I117（8）テキストファイル（その1）

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

テキストファイル

- テキストが格納されたファイル
- 非常に汎用的なデータ形式
- データ交換に広く用いられる
- 行: CR (0x0d) や LF (0x0a) で区切られた文字列

行区切り（line separator）文字

行区切り	代表的なプラットホーム
CR	Mac
LF	UNIX
CR+LF	MS-DOS、WINDOWS

行区切りの確認

プログラム od 等で確認できる

```
% ls -1 | head -2
I117-0.aux
I117-0.dvi
% ls | head -2 |od -c
0000000   I   1   1   7   -   0   .   a   u   x  \n   I   1   1
0000020   0   .   d   v   i  \n
0000026
% ls | head -2 | od -t x1
0000000 49 31 31 37 2d 30 2e 61 75 78 0a 49 31 31 37 2d
0000020 30 2e 64 76 69 0a
0000026
```

この環境は LF (\n 0x0a) が行区切り

標準ライブラリ

C 言語でテキストファイル処理

- ファイル処理は OS に強く依存するため、標準ライブラリを使うのが一般的
- オープン fopen
- 読み込み fgets
- 書き出し fputs, fprintf
- クローズ fclose

典型的操作 – 読み込み

```
FILE *fp;  char line[BUFSIZ];
fp = fopen("foo", "r");
if(!fp) {
    エラー処理
}
while(fgets(line, BUFSIZ, fp)) {
    各種処理
}
fclose(fp);
```

典型的操作 – 書き出し

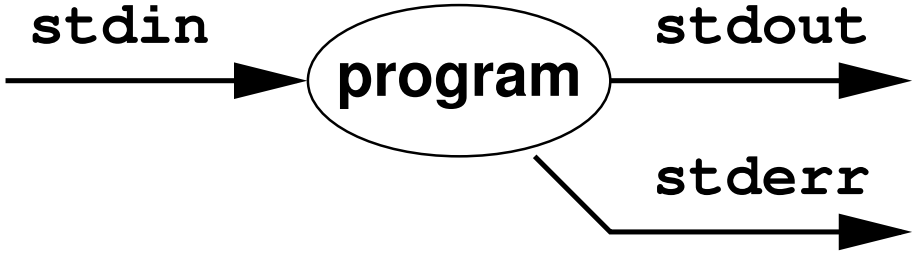
```
fp = fopen("foo", "w");
if(!fp) {
    エラー処理
}
while() {
    各種処理
    fprintf(fp, "%s\n", line);
    あるいは fputs(line, fp);
}
fclose(fp);
```

定義済・オープン済ファイル

標準ライブラリでは

- 標準入力、標準出力、標準エラーが定義済
- プログラム実行時にはオープン済

標準入力 stdin	入力
標準出力 stdout	結果を出力
標準エラー stderr	エラーを出力



定義済・オープン済ファイル *(cont.)*

無指定では標準入力・出力・エラーともに端末に割り当てられている

- キーボードが、標準入力に
- 標準出力とエラーは端末画面に

シェルから呼び出す際にファイルや別プログラムに切替えることが可能

- a.out の入力をファイル lc.c に

```
% ./a.out < lc.c
12 lines
```

定義済・オープン済ファイル *(cont.)*

- プログラム ls の結果を a.out の入力に

```
% ls | ./a.out
```

- a.out の結果をプログラム wc に与える

```
% ./a.out |wc
       1       2       9
```

小さなプログラムではこの 3つのファイルで十分な場面が多い

定数: BUFSIZ

- 標準ライブラリの提供する理論的バッファ長
- 読み込む際のバッファ長として良く利用
  - あくまで目安
  - 行やファイルの長さを保証するわけではない
- 値は処理系依存、典型的には...

```
512, 1024, 4096, 8192
```

- バッファ長は setvbuf で変更可能
  - 触れるのは稀

定数: BUFSIZ *(cont.)*

あるシステムの BUFSIZ を調べる

```
#include <stdio.h>
int main()
{
    printf("BUFSIZ %d\n", BUFSIZ);
}
```

SunOS 5.9 では 1024

```
% ./a.out
BUFSIZ 1024
```

行数計上

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int lno;
    char line[BUFSIZ];
    lno = 0;
    while(fgets(line, BUFSIZ, stdin))
        lno++;
    printf("%d lines\n", lno);
}
```

行数計上 *(cont.)*

実行例: 標準入力から 5行流し込む

