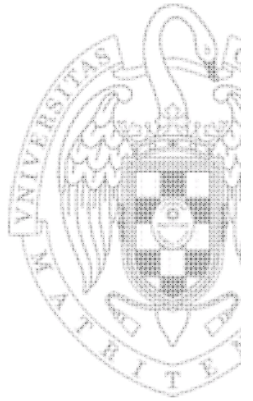
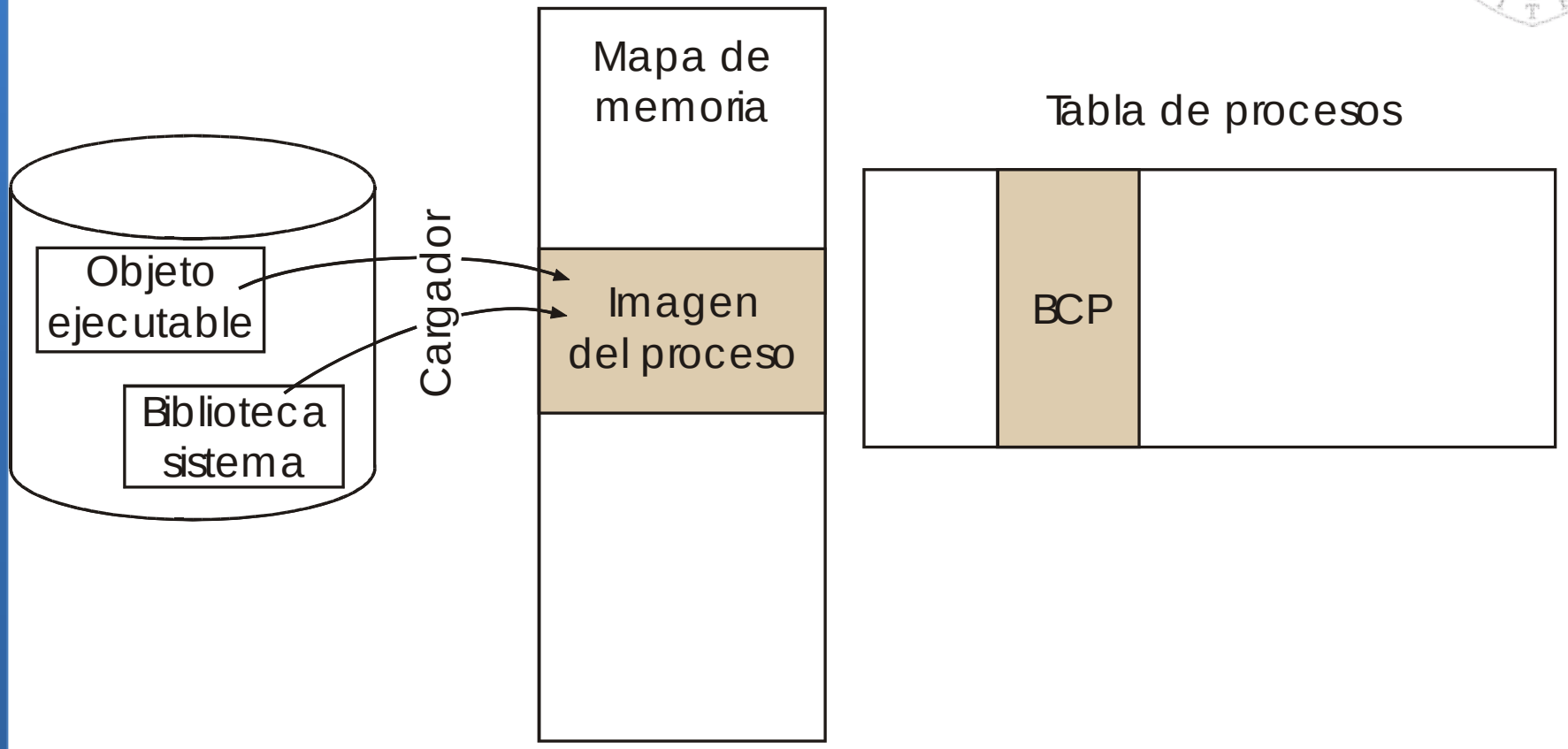


