Universidad del Valle de Guatemala | Josué Isaac Morales González
Construcción de compiladores | 21116
Sección 10 | 07-07-2024

Laboratorio 1

Construcción de la imagen del contenedor

```
(base) isaackeitor@Josues-MacBook-Air lab-1 % docker build -t compiler_env .

[+] Building 1.3s (10/10) FINISHED                                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                        0.0s
 => => transferring dockerfile: 285B                                                                        0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest                                            1.2s
 => [auth] library/ubuntu:pull token for registry-1.docker.io                                               0.0s
 => [internal] load .dockerignore                                                                           0.0s
 => => transferring context: 2B                                                                             0.0s
 => [1/4] FROM docker.io/library/ubuntu:latest@sha256:2e863c44b718727c860746568e1d54afd13b2fa71b160f5cd9058fc436217b30   0.0s
 => [internal] load build context                                                                           0.0s
 => => transferring context: 10.16kB                                                                        0.0s
 => CACHED [2/4] RUN apt-get update -y   && apt-get install -y --no-install-recommends build-essential bison flex   && rm -rf /var/lib/apt/lists/*   0.0s
 => [3/4] COPY files /home/files                                                                            0.0s
 => [4/4] WORKDIR /home                                                                                     0.0s
 => exporting to image                                                                                      0.0s
 => => exporting layers                                                                                     0.0s
 => => writing image sha256:38fa756da80d12dc2765fb04ecfa0986bad7b1f8fdd74273a467a8cc7c8d43b7                0.0s
 => => naming to docker.io/library/compiler_env                                                             0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/pa8ems2a3c2gikql0l79yce8z

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
(base) isaackeitor@Josues-MacBook-Air lab-1 %
```

```
What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
(base) isaackeitor@Josues-MacBook-Air lab-1 % docker run -it compiler_env

root@f5aeb753a9c7:/home#
```

Creación de archivo ScriptShell

```
(base) isaackeitor@Josues-MacBook-Air lab-1 % docker run -it compiler_env

root@f5aeb753a9c7:/home# echo '#!/bin/sh' > buildLanguage.sh
root@f5aeb753a9c7:/home# echo 'flex /home/files/simple_language.l' >> buildLanguage.sh
root@f5aeb753a9c7:/home# echo 'yacc -dtv /home/files/simple_language.y' >> buildLanguage.sh
root@f5aeb753a9c7:/home# echo 'g++ -c lex.yy.c' >> buildLanguage.sh
root@f5aeb753a9c7:/home# echo 'g++ -c y.tab.c' >> buildLanguage.sh
root@f5aeb753a9c7:/home# echo 'g++ -o calc y.tab.o lex.yy.o' >> buildLanguage.sh
root@f5aeb753a9c7:/home#
```

Ejecición de buildLanguage.sh

```
root@f5aeb753a9c7:/home# sh buildLanguage.sh
root@f5aeb753a9c7:/home# ls
buildLanguage.sh  calc  files  lex.yy.c  lex.yy.o  ubuntu  y.output  y.tab.c  y.tab.h  y.tab.o
root@f5aeb753a9c7:/home#
```

Ejercicios

1. *Cree un programa que asigne un valor a una variable.*

```
root@f5aeb753a9c7:/home# ./calc
i = 25
Assign i = 25
```

2. *Cree un programa que realice una operación aritmética simple.*

```
root@f5aeb753a9c7:/home# ./calc
20 + 20 * 20
420
```

3. *Experimente con expresiones más complejas y verifique que el compilador las procese correctamente.*

```
root@5f0fc2d25eb2:/home# ./calc
(50+50)*10
1000
(3 + 5) * (2 - (4 / 2)) + (7 * (8 - (3 + 2)))
21
```

4. *Modifique el lenguaje para incluir la asignación de variables con expresiones aritméticas.*

```
root@9e1beb9e3793:/home# ./calc
10%3
1
((5 % 2) + (4 ^ 2)) % 3
2
2 ^ 3
8
```

5. *Agregue manejo de errores al compilador para detectar tokens invÅLalidos en el programa fuente.*

```
...s/UVG/OCTAVO SEMESTRE/COMPIS/CC-2024/LAB1/lab-1 — root@d0
root@d046072f8988:/home# ./calc
1 ^ &
Please verify the token and try again. Invalid token: &
syntax error
root@d046072f8988:/home#
```

```
root@d046072f8988:/home# ./calc
i + * + 2
syntax error
```

6. *Experimente con la precedencia de operadores en el lenguaje y observe cómo afecta la generación del arbol sintactico*

```
root@d046072f8988:/home# ./calc
2 ^ 3 * 4
32
2 + 3 ^ 2
11
10 % 3 + 4
5
10 % 3 * 4
4
2 + 3 * 4 ^ 2 % 5
5
(2 + 3) * 4 ^ (2 % 5)
80
```

Resultados de las Expresiones

- Expresión: 2 ^ 3 * 4

  Evaluación:
  2 ^ 3 se evalúa primero debido a la mayor precedencia de la potenciación, resultando en 8. Luego, 8 se multiplica por 4.
  Resultado:
  8 * 4 = 32
  Explicación:
  La potenciación tiene mayor precedencia que la multiplicación.

