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 está permitido** el uso de IDEs como Eclipse.

Gestión de errores

Usar las funciones disponibles en el API del sistema (`perror(3)` y `strerror(3)`) 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 a `setuid(2)`. Consultar en el manual el propósito de la llamada y su prototipo.

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

int main(){
    if (setuid(0) == -1){
        //perror("Error!!")
        printf("Error!! %d - %s\n", errno, strerror(errno));
      }
    return 1;
}
```

```
g++ ./NombreArchivo.cpp -o nombreEjecutable
./nombreEjecutable para ejecutar el codigo creado en el cpp
```

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

```
usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej1
ERROR!! 1 - Operation not permitted

*errno es la version numerica del error y la llamada a strerror saca la cadena asociada
```

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.

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

int main(){
   for(int i = 0; i <= 255; ++i){
      printf("Error %d: %s\n", i, strerror(i));
           }
   return 1;
}

Error 0: Success
Error 1: Operation not permitted
Error 2: No such file or directory
Error 3: No such process
Error 4: Interrupted system call
Error 5: Input/output error
Error 6: No such device or address
Error 7: Argument list too long
Error 8: Exec format error
Error 9: Bad file descriptor
Error 10: No child processes
Error 11: Resource temporarily unavailable
Error 12: Cannot allocate memory
Error 13: Permission denied
Error 14: Bad address
Error 15: Block device required
Error 16: Device or resource busy
Error 17: File exists
Error 18: Invalid cross-device link
Error 19: No such device
Error 20: Not a directory
Error 21: Is a directory
Error 22: Invalid argument
Error 23: Too many open files in system
Error 24: Too many open files
Error 25: Inappropriate ioctl for device
Error 26: Text file busy
Error 27: File too large
Error 28: No space left on device
Error 29: Illegal seek
```

Error 30: Read-only file system
Error 31: Too many links
Error 32: Broken pipe
Error 33: Numerical argument out of domain
Error 34: Numerical result out of range
Error 35: Resource deadlock avoided
Error 36: File name too long
Error 37: No locks available
Error 38: Function not implemented
Error 39: Directory not empty
Error 40: Too many levels of symbolic links
Error 41: Unknown error 41
Error 42: No message of desired type
Error 43: Identifier removed
Error 44: Channel number out of range
Error 45: Level 2 not synchronized
Error 46: Level 3 halted
Error 47: Level 3 reset
Error 48: Link number out of range
Error 49: Protocol driver not attached
Error 50: No CSI structure available
Error 51: Level 2 halted
Error 52: Invalid exchange
Error 53: Invalid request descriptor
Error 54: Exchange full
Error 55: No anode
Error 56: Invalid request code
Error 57: Invalid slot
Error 58: Unknown error 58
Error 59: Bad font file format
Error 60: Device not a stream
Error 61: No data available
Error 62: Timer expired
Error 63: Out of streams resources
Error 64: Machine is not on the network
Error 65: Package not installed
Error 66: Object is remote
Error 67: Link has been severed
Error 68: Advertise error
Error 69: Srmount error
Error 70: Communication error on send
Error 71: Protocol error
Error 72: Multihop attempted
Error 73: RFS specific error
Error 74: Bad message
Error 75: Value too large for defined data type
Error 76: Name not unique on network
Error 77: File descriptor in bad state
Error 78: Remote address changed
Error 79: Can not access a needed shared library
Error 80: Accessing a corrupted shared library
Error 81: .lib section in a.out corrupted
Error 82: Attempting to link in too many shared libraries
Error 83: Cannot exec a shared library directly
Error 84: Invalid or incomplete multibyte or wide character
Error 85: Interrupted system call should be restarted
Error 86: Streams pipe error