Servicios POSIX

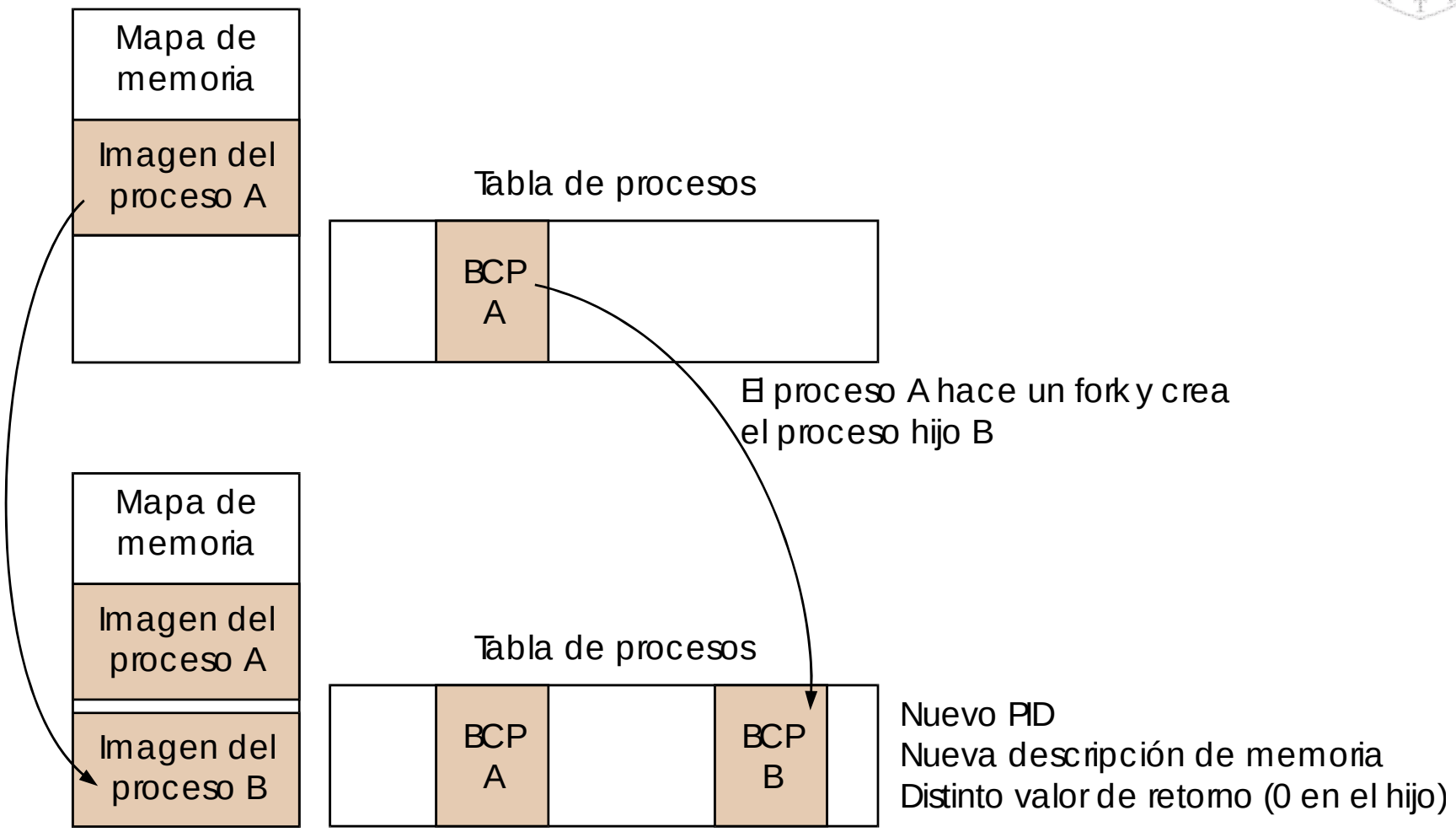
2016-2017

Formación de un proceso

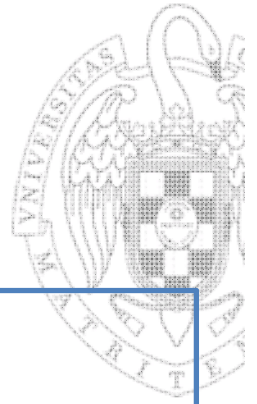


