NOTE: The reason why we would take a memory violation program is that it can help us to trace the internals when crashing.

Assume file **`hello-crash.c`** is placed in directory 'test' under AOSP top-level source tree.

Hello Crash!

- `mm -B`
- ```
  adb push \
    ../out/target/product/crespo/system/bin/hello-crash \
    /data/local
  ```
- `adb logcat -c`
- `adb shell /data/local/hello-crash`

  ```
  Hello World!
  Segmentation fault
  ```
- `adb logcat`


  ```

Magic in Android debuggerd

- **adb logcat**

```
--------- beginning of /dev/log/main
F/libc    (14127): Fatal signal 11 (SIGSEGV) at 0x00000000 (code=1)
...
I/DEBUG   ( 8044): pid: 14127, tid: 14127  >>> /data/local/hello-crash <<<
I/DEBUG   ( 8044): signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 00000000
...
I/DEBUG   ( 8044):         #00  pc 00000000
I/DEBUG   ( 8044):         #01  pc 00008440  /data/local/hello-crash
I/DEBUG   ( 8044):         #02  pc 00016330  /system/lib/libc.so (__libc_init)
I/DEBUG   ( 8044):
I/DEBUG   ( 8044): code around pc:
I/DEBUG   ( 8044): 00000000 ffffffff ffffffff                                ...
I/DEBUG   ( 8044): 00000010 ffffffff ffffffff                                ...
I/DEBUG   ( 8044): 00000020 ffffffff ffffffff                                ...
I/DEBUG   ( 8044): 00000030 ffffffff ffffffff ffffffff ffffffff  ................
I/DEBUG   ( 8044): 00000040 ffffffff ffffffff ffffffff ffffffff  ................
```

`__libc_init` was mentioned in previous diagram

Call stack when executing hello-crash

Debuggerd

```
#00  pc 00000000
#01  pc 00008440  /data/local/hello-crash
#02  pc 00016330  /system/lib/libc.so (__libc_init)
```

- **arm-eabi-addr2line -e **
  **../out/target/product/crespo/symbols/system/bin/hello-rash **
  00008440

  /home/jserv/cyanogen-ics/tests/hello-crash.c:6

```
Line 01  #include <stdio.h>
Line 02  void hello() { printf("Hello World!\n"); }
Line 03  void (*ptr)();
Line 04  int main()
Line 05  {
Line 06      ptr = &hello;
Line 07      (*ptr)();
Line 08      ptr = NULL;
Line 09      (*ptr)();
Line 10      return 0;
Line 11  }
```

Debuggerd

```
#00  pc 00000000
#01  pc 00008440  /data/local/hello-crash
#02  pc 00016330  /system/lib/libc.so (__libc_init)
```

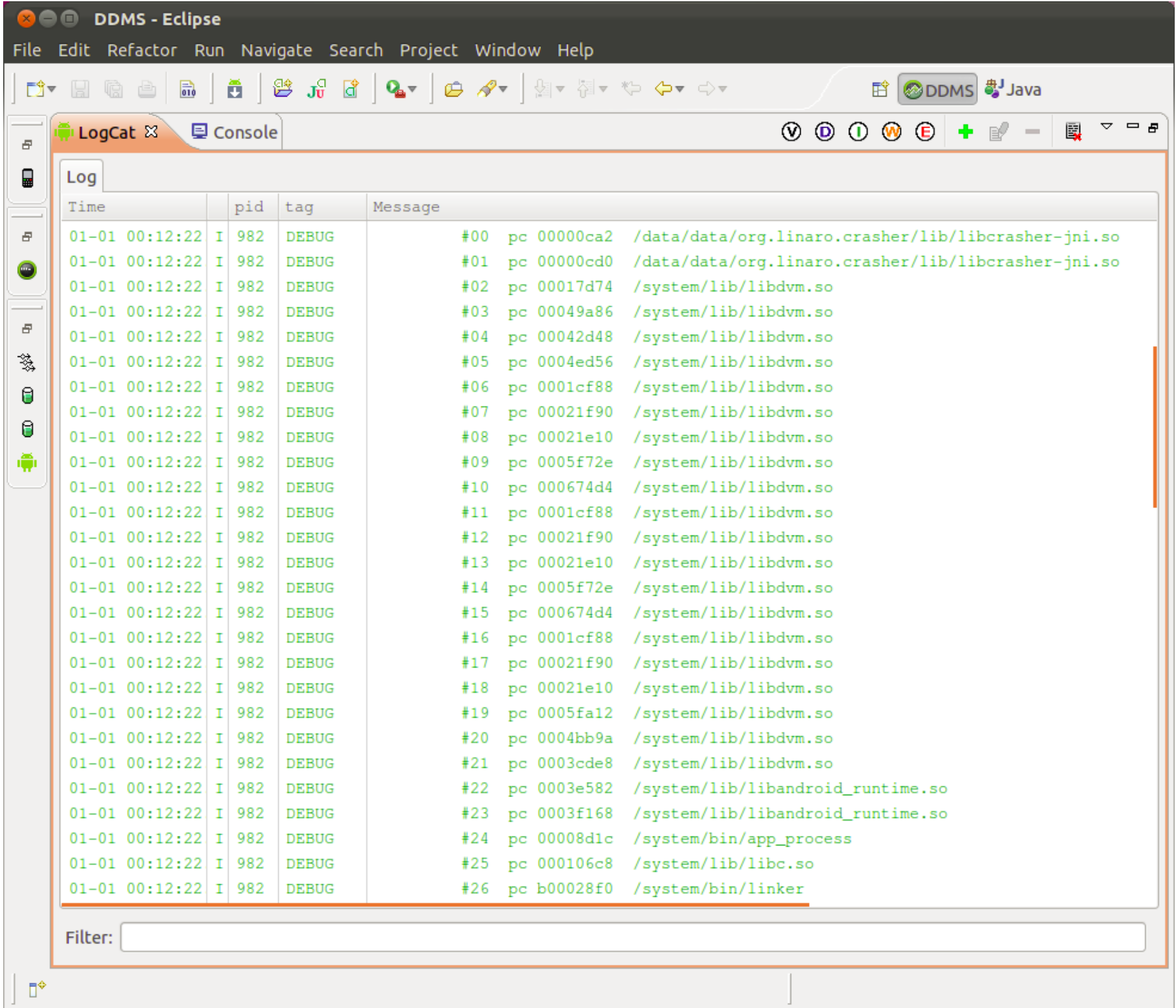
- **addr2line -e **
  **../out/target/product/crespo/symbols/system/lib/libc.so **
  00016330

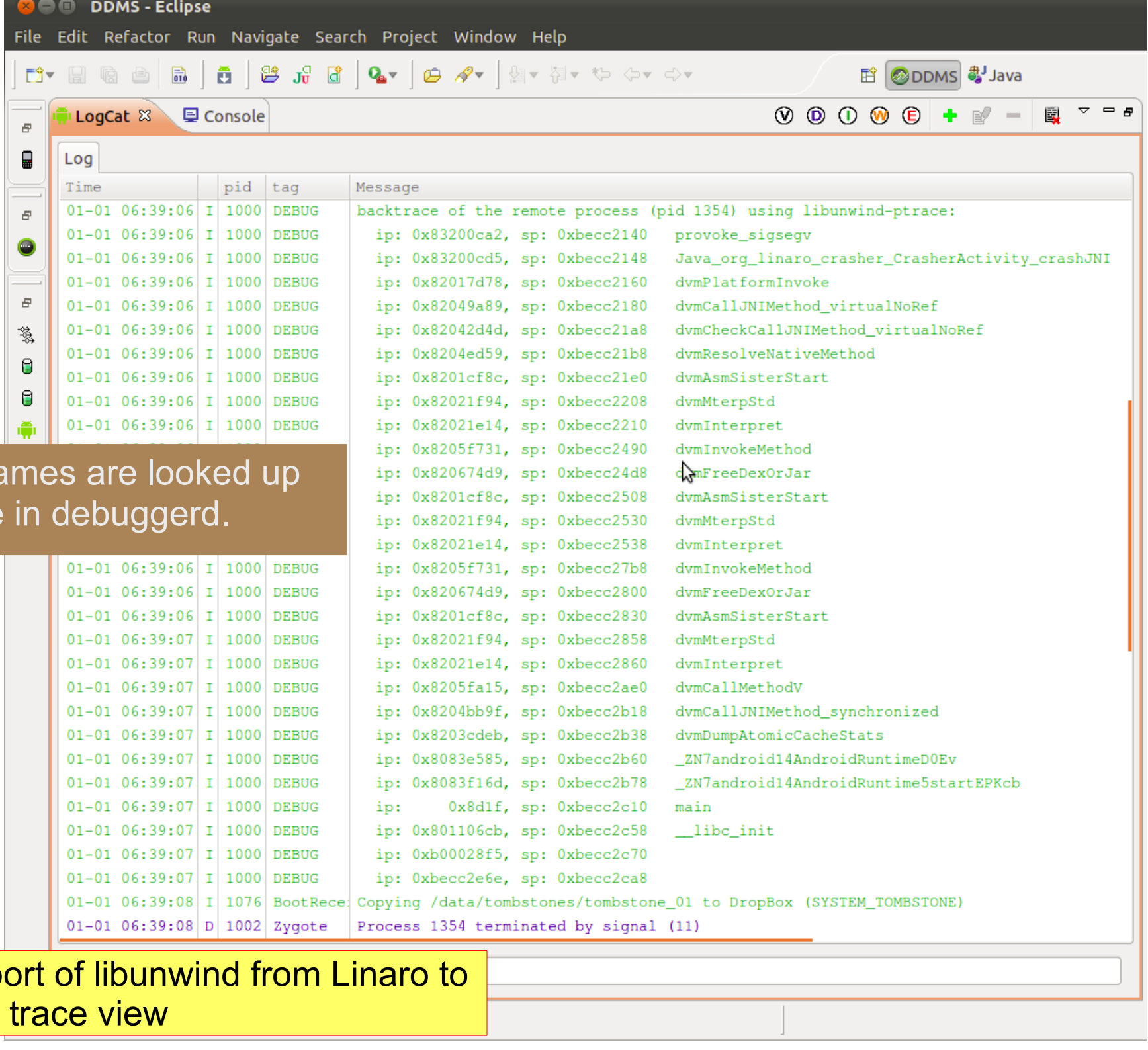
```
/home/jserv/cyanogen-
ics/bionic/libc/bionic/libc_init_dynamic.c:99
```

```
Line 94   __noreturn void __libc_init(uintptr_t *elfdata,
Line 95                    void (*onexit)(void),
Line 96                    int (*slingshot)(int, char**, char**),
Line 97                    structors_array_t const * const structors)
Line 98   {
Line 99       int     argc = (int)*elfdata;
```

How Debuggerd Works

- Android dynamic linker provides its own _start routine that registers a signal handler on SIGSEGV and the like.
- Whenever a dynamically linked executable crashes, the signal handler gets invoked and transmits the thread id of the crashing process to the debuggerd via a local socket.
  - bionic/linker/debugger.c
- The debuggerd uses ptrace to get the register contents of the crashing process and to display the call chain.
  - system/core/debuggerd/debuggerd.c


DDMS - Eclipse
File Edit Refactor Run Navigate Search Project Window Help
DDMS
Java
LogCat
Console
Log
Time pid tag Message
01-01 00:12:22 I 982 DEBUG #00 pc 00000ca2 /data/data/org.linaro.crasher/lib/libcrasher-jni.so
01-01 00:12:22 I 982 DEBUG #01 pc 00000cd0 /data/data/org.linaro.crasher/lib/libcrasher-jni.so
01-01 00:12:22 I 982 DEBUG #02 pc 00017d74 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #03 pc 00049a86 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #04 pc 00042d48 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #05 pc 0004ed56 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #06 pc 0001cf88 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #07 pc 00021f90 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #08 pc 00021e10 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #09 pc 0005f72e /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #10 pc 000674d4 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #11 pc 0001cf88 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #12 pc 00021f90 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #13 pc 00021e10 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #14 pc 0005f72e /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #15 pc 000674d4 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #16 pc 0001cf88 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #17 pc 00021f90 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #18 pc 00021e10 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #19 pc 0005fa12 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #20 pc 0004bb9a /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #21 pc 0003cde8 /system/lib/libdvm.so
01-01 00:12:22 I 982 DEBUG #22 pc 0003e582 /system/lib/libandroid_runtime.so
01-01 00:12:22 I 982 DEBUG #23 pc 0003f168 /system/lib/libandroid_runtime.so
01-01 00:12:22 I 982 DEBUG #24 pc 00008d1c /system/bin/app_process
01-01 00:12:22 I 982 DEBUG #25 pc 000106c8 /system/lib/libc.so
01-01 00:12:22 I 982 DEBUG #26 pc b00028f0 /system/bin/linker
Filter:



ELF symbol names are looked up and browsable in debuggerd.

Use Android port of libunwind from Linaro to Improve stack trace view

Two implementations for `__libc_init`

- Two implementations
  - File `bionic/libc/bionic/libc_init_dynamic.c`

```
/* This function is called from the executable's _start entry point
 * (see arch-$ARCH/bionic/crtbegin_dynamic.S), which is itself
 * called by the dynamic linker after it has loaded all shared
 * libraries the executable depends on.
```

  - File `bionic/libc/bionic/libc_init_static.c`

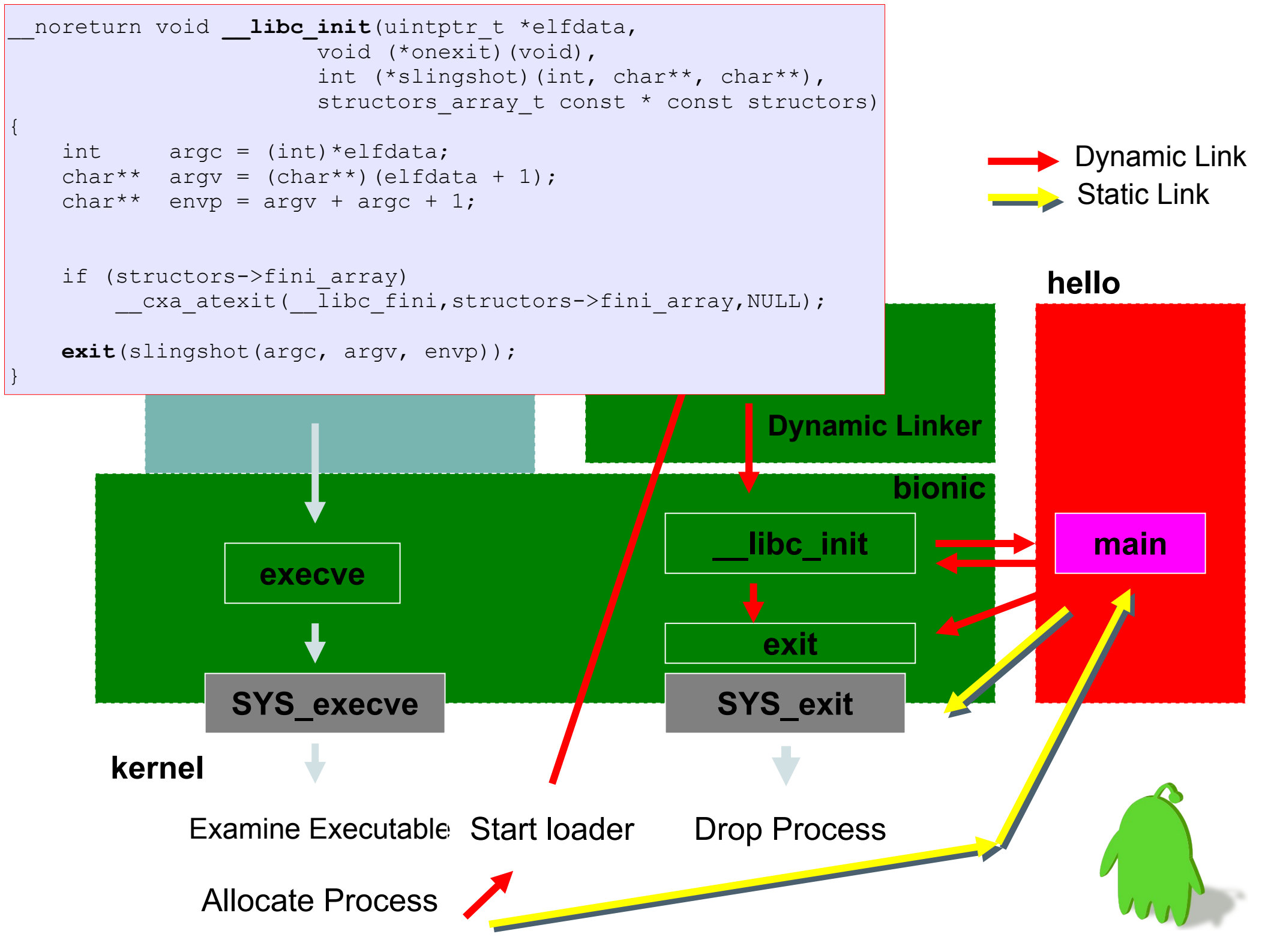
```
__noreturn void __libc_init(uintptr_t *elfdata,
                            void (*onexit)(void),
                            int (*slingshot)(int, char**, char**),
                            structors_array_t const * const structors)
```

- Very similar to each other
- Require the corresponding `crtbegin_{dynamic,static}.S`

```
__noreturn void __libc_init(uintptr_t *elfdata,
                      void (*onexit)(void),
                      int (*slingshot)(int, char**, char**),
                      structors_array_t const * const structors)
{
    int     argc = (int)*elfdata;
    char**  argv = (char**)(elfdata + 1);
    char**  envp = argv + argc + 1;


    if (structors->fini_array)
        __cxa_atexit(__libc_fini,structors->fini_array,NULL);

    exit(slingshot(argc, argv, envp));
}
```



Memory Allocation while executing Hello World!

ELF

- Page size
- Virtual address memory segment (sections)
- Segment size

- Magic number
- type (.o / .so / exec)
- Machine
- byte order
- ...

- Initialized (static) data

- code

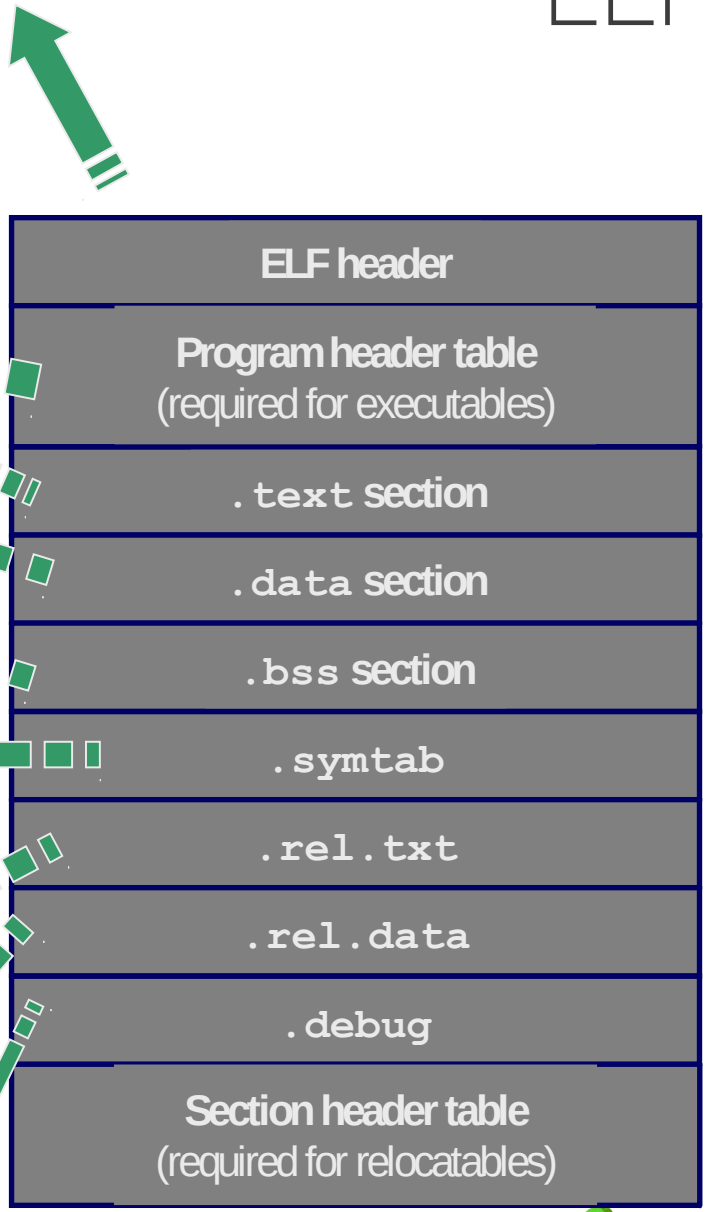
- Un-initialized (static) data
- Block started by symbol
- **Has section header but occupies no space**

- Symbol table
- Procedure and static variable names
- Section name

- Relocation info for .text section
- Addresses of instructions that need to be modified in the executable instructions for modifying.

- Relocation info for .data section
- Address pointer data will need to be modified in the merged executable

- Info for symbolic debugging



