I117（19）テンプレート処理

知念

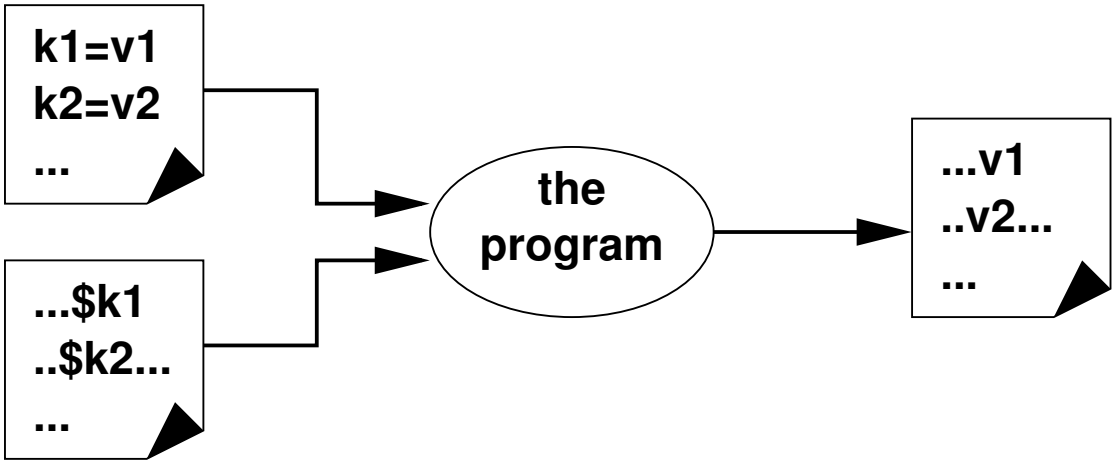
北陸先端科学技術大学院大学 情報科学研究科
School of Information Science,
Japan Advanced Institute of Science and Technology

テンプレート処理

一部だけ異なるなる文章を大量に生成するプログラムを作成する

- 一斉発送データの宛名

ルートと種のファイルから新しいファイルを作る



ルールファイル

変数と値を記録

```
変数名1=値1
変数名2=値2
```

```
% cat rule-a
name=taro
family=hokuriku
reqno=1329
```

種ファイル

変数を使って記述したテキストファイル
変数は $ で表現する

```
% cat seed-a
Dear $family $name,

Your request(#$reqno) is accepted.
```

期待する実行結果

```
Dear hokuriku taro,

Your request(#1329) is accepted.
```

プログラム構成

1) ルールファイル内容を辞書に蓄える
   - 変数名、値を取り出す
   - sdict_add で蓄える
2) 種ファイル内容を変換する
   - 行中の $ を走査
   - 変数名を取り出す
   - sdict_findpos で登録されているか調べる
   - 辞書内容で置換

ルール読み込み

```
int readrule(sdict_a *rule, char *rfname) {
  FILE *fp;  char *p, *q;  int ck, c;
  char line[BUFSIZ],xname[BUFSIZ],xvalue[BUFSIZ];
  fp = fopen(rfname, "r");
  if(!fp) {
    fprintf(stderr,
      "cannot open rule-file ’%s’\n", rfname);
    return -1;
  }
  while(fgets(line, BUFSIZ, fp)) {
    chomp(line);
    p = line;
```

```
    q = xname; c = 0;
    while(*p && *p!='=' && c<BUFSIZ-1) {
        *q++ = *p++;  c++;  }
    *q = '\0';
    if(*p=='=') { p++;  }
    else {  continue;  }
    q = xvalue; c = 0;
    while(*p && c<BUFSIZ-1) {
        *q++ = *p++;  c++;  }
    *q = '\0';
    ck = sdict_add(rule, xname, xvalue);
  }
  fclose(fp);
}
```

文字列変換

```
int filltemplate(FILE *ofp,
         sdict_a *rule, char *sfname) {
  FILE *fp;
  char iline[BUFSIZ],oline[BUFSIZ],xname[BUFSIZ];
  char *p, *q, *t, *u;  int c, m;
  sdict_c *pos;

  fp = fopen(sfname, "r");
  if(!fp) {
      fprintf(stderr, "cannot open seed-file ’%s’\n",
      return -1;
  }
```

```
while(fgets(iline, BUFSIZ, fp)) {
  chomp(iline);
  p = iline;
  q = oline;  c = 0;
  while(*p && c<BUFSIZ-1) {
    if(*p==’$’) {  p++;
      if(!*p) { break;  }
      t = xname; m = 0;
      while(*p && (*p>=’a’ && *p<=’z’)
          && m<BUFSIZ-1) {
        *t++ = *p++; m++;
      }
      *t = ’\0’;
```

```
        pos = sdict_findpos(rule, xname);
        if(!pos) {
          *q++ = '*'; c++;  *q++ = 'E'; c++;
          *q++ = 'R'; c++;  *q++ = 'R'; c++;
          *q++ = '*'; c++;
        }
        else {
          u = pos->value;
          while(*u && c<BUFSIZ-1) {
            *q++ = *u++;  c++;
          }
        }
      }
```

```
      else {
        *q++ = *p++;  c++;
      }
    }
    *q = '\0';
    fprintf(ofp, "%s\n", oline);
  }
  fclose(fp);
}
```

