

# ソフトウェア概論 A/B

-- 分割コンパイルと makefile --

数学科 栗野 俊一 / 渡辺 俊一

# 伝言

---

## 私語は慎むように !!

### □ 色々なお知らせについて

- 栗野の Web Page に注意する事

<http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino>

### □ 講義開始前に済ませておく事

- PC の電源を入れる
- ネットワークに接続しておく事
- 今日の資料に目を通しておく事

### □ 講義前の注意

- 講義前は、栗野は準備で忙しいので TA を捕まえてください

### □ やる気のある方へ

- 今日の資料は、すでに上っています
  - ▷ どんどん、先に進んでかまいません

### □ 本日の CST Portal の出席パスワード : 20150424

- 出席は成績に影響しませんが、折角の機能なので、使いましょう

# 前回(2015/04/17)の復習

---

## □ 前回(2015/04/17)の内容

### ○ プログラムとは

- ▶ 計算機への指示(作業手順)を記述したもの

### ○ コンパイルとは

- ▶ 人間に判り易い形式(C 言語)から計算機が実行できる形(機械語)に変換する

### ○ C 言語

- ▶ printf : メッセージを出力する関数
- ▶ 順接 : 命令を並べると、その順序に実行される
- ▶ 関数 : 幾つかの命令列に名前を付けたもの

## □ 演習

### ○ Compile の仕方を覚える

### ○ プログラムを書いてみよう

- ▶ Hello, World
- ▶ 関数を並べてみよう / 関数を作ってみよう

# 前回 (2015/04/17) の課題

---

## □ 前回 (2015/04/17) の課題

### ○ 次の C Program ファイルを作成し提出しなさい

- ▶ 今回は提出先は二つある ( CST Portal : 去年と同じ / e-mail )

### ○ CST Portal

- ▶ ファイル名 : 20150417-01-YYYY.c (YYYY は学生番号)
- ▶ 内容 : 「Hello, 自分の名前」を100回以上出力する C 言語のプログラム
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)

### ○ NU-AppsG のメール機能を利用して課題を提出する

- ▶ 宛先: kurino.shunichi@nihon-u.ac.jp
- ▶ 表題:「ソフトウェア概論:20150417-1-YYYY」
- ▶ 内容: 自分の学籍番号と名前
- ▶ 添付: 20150417-1-YYYY.c (YYYY は学生番号)

# 本日の課題 (2015/04/24)

---

## □ 今回 (2015/04/24) の課題

### ○ 課題 1:

- ▶ ファイル名 : 20150424-01-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 童謡を演奏するプログラムを作成しなさい
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)
- ▶ 曲は何でもよい

### ○ 課題 2:

- ▶ ファイル名 : 20150424-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 童謡の歌詞を出力するプログラムを作成しなさい
- ▶ ファイル形式 : テキストファイル(C 言語プログラムファイル)
- ▶ 可能な限り引数付きの関数で..
- ▶ 曲は何でもよい

# C 言語で音楽を

---

## □ おまじない

- `#include "s_midi.h"` を冒頭にいれる

- ▶ \*.c と同じフォルダに s\_midi.h をおく(コピーする)

## □ 音の鳴らし方

- `s_midi_play ( S_MIDI_XX );` で音をならす

- ▶ XX C4 がド, D4 がレ、以下 E4, F4, G4, A5, B5, C5

- `s_midi_length ( S_MIDI_LNEN_X );` で音の長さを調節

- ▶ X は 1, 2, 4, 8 の4通り

- ▶ 実は、S\_MIDI\_LNEN\_8 で 500 が指定されたのと同じ

- ▶ 直接音の長さを指定してもよい ( 単位は m sec )

# make と makefile

---

## □ make とは

- 様々な操作を自動的に行ってくれるコマンド
  - ▶ 「ファイルを『作成』する」ので「make (作る)」という名前

## □ Makefile

- 様々な「ファイルの作成方法」を記述したファイル
  - ▶ make は Makefile を読み込んで、ファイルを作成する
- cf. 料理で考えると、Makefile がレシピで、make が料理人、ファイルが料理

## □ Makefile の記述方法

- 基本の形式は次の形
  - <<作りたいファイル>> : <<材料となるファイル>> ...
  - <<TAB コード>><<作りために実行する命令>>

- 例: (hello.exe を作る Makefile)

```
hello.o : hello.c # hello.c を材料に hello.o を作る
    cc -c hello.c # hello.o を作るには「cc -c hello.c」とする
hello.exe : hello.o # hello.o を材料に hello.exe を作る
    cc -o hello.exe hello.o
```

# Makefile の色々

---

## □ Makefile には色々な便利な機能がある

### ○ 変数 : 文字列に名前を付ける事ができる

▶ 変数(の値の)定義 : 「変数名」=「値」とすると変数に値の内容を割り当てる

▶ 変数(の値の)参照 : 「\$(変数名)」と書くと「変数の

### ○ 例: (hello.exe を作る Makefile)

```
BASE=hello      # 変数 BASE に hello という値を割り当てる
```

```
${BASE}.o : ${BASE}.c # ${BASE} はどれも hello に置き換わる
```

```
    cc -c ${BASE}.c # 結果的に「cc -c hello.c」と同じ
```

```
${BASE}.exe : ${BASE}.o
```

```
    cc -o ${BASE}.exe ${BASE}.o
```



