PRÁCTICA: Creación de aplicaciones en C (I)

Objetivos:

-Manejo del Lenguaje C

-Uso de funciones de librería y ficheros cabecera

-Etapas y uso del compilador: gcc

-Manejo de ficheros usando la librería estándar de C

-Crear librerías dinámicas y estáticas

-Gestionar los proyectos con make

Sabe instalar lo necesario para programar en c y sus librerías

Instala o comprueba que están instalados los siguientes paquetes o aplicaciones:

Compilador, librerías y documentación necesaria para programar en C:

sudo apt update

sudo apt install build-essential // Con este paquete se instalan los paquetes esenciales para programar en C

Además de ayuda y documentación extra:

sudo apt install debian-keyring //GnuPG keys of Debian Developers,las **claves GnuPG** (GNU Privacy Guard) de los desarrolladores y mantenedores se usan principalmente para **verificar la autenticidad e integridad de los paquetes**.

sudo apt install gcc-doc //Documentation for the GNU compilers (gcc, gobjc, g++)

sudo apt install glibc-doc //Embedded GNU C Library: Documentation

sudo apt install manpages-dev //Manual pages about using GNU/Linux for development

Instala la ayuda y documentación de la APIs de POSIX:

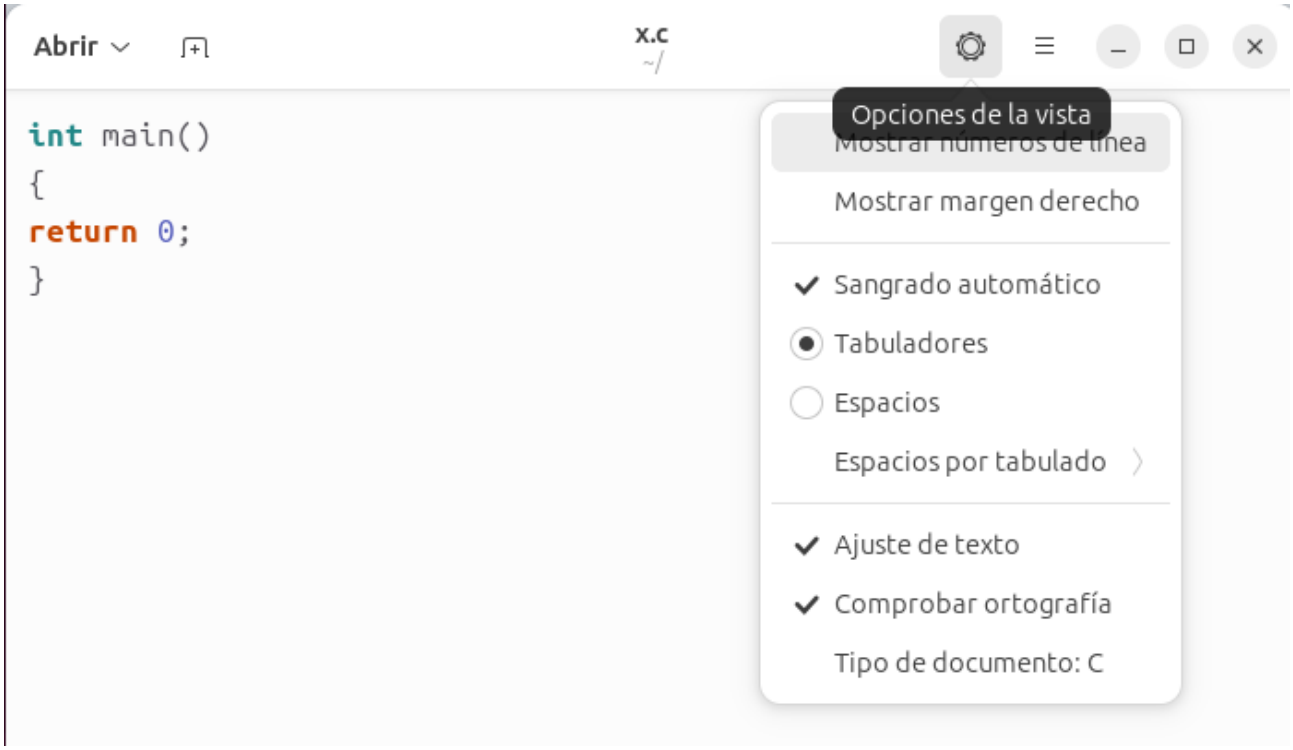
sudo apt install manpages-posix //Manual pages about using POSIX system

POSIX (Portable Operating System Interface, y X viene de UNIX como seña de identidad de la API), norma que define una interfaz estándar del sistema operativo y el entorno, incluyendo un intérprete de comandos (o "shell"), y programas de utilidades comunes para apoyar la portabilidad de las aplicaciones a nivel de código fuente. El nombre POSIX surgió de la recomendación de Richard Stallman, que por aquel entonces en la década de 1980 formaba parte del comité de IEEE.