```
% ls / | head -5 | ./a.out
5 lines
```

厳密には fgets を呼び出した回数

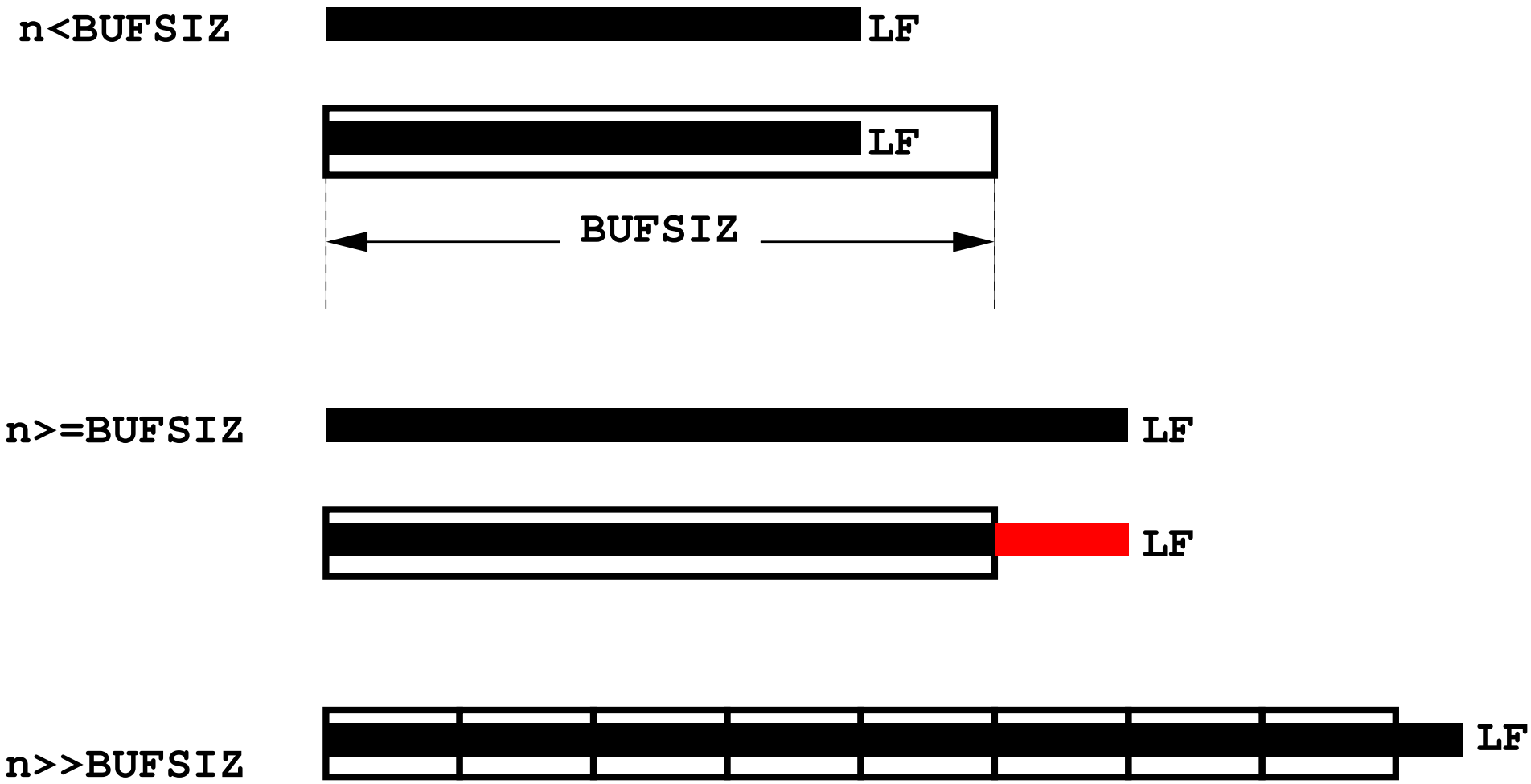
- 長い（BUFSIZを越えた）行を読み込んだ時に差が現れる
- プログラム wc と比較するとわかりやすい

行数計上 *(cont.)*

長い行を含むファイル /tmp/z を作って、wc と先のプログラムに与えてみる

```
% echo ‘man man‘ > /tmp/z
% echo ‘man man‘ >> /tmp/z
% echo ‘man man‘ >> /tmp/z
% cat /tmp/z | wc
       3    5724   38100
% cat /tmp/z | ./a.out
39 lines
```

長い行を含むファイルはエディタで作ってもよい



読み込む行がバッファより長い可能性がある

行数計上 *(cont.)*

line 中の行区切りを数えると wc と同等になる

```
while(fgets(line, BUFSIZ, stdin)) {
    lw = strlen(line);
    if(lw>=1 && line[lw-1]=='\n') {
        lno++;
    }
}
```

