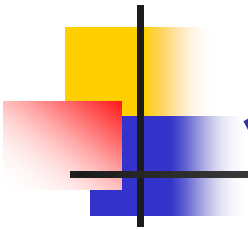
プログラミング及び演習
第6回 ファイル (教科書第9章)
(2017/06/26)

講義担当
大学院情報学研究科知能システム学専攻
教授　森健策

演習担当
大学院情報学研究科知能システム学専攻
小田昌宏

本日の講義・演習の内容



- ファイル　第9章
- 講義・演習ホームページ
  - http://www.newves.org/~mori/17Programming
- ところで,
  - 現在までに教科書 第1-8章を終了
  - 段々難しくなっていると思いますか?
    - 5 簡単
    - 4 ふつう
    - 3 まだまだついてゆける
    - 2 もうそろそろギブアップ!
    - 1 もうだめ

出席と質問

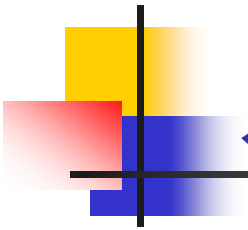
- NUCTにて
  - 回答1
    - 「前ページの回答」
  - 回答2
    - 「先週の演習に一言」
  - 回答3
    - 「学修は順調に進んでいますか？」
- 出席回答は1限中に
- 質問は
  - 17programming @ mori.m.is.nagoya-u.ac.jpへ

UNIXにおける重要概念　ストリーム

- ストリームとは?
  - プログラム・デバイス(ファイル，プリンタ等)との間のデータの流れ
- テキストストリーム
  - 行の集まり
  - 0以上の文字からなり¥nで終わる
  - システムによってはテキストストリームと他の表現形式(CR+LF)との相互変換が必要な場合がある
- バイナリストリーム
  - 内部データを記録するための加工されていないバイトの連続

参考 テープストリーマ

- 磁気テープ装置
- データは1列に書き込まれる
- 磁気テープ操作コマンド mt
  - mt -f /dev/mt0 rewind　　-> 巻き戻し
- QIC-150 - これ一つで150MBytes

