Sistemas Operativos 2020-2, Práctica 01: Introducción a C

Luis Enrique Serrano Gutiérrez (luis@ciencias.unam.mx)
Juan Alberto Camacho Bolaños (juancamacho@ciencias.unam.mx)
Ricchy Alaín Pérez Chevanier (alain.chevanier@ciencias.unam.mx)

Fecha de Entrega: 17 de Febrero de 2020

1 Objetivo

El objetivode esta práctica es introducir al estudiante a la resolución de problemas utilizando el lenguaje de programación C[1].

Es importante en esta parte del curso preguntar todas las dudas referentes a C para que en la siguiente parte del curso podamos realizar las prácticas y proyectos con mayor fluidez.

2 Introducción

2.1 Docker como herramienta de trabajo (opcional)

Para esta práctica utilizaremos la definición de una imagen de docker que contiene las herramientras necesarioas para compilar y ejecutar el código en el que trabajarás.

Para tomar ventaja de esta imagen primero necesitarás instalar Docker sobre un sistema operativo con una terminal que pueda ejecutar scripts escritos con bash. En estos enlaces puedes encontrar isntrucciones de cómo instalar `Docker` en tu sistema operativo.

- Linux: https://docs.docker.com/install/
- MacOS: https://docs.docker.com/docker-for-mac/install/
- Windows: https://docs.docker.com/docker-for-windows/

Una vez que tengas instalado Docker, posiciona la terminal en el direcrtorio de estra práctica y ejecuta el comando `./docker-exec docker-build`[2], el cual descargará y construirá localmente una imagen de ubuntu con las herramientras necesarias para trabajar con esta práctica.

A continuación una lista de todos los comandos disponibles en el script `docker-exec`:

- `./docker-exec compile` compila el archivo *practica01.c* utilizando el contexto de ejecución del contenedor de docker.
- `./docker-exec run` una vez que el archivo *practica01.c* compila sin errores puedes ejecutar ele ejecutable generado con este comando.
- `./docker-exec clean` elemina el archivo ejecutable generado con la acción `compile`
- `./docker-exec "any other command"` ejecuta la instrucción *any other command* dentro del contenedor de docker, por ejemplo puedes ejecutar `./docker-exec "gcc --version"` para ver qué version de *gcc* está instalada dentro del contenedor.

[1] Diversos aspectos de C serán expuestos durante las ayudantías de salón y laboratorio al comienzo del curso, también recomendamos algunas referencias para aprender un poco más sobre dicho lenguaje.

[2] Este script utiliza bash por lo que solo podrá ser ejecutado en terminales con este contexto de ejecución, es posible que no funcione en windows por esta razón, si utilizas windows revisa los comandos que se encuentran dentro de este script, deberian de poderse adaptar facilmente para funcionar en tu sistema operativo.

2.2 Compilador de C (GCC)

Para este curso utilizaremos gcc como comopilador de C. En los sistemas operativos tipo Unix, gcc es un programa que puede ser invocado desde una terminal. Así que asumiendo que tienes una terminal abierta, ejecuta la siguiente instrucción:

$ gcc - -version[3]

Lo anterior desplegará un resumen de la versión de gcc que está instalada en tu sistema. Usualmente los archivos que contienen código fuente de C tienen como extención .c, por ejemplo un archivo que contiene un programa en C podría llamarse hola_mundo.c. Para compilar el archivo hola_mundo.c, asumiendo que el directorio actual de trabajo lo contiene , tenemos que ejecutar un comando como el siguiente dentro de una terminal:

$ gcc -Wall -Werror -o mi_programa hola_mundo.c

Si el archivo hola_mundo.c no tiene errores de sintaxis ni de semántica[4], el comando anterior crea un archivo ejecutable con nombre mi_programa. La linea de arriba utiliza algunas banderas de compilación de gcc, a continuación una breve descripción de cada una de ellas:

- -Wall : Esta bandera indica a gcc a que nos proporcione advertencias que involucren posibles errores semáticos dentro del programa, esta bandera no recibe argumentos, es muy recomendable utilizar para evitar algunos errores de ejecución.
- -Werror : Indica que gcc que trate las adventencias como errores, i.e. no genera el ejecutable hasta que los eliminemos.
- -o : Indica a gcc el nombre que queremos que tenga el ejecutable generado como consecuencia del proceso de compilación.

