# H17年度演習II - 課題3

趙　亮

宇都宮大学工学部情報工学科

2005年12月

## 概要

- 目標：解凍圧縮ソフトの作成により大規模プログラミングの基礎を習う
- トピック：
  - マクロと前処理
  - 分割コンパイル
  - モジュール設計
- 受講の心構え：
  - 説明をよく聞いて
  - メモをとって
  - 自習して
  - 分からないことを遠慮なく質問する．
- 要求：学生一般の自覚

## 予定と参考資料

- 12月8日（今日）：マクロと前処理
- 12月15日：分割コンパイルとモジュール
- 12月22日：例で学ぶモジュール設計
- 参考資料：配布資料とCの解説書（自習）

＊ 時間：12:50--14:20．なお13:35--13:40は目を休む時間とする

# 一 回 目

## 1．マクロ復習1：記号定数

```
#define 記号名 定数
```

```
#define PI 3.14
t = 2 * PI;
s = PI * 2 * 2;
```

↓のコードと同じ効果

```
t = 2 * 3.14;
s = 3.14 * 2 * 2;
```

よくあるミス：余分の等号やセミコロンを入れる

```
#define PI = 3.14  → t = 2 * = 3.14;
#define PI 3.14;   → s = 3.14; * 2 * 2;
```

＊ 確認方法（詳細はHP参照）：cl /EP ファイル名

## 2．マクロ復習2：関数型マクロ

```
#define ABS(x) ((x)<0?-(x):(x))
int a = ABS(-1) + 2;
        ↓
a = ((-1)<0?-(-1):(-1)) + 2;      結果 3
```

悪い定義（エラーか誤動作の可能性あり）

```
#define ABS(x) (x)<0?-(x):(x)
  a = (-1)<0?-(-1):(-1) + 2;      = 1
#define ABS(x) (x<0?-x:x)
  ABS(1-2) = (1-2<0?-1-2:1-2)     = -3
```

＊ マクロ鉄則：個々の引数と定義全体に括弧をつけよう

## 3．マクロ練習

練習：最小値を出す関数型マクロ（15分）

紙で答えと3*MIN(1>2?0:1,3)の展開結果を示そう

```
#define MIN(x,y)
```

＊ 参考：マクロ定義を取り消すには #undef マクロ名

## 4．前処理（プリプロセッサ）

- #で始まる行を処理する．例：
  - #include（他のファイルを読み込む）
  - #define（記号定数やマクロを定義する）
- コンパイルの前に起動される．
- 他のファイルを読み込む，記号定数やマクロを展開する，プログラムのコードを条件付きでコンパイルするといった処理を行なう．

＊ 前処理の対象はCソースでなくてもよい（HP参照）

## 5. 前処理：条件付きコンパイル

- 整数定数を返す式を評価することにより前処理の実行やソースコードのコンパイル処理を制御できる．例：

```
#if !defined(NULL)
#define NULL 0
#endif
```

定数NULLが定義されていなければそれを定義する

* #if !defined(XXX) の短縮形： #ifndef XXX

## 6. 条件付きコンパイルのまとめ

**#if** 式 式を評価し，真の場合次へ
**#ifdef** マクロ名 マクロが定義されているなら次へ
**#ifndef** マクロ名 定義されていないなら次へ
**#else** #if, #ifdef, #ifndefのelse部
**#elif** #if, #ifdef, #ifndefのelse if
**#endif** #if, #ifdef, #ifndefの終了
**defined**(マクロ名) 定義されている場合1，ない場合0

* #if defined(XXX) の短縮形： #ifdef XXX

## 7. 条件付きコンパイルの応用

多重コメント

```
#if 0
  ...
  /* コメント */
  ...
  /* コメント */
  ...
#endif
  ...
```

重複includeの回避

例：stdio.h

```
#ifndef __STDIO_H__
#define __STDIO_H__
  ...
#endif
```

これにより，複数個の
`#include <stdio.h>`
があっても一個のと同じ

* 確かめたいなら，cl /EPで確認しよう（HP参照）

## 8. また応用：基本デバッグ法

- 調べたいデータを出力すること．
- 条件付きコンパイルでさらにうまく働く．

```
/* #define NDEBUG */
...
#ifndef NDEBUG
  printf("(debug) 変数 x = %d\n", x);
#endif
```

製品版ではコメントを外す

デバッグ出力コード

* 汎用性：この方法は一般的に他の言語でも同様に利用できる


## 9. 前処理などの練習（35分）

- echo.c（HP掲載）解読：プログラムが何をやっているのか？
- ヒント：
  - 注釈（コメント）は無視してよい．
  - マクロ__COPYRIGHTと__RCSIDを無視する．
  - コンパイルには，#include <stdio.h>より前の行を注釈するとよい．実行には，生成されたecho.exeのフォルダで .¥echo foo bar ...