- Expresión: 2 + 3 ^ 2

  Evaluación:
  3 ^ 2 se evalúa primero debido a la mayor precedencia de la potenciación, resultando en 9. Luego, 2 se suma a 9.
  Resultado:
  2 + 9 = 11
  Explicación:
  La potenciación tiene mayor precedencia que la suma.

- Expresión: 10 % 3 + 4

  Evaluación:
  10 % 3 se evalúa primero debido a la mayor precedencia del módulo, resultando en 1. Luego, 1 se suma a 4.
  Resultado:
  1 + 4 = 5
  Explicación:
  El módulo tiene mayor precedencia que la suma.

- Expresión: 10 % 3 * 4

  Evaluación:
  10 % 3 se evalúa primero debido a la mayor precedencia del módulo, resultando en 1. Luego, 1 se multiplica por 4.
  Resultado:
  1 * 4 = 4
  Explicación:
  El módulo tiene mayor precedencia que la multiplicación.

- Expresión: 2 + 3 * 4 ^ 2 % 5

  Evaluación:
  Primero, 4 ^ 2 = 16 debido a la mayor precedencia de la potenciación.
  Luego, 3 * 16 = 48 debido a la multiplicación.
  Luego, 48 % 5 = 3 debido a la mayor precedencia del módulo.
  Finalmente, 2 + 3 = 5.
  Resultado:
  5
  Explicación:
  La potenciación tiene mayor precedencia, seguida de la multiplicación y el módulo, y finalmente la suma.

- Expresión: (2 + 3) * 4 ^ (2 % 5)

  Evaluación:
  Primero, 2 % 5 = 2 debido a la mayor precedencia del módulo dentro de los paréntesis.
  Luego, 4 ^ 2 = 16 debido a la mayor precedencia de la potenciación dentro de los paréntesis.
  Luego, 2 + 3 = 5 debido a la suma dentro de los paréntesis.
  Finalmente, 5 * 16 = 80.
  Resultado:
  80
  Explicación:
  Los paréntesis cambian la precedencia normal, forzando la evaluación de las operaciones dentro de ellos primero.

Sin precedencia

```
root@a45f20c1551c:/home# echo '#!/bin/sh' > buildLanguage.sh
root@a45f20c1551c:/home# echo 'flex /home/files/simple_language.l' >> buildLanguage.sh
echo 'yacc -dtv /home/files/simple_language.y' >> buildLanguage.sh
echo 'g++ -c lex.yy.c' >> buildLanguage.sh
echo 'g++ -c y.tab.c' >> buildLanguage.sh
echo 'g++ -o calc y.tab.o lex.yy.o' >> buildLanguage.sh
root@a45f20c1551c:/home# sh buildLanguage.sh
/home/files/simple_language.y: warning: 36 shift/reduce conflicts [-Wconflicts-sr]
/home/files/simple_language.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
root@a45f20c1551c:/home#
```

Resultados de las Expresiones

```
root@a45f20c1551c:/home# ./calc
2 ^ 3 * 4
4096
2 + 3 ^ 2
11
10 % 3 + 4
3
10 % 3 * 4
10
2 + 3 * 4 ^ 2 % 5
50
(2 + 3) * 4 ^ (2 % 5)
80
```

En estos ejemplos se puede observar que no se sigue la precedencia de operaciones, lo que resulta en cálculos incorrectos y, en algunos casos, inconsistentes, incluso cuando la expresión es la misma. Por eso, es importante declarar las directivas %right y %left, que se utilizan para establecer la precedencia y asociatividad de los operadores, resolviendo ambigüedades en las expresiones matemáticas. En Yacc, la precedencia de los operadores se establece en orden ascendente, es decir, los operadores con menor precedencia se declaran primero, mientras que los de mayor precedencia se declaran al final. Además, Yacc tiene reglas predeterminadas para resolver conflictos shift-reduce, como preferir shift sobre reduce. Esto puede hacer que los conflictos no sean evidentes de inmediato, pero garantizará una evaluación coherente y correcta de las expresiones (IBM, 2023).

Link a repositorio:
https://github.com/isaackeitor/CC-2024.git

Referencias

IBM. (2023). yacc grammar file declarations. Obtenido de https://www.ibm.com/docs/en/aix/7.2?topic=information-yacc-grammar-file-declarations