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 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.

```
#include <stdio.h>
#include <errno.h>
#include <unistd.h>

int main() {
   if(setuid(0)){
      perror(“Error en setuid”);
   }
   return 1;
}
```

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 <string.h>
#include <errno.h>

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

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 <string.h>
#include <errno.h>

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

Información del sistema

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

```
[cursoredes@localhost ~]$ uname --kernel-name
Linux
[cursoredes@localhost ~]$ uname --nodename
localhost.localdomain
[cursoredes@localhost ~]$ uname --kernel-release
3.10.0-862.11.6.el7.x86_64
[cursoredes@localhost ~]$ uname --kernel-version
#1 SMP Tue Aug 14 21:49:04 UTC 2018
[cursoredes@localhost ~]$ uname --machine
x86_64
[cursoredes@localhost ~]$ uname --processor
x86_64
[cursoredes@localhost ~]$ uname --hardware-platform
```

```
x86_64
[cursoredes@localhost ~]$ uname --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 <string.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/utsname.h>

int main() {
  struct utsname buf;
  if(uname(&buf) == 0){
    printf("Sistema operativo: %s\n", buf.sysname);
    printf("Nombre de nodo: %s\n", buf.nodename);
    printf("Release de sistema operativo: %s\n", buf.release);
    printf("Version: %s\n", buf.version);
    printf("Id de hardware: %s\n", buf.machine);
  }
  else{
    printf("Error %d: %s\n", errno, strerror(errno));
    exit(1);
  }

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

int main() {
  long valor;
  if((valor = sysconf(_SC_ARG_MAX)) == -1){
    printf("Error %d: %s\n", errno, strerror(errno));
    exit(1);
  }
```

```
  else {
    printf("Longitud maxima de argumentos: %li\n", valor);
  }

  if((valor = sysconf(_SC_CHILD_MAX)) == -1){
    printf("Error %d: %s\n", errno, strerror(errno));
    exit(1);
  }
  else {
    printf("Numero maximo de hijos: %li\n", valor);
  }


  if((valor = sysconf(_SC_OPEN_MAX)) == -1){
    printf("Error %d: %s\n", errno, strerror(errno));
    exit(1);
  }
  else {
    printf("Numero maximo de ficheros abiertos: %li\n", valor);
  }


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

int main() {
  long valor;
  if((valor = pathconf(".",_PC_LINK_MAX)) == -1){
     printf("Error %d: %s\n", errno, strerror(errno));
     exit(1);
  }
  else{
    printf("Numero maximo de enlaces: %li\n", valor);
  }

  if((valor = pathconf(".",_PC_MAX_INPUT)) == -1){
     printf("Error %d: %s\n", errno, strerror(errno));
     exit(1);
  }
  else{
```

```
    printf("Longitud maxima de ruta: %li\n", valor);
  }

  if((valor = pathconf(".",_PC_NAME_MAX)) == -1){
     printf("Error %d: %s\n", errno, strerror(errno));
     exit(1);
  }
  else{
    printf("Tamano maximo de nombre: %li\n", valor);
  }

  return 0;

}
```

Información del usuario

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

```
$ id -a
uid=1000(cursoredes) gid=1000(cursoredes) groups=1000(cursoredes),10(wheel),983(wireshark)
```

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*?

```
 GNU nano 2.3.1            File: ej9.c

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

int main(){
  printf("ID real de usuario: %d\n", getuid());
  printf("ID ifectivo de usuario: %d\n", geteuid());
  return 0;
}


Si el ID real y efectivo no coinciden, el bit setuid está activado.
```

Ejercicio 10. Modificar el programa anterior para que muestre además el nombre de usuario, el

directorio *home* y la descripción del usuario.

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

int main(){
  printf("ID real de usuario: %d\n", getuid());
  printf("ID ifectivo de usuario: %d\n", geteuid());

  uid_t user = getuid();
  struct passwd *pwd =  getpwuid(user);
  printf("Nombre de usuario: %s\n", pwd->pw_name);
  printf("Directorio home: %s\n", pwd->pw_dir);
  printf("Descripcion: %s\n", pwd->pw_gecos);

  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.

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 <string.h>
#include <errno.h>
#include <stdlib.h>
#include <time.h>

int main(){
  time_t tmp;
  if((tmp=time(NULL))==-1){
     printf("Error %d: %s\n", errno, strerror(errno));
     exit(1);
  }
  else{
    printf("Segundos desde Epoch: %li\n", tmp);
  }
```

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

int main(){
  struct timeval tv_ini;
  if(gettimeofday(&tv_ini,NULL)==-1){
     printf("Error %d: %s\n", errno, strerror(errno));
     exit(1);
  }
  else{
    int i,j=0;
    for(i=0;i<1000000; ++i){
      j++;
    }
  }
  struct timeval tv_fin;
  if(gettimeofday(&tv_fin,NULL)==-1){
     printf("Error %d: %s\n", errno, strerror(errno));
     exit(1);
  }
  printf("Duracion bucle: %li\n", tv_fin.tv_usec-tv_ini.tv_usec);

  return 0;
}
```

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

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

int main() {
  time_t t = time(NULL);
  struct tm *local_t = localtime(&t);
  int anyo = local_t->tm_year;

  printf("Año: %i\n",1900+ anyo);
  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 <locale.h>
#include <time.h>

int main() {
  setlocale(LC_TIME,"es_ES.UTF-8");
  time_t tmp = time(NULL);
  struct tm *local_t = localtime(&tmp);

  char buf[100];
  strftime(buf, 100, "%A, %e de %B de %Y, %H:%M", local_t);
  printf("%s\n", buf);
  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"`.