读写文本文件

2024年5月11日 10:08

1. 一个一个字符地读写

```
int fgetc(FILE* stream);
```

getchar() 等价于 fgetc(stdin);

```
int fputc(int c, FILE* stream);
```

putchar(c) 等价于 fputc(c, stdout);

```
// a. 一个字符一个字符地读写：fgetc, fputc
// 把大写转换成小写字母
int c;
while ((c = fgetc(src)) != EOF) {
    fputc(tolower(c), dst);
}
```

2. 一行一行地读写

```
char* fgets(char* str, int count, FILE* stream);
参数:
    str: 指向一个字符数组
    count: 能够写入的最大字符数量(通常是str指向字符数组的长度)
    stream: 输入流
返回值:
    成功: 返回str
    失败: NULL
```

```
int fputs(const char* str, FILE* stream);
参数:
    str: 要写的字符串(以'\0'结尾的字符串)
    stream: 输出流
返回值:
    成功: 返回一个非负值。
    失败: 返回EOF，并设置errno。
```

puts(str) 等价于 fputs(str, stdout)

```
// b. 一行一行地读写
char line[MAXLINE];
char buffer[MAXLINE];
int num = 1;
while (fgets(buffer, MAXLINE, src) != NULL) {
    sprintf(line, "%d. %s", num, buffer);
    fputs(line, dst);
    num++;
}
```

3. 格式化地读写

```
int fscanf(FILE* stream, const char* format, ...);
```

不同的是，scanf 是从标准输入(stdin)中读取数据，而 fsca

scanf 等价于 fscanf(stdin, format, ...)

```
int fprintf(FILE* stream, const char* format, ...);
```

printf 等价于 fprintf(stdout, format, ...)

读写二进制文件

2024年5月11日 11:16

```
size_t fread(void* buffer, size_t size, size_t count, FILE* stream);
```

参数:

buffer: 指向存放数据的数组

size: 每个元素的大小(以字节为单位)

count: 最多可以读取的元素个数

stream: 输入流

返回值:

成功读取元素的个数。当读到文件末尾，或者发生错误时，返回值可能小于count。

我们可以通过feof和ferror函数来判断，到底是读到了文件末尾，还是发生了错误。

```
size_t fwrite(const void* buffer, size_t size, size_t count, FILE* stream);
```

参数:

buffer: 指向存放数据的数组。

size: 每个元素的大小(以字节为单位)

count: 要写入元素的个数

stream: 输出流

返回值:

成功写入元素的个数。当发生错误时，这个值可能小于count。

移动文件位置

2024年5月11日 11:29

```
int fseek(FILE* stream, long int offset, int whence);
```

- SEEK_SET 文件开头
- SEEK_CUR 文件当前位置
- SEEK_END 文件末尾


```
long int ftell(FILE* stream);
```

```
void rewind(FILE* stream);
```
→ 回到文件的开头

课堂小练习

用户输入文件名，将整个文件的内容读入到字符数组中，并在后面添加空字符'\0'。

```
char* readFile(const char* path);
```

① 如何确定文件的大小

先移动到文件末尾，

ftell 获取文件位置。

错误处理

2024年5月11日 14:27

① 如何检测错误

a. 返回值

b. errno （每个线程都有自己errno，线程特定变量）

② 如何打印错误信息

```c
int main(void) {
    // printf("%d\n", errno); // 0: 没有错误
    FILE* stream = fopen("not_exist.txt", "r");

    printf("%d\n", errno);
    printf("%s\n", strerror(errno)); // 错误原因
    perror("prefix");
    return 0;
}
```

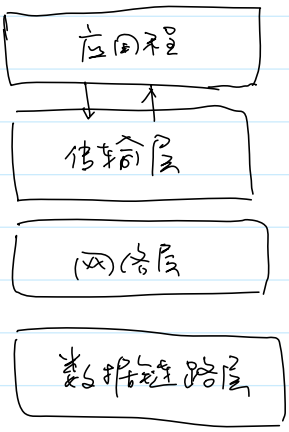
fprintf(stderr, "%s: %s\n", prefix, strerror(errno));

Big Picture

2024年5月11日 14:38

TCP/IP

分层架构.

