Instituto Tecnológico de Costa Rica

Compiladores e Intérpretes

Apuntes de la clase del día viernes 3 de marzo del 2017

Apuntador:

Luis Daniel Cordero Leonhardes (2014089399)

Profesor:

Francisco Torres Rojas

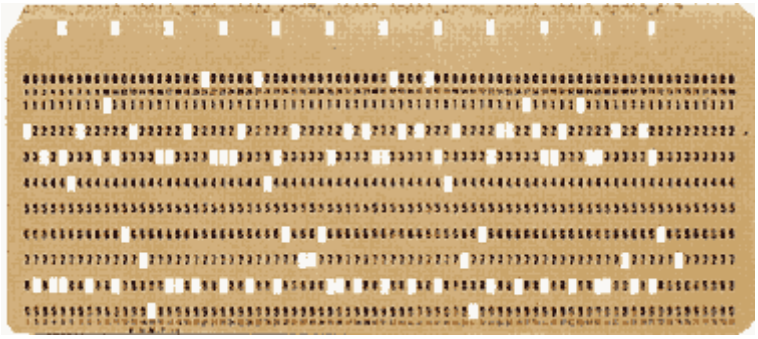
I Semestre 2017

Recordatorios

- Investigar acerca de Flex y Beamer
- Recordar la tarea larga para este miércoles 8 de marzo

Tarjetas perforadas

El Profesor nos enseñó un par de tarjetas perforadas donde una decía Te Amo (dato curioso: el profe no recuerda el nombre de quien le escribió el mensaje), además de enseñarnos su primera matricula de cuando aún estudiaba física. Pero se cambió de carrera dado que el papa lo convenció de que si estudiaba física iba a ser profesor.



Programas relacionados con los compiladores

Muchos softwares del sistema se relacionan directa o indirectamente con los compiladores, como los sistemas operativos y lenguajes de alto nivel que han sido sumamente exitosos, el desarrollo del software actual está asociado totalmente a su uso.

Ensamblador

Fueron inventados en 1950s, todavía viven, es programación de bajo nivel y tiene relación 1 a1 con el lenguaje máquina, busca ser un lenguaje eficiente y de sintaxis simple lo cual lo diferencia del lenguaje máquina.

Utilizan campos fijos. Y un comando en ensamblador está compuesto de las siguientes partes (no necesariamente necesitan estar las 3).

- Etiquetas.
- Código operación.
- Argumentos.

Ejemplo de código en ensamblador

Existen ensambladores de dos pasadas o de una pasada, los primeros son más utilizados.

Tomando el siguiente código como ejemplo:

```
    JMP ACA
    ----
    ----
    ----
    ----
ACA: ----
    ----
```

En la primer pasada lo que se hace es buscar etiquetas y guardarlas en una tabla de símbolos, la cual suponiendo que ACA se encuentra en la posición 417 se vería de la siguiente manera.

Etiqueta	Valor
ACA	417

Luego en la segunda pasada se empieza a crear el código máquina, usando la tabla de símbolos, por lo cual al encontrar la operación JMP a la etiqueta ACA no escribirá acá sino que hará un JMP a la posición 417.

En el ensamblador de una pasada se empieza a traducir el código, pero al encontrar una etiqueta que no ha sido declarada se guarda esta posición, para cuando sea declarada se vuelva a la posición que se había guardado para agregar dicha posición (Backpatching).

Para escribir un compilador se debe saber bastante ensamblador.

Desensamblador

Es el proceso inverso a un ensamblador dado que este toma el lenguaje máquina y lo convierte a ensamblador.

Un problema al utilizar el desensamblador son las etiquetas, debido a que en el lenguaje maquina no se guarda el nombre de la etiqueta, sino su posición, también se dan problemas con los macros, debido a que en vez de generar el macro, escribirá el código ensamblador completo donde debería estar un llamado al macro, para esto se puede hacer que el código identifique donde el código se repite y crear un macro para este, pero no es lo normal en un desensamblador

Recuerda a un simulador de la arquitectura.