Error 87: Too many users
Error 88: Socket operation on non-socket
Error 89: Destination address required
Error 90: Message too long
Error 91: Protocol wrong type for socket
Error 92: Protocol not available
Error 93: Protocol not supported
Error 94: Socket type not supported
Error 95: Operation not supported
Error 96: Protocol family not supported
Error 97: Address family not supported by protocol
Error 98: Address already in use
Error 99: Cannot assign requested address
Error 100: Network is down
Error 101: Network is unreachable
Error 102: Network dropped connection on reset
Error 103: Software caused connection abort
Error 104: Connection reset by peer
Error 105: No buffer space available
Error 106: Transport endpoint is already connected
Error 107: Transport endpoint is not connected
Error 108: Cannot send after transport endpoint shutdown
Error 109: Too many references: cannot splice
Error 110: Connection timed out
Error 111: Connection refused
Error 112: Host is down
Error 113: No route to host
Error 114: Operation already in progress
Error 115: Operation now in progress
Error 116: Stale file handle
Error 117: Structure needs cleaning
Error 118: Not a XENIX named type file
Error 119: No XENIX semaphores available
Error 120: Is a named type file
Error 121: Remote I/O error
Error 122: Disk quota exceeded
Error 123: No medium found
Error 124: Wrong medium type
Error 125: Operation canceled
Error 126: Required key not available
Error 127: Key has expired
Error 128: Key has been revoked
Error 129: Key was rejected by service
Error 130: Owner died
Error 131: State not recoverable
Error 132: Operation not possible due to RF-kill
Error 133: Memory page has hardware error
Error 134: Unknown error 134
Error 135: Unknown error 135
Error 136: Unknown error 136
Error 137: Unknown error 137
Error 138: Unknown error 138
Error 139: Unknown error 139
Error 140: Unknown error 140
Error 141: Unknown error 141
Error 142: Unknown error 142
Error 143: Unknown error 143

```
Error 144: Unknown error 144
Error 145: Unknown error 145
Error 146: Unknown error 146
Error 147: Unknown error 147
Error 148: Unknown error 148
Error 149: Unknown error 149
Error 150: Unknown error 150
Error 151: Unknown error 151
Error 152: Unknown error 152
Error 153: Unknown error 153
Error 154: Unknown error 154
Error 155: Unknown error 155
Error 156: Unknown error 156
Error 157: Unknown error 157
Error 158: Unknown error 158
Error 159: Unknown error 159
Error 160: Unknown error 160
Error 161: Unknown error 161
Error 162: Unknown error 162
Error 163: Unknown error 163
Error 164: Unknown error 164
Error 165: Unknown error 165
Error 166: Unknown error 166
Error 167: Unknown error 167
Error 168: Unknown error 168
Error 169: Unknown error 169
Error 170: Unknown error 170
Error 171: Unknown error 171
Error 172: Unknown error 172
Error 173: Unknown error 173
Error 174: Unknown error 174
Error 175: Unknown error 175
Error 176: Unknown error 176
Error 177: Unknown error 177
Error 178: Unknown error 178
Error 179: Unknown error 179
Error 180: Unknown error 180
Error 181: Unknown error 181
Error 182: Unknown error 182
Error 183: Unknown error 183
Error 184: Unknown error 184
Error 185: Unknown error 185
Error 186: Unknown error 186
Error 187: Unknown error 187
Error 188: Unknown error 188
Error 189: Unknown error 189
Error 190: Unknown error 190
Error 191: Unknown error 191
Error 192: Unknown error 192
Error 193: Unknown error 193
Error 194: Unknown error 194
Error 195: Unknown error 195
Error 196: Unknown error 196
Error 197: Unknown error 197
Error 198: Unknown error 198
Error 199: Unknown error 199
Error 200: Unknown error 200
```

```
Error 201: Unknown error 201
Error 202: Unknown error 202
Error 203: Unknown error 203
Error 204: Unknown error 204
Error 205: Unknown error 205
Error 206: Unknown error 206
Error 207: Unknown error 207
Error 208: Unknown error 208
Error 209: Unknown error 209
Error 210: Unknown error 210
Error 211: Unknown error 211
Error 212: Unknown error 212
Error 213: Unknown error 213
Error 214: Unknown error 214
Error 215: Unknown error 215
Error 216: Unknown error 216
Error 217: Unknown error 217
Error 218: Unknown error 218
Error 219: Unknown error 219
Error 220: Unknown error 220
Error 221: Unknown error 221
Error 222: Unknown error 222
Error 223: Unknown error 223
Error 224: Unknown error 224
Error 225: Unknown error 225
Error 226: Unknown error 226
Error 227: Unknown error 227
Error 228: Unknown error 228
Error 229: Unknown error 229
Error 230: Unknown error 230
Error 231: Unknown error 231
Error 232: Unknown error 232
Error 233: Unknown error 233
Error 234: Unknown error 234
Error 235: Unknown error 235
Error 236: Unknown error 236
Error 237: Unknown error 237
Error 238: Unknown error 238
Error 239: Unknown error 239
Error 240: Unknown error 240
Error 241: Unknown error 241
Error 242: Unknown error 242
Error 243: Unknown error 243
Error 244: Unknown error 244
Error 245: Unknown error 245
Error 246: Unknown error 246
Error 247: Unknown error 247
Error 248: Unknown error 248
Error 249: Unknown error 249
Error 250: Unknown error 250
Error 251: Unknown error 251
Error 252: Unknown error 252
Error 253: Unknown error 253
Error 254: Unknown error 254
Error 255: Unknown error 255
```