```
$ readelf -s hello

Symbol table '.dynsym' contains 5 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000   399 FUNC    GLOBAL DEFAULT  UND puts@GLIBC_2.0 (2)
     2: 00000000   415 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.0 (2)
     3: 08048438     4 OBJECT  GLOBAL DEFAULT   14 _IO_stdin_used
     4: 00000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__

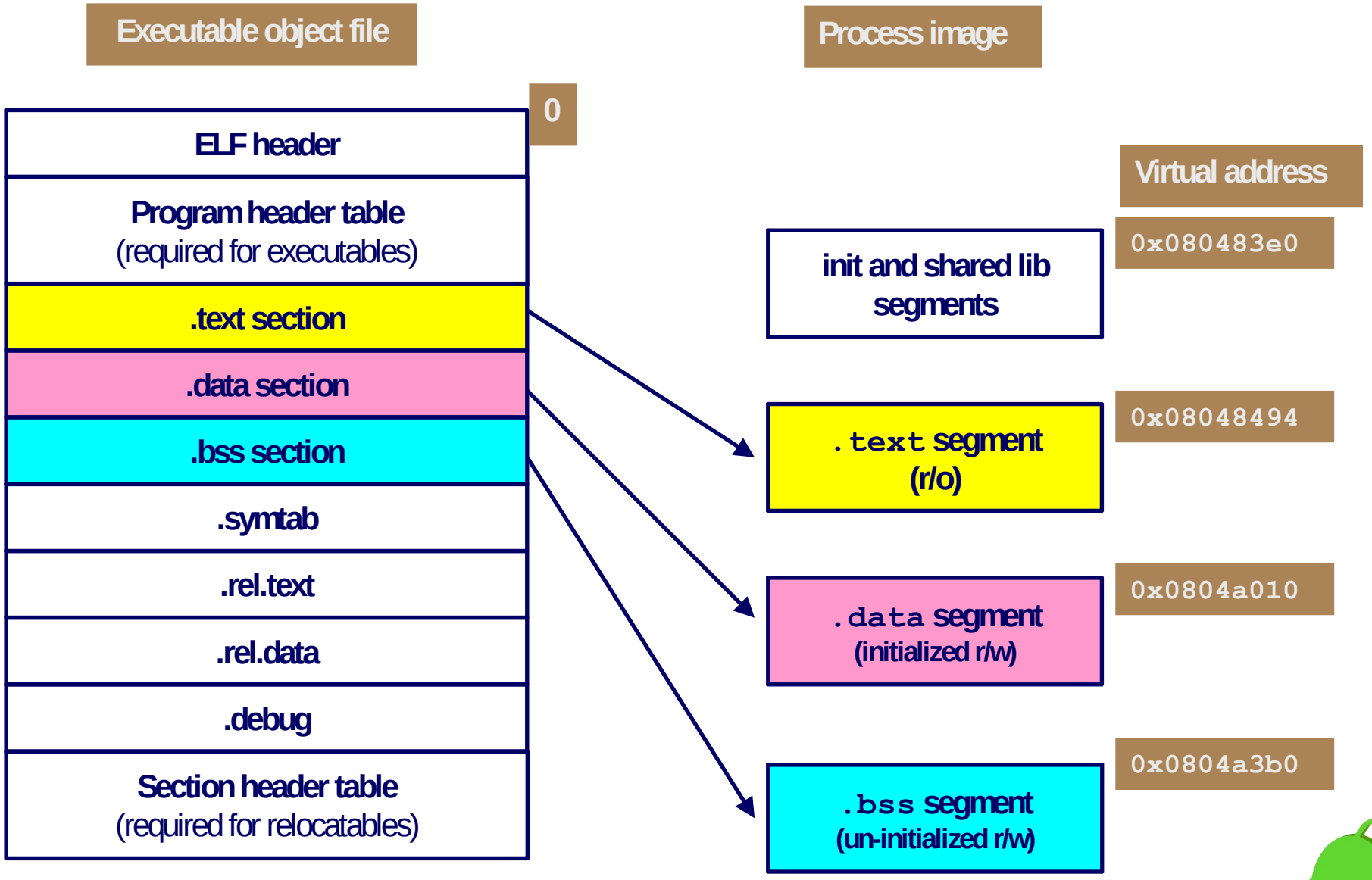
Symbol table '.symtab' contains 81 entries:
```

```
$ cp -f hello hello.strip
$ strip –s hello.strip
```

- -s|--syms|--symbols
  - Displays the entries in symbol table section of the file, if it has one.

```
$ readelf -s hello.strip

Symbol table '.dynsym' contains 5 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000   399 FUNC    GLOBAL DEFAULT  UND puts@GLIBC_2.0 (2)
     2: 00000000   415 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.0 (2)
     3: 08048438     4 OBJECT  GLOBAL DEFAULT   14 _IO_stdin_used
     4: 00000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
```



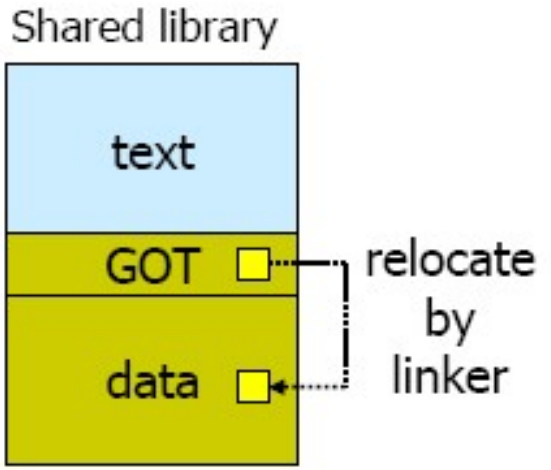
Loading ELF Binaries...

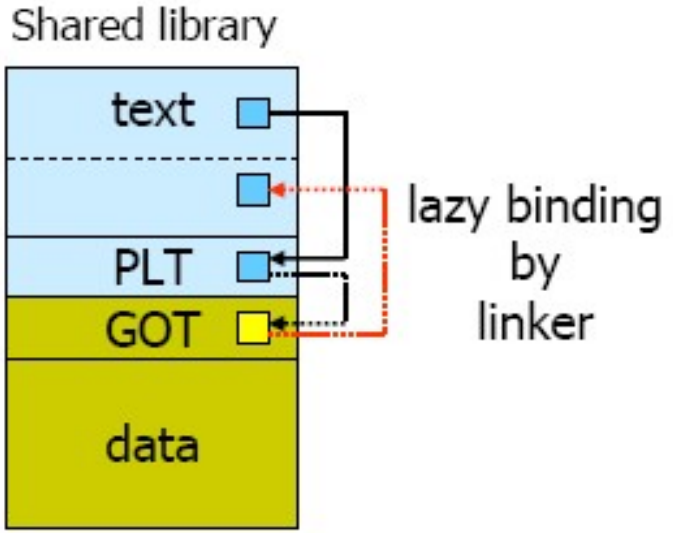
```
// global data reference

    call L1
L1: popl %ebx
    addl $GOTENTRY, %ebx
    movl (%ebx), %eax
    movl (%eax), %eax
```

```
// global procedure reference

    call $PLTENTRY
    …
-----------------------------------------
PLT[0]:   pushl (GOT[1])     // special id
          jmp  *(GOT[2])     // dyn linker
          …
PLTENTRY: jmp  *($GOTENTRY)
PLTLAZY:  pushl $func_id
          jmp  PLT[0]
-----------------------------------------
GOT[1]  : (special id for dyn linker)
GOT[2]  : (entry point in dynamic linker)
          …
GOTENTRY: $PLTLAZY           // changed by
          real entry point   // lazy binding
```

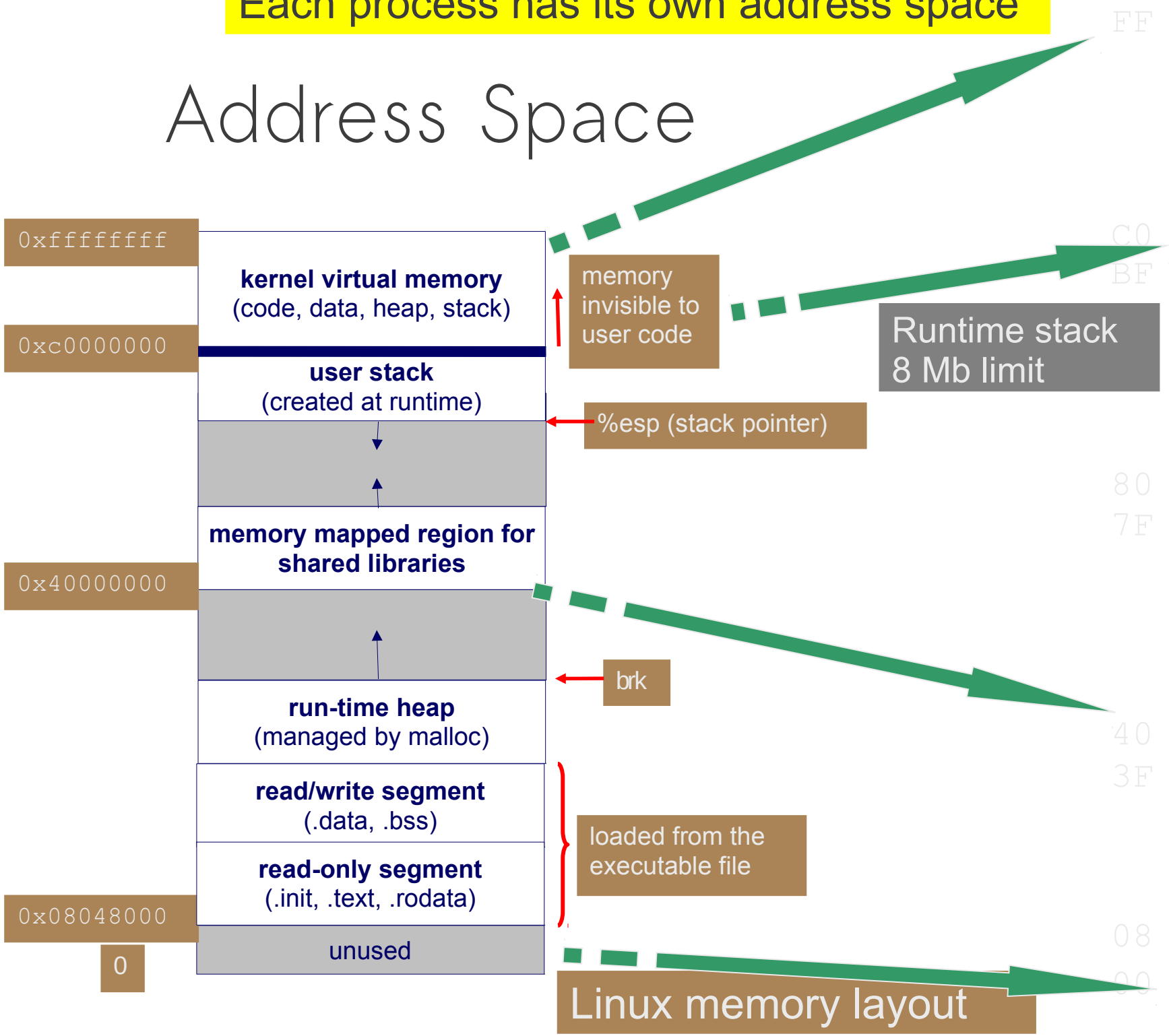




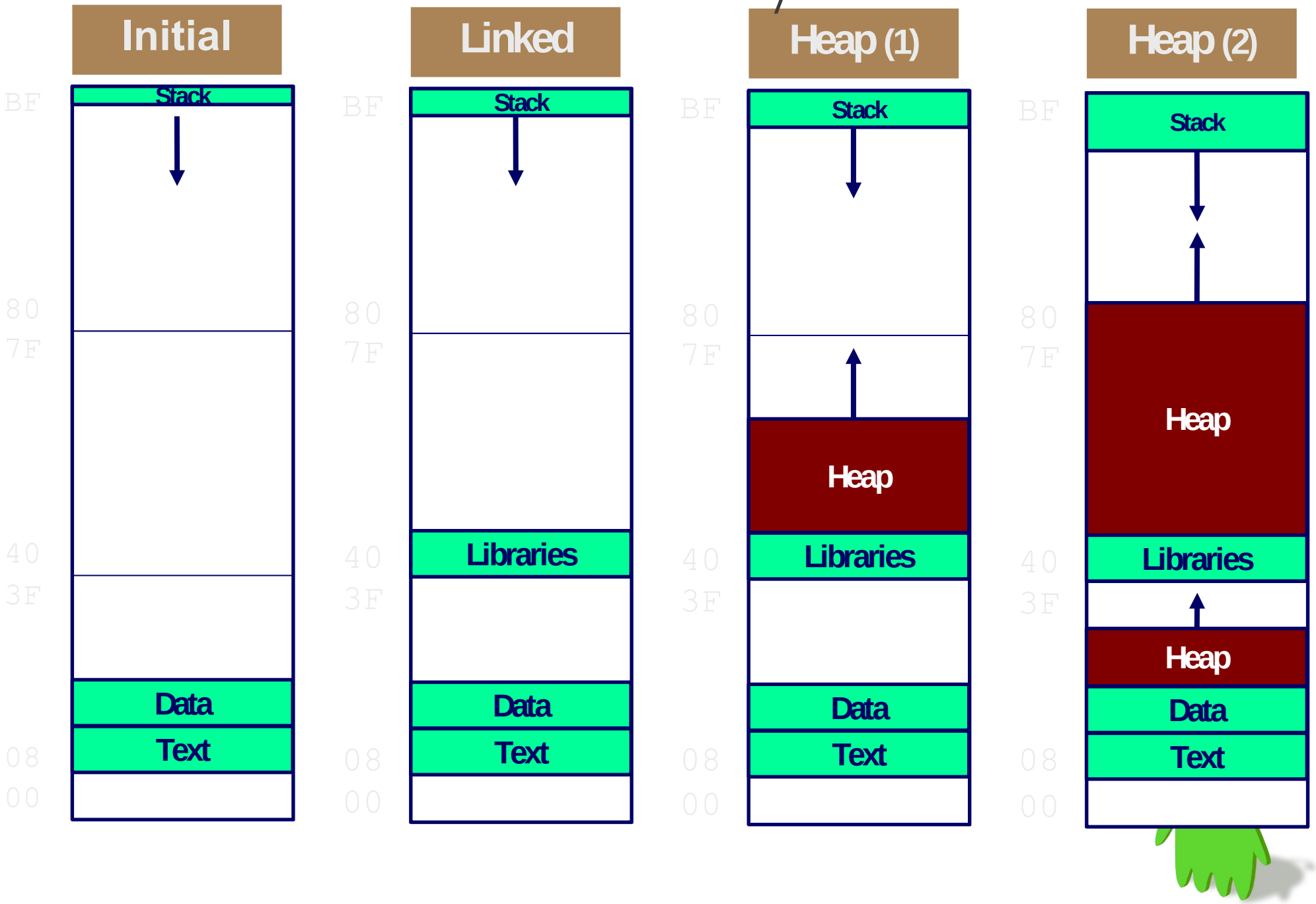
GOT (global offset table):in data segment
PLT (procedure linkage table): in code segment

Each process has its own address space

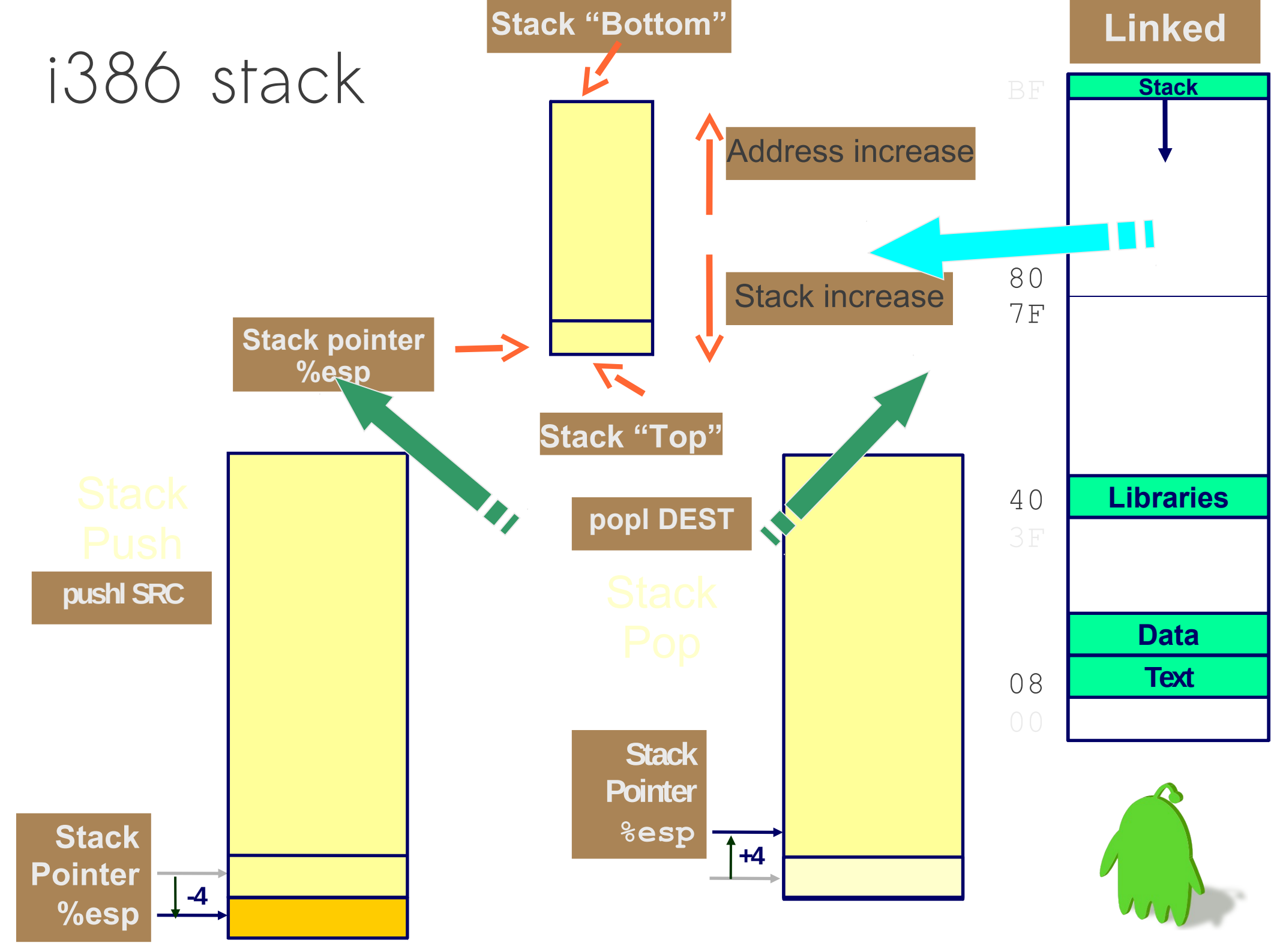
Address Space



Linux Memory Allocation



i386 stack


Stack "Bottom"
Address increase
Stack increase
Stack pointer %esp
Stack "Top"
Stack Push
pushl SRC
popl DEST
Stack Pop
Stack Pointer %esp
-4
+4
Stack Pointer %esp
Linked
BF
Stack
80
7F
40
3F
Libraries
Data
Text
08
00

Recall process maps

- `adb shell cat /proc/18696/maps`

