

Using Lex

The Structure of a Lex Program

(Definition section)

% %

(Rules section)

% %

(User subroutines section)

Example 1-1: Word recognizer ch1-02.l

```
%{
/*
 * this sample demonstrates (very) simple recognition:
 * a verb/not a verb.
 */

%}
%%

[ \t ]+      /* ignore white space */;

is |
am |
are |
were |
was |
be |
being |
been |
do |
does |
did |
```

```
will |
would |
should |
can |
could |
has |
have |
had |
go      { printf("%s: is a verb\n", yytext); }

[a-zA-Z]+ { printf("%s: is not a verb\n", yytext); }

.\n      { ECHO; /* normal default anyway */ }
%%

main()
{
    yylex();
}
```

The definition section

- Lex copies the material between “%{“ and “%}” directly to the generated C file, so you may write any valid C codes here

Rules section

- Each rule is made up of two parts
 - A pattern
 - An action
- E.g.

`[\t ]+       /* ignore white space */ ;`

Rules section (Cont'd)

- E.g.

is |

am |

are |

were |

was |

be |

being |

been |

do |

does |

did |

will |

would |

should |

can |

could |

has |

have |

had |

go

```
{ printf("%s: is a verb\n", yytext); }
```

Rules section (Cont'd)

- E.g.

```
[a-zA-Z]+      { printf("%s: is not a verb\n", yytext); }
```

```
.\n           { ECHO; /* normal default anyway */ }
```

- Lex had a set of simple disambiguating rules:
 1. Lex patterns only match a given input character or string once
 2. Lex executes the action for the ***longest*** possible match for the current input

User subroutines section

- It can consists of any legal C code
- Lex copies it to the C file after the end of the Lex generated code

```
%%
```

```
main()  
{  
    yylex();  
}
```

Regular Expressions

- Regular expressions used by Lex

.

*

[]

^

\$

{ }

\

+

?

|

“...”

/

()

Examples of Regular Expressions

- `[0-9]`
- `[0-9]+`
- `[0-9]*`
- `-?[0-9]+`
- `[0-9]*\.[0-9]+`
- `([0-9]+)|([0-9]*\.[0-9]+)`
- `-?(([0-9]+)|([0-9]*\.[0-9]+))`
- `[eE][-+]?[0-9]+`
- `-?(([0-9]+)|([0-9]*\.[0-9]+))([eE][-+]?[0-9]+)?`

Example 2-1

```
%%  
[\n\t ] ;  
  
-?(([0-9]+)|([0-9]*\.[0-9]+)([eE][-+]?[0-9]+)?) { printf("number\n"); }  
  
.    ECHO;  
%%  
main()  
{  
    yylex();  
}
```

A Word Counting Program

- The definition section

```
%{  
unsigned charCount = 0, wordCount = 0, lineCount = 0;  
%}  
  
word [^ \t\n]+  
eol \n
```

A Word Counting Program (Cont'd)

- The rules section

```
{word} { wordCount++; charCount += yyleng; }  
{eol} { charCount++; lineCount++; }  
.    charCount++;
```

A Word Counting Program (Cont'd)

- The user subroutines section

```
main(argc,argv)
int argc;
char **argv;
{

    if (argc > 1) {
        FILE *file;

        file = fopen(argv[1], "r");
        if (!file) {
            fprintf(stderr,"could not open %s\n",argv[1]);
            exit(1);
        }
        yyin = file;
    }
    yylex();
    printf(" %d %d %d\n",charCount, wordCount, lineCount);
    return 0;
}
```