Práctica 5: 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](#h.14a3ftqman5y)

[Gestión de Errores](#h.dngqig2y9aib)

[Información del Sistema](#h.d769h3pfq9ne)

[Información del Usuario](#h.rg73kma1zd4y)

[Información Horaria del Sistema](#h.m7n4o8t1iyye)

# 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). Se recomienda emplear las llamadas a perror() y strerror().

***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;  } |

**#include <sys/types.h>  
#include <unistd.h>  
#include <stdio.h>  
#include <stdlib.h>**

**int main() {  
 if(setuid(0)) {  
 perror("setuid(0)");  
 exit -1;  
 }**

**return 0;**

**}**

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

**#include <sys/types.h>  
#include <unistd.h>  
#include <stdio.h>  
#include <errno.h>  
#include <string.h>  
#include <stdlib.h>**

**int main() {  
 if(setuid(0)) {  
 printf("setuid(0): %s Código de error %i\n",strerror(errno),errno);  
 exit -1;  
 }**

**return 0;**

**}**

***Ejercicio 3*.** Escribir un programa que recorra en un bucle todos los mensajes de error disponibles en el sistema y los imprima. Considerar inicialmente que el límite de errores posibles es 255. Según la salida del programa, determinar el número de errores definidos en el sistema.

**#include <sys/types.h>  
#include <unistd.h>  
#include <stdio.h>  
#include <errno.h>  
#include <string.h>  
#include <stdlib.h>**

**int main() {  
 int i;  
 for(i=1; i <= 133; i++) {  
 printf("Info: %s Código de error %i\n",strerror(i),i);  
 }  
 return 0;**

**}**

# 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.

**#include <sys/utsname.h>  
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>**

**int main() {  
 struct utsname user;**

**if(uname(&user) == -1) {  
 perror("uname(&user)");  
 exit -1;  
 }**

**printf("Name's machine: %s \nOperating system: %s \nKernel Release: %s Kernel Version: %s \nHardware identiffier: %s \n",user.nodename, user.sysname, user.release, user.version, user.machine);**

**}**

***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.

**#include <sys/types.h>  
#include <stdio.h>  
#include <unistd.h>  
#include <stdlib.h>**

**int main() {  
 if(sysconf(\_SC\_ARG\_MAX) == -1) {  
 perror("sysconf(\_SC\_ARG\_MAX");  
 exit -1;  
 }**

**printf("Numero máximo de argumentos: %ld\n", sysconf(\_SC\_ARG\_MAX));**

**if(sysconf(\_SC\_CHILD\_MAX) == -1) {  
 perror("sysconf(\_SC\_CHILD\_MAX");  
 exit -1;  
 }**

**printf("Número máximo de hijos: %ld\n", sysconf(\_SC\_CHILD\_MAX));**

**if(sysconf(\_SC\_OPEN\_MAX) == -1) {  
 perror("sysconf(\_SC\_OPEN\_MAX)");  
 exit -1;**

**}**

**printf("Número máximo de ficheros: %ld\n", sysconf(\_SC\_OPEN\_MAX));**

**return 0;**

**}**

***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.

**#include <sys/types.h>  
#include <stdio.h>  
#include <unistd.h>  
#include <stdlib.h>**

**int main() {  
 if(pathconf(".",\_PC\_LINK\_MAX) == -1) {  
 perror("pathconf('.',\_PC\_LINK\_MAX");  
 exit -1;  
 }else {  
 printf("Numero máximo de enlaces: %ld\n", pathconf(".",\_PC\_LINK\_MAX));   
 }**

**if(pathconf(".",\_PC\_PATH\_MAX) == -1) {  
 perror("patchconf('.',\_PC\_PATH\_MAX");  
 exit -1;**

**} else {  
 printf("Tamaño máximo de ruta: %ld\n", pathconf(".",\_PC\_PATH\_MAX));   
 }**

**if(pathconf(".",\_PC\_NAME\_MAX) == -1) {  
 perror("pathconf('.',\_PC\_NAME\_MAX)");  
 exit -1;   
 } else {  
 printf("Tamaño máximo de nombre de fichero: %ld\n", pathconf(".",\_PC\_NAME\_MAX)); }**