* 実用ソフトの中で最もシンプルなものがこれではないかな？

## 10. 宿題

bzlib.h解読：
記号定数と関数のリストを作る（次回使う）

* ヒント：注釈を無視して書かれた順にリストするとよい．

## 11. オプション課題

Cは，73年の誕生以来，K&R C, ANSI C, C89/C90, C95, C99を経て進化してきている．参考：

http://ja.wikipedia.org/wiki/C言語

様々なCがあるため，プログラムでコンパイラがどれに順応しているかを確認することが，移植性や効率や性能などの面で非常に重要である．その確認法を自分で調べ，検証プログラムを組んで実行結果まで示せ．

* ヒント：コンパイラが定義する __STDC__ などを用いる

# H17年度演習II - 課題3

趙　亮

宇都宮大学工学部情報工学科

2005年12月

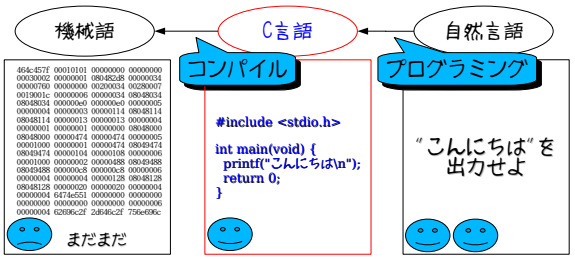
# 二 回 目

## 分割コンパイルとモジュール

## （大規模プログラムの開発に向けて）

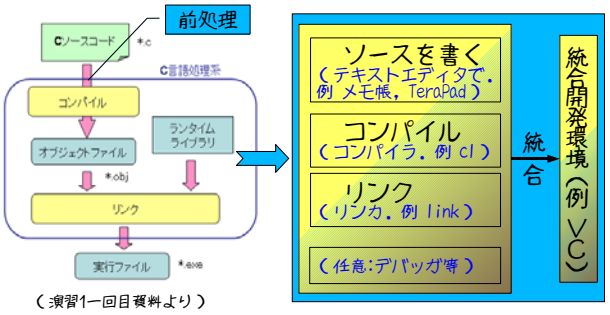
## 1. Cプログラミングとは？

プログラム = 計算機に与える指令の集り




＊ 上達の近道：ソースの解読と繰り返し練習すること．

## 2. Cプログラミング開発の流れ

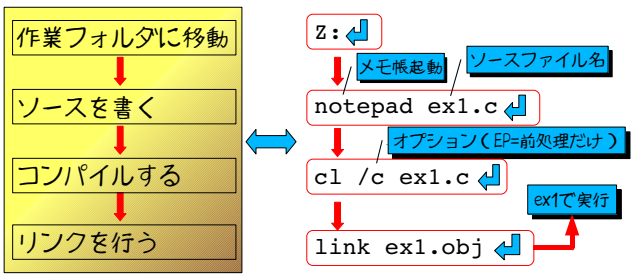



（演習1一回目資料より）

＊ ビルド = 実行ファイルの生成（= コンパイル + リンク）

## 3. 開発手順（コマンドライン版）

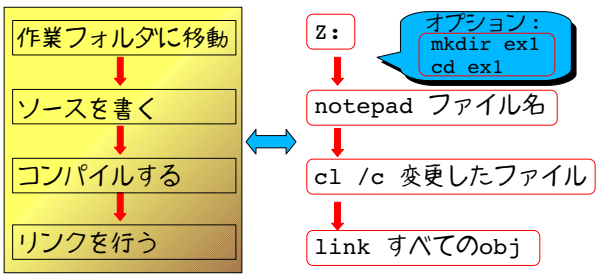
（Visual C++コマンドプロンプト）で（↵ = Enterキー）




＊ cl ex1.c だけの場合は，自動的にlinkが実行される

## 4. 複数ファイルの場合の開発

（Visual C++コマンドプロンプト）（最後の↵を省略）




＊（cl 変更したファイル）だけでは正しくリンクできない

## 5. ソースを複数ファイルに分割

分割コンパイル：大規模プログラムを複数ファイルに分割し個別にコンパイルを行う

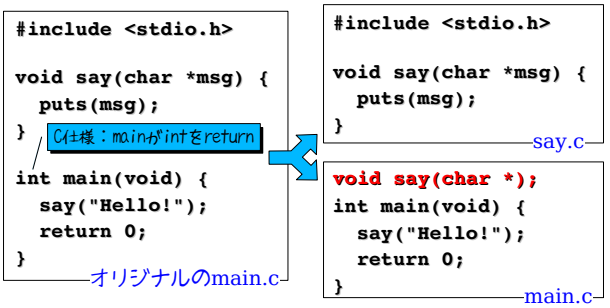
利点：

変更したファイルだけをコンパイルすればよい
⇒ コンパイルが速くなる
デバッグはファイル単位で行える
⇒ デバッグがやりやすくなる
複数人による共同作業はファイル単位でOK
