情報システム工学実験
第2

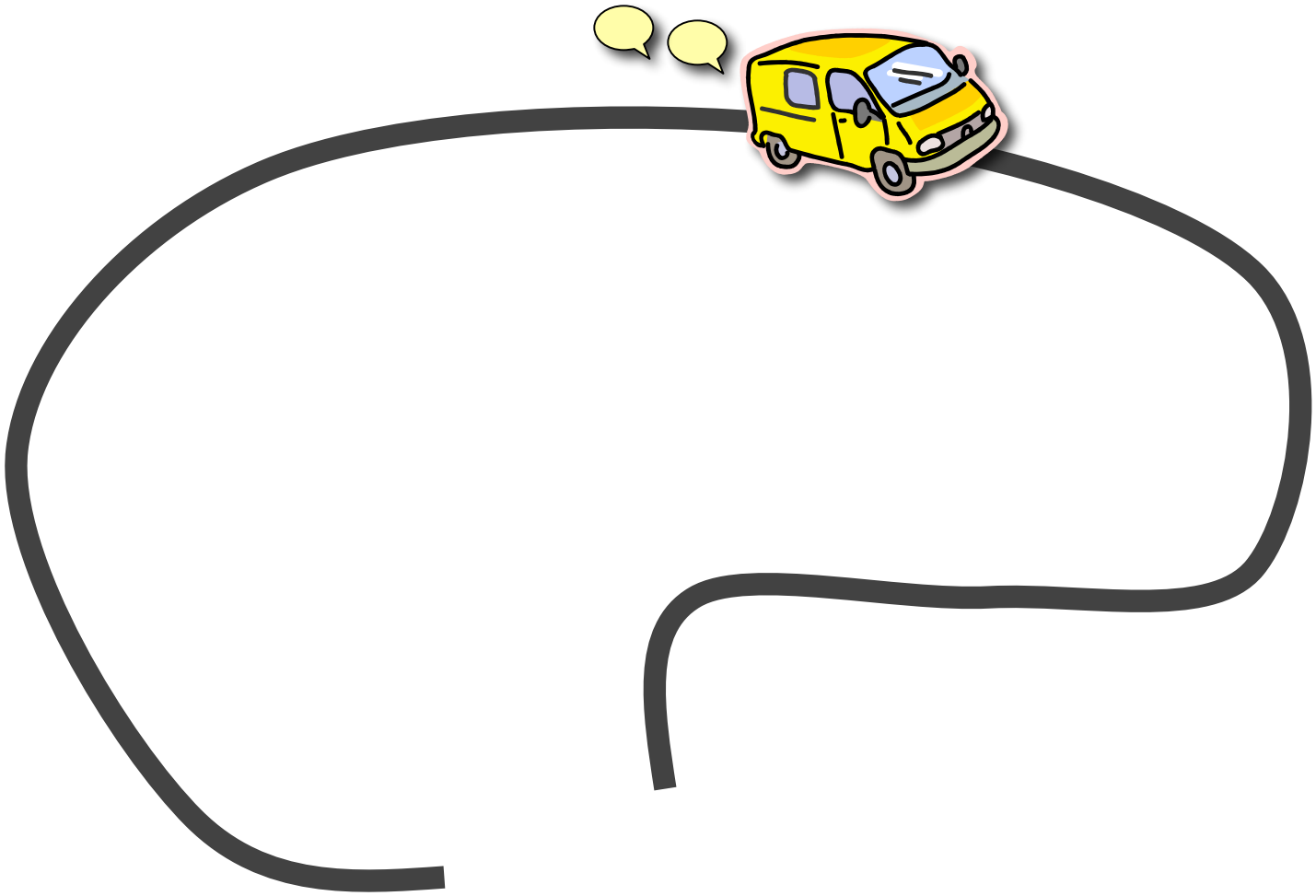
2007.10.1

教科書・参考書

- 教科書（必須）
  「新訂 新C言語入門 ビギナー編」
  林晴比古，ソフトバンククリエイティブ

- 参考書（詳しく知りたい人向け）
  「プログラミング言語C」第2版
  B.W.カーニハン，D.M.リッチー，石田晴久訳，共立出版

ライントレーサ (line tracer)



解きたい問題

これまでの実験

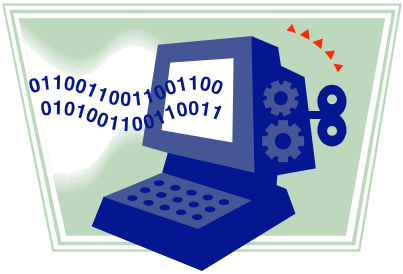
プログラミング

プログラム

コンパイル

この実験

実行可能
ファイル



実行

実行結果

最初のCプログラム

```c
/*
 * Program name: hello.c
 */

#include <stdio.h>

int main()
{
    printf(”hello, world\n”);
    return 0;
}
```

Cプログラム（図1.1）

```java
/*
 * Program name: Hello.java
 */
import java.io.*;
class Hello {
  public static
    void main(String[] arg)
    {
      System.out.
          println(”hello, world”);
      return;
    }
}
```

Javaで書いた場合

変換仕様

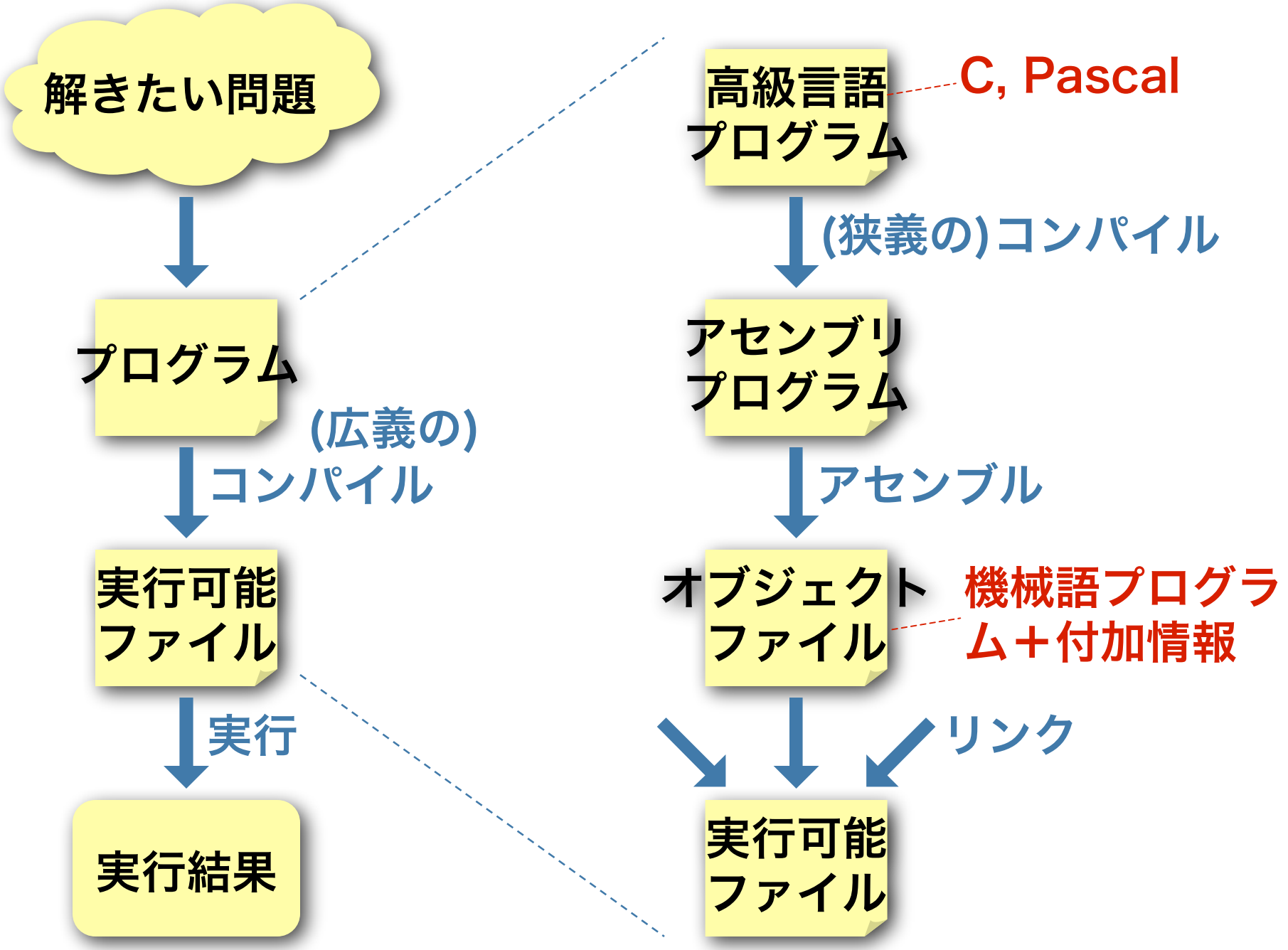
- c = 33 + 25;
  printf(”答えは%dです。\n”, c);