**return 0;**

**}**

# 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?.

**#include <stdio.h>  
#include <unistd.h>  
#include <sys/types.h>**

**int main() {  
 printf("ID real: %i ID efectivo: %i\n",geteuid(),getuid());**

**return 0;  
}**

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

**/\*\*El bit setuid es asignable a ficheros ejecutables y permite que cuando**

**un usuario ejecute dicho fichero, el proceso adquiera los permisos del**

**propietario del fichero ejecutado. EL ejemplo más claro del fichero ejecutable**

**con el bit setuid el su\*\*\*/**

**#include <stdio.h>  
#include <unistd.h>  
#include <sys/types.h>  
#include <pwd.h>**

**int main() {  
 uid\_t uid = getuid();  
 uid\_t euid = geteuid();**

**printf("ID real: %i ID efectivo: %i\n",uid,euid);  
 struct passwd \*info\_user = getpwuid(uid);  
 struct passwd \*info\_userreal = getpwuid(euid);  
 printf("Nombre del login: %s \nDirectorio Home: %s \n\n", info\_user->pw\_name, info\_user->pw\_dir);**

**printf("Información del usuario real:\nID real: %i\nNombre del login: %s \nDirectorio Home: %s \n", info\_userreal->pw\_uid, info\_userreal->pw\_name, info\_userreal->pw\_dir);**

**return 0;  
}**

# 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 esta función y la muestre en el terminal.

**#include <stdio.h>  
#include <time.h>**

**int main() {  
 time\_t t;  
 struct tm \*loctime;  
 char buffer[256];  
 t = time(NULL);  
 loctime = localtime(&t);  
 strftime(buffer,256,"%r", loctime);  
 printf("%s\n", buffer);**

**return 0;  
}**

***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?

**#include <stdio.h>  
#include <time.h>**

**int main() {  
 time\_t t;  
 struct tm \*loctime;  
 char buffer[256];  
 t = time(NULL);  
 loctime = localtime(&t);  
 strftime(buffer,256,"%r", loctime);  
 printf("Conversión de la información a cadena usando strftime: %s\n", buffer);  
 printf("Conversión de la información a cadena usando ctime: %s\n",ctime(&t));**

**return 0;  
}**

***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.

**#include <time.h>  
#include <sys/time.h>  
#include <stdio.h>**

**int main() {  
 int iterations = 10000;  
 struct timeval start, end;  
 gettimeofday(&start, NULL);  
 int i;**

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

**gettimeofday(&end, NULL);**

**printf("%ld\n", ((end.tv\_sec\*iterations + end.tv\_usec) - (start.tv\_sec\*iterations + start.tv\_usec)));**

**return 0;**

**}**

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

**#include <stdio.h>  
#include <time.h>**

**int main() {  
 time\_t t;  
 struct tm \*loctime;  
 char buffer[256];  
 t = time(NULL);  
 loctime = localtime(&t);  
 strftime(buffer,256,"%Y", loctime);  
 printf("Estamos en %s\n",buffer);**

**return 0;  
}**

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

**#include <time.h>  
#include <errno.h>  
#include <stdio.h>  
#include <stdlib.h>**

**int main() {  
 time\_t t;  
 time(&t);**

**if (t == -1) {  
 char error\_msg[50];  
 sprintf(error\_msg, "An error (%i) ocurred", errno);  
 perror(error\_msg);  
 return -1;  
 }**

**struct tm \*timedata = localtime(&t);  
 if (timedata == NULL) {  
 char error\_msg[50];  
 sprintf(error\_msg, "An error (%i) ocurred", errno);  
 perror(error\_msg);  
 return -1;  
 }**

**char buf[100];  
 strftime(buf, 100, "%A, %H:%M", timedata);  
 printf("Hoy es %s\n", buf);**

**return 0;**

**}**