

PHPのfloat型やら int型やらの話

id:hnw

y at hnw dot jp

PHP勉強会 SIDE-B (第1回) 発表資料

と、発表の前に…

- アンケートにご協力ください
 - 勉強会初参加の方！
 - プレゼン未経験の方！

よからさそ!

ようこそ！

(主催者でも何でもありませんが)

勉強会ってナニ？

勉強会ってナニ？

- 主催者や発表者って何が楽しいの？
- 参加するまで僕もそう思っていました

勉強会ってナニ？

- 主催者や発表者って何が楽しいの？
- 参加するまで僕もそう思っていました
 - 「理解できない→気持ち悪い」

勉強会ってナニ？

- 主催者や発表者って何が楽しいの？
- 参加するまで僕もそう思っていました
 - 「理解できない→気持ち悪い」
- 初参加の人にその辺を伝えたい

勉強会のメリット

勉強会のメリット

- 新たな知識の獲得
 - 強制力
 - 自分のアンテナ外のこと聞ける

勉強会のメリット

- 新たな知識の獲得
 - 強制力
 - 自分のアンテナ外のこと聞ける

実はもっと凄いメリットがある！

すごいメリット(1)

- 懇親会！
 - 酔うと技術話が盛り上がる
 - 技術者はシャイなので酒重要
 - オフレコ話
 - 単純に楽しい。有意義なことも。

すごいメリット(2)

- 発表者にもメリット
 - 自分を知ってもらえる
 - 懇親会でいじられるネタ提供
 - 周辺知識を教えてもらったり
 - 1vs多→フィードバックの期待値↑

まとめると

- 他人への影響大→フィードバック大
 - 発表者のメリットが最大
 - 騙されたと思って一度発表側に！
 - 自分が面白いことを発表すればOK

自己紹介

- id:hnw
- カレー好き
 - 先週は週4カレー

自己紹介

The screenshot shows a Hatena Diary page. At the top, there is a navigation bar with 'Hatena::Diary' on the left, a search box, and buttons for '日記' (Diary) and '検索' (Search). On the right side of the navigation bar are links for '最新の日記' (Latest diary), '記事一覧' (List of articles), '日記を書く' (Write diary), '管理' (Management), 'ログアウト' (Logout), and 'ヘルプ' (Help). Below the navigation bar is a dark red header area with the text 'hnwの日記' (hnw's diary) and icons for 'AI' and 'RSS'. Underneath the header, there are links for '<前の日 | 次の日>' (Previous day | Next day) and '[プロフィール]' (Profile). The main content area features a dotted line separator above the article title '2007年5月15日 (火) PHPの奇妙なround関数'. To the right of the title are buttons for '編集' (Edit), 'tB', '92 users', and icons for comments and social media. The article text begins with 'さて、プログラミングの話題もたまには書いてみます。今回はPHPのround関数の挙動が変だ！という話題です。' (Well, I'll write about programming topics from time to time. This time it's about the behavior of PHP's round function being weird!). The next paragraph says 'round()は浮動小数点数を四捨五入する関数で、大抵の言語に同じ名前を実装されているかと思いますが。ではPHPのround関数の何が問題なのか、ちょっと試してみましよう。' (round() is a function that rounds floating-point numbers, and it's implemented with the same name in most languages, but I think. Well, what's the problem with PHP's round function, let's try it out a little). At the bottom, there is a code block showing a terminal command and its output: '\$ uname -sro' followed by 'Linux 2.6.9-42.0.10.plus.c4smp GNU/Linux'.

↑ PHPのround関数についての記事

自己紹介

- @halt(Ethna、vimの人)
- @hiro_y(moonyの人)
- @hnw(round())の人)
- @ichii386(Ethnaの人)
- @iogi(PHP Extension勉強会の人)
- @iteman(Piece Frameworkの人)
- @junya(Sooey、guessworkの人)
- @kensuu(ミルフィールの人)
- @komagata(plnet.jpの人)
- @koyhoge(とにかくすごい人)

↑ 謎の紹介文

自己紹介

- @halt(Ethna、vimの人)
- @hiro_y(moonyの人)
- @hnw(round())の人
- @ichii386(Ethnaの人)
- @iogi(PHP Extension勉強会の人)
- @iteman(Piece Frameworkの人)
- @junya(Sooey、guessworkの人)
- @kensuu(ミルフィールの人)
- @komagata(plnet.jpの人)
- @koyhoge(とにかくすごい人)

