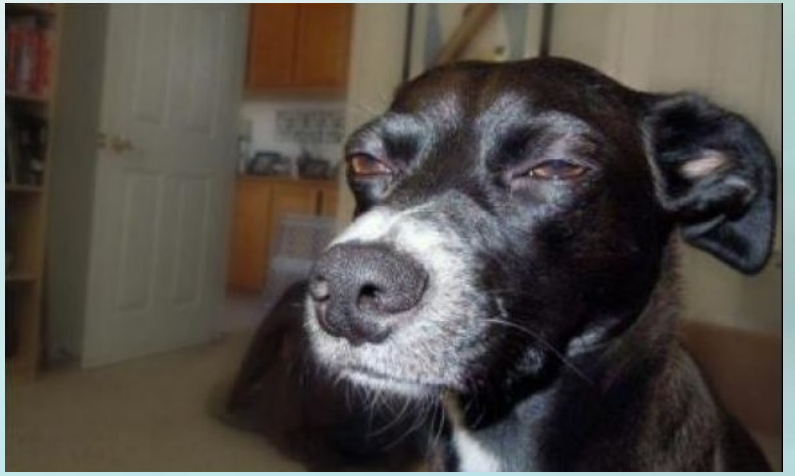
# WakeLocks

Iván Aguilera Calle
Daniel García Moreno

# Índice

# INTRODUCCIÓN

- ¿Qué es un wakelock?
  - Proceso en segundo plano.
  - Fuerza a tener la CPU activa.
- Wakelock ≠ proceso sincronización cuentas.



Significado: mantenerse despierto

# MOTIVACIÓN

- Modos de energía utilizados por el Kernel:
  - **Modo Idle**: procesador totalmente inactivo (no se ejecutan instrucciones).
  - **Modo bajo consumo**: procesador en baja frecuencia y voltaje.
  - **Modo alto rendimiento**: procesador en alta frecuencia y voltaje.

- Suspender ordenador de sobremesa
  - Kernel entra en modo suspensión.
    - Las aplicaciones se interrumpen.
    - Se apagan los cores, pantalla, discos…
    - Se mantiene activa la RAM.

# MOTIVACIÓN


- La naturaleza de Android (carácter multimedia) obliga a necesitar un mecanismo que no interrumpa la ejecución de los procesos en segundo plano.
  p.e: whatsapp, reproductor de música, llamadas, ejecuciones críticas…
- Este mecanismo se llama Wakelock.
- Para ahorrar batería, los dispositivos Android se duermen rápidamente después de un corto periodo de inactividad del usuario.
- El 27,2% de las apps de Google Play utiliza wakelocks.
- El 61,3% de las apps de código abierto sufren bugs por el uso de wakelocks.
  - Pueden causar “crash”, gasto extra energía…

