
WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

POSIX Regular Expressions

Expression	Matches	Example
c	the one non-operator character c	`a`
\c	character c literally *	`\*`
“s”	string s literally	`“hello”`
.	any character except newline	`a.b`
^	beginning of a line	`^abc`
$	end of a line	`abc$`
[s]	any one of the characters in string s	`[abc]`
[^s]	any one character not in string s	`[^abc]`
r*	≥ 0 occurrences of r	`a*`
r+	≥ 1 occurrences of r	`a+`
r?	0 or 1 occurrences of r	`a?`
r1 r2	r1 followed by r2	`ab`
r1 | r2	r1 or r2	`a | b`
(r)	same as r	`(a|b)`
r1 / r2	r1 when followed by r2	`abc / 123`

Notes on Regular Expressions

Special matching

- Dash within [] for ranges ([A-Z] [a-z] [A-Za-z] [0-9])
- If blank (space) is inside brackets, will match as a character
- Special characters: \t, \n, \\, \”
- \s matches any whitespace character: [\t\n\r\f]

More info and examples:

`http://marvin.cs.uidaho.edu/Handouts/regex.html`

Flex Example 0:

```
/* Just like UNIX wc */
%{
int chars = 0;
int words = 0;
int lines = 0;
%}

%%
[a-zA-Z]+ { words++; chars += strlen(yytext); }
\n        { chars++; lines++; }
.         { chars++; }
%%

main()
{
  yylex();
  printf("%8d%8d%8d\n", lines, words, chars);
}
```

Flex Example 1:

```
    /*** Definition section ***/

%{
/* C code to be copied verbatim */
#include <stdio.h>
%}

/* This tells flex to read only one input file */
%option noyywrap

%%
    /*** Rules section ***/

    /* [0-9]+ matches a string of one or more digits */
[0-9]+  {
            /* yytext is a string containing the matched text. */
            printf("Saw an integer: %s\n", yytext);
        }
.|\n    {   /* Ignore all other characters. */   }
```

Flex Example 1: (cont...)

```
%%
/*** C Code section ***/


int main(void)
{
    /* Call the lexer, then quit. */
    yylex();
    return 0;
}
```

Flex Example 2:

```
%{
#include <iostream>
using namespace std;
#define YY_DECL extern "C" int yylex()
%}

%%
[ \t\n]          ;
[0-9]+\.[0-9]+  { cout << "Found a floating-point number:" << yytext <<
endl; }
[0-9]+          { cout << "Found an integer:" << yytext << endl; }
[a-zA-Z0-9]+    { cout << "Found a string: " << yytext << endl; }

%%
main() {
 // lex through the input:
 yylex();
}
```

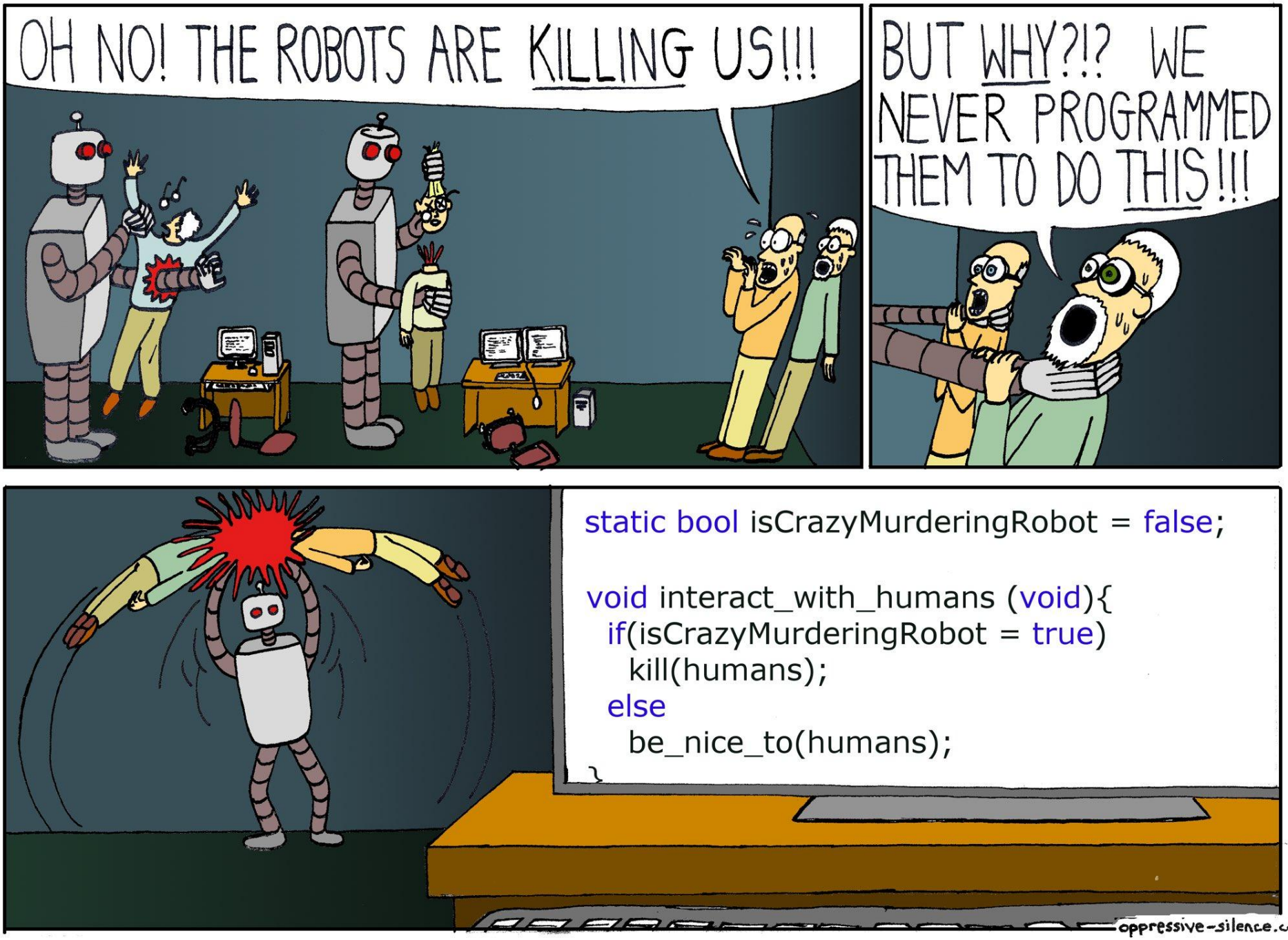
Flex Example 0.1:

```
/*
 * Sample Scanner2:
 * Description: Count the number of characters and the number
 * of lines from standard input
 */

 int num_lines = 0, num_chars = 0;

%%
\n    ++num_lines; ++num_chars;
.     ++num_chars;
%%

main()
{
  yylex();
  printf("# of lines = %d, # of chars = %d\n", num_lines, num_chars);
}
```


OH NO! THE ROBOTS ARE KILLING US!!!
BUT WHY?!? WE NEVER PROGRAMMED THEM TO DO THIS!!!
static bool isCrazyMurderingRobot = false;
void interact_with_humans (void){
if(isCrazyMurderingRobot = true)
kill(humans);
else
be_nice_to(humans);
}
oppressive-silence.com