Información del sistema

Ejercicio 4. El comando del sistema `uname(1)` muestra información sobre diversos aspectos del sistema. Consultar la página de manual y obtener la información del sistema.

```
man 1 uname

UNAME(1)

NAME
     uname - print system information

SYNOPSIS
     uname [OPTION]...

DESCRIPTION
     Print certain system information.  With no OPTION, same as -s.

     -a, --all
          print all information, in the following order, except omit -p and -i if unknown:

     -s, --kernel-name
          print the kernel name

     -n, --nodename
          print the network node hostname

     -r, --kernel-release
          print the kernel release

     -v, --kernel-version
          print the kernel version

     -m, --machine
          print the machine hardware name

     -p, --processor
          print the processor type (non-portable)

     -i, --hardware-platform
          print the hardware platform (non-portable)

     -o, --operating-system
          print the operating system

     --help display this help and exit

----------

usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ uname -a
Linux usuarioso-virtualbox 5.4.0-62-generic #70-Ubuntu SMP Tue Jan 12 12:45:47 UTC 2021
x86_64 x86_64 x86_64 GNU/Linux
```

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


int main(){
   struct utsname info;
   if (uname (&info) == -1)){
      printf("Error!! %d - %s\n", errno, strerror(errno));
      return -1;
   }
   else {
      printf("SysName: %s\n", info.sysname);
      printf("Nodename: %s\n", info.nodename);
      printf("Release: %s\n", info.release);
      printf("Version: %s\n", info.version);
      printf("Machine: %s\n", info.machine);
   }
   return 1;
}

usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej5
SysName: Linux
Nodename: usuarioso-virtualbox
Release: 5.4.0-62-generic
Version: #70-Ubuntu SMP Tue Jan 12 12:45:47 UTC 2021
Machine: x86_64
```

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.

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


int main() {
        printf("Longitud máxima de argumentos: %li\n", sysconf(_SC_ARG_MAX));
        printf("Número máximo de hijos: %li\n", sysconf(_SC_CHILD_MAX));
        printf("Número máximo de ficheros: %li\n", sysconf(_SC_OPEN_MAX));
        return 1;
}
```

```
usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej6
Longitud máxima de argumentos: 2097152
Número máximo de hijos: 7592
Número máximo de ficheros: 8192

* Para ver la pagina del manual que nos interesa del comando hay que poner man 3 sysconf
```

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


int main() {
        printf("Número máximo de enlaces: %li\n", pathconf("/", _PC_LINK_MAX));
        printf("Tamaño máximo de una ruta: %li\n", pathconf("/", _PC_PATH_MAX));
        printf("Tamaño máximo de nombre de  fichero: %li\n", pathconf("/", _PC_NAME_MAX));
        return 1;
}


usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej7
Número máximo de enlace: 65000
Tamaño máximo de una ruta: 4096
Tamaño máximo de un nombre de  fichero: 255

PROFE DICE QUE . EN VEZ DE / EN TODOS MENOS RUTA PERO ME SALE IGUAL
```

Información del usuario

Ejercicio 8. El comando `id(1)` muestra la información de usuario real y efectiva. Consultar la página de manual y comprobar su funcionamiento.

```
man 1 id

Poner man -option para probarlo
```

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


int main(){
   printf("El UID real es: %d\n", getuid());
   printf("El UID efectivo es: %d\n", geteuid());
   return 1;
}

usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej9
El UID real es: 1000
El UID efectivo es: 1000

Podemos asegurar que está activado el bit setuid cuando no coinciden UID real y efectivo

*
man getuid
getuid() returns the real user ID of the calling process.
geteuid() returns the effective user ID of the calling process.
*
```

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

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <sys/utsname.h>
#include <sys/types.h>
#include <pwd.h>


int main(){
   printf("UID Real: %d\n", getuid());
   printf("UID efectivo: %d\n", geteuid());

   struct passwd *password = getpwuid(getuid());
   char* name = password->pw_name;
   char* passwd = password->pw_passwd;
   char uid = password->pw_uid;
   char gid = password->pw_gid;
   char* gecos = password->pw_gecos;
   char* dir = password->pw_dir;
   char* shell = password->pw_shell;

   printf("Nombre de usuario: %s\n", name);
   printf("Contraseña: %s\n", passwd);
   printf("ID de usuario: %d\n", uid);
   printf("ID de grupo: %d\n", gid);
   printf("Descripción de usuario: %s\n", gecos);
   printf("Home: %s\n", dir);
   printf("Shell: %s\n", shell);

   return 1;
}

usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej10
UID Real: 1000
UID efectivo: 1000
Nombre de usuario: usuarioso
Contraseña: x
ID de usuario: -24
ID de grupo: -23
Descripción de usuario: usuarioso
Home: /home/usuarioso
Shell: /bin/bash
```

Información horaria del sistema

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