⇒ 共同作業がやりやすくなる

} 開発が速くなる

＊ 参考：大規模？小規模？ 趙の判断基準はおおよそ1000行．

## 6. 分割コンパイルの練習 15分




＊ ビルド：cl say.c main.c（実行：sayで）

# 7. 分割コンパイル練習続き 9分

main.c（改定版）

```
void say(char *msg);
int main(int argc, char *argv[]) {
  int i;
  for (i=0; i<argc; i++) {
    say(argv[i]);
  }
  return 0;
}
```

コマンドライン引数の配列

コマンドライン引数の数

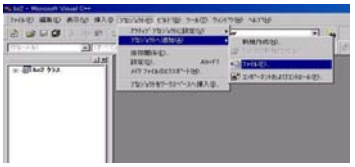
引数を全部改行付きで出力する

コンパイル：cl /c main.c
リンク：link say.obj main.obj
実行：say（sayが出力される）
　　say Hi（何が出力される？）

* link /out:ex1.exe ... のように実行ファイル名を指定できる

# 8. 統合開発環境による開発

Visual C/C++統合開発環境（略称 IDE）でファイル追加：
〔プロジェクト〕->〔プロジェクトへ追加〕->〔ファイル〕



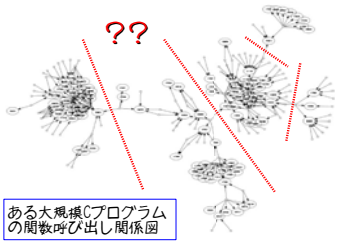
コマンドラインでの開発：小規模向け，簡単，自由度高
統合開発環境での開発：複雑，自由度低，大規模向け

* 通常IDEは更新（≒変更）されたファイルのみをコンパイルする


* 休まない人は，統合開発環境で前の練習をやってみよう

# 9. 分割コンパイルからモジュール化

. . . 分割コンパイルはいいが，効率的に開発するためには，適切な分割方法を見つける必要がある．




ある大規模Cプログラムの関数呼び出し関係図

モジュール
ある特定の機能を持って他のモジュールと一緒にプログラムを構成する．
通常，分割コンパイルを用いて，共通のヘッダファイル（.h）を持つ．

* 具体的に何が適切かは，経験や力に依存する

# 10. モジュールとAPI

共通のヘッダファイルを用いてモジュールを定義．
（実装の詳細が分からなくても使うことができる）

例：#include <stdio.h> により（モジュール）標準入出力のライブラリが使えるようになり，それには関数printf, scanfなどが入っている．
stdio.hには標準入出力のAPIが定義されている．

