**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. **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;  } |
| --- |

| #include <sys/types.h>  #include <unistd.h>  #include <stdio.h>  int main() {  if(setuid(0)){  perror("Error");  }  return 1;  }  ***Salida:*** *Error: Operation not permitted* |
| --- |

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

| #include <sys/types.h>  #include <unistd.h>  #include <stdio.h>  #include <errno.h>  #include <string.h>  int main() {  if(setuid(0)){  printf("Codigo de error:%i\n",errno);  printf("Mensaje asociado:%s\n", strerror(errno));  perror("Error");  }  return 1;  }  Salida:  Código de error:1  Mensaje asociado: Operation not permitted  Error: Operation not permitted |
| --- |

***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 <sys/types.h>  #include <unistd.h>  #include <stdio.h>  #include <errno.h>  #include <string.h>  int main(){  for(int i=0;i<255;i++){  printf("Error%i:%s\n",i, strerror(i));  }  return 1;  }  **Salida:** Solo imprime 133 errores, después se limita a decir Unknow error |
| --- |

# **Información del sistema**

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

| **Manual:** man uname  **uname -s (kernel name)**  Linux  **uname -n (nodename)**  pto0601  **uname -r (kernel release)**  5.15.0-39-generic  **uname -v (kernel version)**  #42-Ubuntu SMP Thu Jun 9 23:42:32 UTC 2022  **uname -m (machine)**  x86\_64  **uname -p (processor)**  x86\_64  **uname -i (hardware platform)**  x86\_64  **uname -o (operating system)**  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 <sys/types.h>  #include <unistd.h>  #include <stdio.h>  #include <stdlib.h>  #include <sys/utsname.h>  int main(){  struct utsname buf;  if(uname(&buf)){  perror("Error");  exit(1);  }  printf("Kernel name:%s\n", buf.sysname);  printf("Nodename:%s\n", buf.nodename);  printf("Kernel release:%s\n", buf.release);  printf("Kernel version:%s\n", buf.version);  printf("Machine:%s\n", buf.machine);  return 0;  }  **Salida:**  Kernel name:Linux  Nodename:pto0601  Kernel release:5.15.0-39-generic  Kernel version:#42-Ubuntu SMP Thu Jun 9 23:42:32 UTC 2022  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 abiertos.

| #include <sys/types.h>  #include <unistd.h>  #include <stdio.h>  #include <stdlib.h>  int main(){  long n;  if((n=sysconf(\_SC\_ARG\_MAX))==-1){  perror("Error en ARG\_MAX");  exit(1);  }  else printf("ARG\_MAX: %li\n",n);  if((n=sysconf(\_SC\_CHILD\_MAX))==-1){  perror("Error en CHILD\_MAX");  exit(1);  }  else printf("CHILD\_MAX: %li\n",n);  if((n=sysconf(\_SC\_OPEN\_MAX))==-1){  perror("Error en OPEN\_MAX");  exit(1);  }  else printf("OPEN\_MAX: %li\n",n);  return 0;  }  **Salida:**  ARG\_MAX: 140062642103024  CHILD\_MAX: 140062642103024  OPEN\_MAX: 140062642103024 |
| --- |

***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 <sys/types.h>  #include <unistd.h>  #include <stdio.h>  #include <stdlib.h>  int main(){  long n;  if((n=pathconf(".",\_PC\_LINK\_MAX))==-1){  perror("Error en \_LINK\_MAX");  exit(1);  }  else printf("\_LINK\_MAX: %li\n",n);  if((n=pathconf(".",\_PC\_PATH\_MAX))==-1){  perror("Error en PATH\_MAX");  exit(1);  }  else printf("PATH\_MAX: %li\n",n);  if((n=pathconf(".",\_PC\_NAME\_MAX))==-1){  perror("Error en \_NAME\_MAX");  exit(1);  }  else printf("\_NAME\_MAX: %li\n",n);  return 0;  }  **Salida:**  \_LINK\_MAX: 65000  PATH\_MAX: 4096  \_NAME\_MAX: 255 |
| --- |