Servicios POSIX: fork

- Crea un proceso clonando al padre



fork. Crea un proceso



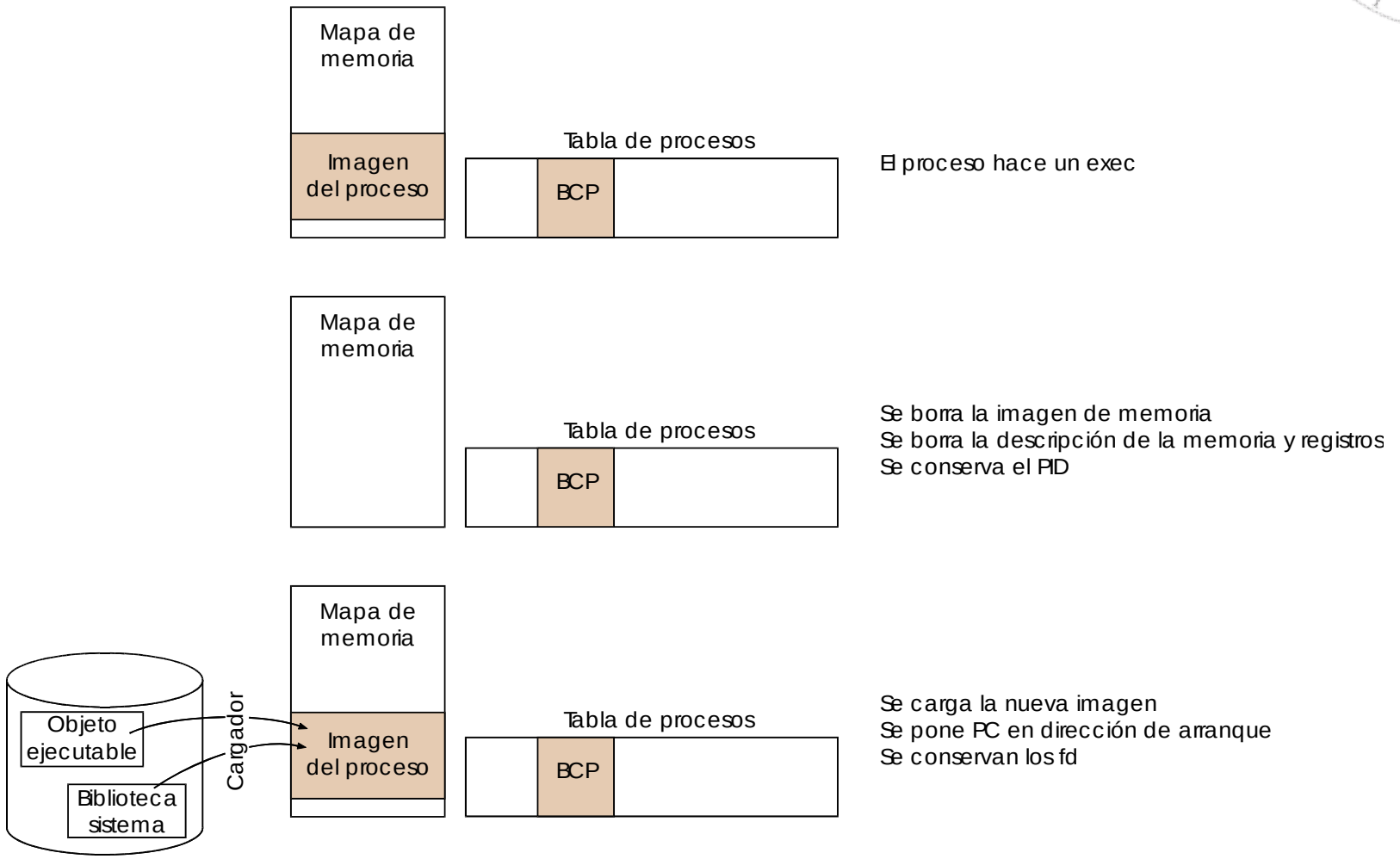
- Servicio:

