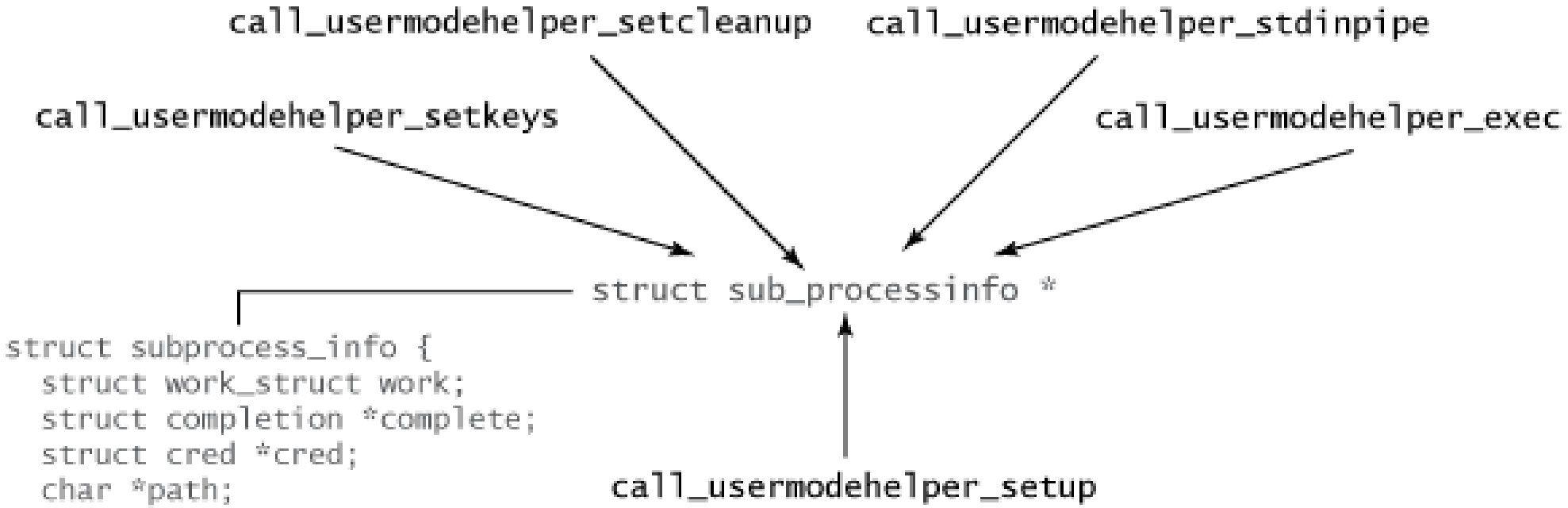
# UserMode_Helper API

- `#include <linux/kmod.h>`
- API básico
  - `call_usermodehelper_setup` — `Preparar manejador para helper`
  - `call_usermodehelper_setcleanup` — `Establecer cleanup function`
  - `call_usermodehelper_stdinpipe` — `Create stdin pipe`
  - `call_usermodehelper_exec` — `Exec(call) proceso usuario`
    - Tres posibles modos:
      - **UMH_NO_WAIT (no se espera)**
      - **UMH_WAIT_PROC (complete proceso usuario)**
      - **UHM_WAIT_EXEC (invocación)**
    - Hijo de `keventd`
- API simplificado
  - `call_usermodehelper` — `call`
  - `call_usermodehelper_pipe` — `call + stdin pipe`


- **http://www.ibm.com/developerworks/linux/library/l-user-space-apps/index.html**

