

東北大学全学教育科目

# 情報基礎 A

---

Linux の利用

C言語によるプログラミングその1

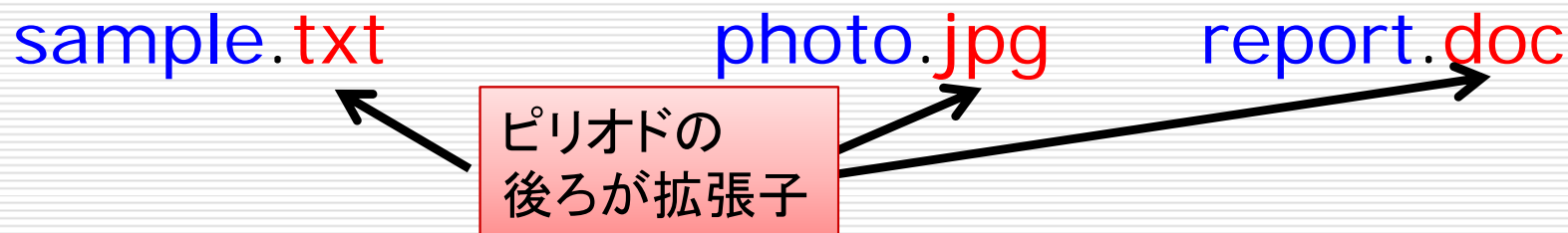
担当: 大学院情報科学研究科

塩浦 昭義

# ファイル名と拡張子

---

- ファイル名の多くは拡張子をもつ
- 拡張子はファイルの種類を表す
- 拡張子がついていると、ファイルをクリックしただけで適切なソフトウェアを起動させることも可能



# 拡張子の例

同じソフトウェアの  
ファイルでも、  
バージョンが違くと  
拡張子が違う

| 拡張子       | ファイルの種類                                   |
|-----------|---|
| txt       | テキストファイル                                  |
| doc       | Microsoft Word のファイル (2003以前のバージョン)       |
| docx      | Microsoft Word のファイル (2007以降のバージョン)       |
| odt       | OpenOffice Writer のファイル                   |
| xls       | Microsoft Excel のファイル (2003以前のバージョン)      |
| xlsx      | Microsoft Excel のファイル (2007以降のバージョン)      |
| ods       | OpenOffice Calc のファイル                     |
| ppt       | Microsoft Powerpoint のファイル (2003以前のバージョン) |
| odp       | OpenOffice Impress のファイル                  |
| html, htm | HTML文書 (ウェブページ)                           |
| jpg, jpeg | JPEG形式の画像ファイル                             |
| mpeg      | MPEG形式の動画・音声ファイル                          |

# 拡張子に関する注意

- ファイル名から拡張子を消すと、ファイルの種類がわからなくなる
  - どのソフトウェアを使えば良いのか、全くわからなくなる
  - 間違って消した場合は、再び正しい拡張子を付ければOK

Word のファイル  
report.doc

ファイル名変更

Word で開けなくなる  
report



- 拡張子のみを変更しても、ファイルの種類は変わらない
  - ファイルの種類を変更したい場合は、適切なソフトウェアを使う必要あり

Word のファイル  
report.doc

拡張子変更

ウェブページに変更  
されるわけではない

report.html



---

# Linux の利用

---

# Linux (リナックス) と Windows

---

- 2つの異なるオペレーティングシステム(OS)
    - コンピュータ上で各種ソフトウェアを動かすための基礎となるソフトウェア
  - よくある例え(自分自身で確認してください)
    - Windowsは**オートマチック**操作の自動車
      - 初心者でも扱いやすい, 熟練者には少し物足りない
    - Linuxは**マニュアル**操作の自動車
      - 初心者には扱いが難しい, 熟練すると自在に扱える
  - 2つのOSに善し悪しはない. 好みで選ぶ
-

# LinuxとWindowsの比較： オープンソースとクローズドソース

---

- Linux はオープンソース
    - ソフトウェアの中身(プログラム, ソース)が一般に公開
    - 世界中の利用者が自由に開発に関わることが出来る
    - 自由に修正可能
  