  `pid_t fork(void);`
- Devuelve:
  - El identificador de proceso hijo al proceso padre y 0 al hijo
  - -1 el caso de error
- Descripción:
  - Crea un proceso hijo que ejecuta el mismo programa que el padre
  - Hereda los ficheros abiertos (se copian los descriptores).
  - Las alarmas pendientes se desactivan.
  - El proceso hijo sólo tiene un hilo.

```
void main()
{
   pid_t pid;
   …
   pid = fork();
   if (pid == 0) {
   /* proceso hijo */
   ...
   }
    else if (pid>0){
  /* proceso padre */
   ...
  }
  else{
 /* error al crear */
   ...
}
...
```

Servicios POSIX: exec

- Cambia el programa de un proceso



exec. Cambio del programa de un proceso

- Servicios:

```
int execl (const char *path, const char *arg, ...)
int excelp(const char *file, const char *arg, ...)
int execvp(const char *file, char *const argv[])
```

- Argumentos:
  - `path, file`: nombre del archivo ejecutable
  - `arg`: argumentos
- Descripción:
  - Devuelve -1 en caso de error, en caso contrario **no** retorna.
  - Cambia la imagen de memoria del proceso.
  - El mismo proceso ejecuta otro programa.
  - Los ficheros abiertos permanecen abiertos
  - Las señales con la acción por defecto seguirán por defecto, las señales con manejador tomarán la acción por defecto.

exit. Terminación de un proceso

- Servicios:

```
int exit(int status);
```

- Argumentos:
  - Código de retorno al proceso padre
- Descripción:
  - Finaliza la ejecución del proceso.
  - Se cierran todos los descriptores de ficheros abiertos.
  - Se liberan todos los recursos del proceso