```
00008000-00009000 r-xp 00000000 b3:02 8959       /data/local/hello
00009000-0000a000 rwxp 00001000 b3:02 8959       /data/local/hello
40061000-40062000 r-xp 00000000 00:00 0
40079000-40081000 r-xs 00000000 00:0b 392        /dev/__properties__ (deleted)
40087000-400c9000 r-xp 00000000 b3:01 548        /system/lib/libc.so
400c9000-400cc000 rwxp 00042000 b3:01 548        /system/lib/libc.so
400cc000-400d7000 rwxp 00000000 00:00 0
400d7000-400ec000 r-xp 00000000 b3:01 597        /system/lib/libm.so
400ec000-400ed000 rwxp 00015000 b3:01 597        /system/lib/libm.so
40101000-40102000 r-xp 00000000 b3:01 644        /system/lib/libstdc++.so
40102000-40103000 rwxp 00001000 b3:01 644        /system/lib/libstdc++.so
b0001000-b0009000 r-xp 00001000 b3:01 128        /system/bin/linker
b0009000-b000a000 rwxp 00009000 b3:01 128        /system/bin/linker
b000a000-b0015000 rwxp 00000000 00:00 0
beb07000-beb28000 rw-p 00000000 00:00 0          [stack]
ffff0000-ffff1000 r-xp 00000000 00:00 0          [vectors]
```

Meaning of each field:

`start-end  perm offset major:minor inode image`

Start-end: The beginning and ending virtual addresses for this memory area.
Perm: a bit mask with the memroy area's read, write, and execute permissions
Offset: Where the memory area begins in the file
Major/Minor: Major and minor numbers of the device holding the file (or partition)

Map vs. ELF sections

- **`arm-eabi-objdump -h \`**
  `../out/target/product/crespo/system/bin/hello`

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .interp       00000013  00008114  00008114  00000114  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
...
  4 .rel.plt      00000018  000083dc  000083dc  000003dc  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .plt          00000038  000083f4  000083f4  000003f4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  6 .text         00000054  00008430  00008430  00000430  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  7 .rodata       0000000d  00008484  00008484  00000484  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
```

- **`adb shell cat /proc/18696/maps`**

```
00008000-00009000 r-xp 00000000 b3:02 8959       /data/local/hello
00009000-0000a000 rwxp 00001000 b3:02 8959       /data/local/hello
```

`.data` section of 'hello' program → Readable and Writable

`.text` section of 'hello' program → Read-only

Recall process maps

- **adb shell cat /proc/18696/maps**

```
00008000-00009000 r-xp 00000000 b3:02 8959      /data/local/hello
00009000-0000a000 rwxp 00001000 b3:02 8959      /data/local/hello
40061000-40062000 r-xp 00000000 00:00 0
40079000-40081000 r-xs 00000000 00:0b 392       /dev/__properties__ (deleted)
40087000-400c9000 r-xp 00000000 b3:01 548       /system/lib/libc.so
400c9000-400cc000 rwxp 00042000 b3:01 548       /system/lib/libc.so
400cc000-400d7000 rwxp 00000000 00:00 0
400d7000-400ec000 r-xp 00000000 b3:01 597       /system/lib/libm.so
400ec000-400ed000 rwxp 00015000 b3:01 597       /system/lib/libm.so
40101000-40102000 r-xp 00000000 b3:01 644       /system/lib/libstdc++.so
40102000-40103000 rwxp 00001000 b3:01 644       /system/lib/libstdc++.so
b0001000-b0009000 r-xp 00001000 b3:01 128       /system/bin/linker
b0009000-b000a000 rwxp 00009000 b3:01 128       /system/bin/linker
b000a000-b0015000 rwxp 00000000 00:00 0
beb07000-beb28000 rw-p 00000000 00:00 0         [stack]
ffff0000-ffff1000 r-xp 00000000 00:00 0         [vectors]
```

.bss section of libc

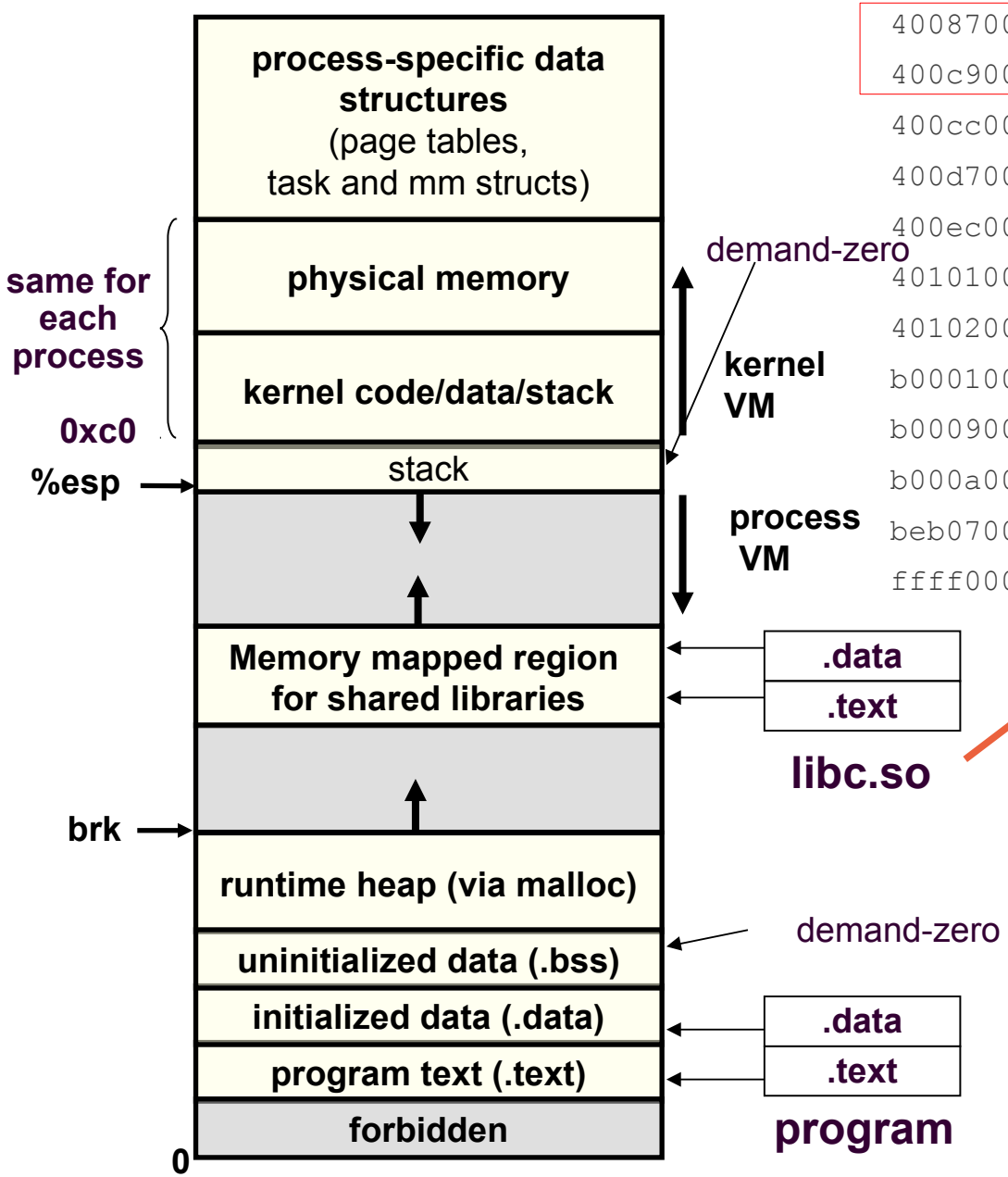
.data section of dynamic linker

.bss section of dynamic linker

.text section of dynamic linker

- `adb shell cat /proc/18696/maps`

```
00008000-00009000 r-xp 00000000 /data/local/hello
00009000-0000a000 rwxp 00001000 /data/local/hello
40061000-40062000 r-xp 00000000 00:00 0
40079000-40081000 r-xs 00000000 /dev/__properties
40087000-400c9000 r-xp 00000000 /system/lib/libc.so
400c9000-400cc000 rwxp 00042000 /system/lib/libc.so
400cc000-400d7000 rwxp 00000000 00:00 0
400d7000-400ec000 r-xp 00000000 /system/lib/libm.so
400ec000-400ed000 rwxp 00015000 /system/lib/libm.so
40101000-40102000 r-xp 00000000 /system/lib/libstdc++.so
40102000-40103000 rwxp 00001000 /system/lib/libstdc++.so
b0001000-b0009000 r-xp 00001000 /system/bin/linker
b0009000-b000a000 rwxp 00009000 /system/bin/linker
b000a000-b0015000 rwxp 00000000 00:00 0
beb07000-beb28000 rw-p 00000000 [stack]
ffff0000-ffff1000 r-xp 00000000 [vectors]
```



NOTE: reverse order between process view and maps

Case Study

GDB meets Angry Birds

Setup (1)

- Download Android NDK
  - http://developer.android.com/sdk/ndk/
  - Version: r7b / r5c
- Download apktool
  - a tool for reverse engineering 3rd party, closed, binary Android apps. It can decode resources to nearly original form and rebuild them after making some modifications
  - http://code.google.com/p/android-apktool/
- Download Angry Birds Rio from Androi Market
  - Date: Feb 3, 2012

APK content

```
$ unzip Angry+Birds.apk
Archive:  Angry+Birds.apk
...
  inflating: AndroidManifest.xml
 extracting: resources.arsc
 extracting: res/drawable-hdpi/icon.png
 extracting: res/drawable-ldpi/icon.png
 extracting: res/drawable-mdpi/icon.png
  inflating: classes.dex
  inflating: lib/armeabi/libangrybirds.so
  inflating: lib/armeabi-v7a/libangrybirds.so
  inflating: META-INF/MANIFEST.MF
  inflating: META-INF/CERT.SF
  inflating: META-INF/CERT.RSA
```

Dalvik DEX

JNI

manifest + signature

AndroidManifest


```
$ unzip Angry+Birds.
Archive:  Angry+Bird
...
...
  inflating: AndroidManifest.xml
 extracting: resources.arsc
```

```
$ file AndroidManifest.xml
AndroidManifest.xml: DBase 3 data file (2328 records)

$ apktool d ../AngryBirds/Angry+Birds.apk
I: Baksmaling...
I: Loading resource table...
...
I: Decoding file-resources...
I: Decoding values*/* XMLs...
I: Done.
I: Copying assets and libs...
$ file Angry+Birds/AndroidManifest.xml
Angry+Birds/AndroidManifest.xml: XML  document text
```

so

Setup (2)

- Pull Angry Birds' APK and extract

```
adb pull /data/app/com.rovio.angrybirdsrio-1.apk
java -jar apktool.jar d com.rovio.angrybirdsrio-1.apk
```

- Prepare environment to satisfy NDK

```
cp -af com.rovio.angrybirdsrio-1/lib/ libs
mkdir -p obj/local/armeabi obj/local/armeabi-v7a
cp -f libs/armeabi/libangrybirds.so ./obj/local/armeabi/ \
                                    ./obj/local/armeabi-v7a/
echo "set solib-search-path `pwd`/obj/local/armeabi" > \
     libs/armeabi/gdb.setup
cp -f libs/armeabi/gdb.setup libs/armeabi-v7a/gdb.setup
```

- Use gdbserver from NDK

```
adb push android-ndk-r7b/toolchains/arm-linux-androideabi-4.4.3/prebuilt/gdbserver \
          /data/data/com.rovio.angrybirdsrio/lib
```

Use GDB from Android NDK(1)

- $ **android-ndk-r7b/ndk-gdb --verbose -force **
  --launch=com.rovio.ka3d.App

```
Android NDK installation path: /tmp/angry-birds/android-ndk-r7b
Using default adb command: /usr/bin/adb
ADB version found: Android Debug Bridge version 1.0.26
Using ADB flags:
Using auto-detected project path: .
Found package name: com.rovio.angrybirdsrio
...
Launching activity: com.rovio.angrybirdsrio/com.rovio.ka3d.App
## COMMAND: /usr/bin/adb shell am start -n
com.rovio.angrybirdsrio/com.rovio.ka3d.App
Starting: Intent { cmp=com.rovio.angrybirdsrio/com.rovio.ka3d.App }
...
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
warning: shared library handler failed to enable breakpoint
0x40043384 in epoll_wait ()
   from /tmp/angry-birds/obj/local/armeabi/libc.so
(gdb)
```

Use GDB from Android NDK(2)

- (gdb) **info shared**

```
From        To          Syms Read   Shared Object Library
                        No          /system/bin/linker
0x40041fc0  0x4006f99c  Yes         armeabi/libc.so
                        No          libstdc++.so
                        No          libm.so
                        No          liblog.so
                        No          libcutils.so
                        No          libz.so
...
0x507b0ae8  0x508ec948  Yes         armeabi/libangrybirds.so
```

Use GDB from Android NDK(3)

```
(gdb) set $m = (int*)malloc(480*699*4)
(gdb)
(gdb) set $f = fopen("/sdcard/outputfile", "wb+")
(gdb) call fwrite($m, 1, 480*699*4, $f)
$1 = 1342080
(gdb) call fclose($f)
$2 = 0
(gdb) x/s __progname
0xbef30d26:  "com.rovio.angrybirdsrio"

(gdb) print __page_size
$3 = 4096
```

You can dump memory/variables/buffer into storage for observation purpose.

Case Study

Binder IPC: The heart of Android

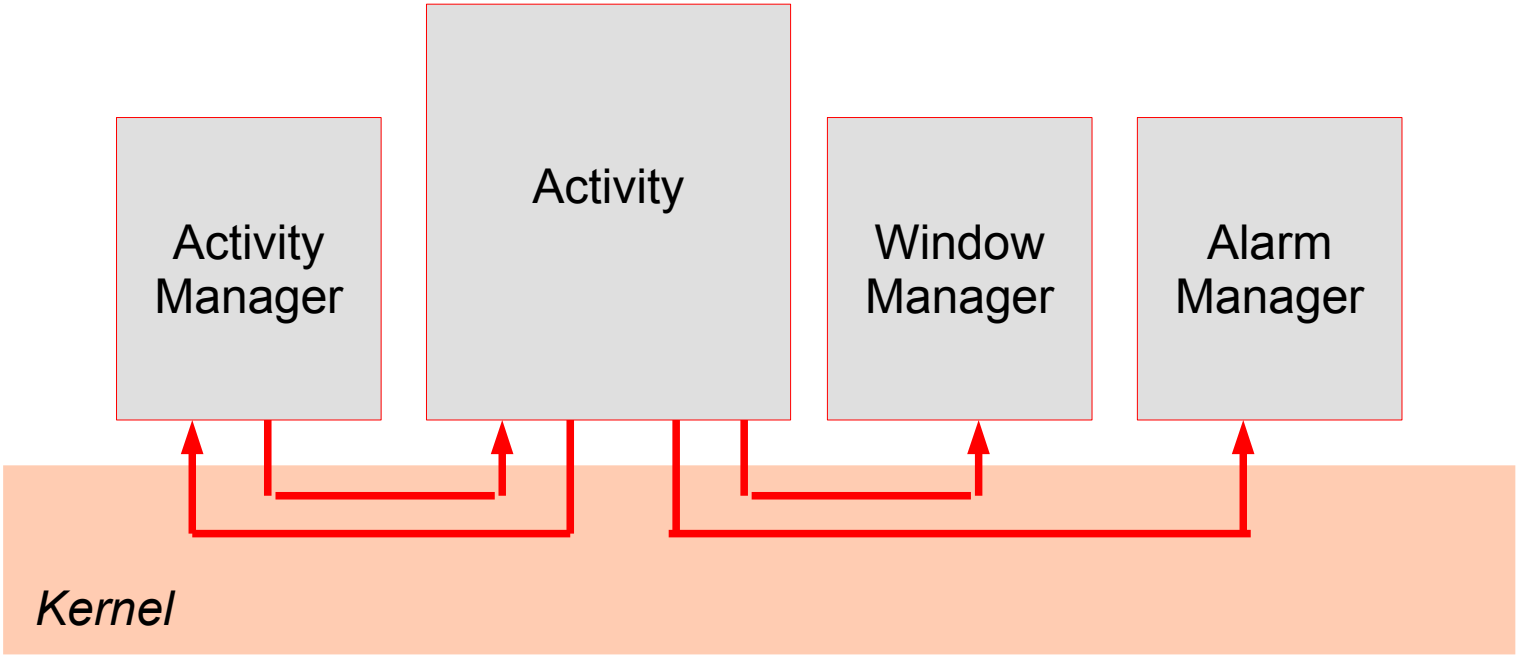
For detailed interaction and implementations, please check my presentation in 南台科技大學資工系 (March 19, 2012)