↑ 謎の紹介文

自己紹介

○ 2008年06月26日  [TAKESAKO](#) php ミスター浮動小数点   

↑ 謎のブックマコメント

自己紹介

○ 2008年06月26日  [TAKESAKO](#) php **ミスター浮動小数点**  ★★

↑ 謎のブックマコメント

こんな相談を受けた

- 「プログラム中で小数は使わない方がいい」って聞きました。なぜですか？
(千葉県・20歳・プログラマー)

ようやく

今日の

本題です

- intとfloat
- PHPでの実例
- その他の注意点
- まとめ

PHPの数値型

- int型
 - 32bit整数
- float型
 - 64bit浮動小数点数

→PHP以外とも共通の道具

しばらく一般論

CPUについて

- 使っているCPU：32bitアーキテクチャ
 - 整数レジスタ長=32bit
 - 32bit整数同士の加減乗除は1命令

レジスタって？

- 計算結果の置き場。数十個程度？
 - 変数の実体みたいなもの
- マシン語レベルでは頻出
 - レジスタ2個の和→レジスタ、など
- CPUから見て最速のデータ格納領域

言語における整数

- intのサイズはアーキテクチャ都合
 - CPUにとって最適なサイズ
 - 最近のCPUだと32bit
 - そのうち64bitになるのかも

CPUと浮動小数点数

- 64bit浮動小数点数
 - IEEE 754で決まっている
- 浮動小数点数用に別レジスタ
- CPUの1命令で加減乗除できる

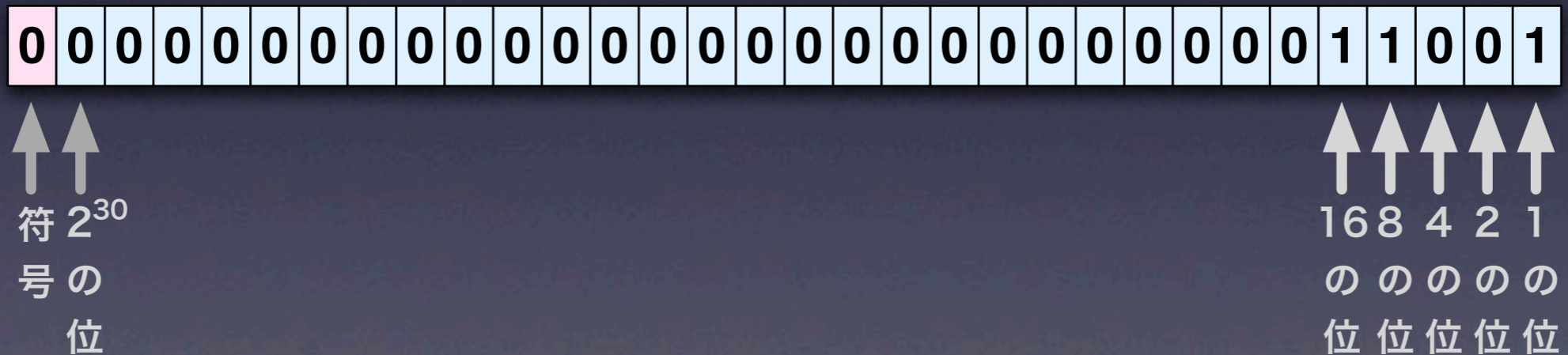
intもfloatも

CPUの都合で

決まっっている

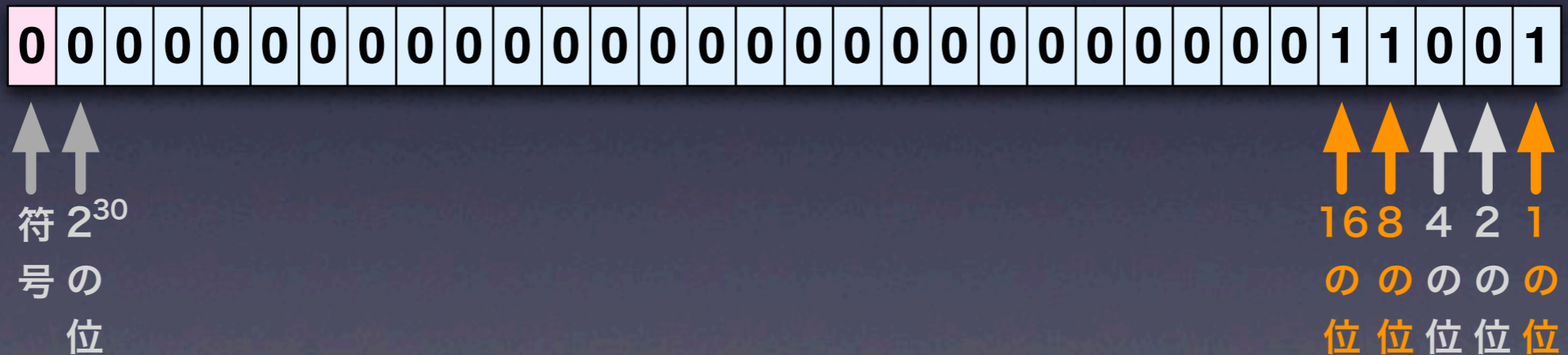
ビットパターン

- intの場合、符号が1bit/残りが数字
 - $-2^{31} \sim 2^{31} - 1$ まで扱える。
 - 例： $25 = 16 + 8 + 1$



ビットパターン

- intの場合、符号が1bit / 残りが数字
 - $-2^{31} \sim 2^{31} - 1$ まで扱える。