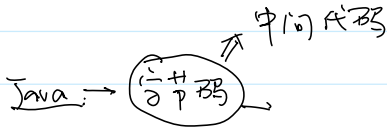


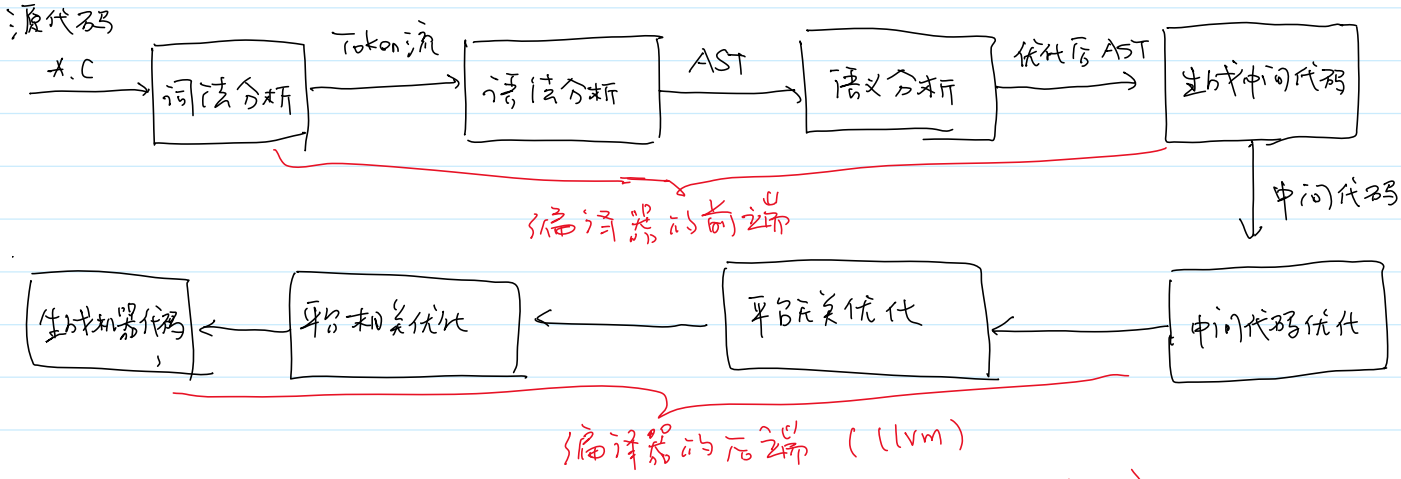
降低复杂度

速度低

流水线模型.

编译器





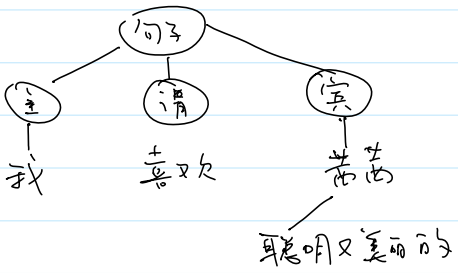
1. 词法分析

我/喜欢/聪明又美丽的/茜茜.

↓ 词: Token

茜茜/聪明又美丽的/喜欢/我

2. 语法分析



Abstract Syntax Tree

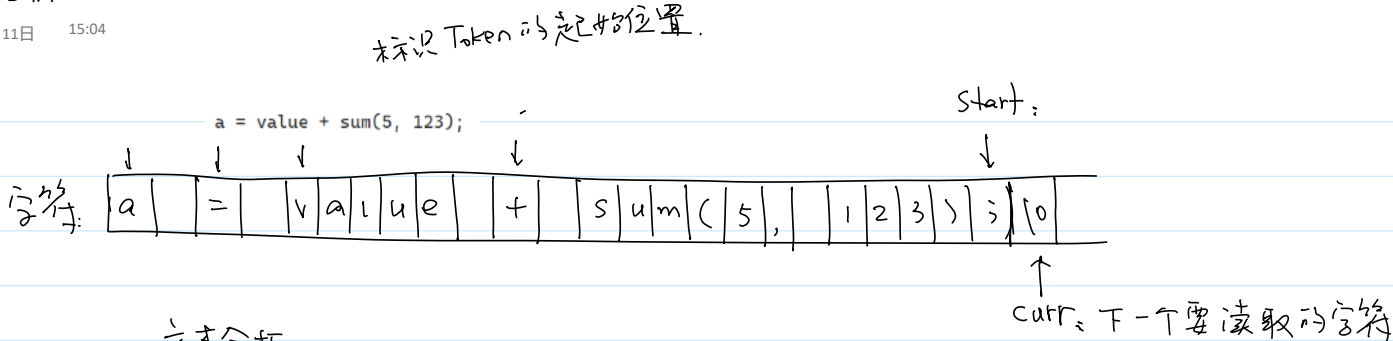
#13、 语义分析

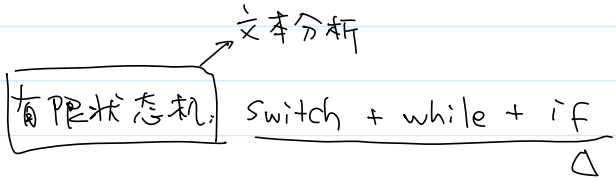
结合上下文，推断句子的含义.

↳ 作用域.

词法分析

2024年5月11日 15:04






```
printf("%s '%.*s'\n", strtoken(token), token.length, token.start);
```

%.*s

'printf'

scanner.curr
scanner.start
scanner.line

scanner.start

```
typedef struct {
    TokenType type;
    const char* start;  // start指向source中的字符，source为读入的源代码。
    int length;         // length表示这个Token的长度
    int line;           // line表示这个Token在源代码的哪一行，方便后面的报错
    // 附加题: int column;
} Token;
```