Fundamentos de los Sistemas Operativos

Práctica 2: lenguaje C

En esta práctica, trabajarás sobre las características del lenguaje C necesarias para el desarrollo de los programas de las prácticas posteriores de la asignatura.

1 Introducción

El lenguaje C nació junto con el sistema operativo UNIX hace unos cuarenta años y desde entonces se ha convertido en uno de los lenguajes de programación más utilizados en todo el mundo, sobre todo en el ámbito de la *programación de sistemas*: sistemas operativos, controladores de dispositivos, sistemas empotrados, etc. En la actualidad es uno de los tres lenguajes de programación más empleados a escala global[1]. De los diez lenguajes de programación más utilizados, siete son derivados del C y tienen una sintaxis muy similar.

La influencia del C ha sido y sigue siendo muy poderosa en el mundo del desarrollo de software. Por todos esos motivos, es conveniente que como futuro Ingeniero en Informática conozcas el lenguaje C. En la asignatura de FSO, las prácticas de programación se desarrollarán precisamente en este lenguaje.

Esta práctica está centrada en que adquieras un cierto nivel de competencia en el lenguaje de programación C. Estos son los objetivos de aprendizaje que te planteamos:

- Tener una visión general de las características del lenguaje C.
- Aprender cómo editar y compilar programas en C en entornos GNU/Linux.
- Entrenarse en la implementación de algoritmos en C.
- Adiestrarse con la gestión de memoria en C: punteros, matrices (*arrays*), cadenas de texto (*strings*) y memoria dinámica.

2 Requisitos previos

Partimos de que ya utilizas con fluidez algún otro lenguaje de programación, como Ada o **Java**, sobre todo este último, que se imparte en los primeros semestres del Grado en Ingeniería Informática de la EII.

[1] La web www.tiobe.com publica mensualmente un *ranking* del uso de lenguajes de programación.

3 Plan de actividades y orientaciones

Ejercicio entregable

Esta práctica te propone varios ejercicios de entrenamiento y también un **ejercicio entregable**. Este último tendrás que entregarlo a los profesores a través de la plataforma Moodle y será calificado dentro de la parte práctica de la asignatura. Lo mismo ocurrirá con las siguientes prácticas.

Al final de esta ficha te damos las instrucciones para realizar la entrega del ejercicio.

Equipo de trabajo

El ejercicio entregable de esta práctica se realizará y evaluará de forma individual.

Actividades propuestas

Actividades	*Objetivos / orientaciones*
Leer la documentación sobre lenguaje C	Tener una visión del lenguaje C, sobre todo desde la perspectiva de alguien que ya conoce Java.
Sesiones prácticas	Habrá dos sesiones prácticas en el laboratorio. El profesor realizará ejemplos y pequeñas tareas de edición y compilación de programas en C. También se harán demostraciones de las características más peculiares del C (punteros, cadenas, memoria dinámica).
Ejercicios de entrenamiento	Es muy recomendable que realices los ejercicios propuestos en esta ficha, para adiestrarte y confirmar que tienes un nivel de competencia suficiente en C.
Ejercicio entregable	Debes realizar la tarea propuesta en esta ficha y entregarla en Moodle. Te servirá para poner en acción al lenguaje C y sus peculiaridades en el manejo de memoria.

4 Libros y recursos en línea

En el Moodle hemos publicado varios materiales de elaboración propia sobre el lenguaje C en un nivel introductorio. Este es el material que recomendamos para que aprendas sobre C en el tiempo requerido para esta práctica.

También hemos seleccionado varios materiales (libros y recursos en línea de otras universidades) que te pueden servir como material complementario para resolver dudas y ampliar conocimientos durante todo el semestre.

Recursos en línea de la ULPGC

Este material lo verás también enlazado desde el Moodle.