 - 例: $25 = 16 + 8 + 1$

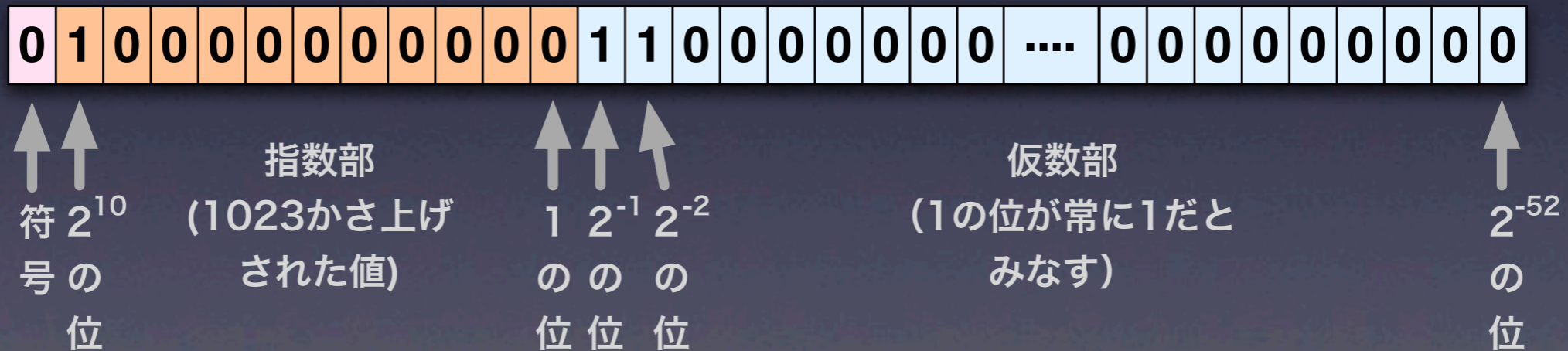


ビットパターン

- floatの場合

- 符号1bit / 指数部11bit / 仮数部52bit

- 例： $3.5 = 1.75 \times 2^{(1024-1023)}$



- intとfloat
- PHPでの実例
- その他の注意点
- まとめ

小数のトラブル

```
ringo:~ php -r 'var_dump(0.5+0.5);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.1+0.1+0.1+0.1+0.1);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.5==0.5+0.1+0.1+0.1+0.1+0.1);'  
bool(false)  
ringo:~
```

- 両方1に見えるけど実は違います。
 - var_dumpは不正確。

小数のトラブル

```
ringo:~ php -r 'var_dump(0.5+0.5);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.1+0.1+0.1+0.1+0.1);'  
float(1) 一見同じ値だけど...  
ringo:~ php -r 'var_dump(0.5+0.5==0.5+0.1+0.1+0.1+0.1+0.1);'  
bool(false)  
ringo:~
```

- 両方1に見えるけど実は違います。
 - var_dumpは不正確。

小数のトラブル

```
ringo:~ php -r 'var_dump(0.5+0.5);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.1+0.1+0.1+0.1+0.1);'  
float(1)  
ringo:~ php -r 'var_dump(0.5+0.5==0.5+0.1+0.1+0.1+0.1+0.1);'  
bool(false)  
ringo:~
```