- Debe reconocer los formatos en binario.
- Reconoce códigos de operación y formatos de direccionamiento.

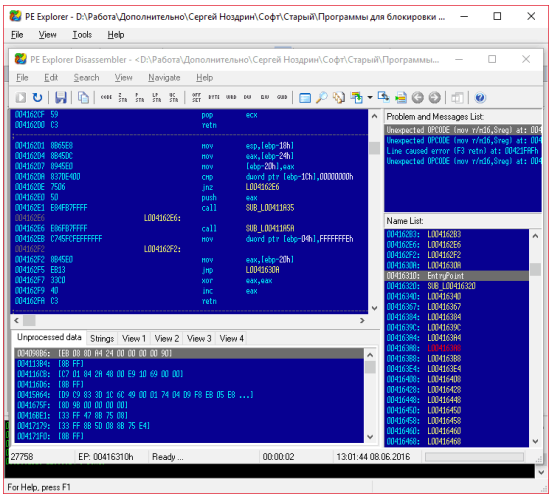
Datos vs Código

Es complicado diferenciar la sección de datos y código, por ejemplo saber que lo que se encuentra en una posición dada es un define byte y no una instrucción.

¿Macro, pseudoinstrucciones?

Dado que los macros y pseudoinstrucciones se utilizan para referenciar un lugar del código en el cual se desea agregar una serie de instrucciones, al ser traducido a lenguaje maquina no queda nada del macro o pseudocódigo, sino esta serie de instrucciones mencionada.

El desensamblador tiene diversos usos como por ejemplo, debuggin, ingeniería inversa, antivirus, o para recuperar códigos viejos donde se perdió el fuente.



Desensamblador

El Profe recomienda ver la serie Halt and Catch fire donde roban el código del IMB PC desensamblando el código.

Imagen de la serie

Interpretes

Combina la traducción con la ejecución. Corre el código línea por línea, además utiliza muchas técnicas equivalentes a la del compilador, como su gran virtud destaca su independencia de la arquitectura, además de que corre como run time enviroment (Máquina Virtual), su mayor defecto es que al ejecutarse debe traducirse cada línea de código, a diferencia del compilador donde el código se traduce una sola vez.

Algunos ejemplos de intérpretes son: LISP, BASIC, Python, Ruby y Forth.

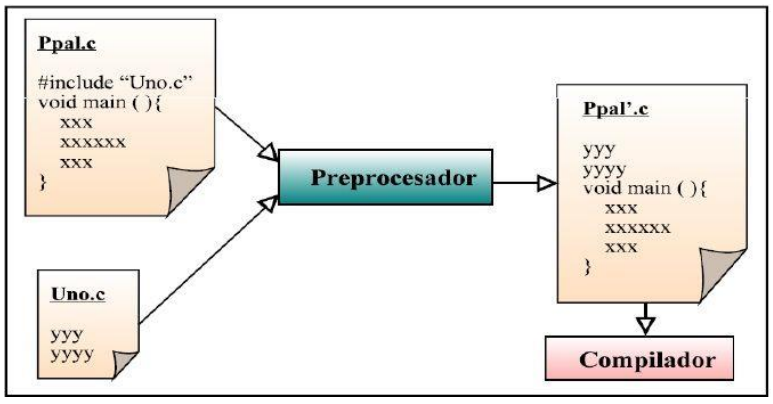
Preprocesadores

Se ejecutan antes del proceso, antes se debía ejecutar el preprocesador y luego el compilador, pero como siempre era necesario hacer esto se optó por unir estos dos, y a la hora de compilar ejecuta el preprocesador antes, pero sin que el programador lo note. Un ejemplo de este uso del preprocesador se da en C, un ejemplo de un comando para el preprocesador son los includes.

Las funciones típicas del preprocesador son:

- Incluir archivos.
- Remplazar hileras.
- Definir macros.
- Eliminar comentarios.
- Compilación condicional.