```
man 1 date
    FORMAT controls the output.  Interpreted sequences are:
     %%   a literal %
     %a   locale's abbreviated weekday name (e.g., Sun)
     %A   locale's full weekday name (e.g., Sunday)
     %b   locale's abbreviated month name (e.g., Jan)
     %B   locale's full month name (e.g., January)
     %c   locale's date and time (e.g., Thu Mar  3 23:05:25 2005)
     %C   century; like %Y, except omit last two digits (e.g., 20)
     %d   day of month (e.g., 01)
     %D   date; same as %m/%d/%y
     %e   day of month, space padded; same as %_d
     %F   full date; same as %Y-%m-%d
     %g   last two digits of year of ISO week number (see %G)
     %G   year of ISO week number (see %V); normally useful only with %V
     %h   same as %b
     %H   hour (00..23)
     %I   hour (01..12)
     %j   day of year (001..366)
     %k   hour, space padded ( 0..23); same as %_H
     %l   hour, space padded ( 1..12); same as %_I
     %m   month (01..12)
     %M   minute (00..59)
     %n   a newline
     %N   nanoseconds (000000000..999999999)
     %p   locale's equivalent of either AM or PM; blank if not known
     %P   like %p, but lower case
     %q   quarter of year (1..4)
     %r   locale's 12-hour clock time (e.g., 11:11:04 PM)
     %R   24-hour hour and minute; same as %H:%M
     %s   seconds since 1970-01-01 00:00:00 UTC
     %S   second (00..60)
     %t   a tab
     %T   time; same as %H:%M:%S
     %u   day of week (1..7); 1 is Monday
     %U   week number of year, with Sunday as first day of week (00..53)
     %V   ISO week number, with Monday as first day of week (01..53)
     %w   day of week (0..6); 0 is Sunday
     %W   week number of year, with Monday as first day of week (00..53)
     %x   locale's date representation (e.g., 12/31/99)
     %X   locale's time representation (e.g., 23:13:48)
     %y   last two digits of year (00..99)
     %Y   year
     %z   +hhmm numeric time zone (e.g., -0400)
     %:z  +hh:mm numeric time zone (e.g., -04:00)
     %::z +hh:mm:ss numeric time zone (e.g., -04:00:00)
     %:::z numeric time zone with : to necessary precision (e.g., -04, +05:30)
     %Z   alphabetic time zone abbreviation (e.g., EDT)
```

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

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


int main(){
   time_t tiempo = time(NULL);
   printf("Tiempo desde el Epoch: 1970-01-01 00:00:00 +0000 (UTC) : %li segundos\n", tiempo);
   return 1;
}


usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej12
Tiempo desde el Epoch: 1970-01-01 00:00:00 +0000 (UTC) : 1635165927 segundos
```

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

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include <sys/time.h>
#include <unistd.h>
#include <string.h>


int main(){
   struct timeval tv;
   gettimeofday(&tv, NULL);
   int inicio  = tv.tv_usec;

   for(int i = 0; i < 1000000; ++i);

   gettimeofday(&tv, NULL);
   int final = tv.tv_usec;

   printf("Ha tardado %d milisegundos\n", final - inicio);
   return 1;
}


usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej13
Ha tardado 2135 milisegundos
```

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

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include <sys/time.h>
#include <unistd.h>
#include <string.h>


int main(){
   time_t t = time(NULL);
   struct tm* tiempo = localtime(&t);

   printf("Año actual: %d \n", 1900 + tiempo->tm_year);
   return 1;
}


usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej14
Año actual: 2021
```

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

```
#include <stdio.h>
#include <errno.h>
#include <time.h>
#include <sys/time.h>
#include <unistd.h>
#include <string.h>
#include <locale.h>


int main(){
   setlocale(LC_TIME, "es_ES");
   time_t t = time(NULL);
   if(t == -1){
      perror("Error");
      return -1;
   }

   struct tm* tiempo = localtime(&t);
   printf("Año actual: %d \n", 1900 + tiempo->tm_year);

   char *buffer;
   strftime(buffer, 100, "%A, %d de %B de %Y, %R", tiempo); //%R es %H:%M
   printf("Estamos a: %s\n", buffer);

   return 1;
}


usuarioso@usuarioso-virtualbox:~/Desktop/ASOR$ ./ej15
Año actual: 2021
Estamos a: Lunes, 25 de Octubre de 2021, 15:47
```

Nota: Para establecer la configuración regional (*locale*, como idioma o formato de hora) en el programa según la configuración actual, usar la función `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"`.