I117（10）テキストファイル（その3）

知念

北陸先端科学技術大学院大学 情報科学研究科
School of Information Science,
Japan Advanced Institute of Science and Technology

文字列頻度計上

ファイル中の文字列を登場頻度を数える
以下のようなファイルを対象とする。

```
Aikawa
Aikawa
Aoyama
Asaka
Asaba
```

先日登場した辞書プログラムを応用できる
メモリ上の配列ではなくファイル内容を格納する

文字列頻度計上 *(cont.)*

辞書領域確保

```
#define DICTLEN     (10)
dict_t *dict;
int    dictlen, dictuse;
...
dictlen = DICTLEN;
dict = (dict_t*)malloc(
        sizeof(dict_t)*dictlen);
dictuse = 0;
```

必要数が分からないので、ひとまず固定

読み込んだ行をそのまま格納、頻度計上

```
while(fgets(line, BUFSIZ, stdin)) {
  ref.value = line;
  ppos = bsearch(&ref, dict, dictuse,
                 sizeof(dict[0]), dictcmp);
  if(!ppos) { /* not found */
    dict[dictuse].value = strdup(ref.value);
    dict[dictuse].count = 1;  dictuse++;
    qsort(dict, dictuse, sizeof(dict[0]),
        dictcmp);
  }
  else {  ppos->count++; }
}
```

```
for(i=0;i<dictuse;i++){
    printf("%2d %d %s\n", i, dict[i].count,
        dict[i].value);
}
```

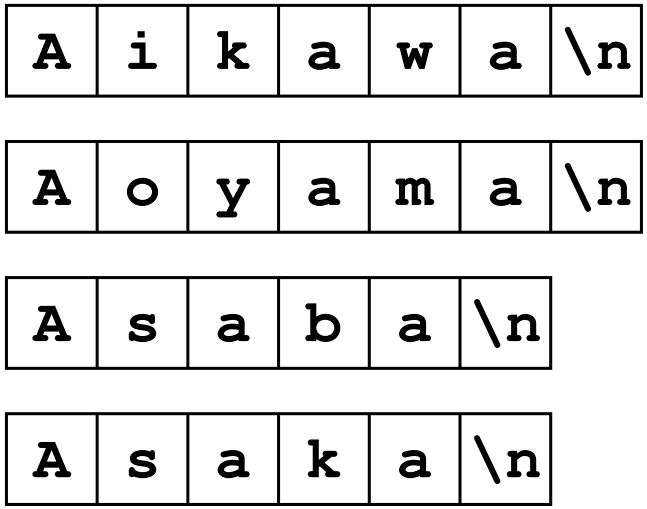
結果 — 行区切りが含まれている

```
 0 2 Aikawa

 1 1 Aoyama

 2 1 Asaba

 3 1 Asaka
```



文字列頻度計上 *(cont.)*

大筋動作する

行区切りを除去

- 読み込む時点で行区切りを除去する
  区切り文字を '\0' にする関数を用意

```
int chomp(char *src) {
    char *p=src;
    while(*p) {
        if(*p=='\n'||*p=='\r') {
            *p = '\0';  return 0;  }
        p++;
    }
    return 0;  }
```

行区切りを除去 *(cont.)*

行を読み込んだ後にこの関数を呼び出す

```
while(fgets(line, BUFSIZ, stdin)) {
  chomp(line);
  ref.value = line;
```

結果 — 行区切りがなくなる

```
 0 2 Aikawa
 1 1 Aoyama
 2 1 Asaba
 3 1 Asaka
```

格納容量の調整 – マクロの利用

固定では使いづらいので調整する方法を紹介する
未定義ならプログラム内で定義

```
#ifndef DICTLEN
#define DICTLEN     (10)
#endif
```

足りなくなるとエラーを表示

```
if(!ppos) { if(dictuse>=dictlen) {
    fprintf(stderr, "no more memory\n");
    continue;  }
```

格納容量の調整 – マクロの利用 *(cont.)*

コンパイル時に特定の数を与えて調整可能

```
% cc -DDICTLEN=4 wfc01c.c
% cat x0 | ./a.out
 0 2 Aikawa
 1 1 Aoyama
 2 1 Asaba
 3 1 Asaka
```

```
% cc -DDICTLEN=3 wfc01c.c
% cat x0 | ./a.out
no more memory
 0 2 Aikawa
 1 1 Aoyama
 2 1 Asaka
```