- **Diapositivas de introducción al C.** Panorámica sobre las características básicas del C que se utilizarán en la asignatura. Disponible en http://sopa.dis.ulpgc.es/cpp/Lenguaje-C-intro-diapos.pdf
- **Introducción al lenguaje C.** El mismo contenido de las diapositivas, desarrollado en un manual. En http://sopa.dis.ulpgc.es/cpp/intro_c/

Libros

Entre los libros sobre C destacamos estas dos obras en español:

- ***C, Manual de referencia.*** *H. Schildt. Osborne/McGraw-Hill,2003.*
- ***Programación en C.*** *Byron S. Gottfried. Segunda edición en español de McGraw-Hill, 2005. ISBN 84-481-9846-8.*

Ambos libros son excelentes para el aprendizaje de C sin conocimientos previos. Contienen cientos de ejercicios y problemas. La Biblioteca de la Universidad tiene decenas de ejemplares, de las ediciones indicadas y de otras anteriores.

C para programadores de Java

Varias universidades disponen de materiales específicamente orientados para personas que ya conocen Java. De este material hemos seleccionado cuatro recursos que te vendrán muy bien si ya tienes experiencia como programador en Java y sólo necesitas conocer las peculiaridades de C. Échales un vistazo y escoge el que prefieras. Aquí te las presentamos ordenadas de más a menos recomendable desde nuestro punto de vista:

- Universidad de Cornell. Diapositivas «C for Java Programmers». Una panorámica breve sobre los aspectos de C que resultan más extraños a un programador en Java. 30 diapositivas. En inglés.
- Universidad de Murcia. Seminario «C para programadores Java». En español. Son tres documentos: Sesión 1 (cuestiones básicas); Sesión 2 (*structs*, punteros, *arrays*); Sesión 3 (funciones, memoria dinámica, E/S).
- Universidad de California en San Diego. C for Java Programmers: a Primer. Una guía completa en formato de manual. 94 páginas. En inglés.
- Universidad de Columbia. Diapositivas «C for Java Programmers» de la asignatura *Advanced Programming*. Una introducción más extensa en formato diapositivas. 126 diapositivas. En inglés.

5 Ejercicios de entrenamiento

Esta es una serie de ejercicios para que practiques los conceptos de lenguaje C que se manejarán en la asignatura. Puedes realizarlos de forma individual o en conjunto con otras personas, como prefieras. Si trabajas en equipo, tiene que ser de una forma activa: recuerda que *el aprendizaje es algo que ocurre en cada individuo*. No seas un simple «piojo pegado» a tu compañera/o.

Los ejercicios van en dificultad creciente. Puedes saltarte aquellos que consideres más sencillos. Lo recomendable es que realices al menos dos de los ejercicios y, de ellos, alguno de los ejercicios 5, 6 y 7.

1. Iniciación: edita y compila en Linux un programa en C que escriba la frase «hola, mundo». Cualquier persona que empiece a aprender C tiene la obligación moral de escribir este programa[2].
2. Escribe un programa que pida por teclado un número y lo muestre en base hexadecimal. Por ejemplo, si el usuario escribe «46», el programa mostrará «2E».
3. Escribe un programa que pida por teclado un número y lo muestre en binario. Por ejemplo, si el usuario escribe «46», el programa mostrará «101110».
4. Escribe un programa que imprima la tabla ASCII, mostrando por cada carácter su valor decimal, su valor en hexadecimal y su representación gráfica. Por ejemplo, para el carácter número 65, se deberá mostrar el «65», el «41» y «A».
5. Escribe un programa que imprima la tabla de los caracteres ASCII imprimibles (del número 32 al 127), con el siguiente formato:
    a. La tabla tendrá tres columnas.
    b. La tabla tendrá esta primera fila de encabezado:
    `DecHexChar | DecHexChar | DecHexChar`
    c. Los caracteres aparecerán ordenados por columnas y mostrarán su valor en decimal, hexadecimal y como letra. Los dos primeros valores estarán alineados a la derecha y la letra estará alineada a la izquierda.
    Por ejemplo, la primera fila debería mostrar esto:
    `32 20 | 64 40 @ | 96 60 ``
    d. En esta dirección web hay un ejemplo de tabla con el formato aquí especificado:
    http://www.cdrummond.qc.ca/cegep/informat/professeurs/alain/images/ASCII1.GIF
6. Escribe una función que cuente el número de vocales que contiene una cadena de texto, que se le pasará como parámetro a la función. Haz un programa de prueba que verifique que la implementación es correcta (procura contemplar un número suficiente de casos de prueba). NOTA: no tengas en cuenta las letras acentuadas.