答えは58です。

コンパイルとリンク

2007.10.4

情報システム工学実験第2


解きたい問題
プログラム
(広義の)
コンパイル
実行可能
ファイル
実行
実行結果
高級言語
プログラム
C, Pascal
(狭義の)コンパイル
アセンブリ
プログラム
アセンブル
オブジェクト
ファイル
機械語プログラム＋付加情報
リンク
実行可能
ファイル

文字列

2007.10.11

情報システム工学実験第2

文字（char型）

```
#include <stdio.h>

int main()
{
    char a, b, c;
    a = 'D';
    b = 'e';
    c = 'c';
    printf("%c%c%c\n", a, b, c);
    return 0;
}
```

- **実行結果**

  Dec

文字列

```
#include <stdio.h>

int main()
{
    char s[6];
    s[0] = 'H';
    s[1] = 'e';
    s[2] = 'l';
    s[3] = 'l';
    s[4] = 'o';
    s[5] = 0;
    printf("%s\n", s);
    return 0;
}
```

- **実行結果**

```
Hello
```

strcpy

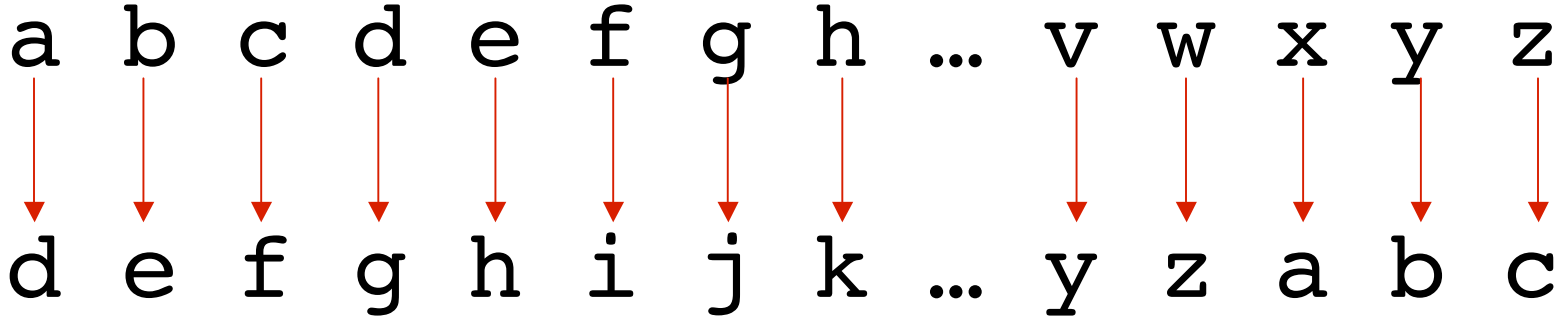
```
#include <stdio.h>
#include <string.h>

int main()
{
    char s[6];
    strcpy(s, "Hello");
    printf("%s\n", s);
    return 0;
}
```

