Fundamentos de los Sistemas Operativos

Práctica 1: 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 cincuenta años. 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, cinco 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 futura o 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 nivel básico 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, vectores (*arrays*), cadenas de texto (*strings*) y memoria dinámica.

2 Requisitos previos

Partimos de que ya utilizas con fluidez algún otro lenguaje de programación, como Python 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.

El código que escribirás en este ejercicio servirá como punto de partida para las siguientes prácticas. En los sucesivos trabajos tendrás que ir añadiendo funcionalidades al código que has ido elaborando anteriormente.

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

Equipo de trabajo

El ejercicio entregable de esta práctica se realizará y evaluará de forma grupal preferentemente (grupos de 2), aunque también se puede llevar a cabo de forma individual. Sin embargo, te invitamos a que lo lleves a cabo en grupo, pues queremos fomentar el trabajo en equipo, algo bastante habitual en el mundo laboral en nuestro ámbito.

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 Programming:*** *A comprehensive look at the C programming language and its features (https://en.wikibooks.org/wiki/C_Programming).*
- ***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.*

Estos libros son excelentes para el aprendizaje de C sin conocimientos previos. Contienen cientos de ejercicios y problemas. El primero está accesible *online* y es gratuito y los dos últimos están disponibles en la Biblioteca de la Universidad y hay decenas de ejemplares, de las ediciones indicadas y de otras anteriores.

Algunos tutoriales gratuitos

A continuación te hacemos una relación de algunos recursos online que puedes emplear para iniciarte en C:

Learn C Programming: práctico tutorial de C para aprender sintaxis básica, variables, constantes, operadores, funciones, estructuras, cadenas.

Learn C: práctico tutorial en inglés que no necesita ninguna experiencia previa para poder seguirlo.

Curso de C Básico Gratuito: curso en español y mediante vídeos de YouTube (29 prácticos vídeos de un total de 2 horas y 52 minutos de duración).

W3schools C Tutorial: completo tutorial de C.

C Programming Language: práctico tutorial para seguir en línea y aprender C.

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 Columbia. Diapositivas «C for Java Programmers» de la asignatura *Advanced Programming*. Una introducción más extensa en formato de diapositivas PowerPoint. 126 diapositivas. En inglés.

5 Ejercicios de entrenamiento

Te ofrecemos 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 grupo, hazlo de 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. Especialmente, te sugerimos que al menos hagas 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:

[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

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:
https://elcodigoascii.com.ar/codigo-americano-estandar-intercambio-informacion/codigo-ascii.png

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.
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: biblioteca «mistring»

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.

Te proponemos que implementes 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 en el ejercicio entregable de esta actividad.

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 instalación de Linux 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()**.

7 Ejercicio entregable: sistema de gestión de reservas de los asientos de una sala de teatro.

En esta primera actividad entregable se propone la implementación de una estructura de datos y una API básica para la gestión de un sistema de reservas de los asientos de una sala de teatro. La sala tendrá una capacidad máxima determinada por la constante CAPACIDAD_MÁXIMA. Además, los asientos de la sala de teatro tendrán un identificador único (*id_asiento*) en el rango [1,CAPACIDAD_MÁXIMA]. Cada asiento reservado guardará un identificador de persona (*id_persona*) que será un entero positivo mayor que cero. Podrán existir múltiples reservas para un mismo identificador de persona. A continuación te mostramos la lista de funciones que debe ofrecer la API. Tendrás que implementar una estructura de datos y código que resuelvan la totalidad de esta API.