Estos usan técnicas de compilación son casi un lenguaje (cerca, pero no es un compilador), los preprocesadores pueden ser independientes del leunguaje.



Ejemplo de preprocesador utilizando include

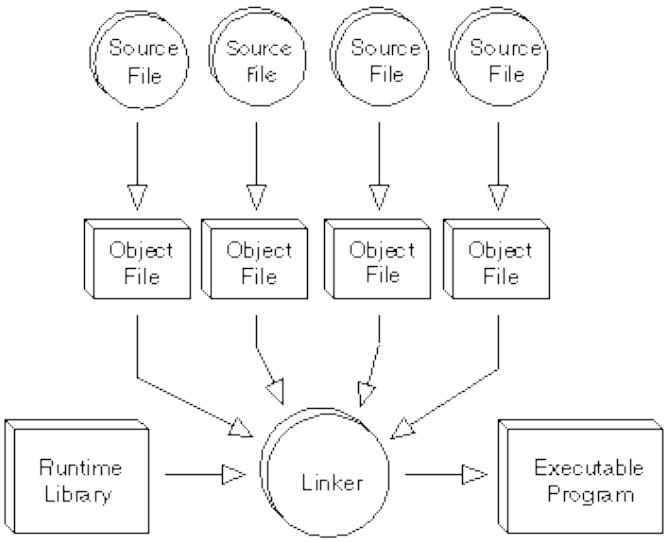
Linker

Se encarga de agregar Bibliotecas que no son parte del lenguaje pero son necesarias para su ejecución, por ejemplo un printf en C, Su traducción literal sería algo como “Ligador” o “Eslabonador”, Los compiladores o ensambladores generan módulos objeto. Luego de utilizar el linker se crea un módulo ejecutable.

El linker muchas veces viene incluido en el compilador, y existen dos tipos de linking

Linking estático: Ya todo lo necesario se encuentra, es más rápido, ocupa mucha memoria, y no noecesita las bibliotecas para funcionar luego de compilado.

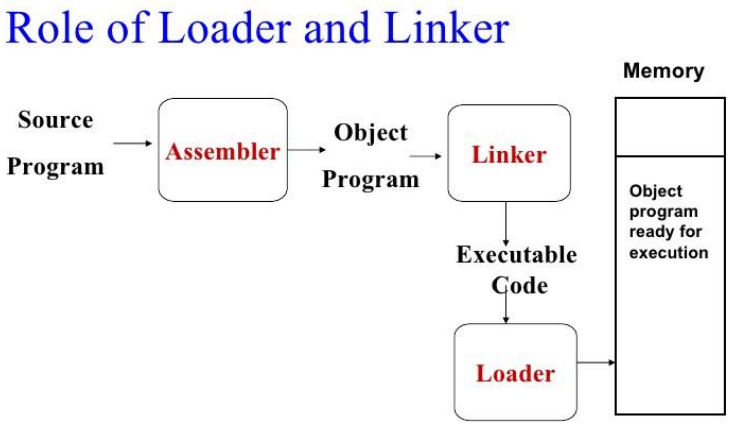
Linking dinámico: Necesita las bibliotecas luego de compilado, cuando necesita alguna función de una biblioteca la busca en esta, como ventaja si la biblioteca se actualiza no es necesario recompilar para usar la nueva versión, este tipo de linking es el más usado actualmente.



Loader

Carga el programa en memoria, en otra época era un programa aparte ahora es parte del sistema operativo. Toma un ejecutable y lo carga en memoria , talvez sea necesario relocalizar. Se encarga de negociar espacio.