- **実行結果**

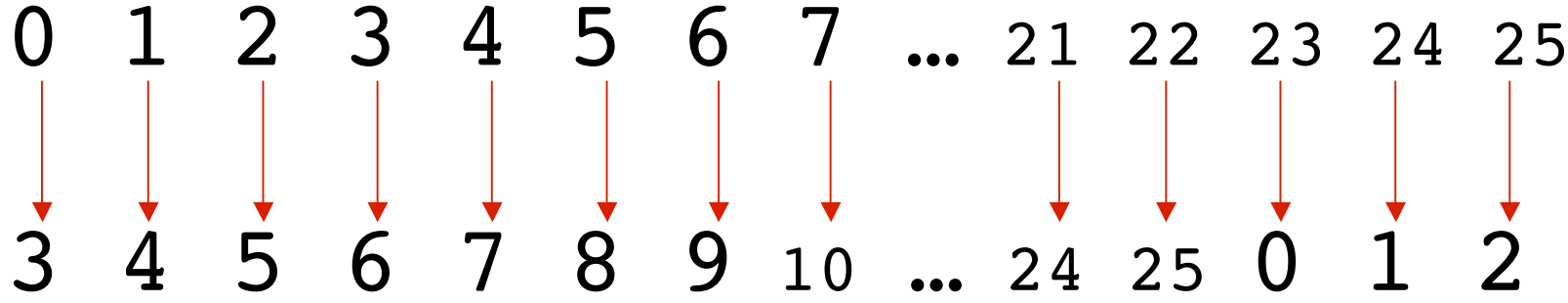
  Hello

シーザー暗号

- k=3のとき



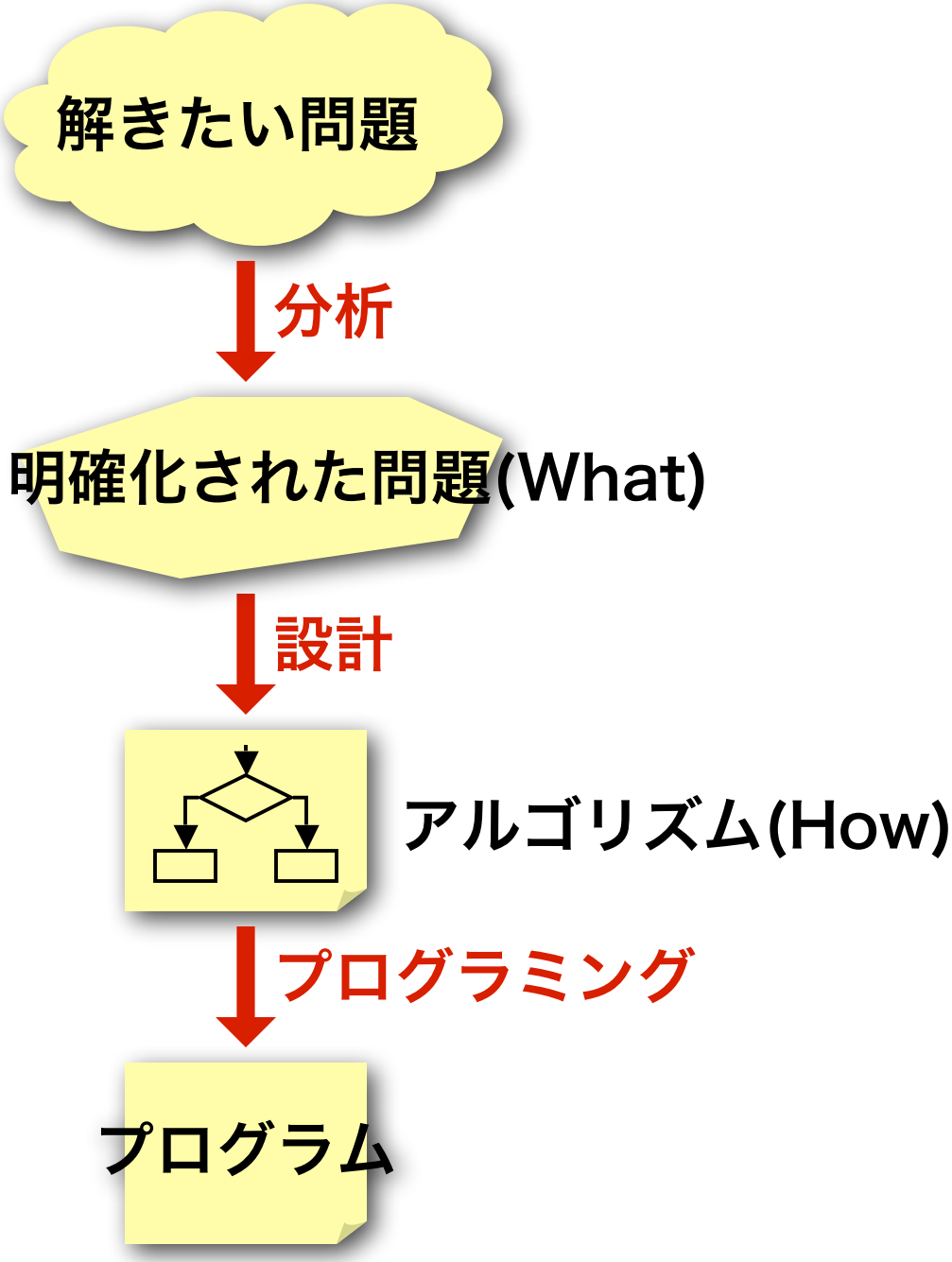
- 0 1 2 3 4 5 6 7 … 21 22 23 24 25



ファイル入出力

2007.10.15

情報システム工学実験第2


解きたい問題
分析
明確化された問題(What)
設計
アルゴリズム(How)
プログラミング
プログラム

課題17の前に:

- 課題a: 利用者にファイル名を入力させ、そのファイルの中身を画面に表示するプログラムを作りなさい (catのようなもの)
  - fgets を使ってプログラムしなさい*
  - fgetc を使ってプログラムしなさい
- 課題b: 利用者に入力ファイル名と出力ファイル名を入力させ、入力ファイルの中身を出力ファイルに書き出すプログラムを作りなさい(cpのようなもの)
  - fgets を使ってプログラムしなさい*
  - fgetc を使ってプログラムしなさい*
- 課題c: 利用者にファイル名を入力させ、そのファイルの中身の英小文字は英大文字に換え、それ以外の文字はそのまま、画面に表示するプログラムを作りなさい

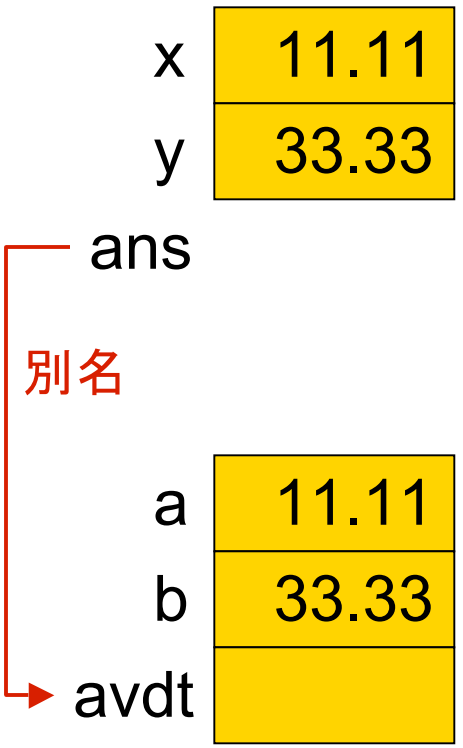
ポインタ引数
ポインタと配列
コマンドライン引数

2007.10.18

情報システム工学実験第2

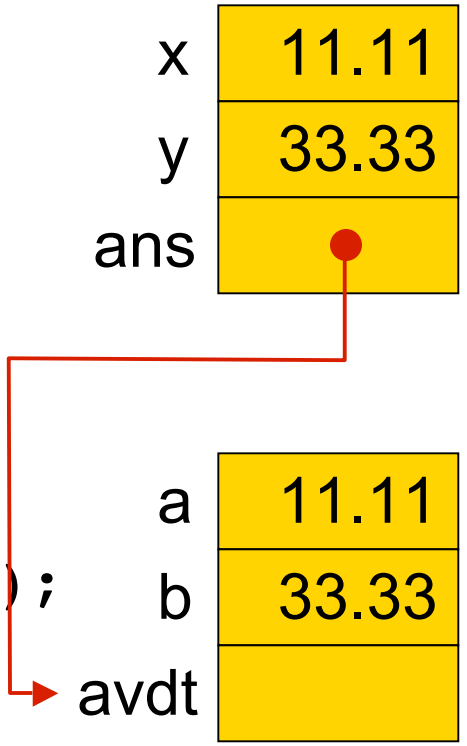
参照渡し(Pascal)

```
program average;
var a, b, avdt: real;

procedure ave_p(x, y: real; var ans: double);
var wk: real;
begin
    wk := (x + y) / 2.0;
    ans := wk
end;

begin
    a := 11.11;
    b := 33.33;
    ave_p(a, b, avdt);
    writeln('average=', avdt)
end.
```



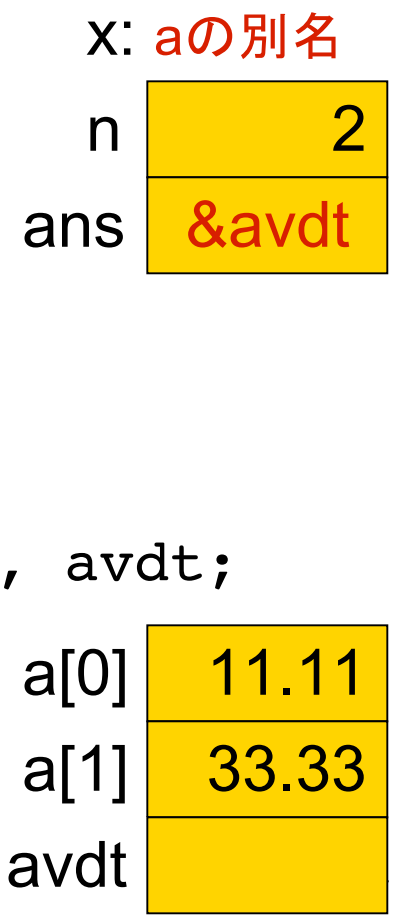
ポインタ引数(C)

```
#include <stdio.h>
void ave_p(double x, double y, double *ans)
{
    double wk;
    wk = (x + y) / 2.0;
    *ans = wk;
}
int main()
{
    double a, b, avdt;
    a = 11.11;
    b = 33.33;
    ave_p(a, b, &avdt);
    printf("average=%f\n", avdt);
    return 0;
}
```



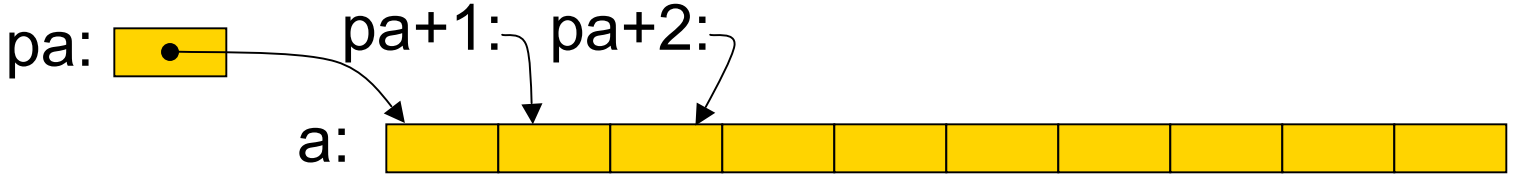
配列引数(Javaとの類似による説明)

```
#include <stdio.h>
void ave_p(double x[], int n, double *ans)
{
    double wk = 0.0;
    int i;
    for (i = 0; i < n; i++)
        wk += x[i];
    *ans = wk / n;
}
int main()
{
    double a[] = { 11.11, 33.33 }, avdt;
    ave_p(a, 2, &avdt);
    printf("average=%f\n", avdt);
    return 0;
}
```



