About TAPE
The Team

Tianhua Fang
Proposal, LRM, scanner, parser, codegen, makefile, demo, tests

Alexander Sato
Proposal, LRM, tests

Priscilla Wang
Proposal, LRM, scanner, parser, codegen, tests, demo, presentation

Edwin Chan
Proposal, LRM, scanner, parser, codegen, makefile, tests, demo, presentation
The goal:

To create a simple language that allows users to easily manipulate files.
Language Structure
Architecture

Source Code → Scanner → Parser → Abstract Syntax Tree

Abstract Syntax Tree → Semantic Analyzer → Codegen → LLVM Code
Scanner

Key Words that are reserved:

“If”, “else”, “while”, “for”, “return”

Reserved names:

“Void”, “string”, “int”, “file”
Parser & ast

```plaintext
stmt:  
  expr SEMI { Expr $1 }  
  RETURN SEMI { Return Noexpr }  
  RETURN expr SEMI { Return $2 }  
  LBRACE stmt_list RBRACE { Block(List.rev $2) }  
  IF LPAREN expr RPAREN stmt ELSE stmt { If($3, $5, $7) }  
  FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt { For($3,$5,$7,$9) }  
  WHILE LPAREN expr RPAREN stmt { While($3,$5,$7) }  

expr:  
  LITERAL { Literal[$1] }  
  STRINGLIT { StringLit[$1] }  
  NEWSTRINGLIT { NewStringLit[$1] }  
  STRINGLIT ASSIGN expr { Assign($1, $3) }  
  CHAR_LITERAL { CharLit[$1] }  
  expr PLUS expr { Binop($1, Plus, $3) }  
  expr MINUS expr { Binop($1, Minus, $3) }  
  expr TIMES expr { Binop($1, Times, $3) }  
  expr EQUAL expr { Binop($1, Equal, $3) }  
  expr UNEQUAL expr { Binop($1, Unequal, $3) }  
  expr LESS expr { Binop($1, Less, $3) }  
  expr GREAT expr { Binop($1, Great, $3) }  
  expr LESSEQ expr { Binop($1, Lesseq, $3) }  
  expr GREATAEQ expr { Binop($1, Greateq, $3) }  
  NOT expr { Unop(Not, $5) }  
  TRUE { BoolLit(true) }  
  FALSE { BoolLit(false) }  
  STRINGLIT LPARAREN actual_opt RPARAREN { Call($1,$3) }  
  STRINGLIT LBRACKET expr RBRACKET { Array($1,$3) }  
  STRINGLIT ASSIGN NEW LBRACKET expr RBRACKET { Init($1,$5) }  
  STRINGLIT LBRACKET expr RBRACKET ASSIGN expr { ArrayAssign($1,$3,$6) }  

expr_opt:  
  / nothing */ { Noexpr }  
  | expr {$1}  
```

```
```
if List.mem 'print' (List.map (fun fd -> fd.fname) functions)
then raise (Failure ("function print may not be defined")) else ();

(*Check for duplicate. 2 functions cannot have same name, therefore also does not allow overload*)
report_duplicate (fun n -> "duplicate function " ^ n)
(List.map (fun fd -> fd.fname) functions);

(* Function declaration for a named function (build in Function)*
(* Use a array to hold the details then throw to the built_in_decls by list.fold *)
let built_in_decls_funcs = [
  { typ = Char; fname = "tolower"; formals = [[Char, "x"] ]; locals = []; body = [] ];
  { typ = Char; fname = "toupper"; formals = [[Char, "x"] ]; locals = []; body = [] ];
  { typ = String; fname = "TAPE"; formals = [[String, "x"]]; locals = []; body = [] ];
  { typ = Void; fname = "print_l"; formals = [[Int, "x" ]]; locals = []; body = [] ];
  { typ = String; fname = "fget"; formals = [[String, "x" ]]; locals = []; body = [] ];
  { typ = String; fname = "open"; formals = [[String, "x" ]]; locals = []; body = [] ];
  { typ = Int; fname = "write"; formals = [[String, "x" ]]; locals = []; body = [] ];
  { typ = Void; fname = "print_c"; formals = [[Char, "x" ]]; locals = []; body = [] ];
  { typ = String; fname = "read"; formals = [[String, "x" ]]; locals = []; body = [] ];
  { typ = String; fname = "find"; formals = [[String, "x" ]]; locals = []; body = [] ];
  { typ = String; fname = "cpy"; formals = [[String, "x" ]]; locals = []; body = [] ];

in

let built_in_decls_names = [ "tolower", "toupper", "TAPE", "print_l", "fget", "open", "write", "print_c", "read", "find", "cpy", "length" ];

in

let built_in_decls = List.fold_right2 (StringMap.add)
built_in_decls_names
built_in_decls_funcs
(StringMap.singleton "print_s"
  [ { typ = Void; fname = "print_s"; formals = [[String, "x" ]]; locals = []; body = [] } ])

in

let function_decls = 
List.fold_left (fun m fd -> StringMap.add fd.fname fd m)
built_in_decls functions
let $\text{ltype of typ}$ = function
  | $\text{A.Int}$ -> $\text{i32\_t}$
  | $\text{A.String}$ -> $\text{ptr\_t}$
  | $\text{A.Void}$ -> $\text{void\_t}$
  | $\text{A.Bool}$ -> $\text{i1\_t}$
  | $\text{A.Char}$ -> $\text{i8\_t}$

