Android Mobile OS

Corso di Laurea in Ingegneria Informatica

Università degli Studi di Napoli Federico II

Anno Accademico 2024/2025, Canale San Giovanni

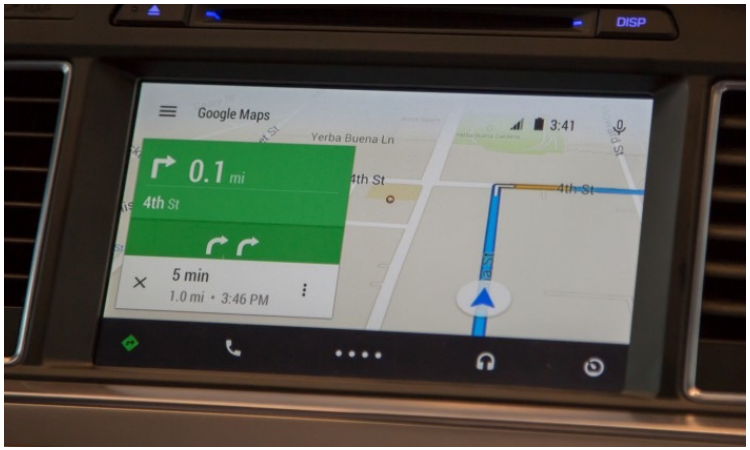
Indice e riferimenti

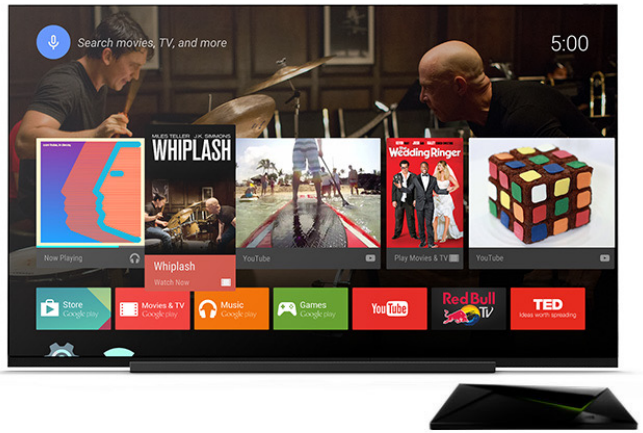
- Argomenti:
  - Storia e principi di progettazione di Android
  - Applicazioni e processi in Android
  - Inter-Process Communication in Android

- Riferimenti:
  - A. Tanenbaum, H. Bos, "*Modern Operating Systems*," Pearson ed., 4ª edizione, 2015 (**Capitolo 10: Unix, Linux, and Android**)

Android

- Android è un popolare SO usato in
  - dispositivi mobili (smartphones, tablets)
  - prodotti consumer (TV, media players, smartwatches, ...)
  - sistemi embedded (cruscotti, navigatori, domotica, ...)



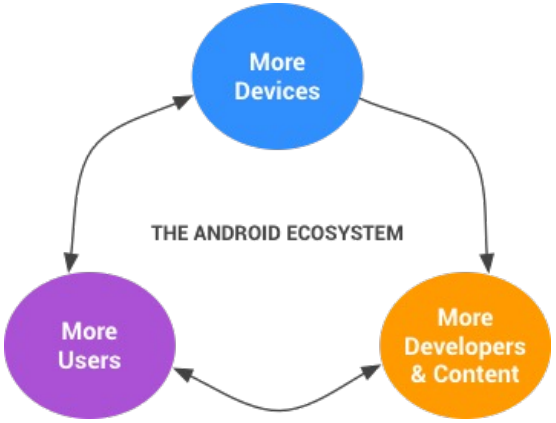


Android

- Android è basato sul kernel Linux e altro software open-source
- Utilizza i meccanismi tradizionali di UNIX (processi, utenti, IPC, memoria virtuale, ...), in modo innovativo

Google e Android

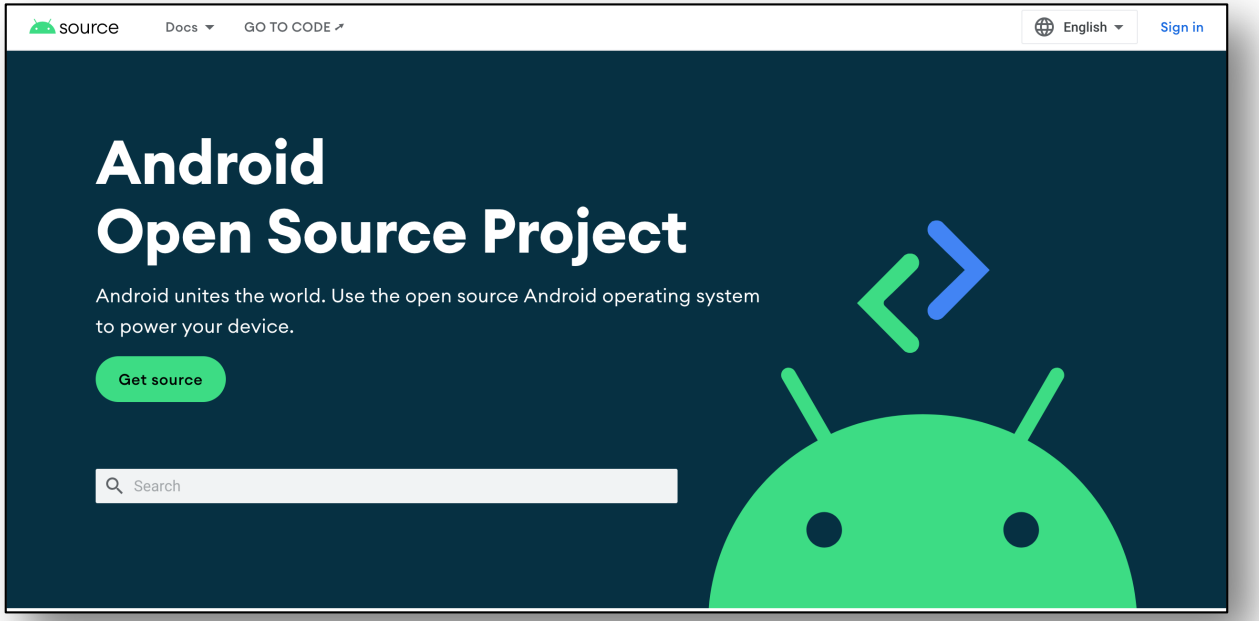
- Google acquisisce Android Inc. nel 2005, guidandone lo sviluppo fino al rilascio di Android 1.0 nel 2007
- Android mira a creare un **ecosistema** che coinvolga **sviluppatori** e **produttori** al di fuori di Google



Google e Android

Google e Android

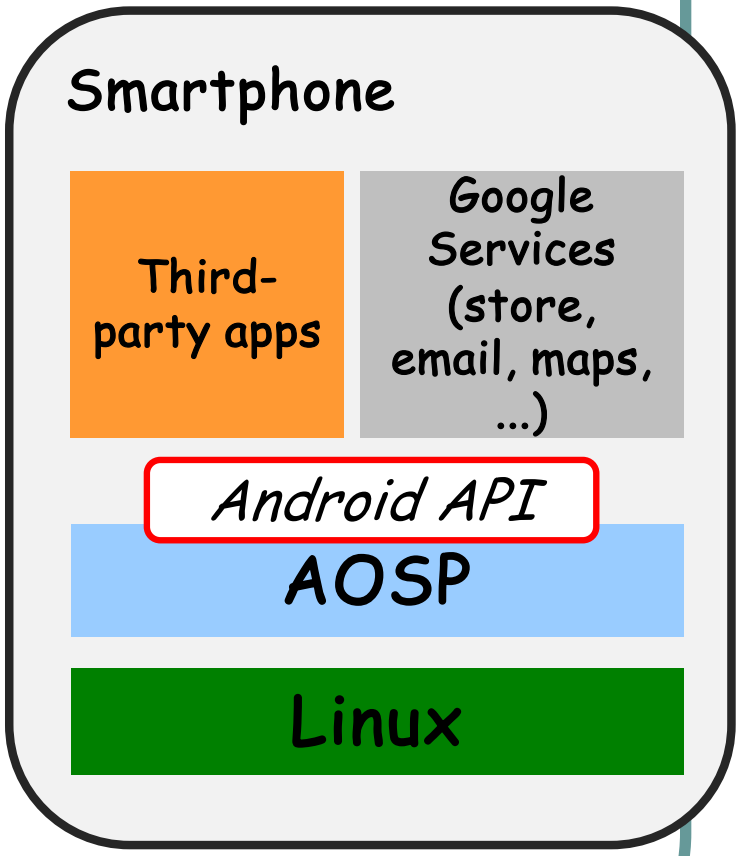
- Google continua a guidare lo sviluppo di Android:
  - Rilasciando le **specifiche** e i **test** di **compatibilità** per Android
  - Contribuendo alla **piattaforma open-source** e ai **kit di sviluppo**
  - Sviluppando uno **app store** e i propri **servizi commerciali** per Android



https://source.android.com/

Android Open-Source Project

- La versione open-source di Android (**AOSP**) è una collezione di software e tool
- Fornisce interfacce di programmazione (**API**) per app
- AOSP è **neutrale** rispetto alle applicazioni commerciali (di Google o di altri)



Epic Games vs. Google

Epic win: Jury decides Google has illegal monopoly in app store fight

Illustration by Cath Virginia / The Verge

/ The jury decided Google's sweetheart deals were too much.

By Sean Hollister, a senior editor and founding member of The Verge who covers gadgets, games, and toys. He spent 15 years editing the likes of CNET, Gizmodo, and Engadget.

Updated Dec 12, 2023, 2:25 AM GMT+1 | 320 Comments / 320 New

If you buy something from a Verge link, Vox Media may earn a commission. ***See our ethics statement.***

https://www.theverge.com/23994174/epic-google-trial-jury-verdict-monopoly-google-play

Versioni di Android

Android 1.5

Cupcake

Android 1.6

Donut

Android 2.0/2.1

Eclair

Android 2.2.x

Froyo

Android 2.3.x

Gingerbread

Android 3.x

Honeycomb

Android 4.0.x

Ice Cream Sandwich

Android 4.1.x

Jelly Bean

Android 4.4.x

KitKat

Android 5.0

Lollipop

Android 6.0

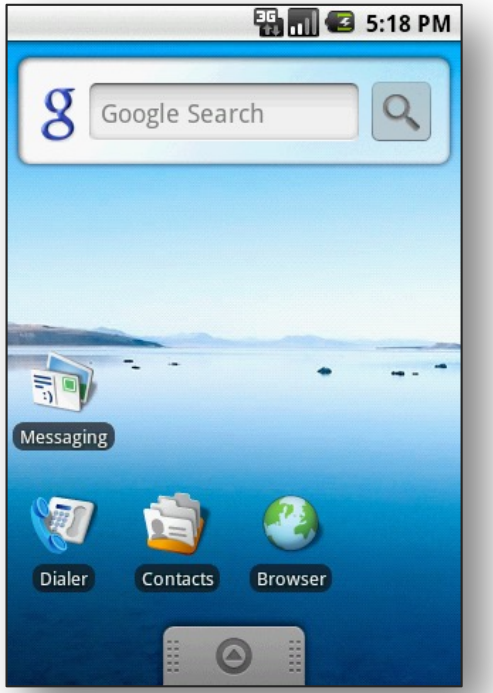
Marshmallow

Android 7.0

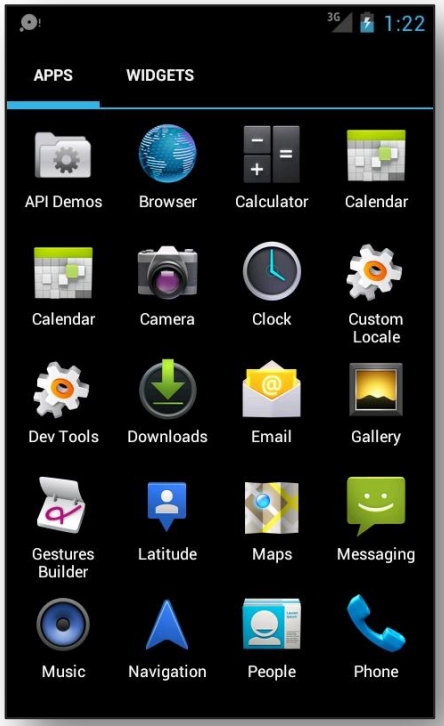
Nougat

https://en.wikipedia.org/wiki/Android_version_history

Versioni di Android



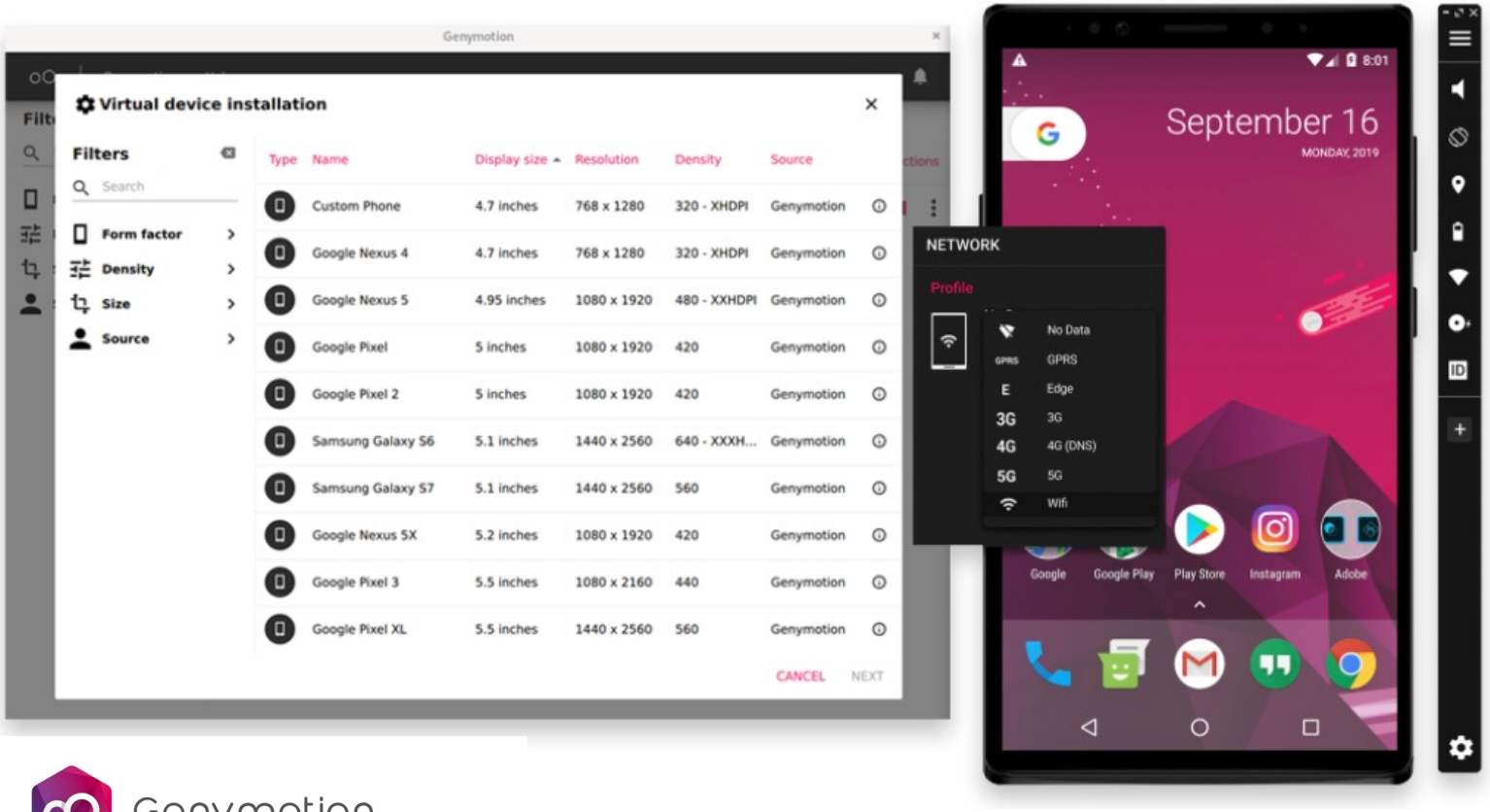
Cupcake



Jelly Bean

Nougat

Android Emulator



Genymotion Desktop

Local virtual devices with high performances.

- ✓ Emulate 3000+ virtual Android device configurations (Android versions, screen size, hardware capacities, etc.)
- ✓ Simulate every imaginable scenarios thanks to our full set of hardware sensors (GPS, network, multitouch, etc.)

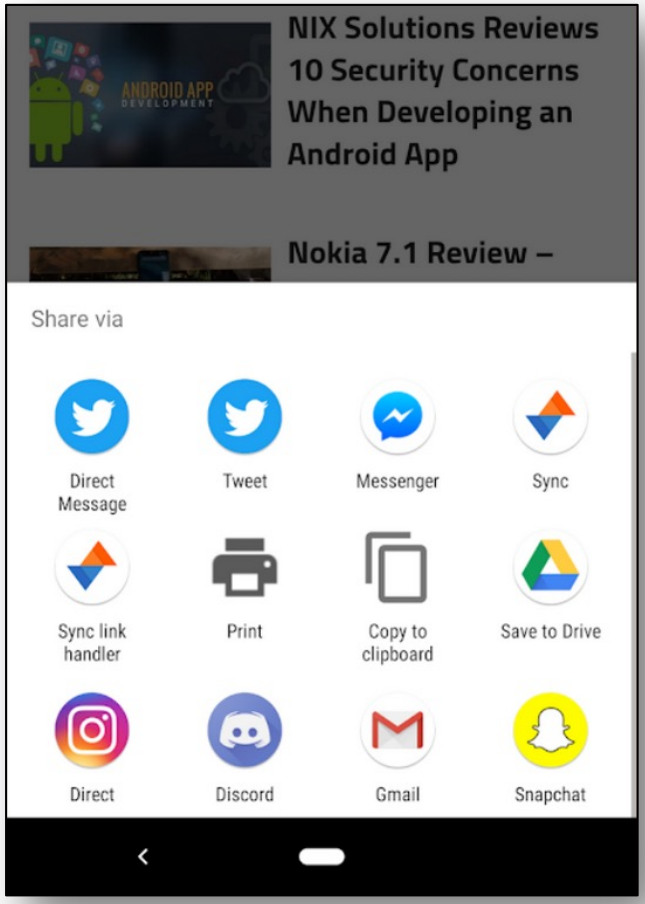
https://www.genymotion.com/

Obiettivi di progetto di Android OS

- Ottimizzato per interazioni brevi, con avvio e switch veloce delle app (avvio delle app di base in al più 200ms!)