ポインタと配列

- `int a[10], *pa;  pa = &a[0];` のとき、`pa+1`は`&a[1]`と等しい。



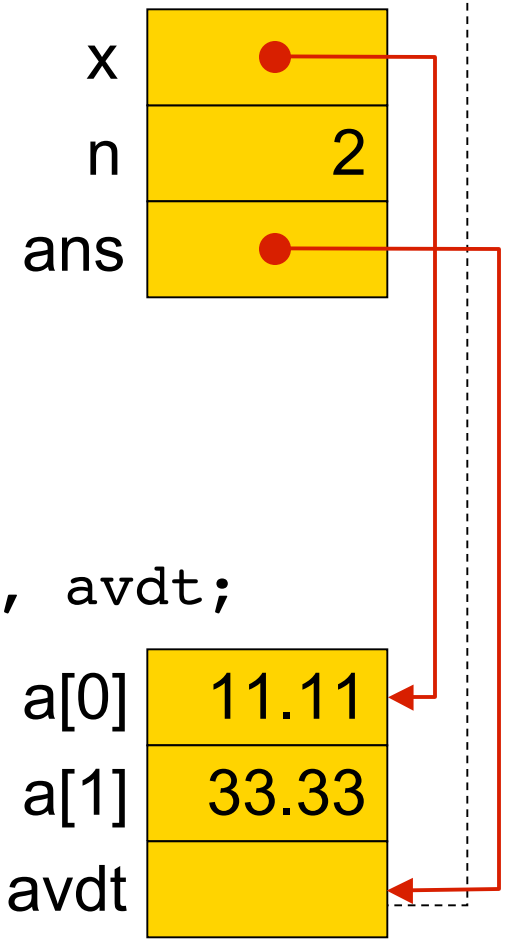
  - `for (i=0; a[i]>0; i++) …`
  - `for (pa=&a[0]; *pa>0; pa++) …`
- 配列名は、先頭要素のアドレスを表す。
  - `pa = a; /* pa=&a[0];`と同じ `*/`
- `a[i]` は `*(a+i)` と同じ。

配列引数(ポインタによる説明)

```
#include <stdio.h>
void ave_p(double x[], int n, double *ans)
{
    double wk = 0.0;
    int i;
    for (i = 0; i < n; i++)
        wk += x[i];
    *ans = wk / n;
}
int main()
{
    double a[] = { 11.11, 33.33 }, avdt;
    ave_p(a, 2, &avdt);
    printf("average=%f\n", avdt);
    return 0;
}
```

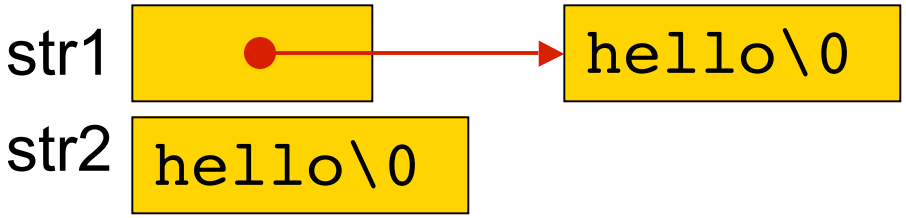
double *xと同じ

*(x+i)と同じ



文字列の型

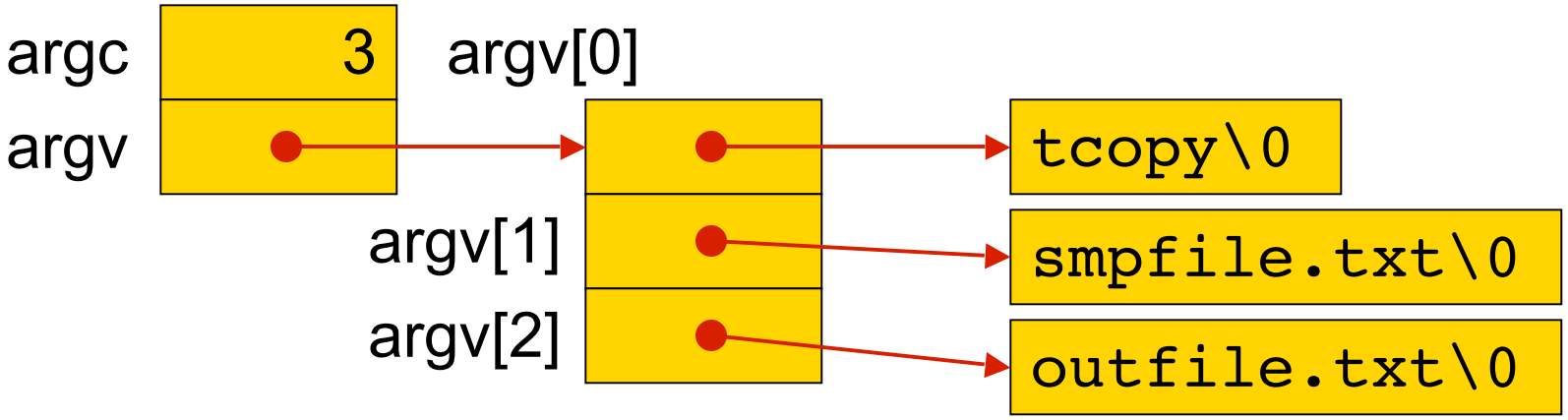
- char[] と char * はどちらも文字列型。
- char *str1 = "hello";
  char str2[] = "hello";
  - str1もstr2も文字列として使える。ただし、メモリ上の配置法は違う。



- 仮引数の場合は、char *str と char str[] は全く同じ。

コマンドライン引数

```
% tcopy smpfile.txt outfile.txt
```



ビット演算
通用範囲と記憶クラス
スタックフレーム

2007.10.22

情報システム工学実験第2

ビット演算

a	= 00005555	00000000 00000000 01010101 01010101
b	= 000000FF	00000000 00000000 00000000 11111111
a&b	= 00000055	00000000 00000000 00000000 01010101
a|b	= 000055FF	00000000 00000000 01010101 11111111
a^b	= 000055AA	00000000 00000000 01010101 10101010
~a	= FFFFAAAA	11111111 11111111 10101010 10101010

課題（ビット演算）

- 利用者に文字列を入力させ、その各文字について
  - 文字コード（16進法）
  - 文字コードを2進法で表したときに値が1である桁の数

  を表示するプログラムを作りなさい

```
% ./nbit
Enter a string: hello
h 68 3
e 65 4
l 6C 4
l 6C 4
o 6F 6
```

変数の通用範囲(scope)

```
#include <stdio.h>
int n; /* 大域変数 */
void fact(int a)
{
    if (a < 2) n = 1;
    else       n = fact(a-1) * a;
}
int main()
{
    int i;  /* 関数内局所変数 */
    for (i = 0; i < 5; i++) {
        int j;  /* ブロック内局所変数 */
        f(i); for (j = 0; j < i; j++) n += 1;
        printf("%d\n", n);
    }
    return 0;
}
```

変数の記憶クラス

```
#include <stdio.h>
int n; /* 静的な変数 */
void fact(int a)
{
    if (a < 2) n = 1;
    else       n = fact(a-1) * a;
}
int main()
{
    int i;  /* 自動変数 */
    for (i = 0; i < 5; i++) {
        int j;  /* 自動変数 */
        f(i); for (j = 0; j < i; j++) n += 1;
        printf("%d\n", n);
    }
    return 0;
}
```