Processes running on Android

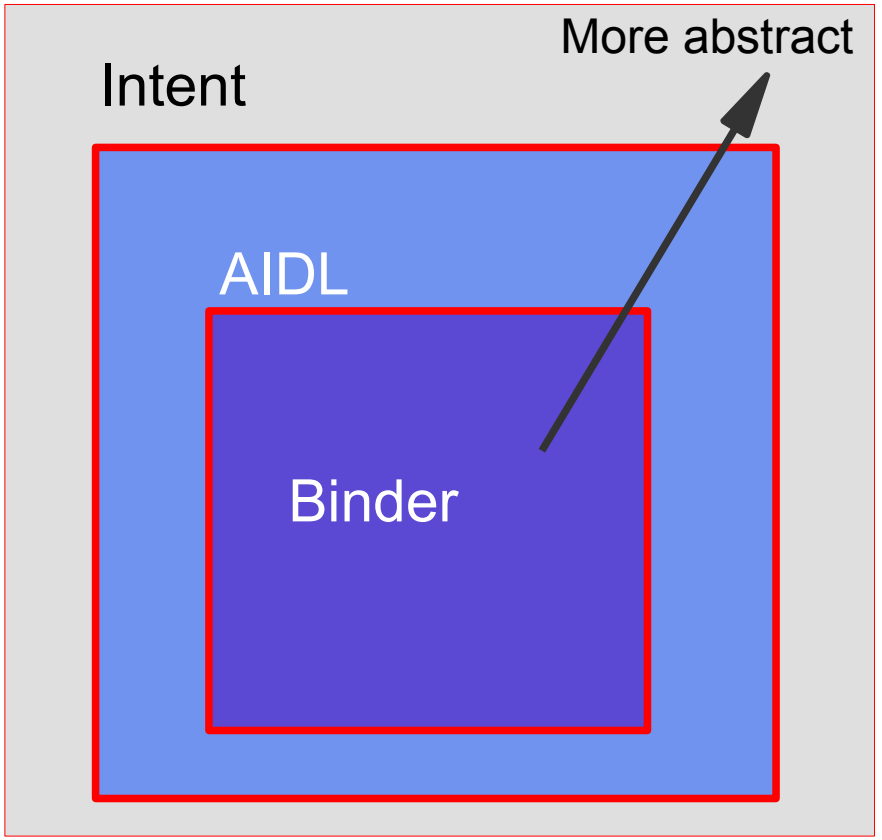
```
$ ps
    ...
root      37    1     248    156   c00aef2c 0000875c S /sbin/ueventd
system    42    1     768    260   c022950c afd0b6fc S /system/bin/servicemanager
root      43    1     3824   564   ffffffff afd0bdac S /system/bin/vold
root      44    1     3796   560   ffffffff afd0bdac S /system/bin/netd
root      45    1     628    264   c02588c0 afd0c0cc S /system/bin/debuggerd
radio     46    1     4336   672   ffffffff afd0bdac S /system/bin/rild
root      47    1     62224  27576 c00aef2c afd0b844 S zygote
media     48    1     16828  3736  ffffffff afd0b6fc S /system/bin/mediaserver
bluetooth 49    1     1216   572   c00aef2c afd0c59c S /system/bin/dbus-daemon
root      50    1     776    316   c02a8424 afd0b45c S /system/bin/installd
keystore  51    1     1704   432   c02588c0 afd0c0cc S /system/bin/keystore
shell     52    1     696    336   c0050934 afd0c3ac S /system/bin/sh
root      53    1     3356   160   ffffffff 00008294 S /sbin/adbd
system    67    47    172464 32596 ffffffff afd0b6fc S system_server
system    115   47    80028  20728 ffffffff afd0c51c S com.android.systemui
app_24    124   47    80732  20720 ffffffff afd0c51c S com.android.inputmethod.latin
radio     135   47    87848  20324 ffffffff afd0c51c S com.android.phone
app_18    144   47    89136  24160 ffffffff afd0c51c S com.android.launcher
app_7     165   47    86136  22736 ffffffff afd0c51c S android.process.acore
app_0     197   47    73996  17472 ffffffff afd0c51c S com.android.deskclock
app_14    208   47    75000  18464 ffffffff afd0c51c S android.process.media
app_3     219   47    72228  17652 ffffffff afd0c51c S com.android.bluetooth
app_25    234   47    85336  17836 ffffffff afd0c51c S com.android.mms
app_26    254   47    74656  19080 ffffffff afd0c51c S com.android.email
app_27    266   47    74912  18100 ffffffff afd0c51c S com.android.providers.calendar
app_1     285   47    71616  16280 ffffffff afd0c51c S com.android.protips
app_19    293   47    72184  16572 ffffffff afd0c51c S com.android.music
app_21    301   47    74728  17208 ffffffff afd0c51c S com.android.quicksearchbox
app_28    311   47    75408  18040 ffffffff afd0c51c S com.cooliris.media
shell     323   52    856    316   00000000 afd0b45c R ps
$
```

More than 30 processes (200+ threads).

IPC = Inter-Process Communication

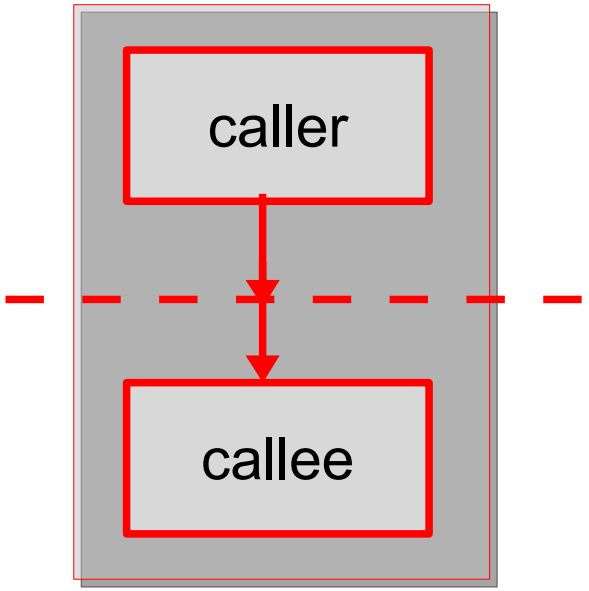


IPC Abstraction



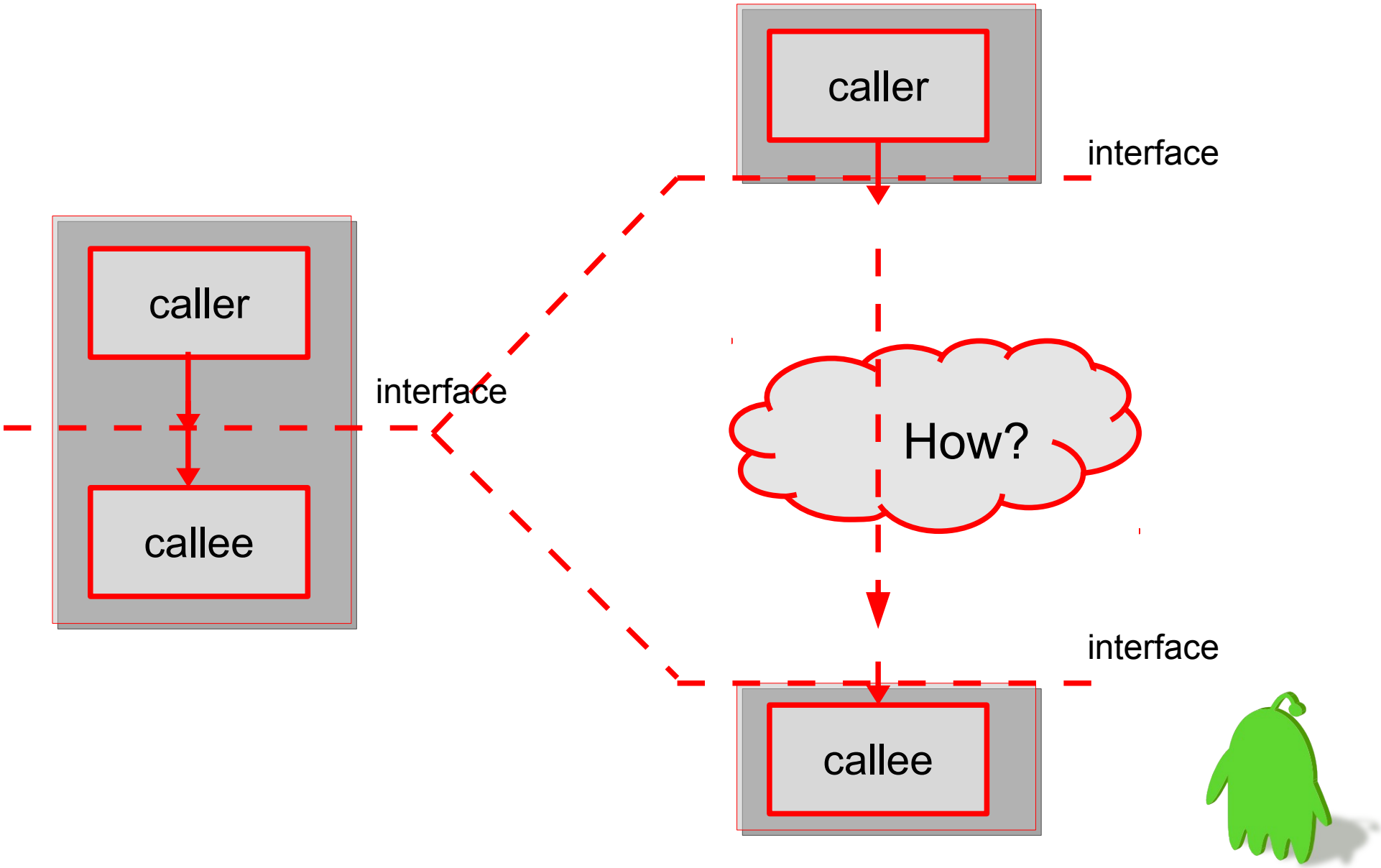
- Intent
  - The highest level abstraction
- Inter process method invocation
  - **AIDL**: Android Interface Definition Language
- binder: kernel driver
- ashmem: shared memory