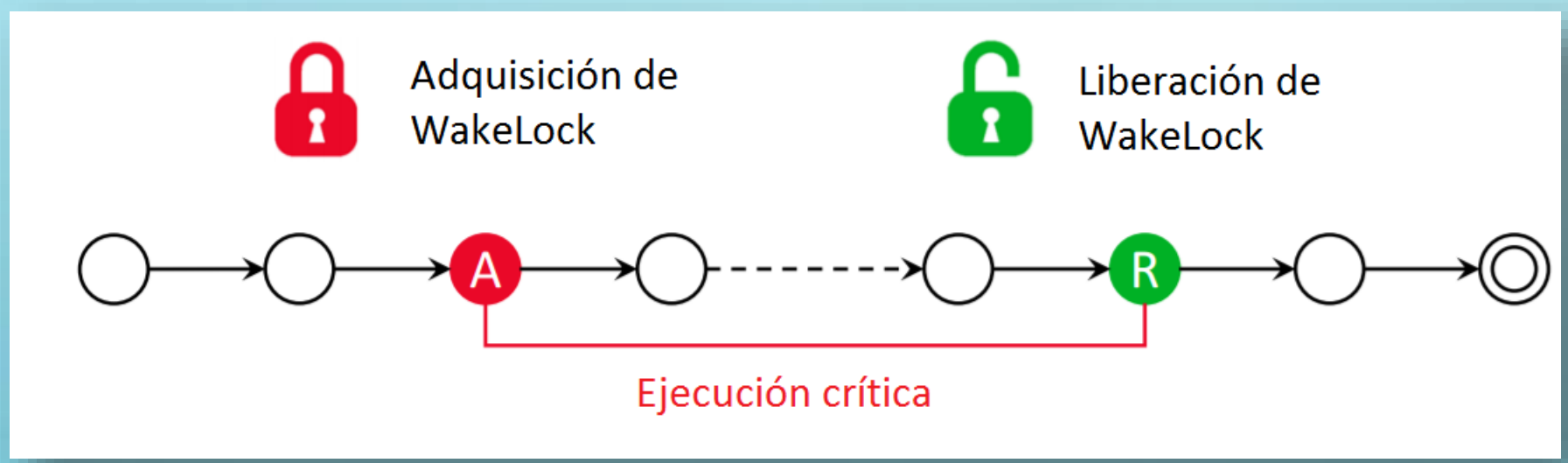
# TIPOS DE WAKELOCKS

- Wakelock parcial:
  - CPU activa.
  - Pantalla no necesariamente encendida (puede estar apagada).
- Wakelock total:
  - CPU activa.
  - Pantalla se queda activa.

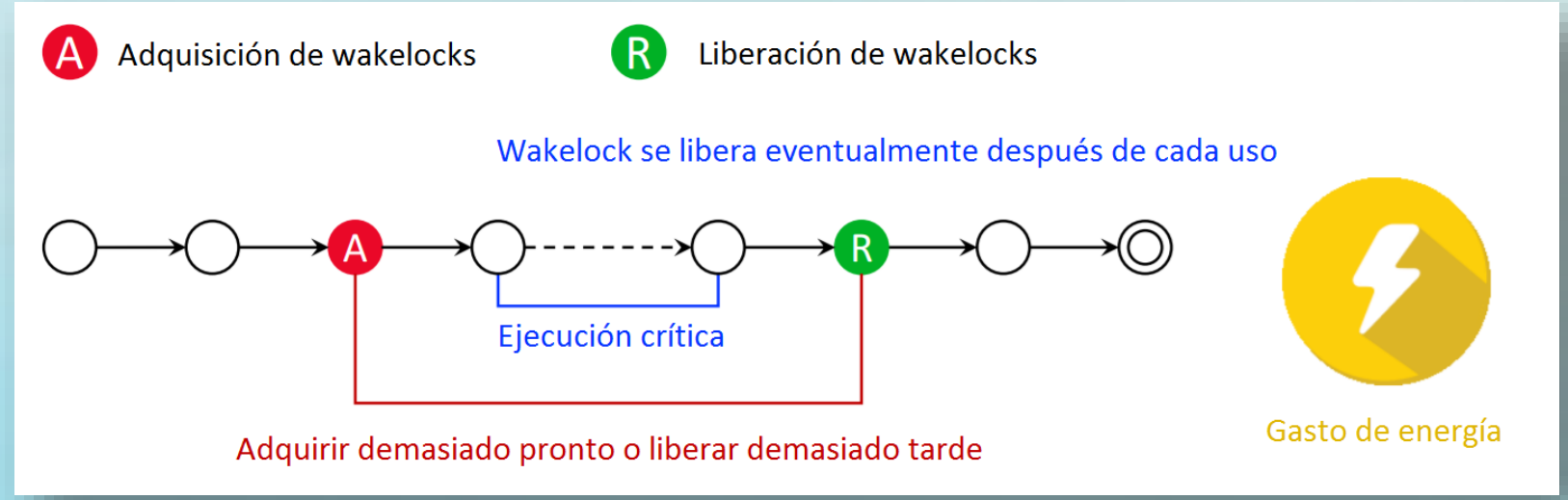
| TIPO | CPU | PANTALLA | TECLADO |
|---|---|---|---|
| Parcial | Encendida | Apagado | Apagado |
| Total | Encendida | Brillante | Encendido |
| Pantalla tenue | Encendida | Tenue | Apagado |
| Pantalla brillante | Encendida | Brillante | Apagado |
| Sensor de proximidad | La pantalla se apaga cuando el sensor se activa | | |