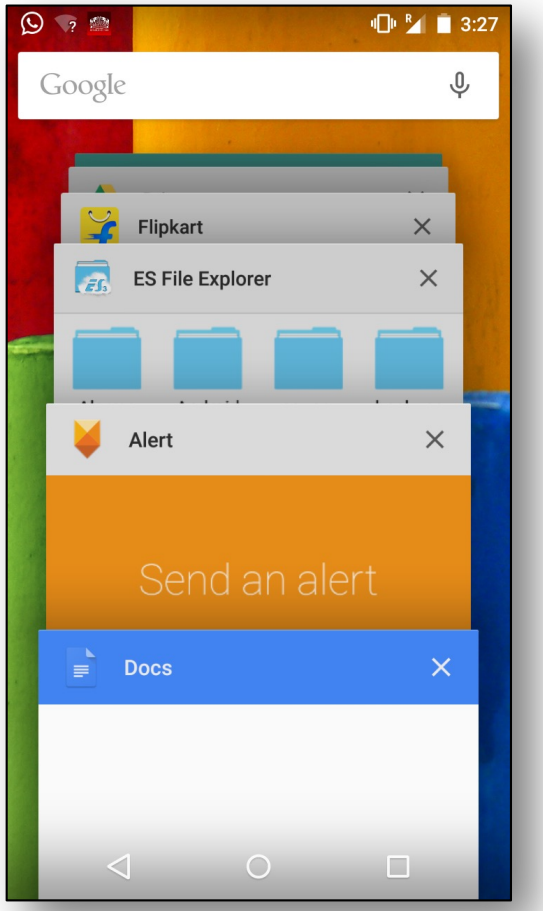
Obiettivi di progetto di Android OS

- Ottimizzato per interazioni brevi, con avvio e switch veloce delle app (avvio delle app di base in al più 200ms!)
- Non più applicazioni desktop "monolitiche", ma applicazioni piccole e in grado di interagire



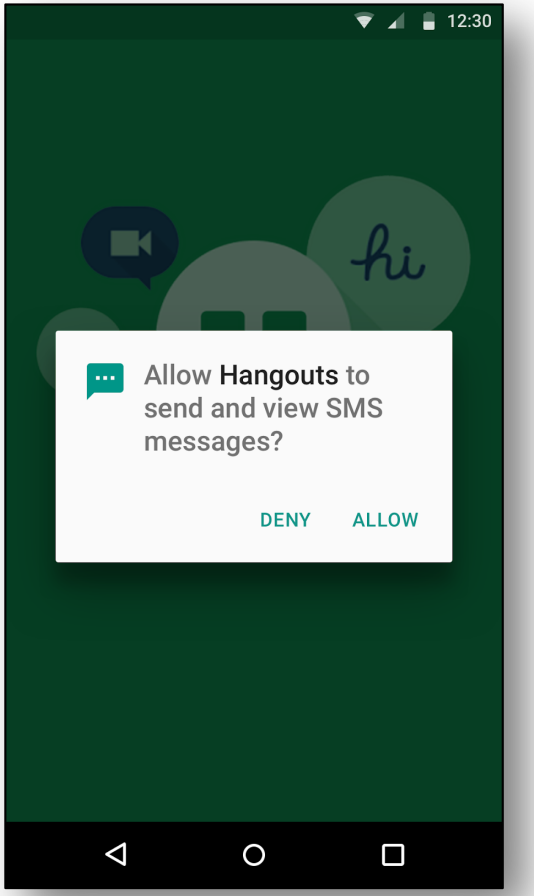
Obiettivi di progetto di Android OS

- Ottimizzato per interazioni brevi, con avvio e switch veloce delle app (avvio delle app di base in al più 200ms!)
- Non più applicazioni desktop "monolitiche", ma applicazioni piccole e in grado di interagire
- Gestione automatica della memoria e del ciclo di vita delle app



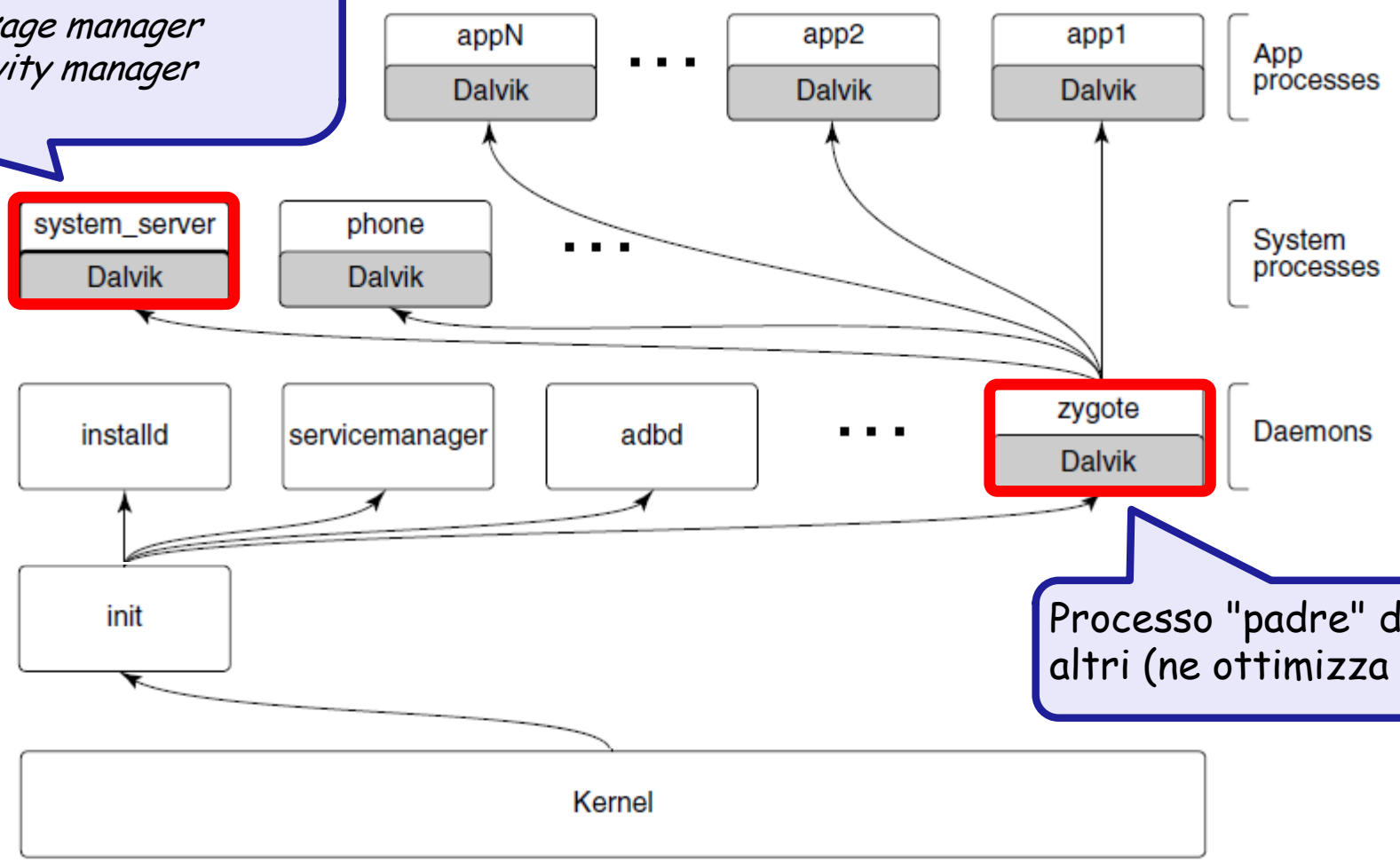
Obiettivi di progetto di Android OS

- Ottimizzato per interazioni brevi, con avvio e switch veloce delle app (avvio delle app di base in al più 200ms!)
- Non più applicazioni desktop "monolitiche", ma applicazioni piccole e in grado di interagire
- Gestione automatica della memoria e del ciclo di vita delle app
- Modello di sicurezza che consenta di eseguire app anche non "fidate"



Gerarchia dei processi in Android

Processo che esegue vari servizi di sistema:
- *package manager*
- *activity manager*
- *...*



Processo "padre" di tutti gli altri (ne ottimizza l'avvio)

Principali componenti di sistema

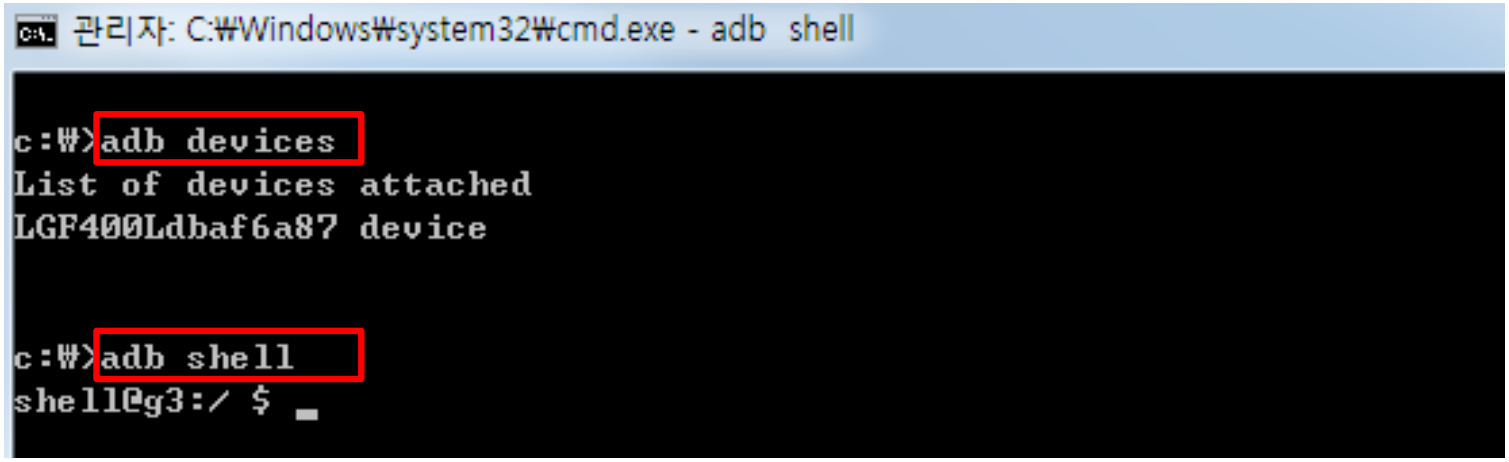
- **init**: gestisce il boot, avviando demoni di basso livello (tra cui *zygote*)
- **adbd**: crea processi *shell* per connessioni remote (es. USB)
- **zygote**: il padre di tutti i processi a più alto livello, basati sul linguaggio Java (usato dalla maggior parte del codice di Android)
- **Dalvik**: l'ambiente di esecuzione del linguaggio Java (oggi rimpiazzato da **ART**)

Principali componenti di sistema

- **System Server**: esegue molti servizi di sistema, quali:
  - *package manager*
  - *activity manager*
  - *power manager*
  - *...*
- **Altri processi di sistema**: Service Manager, RILD, NETD, VOLD, MediaServer, …
- **Android Framework**: insieme di *classi Java*, usate dalle app come *interfaccia API* ad Android OS

ADB shell

- È possibile collegare al PC via USB un dispositivo Android, e accedere alla shell via ADB
  - Copiare files, accedere al filesystem
  - Installare apps
  - Accedere ai log di sistema
  - ...



https://www.xda-developers.com/install-adb-windows-macos-linux/

Android storage

```
$ adb shell

root@myphone:/ # ls -l
drwxr-xr-x root     root              2021-12-13 05:37 acct
drwxrwx--- system   cache             2021-12-13 05:35 cache
lrwxrwxrwx root     root              1969-12-31 19:00 charger -> /sbin/healthd
dr-x------ root     root              2021-12-13 05:37 config
lrwxrwxrwx root     root              2021-12-13 05:37 d -> /sys/kernel/debug
drwxrwx--x system   system            2021-12-13 05:37 data
...

root@myphone:/ # mount
...
rootfs / rootfs ro,size=1023972k,nr_inodes=255993 0 0
/dev/block/sda6 /system ext4 ro,noatime,data=ordered 0 0
/dev/block/sdb1 /cache ext4 rw,nosuid,nodev,noatime,data=ordered 0 0
/dev/block/sdb3 /data ext4 rw,nosuid,nodev,noatime,discard,...
/dev/fuse /mnt/shell/emulated fuse rw,nosuid,nodev,...
...
```

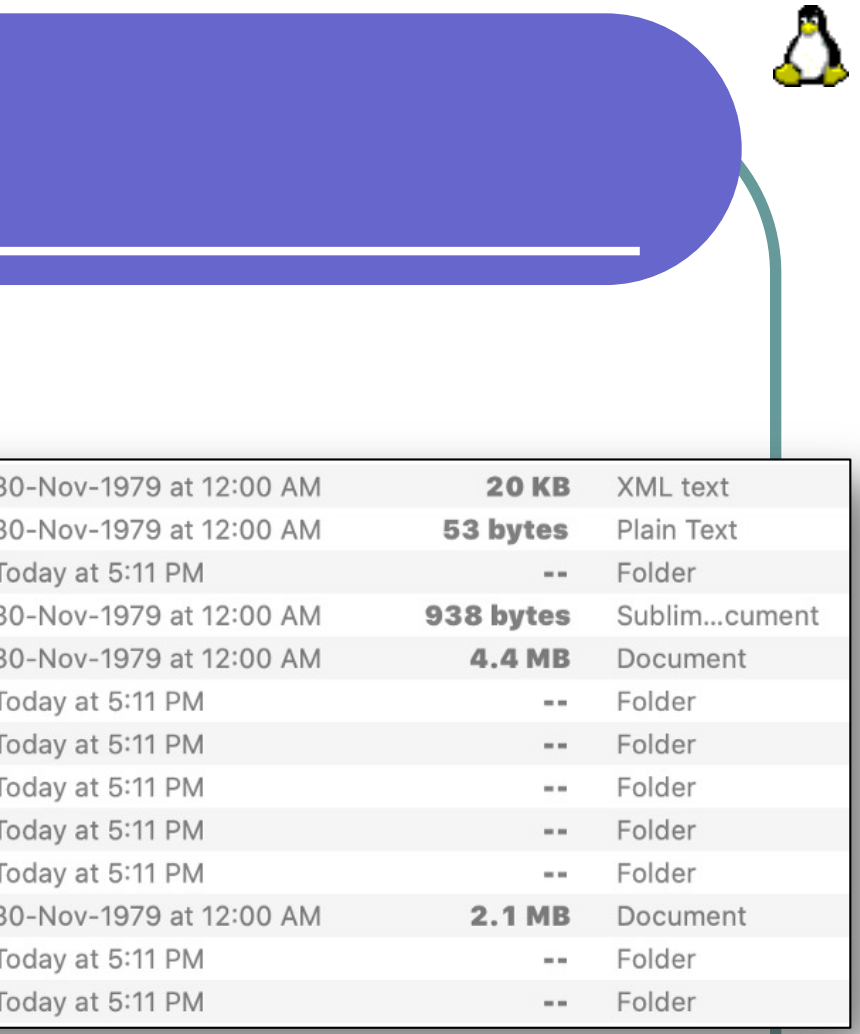
Android storage

- rootfs (/): partizione in-RAM (initrd), contiene i file di avvio e configurazione
- /system: file di Android OS
- /data: apps, contatti, messaggi, config, etc.
- /cache: dati e apps acceduti di frequente
- /sdcard: mountpoint memoria SD esterna (oppure, partizione "emulata", salvata in /data)

Applicazioni Android

- In Android, una applicazione ("app") è un **contenitore** di:
  - codice (tipicamente Java)
  - file di configurazione
  - grafica
  - altri dati
- Distribuita come singolo file
  - archivio *zip* con estensione **.apk** (Android Package)
  - diverso da un tradizionale file eseguibile Linux

Android Package



AndroidManifest.xml	30-Nov-1979 at 12:00 AM	**20 KB**	XML text
androidsupportmultidexversion.txt	30-Nov-1979 at 12:00 AM	**53 bytes**	Plain Text
▶ assets	Today at 5:11 PM	--	Folder
build-data.properties	30-Nov-1979 at 12:00 AM	**938 bytes**	Sublim...cument
classes.dex	30-Nov-1979 at 12:00 AM	**4.4 MB**	Document
▶ jsr305_annotations	Today at 5:11 PM	--	Folder
▶ META-INF	Today at 5:11 PM	--	Folder
▶ net	Today at 5:11 PM	--	Folder
▶ org	Today at 5:11 PM	--	Folder
▶ res	Today at 5:11 PM	--	Folder
resources.arsc	30-Nov-1979 at 12:00 AM	**2.1 MB**	Document
▶ zoneinfo	Today at 5:11 PM	--	Folder
▶ zoneinfo-global	Today at 5:11 PM	--	Folder

https://proandroiddev.com/dissection-of-an-apk-7ae9321c7b9a

Android Package

1. **Manifesto**: file XML, contiene dichiarazioni su cosa fa l'applicazione, e su come eseguirla ed interfacciarsi con essa
2. **Risorse**: file con stringhe, GUI layout in XML, immagini, etc.
3. **Codice**: bytecode e librerie native
4. **Firma digitale**: identifica l'autore in maniera sicura