```
% cat /tmp/z | ./a.out
3 lines
```

ファイル中の文字列整列

何度も登場した単純な文字列のソート
ただし、今回はデータがソースファイルの外にある

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int Xstrcmp(const void *x,
    const void *y) {
  char *xp=*(char**)x, *yp=*(char**)y;
  return strcmp(xp, yp); }
```

例によって比較関数をおく

```
#define NLINES  (100)
char tmp[BUFSIZ];  char **lines;
int  n, i;
n = 0;
lines = (char**)malloc(sizeof(char*)*NLINES);
while(fgets(tmp, BUFSIZ, stdin)) {
    if(n>NLINES-1) {
        break;
    }
    lines[n] = strdup(tmp);
    n++;
}
qsort(lines, n, sizeof(char*), Xstrcmp);
```

ファイル中の文字列整列 *(cont.)*

実行例

input

```
% cat in.lsort
jaist
asahida
tatukuchi
ishikawa
japan
```

output

```
% ./a.out < in.lsort
0:  asahida
1:  ishikawa
2:  jaist
3:  japan
4:  tatukuchi
```

整列成功

ファイル中の数字の演算

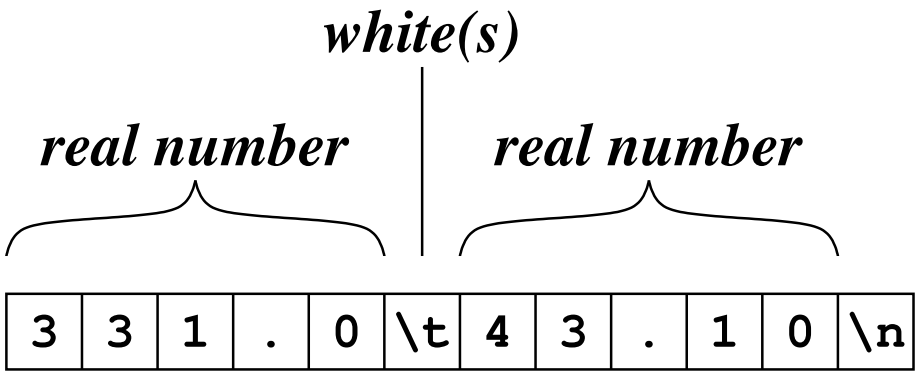
以下のような一行に二つの数字（実数）が入ったファイルが対象

```
331.0    43.10
332.5    39.41
380.6    48.18
```

一行に実数、空白、実数の順で格納されている

- 実数＜0-9とピリオド(0x2e)＞
- 空白＜スペース(0x20)とタブ(\t, 0x09)＞

ファイル中の数字の演算 *(cont.)*




```
while (<行読み込み>) {
    実数取り出し
    空白取り出し(あるいは読み飛ばし)
    実数取り出し
}
```

一緒に入っている実数と空白の処理を同時にする

実数取出マクロ

行(char*p)から実数を取り出す処理をマクロにする

```
#define takereal(r) { \
  char tmp[BUFSIZ]; char *q = tmp;\
  r = -1.0; \
  while((*p>='0'&&*p<='9')||*p=='.'){\
      *q++ = *p++; } \
  *q = '\0'; \
  if(*tmp) r = atof(tmp); \
  while(*p&&(*p==' '||*p=='\t')) p++; }
```

空の場合は -1 を代入する

演算の例

- 行内の演算
  四則演算
  - ◇ 除算
- 行をまたぐ演算
  統計
  - ◇ 平均、分散

行内の演算; 除算

実数取り出しをマクロにしておくと簡潔に書ける

```
char line[BUFSIZ];
char *p;
double a1, a2;
while(fgets(line, BUFSIZ, stdin)) {
     p = line;
     takereal(a1);
     takereal(a2);
     printf("%.2f\n", a1/a2);  }
```

※ マクロでなければ長い記述になっただろう

行内の演算; 除算 *(cont.)*

実は先のデータは自動車の走行距離と所要ガソリン量で、この割算は燃費の計算に相当する

```
% ./a.out < G1
7.68
8.44
7.90
```

最後の値は 7.90 [km/l] を意味する

行をまたぐ演算; 総和

- 総和を格納する変数を設けて、初期化
- 最後に出力

```
double a1, sum;
sum = 0.0;
while(fgets(line, BUFSIZ, stdin)) {
    p = line;
    takereal(a1);
    sum += a1;
}
printf("%.2f\n", sum);
```

行をまたぐ演算; 平均

平均は行数も数えて、総和を割る

```
double sum = 0.0;
int num=0;
while(fgets(line, BUFSIZ, stdin)) {
    p = line;
    takereal(a1);
    sum += a1;  num++;
}
printf("%.2f %d ave %.2f\n",
    sum, num, sum/num);
```