* API: Application Programming Interface

# 11. モジュールとAPI定義の例

say.h

```
void say(char *);
```

APIを定義

main.cは関数sayの実装を知らない

```
#include "say.h"
int main(int argc, char *argv[]) {
  int i;
  for (i=0; i<argc; i++) {
    say(argv[i]);
  }
  return 0;
}
```

main.c

```
#include <stdio.h>
void say(char *msg) {
  puts(msg);
}
```

say.c

* 前の練習と同じようにビルドされ，同じ結果になる

# 12. libbzip2で見るモジュール設計

APIの定義：汎用性と独立性を充分考慮すること

例：ファイル圧縮/解凍のライブラリlibbzip2の簡易版API

```
#include "bzlib.h" /* libbzip2を利用するために必要 */
BZFILE * BZ2_bzopen(const char *, const char *);
int BZ2_bzread(BZFILE *, void *, int);
int BZ2_bzwrite(BZFILE *, void *, int);
void BZ2_bzclose(BZFILE *);
```

ファイルを開く

ファイルから読み込む

ファイルに書き出す

ファイルを閉じる

* 利用の流れ：開く -> 読み込む／書き出す -> 閉じる

# 13. libbzip2 API詳解

```
BZFILE * BZ2_bzopen(const char *path, const char *mode);
```

解説：bz2形式のファイルを開いてBZFILE型のポインタを返す．
path：ファイル名（拡張子bz2）．
mode："r" = 読み込む，"w" = 書き出す．

```
int BZ2_bzread(BZFILE *b, void *buf, int len);
int BZ2_bzwrite(BZFILE *b, void *buf, int len);
```

解説：読み込む(BZ2_bzread)/書き出す(BZ2_bzwrite)．
　実際に読み込んだ/書き出した未圧縮データのバイト数を返す．
b：BZ2_bzopenで開いたファイル（ポインタ）
buf：未圧縮データを格納するバッファ
len：未圧縮データの長さ(単位バイト)

* 完全版のAPI定義はbzlib.hに入っている．. . 宿題をやったか

## 14. bz2解凍プログラムbz2.c

```
#include <stdio.h>
#include "bzlib.h"
#define BUFSIZE 1024

int main(void) {
  BZFILE *b;
  char buf[BUFSIZE];
  int len;
  b = BZ2_bzopen("sample.bz2", "r");
  while (len = BZ2_bzread(b, buf, BUFSIZE))
    fwrite(buf, sizeof(char), len, stdout);
  BZ2_bzclose(b);
  return 0;
}
```

コンパイル：
cl /c bz2.c
リンク：
link bz2.obj libbz2.lib
実行：bz2

bufからlen個charを標準出力へ出力

＊ bzlib.h, libbz2.lib, sample.bz2と同じフォルダに置く

## 15. 宿題

1. 解凍ファイル名がコマンドライン引数から取得するように．
（例えばコマンドラインで bz2 foo.bz2 のように実行すると
foo.bz2が解凍され，結果が表示される．）

2. bz2形式で圧縮できるソフト，例：Lhaz
http://www.vector.co.jp/soft/win95/util/se107748.html
で圧縮したファイルが作ったbz2.exeで解凍できることを確認．

次回：圧縮プログラムを作成する予定．

＊ 情報らしいことをしたいなら，プログラミングをマスタしよう

# H17年度演習II - 課題3

趙　亮

宇都宮大学工学部情報工学科

2005年12月

# 三 回 目

## 例で学ぶモジュール設計

## （bz2形式対応圧縮解凍ソフトの作成）

## 1. ファイルとファイルの拡張子

ファイル：名前で管理できるデータ

拡張子：ファイル名において最後の.より後ろの部分

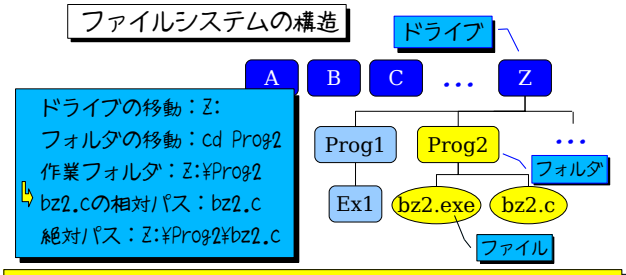
| | |
|---|---|
| 例1: Cのヘッダファイル stdio.h, | 拡張子 h |
| 例2: Cのソースファイル ex1.c, | 拡張子 c |
| 例3: オブジェクトファイル ex1.obj, | 拡張子 obj |
| 例4: 実行可能ファイル cl.exe, | 拡張子 exe |
| 例5: PDFファイル report.pdf, | 拡張子 pdf |
| 例6: HTMLファイル index.html, | 拡張子 html |

＊ 通常拡張子はファイルの種類を示すようにつけられる

## 2. Windowsのファイルシステム

実行：.exe, .bat, .com, ... 書類：.c, .h, .txt, ...