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.email">
    <application>

        <activity android:name="com.example.email.MailMainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name="com.example.email.ComposeActivity">
            <intent-filter>
                <action android:name="android.intent.action.SEND" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="*/*" />
            </intent-filter>
        </activity>
```

L'applicazione è identificata da un **package Java**

Uno o più **punti di ingresso** alle funzionalità della app:

- *Activity*
- *Receiver*
- *Service*
- *Content provider*

Nome della **classe Java** che implementa il componente; è istanziata dal sistema su domanda

Package Manager

- Il **package manager** è un servizio di sistema che:
  - tiene traccia delle applicazioni installate
  - gestisce i loro permessi, le loro azioni e relative activity
- Android chiama il package manager quando si verifica un **evento da gestire** (Intent)

Package Manager

- Elenca le app installate e i relativi package Java

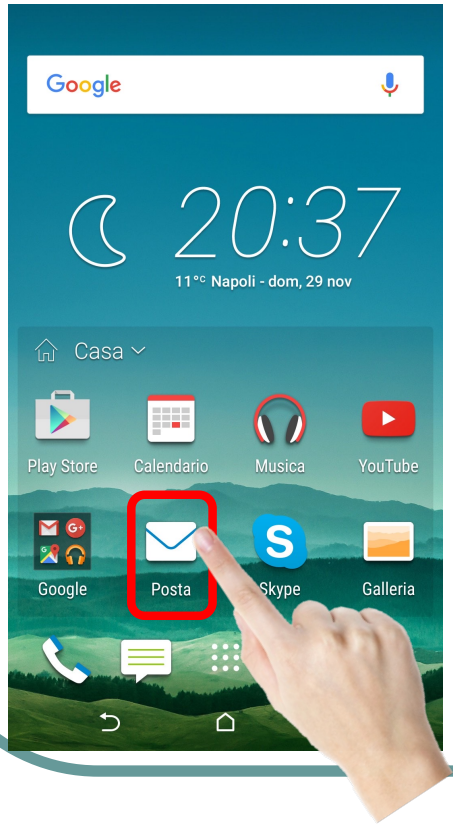
```
$ adb shell  "pm list packages"
```

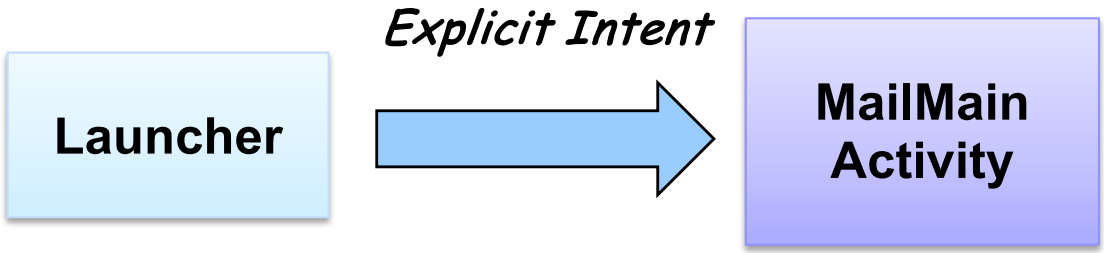
- Scarica il file APK di una app installata

```
$ adb shell "pm path com.android.browser"

$ adb pull /system/app/Browser/Browser.apk
```

Intent

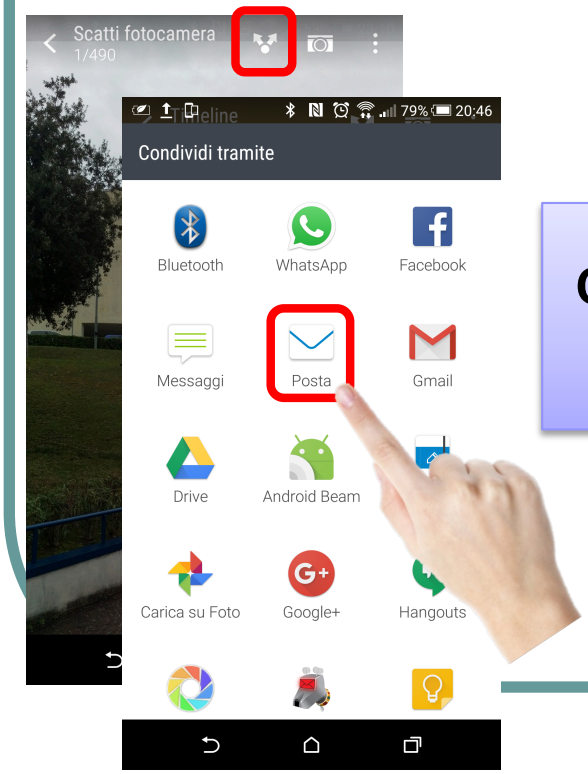
- **Explicit Intent**: indica il **package name della app** (es. *com.example.mail*), e il **class name della Activity** destinataria (es. *com.example.mail.MailMainActivity*)

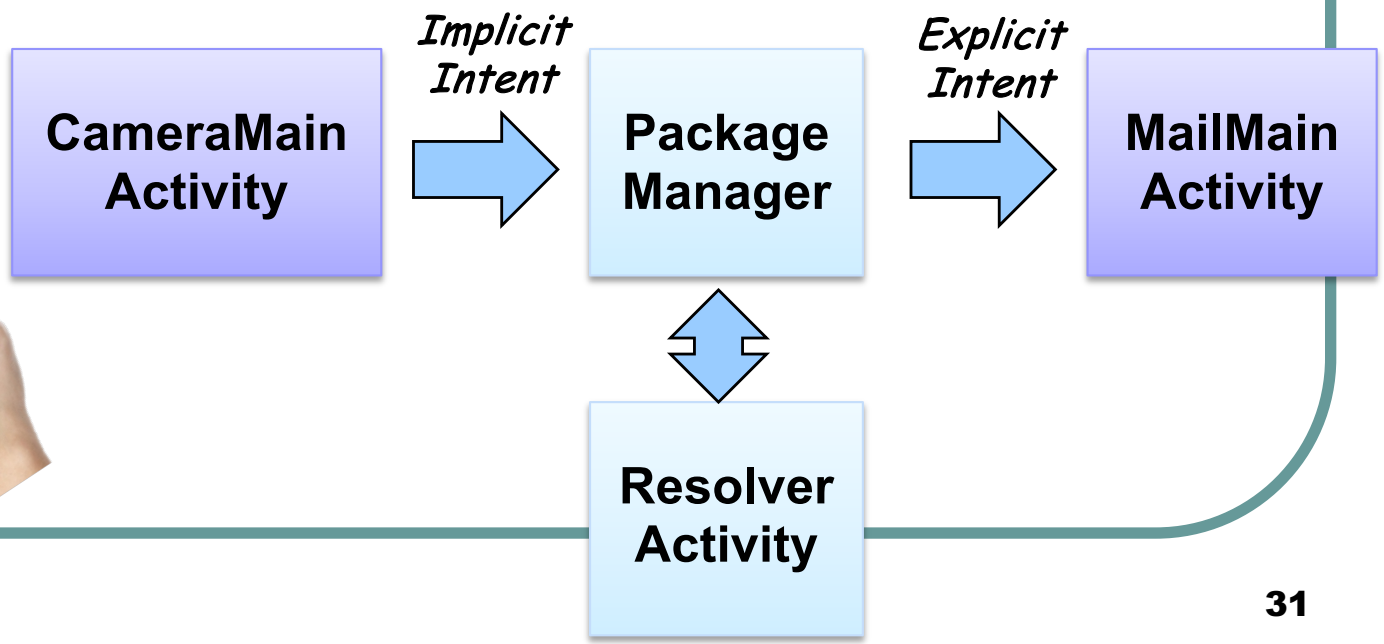




Intent

- **Implicit Intent**: descrive una **azione** da svolgere
  - Es., l'azione di "condividere una immagine"
  - Se l'azione è supportata da più app, il sistema mostra un menù di selezione all'utente





Intent

- Intent implicito per apertura di un file immagine

```
$ adb push  UNINA.png  /sdcard/Download

$ adb shell "am start -t image/* -d file:///sdcard/Download/UNINA.png"
```

- Intent esplicito

```
$ adb shell "am start -t image/* -d file:///sdcard/Download/UNINA.png
                -a ACTION_VIEW
                -n com.android.gallery3d/.filtershow.crop.CropActivity"
```

Intent

- Intent implicito per avvio app camera

```
$ adb shell "am start -a android.media.action.STILL_IMAGE_CAMERA"

$ adb shell "input keyevent 27"
```

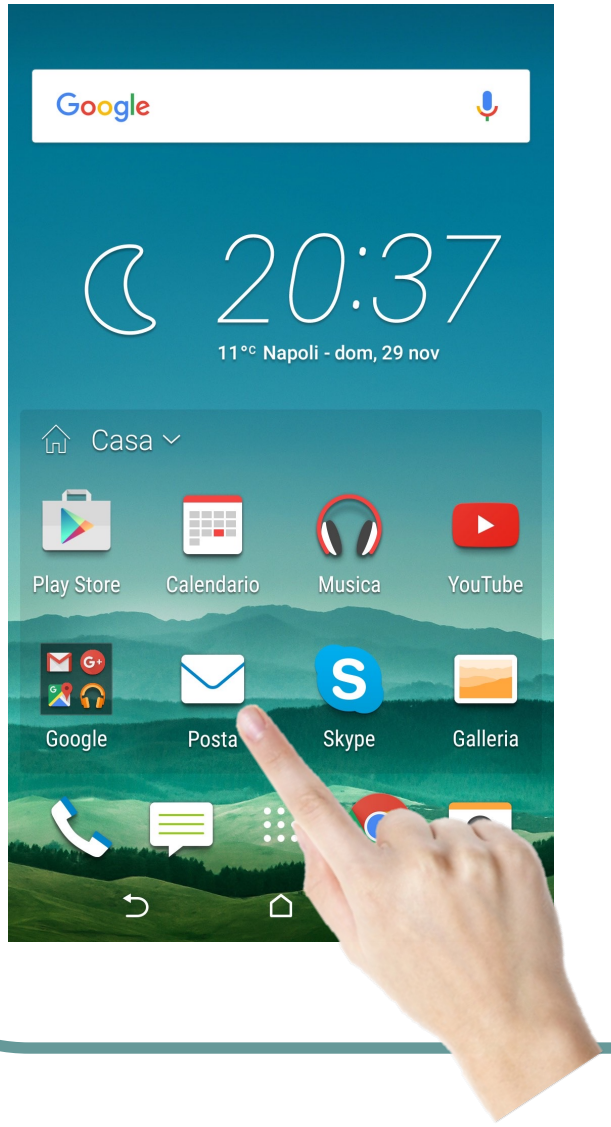
Activity

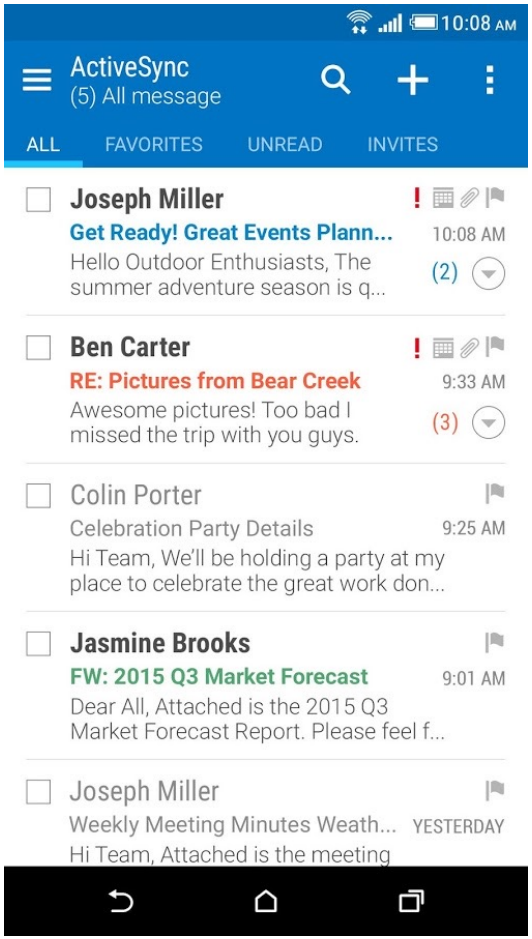
- Una **Activity** è una schermata grafica che interagisce con l'utente
- Lo **Activity Manager** è il servizio che avvia e gestisce le Activity

Nell'esempio precedente (app Email):

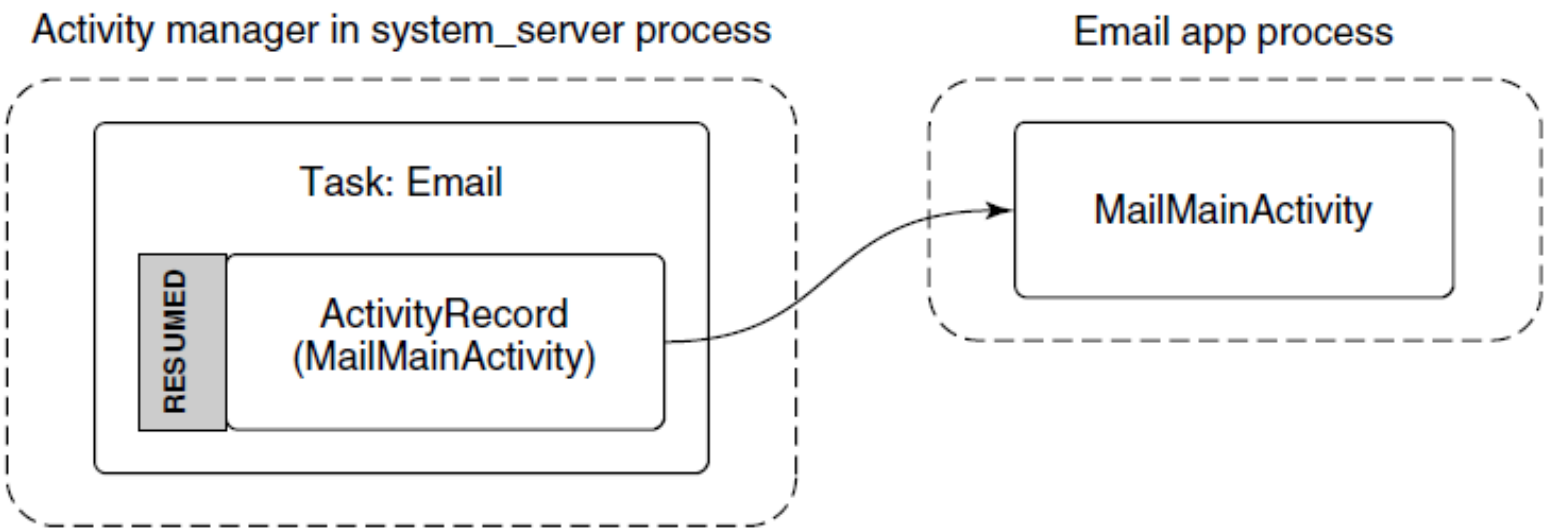
- **MailMainActivity**: visualizza la lista messaggi (si avvia al lancio dell'app dalla schermata "home")
- **ComposeActivity**: crea un nuovo messaggio (avviata dalla MailMainActivity, o da altre app su richiesta)

Il ciclo di vita di una Activity





Il ciclo di vita di una Activity



- Android avvia di un **nuovo processo**
- La **Activity "MailMain"** è eseguita nel processo
- Lo **Activity Manager** inserisce una riga ("record") nella sua lista di Activity

Activity

- Elenco delle activity relative ad un package Java

```
$ adb shell  "dumpsys package | grep -i 'com.android.calendar' |
                                                    grep 'Activity'"
```

- Info su activity corrente sul display (foreground)

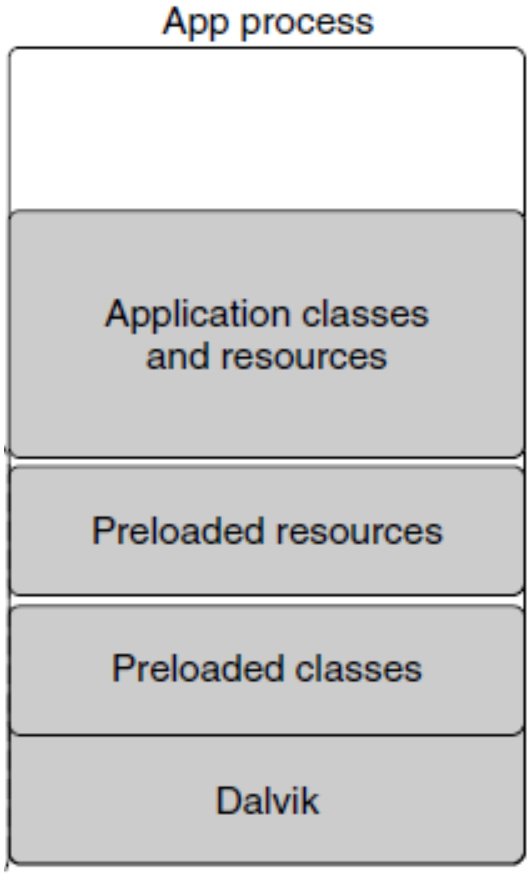
```
$ adb shell "dumpsys window windows"

$ adb shell "dumpsys activity | grep top-activity"   # ottiene il PID
```

- Stack delle activity avviate