  - Windowsはクローズドソース
    - ソフトウェアの中身は非公開
    - 一企業(Microsoft)が開発
    - 修正は不可能
-

# LinuxとWindowsの比較： オープンソースとクローズドソース

---

- Linux はオープンソース
    - 基本的に無料(有料のソフトやサービスがつくこともある)
    - 企業からのサポートなし, 困ったときは自力で解決(他のユーザーに助けってもらうことも可能)
  - Windowsはクローズドソース
    - 有料
    - Microsoft による利用者へのサポート
-



# LinuxとWindowsの比較： 操作方法とソフトウェア

---

## □ 操作方法

- Windowsでは視覚的にわかりやすいGUI(Graphic User Interface)が主流. マウスだけで操作可能
- Linuxではコマンド入力によるCUI(Command User Interface)をよく使う. キーボードだけで操作可能

## □ 使えるソフトウェア

- Windowsと同様のソフトウェアが利用可能
  - Linux では Microsoft Office は利用不可  
→OpenOffice で(ある程度)代用可能
  - Linux では Internet Explore は利用不可  
→Firefox で代用可能
-

# Linuxディストリビューション

---

- Linuxは通常、利用者が使いやすいパッケージという形で配布されている(ディストリビューション)
  - インストールに便利なプログラム付き
  - 各種ソフトウェア(ウェブブラウザ, メール用ソフト, オフィススイート, など)
- Linuxは一般に公開されている
  - 様々な企業, 団体がLinuxディストリビューションを開発
    - 同じLinuxでも, ディストリビューションによって見かけや内容が違う
  - Ubuntu, Fedora, Vine, Debian, Red Hat, KNOPPIX, **CentOS**

# Linux の利用

The image shows a Linux desktop environment with a blue background and white floral patterns. On the left side, there is a vertical dock with several icons: a document icon labeled 'KWrite', the Firefox logo labeled 'firefox', a trash can icon labeled 'ごみ箱', a document icon labeled 'アプリケーション起動...', a folder icon labeled 'ホーム', and a terminal icon labeled '端末'. Five callout boxes with white backgrounds and black text point to these icons. The top callout points to KWrite and says 'テキストエディタ KWrite'. The second callout points to Firefox and says 'Webブラウザ Firefox'. The third callout points to the terminal icon and says '端末(コマンド入力に使う)'. The fourth callout points to the terminal icon and says '他のプログラムはここをクリック'. The fifth callout points to the terminal icon and says 'ログアウトの時もここをクリック'. At the bottom of the screen, there is a taskbar with a panel containing icons for applications, a window manager indicator with numbers 1-4, a system tray with a clock showing '10:20 水曜日 2011-07-06', and a system status icon.

テキストエディタ  
KWrite

Webブラウザ  
Firefox

端末(コマンド入  
力に使う)

他のプログラム  
はここをクリック

ログアウトの時  
もここをクリック

Linux Tips - Mozilla Firefox

10:20  
水曜日  
2011-07-06

# コンピュータの操作方法

---

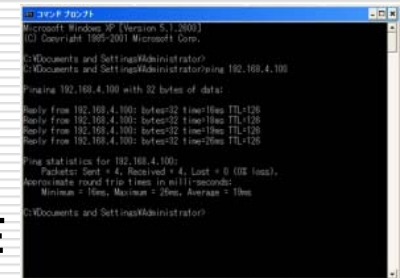
## □ GUI (Graphical User Interface)

- 現在の主流
- 主にマウスを利用, グラフィックを多用して視覚的にわかりやすい



## □ CUI (Character User Interface)

- 古典的な操作方法. 現在も使われる.  
Linux ユーザはよく使う
- キーボードから文字(コマンド)を入力して操作



# GUIとCUIの比較

---

|                         | GUI                  | CUI                      |
|-------------------------|----------------------|--------------------------|
| 操作方法                    | 直感的で簡単               | わかりにくい<br>コマンドを覚えていないと駄目 |
| 簡単な処理                   | 簡単                   | コマンドを覚えていれば簡単            |
| 複雑な処理<br>(操作の記録, 自動化など) | 難しい<br>効率が悪い         | コマンドを組み合わせると簡単,<br>効率的   |
| 必要な計算機資源                | 多い<br>計算機の性能が悪いと使えない | 少ない<br>計算機の性能が悪くてもOK     |
| 主な利用者                   | 一般ユーザ, 初心者           | 管理者, プログラマ, 上級者          |