# UserMode_Helper API

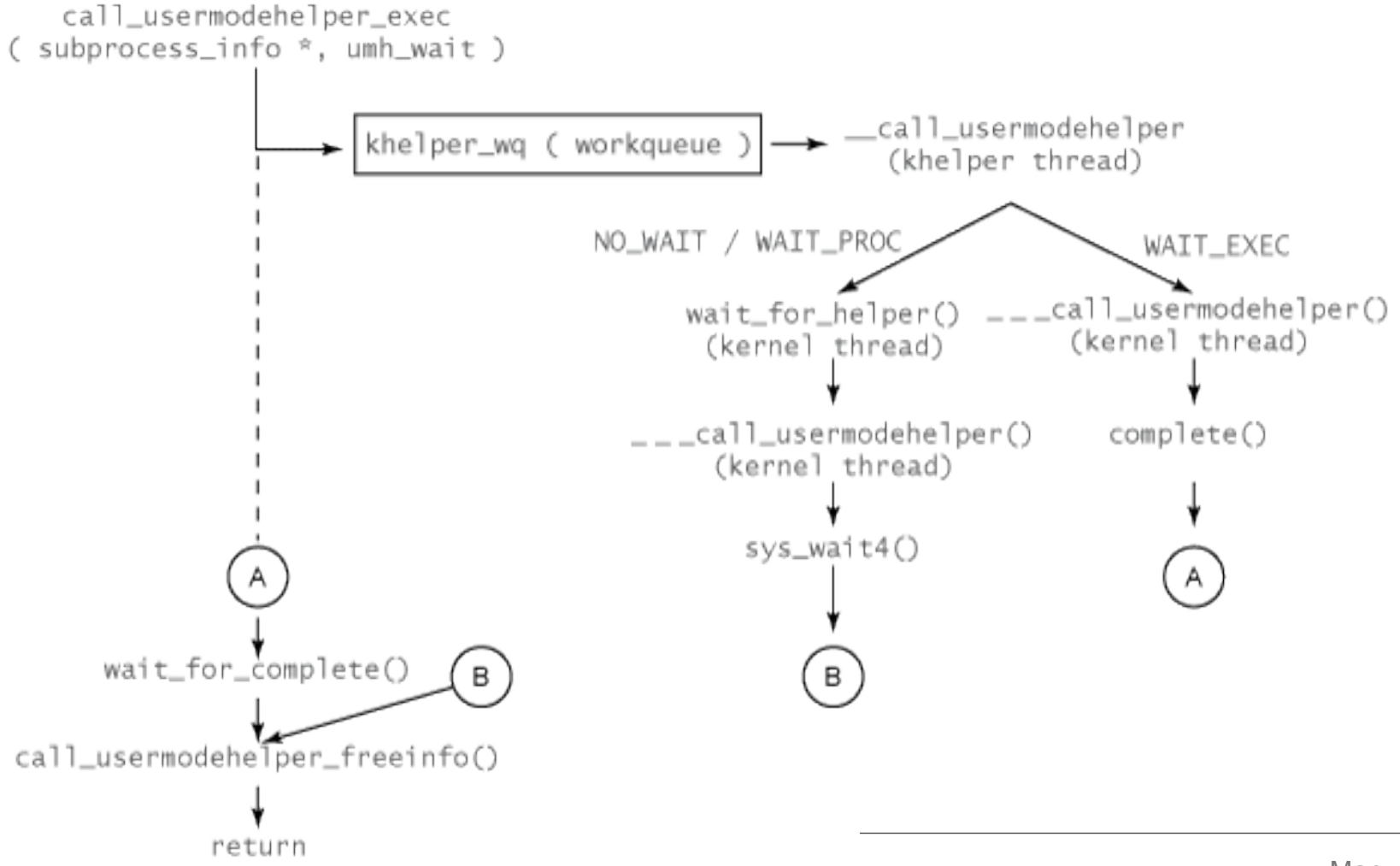



```
struct subprocess_info {
  struct work_struct work;
  struct completion *complete;
  struct cred *cred;
  char *path;
  char **argv;
  enum umh_wait wait;
  int retval;
  struct file *stdin;
  void (*cleanup)(char **argv, char **envp);
};
```

Manuel Prieto-Matías
Universidad Complutense de Madrid

# UserMode_Helper API

- **kernel_execve (init)**




Manuel Prieto-Matías
Universidad Complutense de Madrid

# UserMode_Helper API

```
#include <linux/kmod.h>
static int ejemplo( void ){
struct subprocess_info *sub_info;
char *argv[] = { "/usr/bin/logger", "hola AISO!",NULL };
static char *envp[] = {
      "HOME=/",
      "TERM=linux",
      "PATH=/sbin:/bin:/usr/sbin:/usr/bin", NULL };

sub_info = call_usermodehelper_setup( argv[0], argv, envp,GFP_ATOMIC);
/* GFP_ATOMIC (no sleep) o GFP_KERNEL (sleep posible)(kzalloc) */

if (sub_info == NULL) return -ENOMEM;

return call_usermodehelper_exec( sub_info, UMH_WAIT_PROC );
```

# UserMode_Helper API

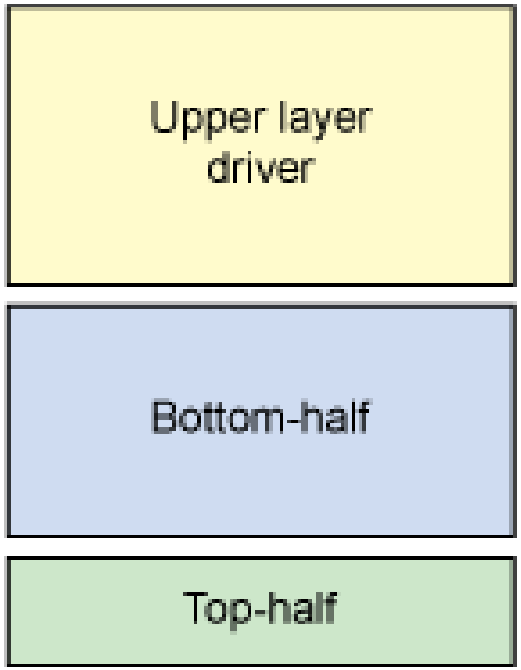
- `#include <linux/kmod.h>`

```
static int ejemplo( void ){
 char *argv[] = { "/usr/bin/logger", "hola AISO!",NULL };
static char *envp[] = {
      "HOME=/",
      "TERM=linux",
      "PATH=/sbin:/bin:/usr/sbin:/usr/bin", NULL };

return call_usermodehelper( argv[0], argv, envp,UMH_WAIT_PROC );
```

# Top-halfs y bottom-halfs

- Es necesario diferir el trabajo...