```
$ adb shell "dumpsys activity activities"
```

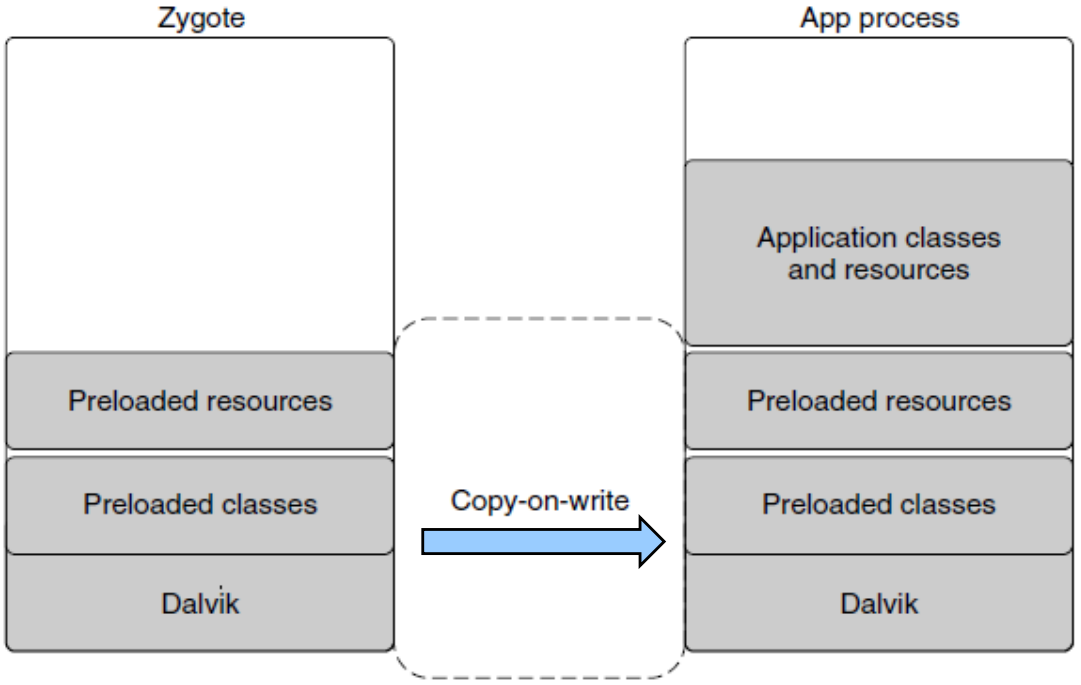
Gestione dei processi

- Il nuovo processo avvia una istanza di Java Virtual Machine
  - Dalvik (prime versioni)
  - Android RunTime (oggi)
- Contiene classi e risorse sia di Android OS, sia della app

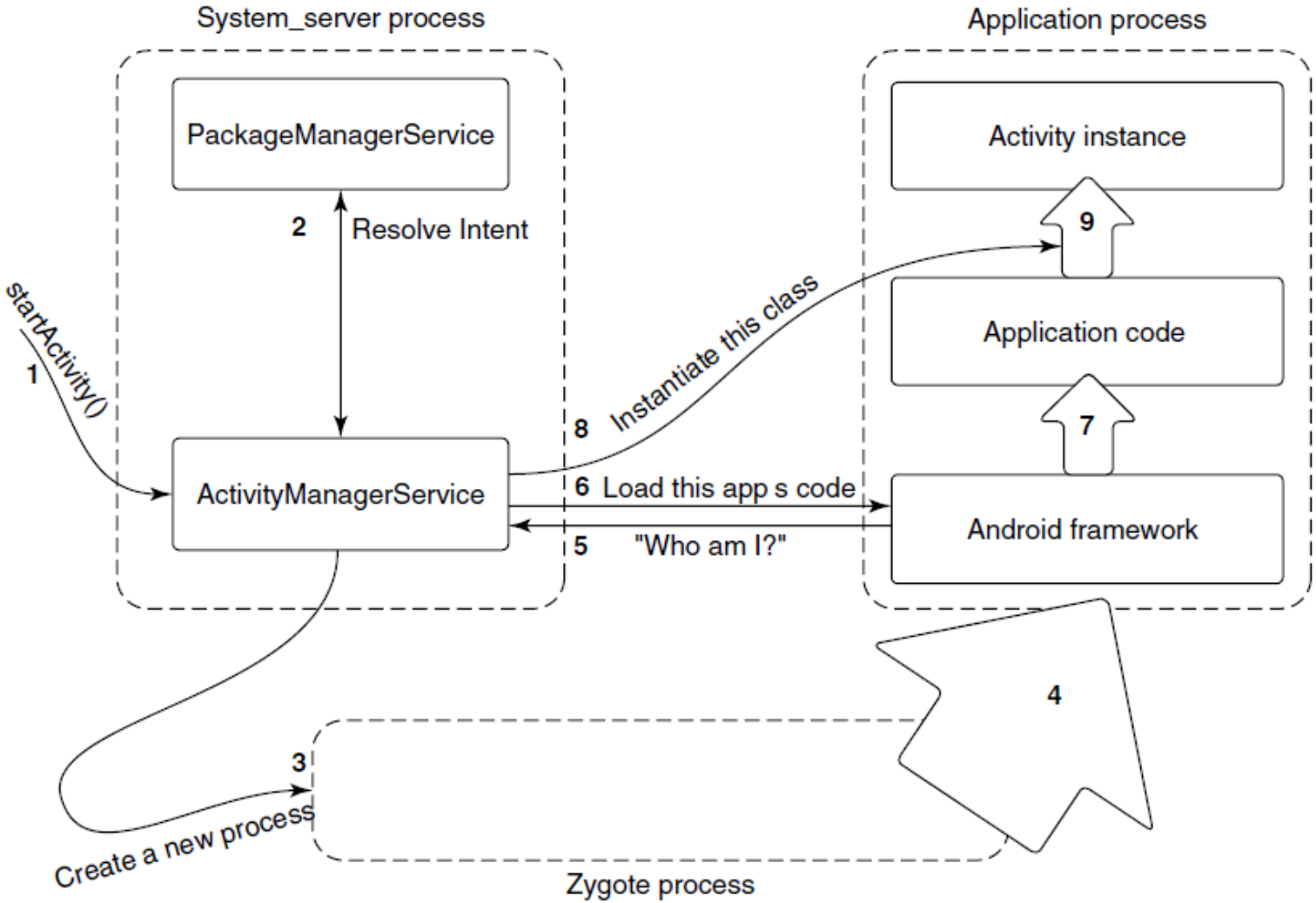


Creazione di un processo Zygote

- Per accelerare l'avvio, Android "clona" con fork() il processo Zygote
- Già inizializzato con classi e risorse

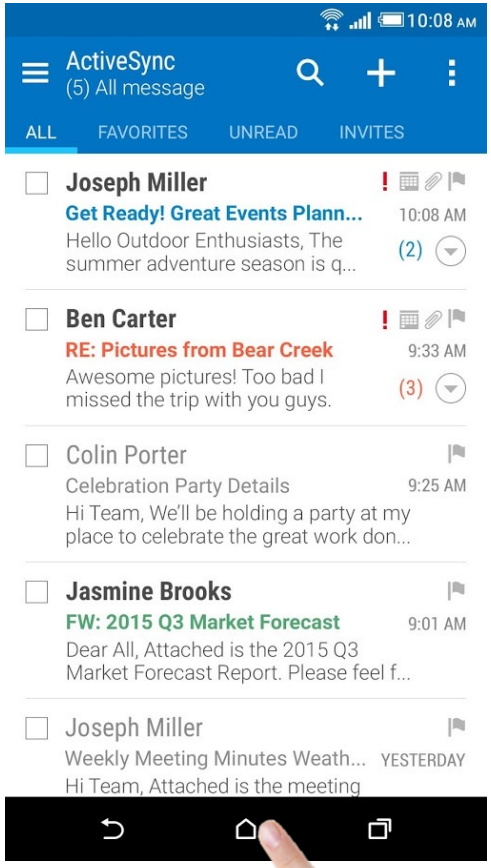


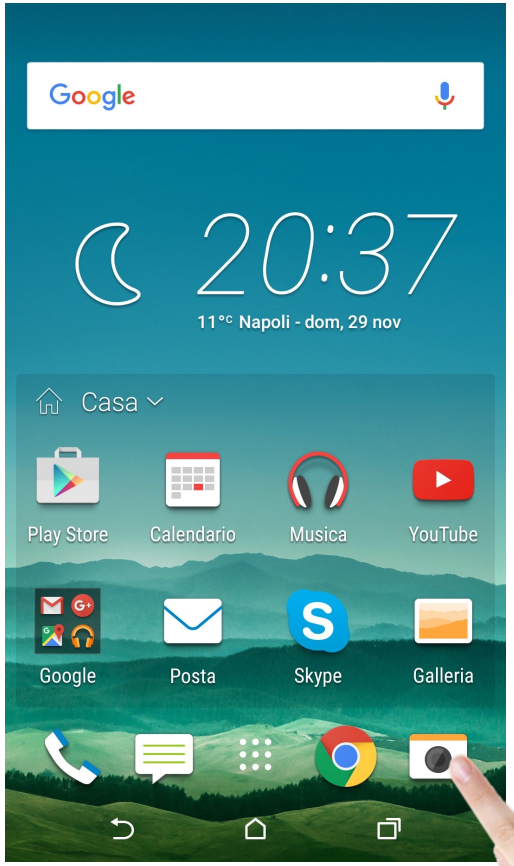
Creazione di un processo Zygote



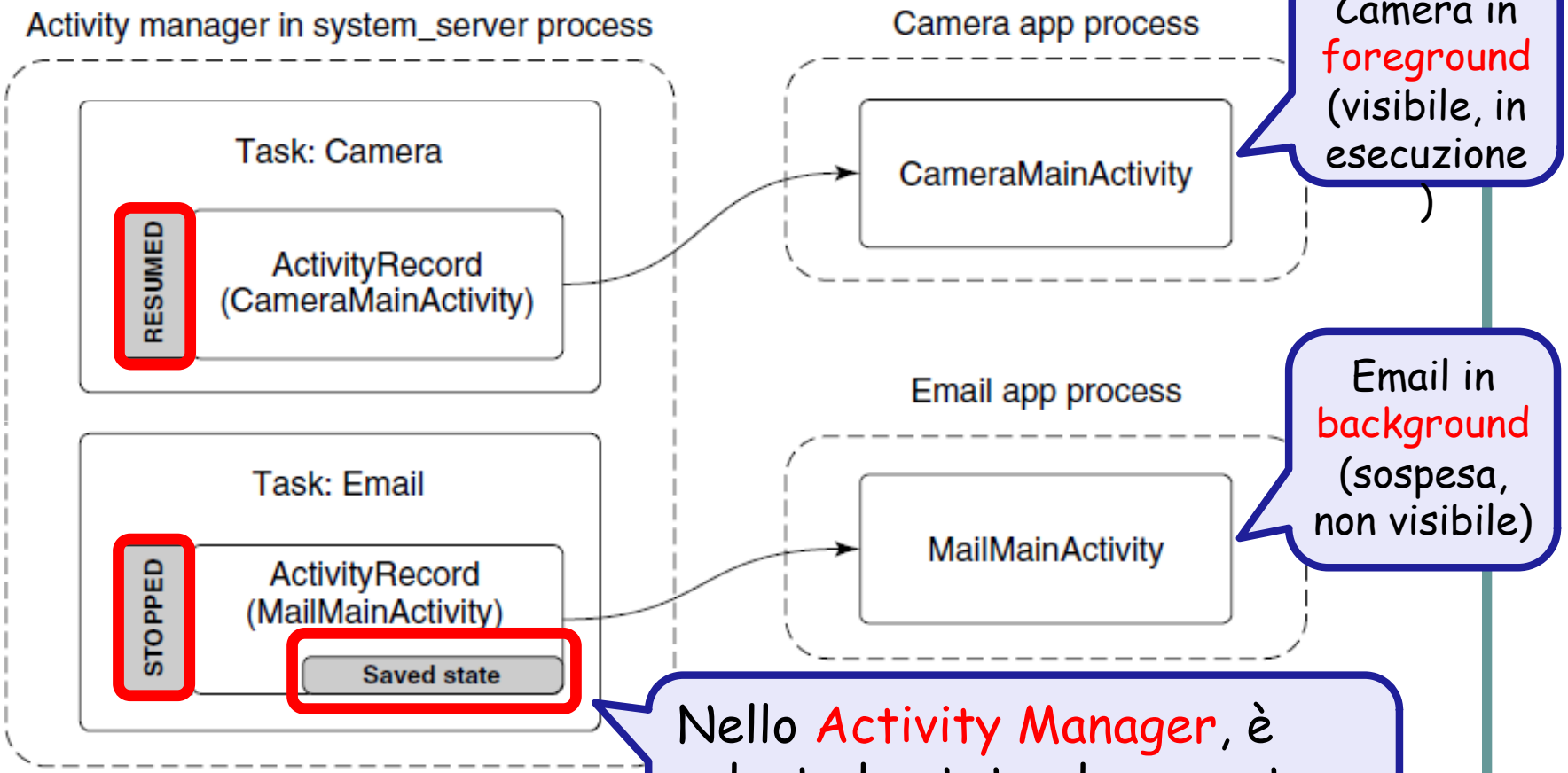
Il ciclo di vita di una Activity

Ritorno alla schermata "home", avvio della app "camera"

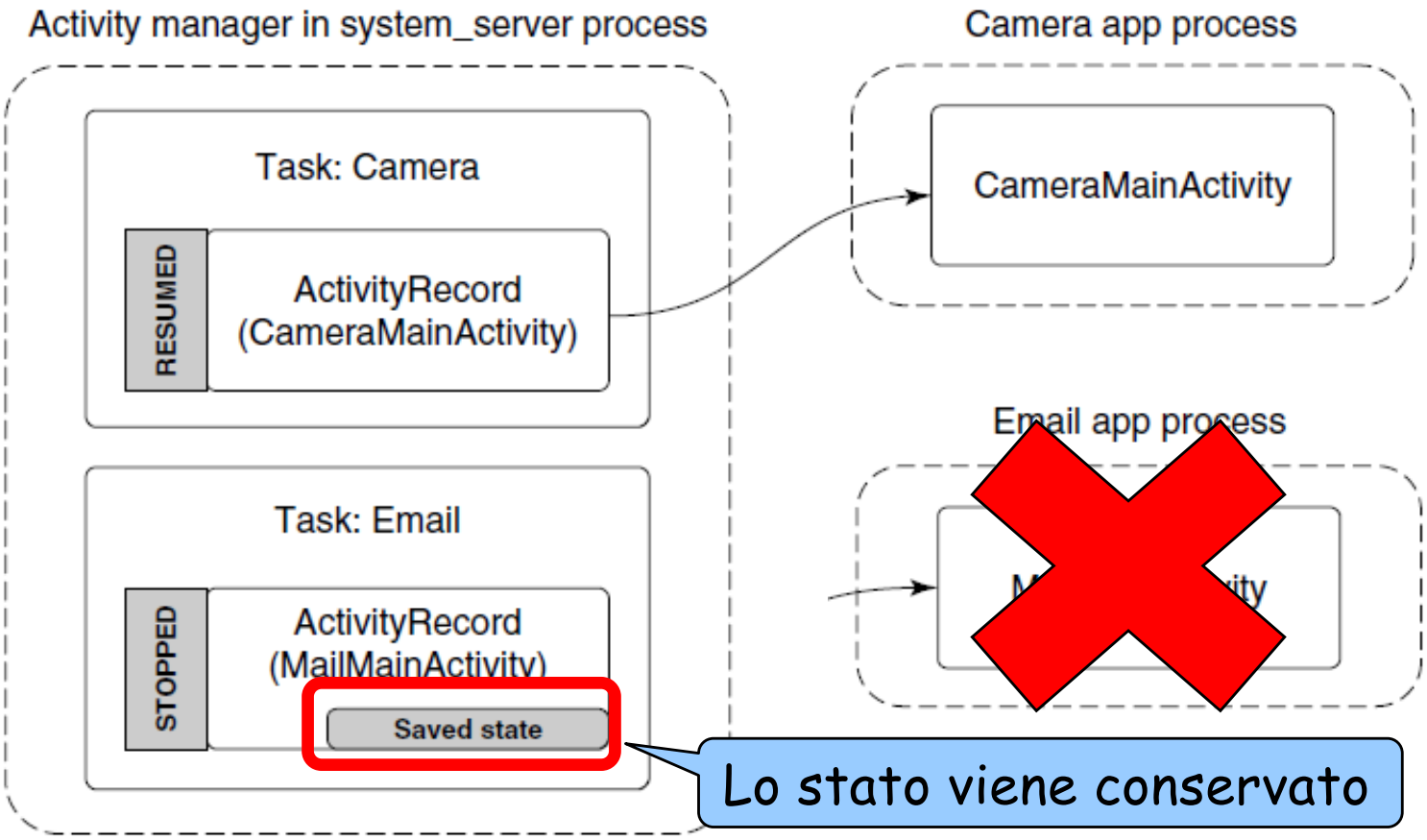




Il ciclo di vita di una Activity



Il ciclo di vita di una Activity



In caso di scarsità di memoria libera, il SO **non può swappare i processi** (non c'è un disco capiente!).
Può invece **deallocarli**, dopo aver **salvato il loro stato**

Gestione della memoria

- I processi non escono esplicitamente, ma sono lasciati **cached**, finché non vengono terminati dal SO
- Se la RAM disponibile scarseggia, lo **out-of-memory (OOM) killer** uccide i componenti a minor importanza

Category	Description	oom_adj
SYSTEM	The system and daemon processes	–16
PERSISTENT	Always-running application processes	–12
FOREGROUND	Currently interacting with user	0
VISIBLE	Visible to user	1
PERCEPTIBLE	Something the user is aware of	2
SERVICE	Running background services	3
HOME	The home/launcher process	4
CACHED	Processes not in use	5

Lo Activity Manager assegna un punteggio di importanza alle Activity

Gestione della memoria

Process	State	Importance
system	Core part of operating system	SYSTEM
phone	Always running for telephony stack	PERSISTENT
email	Current foreground application	FOREGROUND
camera	In use by email to load attachment	FOREGROUND
music	Running background service playing music	PERCEPTIBLE
media	In use by music app for accessing user's music	PERCEPTIBLE
download	Downloading a file for the user	SERVICE
launcher	App launcher not current in use	HOME
maps	Previously used mapping application	CACHED

La **Email app** sta usando la **Camera** per caricare un allegato…

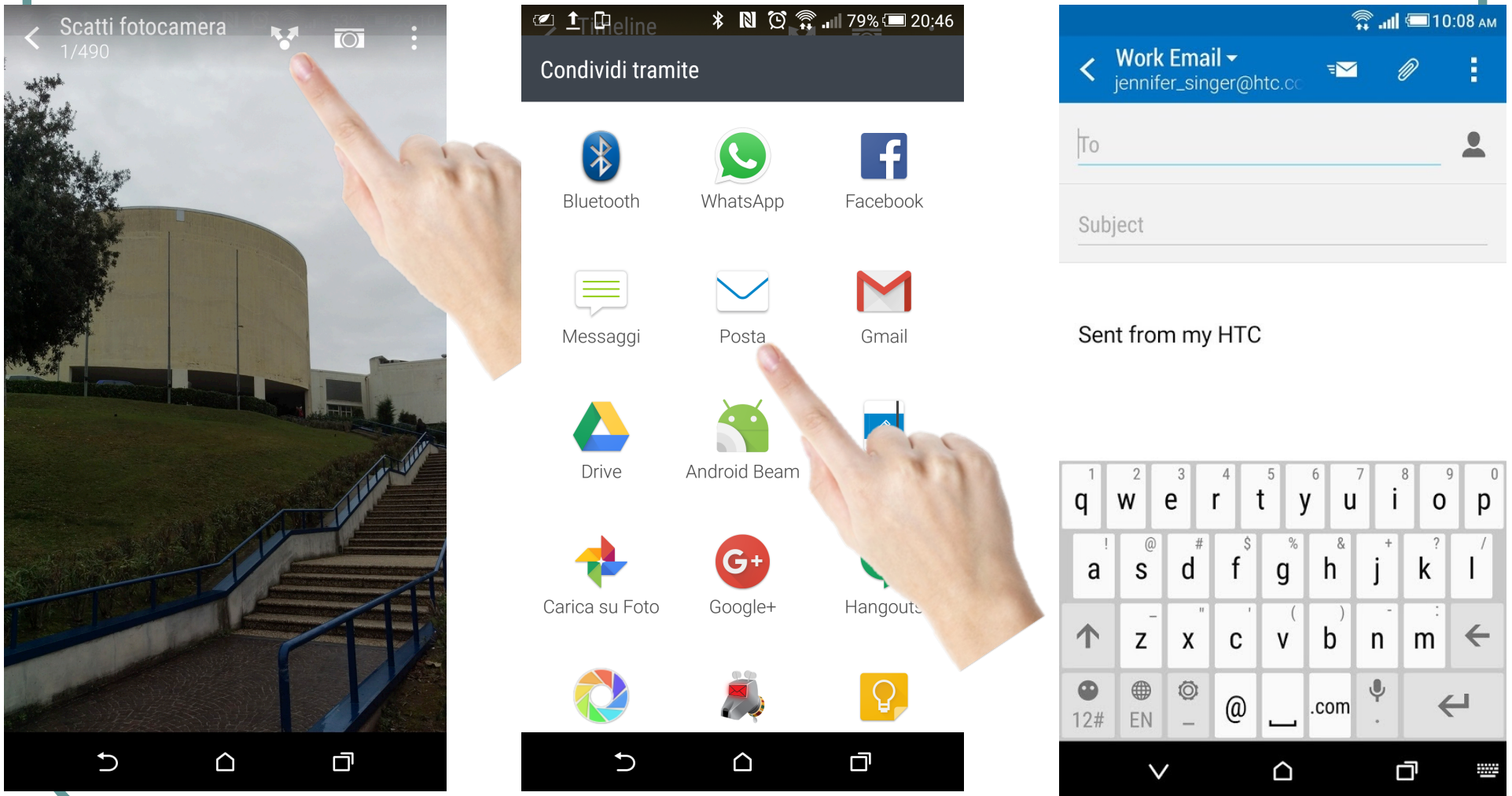
… in background, è in esecuzione un brano

Maps è stata usata recentemente

Process	State	Importance
system	Core part of operating system	SYSTEM
phone	Always running for telephony stack	PERSISTENT
email	Current foreground application	FOREGROUND
music	Running background service playing music	PERCEPTIBLE
media	In-use by music app for accessing user's music	PERCEPTIBLE
download	Downloading a file for the user	SERVICE
launcher	App launcher not current in use	HOME
camera	Previously used by email	CACHED
maps	Previously used mapping application	CACHED+1

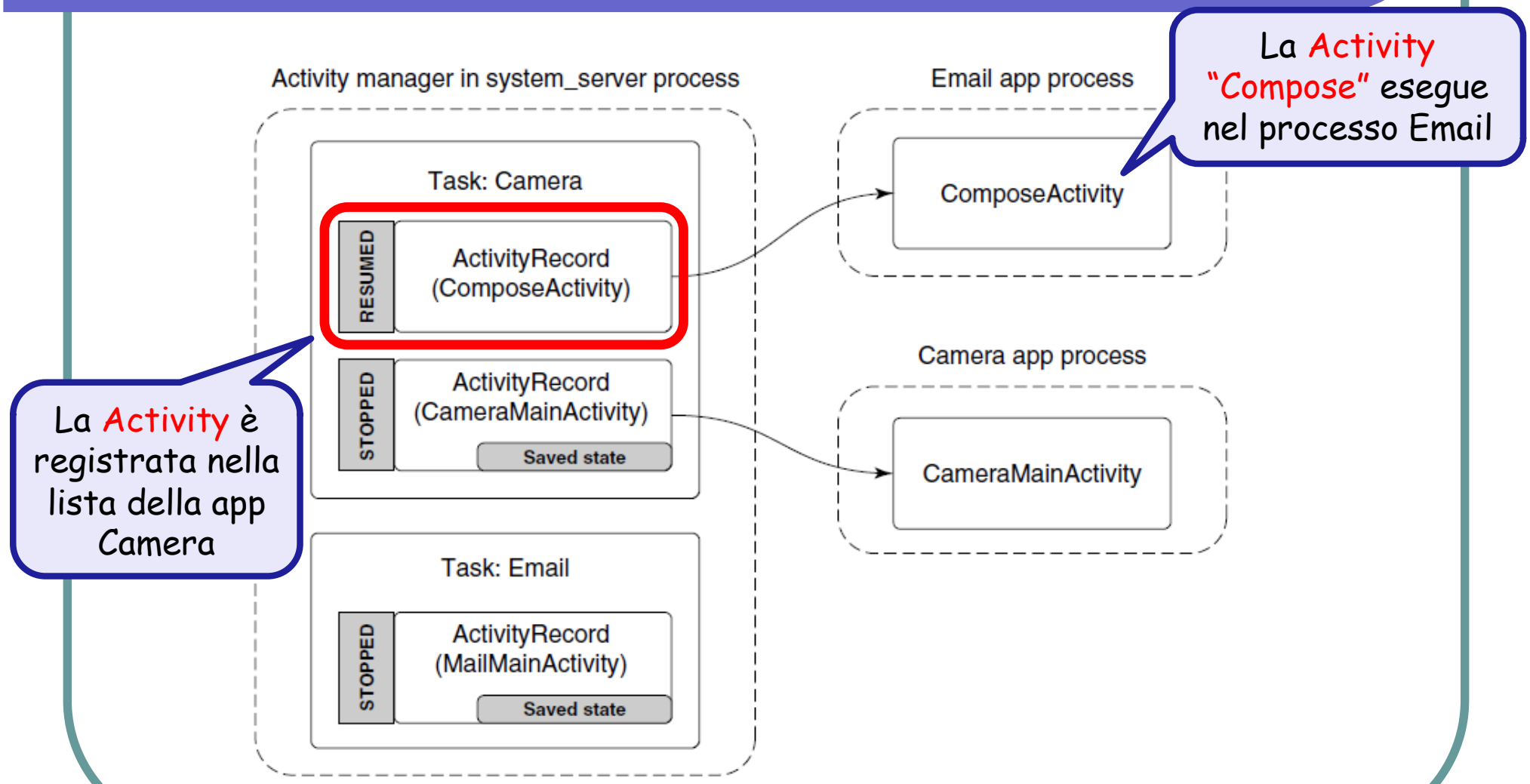
Quando Camera non è più utilizzata da Email, la sua priorità è ridotta

Il ciclo di vita di una Activity



L'utente può usare l'app **Camera** per condividere foto tramite **Email**

Il ciclo di vita di una Activity



Android security

- Nei SO tradizionali, i processi hanno gli **stessi permessi** dell'utente che li ha avviati
- Tutti i file e le risorse dell'utente sono accessibili dalle applicazioni eseguite dall'utente

Questo modello tradizionale è **poco sicuro** in un contesto mobile.

L'utente utilizza app di terze parti, spesso poco affidabili.

Sandboxing

- In Android, la app è considerata un "ospite" del sistema
- Quando si installa una app:
  - si crea un utente Linux univoco e dedicato per quella app
  - si crea una cartella "home" privata per quella app

UID	Purpose
0	Root
1000	Core system (system_server process)
1001	Telephony services
1013	Low-level media processes
2000	Command line shell access
10000–19999	Dynamically assigned application UIDs
100000	Start of secondary users

UID: User ID

Sandboxing