Method invocation

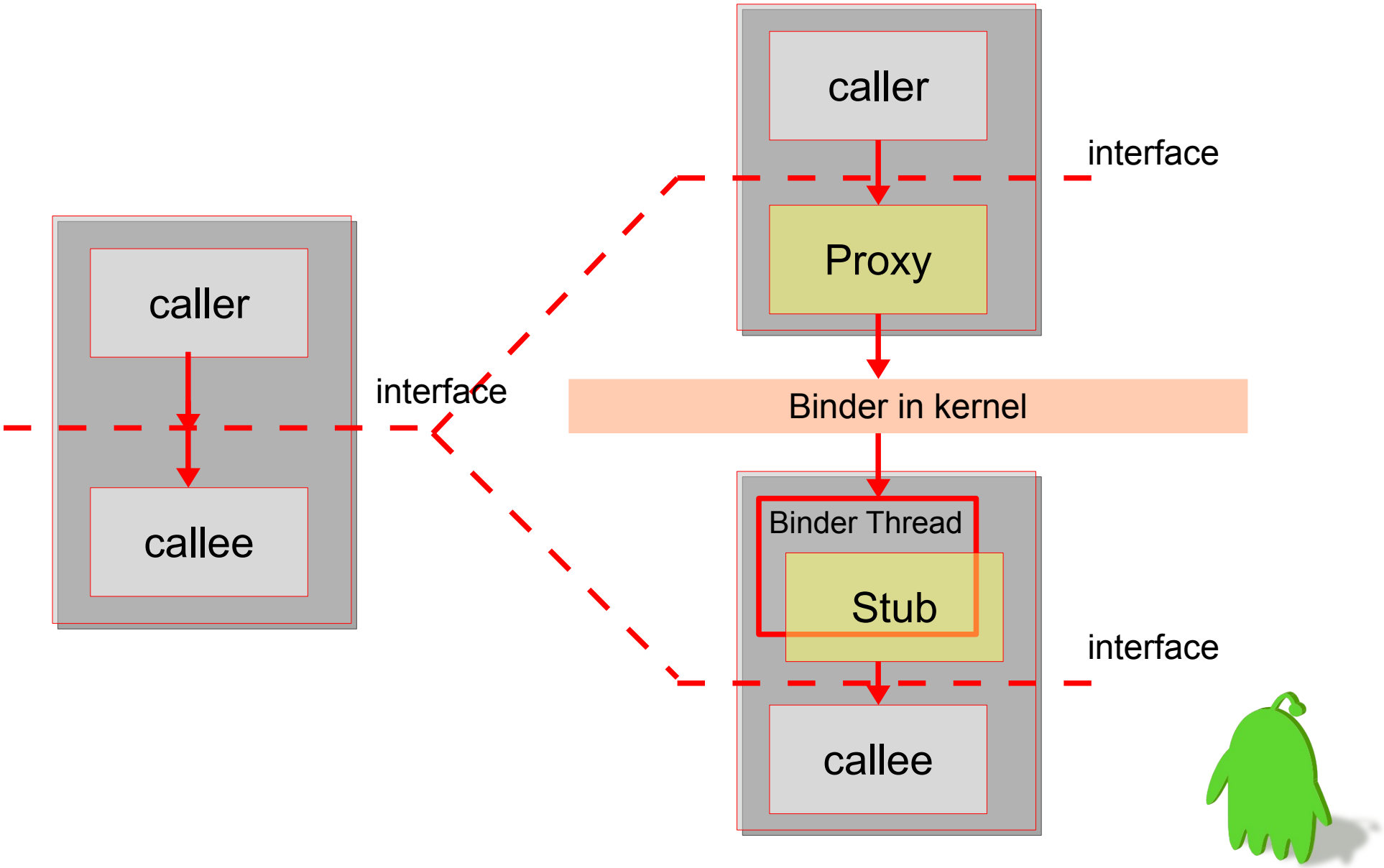


In the same process

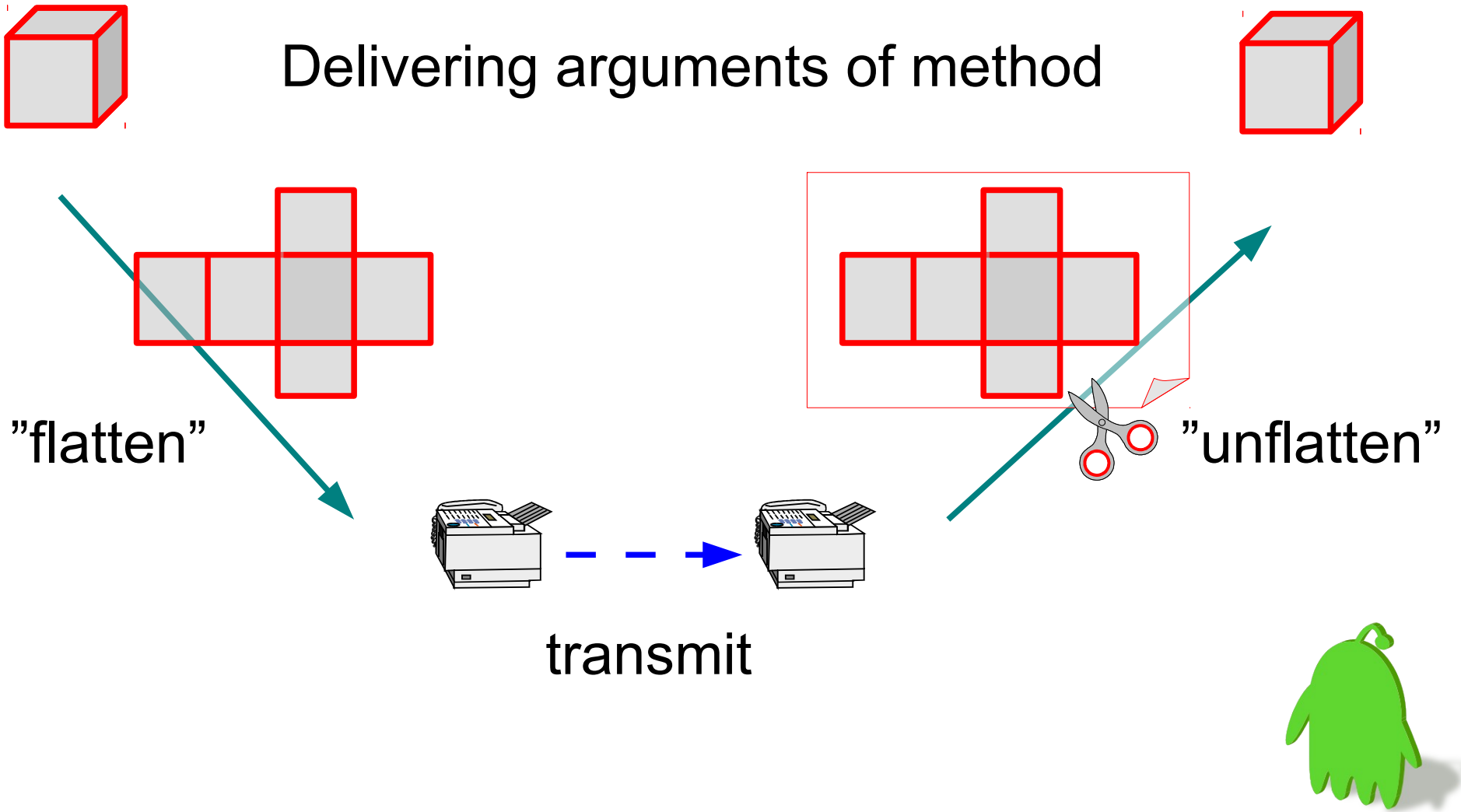
Inter-process method invocation



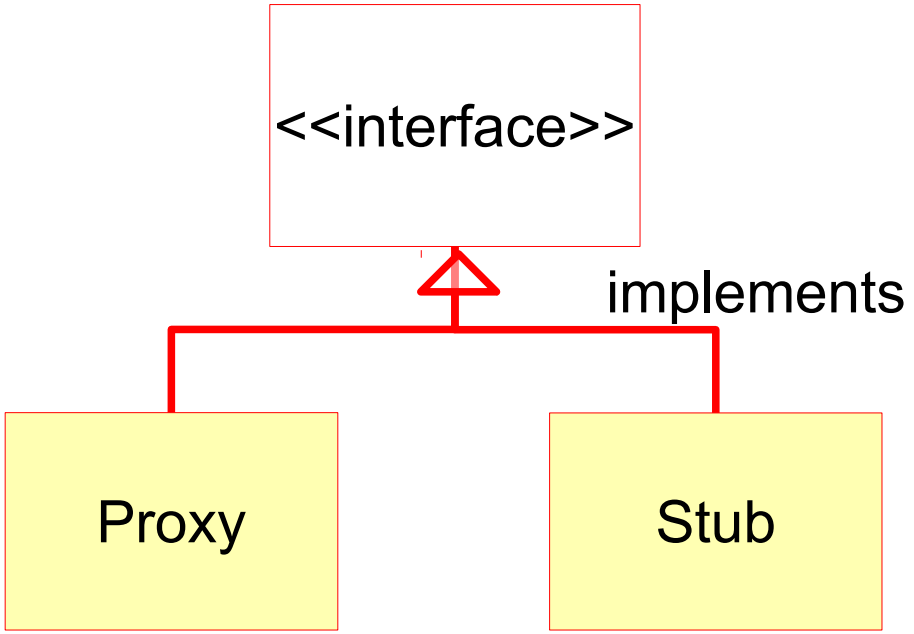
Inter-process method invocation



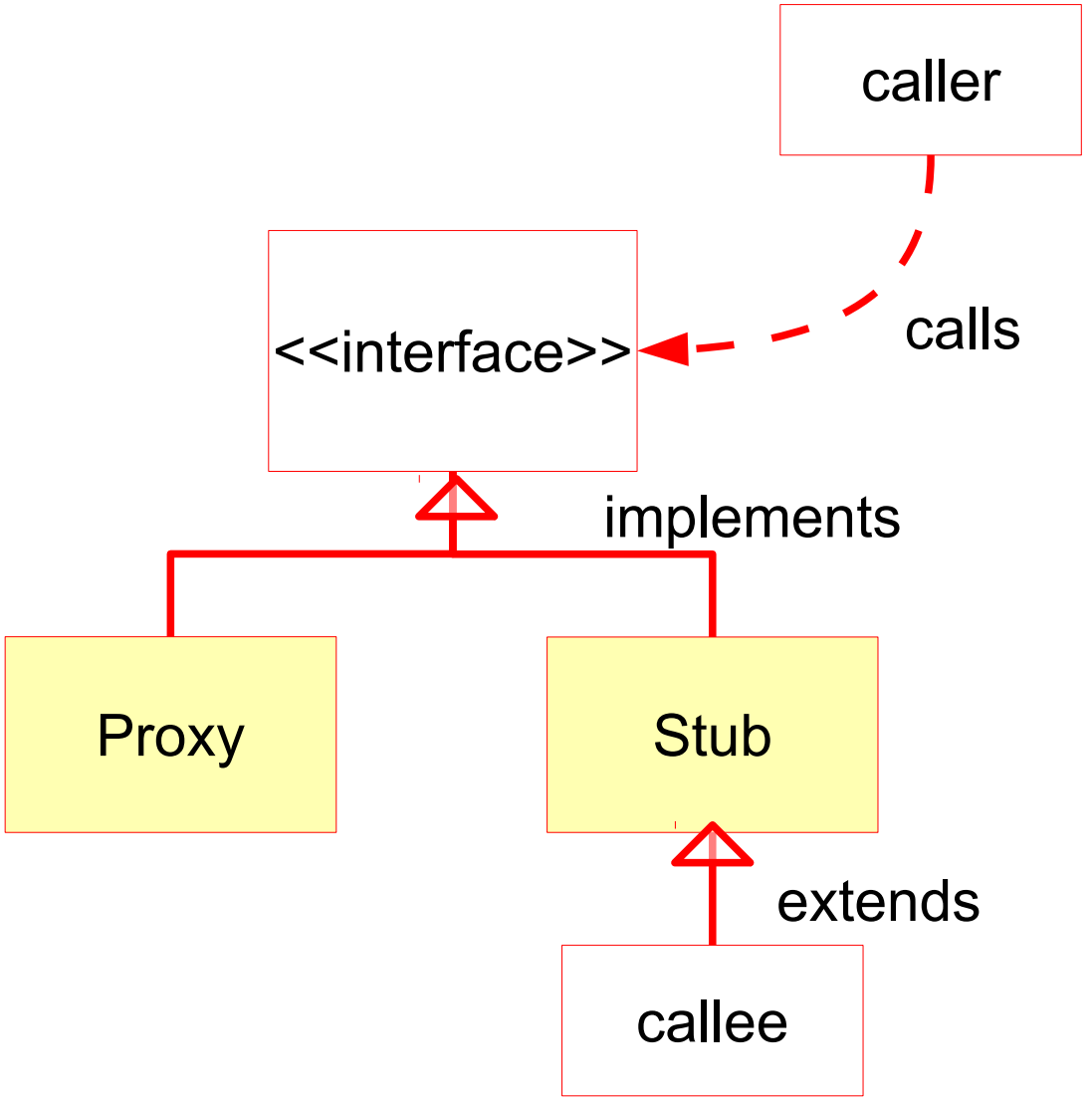
android.os.Parcel



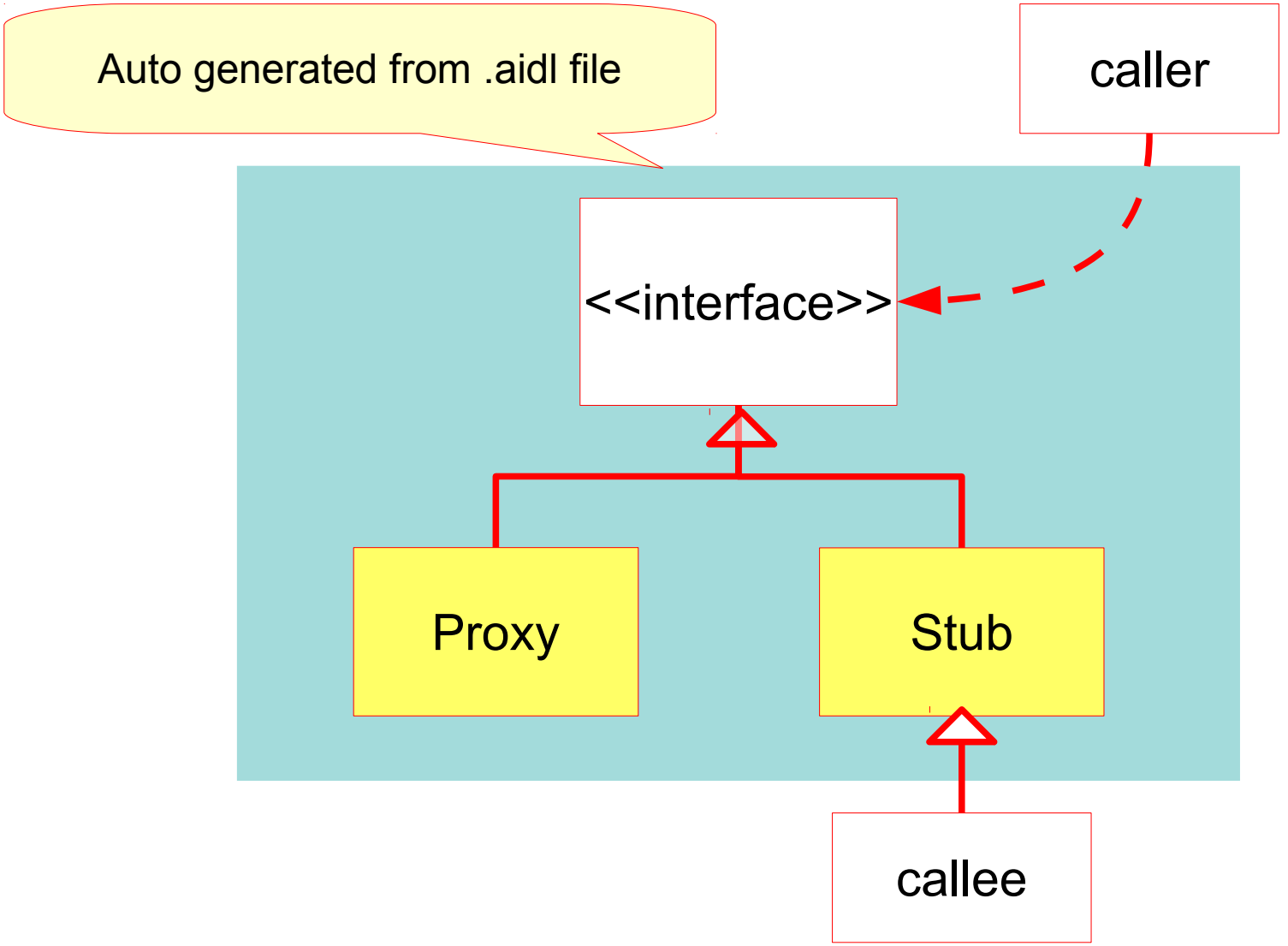
UML Representation



UML Representation

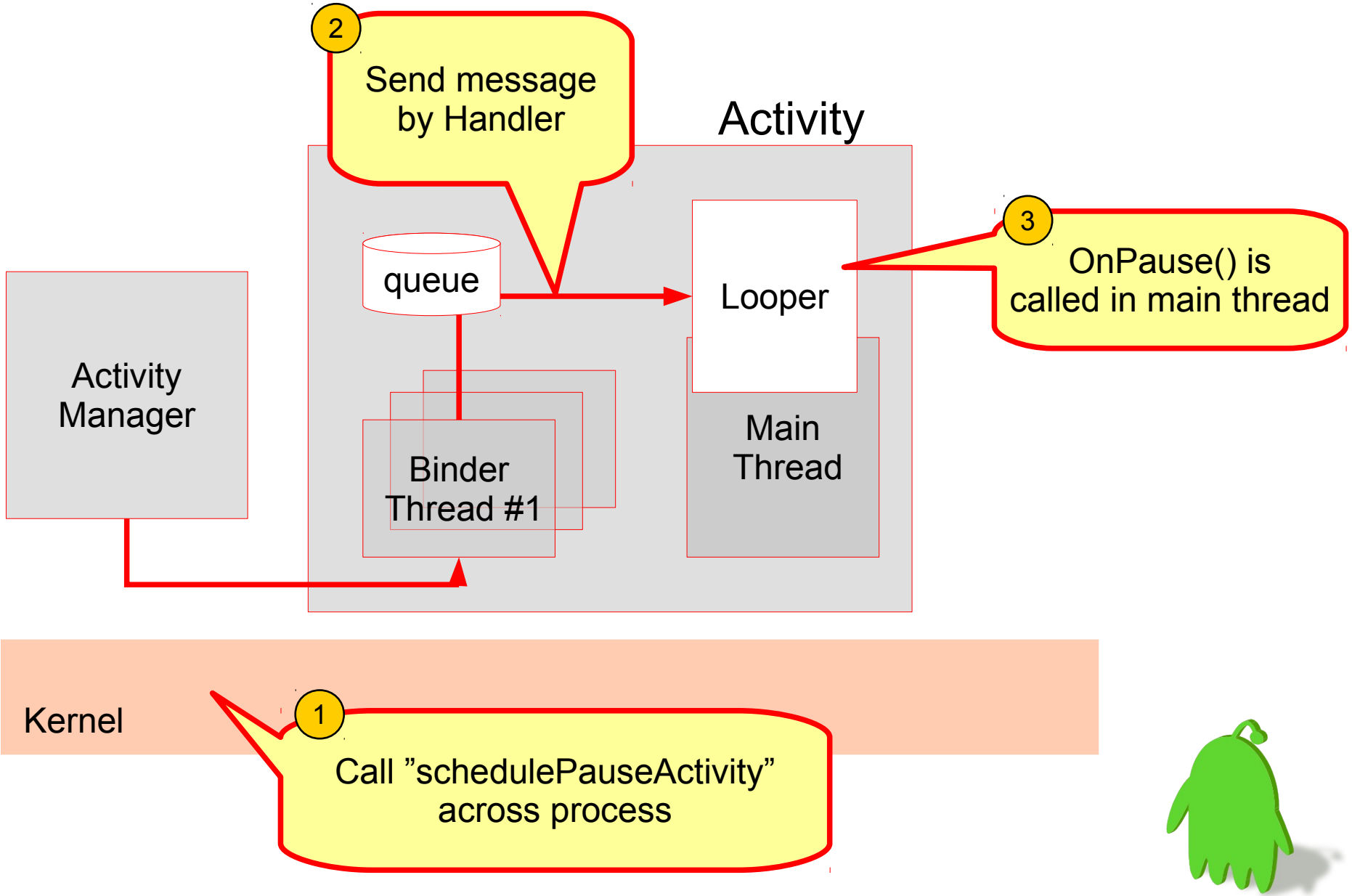


AIDL

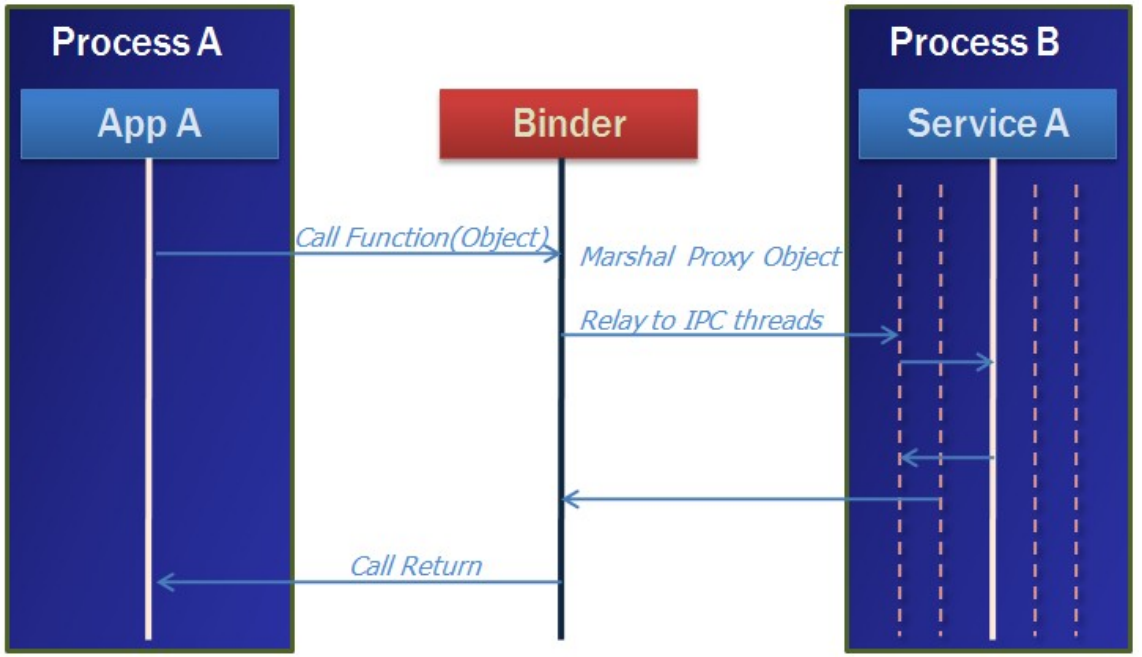

Auto generated from .aidl file
caller
<<interface>>
Proxy
Stub
callee

Use Case:

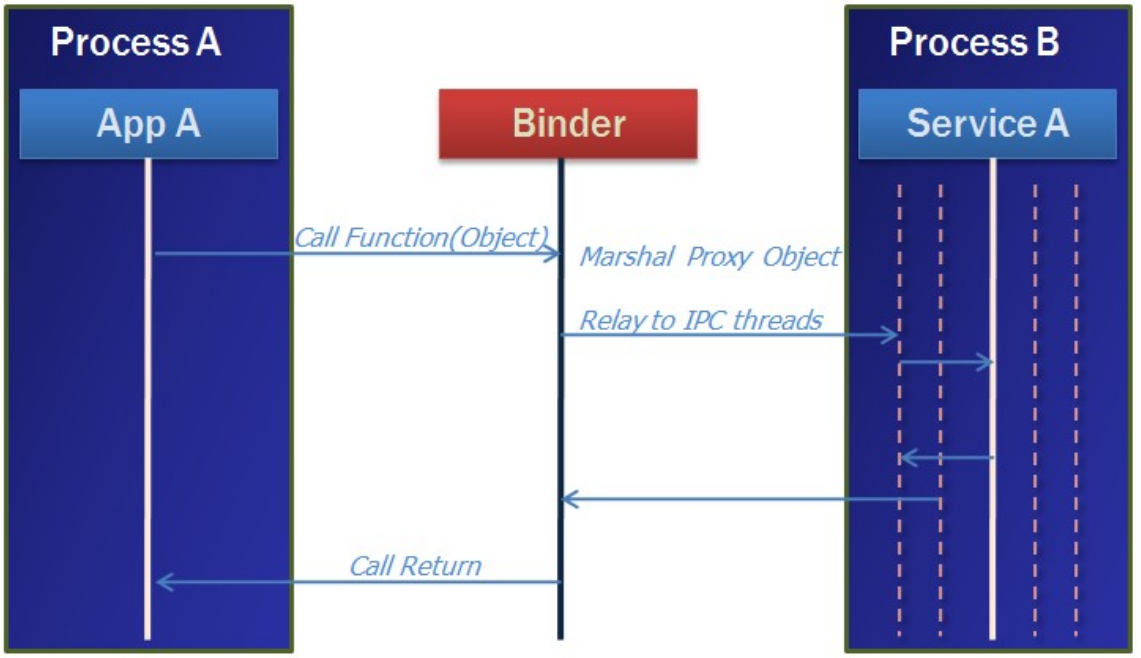
Who calls onPause() in Activity?



- Multi-thread aware
  - Have internal status per thead
  - Compare to UNIX socket: sockets have internal status per file descriptor (FD)

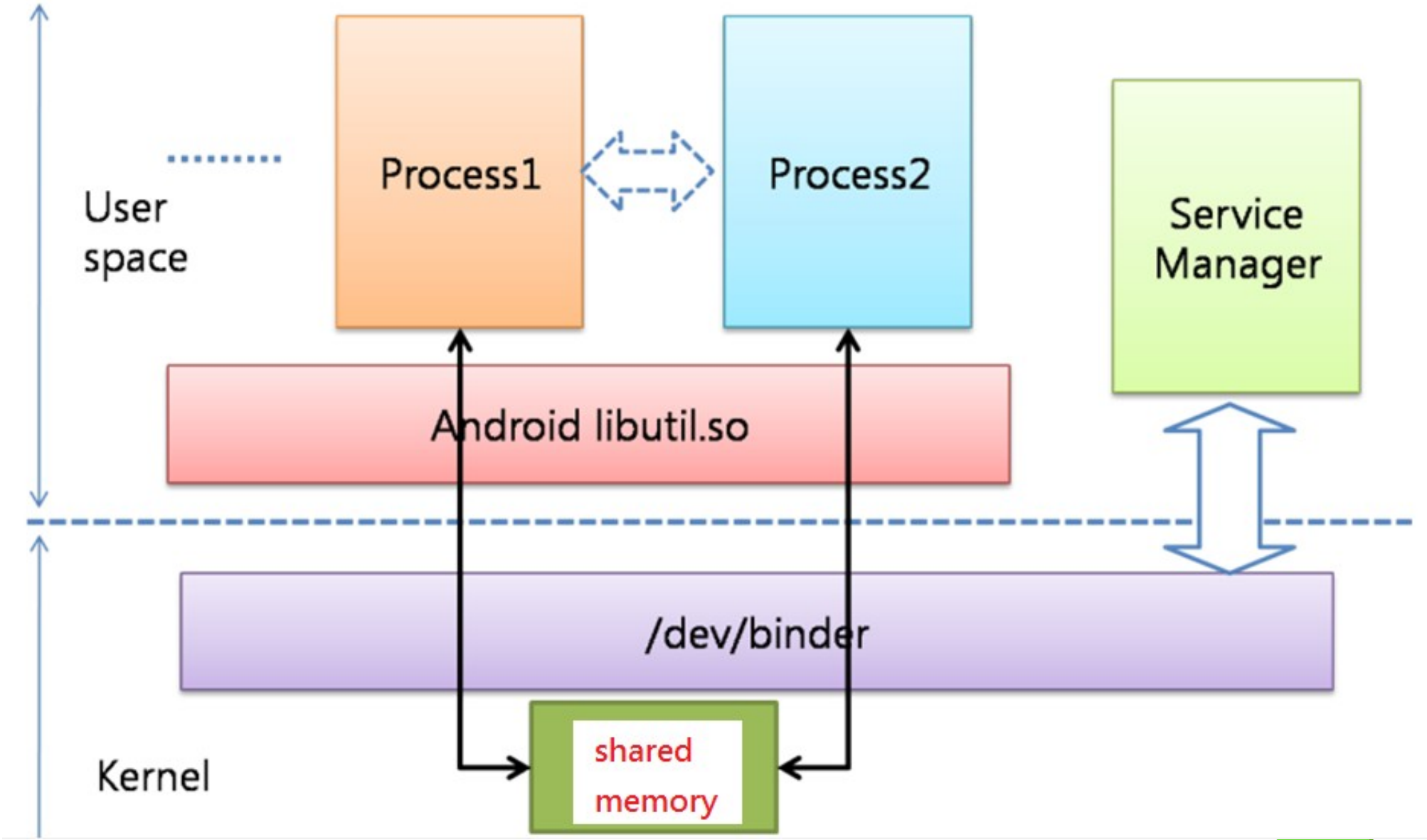


Binder



- ✓ A pool of threads is associated to each service application to process incoming IPC (Inter-Process Communication).
- ✓ Binder performs mapping of object between two processes.
- ✓ Binder uses an object reference as an address in a process's memory space.
- ✓ Synchronous call, reference couting