- http://www.ibm.com/developerworks/linux/library/l-tasklets/

# Top-halfs y bottom-halfs

- Mecanismos para diferir el trabajo...
  - Softirq **(kernel/softirq.c)**
    - Estático (32 entradas) – desde linux 2.3
    - Actualmente sólo se usan 9
    - **ksoftirqd**
  - Tasklets **(include/linux/interrupt.h)**
    - Dinámico – desde linux 2.3
    - 2 Niveles **TASKLET_SOFTIRQ, HI_SOFTIRQ**
  - Workqueues **(include/linux/workqueue.h)**
    - Dinámico – desde linux 2.5
    - Process Context

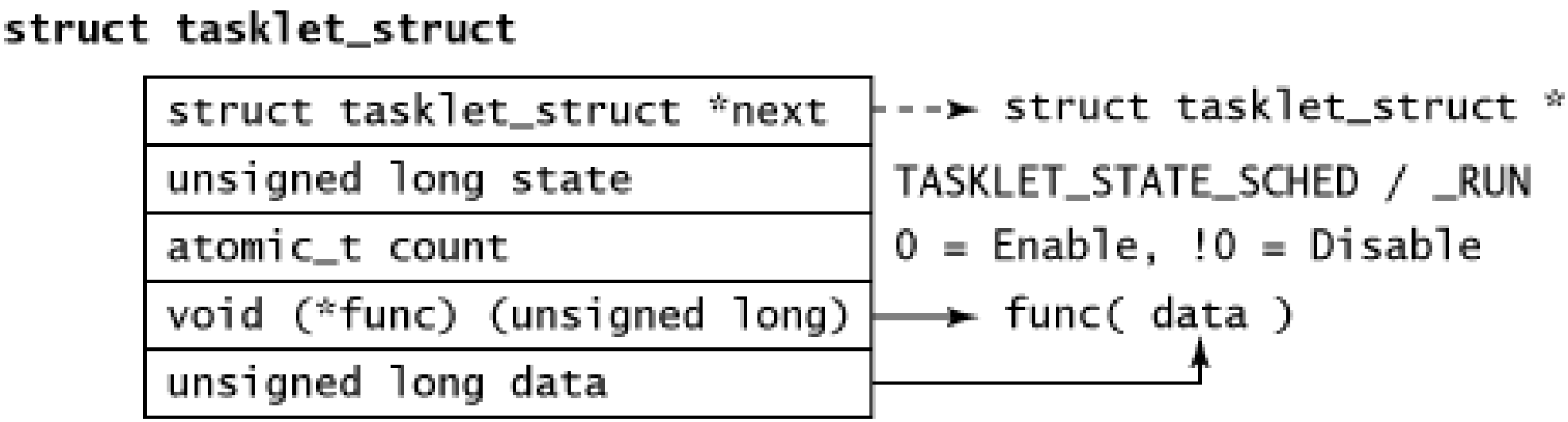
# Tasklets

- Declaración
  - `DECLARE_TASKLET( name, func, data );`
  - `DECLARE_TASKLET_DISABLED( name, func, data);`
- Funciones
  - `void tasklet_init( struct tasklet_struct *, void (*func) (unsigned long),unsigned long data );`
  - Desabilitar (no scheduling)
    - `void tasklet_disable_nosync( struct tasklet_struct * );`
    - `void tasklet_disable( struct tasklet_struct * );`
  - Habilitar
    - `void tasklet_enable( struct tasklet_struct * );`
    - `void tasklet_hi_enable( struct tasklet_struct * );`

# Tasklets

- Scheduling (Run)
  - `void tasklet_schedule( struct tasklet_struct * );`
  - `void tasklet_hi_schedule( struct tasklet_struct * );`
- Stop
  - `void tasklet_kill( struct tasklet_struct * );`
  - `void tasklet_kill_immediate( struct tasklet_struct *, unsigned int cpu );`

# Tasklets




Manuel Prieto-Matías
Universidad Complutense de Madrid

# UserMode_Helper API

- `#include <linux/interrupt.h>`

```
char my_tasklet_data[]="hemos llamado a aiso_tasklet";
/* Bottom Half Function */
void my_tasklet_function( unsigned long data ){
  printk( "%s\n", (char *)data );
  return;
}
DECLARE_TASKLET( my_tasklet, my_tasklet_function,
            (unsigned long) &my_tasklet_data );
int init_module( void ){
  /* Schedule Bottom Half */
  tasklet_schedule( &my_tasklet );
  return 0;
}
void cleanup_module( void ){
  /* Stop the tasklet before we exit */
  tasklet_kill( &my_tasklet );
  return;
}
```