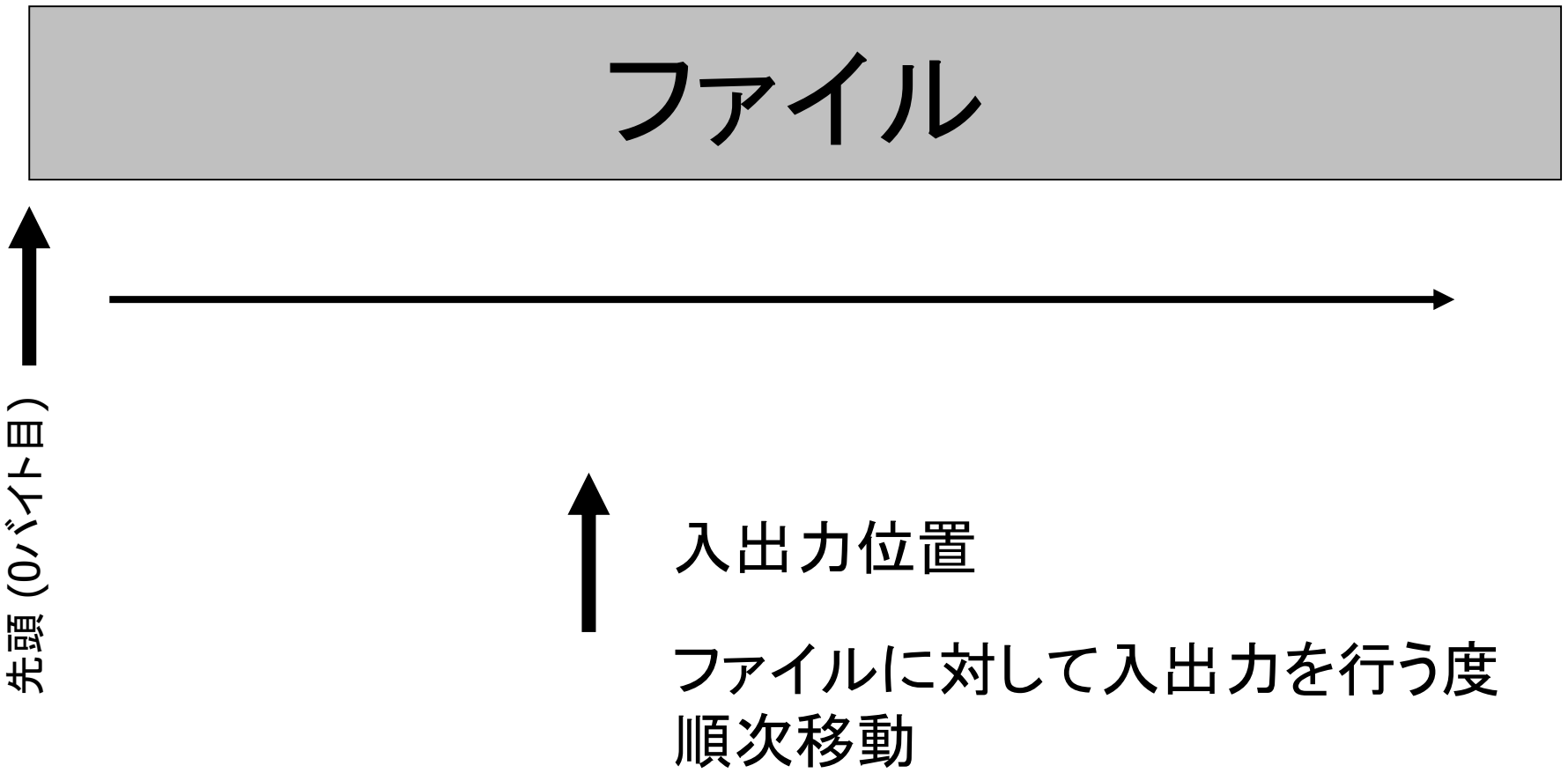


ストリームとファイル・デバイスとの結びつけ

- ストリームはファイル・デバイスと
  - オープンすることで結びつけ
  - クローズすることで結びつきを解消

ファイル内での位置

- ファイルも単なるバイト列の並び



ファイルへの直接読み書き

- ファイルを直接読み書きするには
  - ファイルをopen
  - ファイルからの読みこみ (orファイルへの書き込み)
    - ファイル操作関数を用いる
  - ファイルをclose
- (特に書き込みの場合) ファイルを正しくクローズしないと正しく書き込まれない場合がある

ファイルのopen

- #include <stdio.h>
  FILE *fopen(char filename[],
  char openMode[]);
- fileName[]で指定された名前を持つファイルをopenMode[]で指定されたモードでオープン
- オープンに成功した場合→FILE型構造体へのポインタを返す
- オープンに失敗した場合→NULLポインタを返す

使用例 fopen

- FILE *fp;
  fp = fopen("test.dat", "rt");
- FILE *fp;
  fp = fopen("test.dat", "wb");
- char filename[256];
  strcpy(filename, "/home/mori/test.dat");
  if ((fp=fopen(filename,"rb"))==NULL){
  printf("error¥n");
  return(1);
  }

ファイル名とパスの指定

- 実行ファイルと同じ位置のディレクトリ(カレントディレクトリ)に書き出す場合
  - "test.dat" → "./test.dat"と解釈される
- 相対パスで書き出す場合
  - "../../test.dat"
  - (Windowsでは) "..¥¥..¥¥test.dat" (¥¥で¥を表すことに注意)
- 絶対パスで書き出す場合
  - "/tmp/test.dat"
  - (Windowsでは) "¥¥tmp¥¥test.dat" (¥¥で¥を表すことに注意)

オープンモード

- 各指定子と読み書き開始位置
  - r　　読み込み　　先頭
  - w　　書き込み　　先頭
    ファイルが存在しない場合作成される
  - r+　　読み込み+書き込み　　先頭
  - w+　　読み込み+書き込み　　先頭
    ファイルが存在しない場合作成される
  - a　　追加　　終端
    ファイルが存在しない場合作成される
  - a+　　読み込み+追加　　終端
    ファイルが存在しない場合作成される

オープンモードオプション

- r, r+, w, w+, a, a+の後に続ける
- bとtの2つ
  - b　バイナリモード
    バイナリデータを取り扱うときに明示
  - t　　テキストモード
    UNIX　改行コード LF (0x0a)
    Windows 改行コード CR+LF　(0x0d+0x0a)
    読み込み時に0x0aに統一
    書き込み時にプラットフォームに適した形へ

ファイルのclose

- #include <stdio.h>
  int fclose(FILE *fp);
- fopen()で返されたFILE型ポインタfpが制御しているストリームをcloseする
- 正常にcloseできたときは0
- 失敗したときにはEOFを返す

書式付入力関数

- #include <stdio.h>
  int fscanf(FILE *fp, char format[], …);
  int fprintf(FILE *fp, char format[], …);
- scanf, printfのファイル入出力版
- fpで指し示されるファイルに対して入出力
- fscanf(stdin, …)と書くとscanf(…)に相当
- fprintf(stdout, …)と書くとprintf(…)に相当

