lo solicite. El proceso que solicita el servicio se convierte en el proceso padre del nuevo proceso, que es, a su vez, el proceso hijo.

El prototipo de esta función es el siguiente:

```
pid_t fork();
```

Ejecutar un programa

El servicio exec de POSIX tiene por objetivo cambiar el programa que está ejecutando un proceso. En POSIX existe una familia de funciones exec, cuyos prototipos se muestran a continuación:

```
int execl(char *path, char *arg, ...);
int execv(char *path, char *argv[]);
int execle(char *path, char *arg, ...);
int execve(char *path, char *argv[], char *envp[]);
int execlp(char *file, const char *arg, ...);
int execvp(char *file, char *argv[]);
```

Terminar la ejecución de un proceso

Cuando un programa ejecuta dentro de la función `main` la sentencia `return(valor)`, ésta es similar a `exit(valor)`. El prototipo de la función `exit` es:

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

Esperar por la finalización de un proceso hijo

Permite a un proceso padre esperar hasta que termine la ejecución de un proceso hijo (el proceso padre se queda bloqueado hasta que termina un proceso hijo). Existen dos formas de invocar este servicio:

```
pid_t wait(int *status);
pid_t waitpid(pid_t pid, int *status, int options);
```

Ambas llamadas esperan la finalización de un proceso hijo y permiten obtener información sobre el estado de terminación del mismo.

Suspender la ejecución de un proceso

Suspende al proceso durante un número determinado de segundos. Su prototipo es:

```
int sleep(unsigned int seconds)
```

El proceso se suspende durante un número de segundos pasado como parámetro. El proceso despierta cuando ha transcurrido el tiempo establecido o cuando se recibe una señal.

2.5. SERVICIOS POSIX DE GESTIÓN DE PROCESOS LIGEROS

En esta sección se describen los principales servicios POSIX relativos a la gestión de procesos ligeros. Estos servicios se han agrupado de acuerdo a las siguientes categorías:

- Atributos de un proceso ligero.
- Creación e identificación de procesos ligeros.
- Terminación de procesos ligeros.

Atributos de un proceso ligero

Cada proceso ligero en POSIX tiene asociado una serie de atributos que representan sus propiedades. Los valores de los diferentes atributos se almacenan en un objeto atributo de tipo `pthread_attr_t`. Existen una serie de servicios que se aplican sobre el tipo anterior y que permiten modificar los valores asociados a un objeto de tipo atributo. A continuación se describen las principales funciones relacionadas con los atributos de los procesos ligeros.

Crear atributos de un proceso ligero

Este servicio permite iniciar un objeto atributo que se puede utilizar para crear nuevos procesos ligeros. El prototipo de esta función es:

```
int pthread_attr_init(pthread_attr_t *attr);
```

Destruir atributos

Destruye el objeto de tipo atributo pasado como argumento a la misma. Su prototipo es:

```
int pthread_attr_destroy(pthread_attr_t*attr);
```

Asignar el tamaño de la pila

Cada proceso ligero tiene una pila cuyo tamaño se puede establecer mediante esta función cuyo prototipo es el siguiente:

```
int pthread_attr_setstacksize(pthread_attr_t*attr, int stacksize);
```

Obtener el tamaño de la pila

El prototipo del servicio que permite obtener el tamaño de la pila de un proceso es:

```
int pthread_attr_getstacksize(pthread_attr_t*attr, int stacksize);
```

Establecer el estado de terminación

El prototipo de este servicio es:

```
int pthread_attr_setdetachstate(pthread_attr_t*attr,
int detachstate);
```

Si el valor del argumento detachstate es PTHREAD_CREATE_DETACHED, el proceso ligero que se cree con esos atributos se considerará como independiente y liberará sus recursos cuando finalice su ejecución. Si el valor del argumento detachstate es PTHREAD_CREATE_JOINABLE, el proceso ligero se crea como no independiente y no liberará sus recursos cuando finalice su ejecución. En este caso es necesario que otro proceso ligero espere por su finalización. Esta espera se consigue mediante el servicio pthread_join, que se describirá más adelante.

Obtener el estado de terminación

Permite conocer el estado de terminación que se especifica en un objeto de tipo atributo. Su prototipo es:

```
int pthread_attr_getdetachstate(pthread_attr_t*attr,
      int*detachstate);
```

Creación, identificación y terminación de procesos ligeros

Los servicios relacionados con la creación e identificación de procesos ligeros son los siguientes:

Crear un proceso ligero

Este servicio permite crear un nuevo proceso ligero que ejecuta una determinada función. Su prototipo es:

```
int pthread_create(pthread_t*thread, pthread_attr_r*attr,
   void*(*start_routine)(void*), void*arg);
```

El primer argumento de la función apunta al identificador del proceso ligero que se crea, este identificador viene determinado por el tipo pthread_t. El segundo argumento especifica los atributos de ejecución asociados al nuevo proceso ligero. Si el valor de este segundo argumento es NULL, se utilizarán los atributos por defecto, que incluyen la creación del proceso como no independiente. El tercer argumento indica el nombre de la función a ejecutar cuando el proceso ligero comienza su ejecución. Esta función requiere un sólo parámetro que se especifica con el cuarto argumento, arg.

Obtener el identificador de un proceso ligero

Un proceso ligero puede averiguar su identificador invocando este servicio, cuyo prototipo es el siguiente:

```
pthread_t pthread_self(void)
```

Esperar la terminación de un proceso ligero

Este servicio es similar al wait, pero a diferencia de éste, es necesario especificar el proceso ligero por el que se quiere esperar, que no tiene porqué ser un proceso ligero hijo. El prototipo de la función es:

```
int pthread_join(pthread thid, void**value);
```


La función suspende la ejecución del proceso ligero llamante hasta que el proceso ligero con identificador thid finalice su ejecución. La función devuelve en el segundo argumento el valor que pasa el proceso ligero, que finaliza su ejecución en el servicio pthread_exit, que se verá a continuación. Únicamente se puede solicitar el servicio pthread_join sobre procesos ligeros creados como no independientes.

Finalizar la ejecución de un proceso ligero

Es análogo al servicio exit sobre procesos. Su prototipo es:

```
int pthread_exit(void*value)
```

Incluye un puntero a una estructura que es devuelta al proceso ligero que ha ejecutado la correspondiente llamada a pthread_join, lo que es mucho más genérico que el parámetro que permite el serivicio wait.

2.6. SERVICIOS POSIX DE SEÑALES

En esta sección se describen los principales servicios POSIX relativos a la gestión de señales. Los más importantes son el envío y la captura de señales.

Algunas señales como SIGSEGV o SIGBUS las genera el sistema operativo cuando ocurren ciertos errores. Otras señales se envían de unos procesos a otros utilizando el siguiente servicio:

Enviar una señal

Permite enviar una señal a un proceso. El prototipo de este servicio en lenguaje C es el siguiente:

```
int kill(pid_t pid, int sig);
```

Envía la señal sig al proceso o grupo de procesos especificado por pid.

Armado de una señal

El servicio que permite armar una señal es:

```
int sigaction(int sig, struct sigaction*act,
 struct sigaction*oact);
```

Esta llamada tiene tres parámetros: el número de señal para la que se quiere establecer el manejador, un puntero a una estructura de tipo struct sigaction para establecer el nuevo manejador y un puntero a una estructura también del mismo tipo que almacena información sobre el manejador establecido anteriormente.

Espera por una señal

Cuando se quiere esperar la recepción de una señal se utiliza el siguiente servicio pause. Este servicio bloquea al proceso que la invoca hasta que llegue una señal. Su prototipo es:

```
int pause(void);
```