2.3 Referencias para aprender C

En internet existen una gran variedad de referencias y tutoriales para aprender C y para consultar su API, así que eres libre de consultar los materiales que tú desees, en particular puedes descargar un buen libro de la siguiente dirección:

http://cslibrary.stanford.edu/101/

Para consultar el API pueden consultar el siguiente sitio:
http://www.cplusplus.com/reference/

Cabe mencionar que ambas fuentes para algunos componentes están enfocados a sistemas operativos unix-like.

3 Ejercicios de Programación

En un mismo archivo debes de escribir un programa que resuelva los siguientes problemas:

3.1 Pruebas Unitarias

Escribe pruebas unitarias para verificar el correcto funcionamiento de cada una de tus soluciones (La solución a cada problema vale lo mismo que las pruebas unitarias que lo acompañan). Es muy importante conocer el concepto de software testing, pues durante todo el curso contarán con pruebas para evaluar sus soluciones. Para cada función puedes escribir una función adicional que contenga sus pruebas unitarias asociadas. Hay un pequeño ejemplo de cómo hacer *assertions* en el archivo `práctica01.c`.

[3] Puedes ejecutar este comando en el contenedor de docker utilizando el script docker-exec
[4] En realidad sólo destecta algunos errores semánticos

3.2 $A[i] = i$

Problema:

Dado un arreglo de enteros distintos ordenado ascendentemente, escribe una función que decida si existe un índice i tal que $A[i] = i$. Tu solución tiene que ser $o(n)$, es decir en ningún caso puedes dar un algoritmo que se tarde tanto como tiempo lineal.

Notas:

- El arreglo puede contener valores positivos y negativos.
- No pierdas de vista que el arreglo está ordenado ascendentemente y que los elementos del arreglo son todos distintos.

3.3 Cadenas

Problema:

Averigua cómo representar strings en C (no existe un tipo de dato en C que los represente directamente), y escribe una función que tome un string y que invierta el orden de las palabras que contiene, por ejemplo si el string es "esta es una cadena no muy larga" entonces la función debe de crear el string "larga muy no cadena una es esta". ¿En términos de complejidad computacional asintótica, qué tiempo toma tu solución para resolver el problema (responde esta pregunta como parte de tu reporte)?

Notas:

- Realiza una copia de la cadena utilizando la función malloc.[5]
- Puedes utilizar cualquier función que se encuentre en `string.h` como subrutina de esta función.
- Para realizar pruebas sobre esta función considera todos los posible casos que involucran espacios en blanco que separan a cada una de las palabras de la sentencia que forma la cadena objetivo.
- Considera cualquier caractér que no sea una letra como si lo fuera salvo los separadores de palabras.
- Es fácil obtener una solución que resuelva el problema en tiempo lineal.

3.4 Árboles Binarios de Búsqueda

Problema:

Escribe una función que dado un árbol binario decida si este representa un árbol binario de búsqueda. Puede asumir por simplicidad que el árbol contiene enteros en cada nodo, además de que no tiene repeticiones.

Notas:

- Un árbol binario es de búsqueda si:
  - El subárbol izquierdo de cualquier nodo contiene solamente nodos con valor menor que el del nodo.
  - El subárbol derecho de cualquier nodo contiene solamente nodos con valor mayor que el del nodo.
  - Los subárboles derechos e izquierdos de cualquier nodo también deben de ser un árbol binario de búsqueda.
- Un árbo vacío está representado por una referecia nula.
- Dado que escribir árboles para realizar las pruebas es suficientemente tedioso, entonces sólo prueba con unos cuantos casos significativos.

[5] La función malloc reserva memoria en el heap.

3.5 Sobre struct node (1 extra)

Problema:

La estructura `struct node` puede ser utilizada para representar listas doblemente ligadas, árboles binarios o incluso ninguno de los dos. Dado un nodo aleatorio decide si este forma una lista doblemente ligada, un árbol, o ninguno de los dos.

4 Notas

Con esta práctica recibirás un archivo que contiene la definición de la estructura para representar árboles binarios, dicha definición tiene que ser utilizada como parte de tu solución; además en el mismo archivo estarán incluidas las firmas de las funciones que tienen que implementar.

Es recomendable que antes de hacer la práctica averigües cómo funcionan la función malloc y free. También invenstiga cómo se manejan los valores true y false en C dado que este no tiene el tipo de dato boolean per se.

5 Entrega

Sigue los lineamientos de entrega para esta práctica, y asegurate de que tu entregable corra propiamente las pruebas unitarias que realizaste como parte de tu solución.