1文字単位入出力関数

- int fgetc(FILE *fp);
- int getc(FILE *fP);
  - fp(で指し示されるストリーム)から一文字入力
  - getchar()と同様の動作
- int fputc(int c, FILE *fp);
- int putc(int c, FILE *fp);
  - fpで指し示されるストリームに一文字出力
  - putchar()と同様の動作

文字列単位の入出力関数

- char *fgets(char s [], int num, FILE *fp);
  - num-1文字分もしくは改行文字が検出されるまで，もしくはEOFが検出されるまでfpからsに読み込む
  - 読み込まれた文字列の最後には終端文字が加えられる
- char fputs(char s[], FILE *fp)
  - 文字列をfpに書き出す
  - (終端文字は書き出されない)

直接入出力関数

- 変数・定数の値を文字ではなく内部表現そのままで書き出す関数
- 多バイトデータを書き出すときはデータの量が少なくなるメリット (→例　画像,音)
  - 例) int a = 2354543;
    直接表現　6f ed 23 00　(0x0023ed6f) (=4byte)
    文字書き出し　"2354543"+" " (7byte+1)
- 内部表現は環境により異なるので十分注意
  - ファイルに記録されている内容を正しく読み書き出きない可能性
  - Intel形式とモトローラ形式
  - Big Endean と Little Endean

直接入力関数 (fwrite)

- size_t fwrite(void ptr[], size_t size,
  size_t nitems, FILE *stream);
  - 配列ptrからsizeバイトのバイト列(データ列)を最大nitems個fpから書き出す
  - nitems個書き出したとき，EOFを検出したとき，もしくは，エラーが検出されたとき終了し，書き出された個数を返す
- 使い方
  - #define ARRAYSIZE 100
    FILE *fp;
    int data[ARRAYSIZE];
    ……dataの中に何か値を入れる
    fp=fopen("test.dat","wb');
    fwrite((void *)data, sizeof(int), ARRAYSIZE, fp);
    fclose(fp);

直接入力関数 (fread)

- size_t fread(void ptr[], size_t size,
  size_t nitems, FILE *fp);
  - 配列ptrにsizeバイトのバイト列(データ列)を最低限nitems個fpから読み出す
  - nitems個読み出したとき，EOFを検出したとき，もしくは,エラーが検出されたとき終了し，読み出された個数を返す
- 使い方
  - #define ARRAYSIZE 100
    FILE *fp;
    int data[ARRAYSIZE];
    fp=fopen("test.dat","rb");
    fread((void *)data, sizeof(int), ARRAYSIZE, fp);
    fclose(fp);

fwriteによる書き出し (lec6-fwrite.c)

```
#include <stdio.h>
#define ARRAYSIZE 100
main()
{
  FILE *fp;
  int data[ARRAYSIZE];
  int i;
  fp=fopen("test.dat","wb");
  for(i=0;i<ARRAYSIZE;i++){
    data[i]=i*10;
  }
  fwrite(data, sizeof(int), ARRAYSIZE, fp);
  fclose(fp);
}
```

freadによる読み出し (lec6-fread.c)

```
#include <stdio.h>
#define ARRAYSIZE 100
main()
{
  FILE *fp;
  int data[ARRAYSIZE];
  int i;
  fp=fopen("test.dat","rb");
  fread(data, sizeof(int), ARRAYSIZE, fp);
  for(i=0;i<ARRAYSIZE;i++){
    printf( "data[%d]=%d¥n",i,data[i]);
  }
  fclose(fp);
}
```

