实验一 讲解

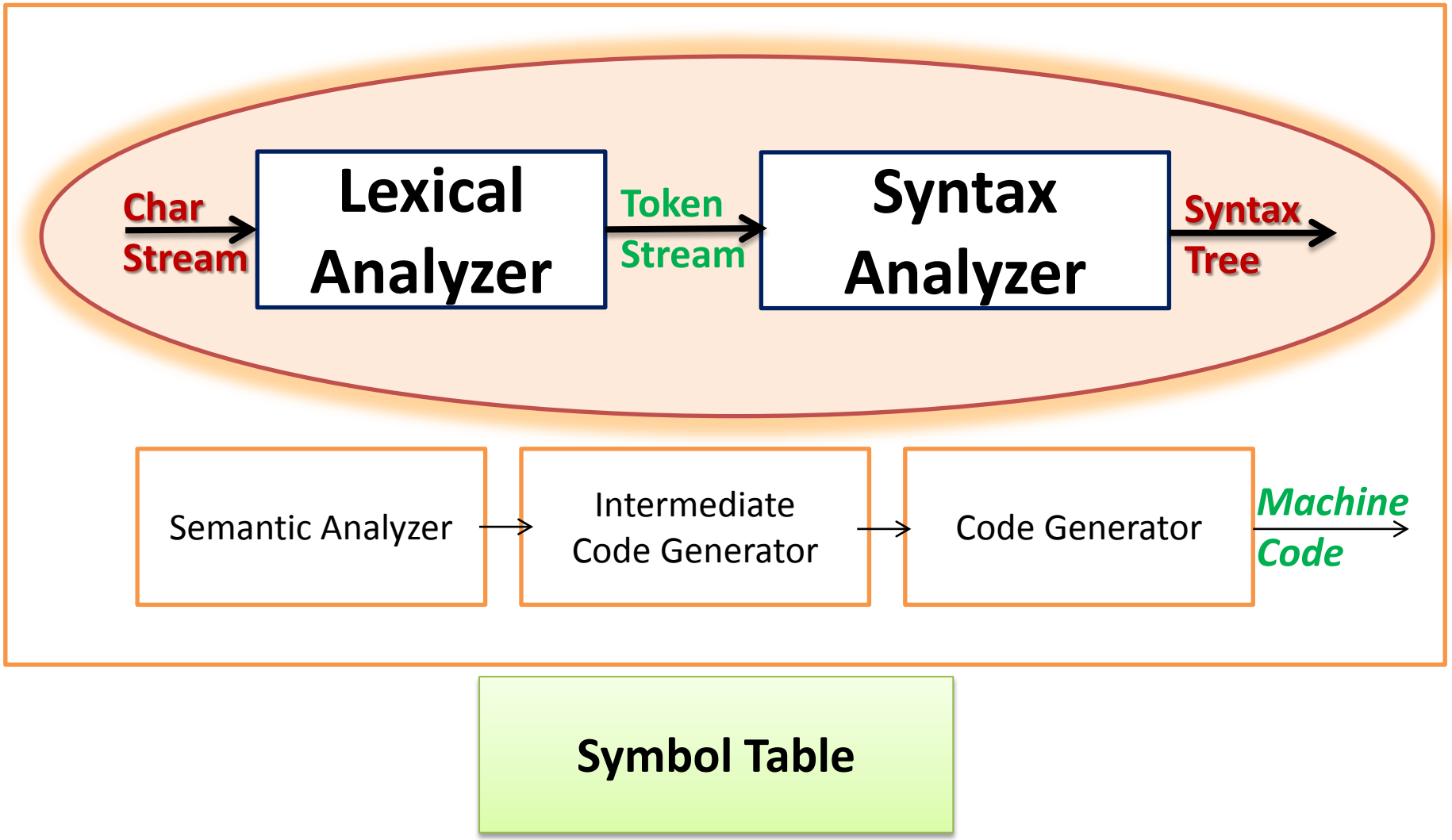
with pure C

助教：胡光能，解定宝

编译原理讲师：戴新宇

hugn@nlp.nju.edu.cn

Phases of a Compiler



Outline

- 提交说明
- 实验任务（必做 + 选做）
- 编译环境及过程
  - 词法分析与flex
  - 语法分析与bison
- 实验讲解
  - 文法二义性消除
  - 语法树创建与打印
  - 文法符号结点的数据结构
  - 语法解析的错误恢复