# **Información del usuario**

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

| **Manual**: man id  **id -z (context)**  **id -g (group)**  **id -G (groups)**  **id -n (name)**  **id -r (real ID)**  **id -u (user)**  Salida:  uid=1555(usuario\_local) gid=100(users) grupos=100(users),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),108(netdev),113(bluetooth),118(scanner),126(sambashare),132(render) |
| --- |

***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 <sys/types.h>  #include <unistd.h>  #include<stdio.h>  int main(){  printf("ID real:%i\n",getuid());  printf("ID efectivo:%i\n",geteuid());  printf("UID real:%i\n",getgid());  printf("UID efectivo:%i\n",getegid());  return 0;  }  **Salida:**  ID real:1555  ID efectivo:1555  UID real:100  UID efectivo:100 |
| --- |

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

| #include <sys/types.h>  #include <unistd.h>  #include<stdio.h>  #include <stdlib.h>  #include <pwd.h>  int main(){  uid\_t id=getuid();  printf("Usuario ID real:%i\n",id);  struct passwd\* pw;  if((pw=getpwuid(id))==NULL){  perror("Error getpwuid");  exit(1);  }  printf("Nombre de usuario: %s\n",pw->pw\_name);  printf("Home:%s\n",pw->pw\_dir);  printf("Descripcion:$s\n",pw->pw\_gecos);  printf("Id efectivo:%i\n",getuid);  printf("Id de grupo real:%i\n",getgid());  printf("Id de grupo efectivo:%i\n",getegid());  return 0;  }  Salida:  Usuario ID real:1565  Nombre de usuario: usuario\_vms  Home:/home/usuario\_vms  Descripcion:$s  Id efectivo:444395728  Id de grupo real:100  Id de grupo efectivo:100 |
| --- |

# **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.

| Manual: man date  %d= dias  %m= mes  %y = año |
| --- |

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

| #include <sys/types.h>  #include <unistd.h>  #include<stdio.h>  #include <stdlib.h>  #include <time.h>  int main(){  time\_t t=time(NULL);  if(t==(time\_t)-1){  perror("Error en time");  exit(1);  }  printf("Tiempo en segundos desde Epoch:%li\n",t);  return 0;  }  Salida:  Tiempo en segundos desde Epoch:1667821488 |
| --- |

***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 <sys/types.h>  #include <unistd.h>  #include<stdio.h>  #include <stdlib.h>  #include <sys/time.h>  int main(){  struct timeval timeIni;  if(gettimeofday(&timeIni,NULL)){  perror("Error en gettimeofday(primera llamada)");  exit(1);  }  int v=0;  for(int i=0;i<1e6;i++){  v++;  }  struct timeval timeFin;  if(gettimeofday(&timeFin,NULL)){  perror("Error en gettimeofday(segunda llamada)");  exit(1);  }  printf("Segundos transcurridos:%li\n",timeFin.tv\_sec-timeIni.tv\_sec);  printf("Microsegundos transcurridos:%li\n",timeFin.tv\_usec-timeIni.tv\_usec);  return 0;  }  Salida:  Segundos transcurridos:0  Microsegundos transcurridos:4476 |
| --- |

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

| #include <sys/types.h>  #include <unistd.h>  #include<stdio.h>  #include <stdlib.h>  #include <time.h>  int main(){  time\_t segEpoch=time(NULL);  if(segEpoch==(time\_t)-1){  perror("Error en funcion time");  exit(1);  }  struct tm\* time=localtime(&segEpoch);  printf("Estamos en el año:%i\n",1900+time->tm\_year);  return 0;  }  Salida  Estamos en el año:2022 |
| --- |

***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 <sys/types.h>  #include <unistd.h>  #include<stdio.h>  #include <stdlib.h>  #include <time.h>  #include <locale.h>  int main(){  time\_t segEpoch=time(NULL);  if(segEpoch==(time\_t)-1){  perror("Error en funcion time");  exit(1);  }  struct tm\* time=localtime(&segEpoch);  char\* s;  setlocale(LC\_ALL,"");  strftime(s,200,"%A,%e de %B de %Y, %R",time);  printf("%s\n",s);  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".