# GUIの不便なところを体感

---

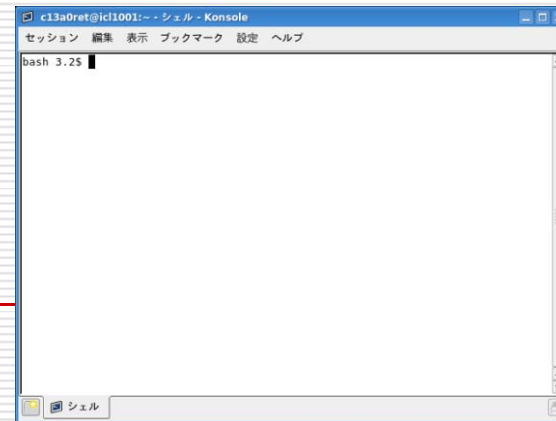
## □ 準備:

1. 「ホーム」のフォルダを開く
2. test という名前のフォルダを作る
3. test フォルダの中に a, b, c という名前のフォルダを作る
4. a, b, c それぞれのフォルダの中に, 異なるファイル名のファイルを1つずつ入れる  
(テキストエディタで新しいファイルを作っても良い)

## □ GUIでの実験

- マウスを使って、a, b, c それぞれに入っているファイルを全部、ホームフォルダにコピーする
-

# CUIの便利なところを体感



## □ 準備:

- 先ほどコピーした、ホームフォルダのファイルをすべて削除  
(フォルダ a, b, c は削除しない)

## □ CUIでの実験

1. 端末を開く
2. 「cp /test/\*/\* .」と入力したのち、Enterキーを押す
3. ホームフォルダを開いて、中身を確認

CUIでは、コマンドさえ覚えれば、  
複雑な作業、大量の単調な作業を  
効率的に行うことが可能

---

# コンピュータとプログラミング

---

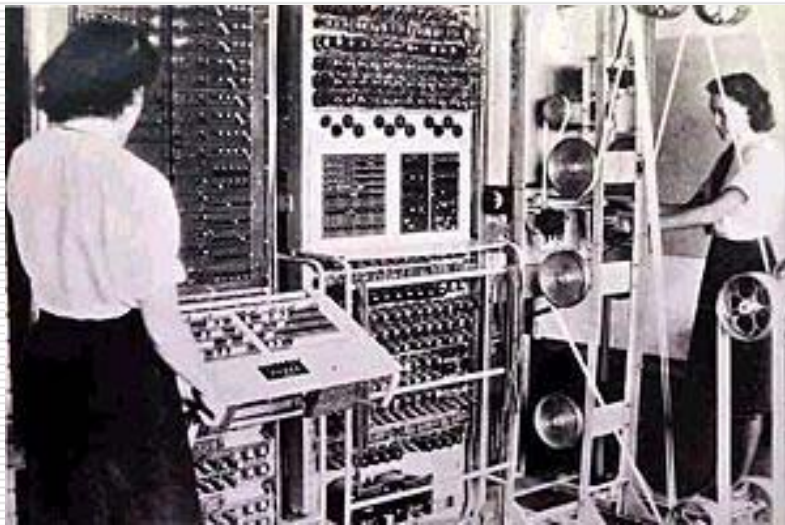


# コンピュータとプログラミングの歴史

---

□ コンピュータとは: 複雑な計算を短時間で行ってくれる道具

1944年 世界初の電子式の計算機 Colossusが  
完成(暗号解読用)



Colossus mark2

計算手順の変更は  
配線などの物理的変更  
により実現



操作が難しく  
汎用性は乏しい

# コンピュータとプログラミングの歴史

1946年 計算手順をソフトウェアとして分離するアイデア

初期の  
計算機

ノイマン型計算機

分離

Hardware (ハードウェア)

計算機を構成する機械や回路

Software (ソフトウェア)

計算機で実行する計算処理の  
手順を文書化したもの  
(プログラム)