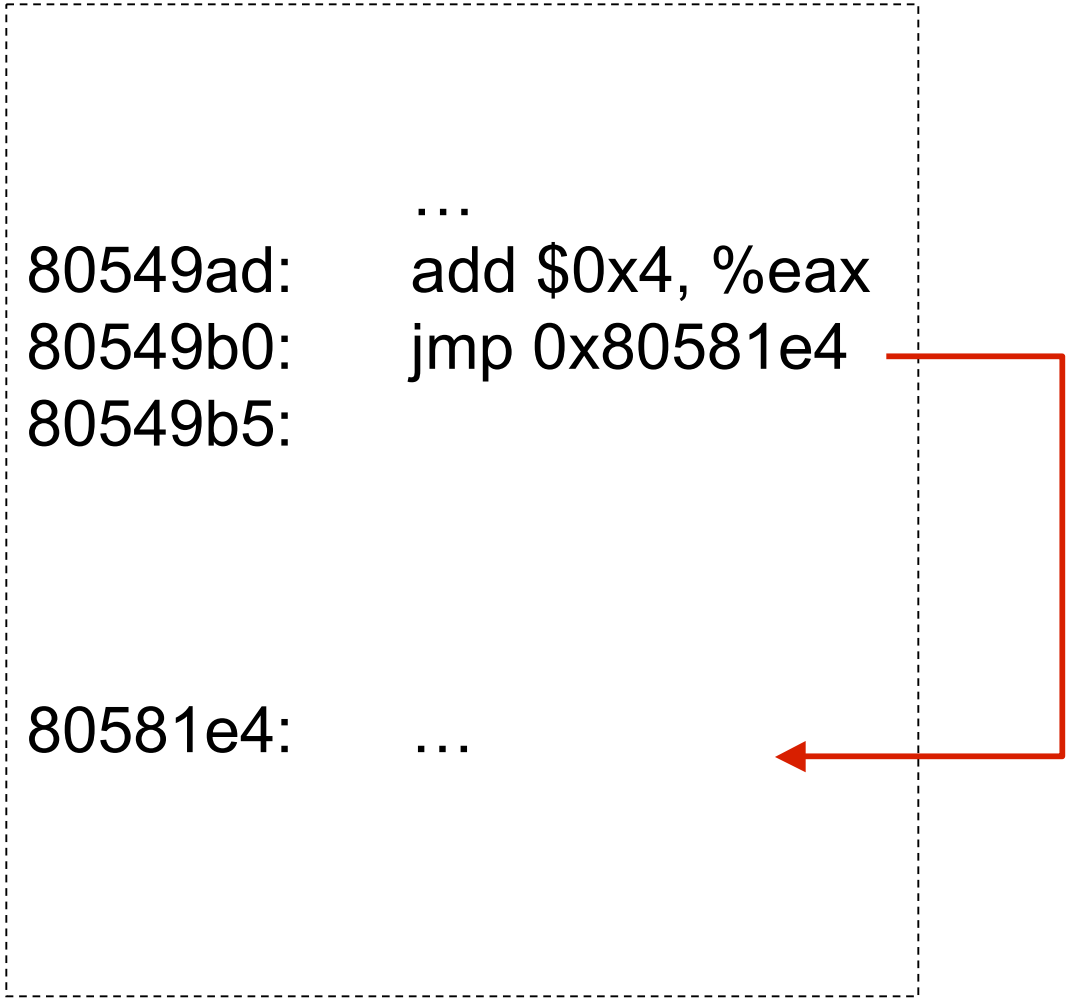
変数の記憶クラス

```
#include <stdio.h>
int n; /* 静的な変数 */
void fact(int a)
{
    if (a < 2) n = 1;
    else       n = fact(a-1) * a;
}
int main()
{
    static int i;  /* 静的な変数 */
    for (i = 0; i < 5; i++) {
        int j;  /* 自動変数 */
        f(i); for (j = 0; j < i; j++) n += 1;
        printf("%d\n", n);
    }
    return 0;
}
```

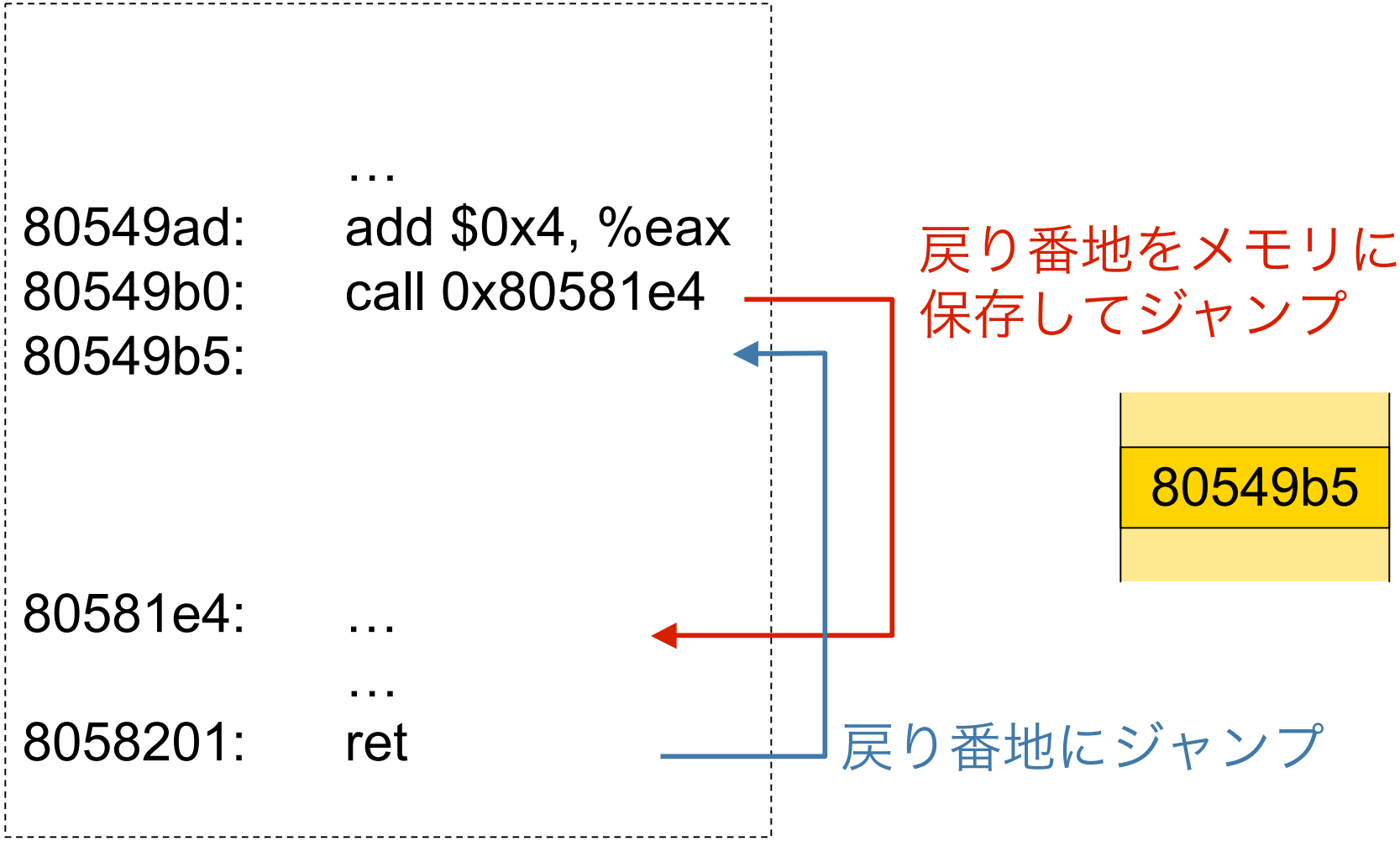
Q. 各変数の通用範囲と記憶クラスは?

```
#include <stdio.h>
void f(int x);
int n1, n2, wa, sa;
int main()
{
    int x = 50;
    n1 = 300;
    n2 = 100;
    f(x);
    printf("%d,%d,%d,%d\n", n1,n2,wa,sa);
    return 0;
}
void f(int x)
{
    wa = n1 + n2 + x;  sa = n1 - n2 + x;
}
```