[2] *Hello, world* es una frase mítica en la Ingeniería Informática. Su origen está precisamente en el lenguaje C: http://en.wikipedia.org/wiki/Hello_world_program

7. Escribe un programa que lea de teclado una línea de texto y a continuación imprima las palabras del texto (las palabras están separadas por espacios). Cada palabra aparecerá en una línea diferente.
   Por ejemplo, si tecleamos:
   ```
   una línea cualquiera
   ```
   la salida debe ser:
   ```
   una
   línea
   cualquiera
   ```

6 Ejercicio para trabajar en el laboratorio

En las sesiones del laboratorio trabajaremos con un caso práctico en el que tendremos ocasión de utilizar varios elementos del lenguaje C, como punteros, vectores y funciones. También trabajaremos con la construcción de una *biblioteca (library*), mediante ficheros fuentes de compilación separada.

A continuación, te damos el esquema general de lo que abordaremos en el laboratorio, para que te sirva de material de apoyo.

Crearemos un vector global de tamaño prefijado, que representará un conjunto de asientos libres y ocupados en una sala de cine. Un valor **-1** en el elemento *i* del vector indicará que el asiento *i* está libre. Si el valor es positivo, digamos *N*, significa que el asiento está ocupado por una persona con DNI número *N* (sin la letra del DNI). El vector estará inicializado con todos sus asientos libres.

Sobre ese vector escribiremos un conjunto de funciones en C para hacerle manipulaciones básicas:

- `Find` – para buscarle un asiento libre a una persona (se pasa como argumento el DNI de la persona). Si se encuentra un asiento libre, se devuelve el número de asiento y si no, se devuelve un -1.
- `Clear` – para marcar un asiento como libre. Le pasamos como argumento el número de asiento. Devuelve el DNI de la persona que ocupaba el asiento, o un -1 si el asiento ya estaba libre o hubo algún otro error.
- `Test` – para saber el estado de un asiento, cuyo número se pasa como argumento. Devuelve el DNI de quien ocupa el asiento, un 0 si está libre y un -1 si hay un error.
- `NumClear` – nos devuelve el número de asientos libres.
- `NumSet` – nos devuelve el número de asientos ocupados.

Para comprobar que esta interfaz funciona, construiremos unas funciones de prueba (test), que invocaremos desde el programa principal:

- Una operación `sentarse(`*`dni`*`)`que le encuentre un asiento a una persona y le imprima por pantalla un mensaje explicativo del resultado de la operación.
- Una operación `levantarse(`*`dni`*`)` que libere el asiento ocupado por una persona, y que imprima por pantalla los posibles errores.
- Una operación `reserva_multiple(int` *`npersonas`*`, int*` *`lista_dni`*`)` que tome un vector con un conjunto de DNI y reserve asientos para todos ellos.

La operación debe ser de «todo o nada»: si no hay asientos libres para todas las personas, no se reservará ningún asiento.

Para organizar el código en módulos, la interfaz y la implementación del vector se escribirán en sendos ficheros **asientos.h** y **asientos.c**. El programa principal deberá ir en un fichero fuente diferente, por ejemplo **test_asientos.c**.

Una vez realizado todo o parte del ejercicio, se puede modificar la biblioteca de forma que el número de asientos se pueda definir de forma dinámica, en tiempo de ejecución. Esto se consigue creando el vector mediante la función **malloc()**.

