

Cuaderno de prácticas

Prácticas en el lenguaje C .

Programación para Sistemas

Lenguajes y Sistemas Informáticos e Ingeniería de Software
ETSI Informáticos
Universidad Politécnica de Madrid

Curso 2019-2020
Semestre septiembre-enero

Contents

0.1	Normas de entrega y presentación	1
1	Programación C	3
1.1	Instrucciones	3
1.2	Evaluación	5
1.3	Documentación a entregar	5
1.3.1	Programa bocabajo	5
	Agradecimientos	7

0.1 Normas de entrega y presentación

Cada tarea exige la entrega de una serie de *entregables* (ficheros). Cada entregable debe ser incluido en el formato especificado en el enunciado de la tarea, además de los dos ficheros especificados a continuación. La codificación de caracteres de todos los ficheros deberá ser UTF8 por lo que debes configurar el editor de texto que uses para que la codificación de caracteres sea la indicada¹.

Además de los ficheros específicos de cada tarea, cada entrega ha de incluir obligatoriamente el fichero siguiente:

¹El programa `iconv` convierte entre diferentes codificaciones y casi todos los editores de texto pueden personalizarse para que soporten y generen la codificación deseada.

autores.txt Fichero con los datos de los autores de la tarea. Cada integrante del grupo debe ocupar una línea en dicho fichero. Cada línea presentará varios campos de caracteres separados por el caracter “:”. Los campos son número de matrícula, primer apellido, segundo apellido y nombre. Veamos un ejemplo:

```
910347:García:de la Torre:María Dolores  
930019:Hernado:Pulido:Isabel
```

1 Programación C

Se desea que realice un pequeño programa en C que le permitirán afianzar sus conocimientos sobre este lenguaje. En las siguientes secciones se describirá en detalle cada uno de los programas a realizar.

1.1 Instrucciones

En la presente sección se recogen ciertos aspectos del comportamiento general, que serán de aplicación a todos los programas a realizar.

Identificación y compilación de los programas

Cada programa será identificado por el nombre que recibirá su ejecutable (ej. “bocabajo”). Cada programa derivará, principalmente, de un único fichero fuente en C del mismo nombre, pero con extensión “.c” (ej. “bocabajo.c”).

Además, puede depender de los ficheros `auxiliar.c` `auxiliar.h`, bien sean los proporcionados por el código fuente de apoyo o hechos o modificados por el alumno. Pero la ejecución del siguiente mandato

```
gcc -Wall -Wextra -o ejemplo ejemplo.c auxiliar.c
```

(donde ejemplo es el nombre del programa a realizar) debe funcionar correctamente y crear el programa ejecutable ejemplo

Es obligatorio que la anterior compilación funcione, en caso contrario la nota es cero puntos.

Argumentos de los programas

Todo programa es invocado escribiendo el nombre del ejecutable correspondiente. A la derecha de este nombre (y separado por blancos o tabuladores) podemos escribir otros textos, que denominamos argumentos. Estos argumentos estarán disponibles para ser usados por el proceso que se ejecuta (el programa en ejecución).

Si el programa fue escrito en C, los argumentos podrán ser accedidos a

```
main(int argc, char*argv[])
```

La manera en que un programa utiliza sus argumentos puede ser muy variada, pero generalmente se utilizan para parametrizar su propio comportamiento.

A continuación se describe la sintaxis empleada en la descripción de los argumentos de los programas.

-x

Esta es la notación corta, un signo menos (‘-’) y una letra (en este caso una ‘x’). Se indica una opción del programa.

--help

Esta es la notación larga, dos signos menos (‘--’) y una palabra (en este caso ‘help’). Se indica una opción del programa.

-t segundos

Las opciones se usan para condicionar el comportamiento del programa. Algunas opciones pueden indicar cómo interpretar el siguiente argumento como parámetro del programa.

[arg]

Mediante esta notación se indica una parte opcional que puede aparecer o no.

arg...

Mediante esta notación se indica una parte que puede aparecer una o varias veces.

Opciones de los programas

De forma general, todos los programas que se pide que usted desarrolle debe admitir la siguiente opción.

-h | --help

Todo programa debe reconocer el caso en que sea invocado con esta opción como único argumento. El usuario está solicitando ayuda. El programa debe emitir por la **salida estándar** un texto breve indicando qué hace el programa y qué parámetros admite. A continuación, el programa debe terminar correctamente, esto es, con valor de terminación cero.

Valores de terminación de los programas

No existe acuerdo en el significado de los valores de terminación exceptuando que todo programa que termina correctamente, lo hace con el valor de terminación cero (man 3 **exit**).