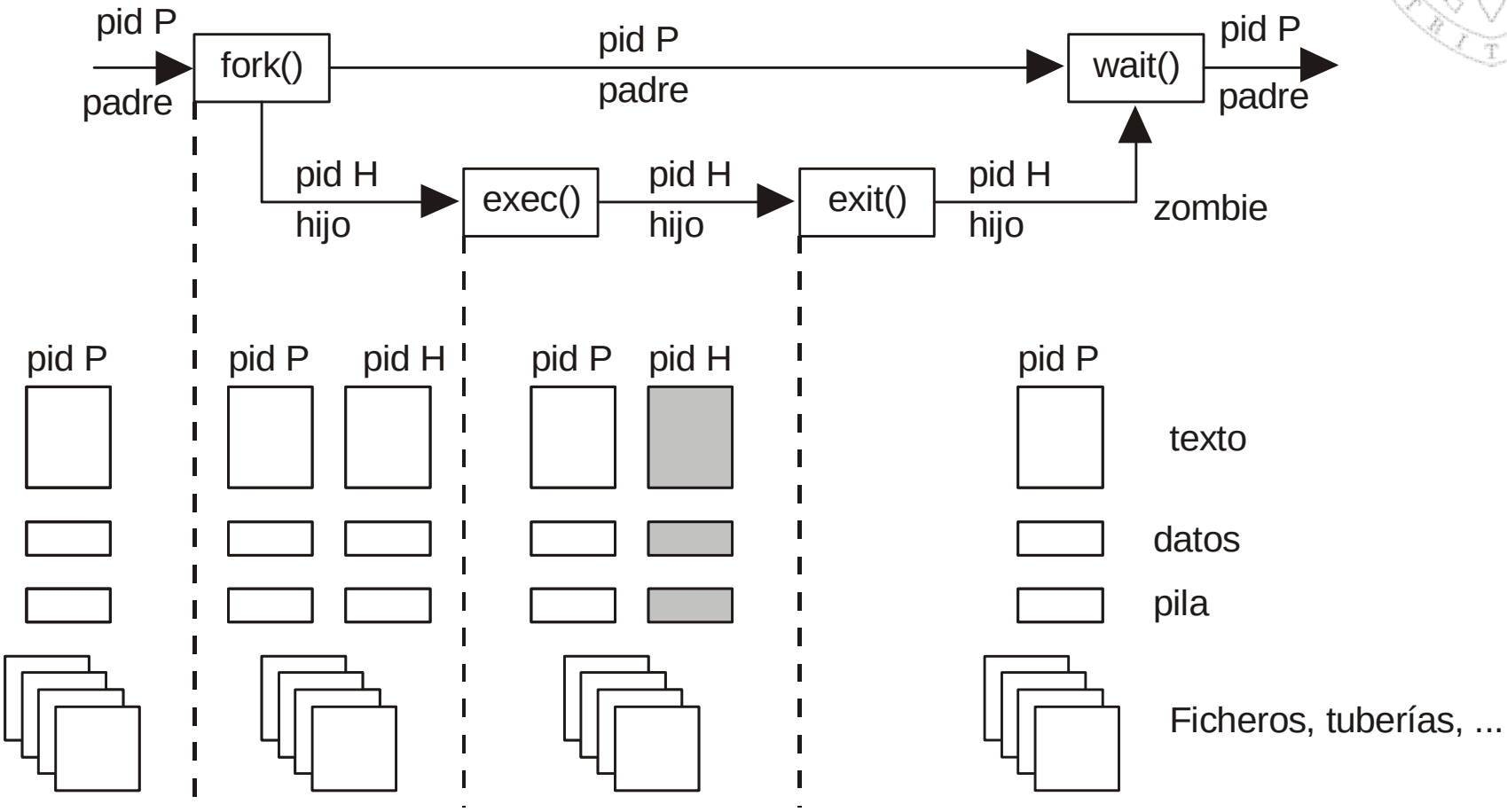
wait. Espera la terminación de un proceso hijo