```
$ adb shell "ps -A"

...
system    670   379   1135408 135996 00000000 f76f9035 S system_server
u0_a13    1045  379   1080708 118104 00000000 f76f9035 S com.android.systemui
wifi      1076  1     8712   4760  00000001 f7746995 S /system/bin/wpa_supplicant
u0_a38    1083  379   1012732 67492 00000000 f76f9035 S com.android.inputmethod.latin
system    1166  379   995684 42340 00000000 f76f9035 S com.genymotion.systempatcher
radio     1189  379   993248 49808 00000000 f76f9035 S com.android.server.telecom
system    1222  379   993672 43648 00000000 f76f9035 S com.genymotion.genyd
radio     1258  379   1022824 72252 00000000 f76f9035 S com.android.phone
u0_a59    1343  379   1249420 229880 00000000 f76f9035 S com.google.android.gms.persistent
u0_a2     1394  379   1002668 60592 00000000 f76f9035 S android.process.acore
dhcp      1498  1     3912   1060  00000001 f76f52b0 S /system/bin/dhcpcd
u0_a59    1533  379   1432272 244252 00000000 f76f9035 S com.google.android.gms
u0_a39    1567  379   1062072 104388 00000000 f76f9035 S com.android.launcher3
u0_a59    1740  379   1163760 108612 00000000 f76f9035 S com.google.android.gms.ui
u0_a5     1763  379   1007304 66808 00000000 f76f9035 S android.process.media
u0_a59    1781  379   997380 47072 00000000 f76f9035 S com.google.process.location
u0_a59    1827  379   1179744 124696 00000000 f76f9035 S com.google.android.gms.unstable
u0_a8     1980  379   1002896 51984 00000000 f76f9035 S com.android.mms
u0_a55    2097  379   990276 38832 00000000 f76f9035 S com.android.smspush
u0_a1     2128  379   996728 53412 00000000 f76f9035 S com.android.providers.calendar
u0_a14    2184  379   997036 53336 00000000 f76f9035 S com.android.voicedialer
u0_a62    2302  379   997036 53376 00000000 f76f9035 S com.google.android.partnersetup
u0_a21    2541  379   1002964 61000 00000000 f76f9035 S com.android.calendar
u0_a31    2640  379   1016140 60476 00000000 f76f9035 S com.android.email
...
```

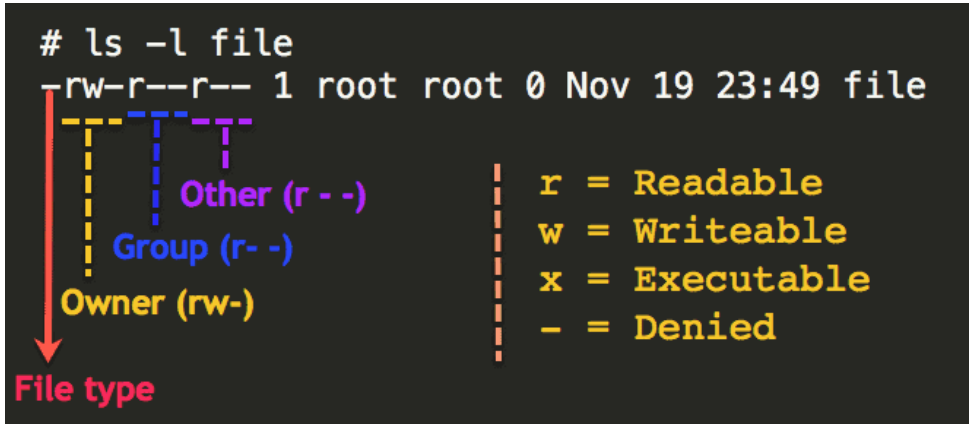
Utenti "fittizi", specifici per ogni app

Sandboxing

```
$ adb shell "ls /data/data"

com.android.backupconfirm
com.android.bluetooth
com.android.browser
com.android.calculator2
com.android.calendar
com.android.camera2
...

$ adb shell "ls -l /data/data/com.android.email/"