提交说明

- 地址：ftp://114.212.190.181: 31
- 用户名和密码：upload
- 格式：学号命名的压缩包 (zip/rar)
- 内容：
  - 源程序(ex1.l, ex1.y; 额外的.c 文件)
  - 可执行程序(命名为 parser)
  - 报告PDF(完成的功能点，编译步骤，实现方法，结点的数据结构表示；不超过3 页)
- 备注：可重新提交，加后缀 _02, _03

实验任务

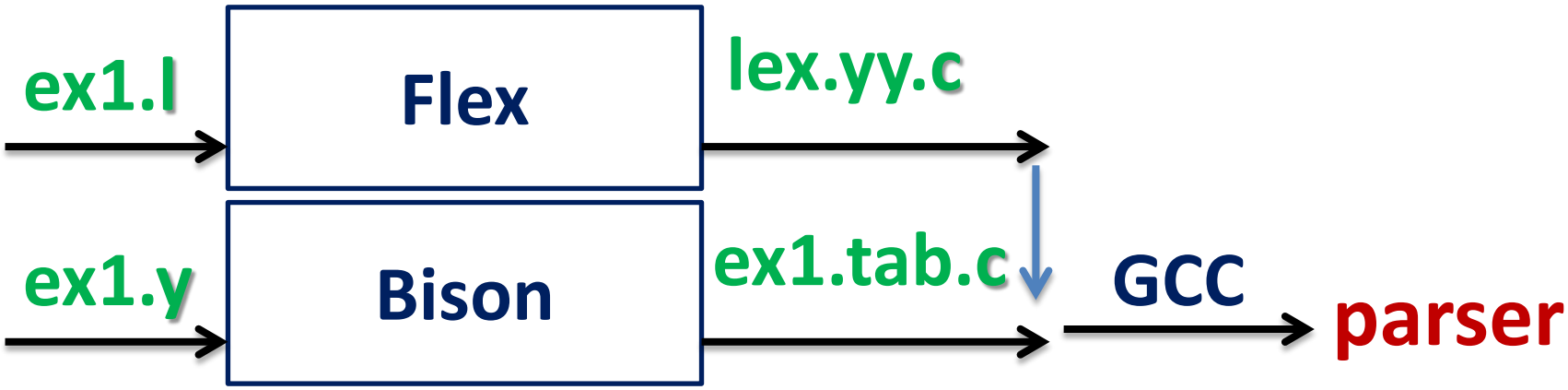
- *预备：熟悉C--文法和实验要求*
- **必做：**
  - 错误类型1: 词法错误(词法未定义字符)
  - 错误类型2: 语法错误(如[3,9])
- **选做：**
  - **选做1：两种风格的注释：//，/* */**
  - **选做2：八进制数：012**
  - **选做3：十六进制：0xa，0Xa，0xA，0XA**
  - **选做4：指数形式的浮点数：1E1，01e1**

编译环境及过程

- **GNU Flex, GNU Bison, GCC, Linux Ubuntu**
  - sudo apt-get install *flex*
  - sudo apt-get install *bison*
- **源文件{ex1.l, ex1.y} → 可执行程序parser**
  - Flex： ex1.l → lex.yy.c
  - Bison: ex1.y → ex1.tab.c
  - GCC： *.c → parser

./parser test.c //测试命令

编译 方法




```
flex  ex1.l
bison  -d  ex1.y
gcc  -o  parser  ex1.tab.c  -ll  (-lfl)
```

-ll: lib of ***l***ex -lfl: lib of ***fl***ex -ly: lib of ***y***acc

Flex & Bison

%{ ***ex1.l***

Declarations

#include "ex1.tab.h"

