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](#_14a3ftqman5y)

[Gestión de errores](#_dngqig2y9aib)

[Información del sistema](#_d769h3pfq9ne)

[Información del usuario](#_2zko56iifsm2)

[Información horaria del sistema](#_1crtpfy04scr)

# 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.

# Gestión de errores

***Ejercicio 1.***Añadir el código necesario para gestionar correctamente los errores generados por setuid(2) con perror(3). Consultar en el manual el propósito de la llamada y su prototipo.

| **int** main() {  setuid(0);  **return** 1;  }  **int main() {**  **if (setuid(0) == -1) {**  **perror("Error al establecer el ID de usuario efectivo");**  **exit(1);**  **}**    **return 0; // El cambio de ID de usuario fue exitoso**  **}** |
| --- |

***Ejercicio 2.*** Imprimir el código numérico de error generado por la llamada del código anterior y el mensaje asociado obtenido con strerror(3).

**#include <errno.h>**

**printf("Código de error %d: %s \n", errno, strerror(errno));**

***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.

for(int i=0;i<255;i++){

printf("Código de error %d: %s \n", i, strerror(i));

}

# Información del sistema

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

* Para obtener el nombre del sistema operativo, puedes usar:

uname -s

* Para obtener la versión del sistema, puedes usar:

uname -r

* Para obtener el nombre del nodo de red (nombre de la máquina), puedes usar

uname -n

* Para obtener el tipo de máquina (hardware), puedes usar:

uname -m

* Para obtener la información completa, puedes usar simplemente:

uname -a

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

#include <stdio.h>

#include <sys/utsname.h>

int main() {

struct utsname system\_info;

// Llamar a uname para obtener información del sistema

if (uname(&system\_info) == -1) {

perror("uname");

return 1;

}

// Mostrar la información del sistema

printf("Nombre del sistema operativo: %s\n", system\_info.sysname);

printf("Nombre del nodo de red: %s\n", system\_info.nodename);

printf("Versión del sistema: %s\n", system\_info.release);

printf("Número de versión: %s\n", system\_info.version);

printf("Arquitectura del hardware: %s\n", system\_info.machine);

return 0;

}

***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.

#include <stdio.h>

#include <unistd.h>

int main() {

printf("%ld\n", sysconf(\_SC\_ARG\_MAX));

printf("%ld\n", sysconf(\_SC\_CHILD\_MAX));

printf("%ld\n", sysconf(\_SC\_OPEN\_MAX));

return 0;

}

***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.

#include <stdio.h>

#include <unistd.h>

int main() {

printf("%ld\n", pathconf("/",\_PC\_LINK\_MAX));

printf("%ld\n", pathconf("/",\_PC\_PATH\_MAX));

printf("%ld\n", pathconf("/",\_PC\_NAME\_MAX));

return 0;

}

# Información del usuario

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

**Imprimir información de usuario y grupo para cada USUARIO especificado, o**

**(cuando el USUARIO se omite) para el proceso actual.**

**-a ignorar, por compatibilidad con otras versiones**

**-Z , --contexto**

**imprimir solo el contexto de seguridad del proceso**

**-g , --grupo**

**imprimir sólo el ID de grupo efectivo**

**-G , --grupos**

**imprimir todos los ID de grupo**

**-n , --name**

**imprime un nombre en lugar de un número, para -ugG**

**-r , --real**

**imprime el ID real en lugar del ID efectivo, con -ugG**

**-u , --usuario**

**imprimir sólo el ID de usuario efectivo**

**-z , --cero**

**delimitar entradas con caracteres NUL, no espacios en blanco;**

***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*?

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

int main() {

printf("%d\n",getuid());

printf("%d\n",geteuid());

return 0;

}

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

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <pwd.h>

int main() {

struct passwd \*user\_info = getpwuid(getuid());

if (user\_info != NULL) {

printf("Nombre de usuario: %s\n", user\_info->pw\_name);

printf("Directorio home: %s\n", user\_info->pw\_dir);

printf("Descripción del usuario: %s\n", user\_info->pw\_gecos);