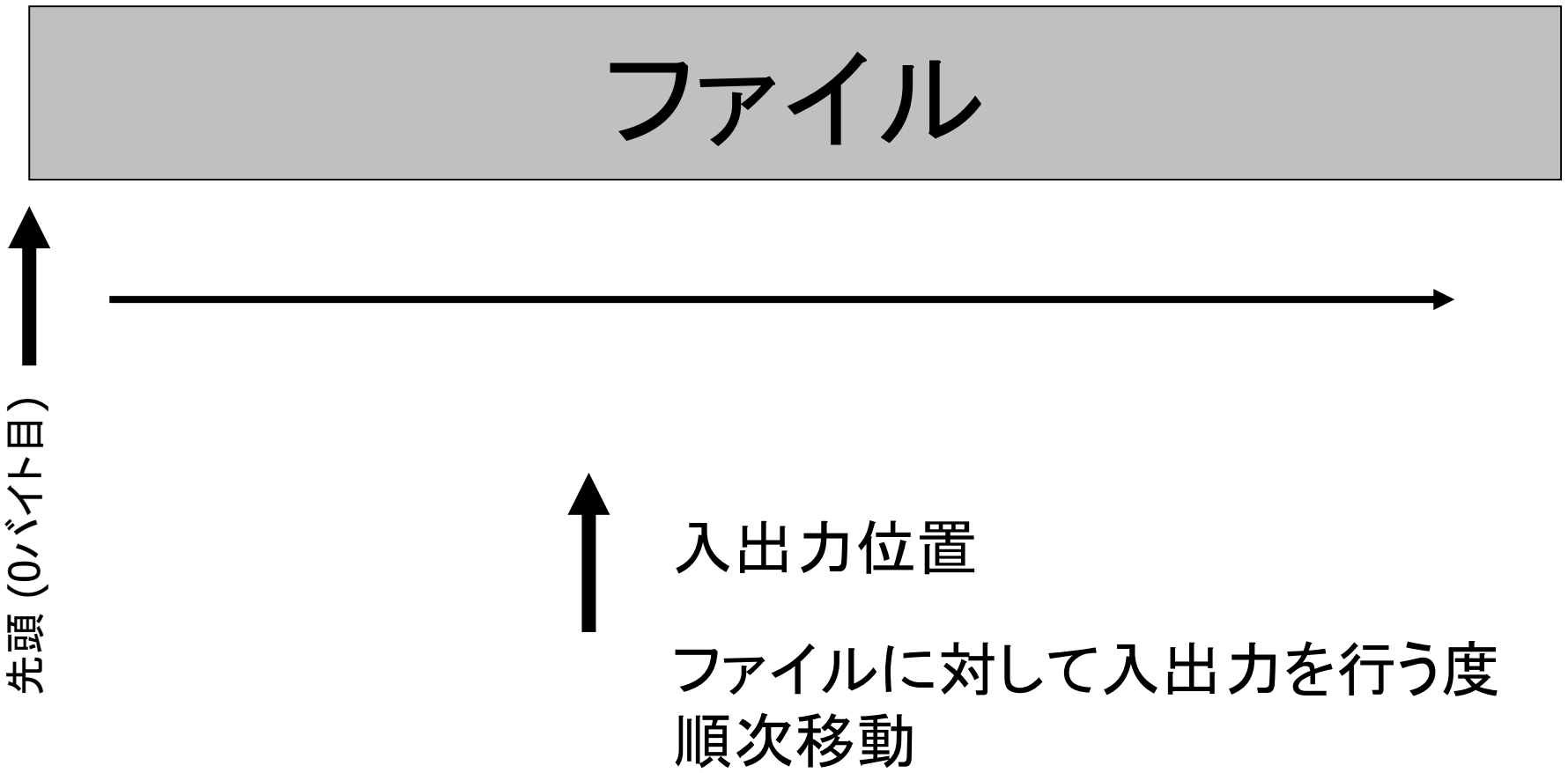
書き出されたファイルの中を覗く

- odコマンドを利用 (od -t x1 test.dat とタイプ)

```
taka{mori}36: od -t x1 test.dat
0000000 00 00 00 00 0a 00 00 00 14 00 00 00 1e 00 00 00
0000020 28 00 00 00 32 00 00 00 3c 00 00 00 46 00 00 00
0000040 50 00 00 00 5a 00 00 00 64 00 00 00 6e 00 00 00
0000060 78 00 00 00 82 00 00 00 8c 00 00 00 96 00 00 00
0000100 a0 00 00 00 aa 00 00 00 b4 00 00 00 be 00 00 00
0000120 c8 00 00 00 d2 00 00 00 dc 00 00 00 e6 00 00 00
0000140 f0 00 00 00 fa 00 00 00 04 01 00 00 0e 01 00 00
0000160 18 01 00 00 22 01 00 00 2c 01 00 00 36 01 00 00
0000200 40 01 00 00 4a 01 00 00 54 01 00 00 5e 01 00 00
0000220 68 01 00 00 72 01 00 00 7c 01 00 00 86 01 00 00
0000240 90 01 00 00 9a 01 00 00 a4 01 00 00 ae 01 00 00
0000260 b8 01 00 00 c2 01 00 00 cc 01 00 00 d6 01 00 00
0000300 e0 01 00 00 ea 01 00 00 f4 01 00 00 fe 01 00 00
0000320 08 02 00 00 12 02 00 00 1c 02 00 00 26 02 00 00
0000340 30 02 00 00 3a 02 00 00 44 02 00 00 4e 02 00 00
.....
```

ファイル内での位置

- ファイルも単なるバイト列の並び



任意の位置に移動するには?

