Proyecto 2: Análisis Sintáctico

Viernes 10 de Mayo

I. Descripción

En este proyecto, Ud deberá desarrollar un *parser* para el Lenguaje C completo. Toda la programación debe realizarse en C sobre Linux, usando las herramientas `flex` y `bison`. No se pueden cambiar las especificaciones de este documento.

Su programa recibe como entrada un programa fuente, aparentemente escrito en lenguaje C, y produce como salida un reporte de **todos** los errores sintácticos detectados. La salida se dirige a consola.

II. Ejecución

El programa se invocará desde la línea de comando de una consola. Recibirá como argumento el nombre del archivo fuente a ser procesado, junto con cualquier opción necesaria para ejecutar el programa al estilo tradicional de `UNIX/Linux`. Si el usuario no da los argumentos, o si estos contienen errores (en cantidad o forma) se desplegará una ayuda explicando las opciones disponibles.

III. Preproceso

En el proyecto anterior se desarrolló un preprocesador que podía manejar las directivas `#include` y `#define` sin parámetros. Dicha funcionalidad debe estar correcta para este proyecto.

Trabajo extra opcional 1: Manejar macros con parámetros

Se debe investigar el comportamiento del compilador `gcc` respecto a la directiva `#include` y reproducirlo completamente en su programa. Por ejemplo, se deben manejar archivos encerrados entre "<" y ">".

IV. Errores Sintácticos

Nótese que el preproceso y el componente léxico serán básicamente los mismos del proyecto programado previo (corregirlos si tuvieran problemas pendientes).

Con la ayuda de `bison`, Ud. escribirá un *parser* del lenguaje C **completo**. Debe detectar la mayor cantidad posible de errores sintácticos y generar **buenos mensajes de error**, señalando el lugar dentro del fuente original donde se da el error sintáctico de la manera más precisa posible.

V. Salida

Por defecto, los errores de sintaxis encontrados se reportan a consola (al estilo de `gcc`). Se debe deplegar la línea del fuente donde ocurre el error y un indicador llamativo de la posición más aproximada posible al lugar donde se detectó el error. Si no se encuentran errores de sintaxis no se da ningún mensaje adicional.

Trabajo extra opcional 2: Si al invocar este programa se da la opción de línea de comando "-B" se debe generar como salida una presentación *Beamer* de gran calidad (la cual será desplegada en modo presentación automáticamente) que incluirá como mínimo lo siguiente:

- Portada identificando al grupo de trabajo, al semestre del curso, y al proyecto.
- Una explicación general del proceso de *parsing* y de la herramienta `bison`.
- Múltiples *slides* con el programa fuente que le entró a la fase de *parsing* (*i.e.*, después del preproceso). Todos los errores sintácticos encontrados deben ser reportados con el mejor mensaje de error posible. Además de alguna manera llamativa se debe indicar el lugar físico adonde se encontró el error. Ponga una cantidad razonable de líneas del fuente original en cada *slide*.

VI. Pruebas

Ya que el preproceso de este proyecto no corresponde al que hace un compilador completo de C ni se maneja análisis semántico, puede que algunos programas que para `gcc` estén correctos sean rechazados por su proyecto y viceversa.

El día de la demostración cada grupo debe contar con 8 programas **grandes**[1] donde su proyecto se comporte de manera muy parecida al `gcc`: 4 de ellos no deben mostrar ningún error ni en `gcc` ni en su proyecto, y 4 de ellos deben mostrar errores sintácticos en ambas plataformas (aunque el reporte de errores no sea idéntico debiera ser lo más parecido posible).

Trabajo extra opcional 3: Cada uno de estos 8 programas da medio punto extra si es obtenido directamente (y usado sin alterar) de `http://www.ioccc.org/`.

VII. Requisitos Indispensables

La ausencia de uno solo de los siguientes requisitos vuelve al proyecto "no revisable" y recibe un 0 de calificación inmediata:

- Todo el código debe estar escrito en C

[1] al menos 500 líneas de código

- No puede dar “Segmentation Fault” bajo ninguna circunstancia
- El proyecto debe compilar y ejecutar en Linux (*i.e.*, debe estar “integrado”, explicaciones del tipo “*todo está bien pero no pudimos pegarlo*” provocan la cancelación automática de la revisión).
- El proyecto se debe presentar en una **máquina física** que levante en Linux (puede ser dual)

VIII. Fecha de Entrega

Demostraciones el **Viernes 10 de Mayo - 9:30am**. Mande además un `.tgz` con todo lo necesario (fuentes, makefile, readme, etc.) a `torresrojas.cursos@gmail.com`. Ponga como subject: `COMPILADORES - Proyecto 2 - Fulano - Mengano - etc.`, donde Fulano, Mengano y etc. son los miembros del grupo.