(*There may have more things need to be put*)
(*declare external function printf*)
let printf_t = L.var_arg_function_type i32_t [[L.pointer_type i8_t []]] in
let printf_func = L.declare_function "printf" printf_t the_module in

let prints_t = L.var_arg_function_type ptr_t [[L.pointer_type i8_t[]]] in
let prints_func = L.declare_function "puts" prints_t the_module in

(*file open and close*)
let open_file_t = L.function_type ptr_t [[L.pointer_type i8_t; L.pointer_type]] in
let open_file_func = L.declare_function "fopen" open_file_t the_module in

let close_file_t = L.function_type i32_t [[i32_t []]] in
let close_file_func = L.declare_function "fclose" close_file_t the_module in

let write_t = L.function_type i32_t [[i32_t; ptr_t []]] in
let write_func = L.declare_function "puts" write_t the_module in

let get_t = L.function_type ptr_t [[ptr_t; i32_t; ptr_t[]]] in
let get_func = L.declare_function "fgets" get_t the_module in

let fwrite_t = L.function_type i32_t [[ptr_t; i32_t; i32_t; ptr_t[]]] in
let fwrite_func = L.declare_function "fwrite" fwrite_t the_module in

let read_t = L.function_type i32_t [[ptr_t; i32_t; i32_t; ptr_t[]]] in
let read_func = L.declare_function "fread" read_t the_module in

let toupper_t = L.function_type i8_t [[i8_t []]] in
let toupper_func = L.declare_function "toupper" toupper_t the_module in

let tolower_t = L.function_type i8_t [[i8_t []]] in
let tolower_func = L.declare_function "tolower" tolower_t the_module in

let calloc_t = L.function_type ptr_t [[i32_t; i32_t]] in
let calloc_func = L.declare_function "calloc" calloc_t the_module in

let strchr_t = L.function_type ptr_t [[ptr_t; ptr_t[]]] in
let strchr_func = L.declare_function "strstr" strchr_t the_module in

let memcpy_t = L.function_type ptr_t [[ptr_t; ptr_t; i32_t[]]] in
let memcpy_func = L.declare_function "memcpy" memcpy_t the_module in

let strlen_t = L.function_type i32_t [[ptr_t[]]] in
Testing
Script

generatedfiles="$generatedfiles ${basename}.ll ${basename}.out" &&
Run "$TAPE" "<" "$1" "" "${basename}.ll" &&
Run "$LLI" "${basename}.ll" "">" "${basename}.out" &&
Compare ${basename}.out ${reffile}.out ${basename}.diff

Substring example

```c
1 int main()
2 {
3     string a;
4     string b;
5
6     a = "I LoVe TAPE";
7     b = substring(0,1000,a);
8
9     print_s(b);
10   return 0;
11 }
```

```c
1 int main()
2 {
3     string a;
4     string b;
5
6     a = "I LoVe TAPE";
7     b = substring(100,1000,a);
8
9     print_s(b);
10   return 0;
11 }
```
Library
Stdlib

Int countWord(string a, string f)

Int tape(string fn, string re)

indexOf(string t, char c)

String substring(int begin, int end, string s)

String str2Upper(string a)

String str2Lower(string a)

String mergeString(string a, string b)

String appendChar(string s, char a)

Int findreplace(string a, string b, string orig, string dest)
Demo
Demo 1: Count, Find & Replace

Goal:

1) Find and print the number of “apple”.
2) Replace “an apple” with “Professor Edwards”.

originalFile.txt

Today is December 14th. I ate an apple for breakfast. Then, at 1pm, I ate another apple. I also ate an apple after dinner.

destinationFile.txt

Today is December 14th. I ate Professor Edwards for breakfast. Then, at 1pm, I ate another apple. I also ate Professor Edwards after dinner.
Demo 2: PLT Grading Example

Studentinfo.csv gives us the student's uni, name, and graduating year.

Studentgrades.csv shows the uni and the grades for each assignment.
Goal:

Our goal is to create a file that has all of the students’ unis and grades.

We do this by “taping” the files together with tape.
Demo 3: Log Analysis

Find string with regular expressions
- List all visitor’s ip
  - Between “/\-/\”: => “/[/\-]/”
- Find pagetyle with click event or pv event
  - And case: /cl.gif/\&/game/
- Find the log for a certain pagetyle
  - OR cases: /movie/\|/manga/
- Find query with star or starwars
  - ? Cases: /star/?/wars/
- Find log that cannot have two words at the same time
  - XOR Cases: /movie/^/starwars/
- Kleene closure
  - * : /fo/*/d/

Typical nginx pingback log

[ip] [time] [Request][content pv/cl][link] [user agent]

Example:

[124.119.30.77] [23/Jun/2016:12:00:00
+0800:1466654400.023] "GET
/pv.gif?uigs_productid=index&uigs_uuid=5a84d33a-da2f-42d1-8ac5-e07&uigs_t=1466654412805&pagetype=rightv
r861&fQuery=%E6%90%9C%E7%8B%A7%88&
sub_pagetyle=webgame&- HTTP/1.0"
"https://www.sogou.com/\&query=%e6%90%9c%e7&ie=ut
f8" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/38.0.2125.122
Safari/537.36 SE 2.X MetaSr 1.0"
Lessons Learned

- Talk to people who have done this before!
  - TA has some insightful comments
  - Ask for help
- Start testing early
- Communication is important
  - Constantly update each other on the work in progress
Moving Forward

- Include stdlib
  - Encountered Problems
  - Move the lib
- Support Bash command