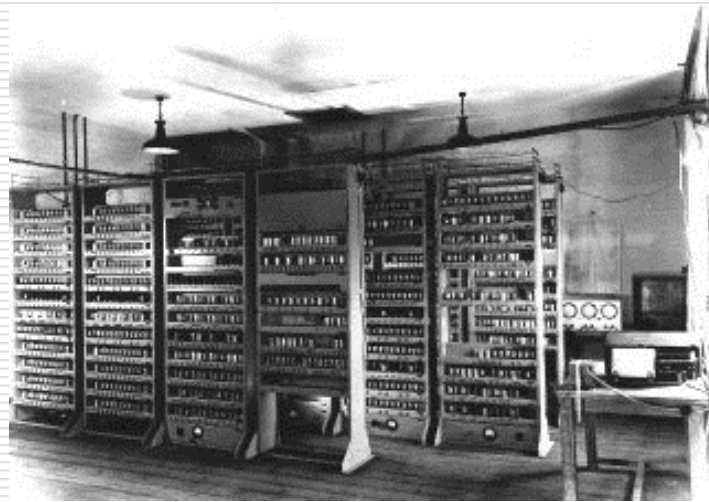
計算処理の変更が  
容易

汎用性が  
向上

# コンピュータとプログラミングの歴史

---

1949年 世界初の実用的 **ノイマン型計算機** **EDSAC** が完成



1970年以降のほとんどの  
計算機はノイマン型

**EDSAC**: Electronic Delay Storage Automatic Calculator

<http://www.infonet.co.jp/ueyama/ip/history/edsac99.9.jpg> (2010/06/10)

---

# プログラムとプログラミング

- プログラム(program) = 「計画書」
  - コンピュータの計算(コンピュータへの命令)の手順を記述したもの
- プログラミング(programming)  
= プログラムを作る(入力すること)
- コンピュータへの計画書は、コンピュータが理解できる言葉(機械語)で書かなければならない
- 機械語でプログラムを作るのは人間にとって難しい

```
000001d0 7c 08 02 a6 bf 61 ff ec 7c 7c 1b 78 7c 9f 23 78 | |....a..|.x|.#x|
000001e0 90 01 00 08 94 21 fc 80 41 9d 00 18 3c 60 00 00 | |.....!..A...<...|
000001f0 38 63 02 b0 48 00 02 59 38 60 00 09 48 00 01 f8 | |8c...H..Y8`..H...|
00000200 83 a4 00 04 3c 80 00 00 38 84 02 c0 7f a3 eb 78 | |....<...8.....x|
00000210 48 00 02 5d 7c 7e 1b 79 40 82 00 94 38 1c ff fc | |H..]|~.y@...8...|
00000220 2b 80 00 01 41 bd 00 ac 80 9f 00 08 3b 81 00 50 | |+...A.....;..P|
00000230 7f 83 e3 78 3b a1 01 50 48 00 01 d5 80 9f 00 0c | |...x;..PH.....|
00000240 7f a3 eb 78 3b 61 02 50 48 00 01 c5 3b e1 00 48 | |...x;a.PH...;..H|
00000250 7f a5 eb 78 7f 83 e3 78 38 80 00 00 38 c0 00 00 | |...x...x8...8...|
00000260 38 e0 00 00 39 00 00 00 39 20 00 ff 7f 6a db 78 | |8...9...9 ...j.x|
00000270 93 c1 00 3c 93 e1 00 38 48 00 01 b5 7c 7d 1b 79 | |...<...8H...|}.y|
00000280 41 82 00 10 3c 60 00 00 38 63 02 c4 48 00 01 10 | |A...<`..8c...H...|
00000290 80 41 00 48 7f 63 db 78 7c 5f 12 14 9b a2 02 08 | |.A.H.c.x|_.....|
000002a0 48 00 01 7d 38 60 00 00 48 00 01 4c 3c 80 00 00 | |H..}8`..H..L<...|
000002b0 7f a3 eb 78 38 84 02 e0 48 00 01 b5 2f 83 00 00 | |...x8...H.../...|
```

```
0x0000716c <+0408> mtctr r12
0x00007170 <+0412> bctr
0x00007174 <+0000> mflr r0
0x00007178 <+0004> stmw r30,-8(r1)
0x0000717c <+0008> stw r0,8(r1)
0x00007180 <+0012> stwu r1,-96(r1)
0x00007184 <+0016> mr r30,r1
0x00007188 <+0020> bcl- 20,4*cr7+so,0x718c <_darwin_gcc3_preregister_frame_info+24>
0x0000718c <+0024> mflr r31
0x00007190 <+0028> bl 0x7b00 <dylid_stub__init_keymgr>
0x00007194 <+0032> addis r2,r31,0
0x00007198 <+0036> lzw r2,3732(r2)
0x0000719c <+0040> lzw r0,8(r2)
0x000071a0 <+0044> stw r0,56(r30)
0x000071a4 <+0048> lzw r0,56(r30)
0x000071a8 <+0052> cmpwi cr7,r0,0
0x000071ac <+0056> beq- cr7,0x71c4 <_darwin_gcc3_preregister_frame_info+80>
0x000071b0 <+0060> lzw r2,56(r30)
```