drwxrwx--x u0_a31   u0_a31       2021-12-13 05:35 cache
drwxrwx--x u0_a31   u0_a31       2021-12-13 05:35 databases
drwxrwx--x u0_a31   u0_a31       2021-12-13 05:35 files
lrwxrwxrwx install  install      2021-12-13 05:35 lib -> /data/app-lib/com.android.email
drwxrwx--x u0_a31   u0_a31       2021-12-13 05:36 shared_prefs
```

Android security

- In Linux/UNIX, tutte le risorse sono rappresentate da file (es. i file in /dev)
  - Ogni file ha 1 solo utente e gruppo proprietario
  - Permessi per lettura, scrittura, esecuzione



Android security

- In Linux/UNIX, tutte le risorse sono rappresentate da file (es. i file in /dev)
  - Ogni file ha 1 solo utente e gruppo proprietario
  - Permessi per lettura, scrittura, esecuzione

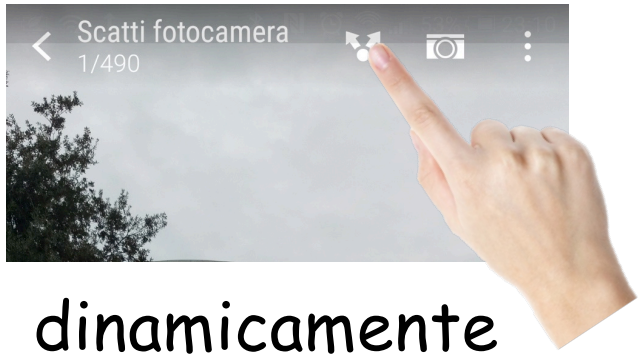
Questo modello tradizionale è **limitato** per un contesto mobile.

Esistono molti tipi di operazioni (es. avvio chiamata, scatto foto, lettura rubrica, etc.), diversi per ogni app.

Permessi

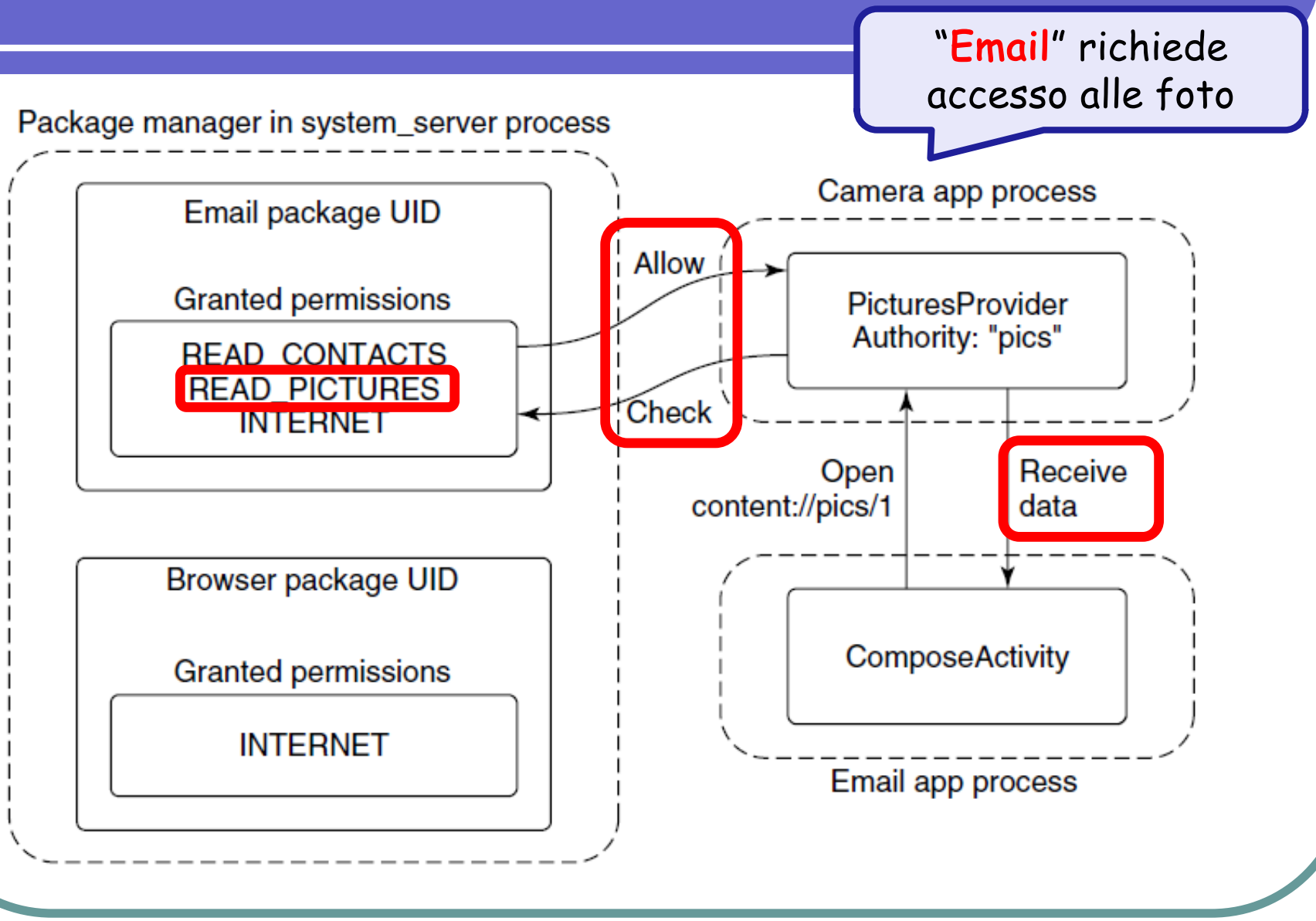
- Android assegna dei "privilegi" alle app
  - condivisione foto
  - accesso internet
  - accesso rubrica
  - ...

staticamente
(alla **installazione**)

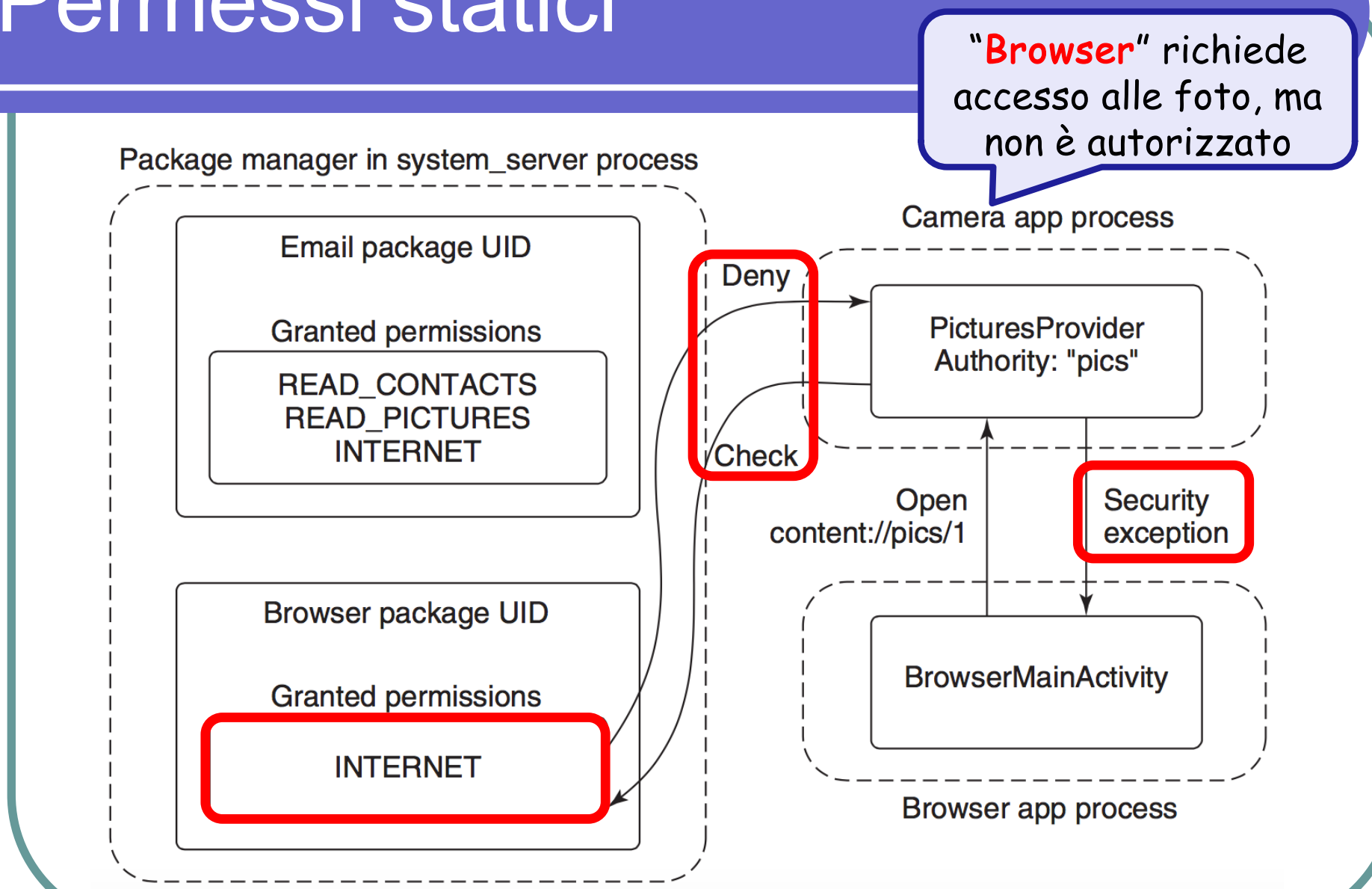


dinamicamente
(in base alle **azioni dell'utente**)

Permessi statici in Android



Permessi statici

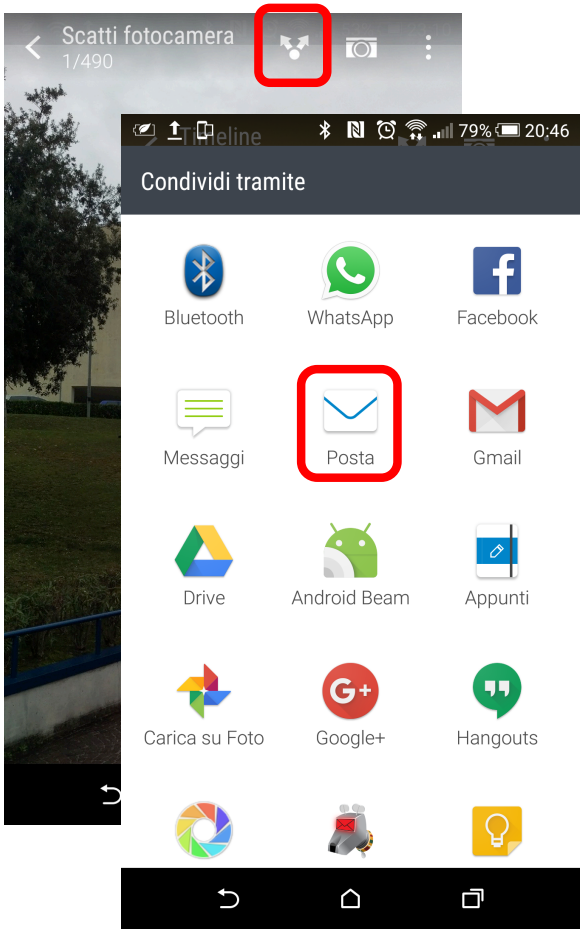


Permessi dinamici

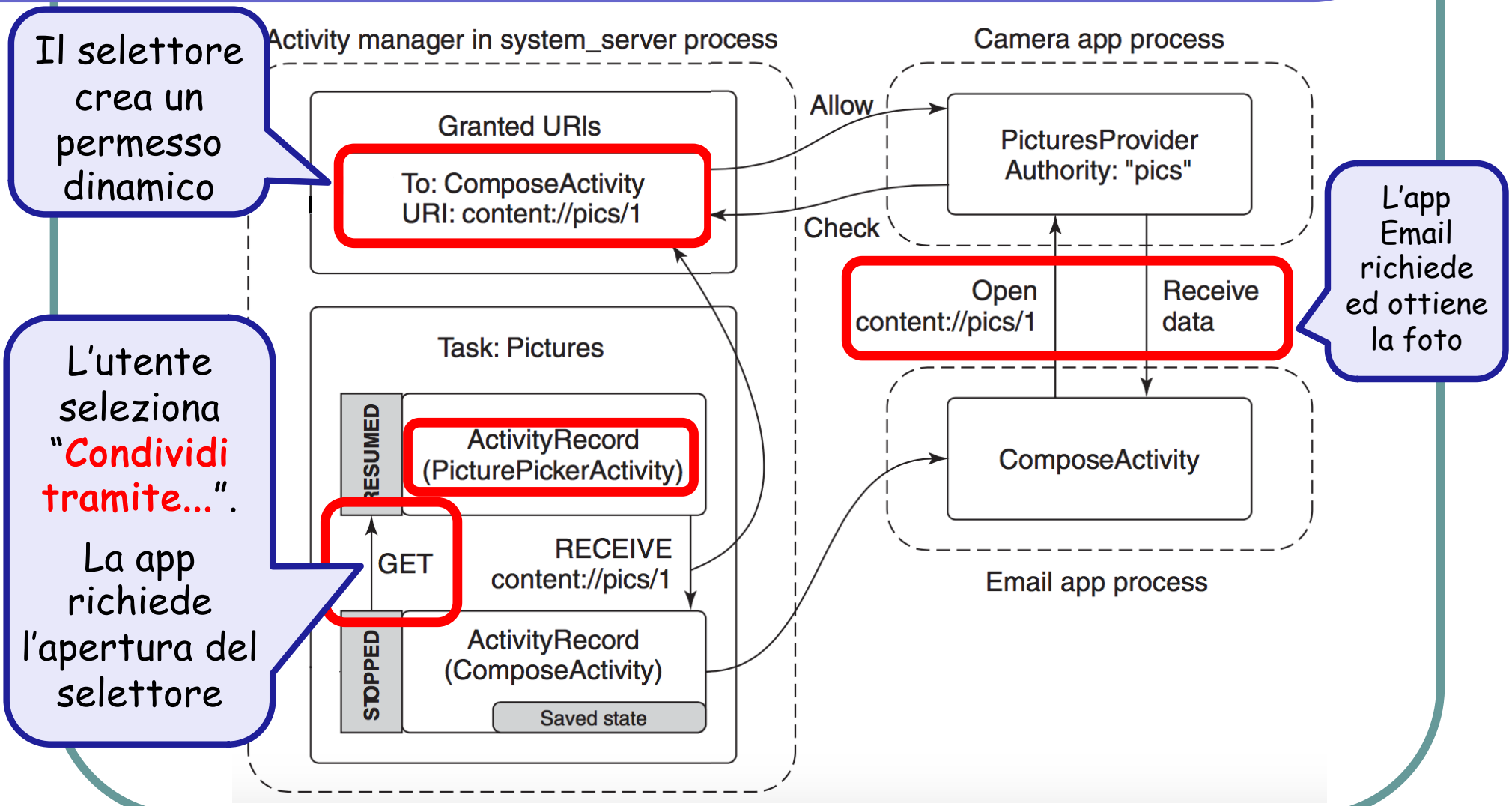
- I **permessi dinamici** danno una autorizzazione temporanea
- In caso di **azioni di condivisione**
- Maggiore flessibilità

Esempio:

- es., l'utente scatta una foto e clicca su un tasto "condividi"
- Si seleziona una applicazione (es. Email) con cui condividere
- Email acquisisce un permesso dinamico



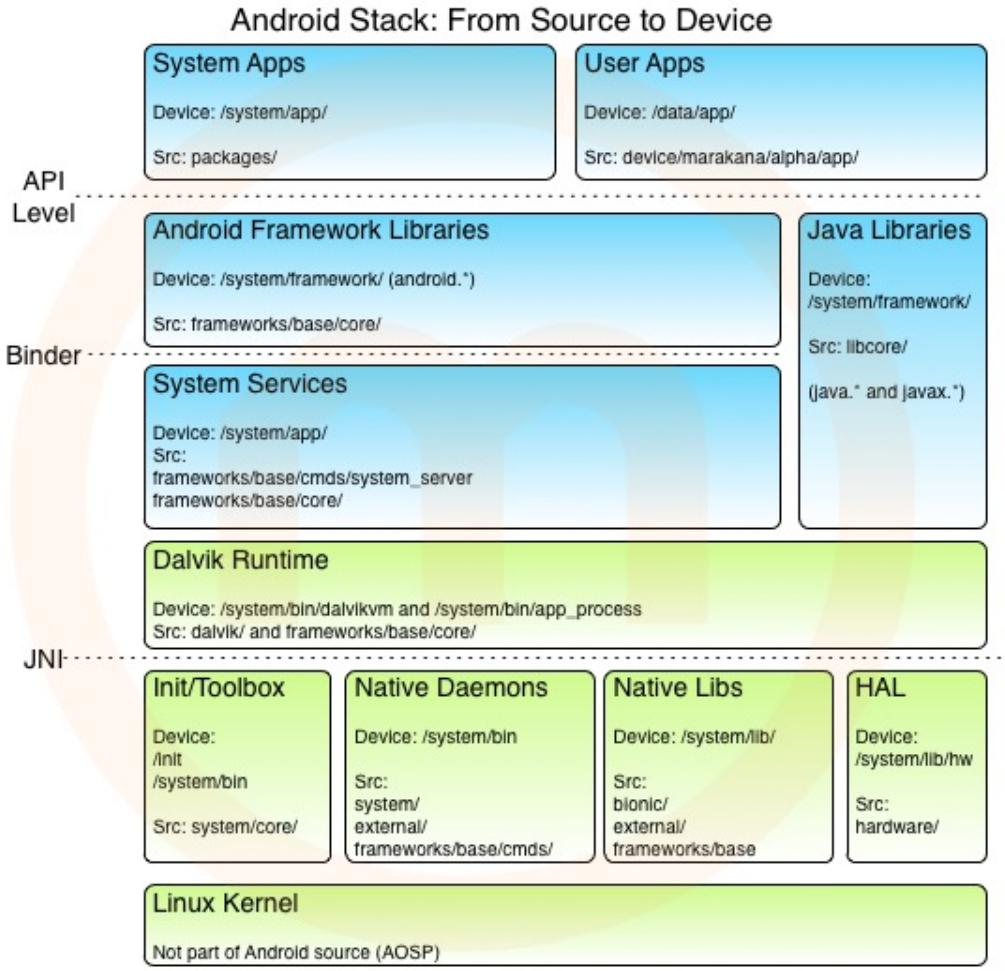
Permessi dinamici in Android



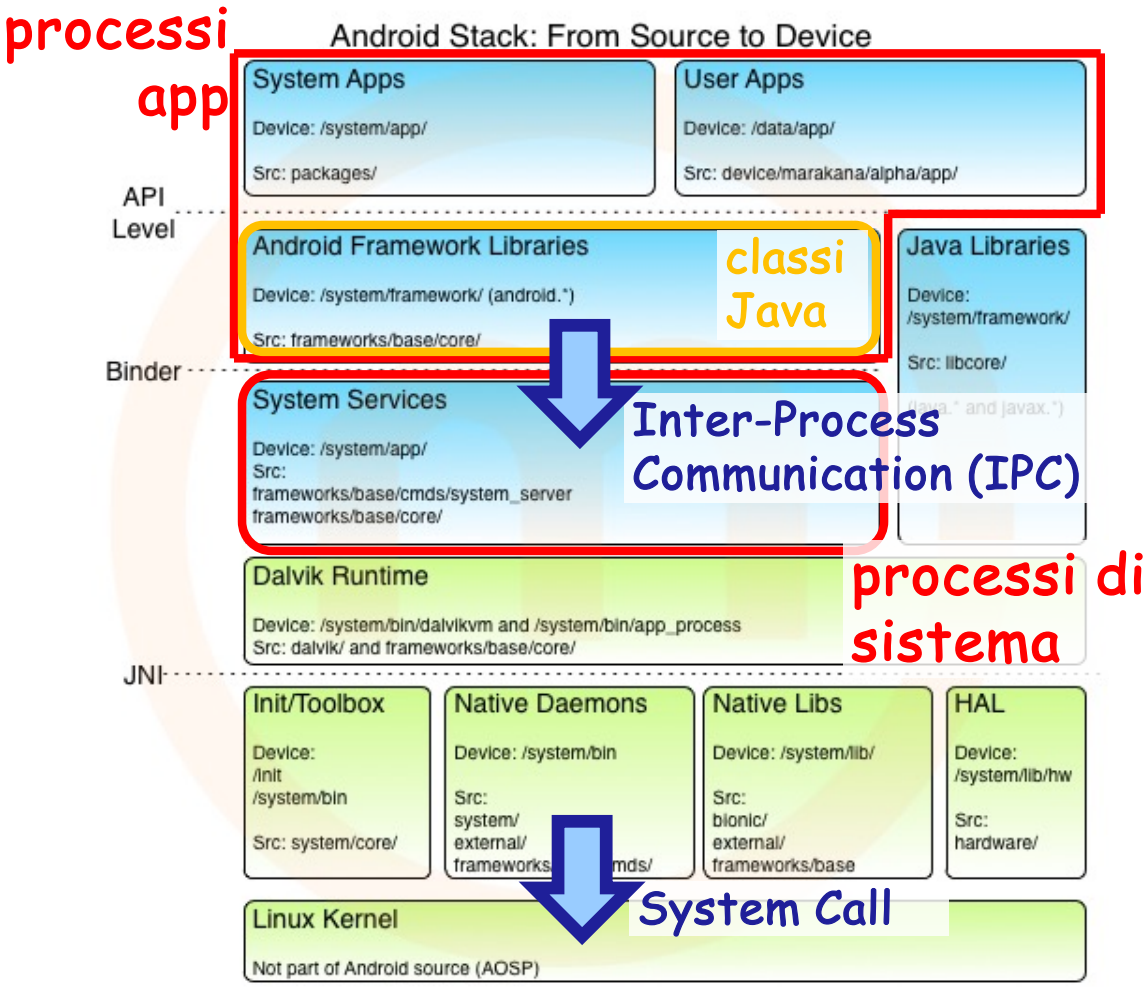
Servizi di sistema Android