行をまたぐ演算; 結果

総和結果

```
% ./a.out <  G1
1044.10
```

平均結果

```
% ./a.out < G1
1044.10 3 ave 348.03
```

行読み込み工夫

- コメント読み飛ばし
  行頭に# (0x23)がある行をコメントとする

```
p = line;
if(*p=='#') {
    continue;
}
```

- 空行読み飛ばし

```
if(*p=='\n') {
    continue; }
```

実数読み込み工夫

行頭に空白がある場合も多いので、空白読飛しと、実数読込の順序を入れ換える

```
#define takereal(r) { \
  char tmp[BUFSIZ]; char *q = tmp;\
  r = -1.0; \
  while(*p&&(*p==' '||*p=='\t')) p++; \
  while((*p>='0'&&*p<='9')||*p=='.') {\
       *q++ = *p++; } \
  *q = '\0'; \
  if(*tmp) r = atof(tmp); }
```

演習

1) テキストファイルの内容を行番号つきで出力するプログラムを作れ★

   非常に長い行(BUFSIZE より長い)は考えないでよい

2) テキストファイルの内容を行番号つきで出力するプログラムを作れ★★

   非常に長い行を含めて考えよ

3) テキストファイルの各行に含まれている実数の個数を数えるプログラムを作れ★

演習 *(cont.)*

4) 以下のようなファイルで、行中の二つ目の実数の平均を求めるプログラムを作れ★

```
35.59 9.09
0.29 105.78
55.86 76.18
9.62 112.53
5.35 43.43
```

5) 上記 4) のプログラムがコメント行を読み飛ばすよう変更せよ★

演習 *(cont.)*

6) 行から任意の場所の実数を取り出す関数を作れ★
関数の引数と戻り値は以下の形とする

```
int nthreal(double *dst,
            char *src, int n);
```

演習 *(cont.)*

7) 累積頻度分布を集計するプログラムを作れ★★

a) データ x_i を昇順に並べよ

b) 各データ頻度 f_i と合計の比率を求めよ $\frac{f_k}{n}$

c) 各データ毎に頻度の累積を求めよ $\Sigma_{i=1}^{k} f_i$

d) 各データ頻度累積と合計の比率を求めよ $\frac{\Sigma_{i=1}^{k} f_i}{n}$

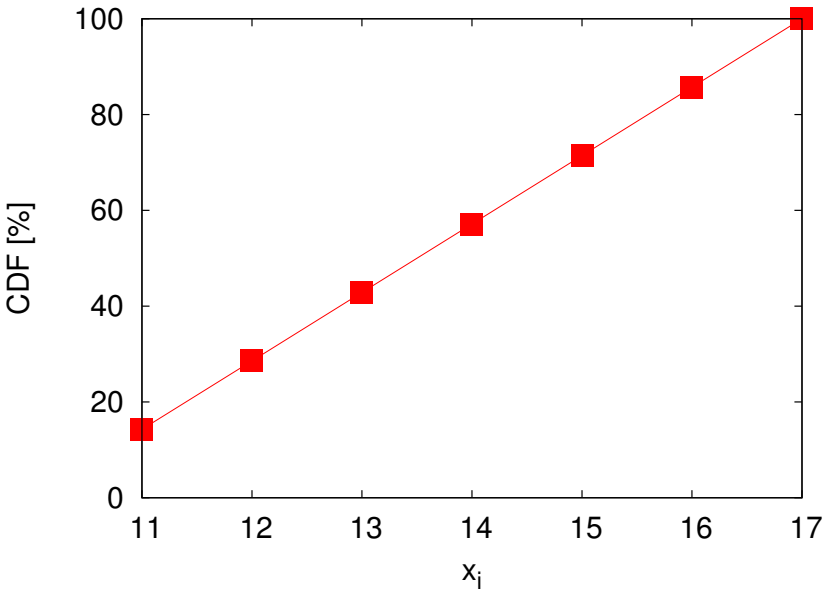
e) データ毎に以下の5項目を一行で出力せよ

$$\left[x_k \ , \ f_k \ , \ \frac{f_k}{n} \ , \ \sum_{i=1}^{k} f_i \ , \ \frac{\Sigma_{i=1}^{k} f_i}{n} \right]$$

※ 比率はパーセントでも良い

w/o dup

11	⇒	11	1	14.29	1	14.29
12		12	1	14.29	2	28.57
13		13	1	14.29	3	42.86
14		14	1	14.29	4	57.14
15		15	1	14.29	5	71.43
16		16	1	14.29	6	85.71
17		17	1	14.29	7	100.00



w/ dup

11					
14					
13					
11		11	2	28.57	2	28.57
14		13	1	14.29	3	42.86
16		14	3	42.86	6	85.71
14	⇒	16	1	14.29	7	100.00