7 Ejercicio entregable: biblioteca «mistring»

Este ejercicio te servirá para poner a prueba tus conocimientos sobre lenguaje C, en especial los punteros y la memoria dinámica. Tendrás que implementar tu propia versión de algunas de las funciones estándares de **<string.h>**, de tratamiento de cadenas de texto. Verás que estas funciones se pueden implementar con punteros de C con un código bastante compacto y eficiente, comparado con el que necesitaríamos con otros lenguajes de alto nivel como Java.

Tu colección de funciones debe estar construida como una biblioteca de C: tendrás que construir sendos ficheros **mistring.h** y **mistring.c** con la interfaz y la implementación de tus funciones. Para verificar que la biblioteca está bien implementada, también tendrás que desarrollar unos programas de prueba.

Especificación de las funciones

Función	Comportamiento
`Int mi_strlen (char* str);`	Devuelve un entero con la longitud de **str** (número de caracteres hasta llegar al NUL).
`char* mi_strcpy (char* s1, char* s2);`	Copia los caracteres de **s2** en **s1** y añade un NUL al final. Devuelve la dirección de **s1**.
`char* mi_strcat (char* s1, char* s2);`	Añade los caracteres de **s2** al final de **s1**. Es decir, concatena **s2** a **s1**. Devuelve la dirección de **s1**.
`char* mi_strdup (char* str);`	Crea un duplicado de **str** mediante memoria dinámica. El contenido del duplicado será idéntico al de **str**. Devuelve la dirección del duplicado.
`Int mi_strequals (char* s1, char* s2);`	Compara las cadenas s1 y s2. Si son idénticas, la función devuelve un 1. Si son diferentes, la función devuelve un 0.

La implementación de todas estas funciones debe hacerse sin recurrir a funciones externas, excepto la función **malloc()**, que será necesaria en algún caso.

Programas de prueba

Para probar y demostrar el funcionamiento de tu biblioteca, tendrás que implementar uno o varios programas de prueba en los que se vean las distintas funciones en acción. Dejamos a tu criterio qué pruebas hacer.

En el Moodle te proporcionamos un fichero fuente **test_mistring.c** que contiene algunas pruebas básicas de las diferentes funciones de la nueva biblioteca. La rutina `ejecuta_tests()` debería ejecutarse sin devolver ningún mensaje de error. Puedes ampliar el código que te entregamos con más pruebas de tu propia cosecha. Recuerda que debes ser implacable con las pruebas. El profesor lo será.

Observaciones y consejos

Si tienes dudas sobre cómo deben comportarse estas funciones, puedes consultar las especificaciones de las versiones originales de las funciones de la biblioteca **<string.h>**. Si estás en Linux, puedes consultar directamente el manual en línea: ej. `man 3 strlen`

Si tu equipo no tiene el manual en línea, lo podrás encontrar en Internet (p.ej. busca «man strlen» en el navegador). También puedes ver ayuda en español sobre estas funciones en la web http://c.conclase.net (aquí tienes el ejemplo de la función strcat).

Para implementar **mi_strdup()** tendrás que conocer cómo reservar memoria dinámica. Esto lo puedes hacer con la función **malloc()**.

Te volvemos a insistir en que tu programa debe contener un número de tests suficiente para demostrar que has hecho bien el trabajo. Nosotros te proporcionamos unas cuantas pruebas, pero debes ser tú quien amplíe los tests para cubrir una variedad grande de casos.

Forma de entrega del trabajo

El contenido de la entrega debe ser:

- **Código fuente.** Todos los fuentes en C que sean necesarios para compilar y probar el trabajo realizado: cabecera y cuerpo de tu biblioteca, programas de test, etc.
- **Ficha de entrega.** Un pequeño resumen del trabajo realizado. Usa la plantilla que está publicada en el Moodle y entrega el documento en formato PDF.

Todo esto debe entregarse empaquetado en un único fichero comprimido. Los únicos formatos que aceptamos son ZIP, TAR y TAR.GZ. **No entregues en formato RAR** (es un formato propietario y puede ser que no podamos abrirlo).

La entrega se realizará en la tarea Moodle que verás en la sección dedicada a la Práctica 2.