違う！

- 両方1に見えるけど実は違います。
 - var_dumpは不正確。

小数のトラブル

```
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.5))));'  
string(16) "3ff0000000000000"  
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.1+0.1+0.1+0.1+0.1))));'  
string(16) "3fefffffffffffffff"  
ringo:~ █
```


小数のトラブル

```
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.5))));'  
string(16) "3ff0000000000000"  
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.1+0.1+0.1+0.1+0.1))));'  
string(16) "3fefffffffffffffff"  
ringo:~ █
```

- 0.1を10回足しても1にならない！

小数のトラブル

```
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.5))));'  
string(16) "3ff0000000000000"  
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.1+0.1+0.1+0.1+0.1))));'  
string(16) "3fefffffffffffffff" 確かに違う  
ringo:~
```

- 0.1を10回足しても1にならない！

小数のトラブル

```
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.5))));'  
string(16) "3ff0000000000000"  
ringo:~ php -r 'var_dump(bin2hex(strrev(pack("d", 0.5+0.1+0.1+0.1+0.1+0.1))));'  
string(16) "3fefffffffffffffff" 確かに違う  
ringo:~
```

- 0.1を10回足しても1にならない！
 - 0.1は2進数では不正確

0.1が不正確？

- 分数と小数の関係
 - $1/3$ を10進小数で表そうとすると0.33333...と無限桁になる
 - 同様に、 $0.1=1/10$ を2進小数で表そうとすると無限桁になる

小数の処理

- 誤差が蓄積することがある
 - 対策は？

小数を扱う(1)

- 整数で扱えないか？
 - 例：小数点以下第2位まで表示
 - 内部的には100倍して計算
 - 出力時に1/100して表示

小数を扱う(2)

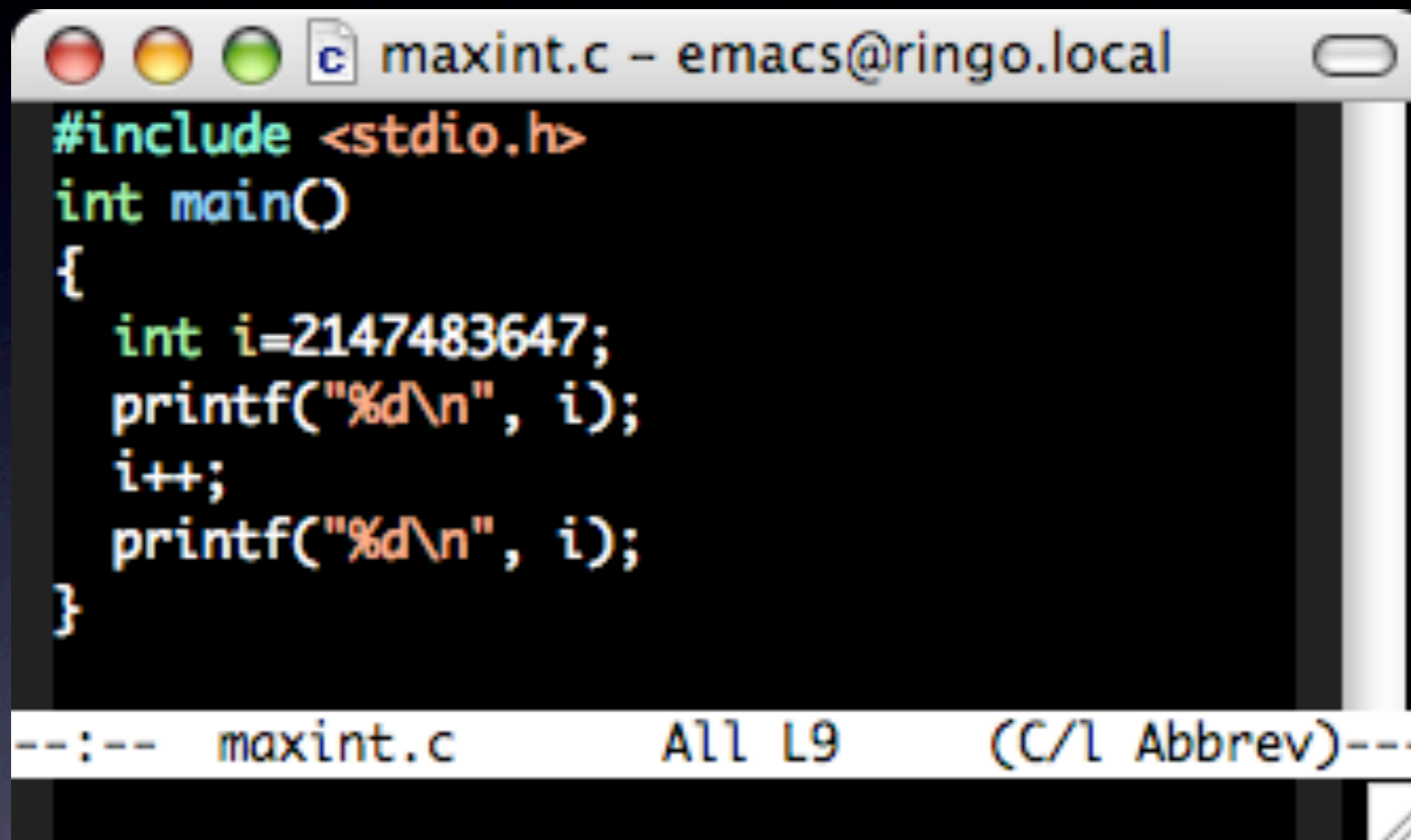
- 誤差を真面目に計算する
 - 任せられる人が居て、
情報共有も問題なければ。
 - 浮動小数点数を扱う限り
誤差の話題からは逃れられない