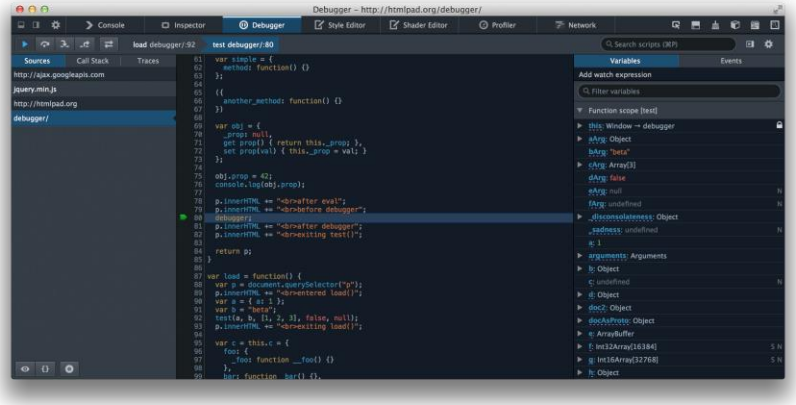
El loader crea estructuras de control, por ejemplo los PCB (Process Control Block), aquí se agrupa toda la información que necesita conocer respecto a un proceso partícula



Debugger

Permite la ejecución de forma controlada, ejecuta instrucción por instrucción, además permite colocar break point (hardware). Permite ver el contenido de variables, esto lo logra utilizando la misma tabla de símbolos que se creó en compilación. El problema es que para que el debugger funcione se debe agregar código extra para que el programa se detenga y analizar los datos de las variables, por lo cual se le puede solicitar al compilador que no genere este código extra.

Nota: No se tiene derecho a utilizar el debugger si no se sabe cómo funciona, sino seria magia negra.



Profiler

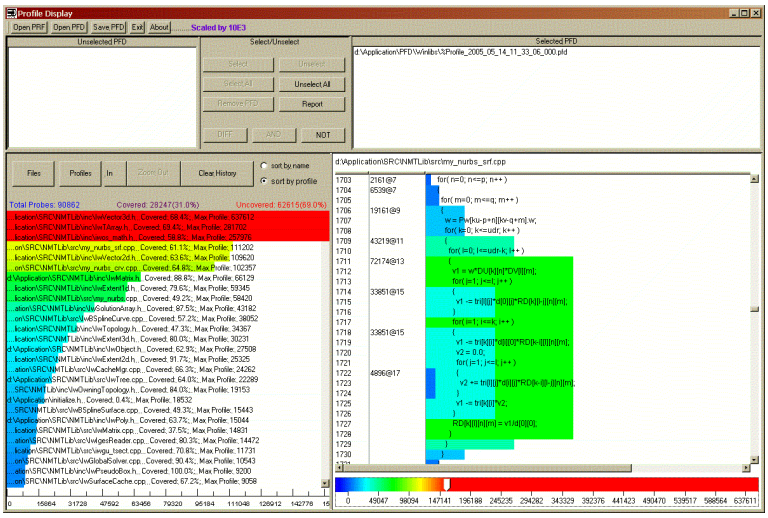
Recolecta estadísticas de ejecución de un programa, puede calcular datos como, que partes del código se utilizan más, o cuánto tiempo tarda cada función en ejecutarse, esto ayuda a mejorar el

rendimiento dado que cuando se va a mejorar una función primero se busca mejorar la función que se use más y que a su vez sea lenta, la función se puede optimizar analizando una mejor manera de hacer la misma función con menos recursos o reescribiendo el código en ensamblado

Si no se utiliza un profiler se puede llegar al error de optimizar una función que casi nunca se usa, haciendo que el trabajo que se dedicó a la optimización no valga la pena.

El compilador se puede beneficiar de datos recolectados, en este caso en vez de que el programador cambie el código es el compilador el que se recompila para optimizarlo, esta tarea se le puede asignar al compilador cada cierto tiempo para que optimice el código.

El profiler busca ser lo menos intrusivo posible, para no alterar los resultados de sus análisis y esto lo logra colocando timers en lugares aleatorios del código, que al llegar a esa línea del código guarda el tiempo que se tardó en llegar a dicho punto.