- Servicios:

```
#include <sys/types.h>
pid_t wait(int *status);
```

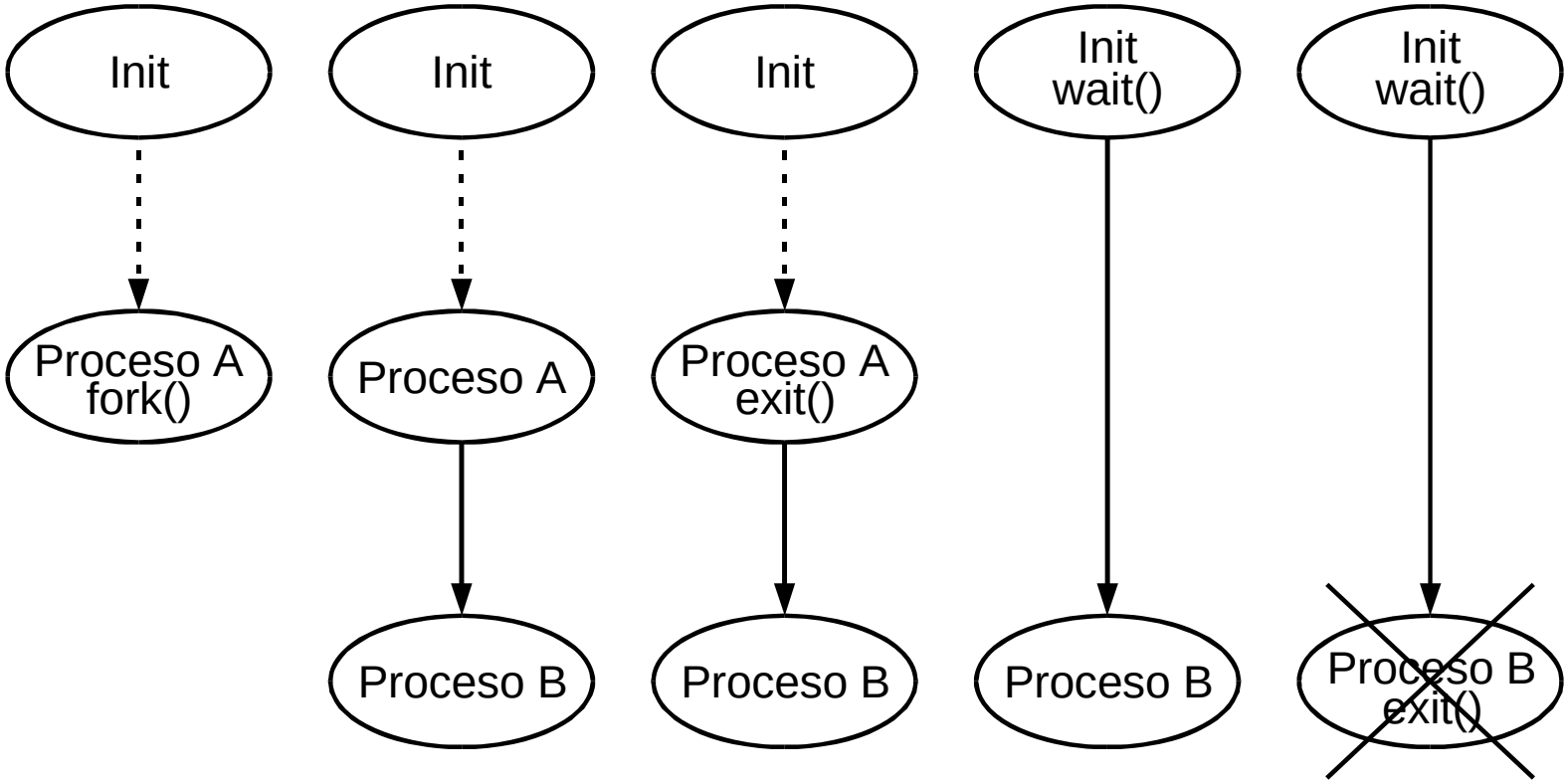
- Argumentos:
  - Devuelve el código de terminación del proceso hijo.
- Descripción:
  - Devuelve el identificador del proceso hijo o -1 en caso de error.
  - Permite a un proceso padre esperar hasta que termine un proceso hijo. Devuelve el identificador del proceso hijo y el estado de terminación del mismo.