No obstante, para esta práctica, intentaremos adoptar el intento de estandarización realizado por BSD con `sysexit.h`.

Todo programa debe realizar ciertas comprobaciones sobre la corrección de sus argumentos, datos de entrada, servicios que utiliza, etc. Si alguna comprobación falla, debe emitir por el **estándar error** un mensaje de error de dos líneas con el siguiente formato:

```
"%s: Error(%s), %s.\n"
"%s+ %s.\n"
```

Cada línea irá prefijada con el nombre del programa que la emite. La primera indicará el código simbólico del error y una descripción genérica del mismo. La segunda línea presentará una descripción detallada de la razón de dicho error.

Ejemplo

```
bocabajo: Error(EX_NOINPUT), no se puede abrir la entrada. "No such file or directory"
bocabajo+ El fichero "ghhjhh" no puede ser leído.
```

A continuación, el programa debe terminar con el valor de terminación indicado.

Recomendaciones generales

- Allá donde se le sugiera que consulte el manual, consúltelo. Le ahorrará mucho tiempo.
- Respete el tipo del texto (mayúsculas y minúsculas) allá donde, en este documento, se presente un formato, un nombre de fichero, de variable, etc.
- Sea sumamente estricto con los formatos de entrada y salida mencionados en este documento. Sólo se podrá evaluar correctamente su práctica si se ajusta a los formatos indicados.

1.2 Evaluación

La entrega de los ficheros es obligatoria. La valoración total del conjunto de la práctica será de 100 puntos.

Se pueden realizar pruebas adicionales a las ya realizadas mediante el tester.

Los profesores podrían pedir una entrevista con los componentes de los grupos si lo consideraran necesario. Dichas entrevistas durarían entre 5 y 10 minutos con cada grupo durante la cuál se realizarían algunas preguntas sobre el diseño de los *programas* y el conocimiento de los mismos.

1.3 Documentación a entregar

Los siguientes ficheros son de entrega obligatoria para esta tarea:

autores.txt

Descrito en la sección 0.1.

auxiliar.c

Este fichero fuente podrá ser utilizado para contener funciones auxiliares, útiles para varios de los programas. Si no va a utilizar este fichero, debe dejarlo tal cual lo encuentre.

auxiliar.h

Este fichero de cabecera deberá contener los prototipos de las funciones auxiliares definidas en `auxiliar.c`. Los programas que deseen utilizar dichas funciones deberán incluir este fichero de cabecera. Si no va a utilizar este fichero, debe dejarlo tal cual lo encuentre.

bocabajo.c

Fichero fuente correspondiente al programa `bocabajo`. Este fichero deberá existir.

1.3.1 Programa **bocabajo**

```
bocabajo [ fichero... ]
```

Puede recibir cualquier número de nombres de fichero de tipo `char *`.

Descripción

Este programa procesa cada fichero en el orden indicado (o en su defecto, la **entrada estándar**) leyendo cada línea del mismo (considerando un máximo de 2048 caracteres por línea) y emite por su **salida estándar** dichas líneas pero en orden inverso, es decir, la primera línea del primer fichero deberá ser la última mostrada, y la primera mostrada la última del último fichero.

Valores de Terminación

EX_OK Terminación correcta.

EX_NOINPUT El fichero "FICHERO" no puede ser leído.

EX_OSERR No se pudo ubicar la memoria dinámica necesaria.

Ejemplos

```
$ ./bocabajo -h
bocabajo: Uso: bocabajo [ fichero... ]
bocabajo: Invierte el orden de las líneas de los ficheros (o de la entrada).
$ cat fich1
uno
dos
tres
cuatro
$ cat fich2
A
Be
Ce
De
E
$ ./bocabajo fich1 fich2
E
De
Ce
Be
A
cuatro
tres
dos
uno
$
```

Consulte el Manual

man 3 fprintf

Para emitir mensajes con formato.

man 3 fopen

Para abrir ficheros.

man 3 fgets

Para leer líneas de texto.

man 3 fputs

Para escribir líneas de texto.

man 3 fclose

Para cerrar los ficheros abiertos.

man 3 malloc

Para ubicar memoria dinámica.

man 3 strdup

Replicar una tira de caracteres en memoria dinámica.

Agradecimientos

Una gran parte del contenido de este cuaderno es fruto del trabajo de profesores del departamento de Arquitectura Arquitectura y Tecnología de Sistemas Informáticos. Su elaboración no habría sido posible sin la inestimable ayuda del profesor Francisco Rosales.