Sabe usar un editor de texto que reconoce C

Editor de C:

```
$gnome-text-editor
```



Sabe manejarse entre los ficheros intermedios de las diferentes etapas de compilación

1.- Crea un programa fuente “pi.c” que contenga el código siguiente:

```
#include <stdio.h>
#define PI 3.1415
int main ()
{
char c;
printf("¿Quieres conocer al número PI? (S/N)");
scanf("%c",&c);
if (c=='S' || c=='s') printf("%f\n",PI);
printf("Fin de programa \n");
return 0;
}
```

a) Preprocesa el programa pi.c y analiza el código.
gcc pi.c -E -o pi.i
Otra opción:
cpp pi.c pi.i
b) Obtén pi.s (ensamblado).
gcc -o pi.s -S pi.c
c) Obtén pi.o (objeto).
gcc -o pi.o -c pi.s
d) Obtén el ejecutable pi a partir del fuente.
gcc -o pi pi.o

Sabe usar la ayuda a la programación

Utilización del man:

A partir de ahora vamos a utilizar comandos del "*bash*", funciones de librería de C también conocidas como las "API-s de C", funciones de librería del SO Linux también conocidas como las "API-s de Linux". Para obtener la ayuda correcta dentro del man, vamos a analizar las secciones de la ayuda del man:

$man man
NOMBRE
man - interfaz de los manuales de referencia del sistema

SINOPSIS
man [opciones de man] [[sección] página ...] ...
man -k [opciones de apropos] regexp ...
man -K [opciones de man] [sección] term ...
man -f [whatis opciones] página ...
man -l [opciones de man] archivo ...
man -w|-W [opciones de man] página ...

DESCRIPCIÓN
man es el paginador de manuales del sistema. Cada argumento de página dado a man normalmente es el nombre de un programa, utilidad o función. La página de manual asociada con cada uno de estos argumentos es, pues, encontrada y mostrada. Si se proporciona una sección, man mirará solo en esa sección del manual. La acción predeterminada es buscar en todas las secciones disponibles siguiendo un orden predefinido (véase DEFAULTS), y mostrar solo la primera página encontrada, incluso si la página existe en varias secciones.

La tabla de abajo muestra los números de sección del manual seguidos por los tipos de página que contienen.
Nº de Sección - Tema

1 Programas ejecutables u órdenes de la shell
2 Llamadas al sistema (funciones proporcionadas por el núcleo)
3 Llamadas a biblioteca (funciones dentro de bibliotecas de programa)
4 Archivos especiales (normalmente se encuentran en /dev)
5 Formatos de archivo y convenios, p.e. /etc/passwd
6 Juegos
7 Miscelánea (incluidos paquetes de macros y convenios), p.e. man(7), groff(7), man-pages(7)
8 Órdenes de administración del sistema (normalmente solo para root)
9 Rutinas del núcleo [No estándar]

EJEMPLOS
$man ls, Enseña la página de manual para el ítem (programa) ls.

$ man 1 write, Busca "write" en la sección 1 del manual.

$ man 2 write, Busca “write” en la sección 2 del manual.
$ man 5 shadow, Busca “shadow” en la sección 5 del manual.
$ man -a passwd, Busca “passwd” en todas las secciones del manual. Saltando de sección en sección. Aparece información de todas(-a de “all”) las secciones

Sabe dónde están los ficheros cabeceras y librerías en linux

1.- ¿Qué contiene el fichero stdio.h?
whereis stdio.h
stdio.h: /usr/include/stdio.h
kepa@kepa-Precision-3551:~$ gnome-text-editor /usr/include/stdio.h
cabeceras de las funciones de entrada y salida como printf
2.- Localiza el prototipo/declaración/cabecera de la función printf en /usr/include/stdio.h
extern int printf (const char *__restrict __format, ...);
3.- ¿Dónde están las librerías o bibliotecas libc.so y libc.a estándar de C en linux?
whereis libc.a
libc.a: /usr/lib/x86_64-linux-gnu/libc.a
whereis libc.so
libc.so: /usr/lib/x86_64-linux-gnu/libc.so
4.- Para más información échale un vistazo a la documentación de la librería estándar de C.
http://www.gnu.org/software/libc/manual/html_mono/libc.html