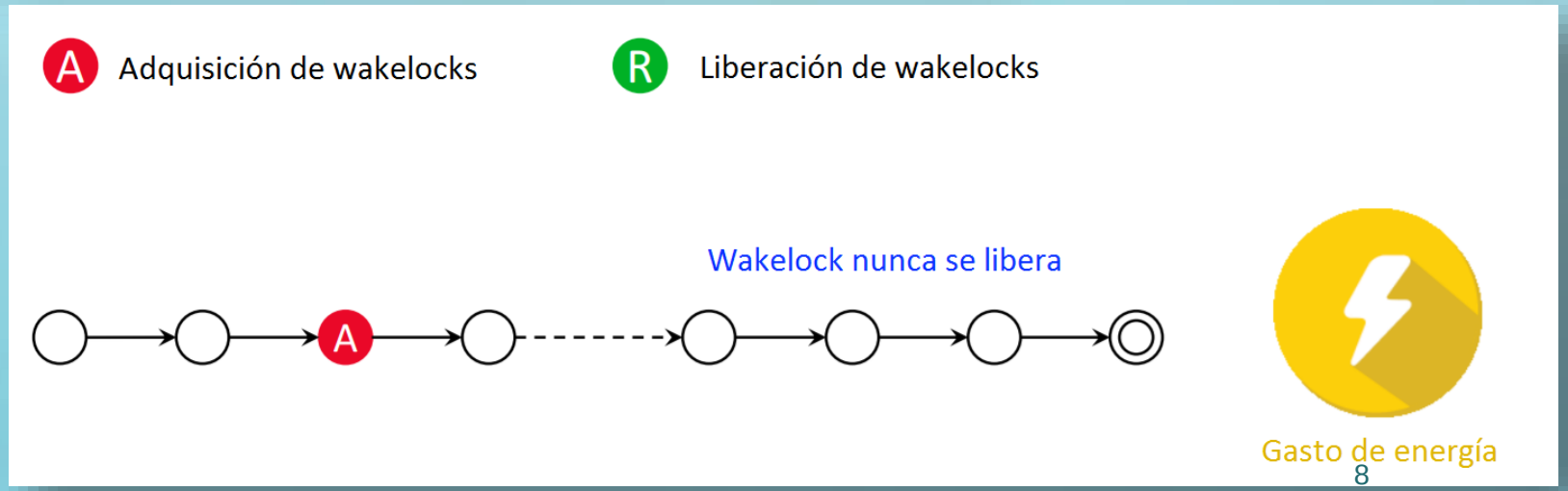
# USO DE WAKELOCKS

- Utilizamos los Wakelocks para mantener cierto hardware encendido por un tiempo determinado.
  - CPU
  - Pantalla
- Ejecución de secciones críticas.
  - Tareas interrumpibles (uninterruptable tasks)

