Universidad Católica del Norte

Escuela de Ingeniería en Computación e Informática

Documentación

Lenguaje Cewe: Documentación y Uso

Autor: Nicolás Gonzalo Cordero Varas
Fecha: 17 Junio 2025
RUT: 20.543.155-1
Docente: José Luis Veas

Índice

1. Guía de Uso e Instalación del Lenguaje CEWE

1.1. Requisitos del Sistema

Para compilar y ejecutar CEWE se necesita:

- Sistema operativo GNU/Linux.
- Flex y Bison.
- Compilador GCC.
- Bash y acceso a terminal.

1.2. Instalación de Paquetes

```
sudo apt update
sudo apt install flex bison gcc build-essential
```

1.3. Estructura del Proyecto

- `scanner.l` - Analizador léxico
- `parser.y` - Analizador sintáctico
- `ast.h, ast.c` - Árbol de Sintaxis Abstracta
- `main.c` - Función principal
- `build.sh` - Script de compilación
- `programa.cewe` - Código de ejemplo en Cewe

1.4. ¿Qué es un Script Bash?

Un script Bash es un archivo con comandos de terminal usados para automatizar tareas. El script `build.sh` automatiza la compilación del compilador Cewe.

1.5. Compilación

```
chmod +x build.sh
./build.sh
```

1.6. Ejecución de Programas Cewe

```
./cewe programa.cewe
```

2. Descripción del Código Fuente

2.1. scanner.l

Define tokens mediante expresiones regulares. Ejemplo:

```
"printuwu"      { return PRINT; }
[0-9]+          { yylval.entero = atoi(yytext); return NUM; }
```

2.2. parser.y

Define la gramática y acciones semánticas con reglas BNF.

```
expresion: expresion '+' expresion {
  $$ = crearNodoOperacion('+', $1, $3);
}
```

2.3. ast.h

Define tipos y estructura del AST:

```
typedef enum {
  NODE_NUM, NODE_IDENTIFICADOR, NODE_OPERACION_BINARIA,
  NODE_LLAMADA_FUNCION, NODE_DECLARACION_FUNCION,
  NODE_RETURN, NODE_IF, NODE_WHILE, NODE_BLOQUE, NODE_PRINT
} ASTNodeType;
```

2.4. ast.c

Funciones para crear, imprimir y evaluar nodos del AST.

```
int evaluarAST(ASTNode* nodo) {
  if (nodo->tipo == NODE_OPERACION_BINARIA) {
    int izq = evaluarAST(nodo->izquierdo);
    int der = evaluarAST(nodo->derecho);
    switch (nodo->operador) {
      case '+': return izq + der;
      case '*': return izq * der;
    }
  }
}
```

2.5. main.c

Función principal para leer, parsear y evaluar el programa Cewe:

```
int main(int argc, char** argv) {
  FILE* archivo = fopen(argv[1], "r");
  yyin = archivo;
  yyparse();
  evaluarAST(astPrincipal);
  fclose(archivo);
}
```

2.6. build.sh

Automatiza la compilación:

```
#!/bin/bash
rm -f lex.yy.c parser.tab.c parser.tab.h cewe
bison -d parser.y
flex scanner.l
gcc -o cewe main.c ast.c parser.tab.c lex.yy.c -lfl
```