Todo en el main

Sabe abrir un fichero en modo lectura

1.- Realiza un programa en C que preguntando al usuario el nombre de un fichero calcule su tamaño en bytes utilizando las funciones **fopen, fgetc y fclose** de las librerías de C . Compila y ejecuta el ejecutable pasándole un fichero de texto, y comprueba que coincide con la salida del comando “ls -l” como se indica en el ejemplo:

$ **gcc -o longFich longFich.c**
$ **./longFich**
¿De qué fichero deseas conocer el tamaño?longFich.c
El tamaño del fichero es de 483
$ **ls -l longFich.c**
-rw-r--r-- 1 kepa kepa **483** 2013-03-01 12:36 longFich.c

```
#include <stdio.h>
int main() {
  FILE * entrada;
  char nombre[100];
  int contador=0;
  char c;
  printf("¿De qué fichero deseas conocer el tamaño?");
  __fpurge(stdin);
  gets(nombre); // o scanf("%s",nombre)
  //abrir fichero
  entrada=fopen(nombre,"r");
  //recorrer el fichero
  c=fgetc(entrada);
  while(c!=EOF){
    contador=contador+1;
    c=fgetc(entrada);
  }
  //cerrar fichero
  fclose(entrada);
  printf("El tamaño del fichero es de %d",contador);
  return 0;
}
```

Sabe abrir un fichero en modo escritura

1.- Realiza un programa en C que preguntando al usuario por el nombre de un fichero escriba en este el abecedario tanto en minúsculas como en mayúsculas, cada uno en una fila, utilizando las funciones **fopen, fputc y fclose** de las librerías de C .Compila y ejecuta el ejecutable, y comprueba con **less** que la salida es como se indica en el ejemplo:

Ejemplo:
$ **gcc -o abece abece.c**
$ **./abece**
Introduce el nombre del fichero a crear: prueba.txt
$ **less prueba.txt**
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

```
#include <stdio.h>
int main() {
  FILE * salida;
  char nombre[100];
  char c;
  printf("Introduce el nombre del fichero a crear:");
  __fpurge(stdin);
  gets(nombre);
  //abrir fichero
  salida=fopen(nombre,"w");
```

```
    //recorrer el fichero
    for(c='a';c<='z';c++) {
      fputc(c,salida);
    }
    c='\n';
    fputc(c,salida);
    for(c='A';c<='Z';c++) {
      fputc(c,salida);
    }
    //cerrar fichero
    fclose(salida);
    return 0;
  }
```

Sabe utilizar los ejecutables en otros programas:

1.-Ejecuta el siguiente programa por lotes o script de Bash que ejecute los dos ejecutables realizados en los ejercicios anteriores. Si el primer ejecutable se ejecuta correctamente que se ejecute el segundo. Para ello si el ejecutable “abece” termina correctamente, ejecute “longFich”, y si no muestre un mensaje de que el primer programa ha fallado. El programa se detalla abajo. Comprueba su uso correcto.

$gnome-text-editor **bashscript.sh**

```
./abece
if [ $? -eq 0 ] ; then
./longFich
if [ $? -eq 0 ] ; then
    echo -e "\nwell done"
    exit 0
else
    echo -e "\nfailed"
    exit 1
else
echo -e "\nfailed"
exit 1
fi
```

$chmod u+x ./bashscript.sh

$./bashscript.sh

main+módulos

Sabe gestionar ficheros usando librerías propias

1.-El siguiente programa principal “numeroaes.c” está incompleto:

```
#include <stdio.h>
int ffichnumcarac(char c, FILE * fp);


int main()
{
char nombre[80],c;
FILE *fp;
int contador=0;
printf ("¿De qué fichero deseas conocer el número de aes?");
scanf("%s",nombre);
fp=fopen(nombre,"r");
if (fp==NULL)
        {
            printf("No se puede abrir el fichero %s \n",nombre);
            return -1;
         }
else
        {       contador=ffichnumcarac('a',fp);
                fclose(fp);
                printf("El número de a-s es de %d",contador);
        }
return 0;
}
int ffichnumcarac(char c, FILE * fp){
}
```

