

いつから  
「関数を呼び出したら、  
いつかは戻ってくる」  
と錯覚していた？

@zick\_minoh

# あんた誰？

- [@zick\\_minoh](#) / zick (実家箕面)
- 大学院生 (M2)
- Lisp / Schemeが好き
  - リリカルLisp (ノベルゲーム用のスクリプト)
  - ニコ動でLisp (ニコニコ動画の謎スクリプト)
  - Pocket StationでLisp (ARM7 / C言語)
  - プリンタでLisp (PostScript)
  - その他 JavaScript, Prolog, PHPなどで



# Henry G. Bakerさんの **Cheney on the M.T.A.** 論文の話

## Scheme(など)から**C言語**への トランスレータの作り方

# Charlie on the M.T.A.

*Oh, will he ever return?  
No, he'll never return,*

# 1 + 2を計算して表示するプログラム

```
int f(int x, int y) {  
    return x + y;  
}  
  
int main() {  
    printf("%d", f(1, 2));  
    return 0;  
}
```

# call と return を考える

- call
  - 引数や戻り先をスタックなどに置いてジャンプ
  - `f(1, 2); /* プログラマは戻り先は意識しない */`
- return
  - スタックなどから戻り先を取り出す
  - スタックなどを縮めてジャンプ
  - `return x + y; /* やはり戻り先は意識しない */`

# 戻り先を意識する

```
void f(void (*ret)(int),
      int x, int y) {
    ret(x + y);
}
void g(int x) {
    printf("%d\n", x);
}
int main() {
    f(g, 1, 2);
    return 0;
}
```

# CPSとは

- CPSとは
  - 次にやること(= **continuation**)を
  - 引数として渡す(= **passing**)
  - やり方(= **style**)

# なぜCPS

- Schemeでは
- 諸事情により
- 全部の関数をCPSにしたい！
  - 詳細はこの会場の人たちの興味から外れてそうなので割愛

# C言語でCPS変換をすると

## スタックがあふれて死ぬ

C言語でCPS変換ができないと

Schemeのコンパイラを  
作るのが面倒になる

それでもC言語でCPS変換したい

- スタックが伸びてきたら、
- 今いる関数とその引数を保存して
- 必要なものをヒープにコピーして
- `longjmp`でスタックを縮めてから
- 保存した情報を使って再開

# 实例

```
void f(void (*ret)(int),
       struct object* obj) {
    if (stack_check()) {
        struct resume *r = malloc(...);
        r->fn = f; r->num_arg = 2;
        r->vars[0]=ret; r->vars[1]=obj;
        copy_to_heap(r);
        longjmp(entry_point, r);
    }
    ret(next_func, obj->field);
}
```

# スタックからヒープへのコピー

- 一般的なごみ集めの手法で作れる
- **Cheneyのcopying GC**
  - 詳細はこの会場の人たちの興味から外れてそうなので割愛

# 続きはWebで

- 詳細は元ネタの論文を読んでください
  - <http://www.pipeline.com/~hbaker1/CheneyMTA.html>