# 関数 (再録)

---

## □ 関数

### ○ 命令列に名前をつけたもの

- ▶ 名前を指定して「呼出す」だけで、その命令列が実行できる

## □ 関数定義

### ○ 命令列を「{」と「}」でかこって、それに関数名をつける

- ▶ この命令列を関数の「本体」と呼ぶ
- ▶ 「void」とか「()」は今回は説明しない

## □ 関数呼び出し

### ○ 関数名を指定している事により、関数の本体の命令列が実行できる

- ▶ 「()」は今回は説明しない

## □ 関数の効用

- 「名前」が付くのでプログラムが理解り易くなる
- 関数を利用するとプログラムがみじかくなる

# 関数の作り方 (その 1)

---

## □ 関数の作り方(引数のない場合)

- 名前を決める

- ▷ cf. subfunc

- どの部分を関数にするかを決める

- 関数にする部分を取り出し、外に出し、ブロックにする

- ▷ ブロックにするには '{' と '}' で囲めばよい

- ▷ 名前を付ける ( cf. void subfunc() )

- もともと部分があった所に関数呼び出しを書込む

- ▷ cf. subfunc();

# 関数呼び出しの挙動

---

- 関数呼び出しは次のように振舞う
  - 関数呼び出しのある場所から関数の先頭にゆく
  - 関数の中身を実行する
  - 関数呼び出しのある場所の次に戻る
- 関数の引数とは
  - 関数の振舞いを変更するための情報 (パラメータ)
    - ▶ 同じ関数でも引数が異れば異なる振舞いをす
- 引数付きの関数の呼び出し
  - 関数の中の変数に、引数の値が入っている

# 分割コンパイル

---

## □ C 言語で記述されたプログラムの構造

### ○ main 関数が必ず必要

▶ 他の関数は main 関数から呼び出される

### ○ 関数の定義

▶ ソースファイル (\*.c) の中に記述する

▶ 同じファイル内である必要はない

## □ 分割コンパイル

### ○ 関数を別のファイルで定義し、個々にコンパイルする事

▶ 後でリンクにより一つの実行ファイルにまとめる

# make と分割コンパイル

---

- 分割コンパイルは複数のファイル进行处理
  - 作業も面倒だし、間違いも起きやすい
    - ▷ コンパイルの手順を記述してコンピュータにやらせちゃおう
- **Makefile**
  - コンパイルの手順などを記述したファイル
- **make**
  - **Makefile** を読んで、コンパイルを自動的に行ってくれる

# 関数の作り方 (その 2)

---

## □ 関数の作り方(引数のある場合)

- ほとんど、同じ部分を探す
- 異なる部分を変数に置き換える
  - ▶ 異なる部分には名前を付ける (関数の引数)
- 置き換えたものを関数の本体にする
- 関数呼び出しでは、引数に対応する異なる値を設定する

# MIDI (1) : 準備

---

## □ Windows 上での作業

### ○ MIDI のプログラミングファイル ( midi.zip ) をダウンロード

- ▶ ダウンロード先は c:\usr\c\20150424 (今週のフォルダ) にする
- ▶ midi.zip を「全て展開」 ( c:\usr\c\20150424\midi )

## □ Ubuntu 上での作業

### ○ 必要なパッケージとそのインストール(一度だけやれば良い)

- ▶ `sudo /home/soft/c/20150424/midi/midi-install.sh`

### ○ 音を鳴らせるための環境設定

- ▶ 次の設定を行う(ubuntu を再起動した時に、一度実行する)
- ▶ `sudo /home/soft/c/20150424/midi/midi-setup.sh`

### ○ 動作確認

- ▶ 次のコマンドを実行して、鍵盤をクリックすると音になる事を確認
- ▶ `vkeybd --addr 128:0`

### ○ 音が鳴らない場合

- ▶ スピーカーの音量が 0 になっている可能性がある
- ▶ システム設定 -> サウンドで、音量が 0 / ミュート になっていない事を確認

# MIDI (2) : プログラミングのしかた

---

## □ Ubuntu での作業

### ○ 作業フォルダへ移動

▷ cd ~/c/20150424/midi

### ○ サンプルプログラムを実行してみる

▷ make test

### ○ 「かえるの歌」も試してみる

▷ make BASE=kaeru test

## □ 課題提出ファイルの作成

### ○ windows 上の作業

▷ サクラエディタで、c:\usr\c\20150424\midi の中の kaeru.c を見る

▷ kaeru.c をコピーして 20150424-01-QQQQ.c を作成する

### ○ Ubutu での作業

▷ 次のコマンドを実行すると、20150424-01-QQQQ.exe が作成される

▷ make BASE=20150424-01-QQQQ test

### ○ windows 上の作業

▷ 演奏がうまくいったら、20150424-01-QQQQ.c を提出