%}

Definitions (**RegEx**)

%%

Rules

%%

subroutines (***e.g main***)

%{ ***ex1.y***

Declarations

#include "lex.yy.c"

%}

Definitions (**%Token**)

%%

Productions

%%

subroutines

启动语法分析器

```
int main(int argc, char** argv){
        if(argc <= 1) return 1;
        FILE* f = fopen(argv[1],"r");
        if(!f) {
                perror(argv[1]);
                return 1;   }
        yyrestart(f);  //输入文件指针置为 yyin = f
        yyparse();     //main函数启动解析器
        return 0;  }
```

Flex预定义变量

- **yytext, yyleng:** 词素字符串
- **yylineno:** *%option yylineno*
- **yylval:** 全局变量，当前词法单元的属性值

```
{id}    { strncpy(id_lexeme, yytext, yyleng);
          yylval.pNode = createNode( …);
          return ID;     }
```

```
%union {
        int       val ;
        char*   str;
        struct  Node*  pNode;
}
```

保留字和标识符

```
%{
Declarations
%}
Definitions
%%
Rules        //将保留字置于标识符{id}之前
%%
subroutines
```

文法二义性: 操作符优先级与结合性

%{

Declarations

%}

Definitions

//优先级与结合性

%%

Productions

%%

subroutines

```
%right  ASSIGNOP
%left   AND
%left   RELOP
%left   PLUS MINUS
%left   STAR DIV
%right  NOT UMINUS
%left   DOT LB RB LP RP
```

Exp | MINUS EXP ***%prec*** UMINUS

文法二义性: IF-ELSE配对

```
Stmt:   IF LP Exp RP Stmt
      | IF LP Exp RP Stmt ELSE Stmt
```

```
%nonassoc IFX
%nonassoc ELSE
```

```
Stmt:   IF LP Exp RP Stmt   %prec IFX
      | IF LP Exp RP Stmt ELSE Stmt
```

语法树创建与打印

- 多叉树的构建：
  - Exp : ID { $$ = createNode($1); }
  - Exp : MINUS EXP { $$ = createNode($1,$2); }
  - *#include<stdarg.h>*

  struct Node* createNode(int arity, …);
- 递归层次的前序遍历

  void printNode(struct Node* root, int nLayer);

文法符号结点的数据结构

- 保存的信息：***struct Node{ … };***
  - 结点类型： 非终结符，终结符(数, 标识符…)
  - 结点名字： Exp，TYPE，ID
  - 所在行号： *%option yylineno*
  - 字符串属性值：TYPE.int，ID.lexeme
  - 数值属性值：INT，FLOAT
  - **多叉树孩子：** *int arity,*

  struct Node children[N];* ***//vector<Node*>***

  -…(可扩展)

语法解析的错误恢复产生式

- Bison在当前状态对yylex()返回的token没有*定义*时即发生了语法错误，调用yyerror:

  yyerror(char str){ printf("syntax error\n"); }*

- **Bison不断丢弃词法单元直至遇到*同步*单元(*例如：分号，右括号*)**
- **机制：错误恢复产生式**

Stmt: error SEMI

参考资料

- Aho 编译原理：
  - http://dinosaur.compilertools.net/
- 斯坦福 编译原理：
  - http://www.stanford.edu/class/archive/cs/cs143/cs143.1128/handouts/050%20Flex%20In%20A%20Nutshell.pdf
  - http://www.stanford.edu/class/archive/cs/cs143/cs143.1128/handouts/120%20Introducing%20bison.pdf
- 指导攻略：
  - http://cs.nju.edu.cn/changxu/2%20compiler/projects/%E6%8C%87%E5%AF%BC%E6%94%BB%E7%95%A5%201.pdf
- 可变参数个数:
  - http://www.cplusplus.com/reference/cstdarg/va_start/
- 错误恢复：
  - http://oreilly.com/linux/excerpts/9780596155971/error-reporting-recovery.html