- intとfloat
- PHPでの実例
- その他の注意点
- まとめ

intを超えた場合

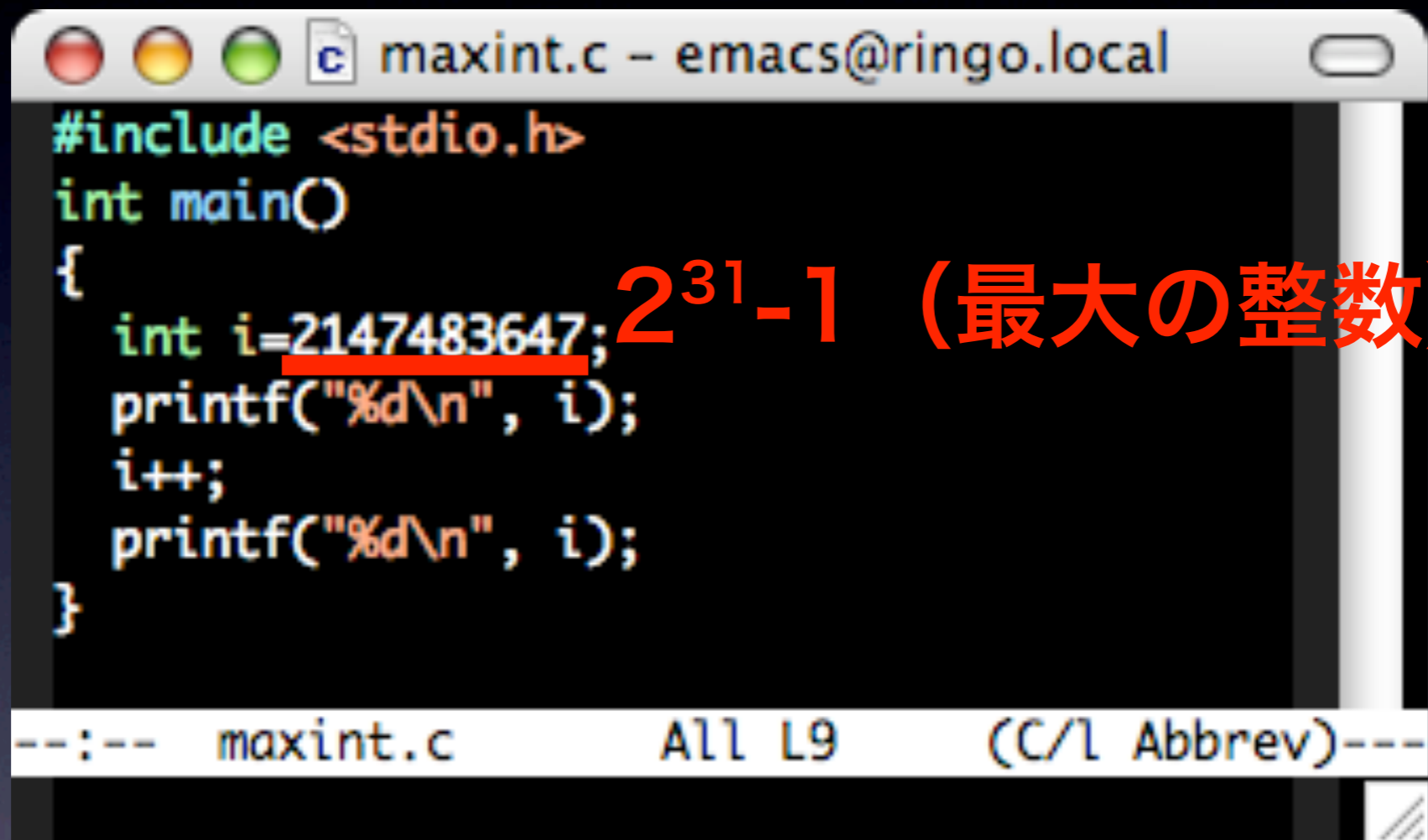
- intには範囲があります。
- Q：範囲外の大きさになったら
何が起こる？

例：Cの場合



```
maxint.c - emacs@ringo.local
#include <stdio.h>
int main()
{
    int i=2147483647;
    printf("%d\n", i);
    i++;
    printf("%d\n", i);
