Práctica 6: Introducción a la programación de Sistemas UNIX

Objetivos

En esta práctica el alumno estudiará el uso básico y convenciones del API de un sistema UNIX y su entorno de desarrollo. En particular se usará las funciones disponibles para la gestión de errores y

Contenidos

Preparación del entorno para la práctica

La realización de esta práctica únicamente requiere del entorno de desarrollo (compilador, editores y utilidades de depuración). Estas herramientas están disponibles en las máquinas virtuales de la asignatura y en la máquina física de los puestos del laboratorio.

En la realización de las prácticas 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 alguna herramienta para la compilación de proyectos como make. Finalmente, el depurador recomendado en las prácticas es gdb. **No está permitido** el uso de IDE como Eclipse.

Gestión de Errores

Usar las funciones disponibles en el API del sistema para gestionar los errores en los siguientes casos. En cada ejercicio añadir las librerías necesarias (`#include`)

Ejercicio 1. Añadir el código necesario para gestionar correctamente los errores generados por la llamada setuid(). Usando la página de manual comprobar el propósito de la llamada setuid y su prototipo.

```
int main()
{
    /* Comprobar la ocurrencia de error y notificarlo con la llamada adecuada */
    setuid(0);
    return 1;
}
```

Ejercicio 2. En el código anterior imprimir el código de error generado por la llamada, tanto en su versión numérica como la cadena asociada.

Ejercicio 3. Escribir un programa que recorra en un bucle todos los mensajes de error disponibles en el sistema y los imprima.

Información del Sistema

Ejercicio 1. El comando del sistema uname(1) muestra información sobre diversos aspectos del sistema. Consultar la página de manual, y obtener la información del sistema.

Ejercicio 2. Escribir un programa que muestre, claramente identificado, cada aspecto del sistema y su valor, comprobar la correcta ejecución de la llamada en cada caso. Consultar uname(2) para más información sobre la llamada al sistema.

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

Ejercicio 4. Repetir el ejercicio anterior pero en este caso para la configuración del sistema de ficheros, pathconf(3). Por ejemplo que muestre el número máximo de enlaces, el tamaño máximo de una ruta y el de un nombre de fichero.

Información del Usuario

Ejercicio 1. El comando del sistema id(1) muestra la información de usuario real y efectiva. Consultar la página de manual y comprobar el funcionamiento del comando.

Ejercicio 2. Escribir un programa que muestre igual que el comando id los uid efectivos y real del usuario. ¿En qué circunstancias podríamos asegurar que el fichero tiene activado el bit setuid?.

Ejercicio 3. Modificar el programa anterior para que se muestre además el nombre de *login,* directorio home e información de usuario del usuario real.

Información Horaria del Sistema

Ejercicio 1. El comando principal para mostrar la hora del sistema es date. Consultar la página de manual y familiarizarse con los distintos formatos disponibles para mostrar la información horaria del sistema.

Ejercicio 2. La función principal para obtener la hora del sistema es time(). Escribir un programa que obtenga la hora usando está función y la muestre en el terminal.

Ejercicio 3. Modificar el ejercicio dos para que se muestre además la hora en formato *legible,* usando la función ctime. ¿Dónde se reserva espacio para el valor de la cadena que devuelve la función? ¿Es necesario liberar el puntero?

Ejercicio 4. Cuando es necesario obtener la información horaria con precisión de microsegundos se puede usar gettimeofday(). Escribir un programa que mida cuánto tarda un bucle de 10000 repeticiones en incrementar una variable en una unidad en cada iteración.

Ejercicio 5. Escribir un programa que muestre el año, p.ej. “Estamos en el año 1982”, usando la función localtime().

Ejercicio 6. Modificar el programa anterior para que usando la función strftime(), imprima además la hora en la forma: “Hoy es Lunes, 10:34”.