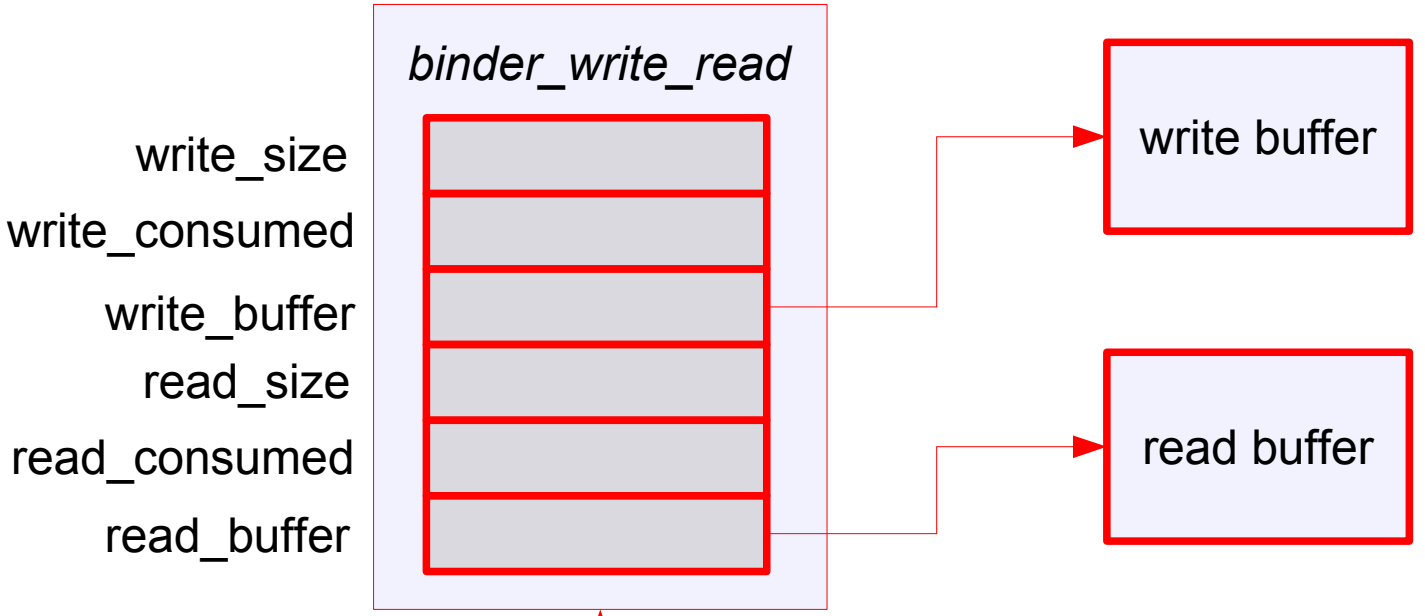
Binder


User space
Process1
Process2
Service Manager
Android libutil.so
/dev/binder
shared memory
Kernel

Binder is different from UNIX socket

	socket	binder
internal status	associated to FD	associated to PID (FD can be shared among threads in the same process)
read & write operation	stream I/O	done at once by **ioctl**
network transparency	Yes	No expected local only

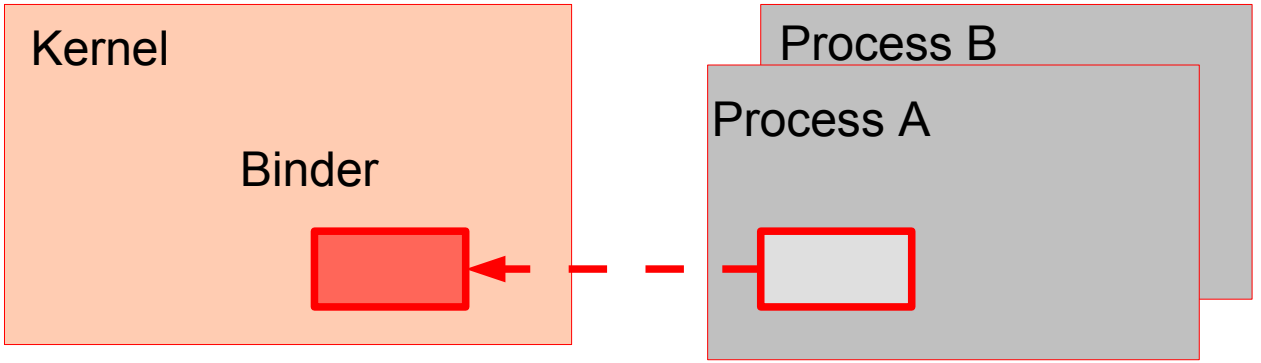
Transaction of Binder




```
if (ioctl(fd, BINDER_WRITE_READ, &bwt ) >= 0)
    err = NO_ERROR;
else
    err = -errno;
```

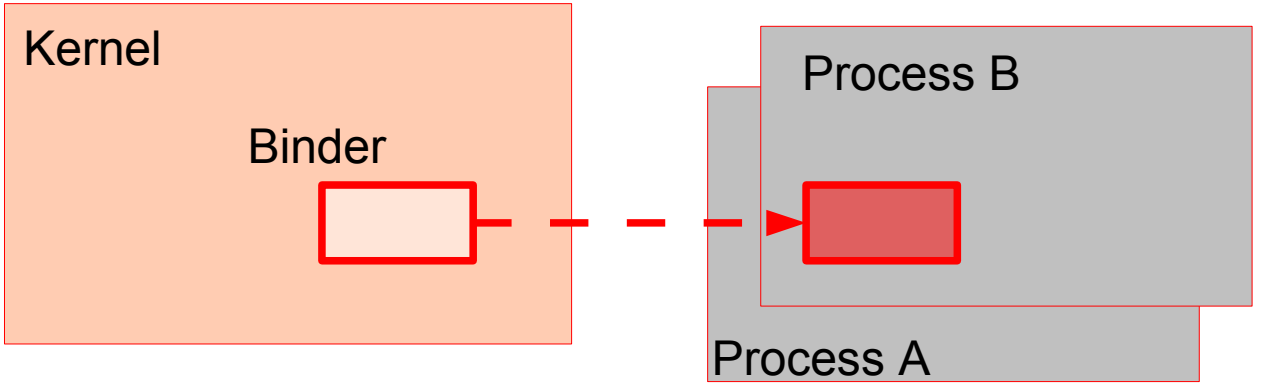
Transaction of Binder

Process A and B have different memory space.
They can not see each other.



Copy memory by **copy_from _user**

Then, wake up process B



Copy memory by **copy_to_user**

Internally, Android uses Binder for graphics data transaction across processes.
It is fairly efficient.

Binder sample program

- Build binder benchmark program

```
cd system/extras/tests/binder/benchmarks
```

```
mm
```

```
adb push \
  ../../../../out/target/product/crespo/data/nativebenchmark/binderAddInts \
  /data/local/
```

- Execute

```
adb shell
```

```
su
```

```
/data/local/binderAddInts -d 5 -n 5 &
```

```
ps
```

```
...
root      17133 16754 4568   860   ffffffff 400e6284 S
/data/local/binderAddInts
root      17135 17133 2520   616   00000000 400e5cb0 R
/data/local/binderAddInts
```

Binder sample program

- Execute

/data/local/binderAddInts -d 5 -n 5 &

ps

```
...
root      17133 16754 4568   860   ffffffff 400e6284 S
/data/local/binderAddInts
root      17135 17133 2520   616   00000000 400e5cb0 R
/data/local/binderAddInts
```

cat /sys/kernel/debug/binder/transaction_log

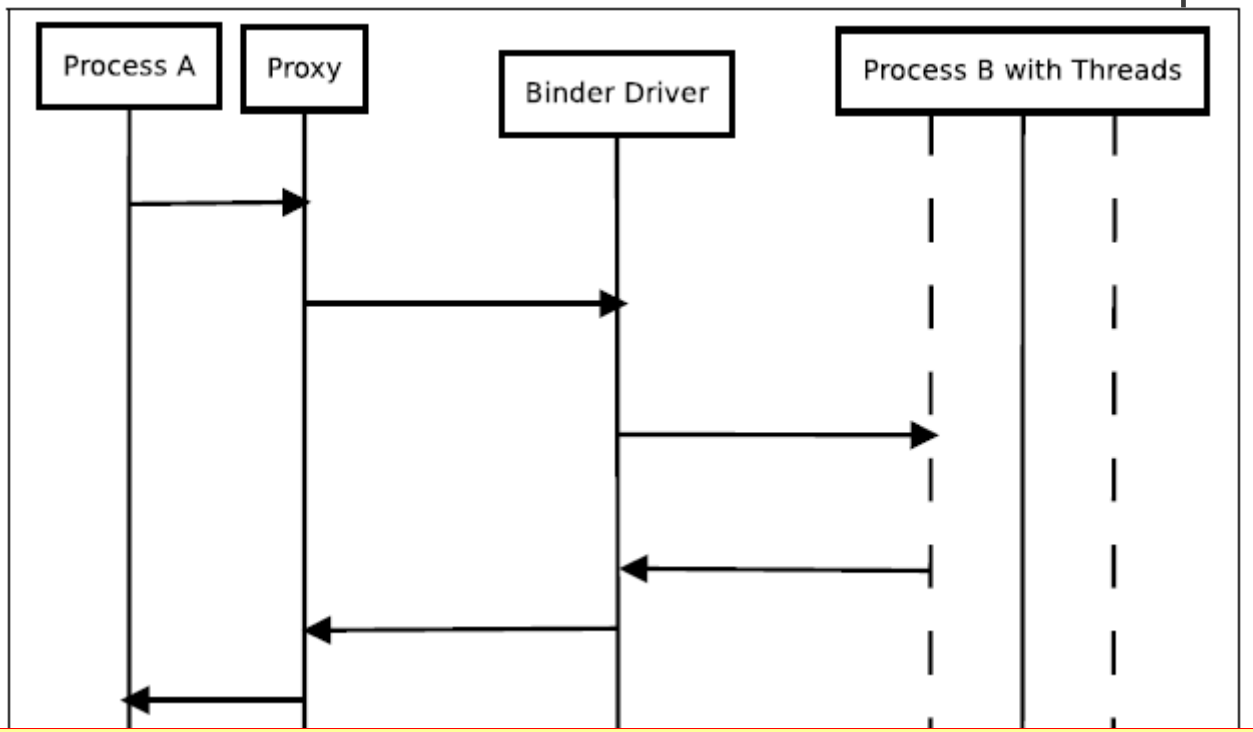
```
transaction_log:3439847: call  from 17133:17133 to 72:0 node
1 handle 0 size 124:4
transaction_log:3439850: reply from 72:72 to 17133:17133 node
0 handle 0 size 4:0
transaction_log:3439855: call  from 17135:17135 to 17133:0
node 3439848 handle 1 size 8:0
...
```

Binder sysfs entries

- **`adb shell ls /sys/kernel/debug/binder`**

```
failed_transaction_log
proc
state
stats
transaction_log
transactions
```

Communication protocol



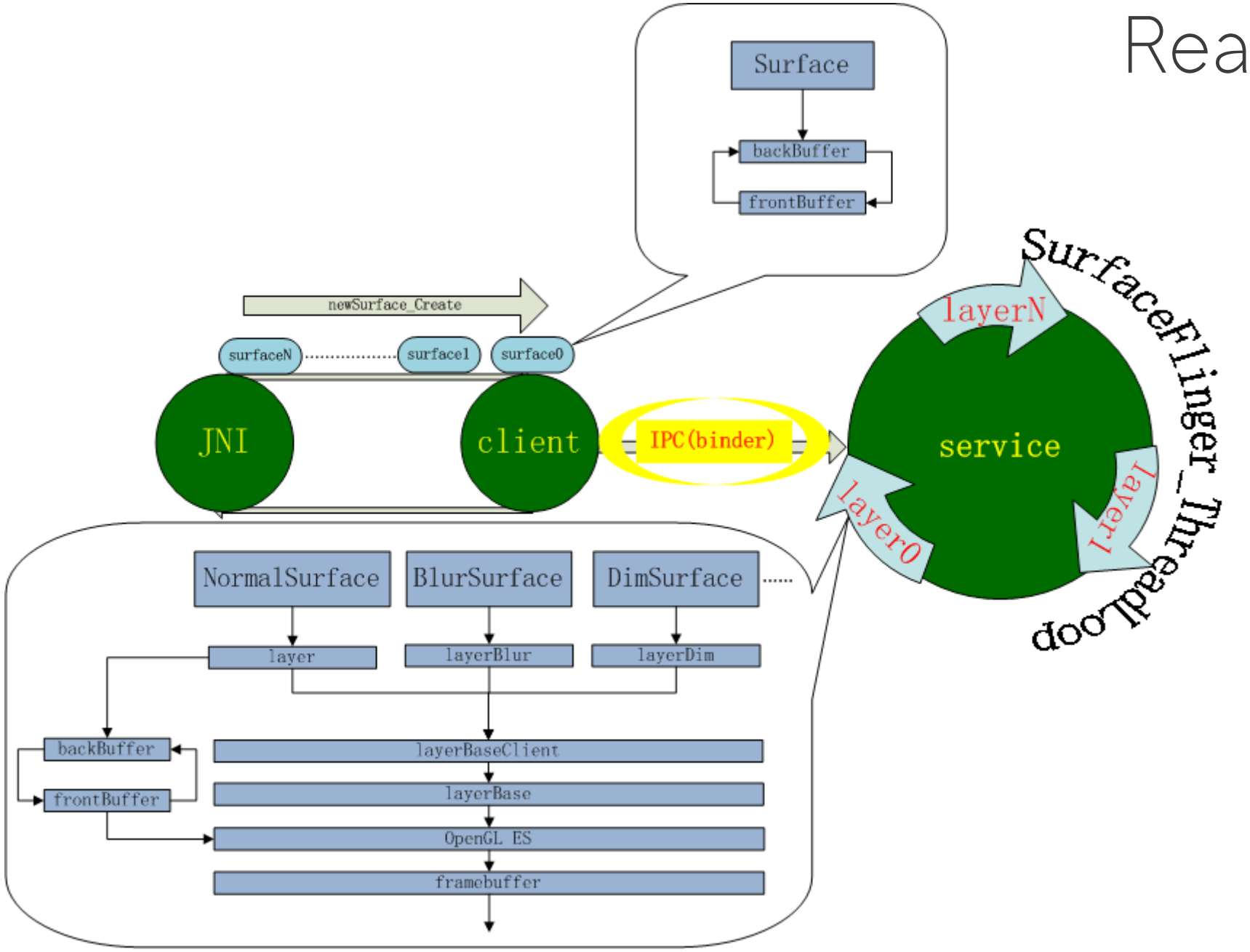
If one process sends data to another process, it is called transaction. The data is called transaction data.

Target	Binder Driver Command	Cookie	Sender ID	Data:

Target Command 0	Arguments 0
Target Command 1	Arguments 1
...	...
Target Command n-1	Arguments n-1

Binder use case: Android Graphics

Real Case



Binder IPC is used for communicating between Graphics client and server.
Taken from http://www.cnblogs.com/xl19862005/archive/2011/11/17/2215363.html

Surface

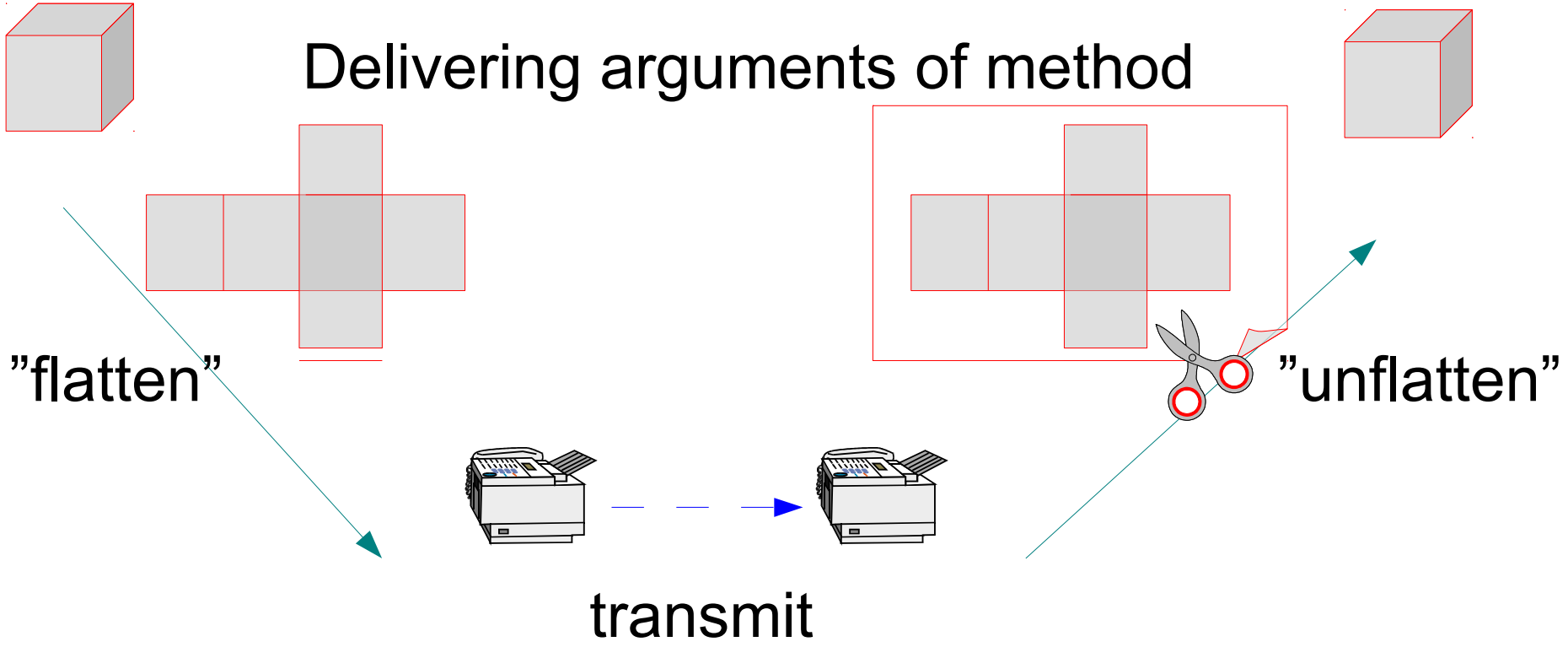
Source: frameworks/base/core/java/android/view/Surface.java

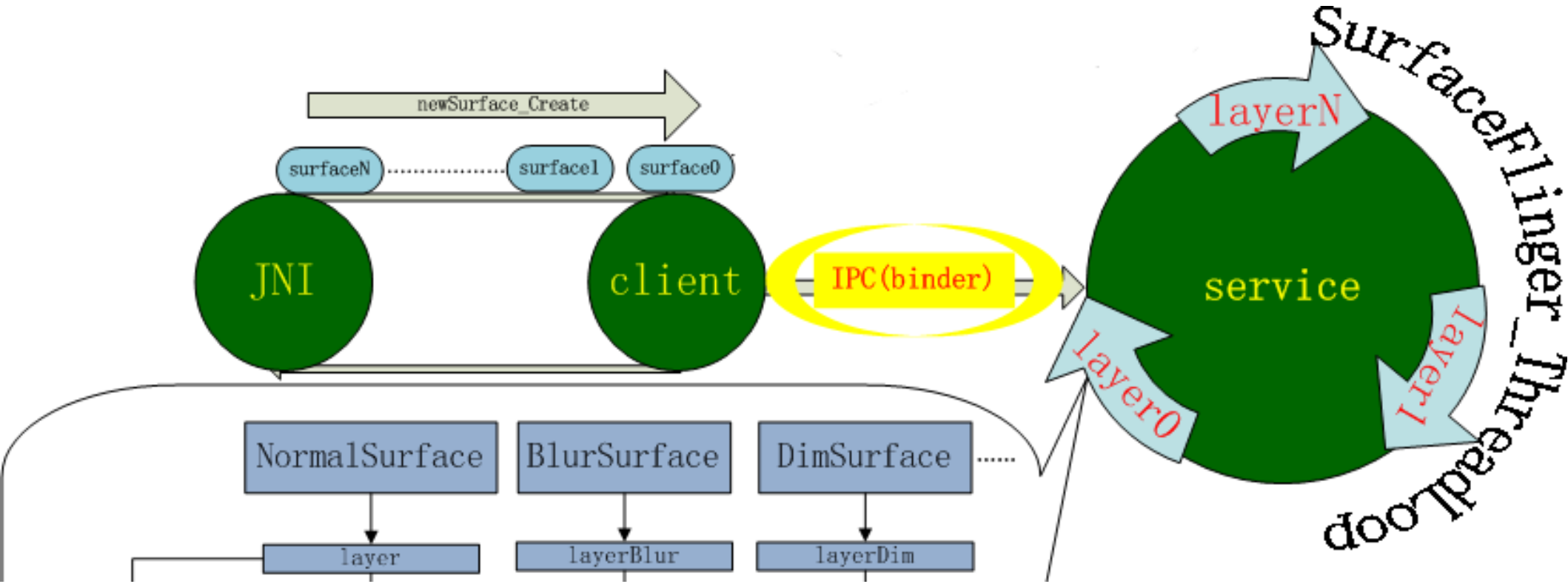
- /* **Handle on to a raw buffer that is being managed by the screen compositor** */

```
public class Surface implements Parcelable {
    public Surface() {
        mCanvas = new CompatibleCanvas();
    }
    private class CompatibleCanvas
                extends Canvas { /* ... */ }
}
```

Surface instances can be written to and restored from a Parcel.