ファイルシステムの構造




＊ プログラムでは Z:¥¥Prog2¥¥bz2.c のように ¥¥ を使うこと

## 3. ファイルの実行（コマンド）

実行ファイル名 引数1 引数2 引数3 ...　スペース区切り

例1：notepad.exe ex1.c

例2：cl.exe /c say.c main.c　.exeは省略可

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

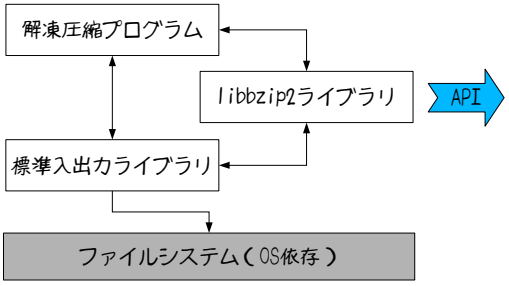
argc = 実行ファイル名を含む引数の数

argv[0] = 実行ファイル名, argv[1] = 引数1,

argv[2] = 引数2, ..., argv[argc] = NULL

＊ ファイル名が相対パスの場合は，作業フォルダに注意せよ

## 4. bz2形式対応解凍圧縮ソフトの構成




＊ libbzip2はbzip2.h，標準入出力はstdio.hを参照する

## 5. libbzip2 API詳解（復習）

```
BZFILE * BZ2_bzopen(const char *path, const char *mode);
```
解説：bz2形式のファイルを開いてポインタ（BZFILE型）を返す
path：（相対または絶対パスの）ファイル名（拡張子bz2）．
mode："r" = 読み込む，"w" = 書き出す．

```
int BZ2_bzread(BZFILE *b, void *buf, int len);
int BZ2_bzwrite(BZFILE *b, void *buf, int len);
```
解説：読み込む(BZ2_bzread)/書き出す(BZ2_bzwrite).
　実際に読み込んだ/書き出した未圧縮データのバイト数を返す
b：BZ2_bzopenで開いたファイル（へのポインタ）
buf：未圧縮データを格納するバッファ
len：最大lenバイトの未圧縮データを読み込む/書き出す

`void BZ2_bzclose(BZFILE *b);` 開いたファイルbを閉じる

## 6. 前回解凍のみのbz2.c

```
#include <stdio.h>
#include "bzlib.h"
#define BUFSIZE 1024

int main(void) {
  BZFILE *b;
  char buf[BUFSIZE];
  int len;
  b = BZ2_bzopen("sample.bz2", "r");
  while (len = BZ2_bzread(b, buf, BUFSIZE))
    fwrite(buf, sizeof(char), len, stdout);
  BZ2_bzclose(b);
  return 0;
}
```

sample.bz2を解凍し標準出力へ出力
読み込みモードでsample.bz2を開く
len==0 <=>ファイル終わり
圧縮データをbufに解凍
bufからlen個charを標準出力へ出力

＊ fwriteは標準入出力APIの一部で後ほど説明する

## 7. 宿題チェック（15分）

1. 解凍ファイル名をコマンドライン引数から取得する．例えば：
　bz2 foo.bz2 のように実行すると foo.bz2 が解凍される．
　bz2 sample.bz2 を実行すると sample.bz2 が解凍される．

2. bz2形式で圧縮できるソフト，例：Lhaz
　http://www.vector.co.jp/soft/win95/util/se107748.html
で圧縮したファイルを作ったbz2.exeで解凍できることを確認．

＊ 1はTAにチェックしてもらい，2は自分でチェックしてよい

## 8. 宿題1の参照実装

bz2.c

```
#include <stdio.h>
#include "bzlib.h"
#define BUFSIZE 1024

int main(int argc, char *argv[]) {
  BZFILE *b;
  char buf[BUFSIZE];
  int len;
  b = BZ2_bzopen(argv[1], "r");
  while (len = BZ2_bzread(b, buf, BUFSIZE))
    fwrite(buf, sizeof(char), len, stdout);
  BZ2_bzclose(b);
  return 0;
}
```