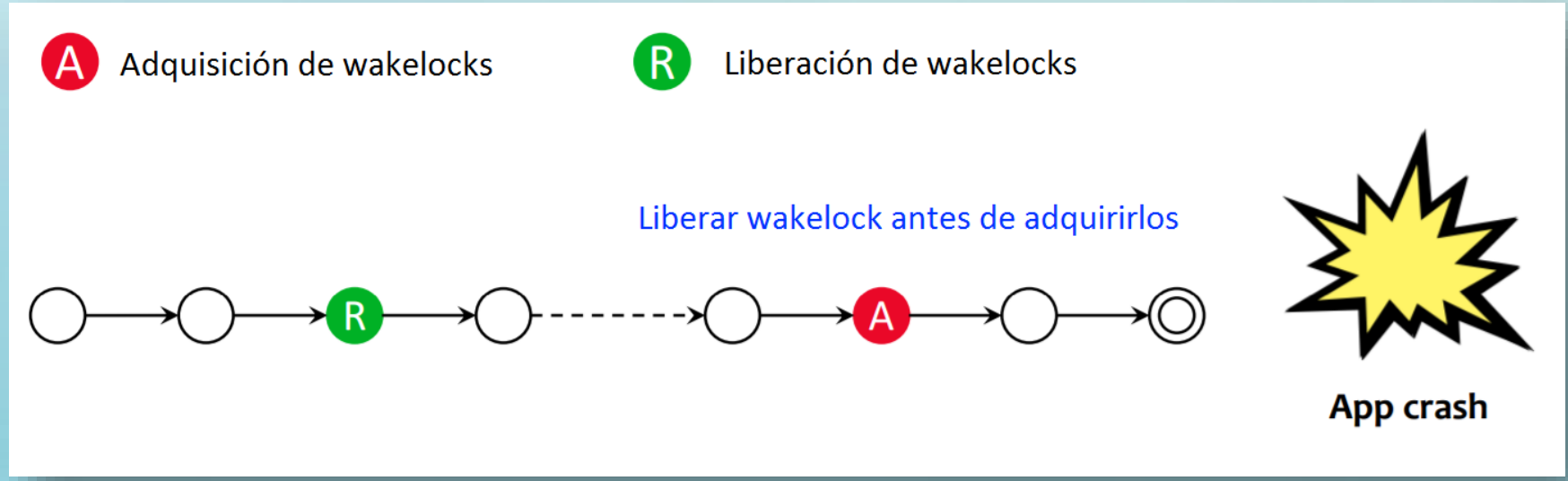


# USOS INCORRECTOS DE WAKELOCKS



# USO INCORRECTO DE WAKELOCKS



# APPS QUE UTILIZAN WAKELOCKS

- Dejaba funcionando los procesos de la cámara.
- Aunque no estuviera funcionando la cámara, consumía como si lo estuviese.


- Mantiene la pantalla activa sin que la tengamos que tocar.
- Wakelock “benigno”.

# APPS QUE UTILIZAN WAKELOCKS

- Mantiene la pantalla activa.
- Adquiere el wakelock cuando se abre un libro.
- Libera el wakelock cuando se vuelve al menú de la aplicación.
- Si el usuario no pasa de página en 5 minutos el wakelock se libera automáticamente.

- Mantiene la pantalla activa.
- Adquiere el wakelock cuando se empieza a navegar.
- Libera el wakelock cuando el usuario deja de navegar o ha alcanzado su destino.

# ESTRUCTURA EN EL KERNEL

```
#include <linux/ktime.h>
#include <linux/device.h>

struct wake_lock {
    struct wakeup_source ws;
}
```

wakelock.h

```
#include <linux/wakelock.h>

struct wakeup_source {
    const char          *name;
    struct list_head    entry;
    spinlock_t          lock;
    struct timer_list   timer;
    unsigned long       timer_expires;
    ktime_t total_time;
    ktime_t max_time;
    ktime_t last_time;
    ktime_t start_prevent_time;
    ktime_t prevent_sleep_time;
    unsigned long       event_count;
    unsigned long       active_count;
    unsigned long       relax_count;
    unsigned long       expire_count;
    unsigned long       wakeup_count;
    bool            active:1;
    bool            autosleep_enabled:1;
};
```

pm_wakeup.h

# FUNCIONES DEL KERNEL

```
static inline void wake_lock_init(struct wake_lock *lock, int type, const char *name){
    wakeup_source_init(&lock->ws, name);
}

static inline void wake_lock_destroy(struct wake_lock *lock){
    wakeup_source_trash(&lock->ws);
}

static inline void wake_lock(struct wake_lock *lock){
    __pm_stay_awake(&lock->ws);
}

static inline void wake_lock_timeout(struct wake_lock *lock, long timeout){
    __pm_wakeup_event(&lock->ws, jiffies_to_msecs(timeout));
}

static inline void wake_unlock(struct wake_lock *lock){
    __pm_relax(&lock->ws);
}

static inline int wake_lock_active(struct wake_lock *lock){
    return lock->ws.active;
}
```

# USO EN EL ESPACIO DE USUARIO

- Un proceso puede adquirir un wakelock escribiendo el nombre en el directorio:

  /sys/power/wake_lock

- Un proceso puede liberar un directorio escribiendo el nombre en el directorio:

  /sys/power/wake_unlock

# EJEMPLOS DE WAKELOCKS EN EL KERNEL

- **Wlan_rx** → lo utiliza el Kernel cuando se están enviando/recibiendo datos por WiFi.
- **PowerManagerService** → contenedor de los wakelocks parciales.
- **Sync** → Activo cuando el proceso Sync está corriendo.
- **Alarm_rtc** → activado cuando un proceso comprueba algo periódicamente.
- **Main** → mantiene el Kernel despierto. Es el último en ser liberado cuando el dispositivo entra en modo suspensión.