Se pide:

a)Completa la función "ffichnumcarac" dentro de "numeroaes.c". A esta función se le pasarán como parámetros el carácter a buscar y el descriptor de fichero. Y nos devolverá el número de veces que aparece dicho carácter en el fichero. Compila y ejecuta dicho programa principal *numeroaes*.

```
int ffichnumcarac(char c, FILE * fp){
        int contador=0;
        char car;
        //recorrer el fichero
        car=fgetc(fp);
        while(car!=EOF){
                if(car==c){
                        contador=contador+1;
                }
        car=fgetc(fp);
        }
        return contador;
}
```

a1) Ahora introduce dicha función "ffichnumcarac" en un módulo que se llamará "general.c". Compila y ejecuta dicho programa principal *numeroaes*.

general.c

```
int ffichnumcarac(char c, FILE * fp){
        int contador=0;
        char car;
        //recorrer el fichero
        car=fgetc(fp);
        while(car!=EOF){
                if(car==c){
                        contador=contador+1;
                }
        car=fgetc(fp);
        }
        return contador;
}
```

general.h

```
int ffichnumcarac(char c, FILE * fp);
```

numeroaes.c

```
#include <stdio.h>
#include "general.h"
int main()
{
char nombre[80],c;
FILE *fp;
int contador=0;
printf ("¿De qué fichero deseas conocer el número de aes?");
scanf("%s",nombre);
fp=fopen(nombre,"r");
if (fp==NULL)
    {
        printf("No se puede abrir el fichero %s \n",nombre);
        return -1;
     }
else
    {   contador=ffichnumcarac('a',fp);
          fclose(fp);
          printf("El número de a-s es de %d",contador);
    }
return 0;
}
```

$gcc -o numeroaes general.c numeroaes.c -I.

o

$gcc -o general.o -c general.c

$gcc -o numeroaes.o numeroaes.c -I.

$gcc -o numeroaes general.o numeroaes.o

librerías estáticas y dinámicas

b) Ahora introduce dicha función "ffichnumcarac" en un módulo que se llamará "general.c". Finalmente introduce dicho módulo en una biblioteca estática cuyo nombre será "libgeneral.a". Y crea el ejecutable *numeroaes* usando dicha librería estática. Para ello modifica el programa principal para que la compilación tenga éxito.

general.h

```
int ffichnumcarac(char c, FILE * fp);
```

numeroaes.c

```
#include <stdio.h>
#include "general.h"
int main()
{
char nombre[80],c;
FILE *fp;
int contador=0;
printf ("¿De qué fichero deseas conocer el número de aes?");
scanf("%s",nombre);
fp=fopen(nombre,"r");
if (fp==NULL)
    {
        printf("No se puede abrir el fichero %s \n",nombre);
        return -1;
     }
else
    {   contador=ffichnumcarac('a',fp);
          fclose(fp);
          printf("El número de a-s es de %d",contador);
    }
return 0;
}
```

general.c

```
#include <stdio.h>

int ffichnumcarac(char c, FILE * fp){

          int contador=0;
```

```
        char car;
        //recorrer el fichero
        car=fgetc(fp);
        while(car!=EOF){
                if(car==c){
                        contador=contador+1;
                }
        car=fgetc(fp);
        }
        return contador;
}
```

```
gcc -o general.o -c general.c

ar -rv ./libgeneral.a general.o
ranlib ./libgeneral.a
nm -s ./libgeneral.a
gcc -I. -L. numeroaes.c -lgeneral -o numeroaes -static
./numeroaes
```

c) Ahora introduce dicha función “ffichnumcarac” en un módulo que se llamará “general.c”. Finalmente introduce dicho módulo en una biblioteca dinámica cuyo nombre será “libgeneral.so”. Y crea el ejecutable *numeroaes* usando dicha librería dinámica. Para ello modifica el programa principal para que la compilación tenga éxito.

```
gcc -o general.o -c general.c -fPIC

ld -o ./libgeneral.so general.o -shared

gcc -I. -L. -o numeroaes2 numeroaes.c -lgeneral -Bdynamic
./numeroaes2
//dice:./numeroaes2: error while loading shared libraries: libgeneral.so: cannot open shared object
file: No such file or directory

LD_LIBRARY_PATH=$LD_LIBARY_PATH:/home/$USER
export LD_LIBRARY_PATH
./numeroaes2
```

Sabe ejecutar un comando de bash shell desde C

¿Para qué sirve la función system?. Ejecuta el siguiente código:

```
#include<stdio.h>
#include <stdlib.h>
int main(){
system("ls -l");
return 0;
}
```

El comando system permite escribir desde el código de C comandos en la terminal, en este caso, imprimiría un lista de los ficheros en el directorio actual.