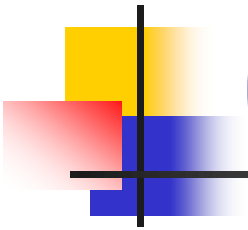
- int fseek(FILE *fp, long offset, int origin);
- fpで示されるストリームでの入出力位置を変更する関数
- どこから(origin)から数えて何バイト目(offset)で設定
  - SEEK_SET
    - 先頭から数えてoffsetバイト目
  - SEEK_CUR
    - 現在位置から数えてoffsetバイト目
  - SEEK_END
    - 終わりから数えてoffsetバイト目
  - SEEK_SET, SEEK_CUR, SEEK_ENDはstdio.hで定義

任意の位置への移動

- ファイルの内容
  - "ABCDEFGHIJKLMNOPQRSTUVWXYZ¥n"
- どんな文字が表示されるか?
  - fseek(fp, 4, SEEK_SET);
    printf("%c¥n", fgetc(fp));
    fseek(fp, 5, SEEK_CUR);
    printf("%c", fgetc(fp));
    fseek(fp, -3, SEEK_CUR);
    printf("%c", fgetc(fp));
    fseek(fp, -10, SEEK_END);
    printf("%c", fgetc(fp));

現在位置は常に次に読み書きすべき場所を表す

読み書き後は現在位置は次の位置に移動



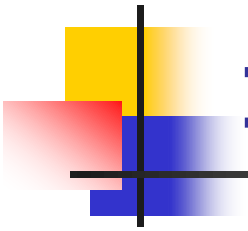
いまはどこ?

- long ftell(FILE *stream);
  - stremにおける現在の入出力位置を返す
  - 例

```
int a;
a=ftell(fp);
printf( "Current file-position indicator %d¥n",a);
```

ファイルの状態を調べる関数

- int feof(FILE *fp);
  - fpで示されたストリームの読み込みが終わりに達しているかを調べる
  - 達している場合　　→ 0以外の値
  - 達していない場合　→ 0
- int ferror(FILE *fp);
  - fpで示されたストリームへの読み書きでエラーが起きているかを調べる
  - 起きている場合　　→ 0以外の値
  - 起きていない場合　→ 0



コマンドライン引数

- コマンド行での引数をプログラムで取り扱う
  - "testprog abc.txt def.txt"
  - "abc.txt", "def.txt"の文字列をプログラムで受け取るには?
- main(int argc, char *argv[])
  - argc 引数の数
  - char *argv[0], char *argv[1]…は引数の文字列を表す. 一番最後にはNULLが入る
  - char *argv[0] は入力コマンド自体を表す

argc, argvの使用例

```
#include <stdio.h>
main(int argc, char *argv[])
{
  int i;
  printf( "argc %d¥n", argc);
  for(i=0;i<argc;i++){
    printf("argv[%d]= %s¥n",
        i, argv[i]);
  }
}
```

```
taka{mori}89: ./a.out fsdf dsf ds
fsd fsdfsdfsf dsf dsf dsaf sa
argc 10
argv[0]= ./a.out
argv[1]= fsdf
argv[2]= dsf
argv[3]= ds
argv[4]= fsd
argv[5]= fsdfsdfsf
argv[6]= dsf
argv[7]= dsf
argv[8]= dsaf
argv[9]= sa
```

argc, argvの使用例

```
#include <stdio.h>
main(int argc, char *argv[])
{
  int i;
  printf( "argc %d¥n", argc);
  for(i=1;i<argc;i++){
    printf("argv[%d]= %d¥n",
        i, atoi(argv[i]));
  }
}
```

```
taka{mori}96: ./a.out 10 20 43
643
argc 5
argv[0]= ./a.out
argv[1]= 10
argv[2]= 20
argv[3]= 43
argv[4]= 643
```

ファイルをコピー (lec6-copy.c)

```
#include <stdio.h>
#include <stdlib.h>
main(int argc, char *argv[])
{
  int c;
  FILE *in, *out;
  if(argc!=3){
    fprintf(stderr, "Illegal command line¥n");
    exit(EXIT_FAILURE);
  }
  if((in=fopen(argv[1],"r"))==NULL){
    fprintf(stderr, "Source file open error¥n");
    exit(EXIT_FAILURE);
  }
  if((out=fopen(argv[2],"w"))==NULL){
    fprintf(stderr, "Destination file open error¥n");
    exit(EXIT_FAILURE);
  }
```

```
  while((c=fgetc(in))!=EOF){
    fputc(c,out);
  }
  fclose(in);
  fclose(out);
}
```