Compiladores 2026-1
Práctica 0: Sistema de procesamiento de Lenguaje

Adrián Lima García

Objetivo

Comprender el funcionamiento de los diferentes programas que intervienen en el proceso de traducción de un programa fuente a un ejecutable.

Desarrollo y Resultados

1. Preprocesador

Comando:

```
cpp programa.c programa.i
```

- El preprocesador expande macros, incluye el contenido de archivos de cabecera y elimina comentarios.
- **Similitudes:** el flujo principal del programa se conserva.
- **Macros y comentarios:** los comentarios desaparecen y las macros se expanden.
- **Comparación con `stdio.h`:** ambos tienen declaraciones y prototipos de funciones.
- **Etapa:** Preprocesamiento.

2. Compilación a Ensamblador

```
gcc -Wall -S programa.i
```

- **-Wall:** muestra todas las advertencias.
- **-S:** indica que solo se genere código ensamblador.
- **Salida:** archivo con extensión `.s`.
- **Etapa:** Compilación.

3. Ensamblador

```
as programa.s -o programa.o
```

- **Hipótesis:** el archivo `.o` debe contener código objeto relocalizable.
- **Contenido:** un binario ELF, no legible directamente.
- **Programa invocado:** el ensamblador GNU (`as`).
- **Etapa:** Traducción a código objeto.

4. Enlazador

Archivos a localizar en el sistema:

```
/lib/ld-linux-x86-64.so.2
/usr/lib/x86_64-linux-gnu/Scrt1.o
/usr/lib/x86_64-linux-gnu/crti.o
/usr/lib/gcc/x86_64-linux-gnu/<versión>/crtbeginS.o
/usr/lib/gcc/x86_64-linux-gnu/<versión>/crtendS.o
/usr/lib/x86_64-linux-gnu/crtn.o
```

Comando:

```
ld -o ejecutable -dynamic-linker /lib/ld-linux-x86-64.so.2 \
/usr/lib/Scrt1.o /usr/lib/crti.o programa.o -lc /usr/lib/crtn.o
```

- Si falla, se puede usar:

```
gcc programa.o -o ejecutable
```

- **Resultado:** se genera un ejecutable que al correr muestra:

```
Hola Mundo !
Resultado : 28.274401
```

5. Macro #define PI

Al quitar el comentario de la línea `#define PI 3.1415926535897` y recompilar:

```
Resultado : 28.274334
```

La diferencia es mínima, pero más exacta.

6. Segundo Programa con Directivas

Ejemplo de `segundo.c`:

```c
#include <stdio.h>      // E/S est ndar
#include <string.h>     // Funciones para cadenas
#define MENSAJE "Compiladores\n"
#undef area             // Elimina definici n previa
#ifdef MENSAJE
    #define EXTRA "Directivas funcionando!\n"
#endif

int main() {
    printf(MENSAJE);
    printf(EXTRA);
    printf("Longitud del MENSAJE: %lu\n", strlen(MENSAJE));
    return 0;
}
```

Directivas empleadas:

1. `#include <string.h>` – para cadenas.
2. `#define` – define macros.
3. `#undef` – elimina macros.
4. `#ifdef / #endif` – código condicional.

—

Conclusiones

- El proceso de traducción en C incluye: preprocesamiento, compilación, ensamblado y enlace.
- Cada archivo intermedio (`.i`, `.s`, `.o`) representa un nivel de abstracción distinto.
- Definir la macro `PI` mejora la precisión del resultado numérico.
- Las directivas del preprocesador permiten modularidad y flexibilidad en el código.