システムプログラミング
第2回課題補足資料

芝浦工業大学 情報工学科
菅谷みどり

演習1: mypower コマンドを開発しよう

- **課題）次の仕様で動作する mypower コマンドを作成する**
  - (1) コマンドの第一引数の数値を x とする
  - (2) y >= 1 の時、累乗 x^y を小数点表記で表示する
  - (3) y < 1 の時、累乗根 $\sqrt[y]{x}$を小数点表記で表示する
  - (4) 第２引数は省略可能で、省略した場合は y = 2.0 とする
  - (5) (1) から (4) の挙動に反する場合には、エラーとする
- **条件**
  - pow(), strtod()などの標準Cライブラリを用いて良い

演習1: mypower コマンドを開発しよう

- **課題）次の仕様で動作する mypower コマンドを作成する**
  - Power コマンドは数値の累乗（るいじょう）を求めます．pow()関数が math ライブラリに既に存在しますが，仕様に制限があります．そのため，より利用しやすい power コマンドを作成することをやってみます．
- **仕様）**
  - 計算は実数mの実数n乗の計算となります．基本的な仕様は，pow()を利用することができます．
    - pow()の仕様を確認しよう $man pow(3)
- **要求仕様**
  - (1) コマンドの第一引数の数値を x とする
  - 第二引数を Y とし，x^y の計算結果を表示する
  - (2) y >= 1 の時、累乗 x^y を小数点表記で表示する
  - (3) y < 1 の時、累乗根 $\sqrt[y]{x}$を小数点表記で表示する
  - (4) 第引数は省略可能で、省略した場合は y = 2.0 とする
  - (5) (1) から (4) の挙動に反する場合には、エラーとする
- **条件**
  - pow(), strtod()などの標準Cライブラリを用いて良い

演習1: mypower コマンドを開発しよう

- **課題）次の仕様で動作する mypower コマンドを作成する**
  - Power コマンドは数値の累乗（るいじょう）を求めます．pow()関数が math ライブラリに既に存在しますが，仕様に制限があります．そのため，より利用しやすい power コマンドを作成することにします．

Power コマンドを考えよう

- **コマンドの必要性の検討**
  - power コマンドは存在しない
    - 確認
      - $which power
      - $find /* -name power
  - pow 関数が math ライブラリにある
    - 2つの引数をとる, double で計算する
  - (1) コマンドの第一引数の数値を x とする
  - (2) 第二引数を Y とし，x^y の計算結果を表示する
    - これが重要

検討方法

- **仕様）**
  - 計算は実数mの実数n乗の計算となります．基本的な仕様は，pow()を利用することができます．
    - pow()の仕様を確認しよう $man pow(3)
- **pow()の使いづらさ（課題）と解決方法の検討**
  - double pow(double x, double y);
    - 関数しかないので，コマンドが存在しない
    - double 型が前提となっている
    - 期待されるのは mypower 3 3
      - 引数両方とも整数値
      - ヒントを参照=> strtod()
  - strtod()
    - strtod 関数は引数 nptr が指す文字列のはじめの数字の部分を double 型の表現に変換します
    - 整数値を，double に変換する関数

仕様案

- **課題にある仕様にそって考える**
  - (1) コマンドの第一引数の数値を x とする
  - 第二引数を Y とし，x^y の計算結果を表示する
  - (2) y >= 1 の時、累乗 x^y を小数点表記で表示する
  - (3) y < 1 の時、累乗根 $\sqrt[y]{x}$を小数点表記で表示する
  - (4) 第引数は省略可能で、省略した場合は y = 2.0 とする
  - (5) (1) から (4) の挙動に反する場合には、エラーとする
- **仕様案をまとめる**
  - (1) 引数を受け取れるようにつくる
  - (2) 整数で受け取り，内部的に strtod()で double に変換.
  - (3) y < 0 の場合も基本的には y乗根 x (pow)で計算する
  - (4) 引数省略処理を追加
  - (5) エラー処理（Usage を表示する）

例

```
int main(int argc, char *argv[]) {
  double x;
  double y;
  char *buf1, *buf2;

  // check number of arguments
  checkArgNum(argc);

  // check 2nd argument's content
  // if 2nd argument's content is NULL,
substitute "2" for argv[2]
  if (check2ndArgContent(argv)) {
    argv[2] = "2";
  }

  // translate char to double
  x = strtod(argv[1], &buf1);
  y = strtod(argv[2], &buf2);

  // if argument is not number
  checkNotNum(buf1, buf2);

  // print result after calculate power
  printf("%lf¥n", powerOfNum(x, y));
}
```

気づいたこと

- **プログラム**
  - 可読性が高い
- **理由**
  - コメントを入れる
  - 関数名を第3者がわかりやすい
  - エラー処理の網羅性
  - メインの処理フローを見ると，何をしてるかわかる
- **README が的確**

演習2：mystrcmp コマンドを開発しよう

- **課題：次の仕様で動作する mystrcmp コマンドを作成する**
- **仕様**
  - (1) コマンドの第 1 引数(s1)、第 2 引数(s2) 2つの文字列を読み込む
  - (2) 文字数が一致しているかをチェックし、結果を出力する
    - 一致の場合には 0,
    - 不一致の場合下記の値を返す
      - s1 < s2 : 0 より小さい値
      - s1 > s2 : 0より大きい値
  - (3) 上記の挙動に反する場合には、エラーとする
- **条件**
  - strcmp() は用いない
- **ヒント**
  - 文字列の比較方法（文字列長に固定的な上限を設けない方が良い)
  - 文字型，文字列型：char と * char, const char, const *char の区別を再確認する
  - 提出前にテストを行おう

仕様を考えよう

- **コマンドの必要性の検討**
  - strcmp コマンドは存在しない
    - Power を例に確認しよう
  - strcmp 関数はある
    - つかうなと言われている
    - 同じことを基本的に行う．また，必要に応じて改善することを考える
- **検討方法に従って考えよう**
  - ヒントを元に考える
- **仕様案を作成する**
  - 引数のチェック
  - メイン処理
  - エラー処理
- **プログラムを改善する**
  - 再提出方法授業中に説明

例

```
int main(int argc, char *argv[]) {
  int len;

  // check number of arguments
  checkArgNum(argc);

  // check if 2nd argument exists
  check2ndArgExist(argv);

  // get string length
  len = getStrLen(argv);

  // compare strings
  compareStr(len, argv);
}
```