- Il SO Android è strutturato in servizi
  - Fotocamera
  - Sensoristica
  - Audio
  - Connettività (Bluetooth, Wifi, ...)
  - ...
- Le app usano i servizi tramite classi Java (**Android Framework**)

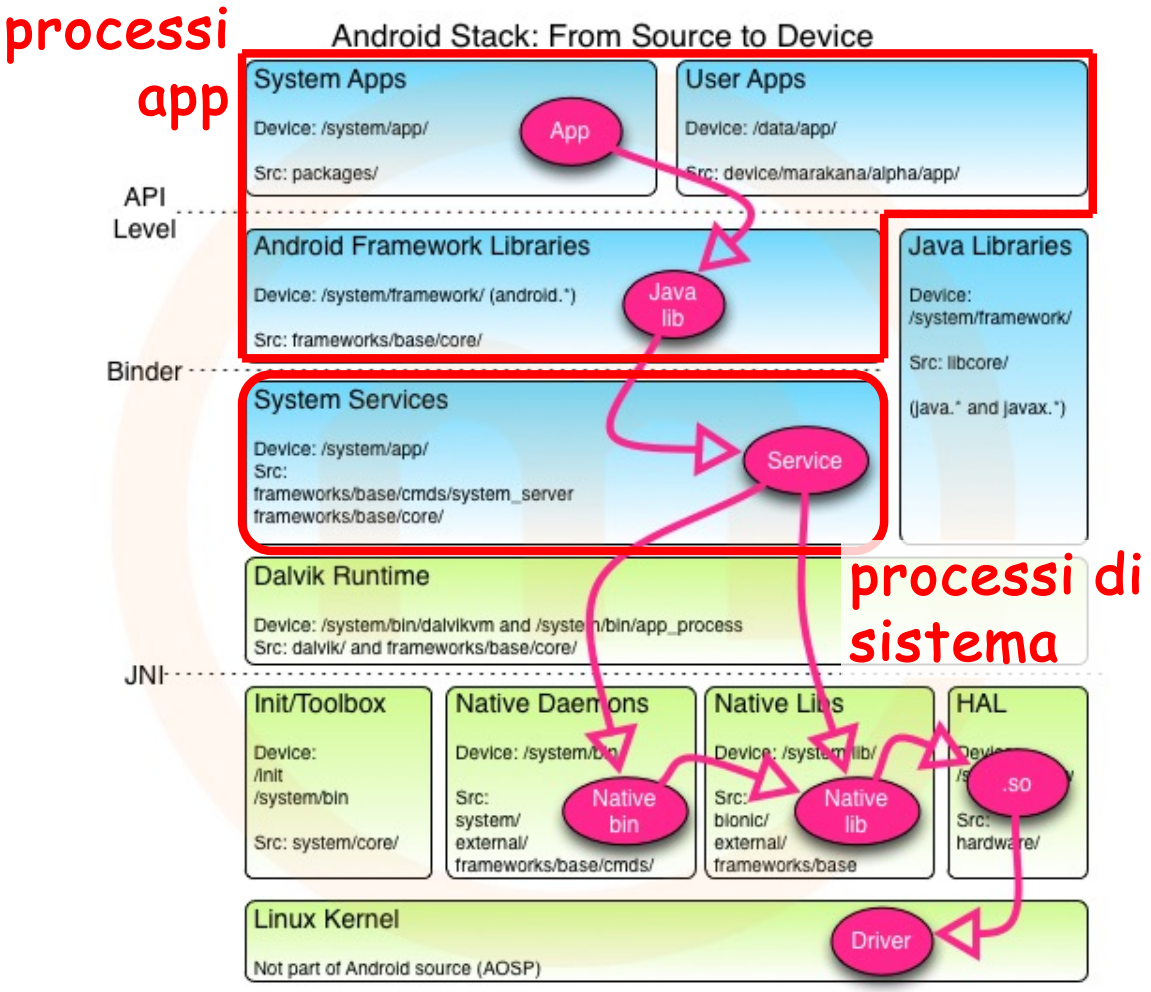
Servizi di sistema Android



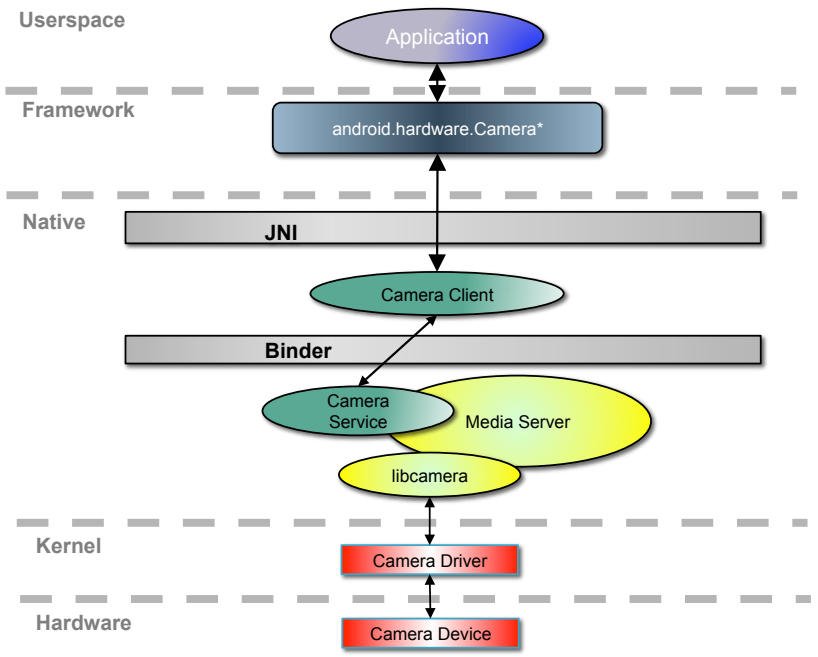
Servizi di sistema Android

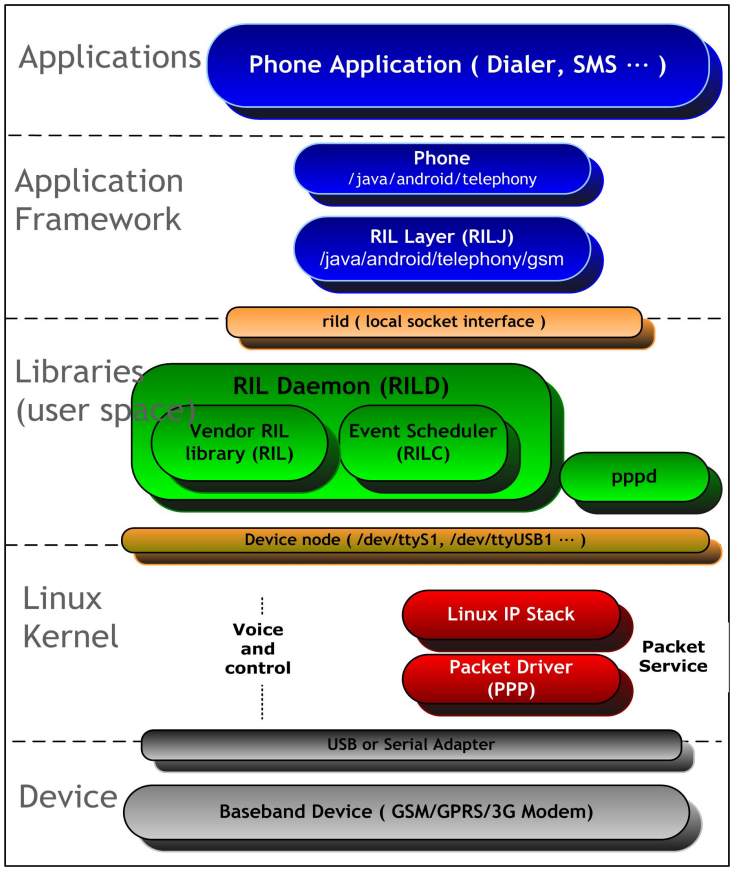


Servizi di sistema Android



Servizi di sistema Android



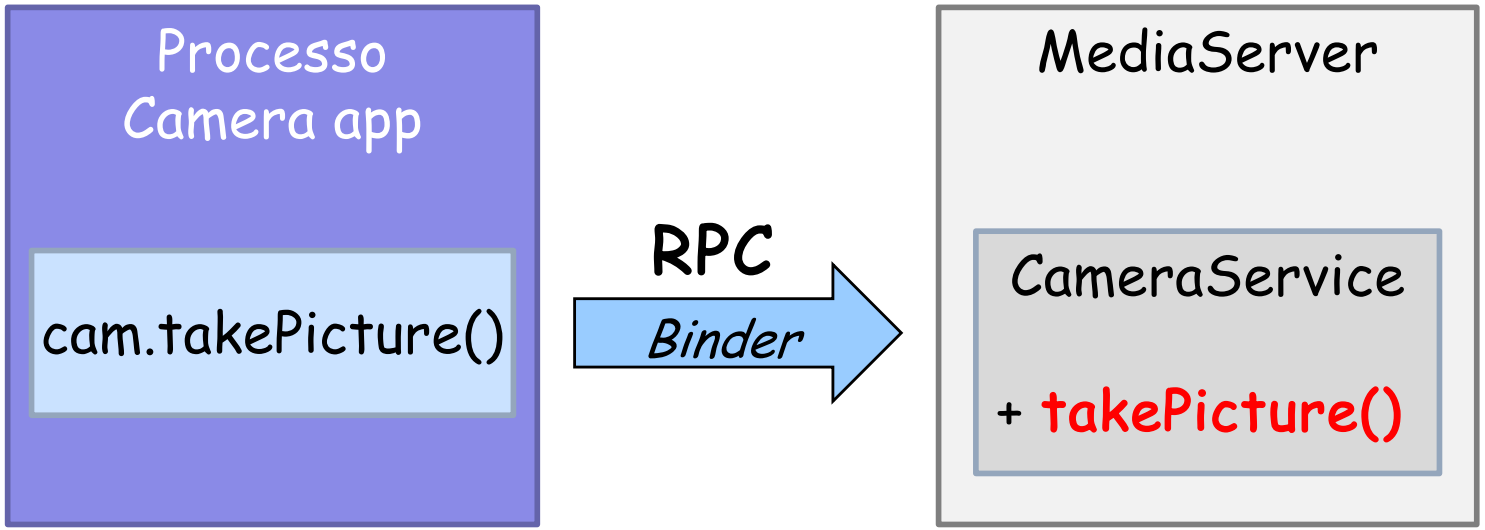


Camera subsystem

Phone subsystem

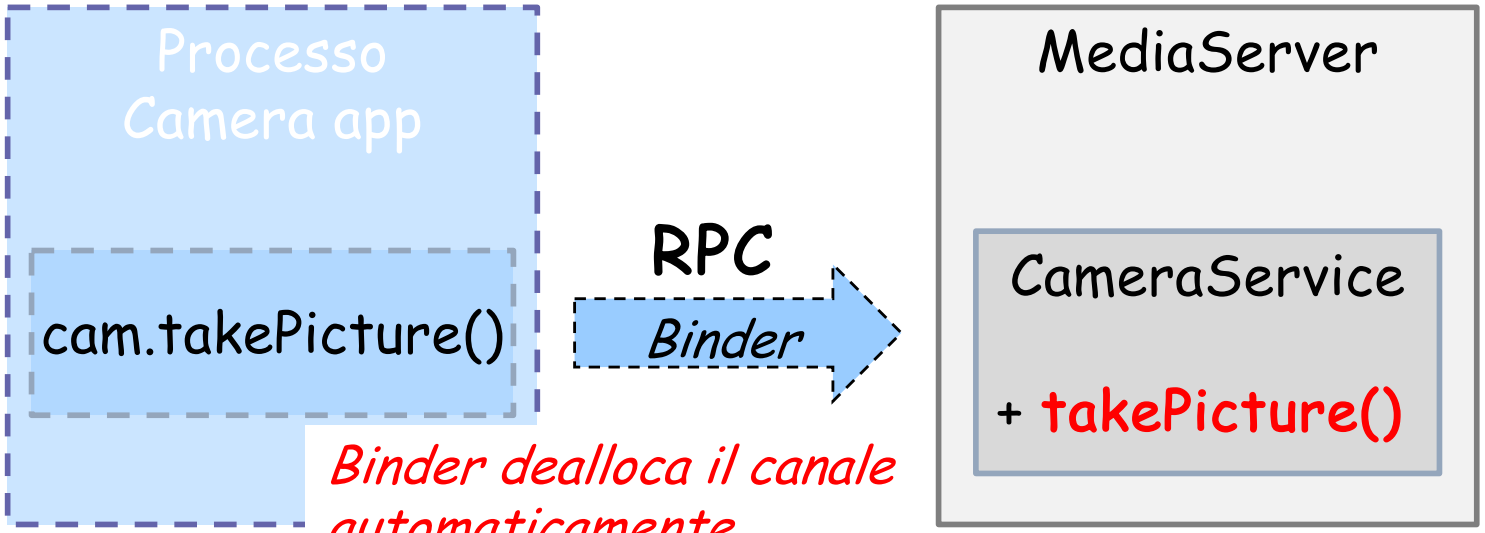
IPC in Android

- **Binder** è il meccanismo IPC di Android per la interazione tra app
- Basato su **Remote Procedure Call** (RPC)
- Usato per inviare Intent e chiamare i servizi di sistema

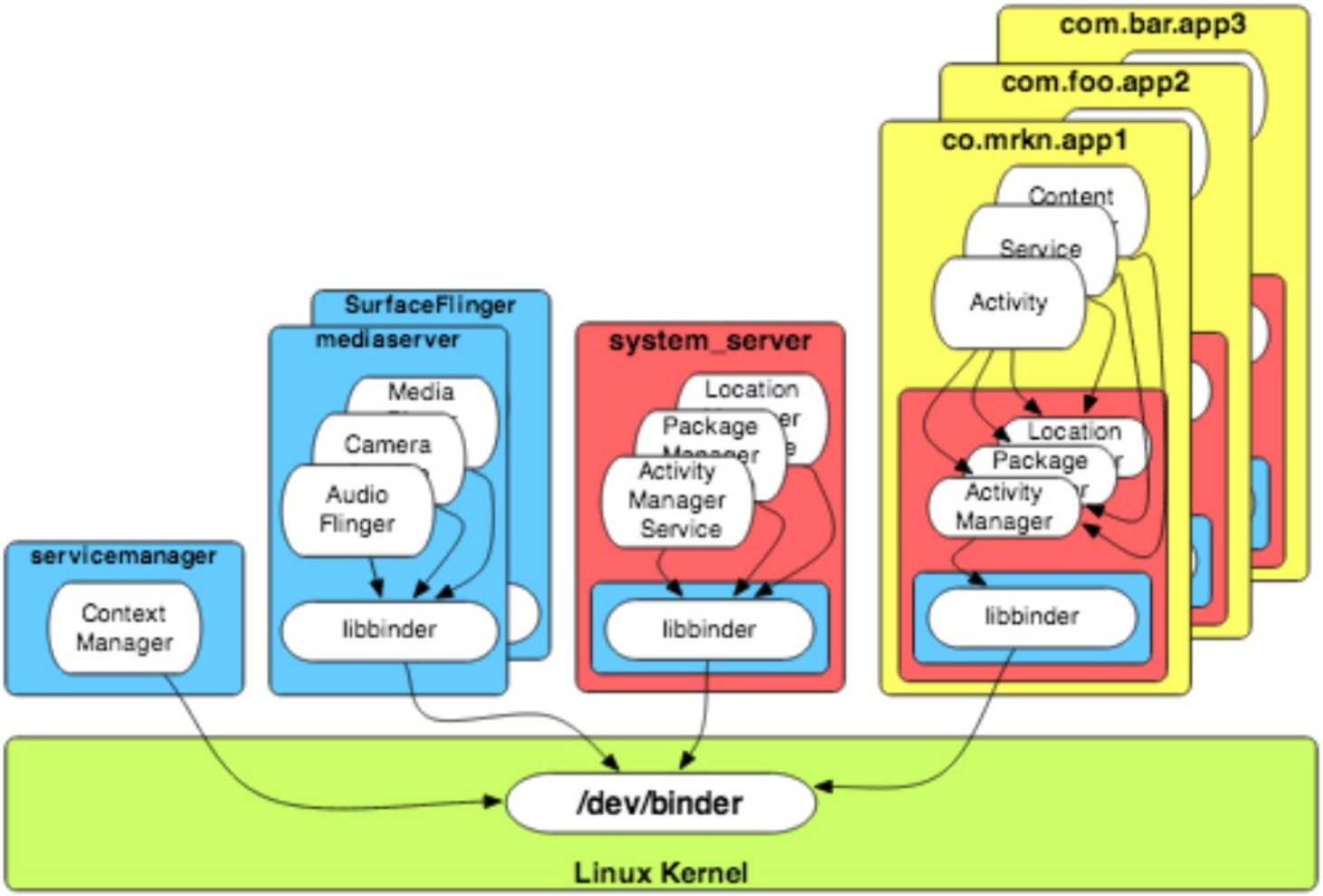


IPC in Android

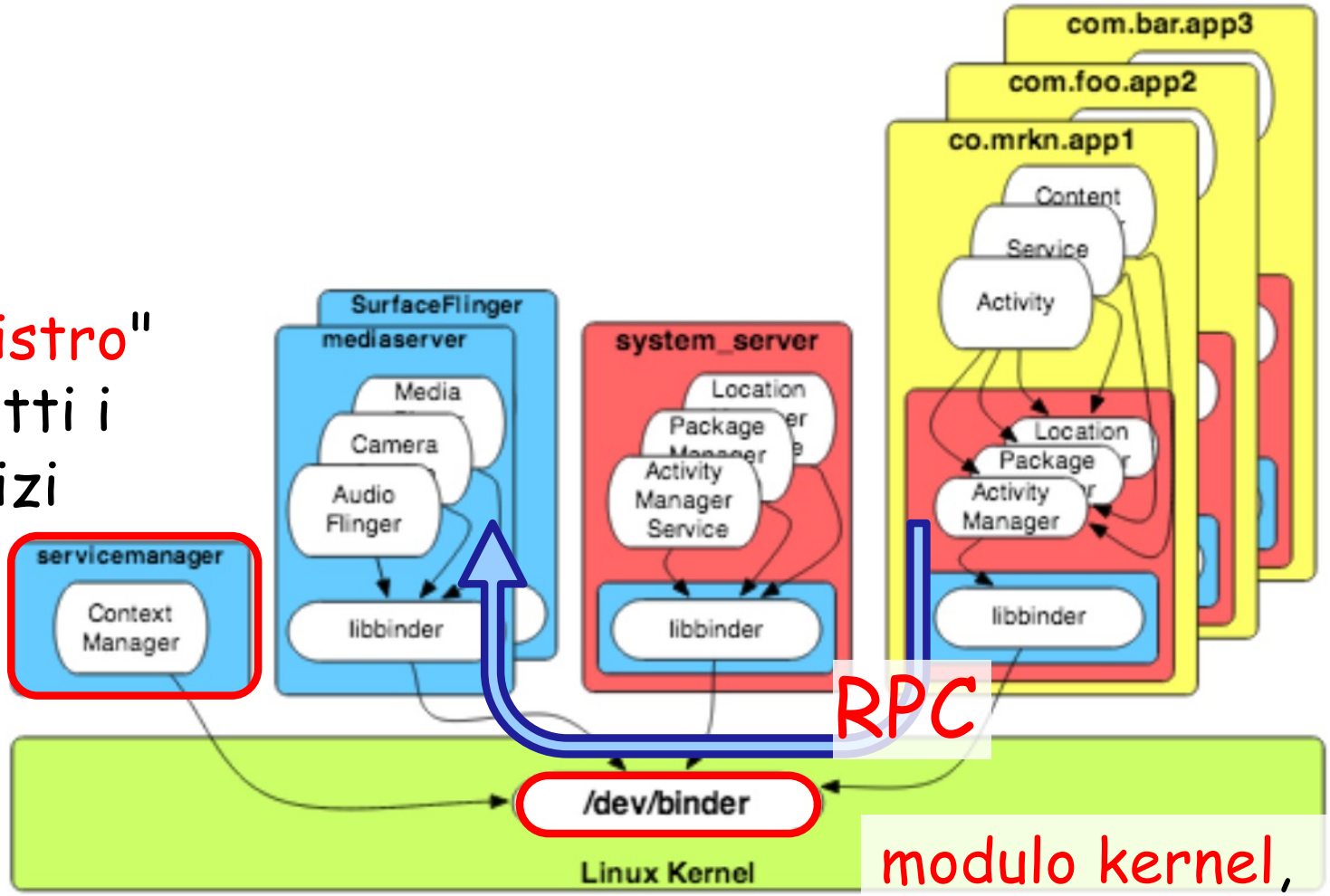
- Android non utilizza le **IPC UNIX System V** (semafori, shared memory, message queue)
- Esse non forniscono un comportamento "robusto" per **ripulire le risorse** di **app buggate o malevole**



IPC in Android



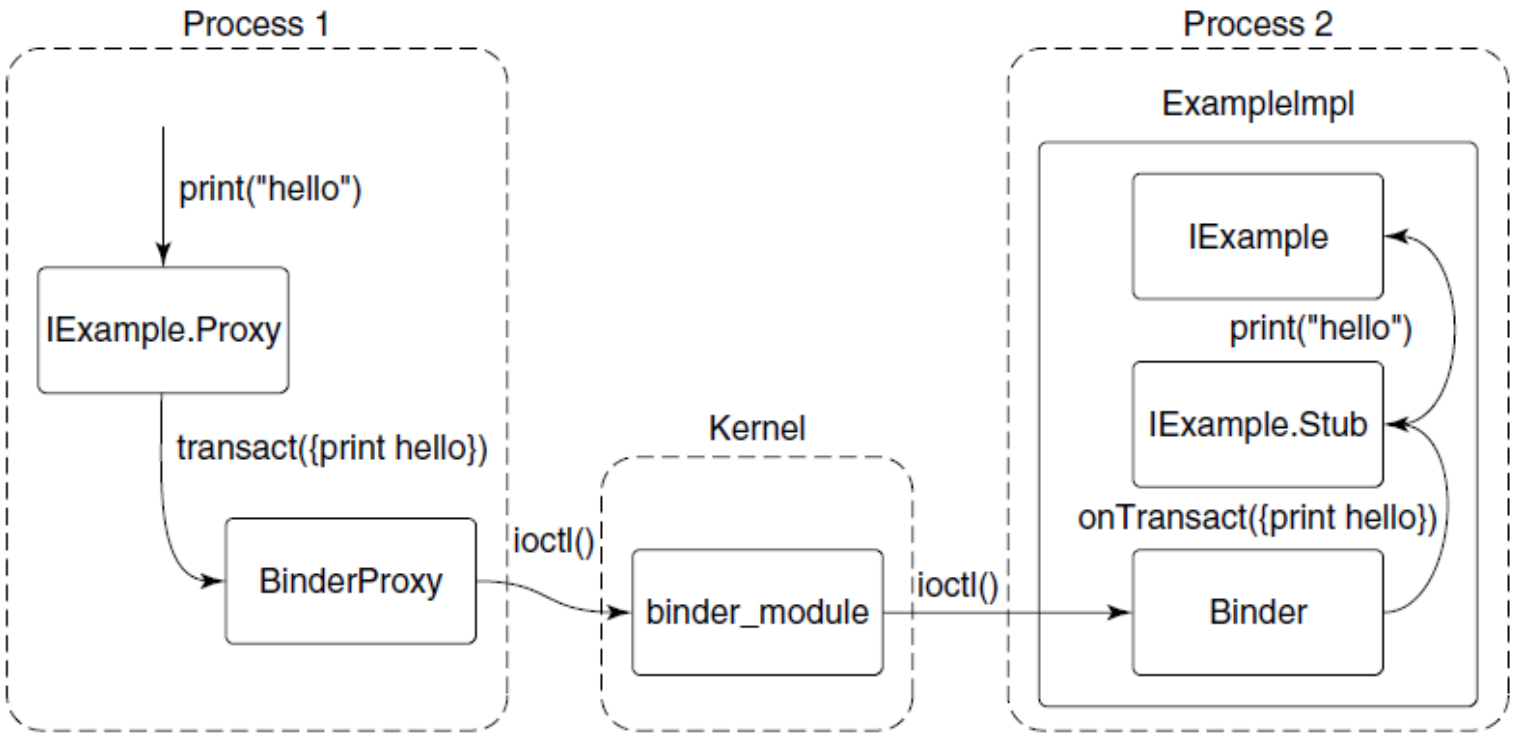
IPC in Android



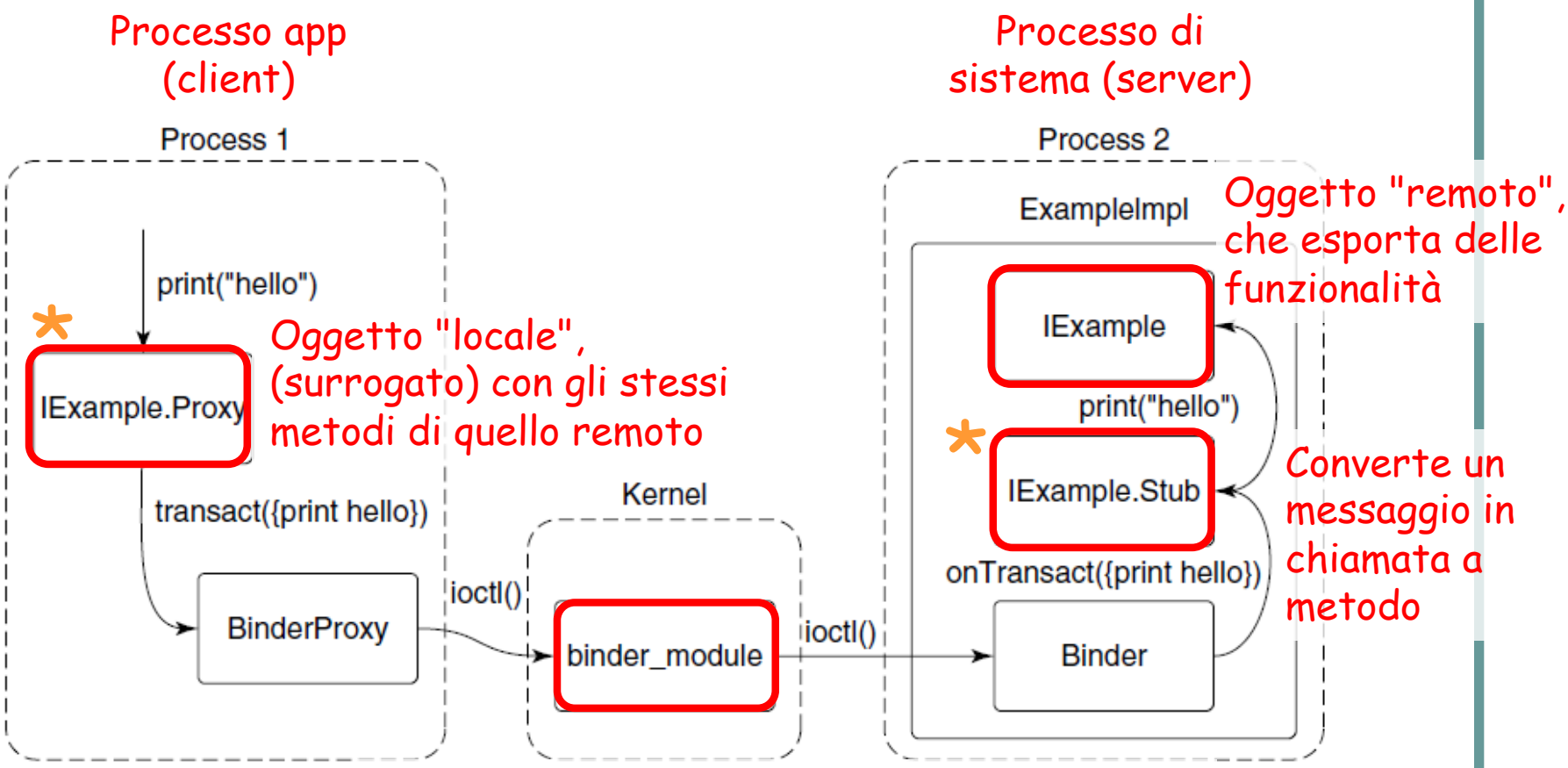
"registro" di tutti i servizi

modulo kernel, richiamato tramite file in /dev

Remote Procedure Call (RPC)



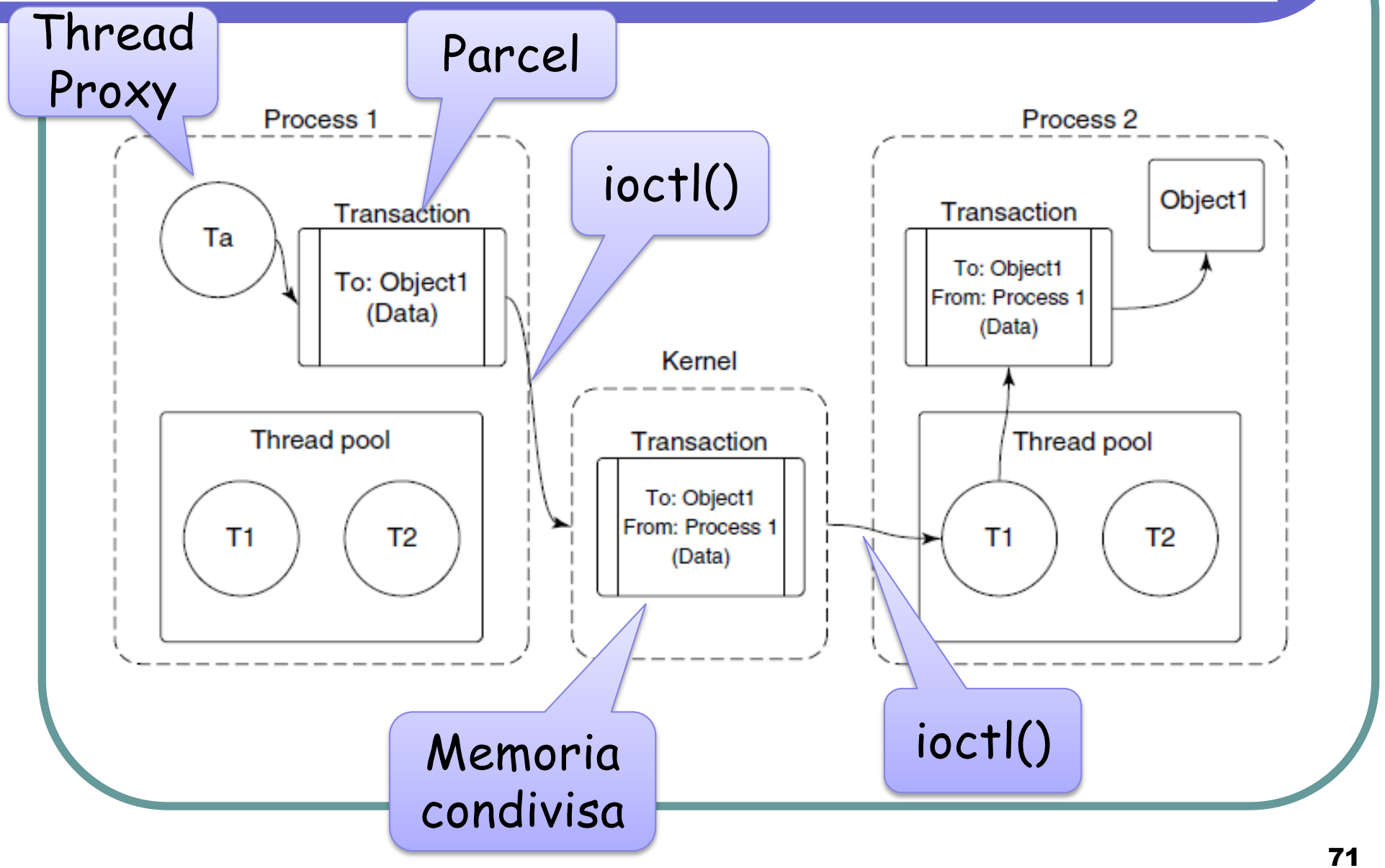
Remote Procedure Call (RPC)



Binder converte la **chiamata a un metodo** (con i parametri di ingresso e uscita) in **messaggi (parcel)** a basso livello

* Classi auto-generate

Binder IPC transactions



Android Interface Definition Language (AIDL)

- Il programmatore definisce in file **AIDL** la **interfaccia** della remote procedure call (metodi/parametri)
- Un **compilatore** elabora il file AIDL, producendo:
  - Una classe **Proxy**, utilizzata dalla app **client**
  - Una classe **Stub**, utilizzata nel server, e classe-padre del servizio

AIDL