Función	Comportamiento
`int reserva_asiento (int id_persona);`	Reserva un asiento libre a una persona (se pasa como argumento un identificador numérico de la persona que realiza la reserva). Si se encuentra un asiento libre, se devuelve el número de asiento y si no, se devuelve un -1.
`int libera_asiento (int id_asiento);`	Marca un asiento como libre. Le pasamos como argumento el número de asiento. Devuelve el ID de la persona que ocupaba el asiento, o un -1 si el asiento ya estaba libre o hubo algún otro error.
`int estado_asiento(int id_asiento);`	Informa del estado de un asiento, cuyo número se pasa como argumento. Devuelve el identificador de quién ocupa el asiento, un 0 si está libre y un -1 si hay un error.
`int asientos_libres ();`	Devuelve el número de asientos libres.
`int asientos_ocupados ();`	Devuelve el número de asientos ocupados.
`int capacidad_sala ();`	Devuelve la capacidad de la sala.
`int crea_sala (int capacidad);`	Crea la estructura de datos que da soporte a la sala y establece todos los asientos como libres. Le pasamos como argumento la capacidad que debe soportar la sala. Devuelve la capacidad de la sala si todo ha ido bien o un -1 si hay algún error.
`int elimina_sala();`	Libera la memoria que da soporte a la estructura de datos que representa la sala. Devuelve un 0 si todo ha ido bien o -1 si hay algún error.

Todas estas funciones devuelven un valor -1 si se ha producido un error al intentar ejecutarse.

Antes de comenzar con el desarrollo te recomendamos que pienses en la estructura de datos que vas a plantear para representar la sala y sus asientos y que dará soporte al sistema de gestión de reservas global de la sala (un vector, un mapa de bits, etc…). Por ejemplo, si decidiéramos emplear un vector para representar la sala, crearemos un vector cuyo tamaño coincida con la capacidad de la sala, que representará un conjunto de asientos libres y ocupados de la sala de teatro. Un valor **-1** en el elemento *i* del vector podría indicar que el asiento *i* está libre. Si el valor es positivo, digamos *N*, significa que el asiento está ocupado por una persona con identificador número *N*. El vector estará inicializado con todos sus asientos libres. Ten en cuenta que la capacidad de la sala no puede estar predefinida, es decir, se trata de un parámetro cuyo valor queda determinado en tiempo de ejecución. Por tanto, tendrás que emplear memoria dinámica (**malloc()**) para la estructura de datos que decidas elegir para representar la sala.

Por otro lado, el código que implementes en la API deberá impedir también las siguientes situaciones: crear una nueva sala sin haber eliminado la anterior; o eliminar una sala sin haberla creado previamente.

La API que te proponemos debe estar construida como una biblioteca de C: tendrás que construir sendos ficheros **sala.h** y **sala.c** con la interfaz y la implementación de tus funciones.

Programas de prueba

Para probar y demostrar el funcionamiento de la API, 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_sala.c** que contiene algunas pruebas básicas de las diferentes funciones de la API. La rutina `ejecuta_tests()` debería ejecutarse sin devolver ningún mensaje de error. Debes 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. Aquí tienes algunas sugerencias de funciones de test que puedes añadir para probar la API que has desarrollado:

- Una operación `estado_sala()` que muestre el estado de todos y cada uno de los asientos de la sala, así como una información resumen (ej. Aforo y ocupación actual).
- Una operación `sentarse(`*`id`*`)` que le encuentre un asiento a una persona y le imprima por pantalla un mensaje explicativo del resultado de la operación.
- Una operación `levantarse(`*`id`*`)` que libere el asiento ocupado por una persona, y que imprima por pantalla los posibles errores.
- Una operación `reserva_multiple(int` *`npersonas`*`, int*` *`lista_id`*`)` que tome un vector con un conjunto de identificadores 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.

Te volvemos a recomendar que tu programa contenga 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 1.

8 Criterios de evaluación

Esta práctica recibirá una calificación en la escala «superada / no superada».

Superada

Para que una entrega sea calificada como *superada* deberá cumplir **todos** estos requisitos:

- Las funciones de **sala.c** funcionan de acuerdo con las especificaciones.
- El código está correctamente separado en módulos (.h, .c, etc.).
- El código supera los tests propuestos por el equipo docente.
- Se ha implementado al menos una prueba para cada función de la biblioteca.
- Se ha entregado una ficha/memoria entendible y conforme con la plantilla.