Uso normal de los servicios



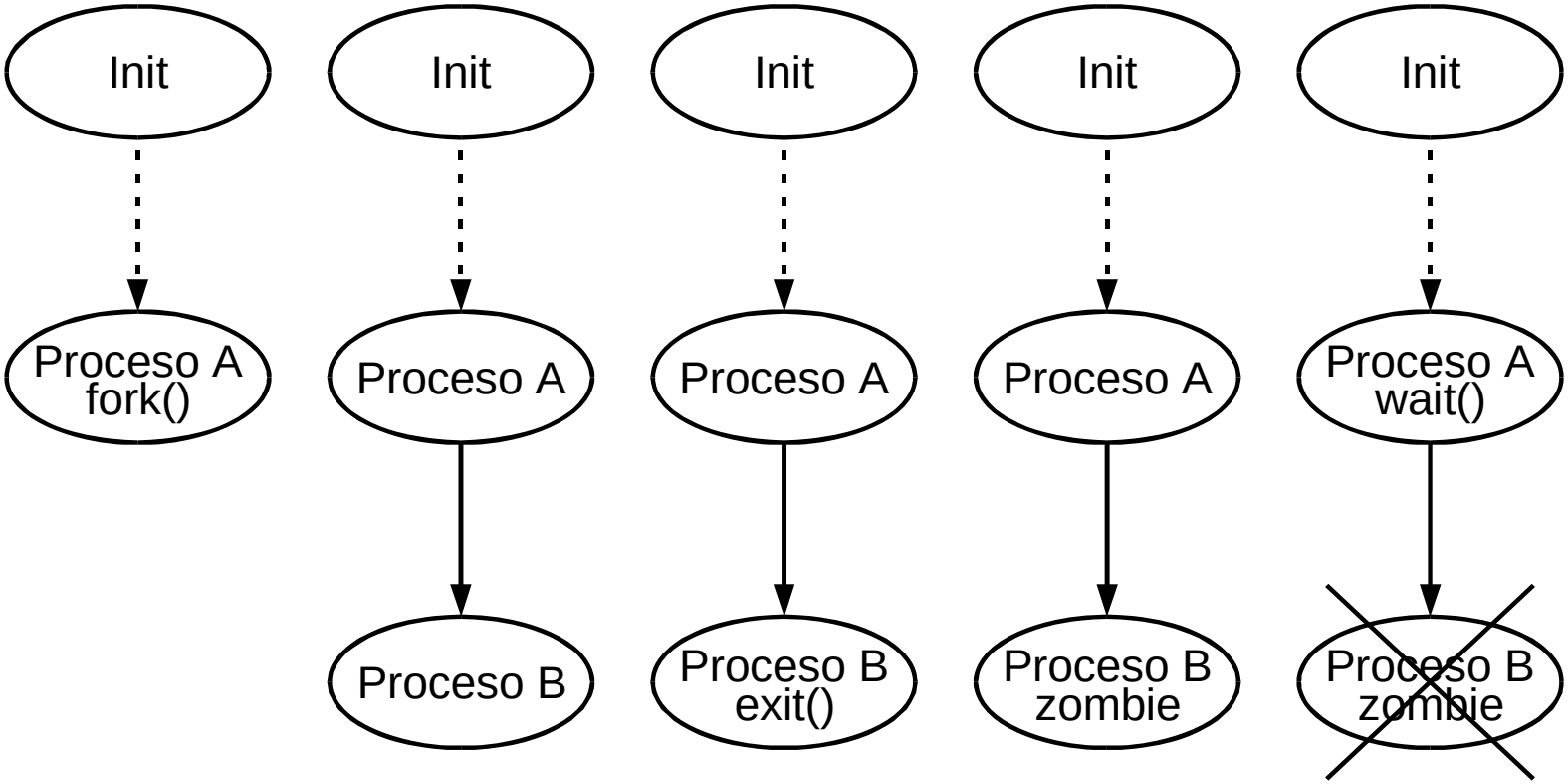
Evolución de procesos I

- El padre muere: *INIT* acepta los hijos

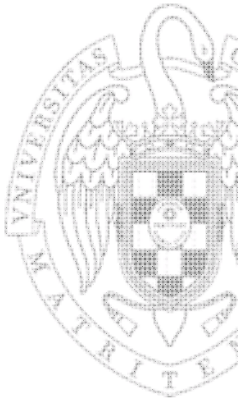


Evolución de procesos II

- *Zombie*: el hijo muere y el padre no hace *wait*



Programa de ejemplo



```
#include <sys/types.h>
#include <stdio.h>
/* programa que ejecuta el mandato ls -l */
main() {
   pid_t pid;
   int status;
   pid = fork();
   if (pid == 0) { /* proceso hijo */
      execlp("ls","ls","-l",NULL);
      exit(-1);
   }
   else        /* proceso padre */
     while (pid != wait(&status));
   exit(0);
}
```