DICTLEN が 3 以下になると容量が足りなくなる
容量を変更できていることが確認できた

格納容量の調整 – 不足時に再確保

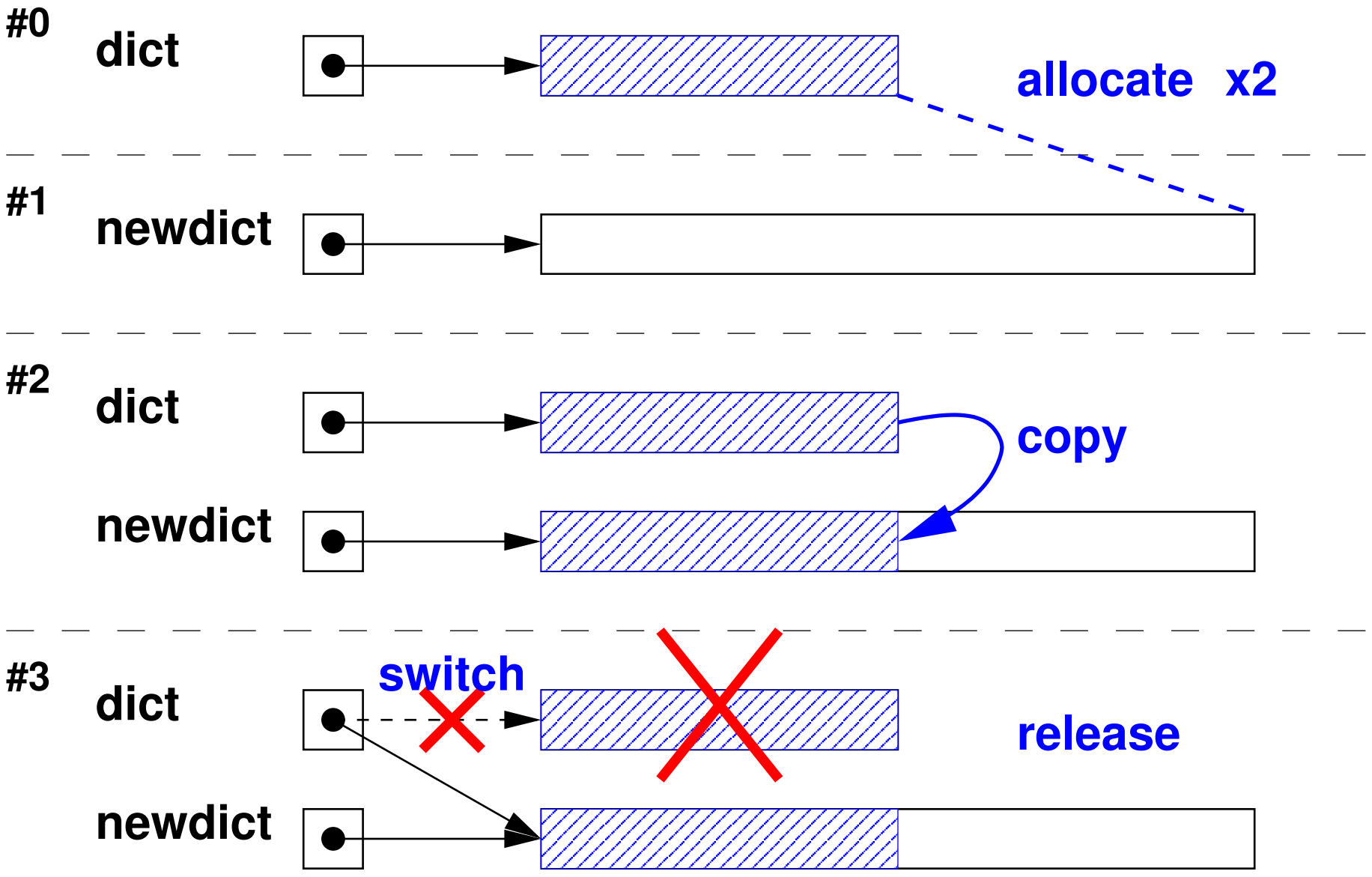
領域拡大（2倍）の関数を用意

```
int expand() {
  dict_t *newdict;  int newlen, i;
  newlen = dictlen*2;
  newdict = (dict_t*)malloc(
                sizeof(dict_t)*newlen);
  for(i=0;i<dictuse;i++){ newdict[i] = dict[i]; }
  free(dict);   dict = newdict;
  fprintf(stderr, "#expand %d -> %d\n",
      dictlen, newlen);
  dictlen = newlen;  }
```