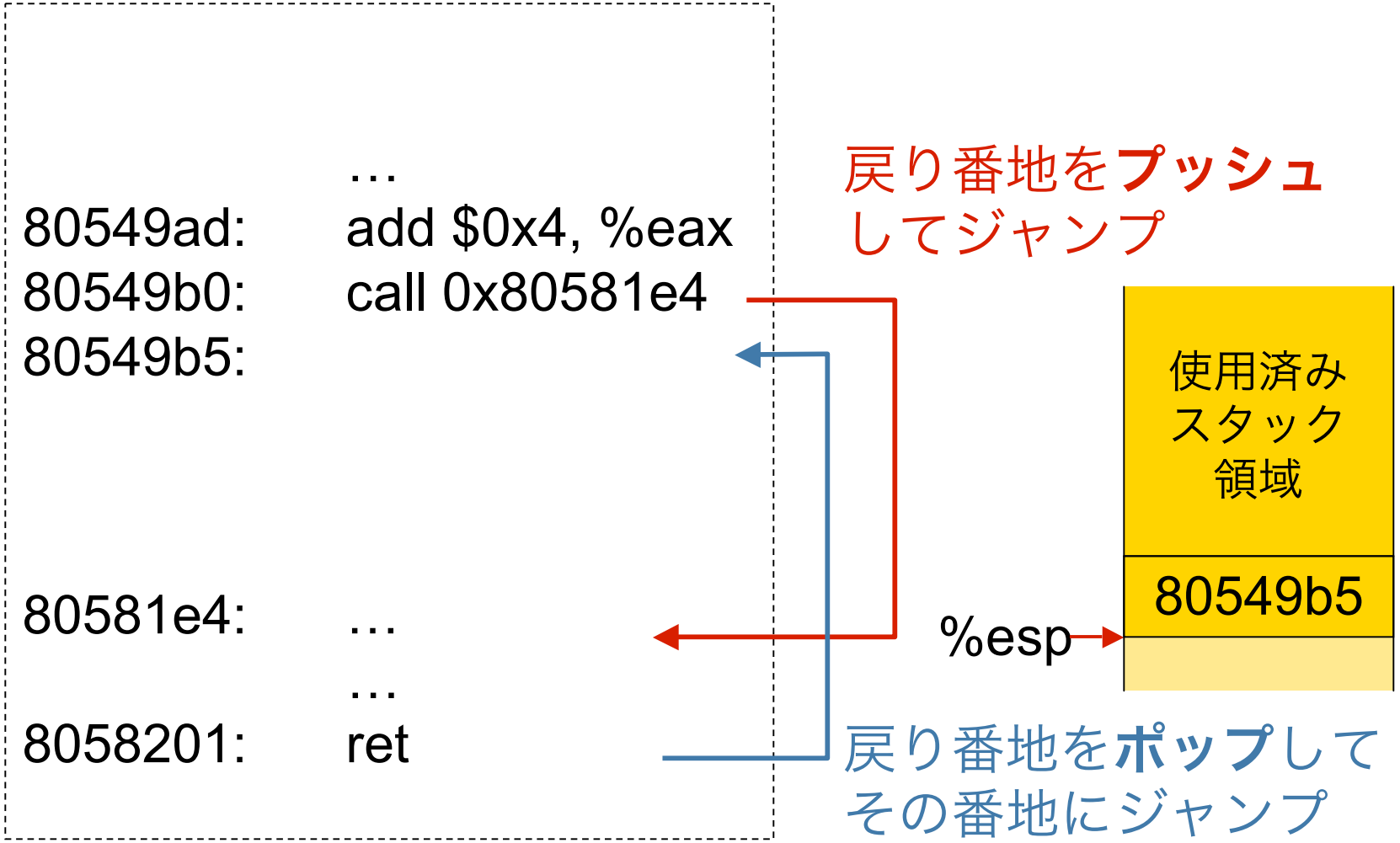
ジャンプ (gotoと類似)



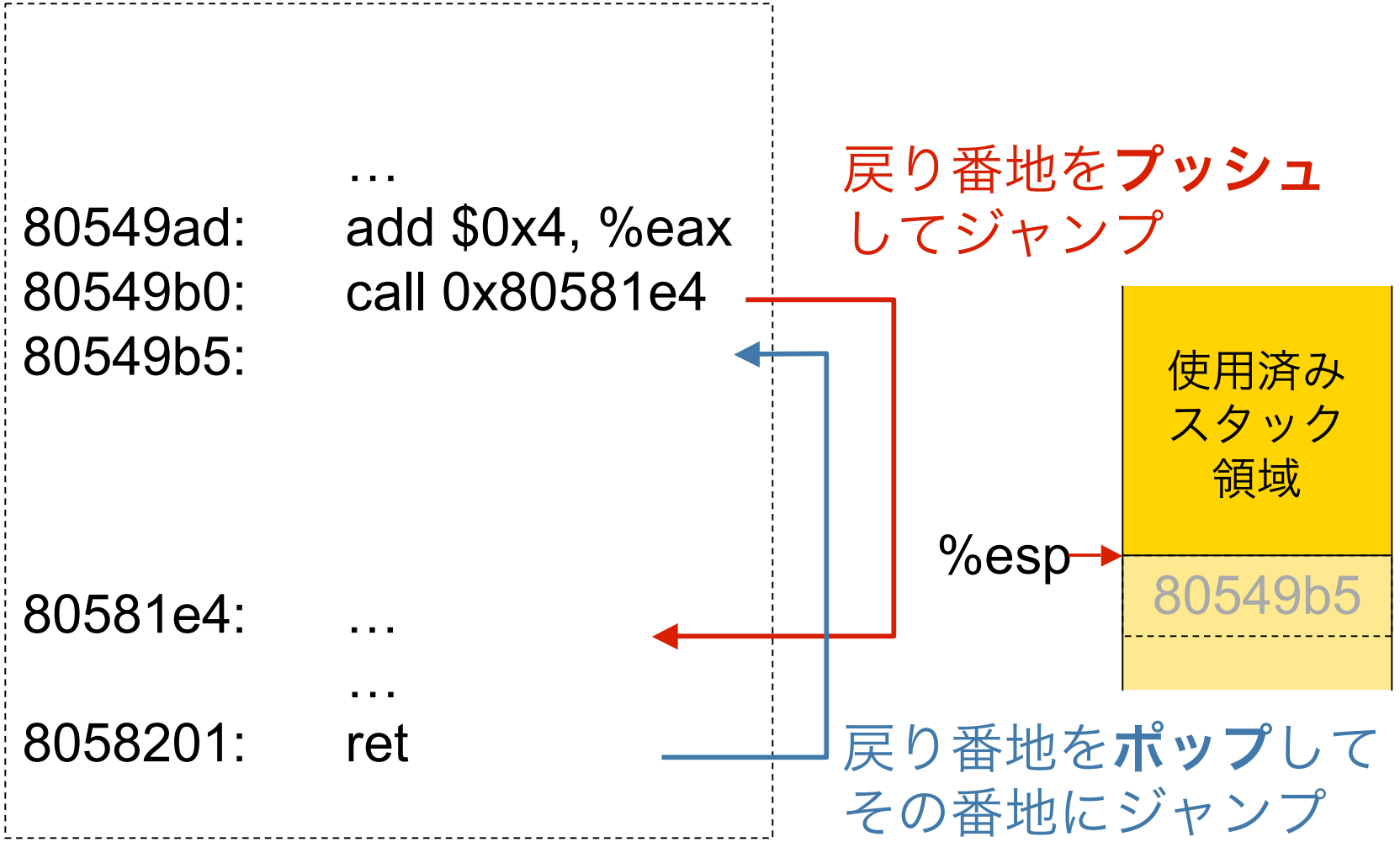
呼び出し(call)と復帰(ret)