Señales: servicios POSIX

- **int kill(pid_t pid, int sig)**
  - Envía al proceso *pid* la señal *sig*
- **int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);**
  - Permite especificar la acción a realizar *act* como tratamiento de la señal *signum*. Permite almacenar la acción previa en *oldact*
- **int pause(void)**
  - Bloquea al proceso hasta la recepción de una señal.
- Ejemplo:

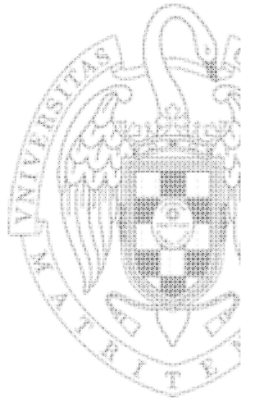
$ kill -SIGINT pid

Envía al proceso con identificador pid la señal de interrupción (Ctrl C).

Si el proceso tiene un manejador para esa señal ejecutará el código del manejador.

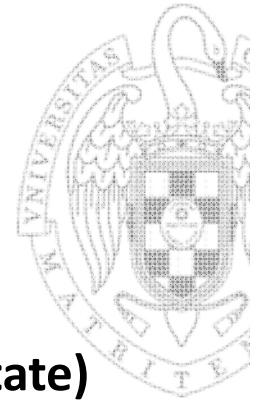
En caso contrario, el proceso muere.

POSIX para la gestión de hilos



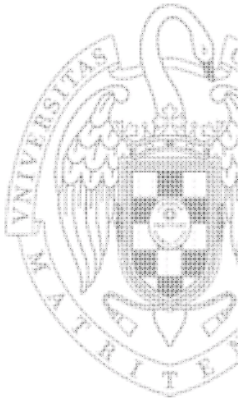
- **int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*func)(void *), void *arg)**
  - Crea un hilo que ejecuta "func" con argumento "arg" y atributos "attr".
  - Los atributos permiten especificar: tamaño de la pila, prioridad, política de planificación, etc.
  - Existen diversas llamadas para modificar los atributos.
- **int pthread_join(pthread_t thid, void **value)**
  - Suspende la ejecución de un hilo hasta que termina el hilo con identificador "thid".
  - Devuelve el estado de terminación del hilo.
- **int pthread_exit(void *value)**
  - Permite a un hilo finalizar su ejecución, indicando el estado de terminación del mismo.
- **pthread_t pthread_self(void)**
  - Devuelve el identificador del thread que ejecuta la llamada.

POSIX para la gestión hilos (II)



- **int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate)**
  - Establece el estado de terminación de un hilo.
  - Si *detachstate* = PTHREAD_CREATE_DETACHED el hilo liberará sus recursos cuando finalice su ejecución.
  - Si *detachstate* = PTHREAD_CREATE_JOINABLE no se liberarán los recursos, es necesario utilizar pthread_join().

Programa de ejemplo



```
#include <stdio.h>
#include <pthread.h>
#define MAX_THREADS 10
void* func(void* arg)  {
   printf("Thread %d \n", pthread_self());
   pthread_exit(0);
}
int main(void)   {
   int j;
   pthread_attr_t attr;
   pthread_t thid[MAX_THREADS];
   pthread_attr_init(&attr);
   for(j = 0; j < MAX_THREADS; j ++)
        pthread_create(&thid[j], &attr,  func, NULL);
   for(j = 0; j < MAX_THREADS; j ++)
        pthread_join(thid[j], NULL);
   return 0;
}
```