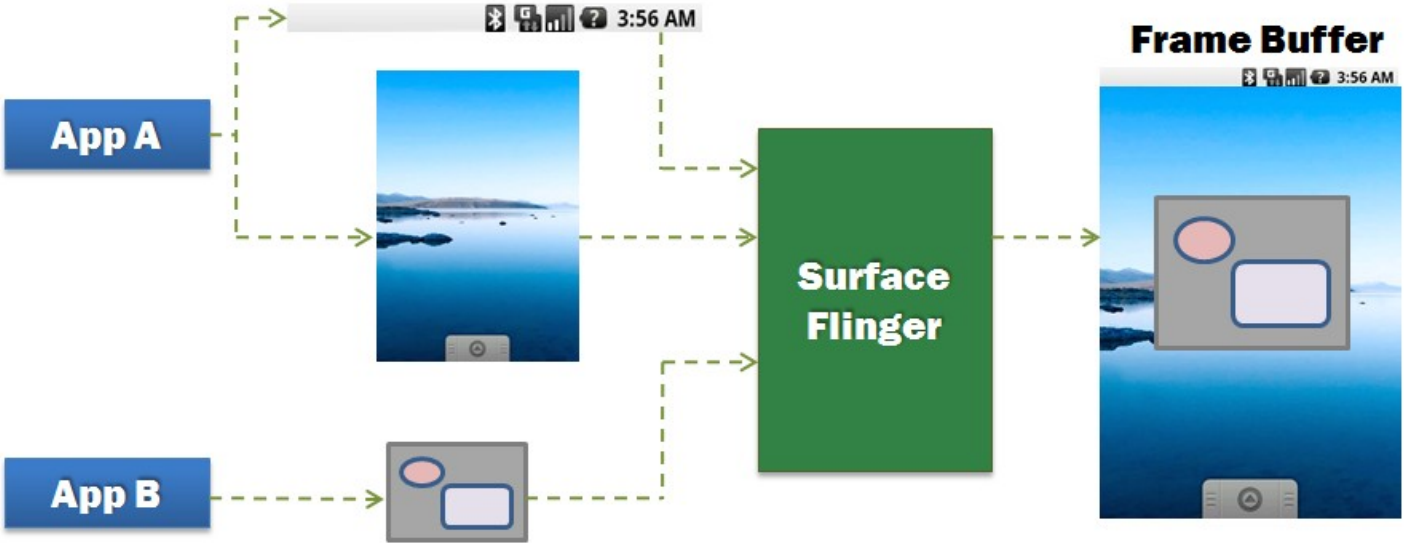
Delivering arguments of method

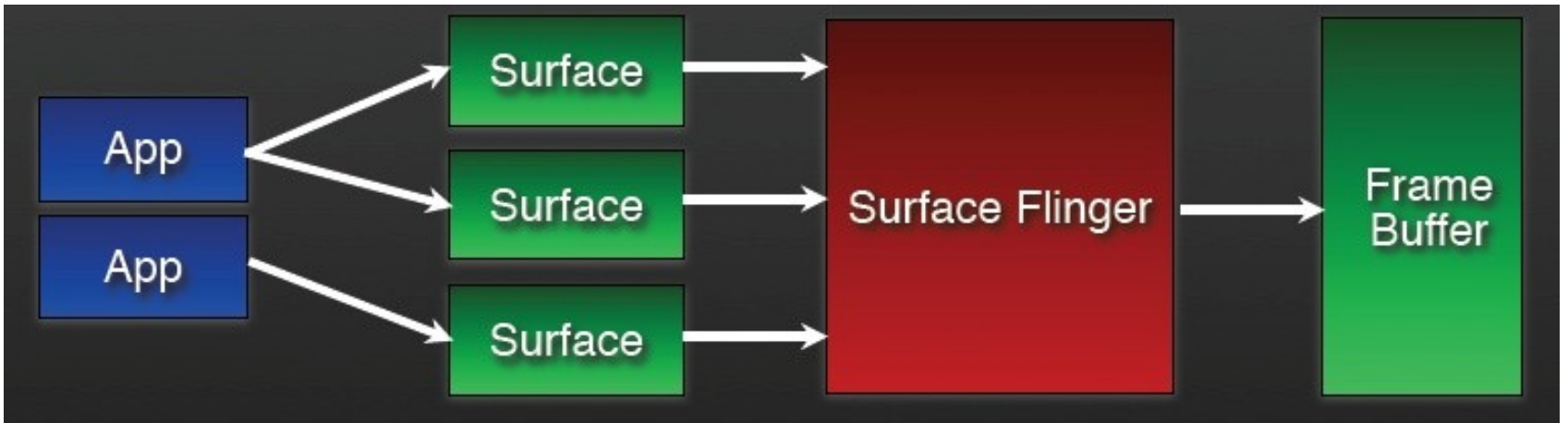


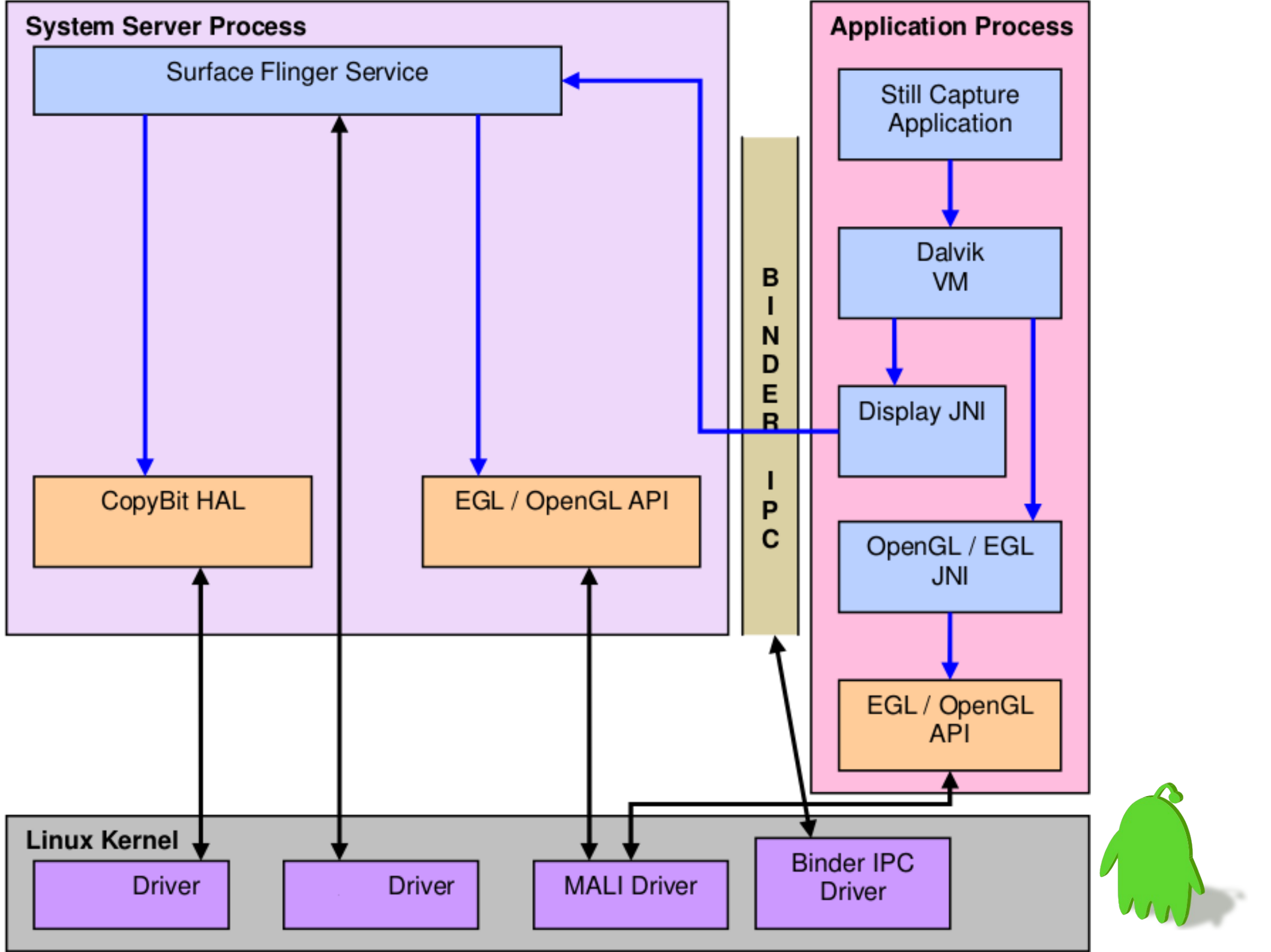


Android SurfaceFlinger

- Properties
  - Can combine 2D/3D surfaces and surfaces from multiple applications
  - Surfaces passed as buffers via Binder IPC calls
  - Can use OpenGL ES and 2D hardware accelerator for its compositions
    - Double-buffering using page-flip






System Server Process
Surface Flinger Service
CopyBit HAL
EGL / OpenGL API
BINDER IPC
Application Process
Still Capture Application
Dalvik VM
Display JNI
OpenGL / EGL JNI
EGL / OpenGL API
Linux Kernel
Driver
Driver
MALI Driver
Binder IPC Driver

Case Study

Android Power Management:

how to interact between system and framework

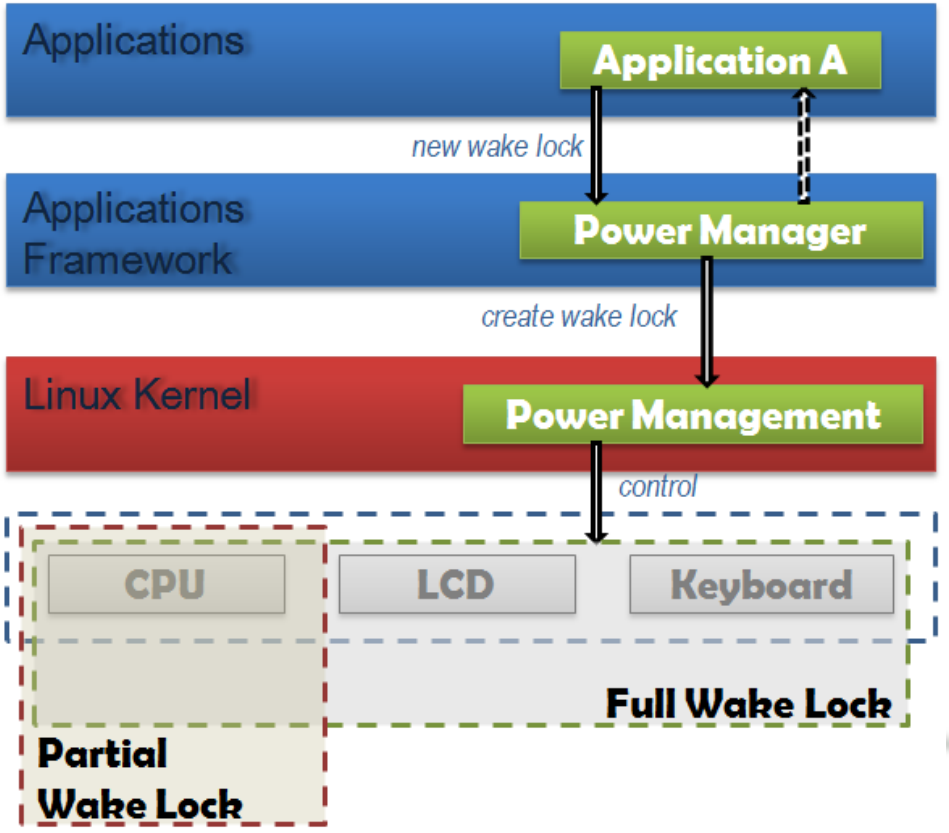
Base: Linux Kernel

- Android does rely on Linux Kernel for core system services
  - Memory/Process Management
  - Device Driver Model
  - sysfs, kobject/uevent, netlink
- Android Kernel extensions
  - Binder
  - android_power
    - /sys/android_power/, /sys/power/

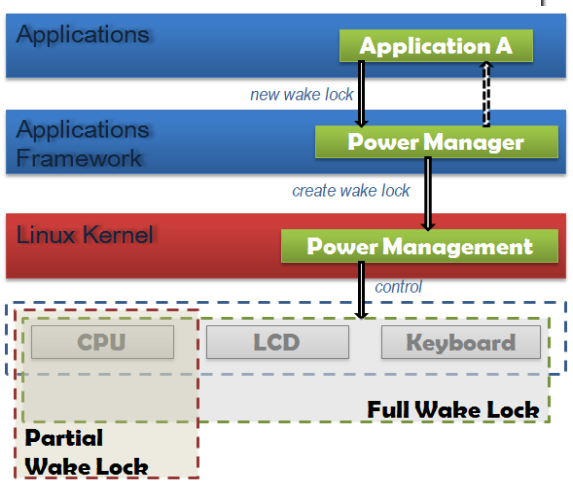
Key Idea: Android attempts to provide an abstraction layer between hardware and the related software stack.

Android's PM Concepts

- Android PM is built on top of standard Linux Power Management.
- It can support more aggressive PM, but looks fairly simple now.
- Components make requests to keep the power on through "**Wake Locks**".
  - PM does support several types of "Wake Locks".



- If there are no active wake locks, CPU will be turned off.
- If there is are partial wake locks, screen and keyboard will be turned off.



Applications

Application A

Application B

Application C

```
WI = newWakeLock(…);
WI.acquire();
WI.release();
```

Applications Framework

PowerManager
Android.os.PowerManager

Power
Android.os.Power

PowerManagerService
Andorid.server.PowerManagerService

JNI

Libraries (user space)

Core Libraries

X

Power
/lib/hardware/power.c

Linux Kernel

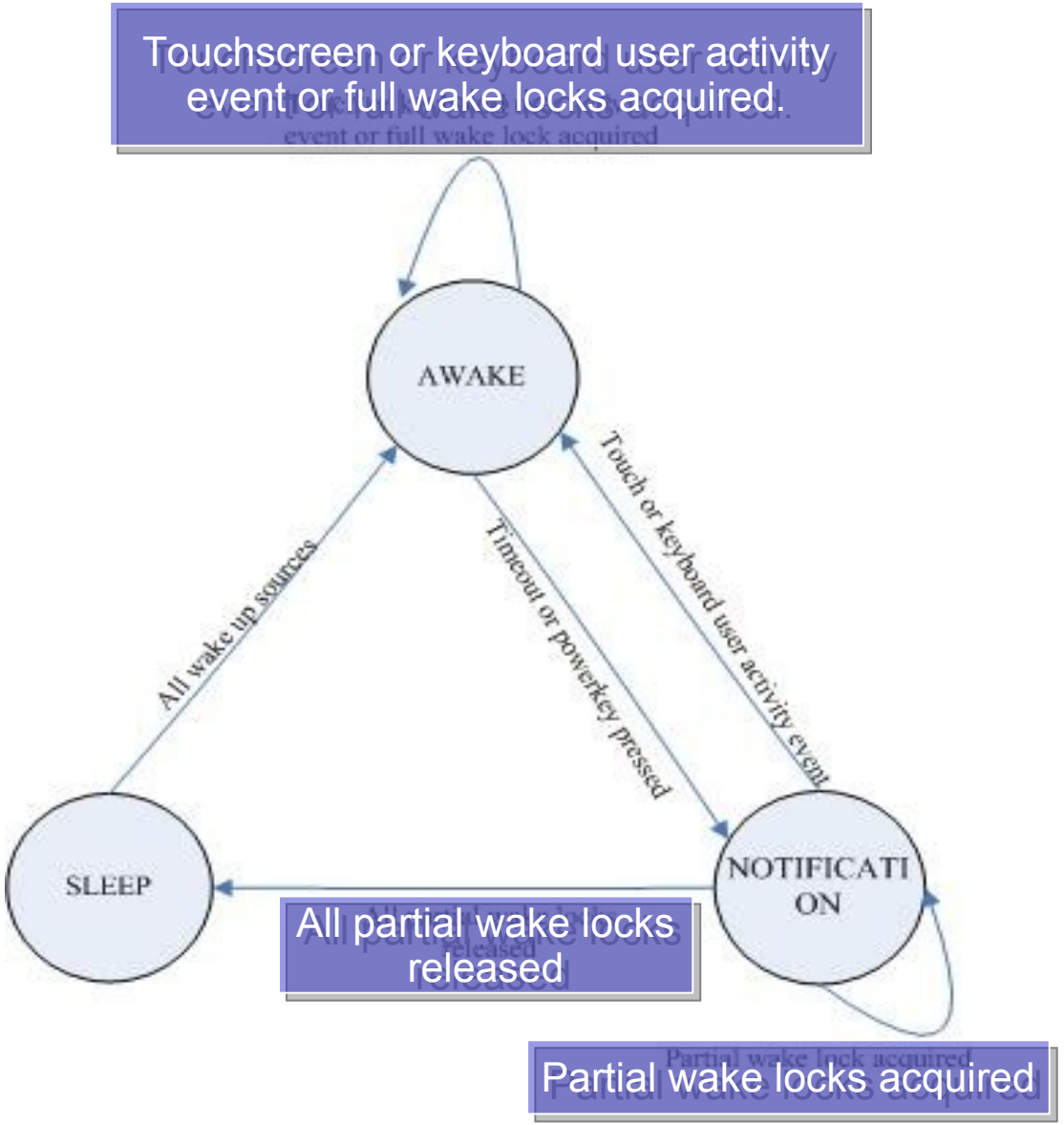
Linux Drivers

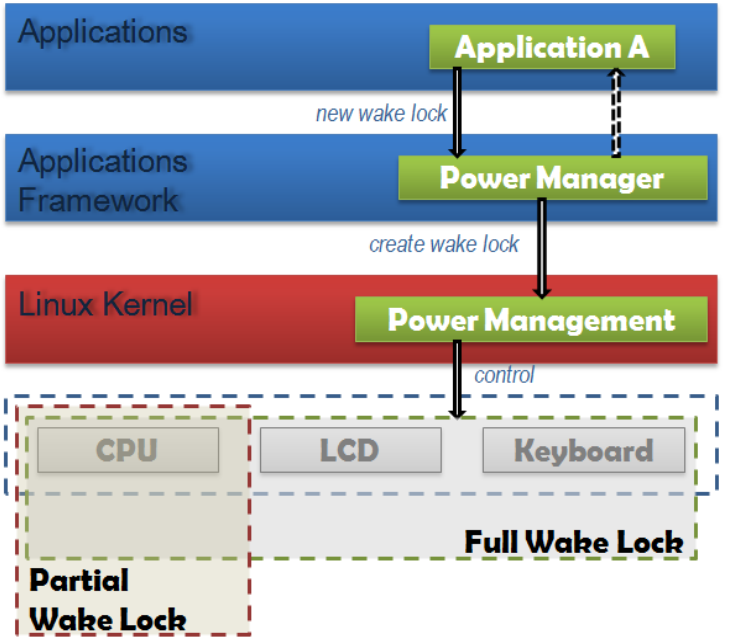
Android Power Management
/drivers/android/power.c