呼び出し制御スタック

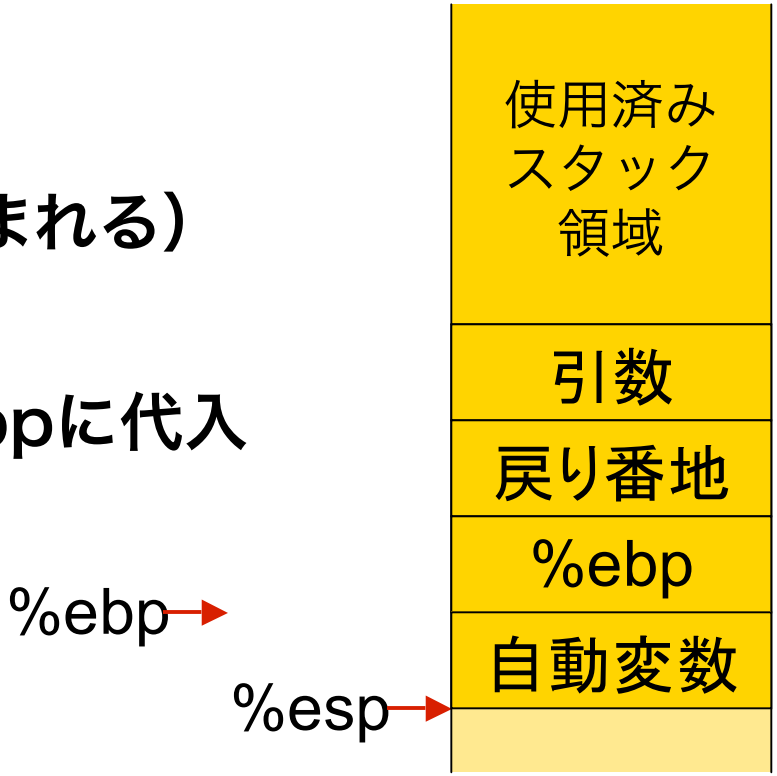

…
80549ad: add $0x4, %eax
80549b0: call 0x80581e4
80549b5:
80581e4: …
…
8058201: ret
戻り番地を**プッシュ**してジャンプ
使用済みスタック領域
80549b5
%esp
戻り番地を**ポップ**してその番地にジャンプ

呼び出し制御スタック


…
80549ad: add $0x4, %eax
80549b0: call 0x80581e4
80549b5:
80581e4: …
…
8058201: ret
戻り番地を**プッシュ**してジャンプ
使用済みスタック領域
%esp
80549b5
戻り番地を**ポップ**してその番地にジャンプ

スタックフレームの構築・解放

- 呼び出し元：
  1. 実引数をスタックに積む
  2. callを実行（戻り番地が積まれる）
- 呼び出された関数：
  3. bpを保存し、現在のspをbpに代入
  4. 自動変数領域を確保
  5. spとbpを復元
  6. retを実行
- 呼び出し元：
  7. 実引数領域を解放

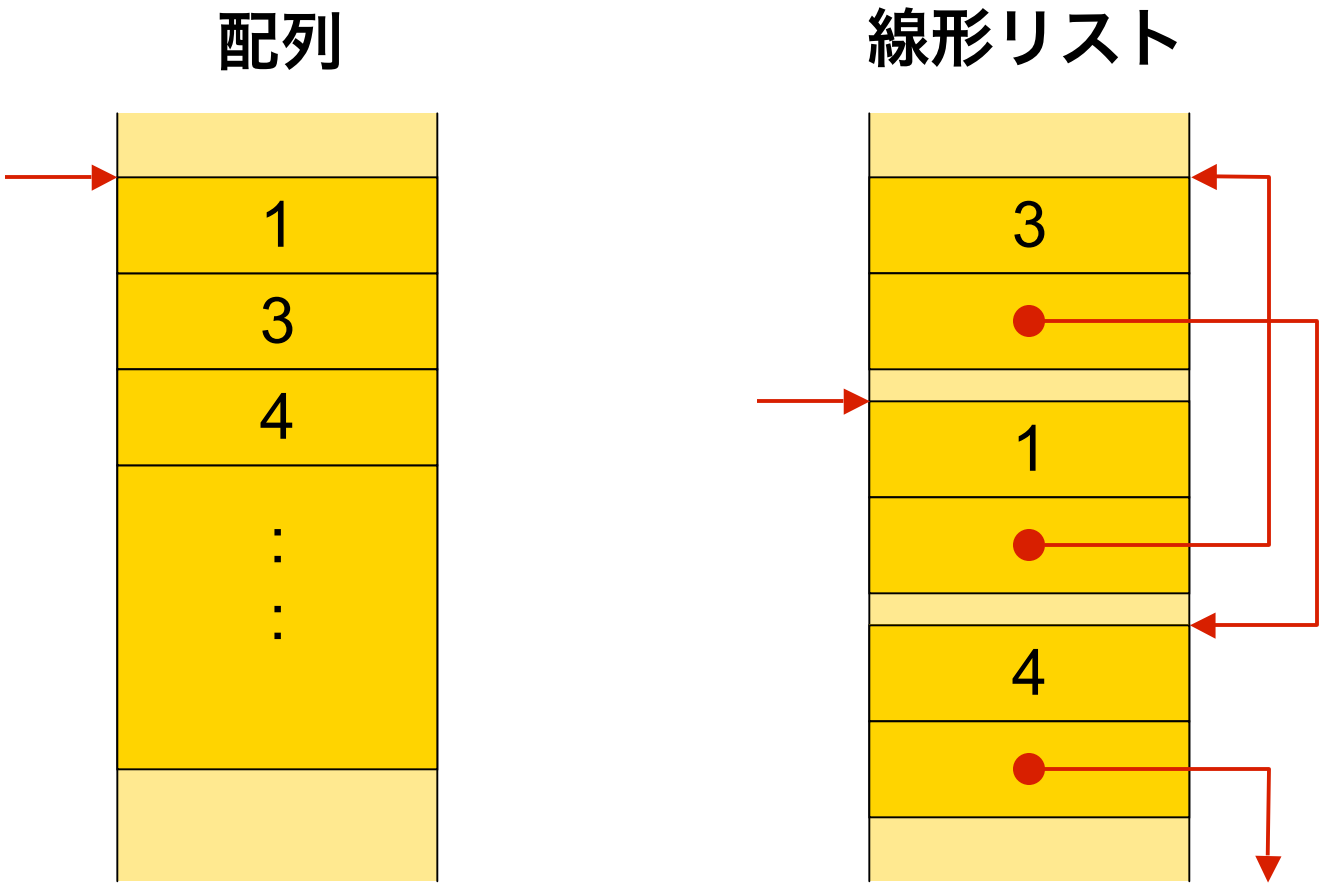


配列と線形リスト
低水準入出力

2007.10.29

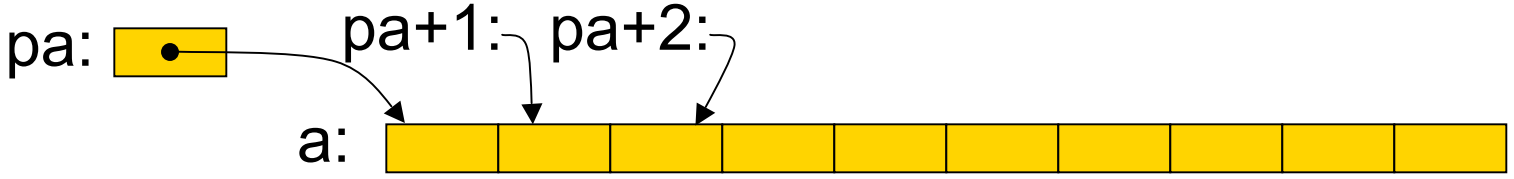
情報システム工学実験第2

配列と線形リスト



ポインタと配列（第5回）

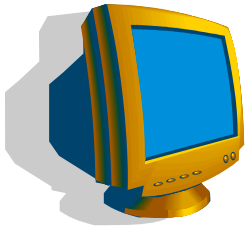
- `int a[10], *pa;  pa = &a[0];` のとき、`pa+1`は`&a[1]`と等しい。



