

CS344 - Scanning & flex

Note Title

1/27/2012

Announcements

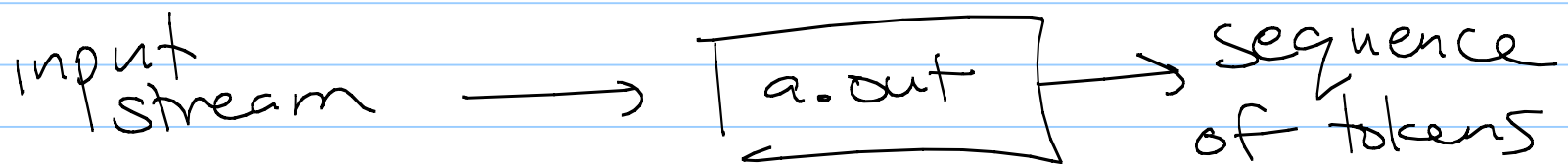
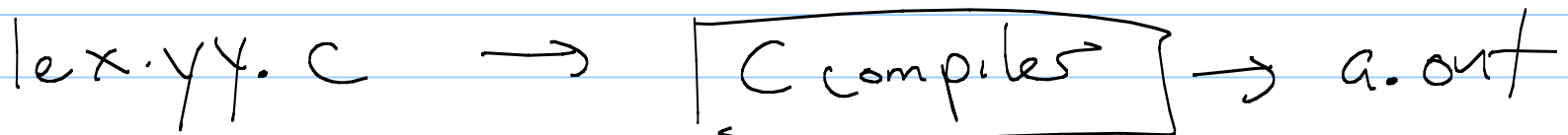
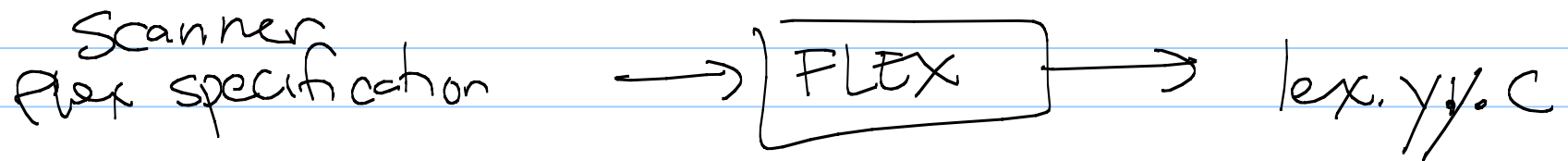
- No office hours tomorrow at 9

↳ will be around at 2-ish

- HW due Thursday

Flex

- A C driven scanning program.



To compile:

> flex file.lex

> gcc lex.yy.c -lfl

> ./a.out (< otherfile.txt)
if input

(if using stdin, might need ctrl-d)

Format for .lex files:

①

definitions

% %

②

rules

% %

③

user code

(see examples)

①

Definitions

New definitions to make life easier.

Form: name definition

Ex:

digit [0-9]

ID [a-z][a-z0-9]*

Note: These are regular expressions!

Definitions cont:

- An unindented comment (~~/*~~) is copied verbatim to output, up to the next ~~*/~~
- Any indented text or text enclosed in:

```
% { ... }%
```

is also copied verbatim (with

```
% { ... }%
```

 removed)
- `% top` makes sure things are copied to top of output (for example, for `#includes`)

② Rules Section

Format: pattern action

where pattern is unindented, + action
is on the same line

Any indented or `%{ %}` can be
used to declare variables, local
to the scanning routine.

(Other things may cause compile issues)

Allowed Patterns

'x' - matches the character x
'.' - any char except newline

'[xyz]' - matches x, y, or z

'[a-j-o-z]' - matches a, b, j, k, l, m, n, o, z

More patterns

not $[\wedge A-z]$ - chars other than A-z
(negation)

$[\wedge A-z \nmid \n]$ - any char except A-z or
a newline

$[a-z]\{-\}[aeiou]$ - any lower case
Consonant

r^* , r^+

Patterns (again)

'r?' 0 or 1 r's

'r{2-5}' Between 2 + 5 r's

'r{2,}' 2 or more r's

'r{4}' exactly 4 r's

variable '{name}' expansion of name definition

'r\$' r at end of a line

⋮
(post webpage)

Precedence:

foo | bar*
is same as (foo) | (bar)*

(since * has higher precedence than concatenation, ~~+~~ concatenation is higher than or)

C classes

`[[:alpha:]]` matches anything that satisfies `isalpha()`.

Ex:

`[[:alnum:]]`

`[[:alpha:][:digit:]]`

`[[:alpha:][0-9]]`

`[a-zA-Z0-9]`

③ User code

Optional, & just copied directly to the output.

(if empty, leave off 1st %%)

Comments

pattern action ~~/*describes*/~~

- C style : ~~/*~~ ~~*/~~

Exceptions

- No comments in the rule section when a regular expression is expected (so not beginning of line or after scanner states)
- Not on % option line of definitions

How it works

- Finds longest pattern match possible
- That match (or token) is made available to a global char pointer `yytext` w/ length `in yyleng`

Then action is performed

- If no match, next char goes to std out.
(So `% %` is valid.)

Actions

Ex.

```
% %  
"zap me"
```

echoes file
back except
specified
phrase

Ex:

```
% %  
[ [t]+  
[ [t]+ $ putchar(' ');  
/* ignore */
```

← stripping
large white
space

strip excess whitespace

Actions (cont)

- If action contains a $\{$, then action spans until next $\}$ (and may go over many lines)
- Action $|$ means "same action as the next rule"
- Can be arbitrary C code, including a return, \downarrow (when run again continues from where \downarrow it left off.)

Special Actions

- ECHO

- BEGIN followed by name of a start condition places scanner in that condition
(more on this later...)

- REJECT tells scanner to go to second best rule

↳ caution: slow

Ex: Count the # of words

pattern to look for:

character $[a-zA-Z]$
digit $[0-9]$

word $(\{\text{character}\} | \{\text{digit}\}) + [^ (\{\text{character}\} | \{\text{digit}\})]$

Ex:

% %

a |
ab |
abc |
abcd |
• / \n

ECHO; REJECT;

Scans: xyz abcd

Output? abcd abc ab a

Conditional Rules

-State based! activated using BEGIN

Define a set of states

- INITIAL is there by default
- Rest defined in %s or %x in first section

Ex: %s STRING

%%
"

<STRING> [^"]* BEGIN (STRING)
{ action; }

%s are inclusive start conditions
%x are exclusive start conditions

After BEGIN, state is active.

If state is inclusive, then rules with no start conditions are still active.

If state is exclusive, then rules with no start conditions are inactive.

Ex: %s versus %x

%s example
%%

<example> foo action();

bar other_action();

Vs:

%x example
%%

<example> foo action();

<INITIAL, example> bar other_action();

Conditions

- `<*>` matches all states
- default rule is in all states.

Essentially, pretend:

```
<*> ./In ECHO;
```

is a line of your file.

Ex: Scanner to ignore C comments
+ count of current input line

%X comment

%%

int num_line = 1; ← local

" /* " BEGIN (comment);

<comment> [^ * \n] *

<comment> " /* " + [^ * \n]

<comment> \n ++ line_num;

<comment> " /* " + " / " BEGIN (INITIAL);

Can condense

$\langle \text{comment} \rangle \{$
all rules

$\}$