# USO DE WAKELOCKS EN APLICACIONES JAVA

```
PowerManager.WakeLock newWakeLock (int levelAndFlags, String tag);
```

```
public final class PowerManager.WakeLock extends Object{

}
```

```
void acquire (); //Adquirir wakelock
void acquire (long timeout); //liberar  wakelock después de un timeout
boolean isHeld (); //Comprobar si un wakelock está adquirido pero no liberado
void release (int flags); //liberar wakelock con flag RELEASE_FLAG_WAIT_FOR_NO_PROXIMITY
void release ();
void setReferenceCounted (boolean value); //TRUE  -> por cada acquire() debe hacerse un release()
                                          //FALSE -> solo se necesita un release()
```

- Dejar activada la CPU para hacer tareas

```
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
 PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, "Mi wake lock");
 wl.acquire();
   ...
 wl.release();

    TIPOS
PARTIAL_WAKE_LOCK
SCREEN_DIM_WAKE_LOCK
SCREEN_DIM_WAKE_LOCK
FULL_WAKE_LOCK
```

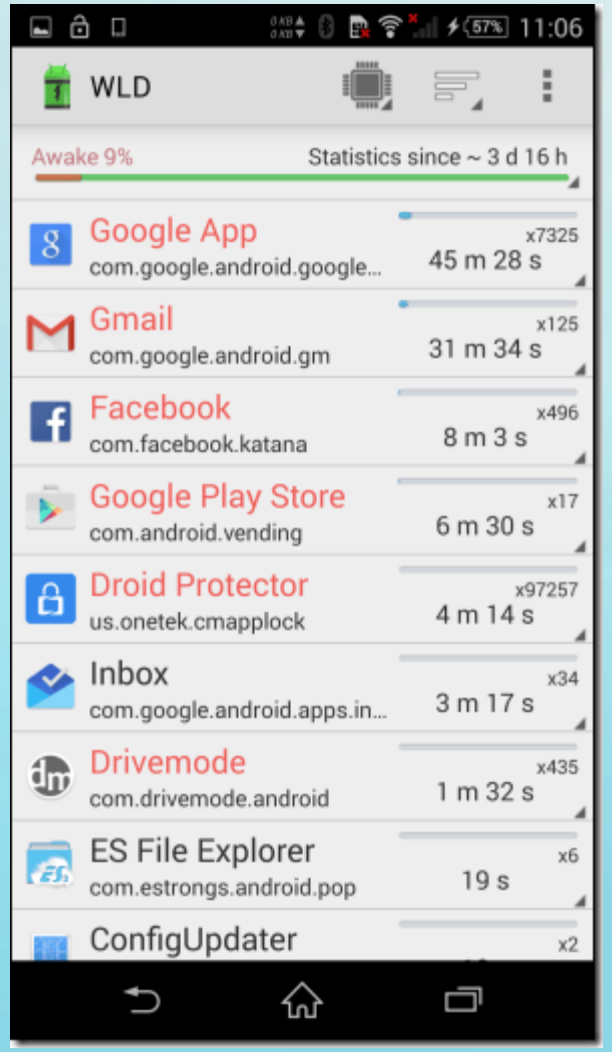
# DETECCIÓN WAKELOCK USANDO ADB

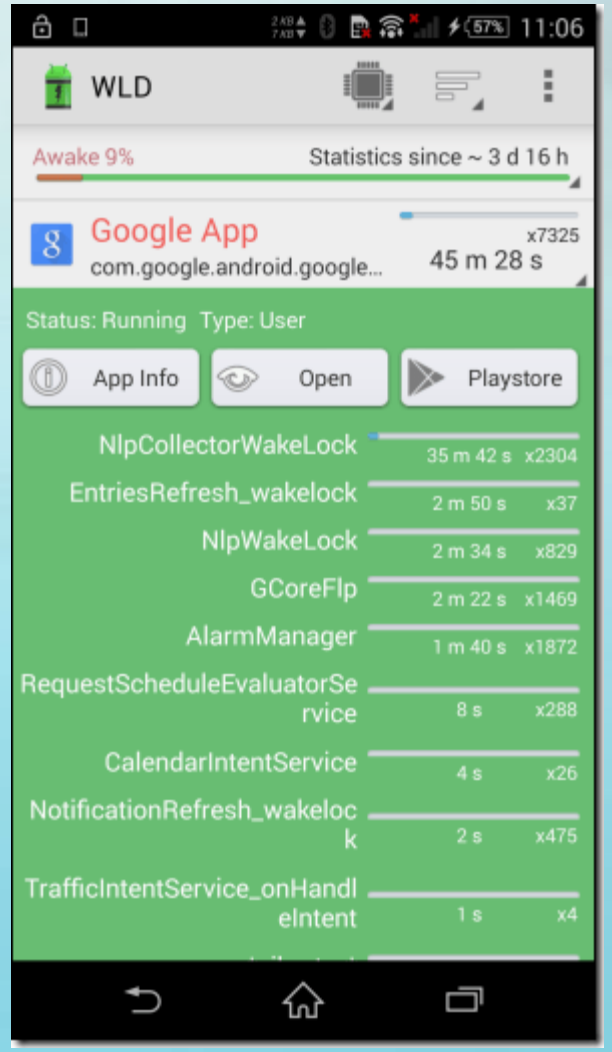
- adb shell cat /proc/wakelocks

| name | count | expire_count | wake_count | active_since | total_time |
|---|---|---|---|---|---|
| "PowerManagerService" | 1502 | 0 | 0 | 0 | 337817677431 |
| "main" | 15 | 0 | 0 | 0 | 984265842688 |
| "alarm" | 1512 | 0 | 792 | 0 | 217778251643 |
| "radio-interface" | 16 | 0 | 0 | 0 | 16676538930 |
| "alarm_rtc" | 804 | 4 | 0 | 0 | 1204136324759 |
| "gps-lock" | 1 | 0 | 0 | 0 | 10753659786 |

# DETECCIÓN WAKELOCKS USANDO APP

- ROOT
- Uso de la CPU y consumo de la aplicación



# CONTROVERSIAS

- Implementación de wakelock causó controversias entre los desarrolladores de Android y la comunidad del Kernel de Linux.
- Problema:
  - Parche de Android a la rama principal del Kernel de Linux donde se implementaba los wakelocks → rechazado
    - Duplicidad del código del API pm_qos.
    - Deshabilitación de la interfaz /sys/power/state.