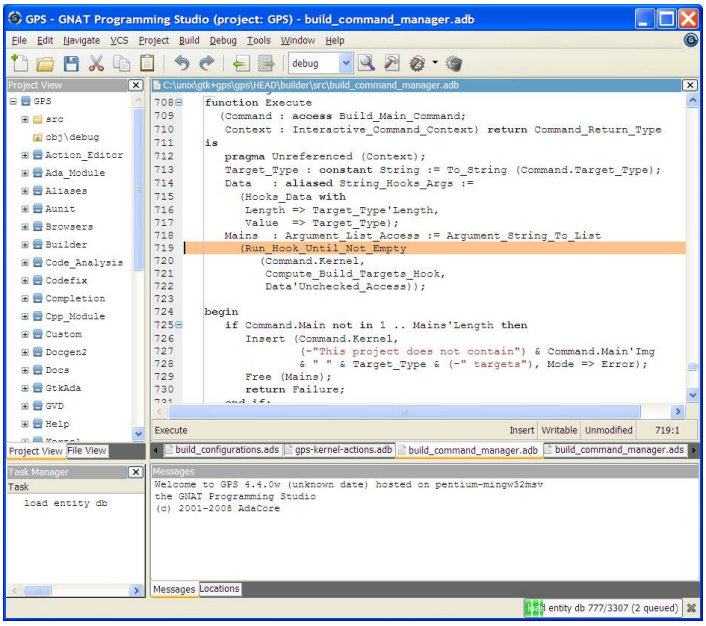


Editor de texto

Ayuda al programador a tener un mejor entendimiento del código, esto lo logra resaltando expresiones regulares, recordando el nombre de las variables declaradas entre otras funciones, algunos ejemplos son sublime, notepad++ pero uno de los más usados es EMACAS creado por Richard Stallman (si, el mismo loco de la clase pasada), y según el profe (no pude comprobarlo) puede hasta diferenciar cuando se escribe en algún lenguaje de programación o cuando se le escribe una carta a la novia.

Los editores usan técnicas de compiladores para sus funciones, expresiones regulares para búsquedas, usan plugin para alterar el comportamiento, además tienen un control de variables utilizando algo similar a una tabla de símbolos. Además permiten marcar con diferentes colores o tipos de letra diferentes partes del código, informar de errores sintácticos.

También existen los integrate development enviroment o IDE que incluye un editor, compilador, debugger entre otros en un sola aplicación.



Ejemplo de un IDE

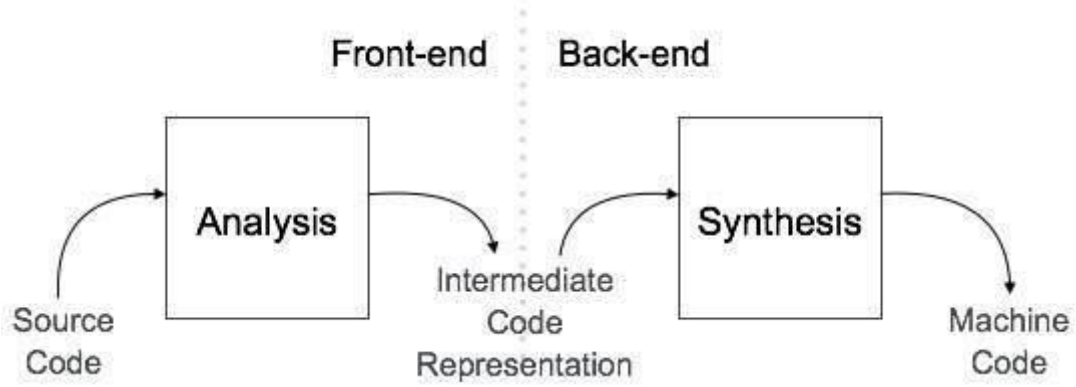
Manejador de versiones.