# プログラミング言語

- コンピュータへの計画書は、コンピュータが理解できる言葉（機械語）で書かなければならない
- 機械語でプログラムを作るのは人間にとって難しい
- より便利なアプローチ

プログラミング言語

人間が理解しやすい言葉  
で書いたプログラム

```
int main()
{
    for(int val=1; val<50; val++){
        int tmp = val;
        if( val%3 == 0 ) { printf("%d\n",val);
        } else if( val%5 == 0 ) { printf("%d\n",val);
        } else {
            while( tmp>0 ) {
                if( tmp%10 == 3 ) {
                    printf("%d\n",val);
                    break;
                }
            }
            tmp /= 10;
        }
    }
}
```

翻訳

機械語のプログラム

コンパイル

```
000001d0 7c 08 02 a6 bf 61 ff ec 7c 7c 1b 78 7c 9f 23 78 | |...a...|.x|.#x|
000001e0 90 01 00 08 94 21 fc 80 41 9d 00 18 3c 60 00 00 | |...!.A...<...|
000001f0 38 63 02 b0 48 00 02 59 38 60 00 09 48 00 01 f8 | |8c..H..Y8`..H...|
00000200 83 a4 00 04 3c 80 00 00 38 84 02 c0 7f a3 eb 78 | |...<...8...x|
00000210 48 00 02 5d 7c 7e 1b 79 40 82 00 94 38 1c ff fc | |H..]l~.y@...8...|
00000220 2b 80 00 01 41 bd 00 ac 80 9f 00 08 3b 81 00 50 | |+...A.....;..Pl
00000230 7f 83 e3 78 3b a1 01 50 48 00 01 d5 80 9f 00 0c | |...x;..PH.....|
00000240 7f a3 eb 78 3b 61 02 50 48 00 01 c5 3b e1 00 48 | |...x;a.PH...;..Hl
00000250 7f a5 eb 78 7f 83 e3 78 38 80 00 00 38 c0 00 00 | |...x...x8...8...|
00000260 38 e0 00 00 39 00 00 00 39 20 00 ff 7f 6a db 78 | |8...9...9 ...j.x|
00000270 93 c1 00 3c 93 e1 00 38 48 00 01 b5 7c 7d 1b 79 | |...<...8H...l}.y|
00000280 41 82 00 10 3c 60 00 00 38 63 02 c4 48 00 01 10 | |A...<...8c..H...|
00000290 80 41 00 48 7f 63 db 78 7c 5f 12 14 9b a2 02 08 | |.A.H.c.x|_.....|
000002a0 48 00 01 7d 38 60 00 00 48 00 01 4c 3c 80 00 00 | |H..}8`..H..L<...|
000002b0 7f a3 eb 78 38 84 02 e0 48 00 01 b5 2f 83 00 00 | |...x8...H.../...|
```

# コンピュータが出来ること, 出来ないこと

---

- 基本的な演算を高速に実行可能
    - 四則演算, 大小関係の比較
    - 数値や文字などのデータを記憶
    - 文字やグラフィックを表示
  - 複雑な演算は, 簡単な演算を組み合わせて実行(人間がプログラムをつくる必要あり)
  
  - プログラムの指示通りに正確に動く
  - プログラムの内容が曖昧だと動いてくれない
  - プログラムが間違っているにもかかわらず, その間違い通りに動く
-

