75.40 Algortimos y Programación I Curso 4
lab0: El Compilador

Dr. Mariano Méndez

Facultad De Ingeniería. Universidad de Buenos Aires

23 de agosto de 2022

1. Introducción

2. El Compilador

El objetivo de este lab es tener un primer contacto con la programación, un lenguaje de programación y herramientas.

2.1. El Compilador

Se debe verificar que el compilador de C esté instalado en la computadora de trabajo. Para realizar la verificación de la instalación del compilador de C se debe invocar al mismo. Para una verificación adecuada se debe ejecutar en una terminal de linux el siguiente comando:

```
$ gcc -v
```

Se pide:

- Mostrar cuál es la salida producida tras la ejecución del comando.
- Determinar si el compilador de C está o no instalado, a partir de la salida anterior.
- Si el mismo no está instalado ejecutar el comando **sudo apt-get install gcc** en una terminal.

2.2. El Primer Programa

Una vez que se ha verificado la existencia del compilador en la instalación del sistema operativo, es necesario implementar, compilar y ejecutar el primer programa en el lenguaje de programación C. El programa más sencillo que puede ser implementado es aquel que imprima un mensaje por la consola. Para ello se debe utilizar cualquier editor de texto, en este caso el programa se llamará hola.c y su contenido será:

```c
#include<stdio.h>

int main() {

    printf("Hello World!");

    return 0;
}
```

Una vez que el texto es copiado se almacena en el sistema de archivos utilizando la opción guardar como con el nombre "hola.c". Para verificar la existencia del archivo se debe ejecutar el comando `ls`

2.3. Usando el Compilador

Como se a visto el compilador realiza el proceso de traducción de un archivo en **código fuente** (nuestro hola.c) a un archivo capaz de ser ejecutado por una computadora. Para ello el mismo realiza una serie de operaciones sobre el código fuente.

- ¿Cuántas son las estas operaciones?

- ¿Cómo se llaman?
- ¿Qué sucede en cada una de ellas?

A continuación se pide que únicamente se compile el programa hola.c, mediante la ejecución de la siguiente linea de comando:

```
$ gcc -c hola.c
```

Listar el contenido del directorio con el comando `ls` y contestar las preguntas:

- ¿Mostrar el resultado de la ejecución del comando?
- Intentar abrir con un editor de texto el archivo hola.o
- ¿Qué sucede?
- ¿Cómo se denomina a un archivo que termina en .o?

2.4. Generar el Primer Ejecutable

Ahora que ya se ha compilado el programa se procederá a generar el primer programa ejecutable. Para ello se debe ejecutar la siguiente línea de comando:

```
$ gcc hola.c
```

Listar el contenido del directorio con el comando `ls` y contestar las preguntas:

- ¿Mostrar el resultado de la ejecución del comando?
- ¿Que hay de raro en el directorio que antes no estaba?
- ¿Qué sucede si ejecuto `./a.out`?
- ¿que sucedió en la ejecución?

2.5. Generar el Primer Ejecutable

El compilador de C cuando se omite el nombre que se le quiere poner al programa ejecutable, este le asigna un nombre que es estándar (es decir que sucede además en todos los compiladores de C), el nombre es **a.out**. No muchos saben que a.out viene de la abreviación de **assembler.output**. Para que el compilador asigne un nombre específico al programa ejecutable se debe usar una opción de compilación , dicha opción es:

```
$ gcc hola.c -o hola
```

Ejecutar el comando y listar el contenido del directorio con el comando `ls` y contestar las preguntas:

- ¿Mostrar el resultado de la ejecución del comando?
- ¿como se llama el programa ejecutable?
- ¿Qué sucede si ejecuto `./hola`?
- ¿que sucedió en la ejecución?

3. Compilación más Avanzada

De la misma forma que se puede especificar el nombre del programa ejecutable a generar, el compilador acepta una enorme cantidad de opciones de compilación y link-edición. A continuación se mostrarán algunas.

3.1. Todas las Advertencias

Con el editor de texto con el cual se esté trabajando se volverá a editar el hola.c:

```c
#include<stdio.h>

int main() {

    int i;
    printf("Hello World!");

    return 0;
}
```

Se guardarán las modificaciones realizadas. En este caso la declaración de una variable entera llamada i. A continuación se ejecutará el comando:

```
$ gcc -Wall hola.c -o hola
```

- ¿Mostrar el resultado de la ejecución del comando?
- ¿Qué sucedió?¿ Se generó el programa ejecutable?
- ¿que sucedió en la ejecución?

3.2. Convertir Advertencias en Errores

Ejecutar el comando

```
$ gcc -Wall -Werror hola.c -o hola
```

contestar las preguntas:

- ¿Mostrar el resultado de la ejecución del comando?
- ¿Qué sucedió?

3.3. Salidas Intermedias

El proceso de compilación tiene diversos pasos y etapas. Cada una de estas tiene ciertas salidas intermedias.

3.3.1. Código Pre-procesado

Ejecutar el siguiente comando:

```
$ gcc -E hola.c > hola.i
```

Listar el contenido del directorio con el comando `ls` y contestar las preguntas:

- ¿Mostrar el resultado de la ejecución del comando?
- ¿qué hay en hola.i?
- ¿Qué será?

3.3.2. Código Assembly

```
$ gcc -S hola.c > hola.S
```

Listar el contenido del directorio con el comando `ls` y contestar las preguntas:

- ¿Mostrar el resultado de la ejecución del comando?
- ¿qué hay en hola.C?
- ¿Qué será?

3.4. Compilar y Linkeditar

Como se dijo anteriormente GCC compila y link-edita en un solo paso de forma automática. Se ha visto que se puede solicitar que sólo compile, pero también se puede solicitar que se link-edite a mano. Para ello se utiliza la opción -l:

```
$ gcc  -Wall hola.c -o hola -lc
```

Listar el contenido del directorio con el comando `ls` y contestar las preguntas:

- ¿Mostrar el resultado de la ejecución del comando?
- ¿Qué sucedió?

3.5. Ver lo que está pasando

Ejecutar el comando

```
$ gcc -Wall -v hola.c -o hola -lc
```

Listar el contenido del directorio con el comando `ls` y contestar las preguntas:

- ¿Mostrar el resultado de la ejecución del comando?
- ¿Qué sucedió?