  - `for (i=0; a[i]>0; i++) …`
  - `for (pa=&a[0]; *pa>0; pa++) …`
- 配列名は、先頭要素のアドレスを表す。
  - `pa = a; /* pa=&a[0];`と同じ `*/`
- `a[i]` は `*(a+i)` と同じ。

ファイルの一般化

すべての入出力は
ファイルの読み書き
によって行われる

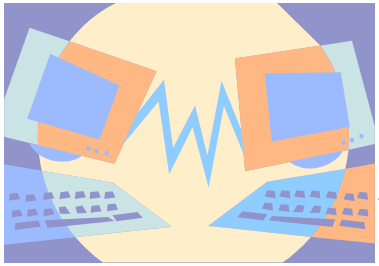


ディスプレイ

キーボード

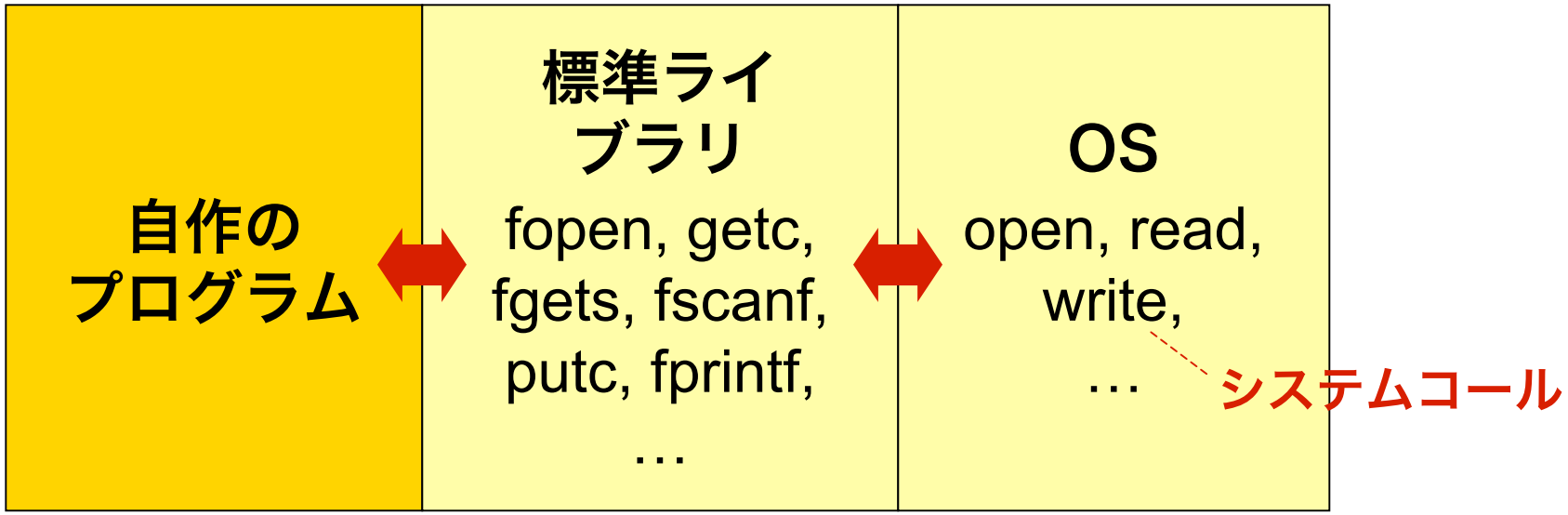
プログラム

ディスク上の
ファイル

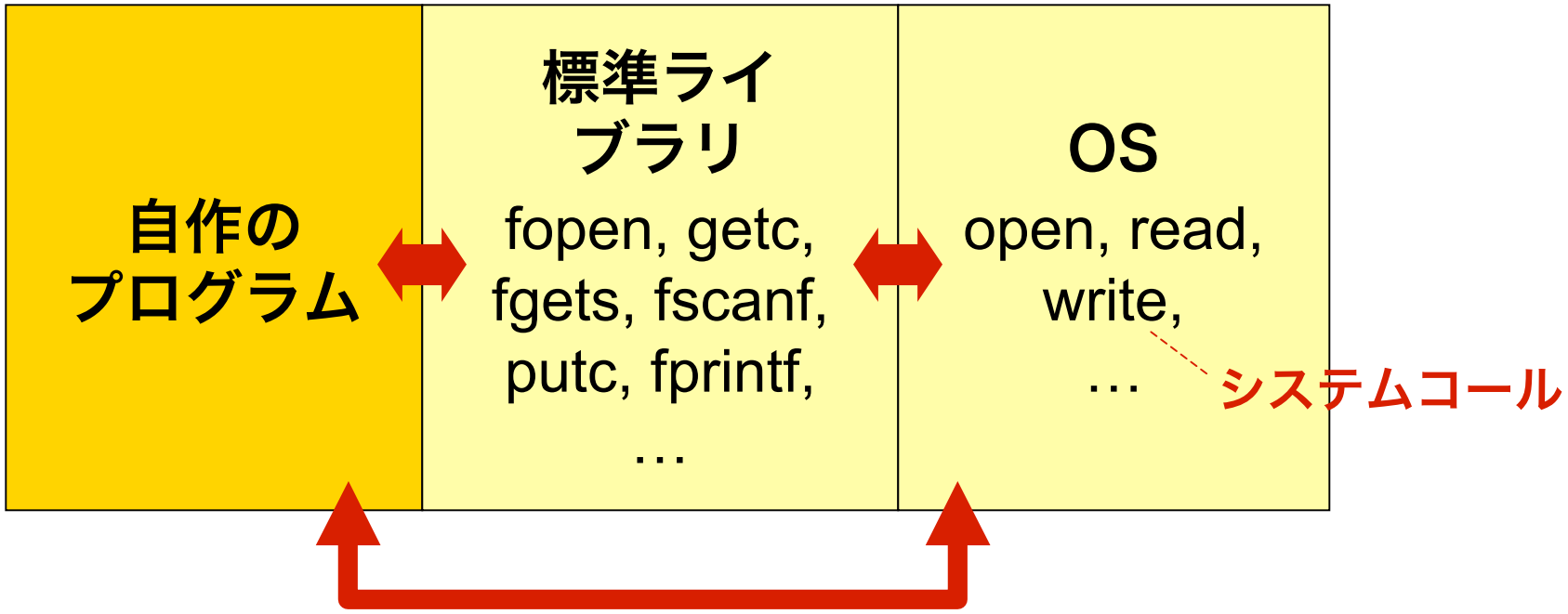


ネットワーク

低水準入出力


自作の
プログラム
標準ライ
ブラリ
fopen, getc,
fgets, fscanf,
putc, fprintf,
…
OS
open, read,
write,
…
システムコール

低水準入出力


自作の
プログラム
標準ライ
ブラリ
fopen, getc,
fgets, fscanf,
putc, fprintf,
…
OS
open, read,
write,
…
システムコール

追加課題

- 課題a: コマンドラインで指定したファイルの大きさ（バイト数）を表示するプログラムを作りなさい。ただし、外部の関数としてopenとread（とperror）のみ使うこと。

- 課題b: コマンドラインで文字列sとファイル名fを指定させ、f中の行のうち文字列sを含むものをすべて表示するプログラムを作りなさい（grepと類似）。ただし、低水準入出力（とperror）だけを使って作りなさい。