---

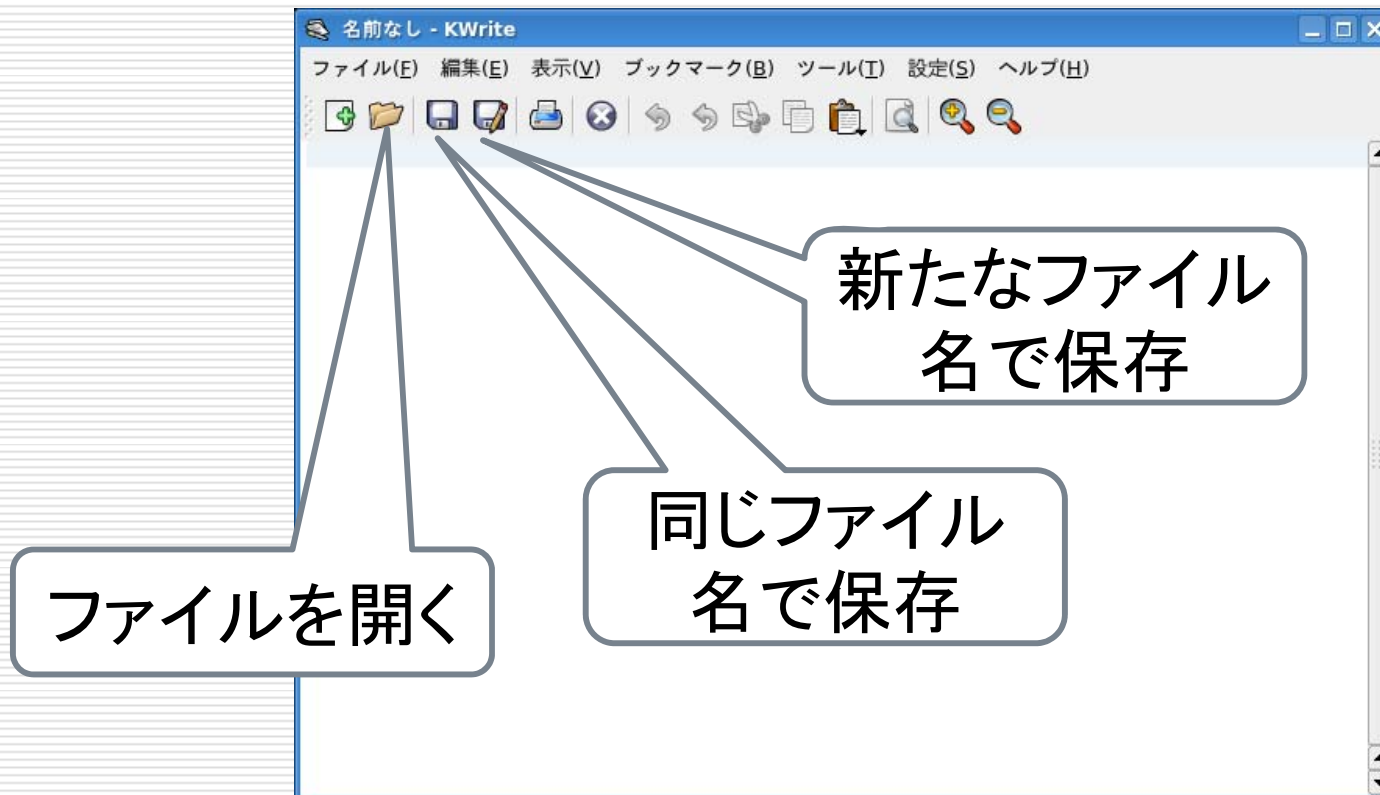
# C言語によるプログラミング

---

# C言語プログラム作成の手順

テキストエディタ(KWrite)にてプログラムを入力、保存

- 保存の際のファイル名は xxx.c (拡張子 c をつける)
- 例外を除き、半角英数字を使う
- **保存するフォルダは「ホーム」**にしてください





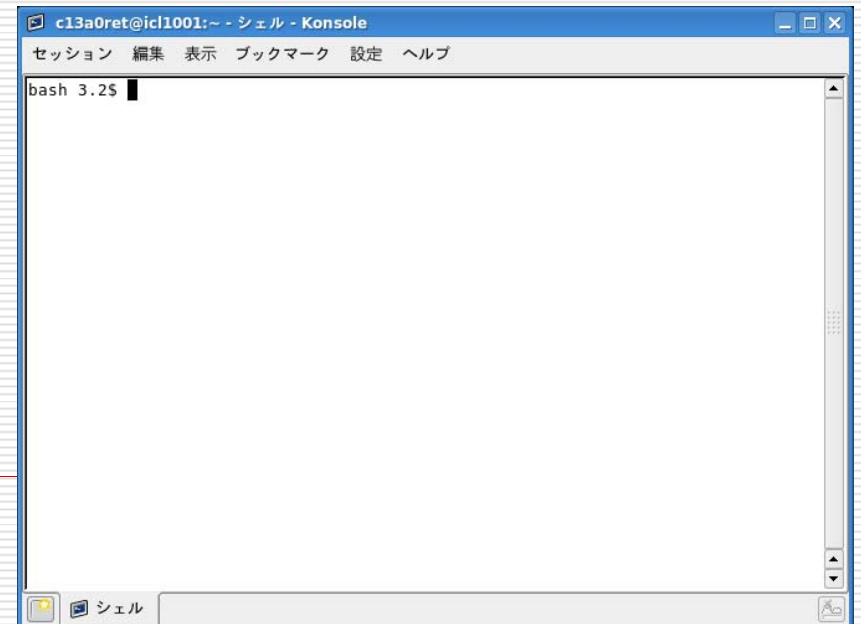
# プログラム実行のやりかた

---

C言語のプログラムは、そのままではコンピュータは理解できない  
→機械語(コンピュータの理解できる言語)に翻訳する必要あり  
(「**コンパイル**」という作業)

## 手順

1. 端末にて “gcc xxx.c” と入力, 実行(エンターキーを押す)  
(xxx.c は実行したいプログラム名)
2. エラーメッセージが出なかったら  
→端末にて “a.out” を実行するとプログラムが動く  
エラーメッセージが出たら  
→次のスライド参照



# エラー発生時の対応

---

プログラムをコンパイルした時にエラーが発生

→ 入力したプログラムに誤りあり

エラーの例 --- 大抵の場合、つまらないミスが原因

- かっこやカンマ、セミコロンがあるべき場所に存在しない
- 命令のスペルが間違っている
- ファイルを保存したフォルダが「ホーム」と異なる

エラーの修正方法

1. プログラムを入念にチェックして、誤りを修正、  
ファイルを再度保存
  2. もう一度プログラムをコンパイル (gcc xxxx.c を実行)
  3. エラーが出たら 1. に戻る
-

# プログラムその1

実際に入力して実行  
してみることに

“Hello.” と表示するプログラム hello.c

```
#include <stdio.h>

