Finalización de Procesos

1- **Formas normales**
   a) Retorno desde la función main
   b) Llamando a la función exit
   c) Llamando a la función _exit

2- **Formas anormales**
   d) Llamando a la función abort
   e) Terminación a causa de una señal (signal)

Nota: Si la llamada a la función exit( ) estuviese escrita en C, la llamada a la función main( ) podría realizarse de la forma :

```
exit(main(argc, argv));
```

Funciones para la finalización normal de un proceso:

_exit: retorna inmediatamente del kernel

```
_exit - terminate the current process

#include <unistd.h>
void _exit(int status);
```

exit: realiza algunas labores de limpieza y retorna del kernel

```
exit - cause normal program termination

#include <stdlib.h>
void exit(int status);
```

exit realiza un “clean shutdown” de la “standard I/O library”:se llama a la función fclose para todos los “open streams” (todos los datos de salida “buffered” son escritos en los ficheros, “flushed”)

Nota: Las diferentes cabeceras se deben a que exit está especificada por ANSI C, mientras que _exit está especificada por POSIX.1

exit() y _exit() tienen un único argumento entero que se denomina “exit status”.

El “exit status” de un proceso está **indefinido** en los siguientes casos:
   a) main realiza un return sin un valor de return
   b) Si _exit o exit son llamadas sin un “exit status”
   c) Si main realiza un return implícito

Nota: Casi todos los shells Unix permiten recoger el “exit status” de un proceso.

Función atexit()

Con ANSI C un proceso puede registrar hasta 32 funciones que pueden ser llamadas automáticamente por exit. Estas funciones se denominan “exit handlers” y se registran llamando a la función atexit( )

atexit - register a function to be called at normal program termination.

```
#include <stdlib.h>
int atexit(void (*function)(void));
```

Devuelve: 0 si OK, no cero si error.

La función exit( ) llama a estas funciones en orden inverso a su registro.

Los “exit handlers” aparecen con ANSI C y se suministran para SVR4 y 4.3+BSD

Con ANSI C y POSIX.1, exit( ) primero llama a los “exit handlers” y luego “fcloses” los “open streams”.

Ejemplo:

```
//  Finalizanción de procesos: atexit( )

#include "error.h"

static void exit1();
static void exit2();

int main( )
{
  if (atexit(exit2) != 0) error("No se registro exit2");
  if (atexit(exit1) != 0) error("No se registro exit1");

  printf("\n main se acabo \n");

  exit(0);
}

static void exit1()
{
  printf("\n exit handler 1 \n");
}

static void exit2()
{
  printf("\n exit handler 2 \n");
}
```