格納容量の調整 – 不足時に再確保 *(cont.)*

- 新しいサイズ newlen を dictlen の2倍とする
- 新しい領域 newdict を malloc で確保
- newdict に dict の内容をコピー
- dict を開放して、newdict を代入する
- newdict に newlen を代入する

※ 経過が分かるように stderr に長さの変化を出力


#0
dict
allocate x2
#1
newdict
#2
dict
copy
newdict
#3
dict
switch
release
newdict

格納容量の調整 – 不足時に再確保 *(cont.)*

領域拡大関数を不足時呼び出す

```
if(!ppos) { /* not found */
    if(dictuse>=dictlen) {
        expand();
    }
```

こうすると、可能な限り格納する
上限は OS やシェルなどの設定による

格納容量の調整 – 不足時に再確保 *(cont.)*

```
% cc -DDICTLEN=3 wfc01d.c
% cat x0 | ./a.out
#expand 3 -> 6
 0 2 Aikawa
 1 1 Aoyama
 2 1 Asaba
 3 1 Asaka
```

```
% cc -DDICTLEN=2 wfc01d.c
% cat x0 | ./a.out
#expand 2 -> 4
 0 2 Aikawa
 1 1 Aoyama
 2 1 Asaba
 3 1 Asaka
```

長さの初期値が 3 や 2 でも拡張して格納している

出力の工夫 – 頻度順整列

データが多いと、頻度順の出力が欲しくなる

```
...
185 1 text
186 1 text.
187 11 the
188 2 these
190 3 to
191 1 to,
...
```

出力の工夫 – 頻度順整列 *(cont.)*

頻度の比較関数を書いて qsort を適用

```
int freqcmp(const void *ap,
        const void *bp) {
    int a, b;
    a = ((dict_t*)ap)->count;
    b = ((dict_t*)bp)->count;
    return a-b;  }
...
qsort(dict, dictuse,
    sizeof(dict[0]), freqcmp);
```

出力の工夫 – 頻度順整列 *(cont.)*

結果 — 昇順、最後の 177件は空行

```
...(略)
197 4 |
198 6 If
199 7 man
200 7 SunOS
201 7 User
202 11 The
203 11 the
204 177
```

演習

1) 実際に文字列頻度を調査せよ★

この講義のホームページに LIST01 というファイルをおいた

URL は以下のとおり

http://www2.jaist.ac.jp/is/private/lecture/i117/2008/LIST01

文字列頻度を計上するプログラムを作成し、このファイル中の文字列とその頻度の一覧を作れ

演習 *(cont.)*

WWW 上のデータは wget で取得できる

```
% wget http://www2.jaist.ac.jp/is/private/lecture/i117/2008/LIST01
--16:25:19--  http://www2.jaist.ac.jp/is/private/lecture/i117/2008/LIST01
           => ‘LIST01’
Resolving www2.jaist.ac.jp... 150.65.38.80
Connecting to www2.jaist.ac.jp|150.65.38.80|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 624 [text/plain]

100%[====================================>] 624           --.--K/s

16:25:20 (11.12 MB/s) - ‘LIST01’ saved [624/624]
```

WWW ブラウザでも取得可能

演習 *(cont.)*

2) 文字列頻度でコメントや空行を無視するようプログラムを変更せよ★

3) 文字列頻度で頻度で整列し降順に出力するようプログラムを変更せよ★

4) 実際に文字列頻度を調査せよ★★

   先と同様にファイル LIST02 を用意した

   そのファイルに対して以下の処理のプログラムとその結果を確認せよ

   a) 昇順(頻度の多い順)に文字列を表示、頻度の多い上位 5つ

演習 *(cont.)*

b) 降順(頻度の少ない順)に文字列を表示、頻度の少ない上位 5つ

ただし、以下の条件をつける

- 先頭に # が付いている行はコメントとして除く

5) 先の問題を C 言語以外のプログラムで解く方法を見付けよ★

- スクリプト言語でもよい
- データベースに格納して問い合わせ言語で解いてもよい