main()
{
    printf("Hello.¥n");
}
```

文字や記号はこの通りに  
正しく入力すること  
すべて半角英数字

レイアウトも出来るだけ  
この通りに正しく入力すること

プログラムは1行にまとめて書くことも可能ですが、  
わかりにくくなるので、きれいに書きましょう

```
#include <stdio.h>main(){printf("Hello.¥n");}
```

# 今日のレポート課題

- プログラム1を作って実行後, 提出 (hello.cのファイル): 授業中に提出
  - 次の天秤の問題の答えを**考える**こと(こちらの締切は7月19日まで)
    - テキストエディタで答えを書いて提出(できるだけ理由も書くこと)
    - 間違っても構いません. 自分の頭で考えてください
- 一個の天秤と8個の宝石があります. 天秤を使うと, 重さの大小を調べることができます. 天秤の皿には複数の宝石を置くことが可能です

**問0:** 1個だけ重さの軽い偽宝石がある. これを

探すには, 天秤を何回使う必要があるか? (答え: 2回)

**問1:** 宝石の重さが全部異なるとき, 天秤を使って, 一番重い宝石を探したい. 天秤を何回使う必要があるか?

**問2:** 宝石の重さが全部異なるとき, 天秤を使って, 宝石を重い順番に並べたい. 天秤を何回使う必要があるか? (宝石の重さは全部異なる)

# 問0の答え

---

8個の宝石, 1個だけ重さの軽い偽宝石

(0) 3個, 3個, 2個に分ける

(1) 3個のグループの重さを比較

(a) 釣り合った

→ 2個のグループの宝石を比較, 軽い方が偽物

(b) どちらかが軽い

→ 軽い3個のグループから2個を選び, 比較

2個の重さが等しい → 残り1個が偽物

2個のうち, どちらかが軽い → 軽い方が偽物