    - No se puede recuperarse si un proceso del espacio de usuario termina sin liberar un wakelock.
      - IMPORTANTE: Android no libera wakelocks cuando un proceso termina. Debe ser el proceso explícitamente el que se encargue de liberar los wakelocks.

# CONTROVERSIAS

- Problema:
  - Parche de Android a la rama principal del Kernel de Linux donde se implementaba los wakelocks → rechazado (CONTINUACIÓN)
    - Código desarrollado a puerta cerrada.

No seguir el esquema de desarrollo del Kernel de Linux

  - "Post early and often": desarrollar código cerrado evita recibir feedback de la comunidad de desarrollo.
  - "Upstream first": comercializarse antes de subirse a la rama principal.
  - "Desarrollo para todos": desarrolladores de sistemas empotrados resuelven problemas específicos en poco tiempo → código sucio, no sujeto a revisiones.

# Panorama actual

- A partir del Kernel de Linux 3.5 se incorpora la noción de wakelock a la rama principal.
- De similar funcionalidad a la de Android.
- Se esperaba que a partir de esta versión Android fuera capaz de usar esta nueva implementación.

# BIBLIOGRAFÍA


- Wakelock Android - https://developer.android.com/reference/android/os/PowerManager.WakeLock.html https://developer.android.com/training/scheduling/wakelock.html
- Controversias y uso Wakelocks - http://elinux.org/Android_Power_Management
- Fuentes del kernel - https://srcxref.dacya.ucm.es/source/xref/linux-kernel-android-kitkat/include/linux/wakelock.h
- Controversias - https://lwn.net/Articles/318611/

# BIBLIOGRAFÍA


- Aplicaciones con Wakelocks - https://w3c-webmob.github.io/wake-lock-use-cases
- Wakelock Detector - https://www.elandroidelibre.com/2014/06/que-son-los-wakelocks-y-como-evitarlos.html
- Tipos Wakelock - http://sccpu2.cse.ust.hk/andrewust/files/ELITE-FSE2016.pdf
- Suspend Blockers - http://static.lwn.net/images/pdf/suspend_blockers.pdf

# ¡MUCHAS GRACIAS!

## ¿DUDAS?