コマンドライン引数を処理できるように
読み込みモードで引数1のファイルを開く

＊ 実用には argc≠2，b==NULL等の場合も考える必要がある

## 9. 圧縮効果の確認（5分）

解凍結果の保存．例：sample.bz2を解凍してsampleに保存する
bz2 sample.bz2 > sample　（前にあったbz2の場合．）

ファイルサイズの確認：dir sample.bz2 sample

```
2005/12/13  21:57               840 sample.bz2
2005/12/19  16:40             1,332 sample
```
サイズ

ファイル内容の確認：type sample (type sample.bz2)

＊ 圧縮率の計算：（上記例の場合）(1332-840)/1332 = 37%


＊ 休まない人は，helpでコマンドラインをもっと知っておこう

## 10. 標準入出力 API詳解

```
#include <stdio.h>
```

```
FILE * fopen(const char *path, const char *mode);
```
解説：ファイル（なんでもよい）を開いてポインタを返す
path：ファイル名，mode："rb" = 読み込む，"wb" = 書き出す

```
size_t fread(void *buf, size_t s, size_t n, FILE *fp);
size_t fwrite(const void *buf, size_t s, size_t n, FILE *fp);
```
解説：読み込む(fread)/書き出す(fwrite).
　実際に読み込んだ/書き出した回数を返す（0 = 失敗/終わり）
buf：データを格納するバッファ，s：一回のデータ量（バイト），
n：最大n回の読み込み/書き出しを行う
fp：ファイルポインタ．stdin = 標準入力，stdout = 標準出力，...

最後に `int fclose(FILE *fp);` で開いたファイルfpを閉じる

## 11. 標準入力を用いた例（15分）

dump.c

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 1024
int main(int argc, char *argv[]) {
  FILE *fp; char buf[SIZE]; int i, n;
  fp = fopen(argv[1], "rb");
  while (n = fread(buf, sizeof(char), SIZE, fp)) {
    for (i=0; i<n; i++) {
      if (isprint(buf[i])) printf("%c", buf[i]);
      else printf(".");
    }
  }
  fclose(fp);
  return 0;
}
```

isprintのため

コンパイル（自動リンク）：
　cl dump.c
実行例：dump dump.exe
　　dump dump.c

＊ まず頭だけで解読してみて，それから実行して確かめよう

## 12. bz2形式圧縮プログラムの流れ

入力ファイルを開く（fopen，rbモード）

↓

出力ファイルを開く（BZ2_bzopen，wモード）

↓

入力ファイルから読み込みながら（fread）
出力ファイルへ書き出していく（BZ2_bzwrite）

↓

ファイルを閉じる（fclose，BZ2_bzclose）

＊ 解凍の場合と逆の関係を持っている

## 13. 圧縮解凍ソフトの作成（課題）

bz2形式対応の圧縮解凍プログラムを作成せよ

一番目のコマンドライン引数が-dで解凍，-cで圧縮する；

二番，三番の引数は入力と出力ファイル名とする．例えば：

bz2 -d foo.bz2 foo を実行すると foo.bz2がfooに解凍される

bz2 -c bar bar.bz2 を実行すると barがbar.bz2に圧縮される

さらに実行結果によりデータ圧縮の効果を考察せよ．

＊ レポート課題．Webページも参照のこと

## 14. レポートについて

1. 指定表紙（Webページからダウンロードできる）を使うこと
2. 指定時間（Webページ参照）まで提出すること
3. 表紙に指定されたチェック項目をよく確認すること
4. 考察項目：圧縮率を用いて圧縮の効果を示すこと
   （できるならzipなど他の形式との比較も入れるとなおよい）
5. オプション課題はやらなくてもよい

＊ レポートは自分の理解に基づいて簡潔明瞭に書くこと

## 15. オプション課題（2）

プログラムの実行時間を測定する

main関数の始めをプログラムの始まり，returnの直前を終わりとし，実行時間を測定できるプログラムを作成する．さらにいくつかのファイルを用いて圧縮と解凍の時間をそれぞれ測定し報告すること．できるなら考察を入れる．なお，オプション課題については書式自由．

＊ ヒント：演習 1 課題 3 二回目の資料