```
package com.example

interface IExample {
        void print(String msg);
}
```

IExample.Stub
+ print(String);
+ onTransact(Parcel);

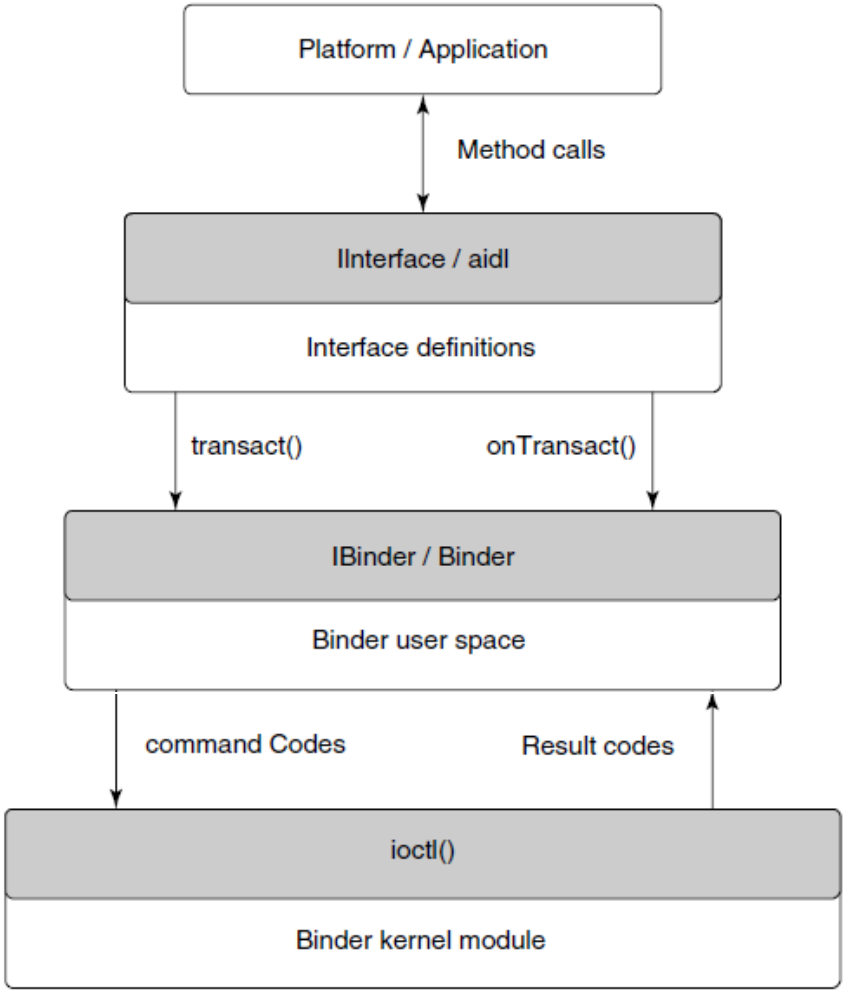
Metodo **astratto**, il programmatore lo implementa nel server con una **classe-figlia**

Metodo **auto-generato**, che riceve messaggi e invoca print()

IExample.Proxy
+print(String);

Metodo **fittizio**, **auto-generato**, converte la chiamata in un messaggio

Binder IPC Architecture



Le applicazioni invocano metodi su un oggetto **Proxy**

A tempo di compilazione, la classe **Proxy** (in **linguaggio AIDL**) è generata in automatico e collegata all'app

In **user-space**, Binder è una libreria condivisa **libbinder.so**, con una API object-oriented

In **kernel-space**, Binder è un modulo kernel accessibile tramite il file **/dev/binder** e la system call **ioctl()**