Práctica 2.1: Introducción a la programación de sistemas Unix

Objetivos

En esta práctica estudiaremos el uso básico del API de un sistema Unix y su entorno de desarrollo. En particular, se usarán funciones para gestionar errores y obtener información.

Contenidos

Preparación del entorno para la práctica

Esta práctica únicamente requiere el entorno de desarrollo (compilador, editores y depurador), que está disponible en las máquinas virtuales de la asignatura y en la máquina física del laboratorio.

Se puede usar cualquier editor gráfico o de terminal. Además, se puede usar tanto el lenguaje C (compilador `gcc`) como C++ (compilador `g++`). Si fuera necesario compilar varios archivos, se recomienda el uso de `make`. Finalmente, el depurador recomendado en las prácticas es `gdb`. **No se recomienda** el uso de IDEs como Eclipse.

Gestión de errores

Usar `perror(3)` y `strerror(3)` para gestionar los errores en los siguientes casos. En cada ejercicio, añadir las librerías necesarias (con `#include`).

Ejercicio 1. Añadir el código necesario para gestionar correctamente los errores generados por `setuid(2)`. Consultar en el manual el propósito de la llamada y su prototipo.

```
int main() {
   setuid(0);
   return 1;
}
```

Ejercicio 2. Imprimir el código numérico de error generado por la llamada del código anterior y el mensaje asociado.

Ejercicio 3. Escribir un programa que imprima todos los mensajes de error disponibles en el sistema. Considerar inicialmente que el límite de errores posibles es 255.

Información del sistema

Ejercicio 4. Consultar la página de manual de `uname(1)` y obtener información del sistema.

Ejercicio 5. Escribir un programa que muestre, con `uname(2)`, cada aspecto del sistema y su valor. Comprobar la correcta ejecución de la llamada.

Ejercicio 6. Escribir un programa que obtenga, con `sysconf(3)`, información de configuración del sistema e imprima, por ejemplo, la longitud máxima de los argumentos, el número máximo de hijos y el número máximo de ficheros abiertos.

Ejercicio 7. Escribir un programa que obtenga, con `pathconf(3)`, información de configuración del sistema de ficheros e imprima, por ejemplo, el número máximo de enlaces, el tamaño máximo de una ruta y el de un nombre de fichero.

Información del usuario

Ejercicio 8. Consultar la página de manual de `id(1)` y comprobar su funcionamiento.

Ejercicio 9. Escribir un programa que muestre, igual que `id`, el UID real y efectivo del usuario. ¿Cuándo podríamos asegurar que el fichero del programa tiene activado el bit *setuid*?

Ejercicio 10. Modificar el programa anterior para que muestre además el nombre de usuario, el directorio *home* y la descripción del usuario.

Información horaria del sistema

Ejercicio 11. Consultar la página de manual de `date(1)` y familiarizarse con los distintos formatos disponibles para mostrar la hora.

Ejercicio 12. Escribir un programa que muestre la hora, en segundos desde el Epoch, usando `time(2)`.

Ejercicio 13. Escribir un programa que mida, en microsegundos, lo que tarda un bucle que incrementa una variable un millón de veces usando `gettimeofday(2)`.

Ejercicio 14. Escribir un programa que muestre el año usando `localtime(3)`.

Ejercicio 15. Modificar el programa anterior para que imprima la hora de forma legible, como `"lunes, 29 de octubre de 2018, 10:34"`, usando `strftime(3)`.

Nota: Para establecer la configuración regional (*locale*, como idioma o formato de hora) en el programa según la configuración actual, usar `setlocale(3)`, por ejemplo, `setlocale(LC_ALL, "")`. Para cambiar la configuración regional, ejecutar, por ejemplo, `export LC_ALL="es_ES"`, o bien, `export LC_TIME="es_ES"`.