}
--:-- maxint.c All L9 (C/l Abbrev)---
```

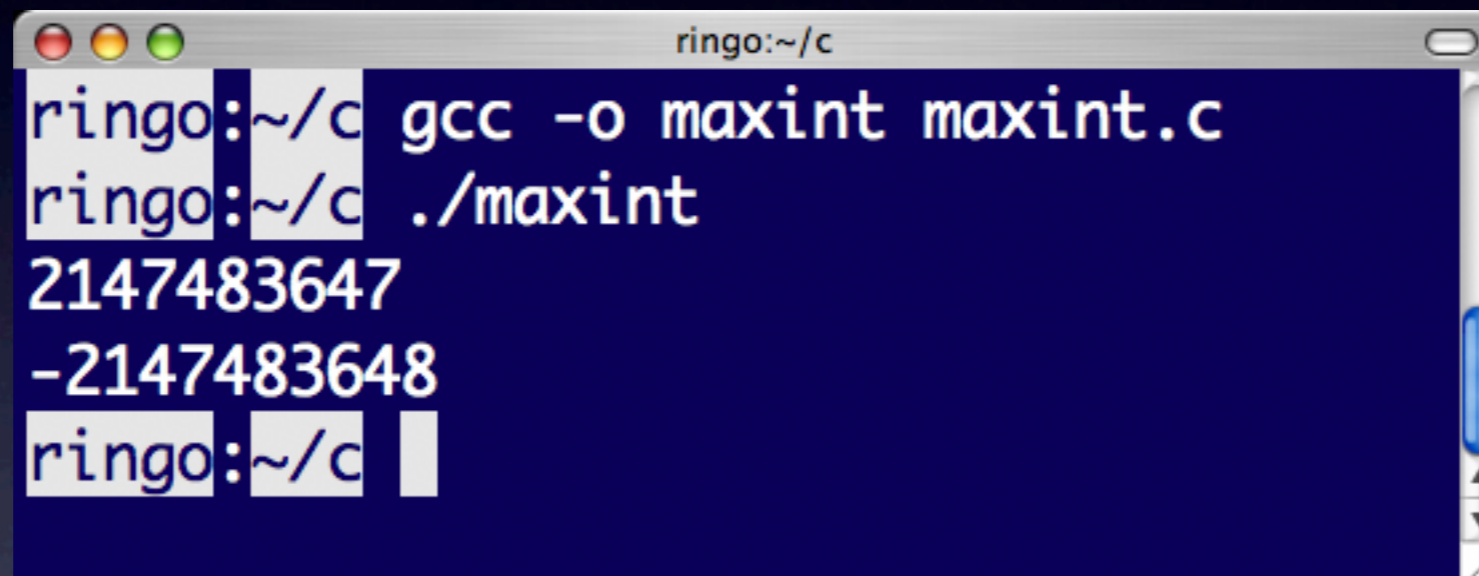
例：Cの場合



```
maxint.c - emacs@ringo.local
#include <stdio.h>
int main()
{
  int i=2147483647;
  printf("%d\n", i);
  i++;
  printf("%d\n", i);
}
--:-- maxint.c All L9 (C/l Abbrev)---
```