} else {

printf("No se pudo obtener la información del usuario.\n");

}

printf("UID Real: %d\n", getuid());

printf("UID Efectivo: %d\n", geteuid());

return 0;

}

# 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.

**Algunos ejemplos de formatos comunes que puedes utilizar con el comando date son:**

* **%H: Hora en formato de 24 horas (00-23).**
* **%I: Hora en formato de 12 horas (01-12).**
* **%M: Minutos (00-59).**
* **%S: Segundos (00-59).**
* **%p: AM o PM para las horas en formato de 12 horas.**
* **%Y: Año con cuatro dígitos.**
* **%y: Año con dos dígitos.**
* **%m: Mes (01-12).**
* **%d: Día del mes (01-31).**
* **%A: Nombre completo del día de la semana.**
* **%a: Nombre corto del día de la semana.**
* **%B: Nombre completo del mes.**
* **%b o %h: Nombre corto del mes.**

**Puedes combinar estos códigos de formato según tus necesidades. Por ejemplo, para mostrar la hora en formato de 24 horas junto con los minutos y los segundos, puedes usar:**

**perl**

**Copy code**

**date "+%H:%M:%S"**

**Esto mostrará la hora actual en formato "HH:MM:SS".**

**Recuerda que la disponibilidad de ciertos formatos puede variar según la implementación de date en tu sistema operativo, por lo que es importante consultar el manual local en tu sistema para obtener información específica sobre los formatos compatibles.**

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

#include <stdio.h>

#include <time.h>

int main() {

time\_t current\_time;

// Obtener la hora actual en segundos desde el Epoch

current\_time = time(0);

if (current\_time == -1) {

perror("time");

return 1;

}

// Mostrar la hora en segundos

printf("Hora en segundos desde el Epoch: %ld\n", current\_time);

return 0;

}

***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).

#include <stdio.h>

#include <sys/time.h>

int main() {

struct timeval start\_time, end\_time;

unsigned long long iterations = 1000000; // Un millón de iteraciones

unsigned long long i;

unsigned long long variable = 0; // Variable a incrementar

// Obtener el tiempo de inicio

gettimeofday(&start\_time, NULL);

// Realizar el bucle que incrementa una variable un millón de veces

for (i = 0; i < iterations; i++) {

variable++;

}

// Obtener el tiempo de finalización

gettimeofday(&end\_time, NULL);

// Calcular el tiempo transcurrido en microsegundos

long long elapsed\_time\_us = (end\_time.tv\_usec - start\_time.tv\_usec);

// Mostrar el tiempo transcurrido en microsegundos

printf("Tiempo transcurrido: %lld microsegundos\n", elapsed\_time\_us);

return 0;

}

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

**#include <stdio.h>**

**#include <time.h>**

**int main() {**

**time\_t tiempo;**

**struct tm \*tiempo\_local;**

**// Obtenemos el tiempo actual**

**tiempo = time(0);**

**// Convertimos el tiempo a la estructura de tiempo local**

**tiempo\_local = localtime(&tiempo);**

**// Accedemos al campo "tm\_year" para obtener el año**

**int anio = tiempo\_local->tm\_year + 1900;**

**// Mostramos el año**

**printf("El año actual es: %d\n", anio);**

**return 0;**

**}**

***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).

**#include <stdio.h>**

**#include <time.h>**

**int main() {**

**time\_t tiempo;**

**struct tm \*tiempo\_local;**

**char formato[80];**

**// Obtenemos el tiempo actual**

**tiempo = time(0);**

**// Convertimos el tiempo a la estructura de tiempo local**

**tiempo\_local = localtime(&tiempo);**

**// Definimos el formato deseado para la fecha y hora**

**strftime(formato, sizeof(formato), "%A, %d de %B de %Y, %H:%M", tiempo\_local);**

**//%A lunes/martes/miercoles.**

**//%d numero**

**//%B mes**

**//%Y año**

**//%H hora**

**//%m minuto.**

**// Mostramos la fecha y hora en el formato legible**

**printf("La hora actual es: %s\n", formato);**

**return 0;**

**}**

***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".