Normalmente se utiliza para proyectos que requieran muchas personas trabajando en el mismo fuente, guarda quien hizo que y cuando, permite regresar a un punto previo m estos pueden tener un conocimiento sintáctico del lenguaje en que se está trabajando, pero esto no es necesario. Ejemplo Git Hub

Estructura básica de un compilador

Entrada: Lenguaje fuente
Salida: Lenguaje objeto

El compilador toma la fuente y la convierte en codigo objeto, pero dentro del compilador existe una representación intermedia, que eta compuesta por dos partes: el front end y el back end.



El front end se encarga de leer el lenguaje y convertirlo a una representación intermedia, mientras el back end toma esa representación y la convierte en código máquina.

Análisis y síntesis

Analizar: Convertir en partes pequeñas para entenderlo, lo que hace el front end.

Sintetizar: Construye algo con varias piezas sueltas, lo que hace el back end.

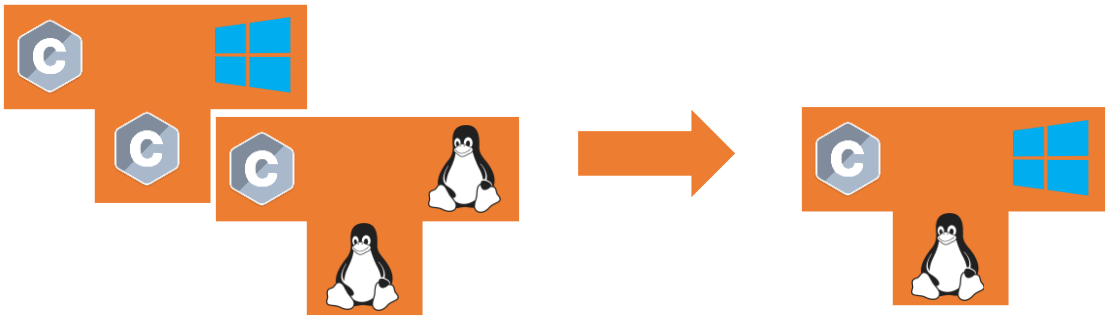
Entre más independiente uno del otro existe mayor flexibilidad.

Ejemplo compilador C

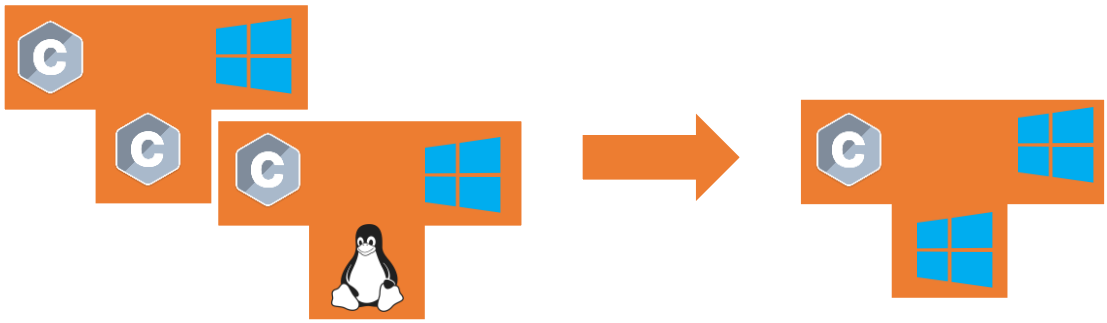
Se desea desarrollar en Linux un compilador de C, este compilador debe correr en Windows y generar código que corra en Windows.

Se tiene el compilador GCC, que se ejecuta sobre Linux y genera Linux, y además se tiene el fuente del compilador GCC, que corre en C y genera Linux, para solucionar este problema se puede estudiar el back end y modificarlo (lo que se hizo en el proyecto, mayormente).

Luego de modificarlo la fuente de gcc para que genere Windows se debe correr sobre el gcc para que genere Windows y corra sobre Linux.



Luego se vuelve a compilar la fuente modificada sobre el compilador resultado.



Ya con esto ha finalizado, y además creado una aberración que no merece existir.