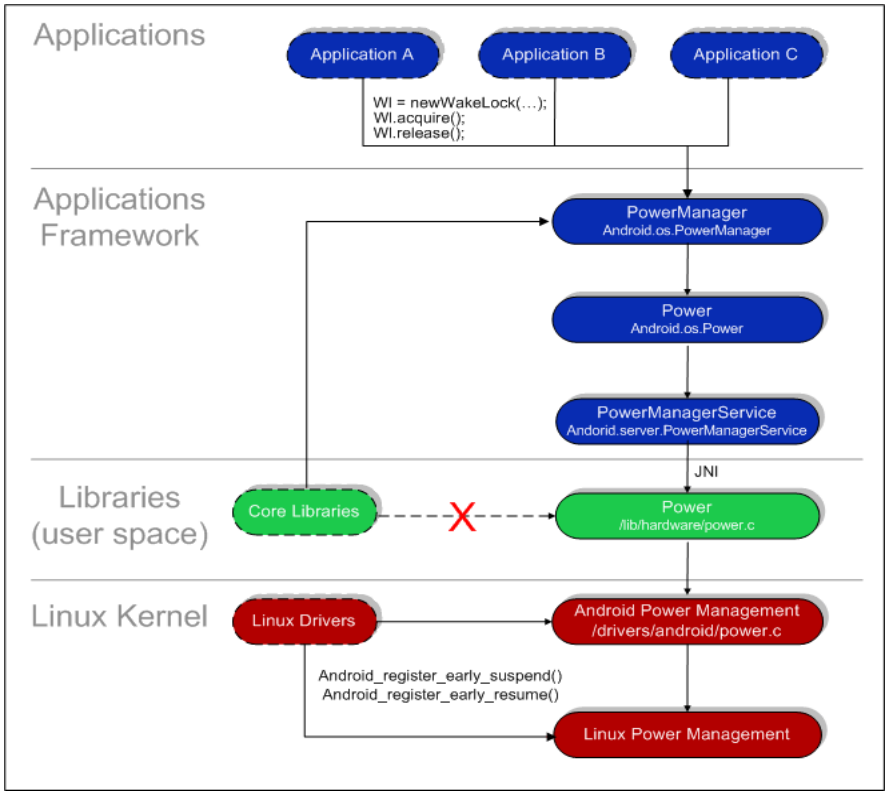
Android_register_early_suspend()
Android_register_early_resume()

Linux Power Management

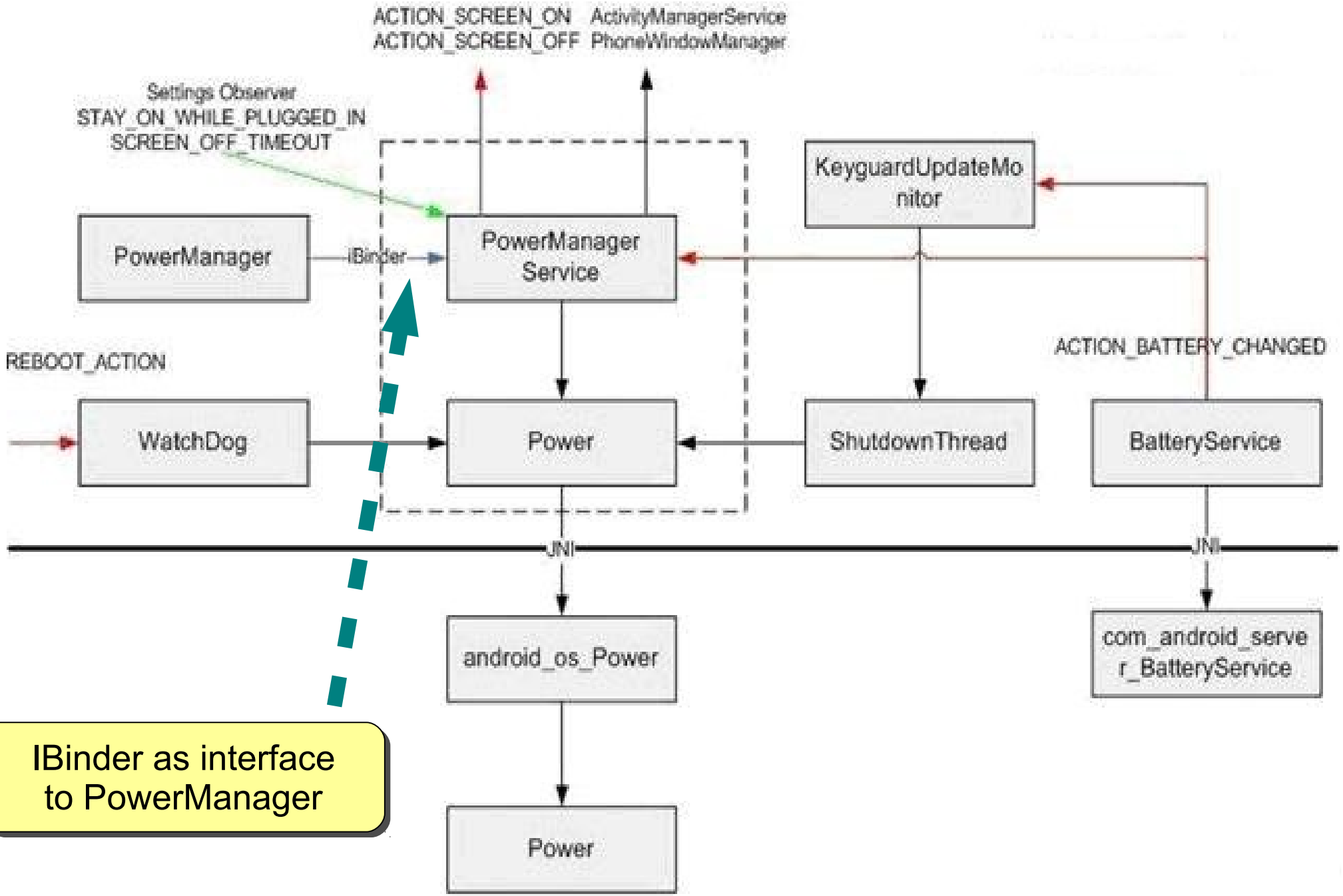
PM State Machine







Design and Implementation



Sample WakeLocks usage: AudioFlinger

- File frameworks/base/services/audioflinger/AudioFlinger.cpp

```
void AudioFlinger::ThreadBase::acquireWakeLock_l() {
    if (mPowerManager == 0) {
        sp<IBinder> binder =
            defaultServiceManager()->checkService(String16("power"));
        if (binder == 0) {
            LOGW("Thread %s can't connect to the PM service", mName);
        } else {
            mPowerManager = interface_cast<IPowerManager>(binder);
            binder->linkToDeath(mDeathRecipient);
        }
    }
    if (mPowerManager != 0) {
        sp<IBinder> binder = new BBinder();
        status_t status =
            mPowerManager->acquireWakeLock(POWERMANAGER_PARTIAL_WAKE_LOCK,
                                           binder, String16(mName));
        if (status == NO_ERROR) { mWakeLockToken = binder; }
        LOGV("acquireWakeLock_l() %s status %d", mName, status);
    }
}
```

android_os_Power

frameworks/base/core/jni/android_os_Power.cpp

```
...
static JNINativeMethod method_table[] = {
    { "acquireWakeLock", "(ILjava/lang/String;)V", (void*)acquireWakeLock },
    { "releaseWakeLock", "(Ljava/lang/String;)V", (void*)releaseWakeLock },
    { "setLastUserActivityTimeout", "(J)I", (void*)setLastUserActivityTimeout },
    { "setLightBrightness", "(II)I", (void*)setLightBrightness },
    { "setScreenState", "(Z)I", (void*)setScreenState },
    { "shutdown", "()V", (void*)android_os_Power_shutdown },
    { "reboot", "(Ljava/lang/String;)V", (void*)android_os_Power_reboot },
};

int register_android_os_Power(JNIEnv *env)
{
    return AndroidRuntime::registerNativeMethods(
        env, "android/os/Power",
        method_table, NELEM(method_table));
}
```

Settings Observe
STAY_ON_WHILE_PLU
SCREEN_OFF_TIME

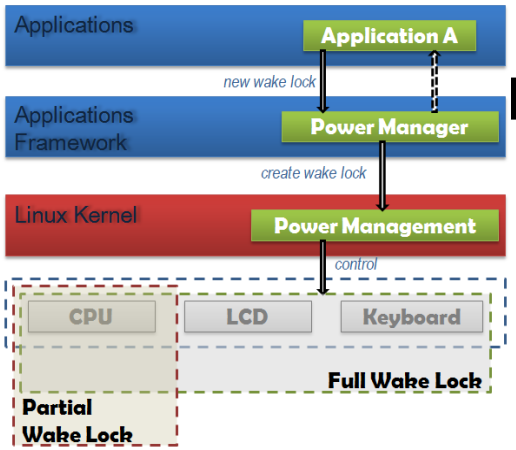
PowerManager

REBOOT_ACTION

WatchDog

JNI

JNI



android_os_Power

Power

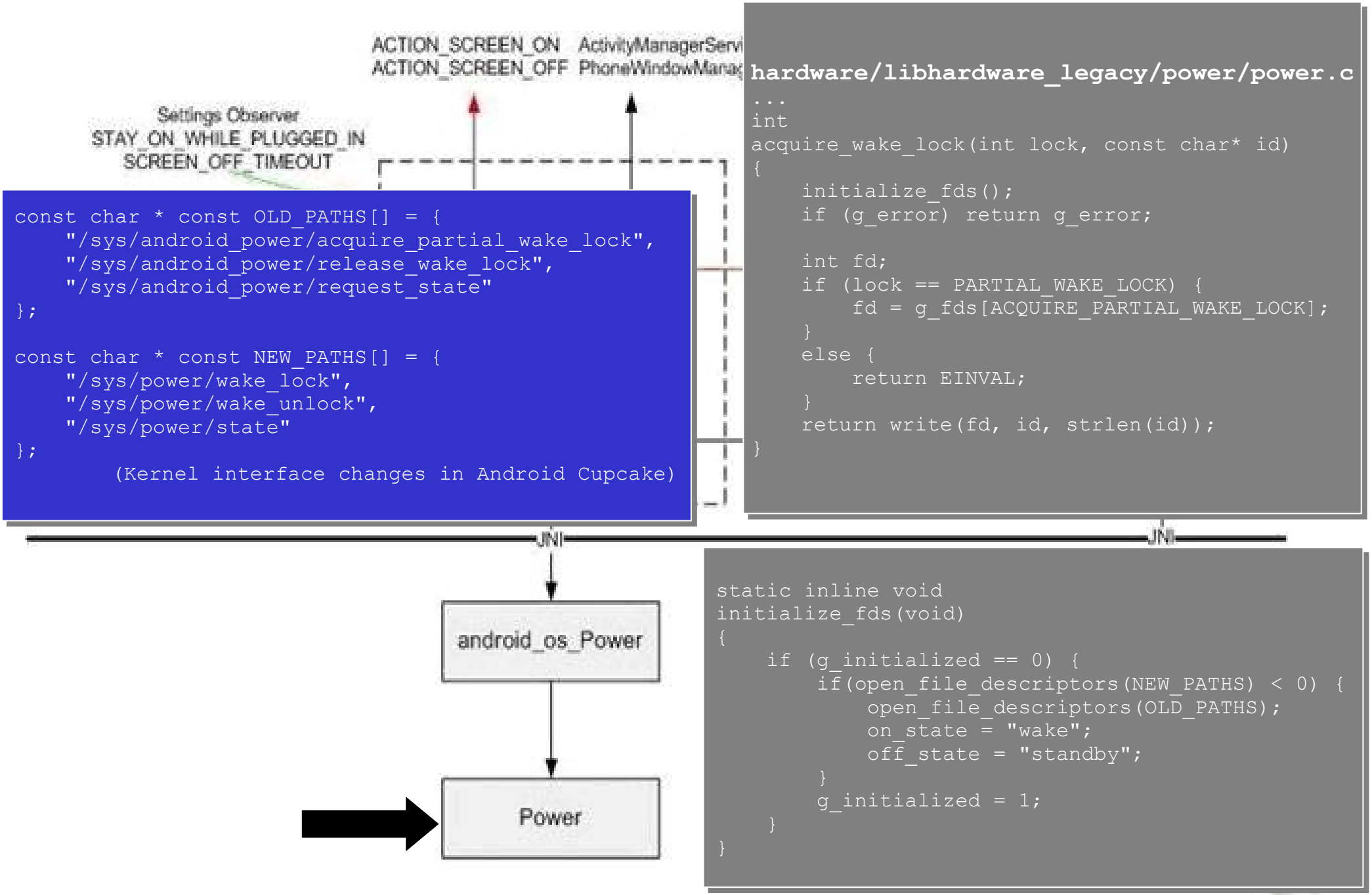
```
static void
acquireWakeLock(JNIEnv *env, jobject clazz,
                jint lock, jstring idObj)
{
    if (idObj == NULL) {
        throw_NullPointerException(env, "id is null");
        return ;
    }

    const char *id = env->GetStringUTFChars(idObj, NULL);

    acquire_wake_lock(lock, id);

    env->ReleaseStringUTFChars(idObj, id);
}
```

Power




```
const char * const OLD_PATHS[] = {
    "/sys/android_power/acquire_partial_wake_lock",
    "/sys/android_power/release_wake_lock",
    "/sys/android_power/request_state"
};

const char * const NEW_PATHS[] = {
    "/sys/power/wake_lock",
    "/sys/power/wake_unlock",
    "/sys/power/state"
};
        (Kernel interface changes in Android Cupcake)
```

hardware/libhardware_legacy/power/power.c

```
...
int
acquire_wake_lock(int lock, const char* id)
{
    initialize_fds();
    if (g_error) return g_error;

    int fd;
    if (lock == PARTIAL_WAKE_LOCK) {
        fd = g_fds[ACQUIRE_PARTIAL_WAKE_LOCK];
    }
    else {
        return EINVAL;
    }
    return write(fd, id, strlen(id));
}
```

```
static inline void
initialize_fds(void)
{
    if (g_initialized == 0) {
        if(open_file_descriptors(NEW_PATHS) < 0) {
            open_file_descriptors(OLD_PATHS);
            on_state = "wake";
            off_state = "standby";
        }
        g_initialized = 1;
    }
}
```

Android PM Kernel APIs

- Source code
  - kernel/power/userwake.c
  - /kernel/power/wakelock.c

```
static int power_suspend_late(
     struct platform_device *pdev,
     pm_message_t state)
{
     int ret =
          has_wake_lock(WAKE_LOCK_SUSPEND) ?
          -EAGAIN : 0;
     return ret;
}

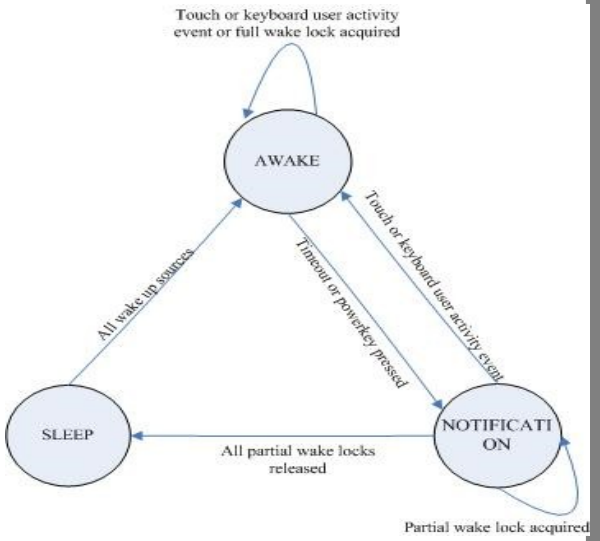
static struct platform_driver power_driver = {
     .driver.name = "power",
     .suspend_late = power_suspend_late,
};
static struct platform_device power_device = {
     .name = "power",
};
```

```
static long has_wake_lock_locked(int type)
{
     struct wake_lock *lock, *n;
     long max_timeout = 0;
     BUG_ON(type >= WAKE_LOCK_TYPE_COUNT);
     list_for_each_entry_safe(lock, n,
                &active_wake_locks[type], link) {
          if (lock->flags & WAKE_LOCK_AUTO_EXPIRE) {
               long timeout = lock->expires - jiffies;
               if (timeout <= 0)
                    expire_wake_lock(lock);
               else if (timeout > max_timeout)
                    max_timeout = timeout;
          } else
               return -1;
     }
     return max_timeout;
}

long has_wake_lock(int type)
{
     long ret;
     unsigned long irqflags;
     spin_lock_irqsave(&list_lock, irqflags);
     ret = has_wake_lock_locked(type);
     spin_unlock_irqrestore(&list_lock, irqflags);
     return ret;
}
```

Android PM Kernel APIs

- kernel/power/wakelock.c




```
static int __init wakelocks_init(void)
{
     int ret;
     int i;

     for (i = 0; i < ARRAY_SIZE(active_wake_locks); i++)
          INIT_LIST_HEAD(&active_wake_locks[i]);


     wake_lock_init(&main_wake_lock, WAKE_LOCK_SUSPEND, "main");
     wake_lock(&main_wake_lock);
     wake_lock_init(&unknown_wakeup, WAKE_LOCK_SUSPEND, "unknown_wakeups");

     ret = platform_device_register(&power_device);
     if (ret) {
          pr_err("wakelocks_init: platform_device_register failed\n");
          goto err_platform_device_register;
     }
     ret = platform_driver_register(&power_driver);
     if (ret) {
          pr_err("wakelocks_init: platform_driver_register failed\n");
          goto err_platform_driver_register;
     }

     suspend_work_queue = create_singlethread_workqueue("suspend");
     if (suspend_work_queue == NULL) {
          ret = -ENOMEM;
          goto err_suspend_work_queue;
     }
```

Review

- Native area
  - dynamic linking
  - Processes
  - Memory layout
  - Binder IPC
  - interactions with frameworks

http://0xlab.org