メイン関数

```
int main(int argc, char *argv[]) {
  char *rulefname=NULL, *seedfname=NULL;
  sdict_a *dict;  int ck;

  if(argc<3) {
    fprintf(stdout,
      "usage: %s <rule> <seed>\n", argv[0]);
    exit(1);  }
  rulefname = argv[1];  seedfname = argv[2];
  dict = sdict_new();  ck = sdict_init(dict);
  readrule(dict, rulefname);
  filltemplate(stdout, dict, seedfname);
}
```

プログラム構成 *(cont.)*

コンパイル

```
% cc -g ftemp.c libsdict.c
ftemp.c:
libsdict.c:
```

実行結果

通常動作

```
% ./a.out rule-a seed-a
Dear hokuriku taro,

Your request(#1329) is accepted.
```

未定義変数の動作

```
% cat seed-b
test $family $name and $dmy
% ./a.out rule-a seed-b
test hokuriku taro and *ERR*
```

変数名エスケープ

現状では変数名の直後に文字を続けることができない
そこで、${name} という表記を追加する

```
% cat seed-c
Dear $family ${name}maru,
% ./a.out rule-a seed-c
Dear hokuriku taromaru,
```

変数名エスケープ *(cont.)*

変数名切りだし箇所で { を見つけたら条件分岐

```
if(*p=='{') {  p++;
  t = xname; m = 0;
  while(*p && (*p!='}') && m<BUFSIZ-1) {
    *t++ = *p++; m++;
  }
  if(*p=='}') { p++; }
  *t = '\0';
} else {
  これまでと同じ
```

文字列操作整理

- 繁雑なポインタ操作を簡略化
- 定型処理を関数化
  以下の組合せを考える
  - 区切り文字で文字列切り出し
  - コピー
  - 長さ制限付き
  - 操作後のコピー元／先を戻り値に

文字列操作整理 *(cont.)*

分かりやすさを優先して、すこしくどい名前に

名前	戻り値	摘要
takealphastrS	S	アルファベット列を切り出す
takestrtilsepS	S	区切り文字まで切り出す
cpytermstrS	S	コピーして終了端を加える
cpytermstrD	D	コピーして終了端を加える

引数は、コピー元、コピー先、コピー先長さ、区切り文字の順

変数が減って分かりやすい
下請け関数に隠れているだけでメモリ消費は減らない

```
int readrule(sdict_a *rule, char *rfname) {
  FILE *fp;  char *p;  int   ck;
  char line[BUFSIZ],xname[BUFSIZ],xvalue[BUFSIZ];

  fp = fopen(rfname, "r");
  if(!fp) {
    fprintf(stderr,
      "cannot open rule-file ’%s’\n", rfname);
    return -1;
  }
```

```
  while(fgets(line, BUFSIZ, fp)) {
    chomp(line);
    p = line;
    p = takestrtilsepS(p, xname, BUFSIZ, '=');
    if(*p!='=') {  continue;  }
    p++;
    p = cpytermstrS(p, xvalue, BUFSIZ);
    ck = sdict_add(rule, xname, xvalue);
  }
  fclose(fp);
}
```

簡潔にまとまる

```
int filltemplate(FILE *ofp, sdict_a *rule,
        char *sfname){
  char iline[BUFSIZ],oline[BUFSIZ],xname[BUFSIZ];
  FILE *fp;  char *p, *q;  int c;  sdict_c *pos;
  fp = fopen(sfname, "r");
  if(!fp) {
    fprintf(stderr,
      "cannot open seed-file '%s'\n", sfname);
    return -1;
  }
  while(fgets(iline, BUFSIZ, fp)) {
    chomp(iline);
    p = iline;  q = oline;  c = 0;
```

```
    while(*p && c<BUFSIZ-1) {
      if(*p==’$’) {  p++;  if(!*p) {  break;  }
        if(*p==’{’) {  p++;
          p = takestrtilsepS(p,xname,BUFSIZ,’}’);
          if(*p!=’}’) {
            fprintf(stderr, "not found }\n");
            goto nextline;  }
          p++;  }
        else {
          p = takealphastrS(p, xname, BUFSIZ);  }
        pos = sdict_findpos(rule, xname);
        if(!pos){q=cpytermstrD("*ERR*",q,BUFSIZ); }
        else {q=cpytermstrD(pos->value,q,BUFSIZ); }
```

```
      }
      else { *q++ = *p++;  c++;  }
    }
    *q = '\0';
    fprintf(ofp, "%s\n", oline);
nextline:  (void)0;
  }
  fclose(fp);
}
```

- readrule, filltemplate の長さが約 74%に
- 下請けが必要なのでプログラム全体は19%の増加
- 見通しが良くなったので許容範囲

演習

1) 前述のテンプレートを扱うプログラムを作成せよ ★
2) 文字 $ を使う場合に備えて、$ をエスケープする機構を考えて、プログラムを変更せよ★★
3) 変数値の加工

   シェルなどでは ${name:s/.c/.o/} 等と書くと、その場で値の置換ができる

   先のプログラムにこのような置換機能を追加せよ ★★★

演習 *(cont.)*

```
% set name=a.c
% echo ${name:s/.c/.o/}
a.o
```