- 最大のintをインクリメント

例：Cの場合

A terminal window with a dark blue background and white text. The window title is "ringo:~/c". The terminal shows the following commands and output:

```
ringo:~/c gcc -o maxint maxint.c
ringo:~/c ./maxint
2147483647
-2147483648
ringo:~/c
```

例：Cの場合



```
ringo:~/c gcc -o maxint maxint.c
ringo:~/c ./maxint
2147483647
-2147483648 -231 (最小の整数)
ringo:~/c
```

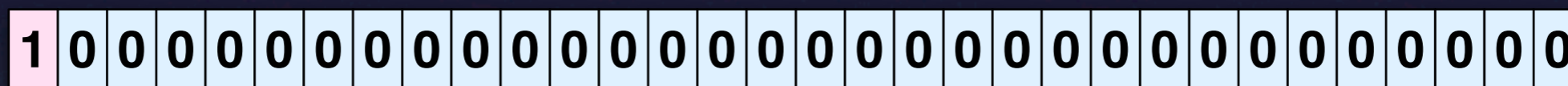
- 最小のintになる

どうということ？

$$2^{31} - 1$$



$$-2^{31}$$



↑↑
符号の位
2³⁰

↑↑↑↑↑
16 8 4 2 1
の の の の の
位 位 位 位 位

- ・ 繰り上がって符号が反転しただけ

PHPの場合

```
ringo:~ php -r 'var_dump(2147483647);'  
int(2147483647)  
ringo:~ php -r 'var_dump(2147483647+1);'  
float(2147483648)  
ringo:~ █
```

- floatになります。
- 数としては（まだ）正確です。

PHPの場合

```
ringo:~ php -r 'var_dump(2147483647);'  
int(2147483647)  
ringo:~ php -r 'var_dump(2147483647+1);'  
float(2147483648)  
ringo:~
```

$2^{31}-1$ (最大の整数)

- floatになります。
- 数としては (まだ) 正確です。

PHPの場合

```
ringo:~ php -r 'var_dump(2147483647);'  
int(2147483647)  
ringo:~ php -r 'var_dump(2147483647+1);'  
float(2147483648)  
ringo:~ floatになった
```

- floatになります。
- 数としては（まだ）正確です。

PHPの場合

```
ringo:~ php -r 'var_dump(9007199254740991==9007199254740992);'  
bool(false)  
ringo:~ php -r 'var_dump(9007199254740993==9007199254740992);'  
bool(true)  
ringo:~ █
```

- 2^{53} あたりから不正確
- 途中計算にも注意が必要
- 仕事で扱うには色々怖いですね

PHPの場合

```
ringo:~ php -r 'var_dump(9007199254740991==9007199254740992);'  
bool(false)  
ringo:~ php -r 'var_dump(9007199254740993==9007199254740992);'  
bool(true)  
ringo:~
```

- 2^{53} あたりから不正確
- 途中計算にも注意が必要
- 仕事で扱うには色々怖いですね

PHPの場合

```
ringo:~ php -r 'var_dump(9007199254740991==9007199254740992);'  
bool(false) 253-1  
ringo:~ php -r 'var_dump(9007199254740993==9007199254740992);' 253  
bool(true)  
ringo:~
```

- 2^{53} あたりから不正確
- 途中計算にも注意が必要
- 仕事で扱うには色々怖いですね

PHPの場合

```
ringo:~ php -r 'var_dump(9007199254740991==9007199254740992);'  
bool(false)  
ringo:~ php -r 'var_dump(9007199254740993==9007199254740992);'  
bool(true)  
ringo:~ 同じ数とみなされている
```

- 2^{53} あたりから不正確
- 途中計算にも注意が必要
- 仕事で扱うには色々怖いですね

- intとfloat
- PHPでの実例
- その他の注意点
- まとめ

まとめ

- PHPのint型 = Cのint型
- PHPのfloat型 = Cのdouble型
 - 10進小数の常識が通用しない世界
- 大きすぎる数も注意
 - 無限に正確に表せるわけではない

ご清聴
ありがとうございます
ございました