



Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Algoritmos y Estructuras de Datos
Curso K2055
Ing. Pablo D. Mendez

Trabajo práctico final

Grupo SinGrupo

Fecha de entrega: 28/11/2025

Legajo	Nombre y Apellido	Correo Institucional	Github
160.664-5	Alain Degoumois	adegoumoispasserini@frba.utn.edu.ar	https://github.com/adego25
215.087-6	Carracedo Ignacio Lautaro	isantillan@frba.utn.edu.ar	https://github.com/nachito-cc
203.665-4	Mauro Ghia Feroldi	mghiaferoldi@frba.utn.edu.ar	https://github.com/MauroGhia1

Repo del grupo:

https://github.com/adego25/TP_SySL_Grupo_Olvida2

Parte Teórica

1. Fases de compilación de un programa C y Tabla de Símbolos

El proceso de compilación en C tiene 4 fases principales:

Preprocesamiento: Expande macros, incluye archivos de cabecera (`#include`) y elimina comentarios.

Compilación: Traduce el código C preprocesado a código ensamblador. Aquí interactúa fuertemente la Tabla de Símbolos, registrando identificadores, tipos y alcances para validar la semántica.

Ensamblado (Assembly): Traduce el código ensamblador a código objeto (lenguaje máquina, archivos `.o` ó `.obj`).

Enlazado (Linking): Combina los archivos objeto y bibliotecas para generar el ejecutable final.

Interacción de la Tabla de Símbolos: Es crucial durante el análisis semántico (fase 2) para verificar tipos y declaraciones, y en la generación de código para asignar direcciones de memoria.

2. Concepto de "Espacio de Nombres" (Namespaces) en C

En C estándar no existe la palabra clave `namespace` como en C++, pero el concepto se maneja a través de alcances (scopes) y categorías de identificadores.

Concepto: Es un contexto donde un identificador es único.

Ejemplos:

Las etiquetas de estructuras (`struct Tag`) están en un espacio de nombre separado de las variables (`int Tag`). Pueden llamarse igual sin chocar.

Variables locales dentro de funciones distintas (scope local).

Rol de la Tabla de Símbolos: La tabla distingue estos identificadores agregando atributos de "categoría" o "scope" a cada entrada. Así sabe que `Tag` (variable) y `Tag` (struct) son entidades distintas.

3. Makefile de C

El makefile es un script utilizado por la herramienta make para automatizar la compilación. Define grafos de dependencias para compilar solo lo que ha sido modificado.

Parámetros principales:

Objetivo (Target): El archivo a generar (ej: programa.exe).

Dependencias: Archivos necesarios para crear el objetivo (ej: main.o, utils.o).

Comandos: Instrucciones de terminal para realizar la tarea (ej: gcc ...).

Ejemplo de uso:

Makefile

Makefile simple

programa: main.o funciones.o

gcc -o programa main.o funciones.o

main.o: main.c

gcc -c main.c

clean:

rm *.o programa

4. Pasos de Make y el Linker

Pasos de Make:

Lee el archivo Makefile.

Compara la fecha de modificación de los "objetivos" contra sus "dependencias".

Si una dependencia es más nueva que el objetivo, ejecuta el comando asociado (recompila).

Repite esto recursivamente hasta llegar al ejecutable final.

Programas Objeto y Linker:

Los programas objeto (.o) se confeccionan en la fase de Ensamblado, después de compilar cada archivo fuente .c por separado.

Función del Linker: Su función es tomar todos los archivos objeto (.o) y las bibliotecas del sistema, resolver las referencias a funciones externas (ej: unir una llamada a printf con su código real en la biblioteca) y generar un único archivo ejecutable.