I117（7）時刻・時間

知念

北陸先端科学技術大学院大学 情報科学研究科
School of Information Science,
Japan Advanced Institute of Science and Technology

時刻取得 time

- 標準ライブラリでは時刻を整数で扱う
- 専用の time_t という型を用意
  - 多くの実装は long

```
typedef long    time_t;
```

  - 将来変更される可能性があるので注意
- 1970/01/01 0:00:00 を起源とする秒数
- 時刻取得は関数 time を用いる

```
        t = time(NULL);
```

現在時刻表示プログラム

```
#include <stdio.h>
#include <time.h>
int main() {
    time_t now = time(NULL);
    printf("%d\n", now);
    printf("%s", ctime(&now));
}
```

結果

```
1213440427
Sat Jun 14 19:47:07 2008
```

時間

- 時間は時刻の差で算出する

$$\Delta t = t_{\mathrm{end}} - t_{\mathrm{begin}}$$

```
tdiff = tend - tbegin
```

関数 sleep の処理時間を計測する

- 関数 sleep は指定された秒数だけ実行を中断する

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
int main() {
    time_t tbegin, tend, tdiff; int x;
    srand(time(NULL)%53);
    x = rand()%7+1;
    tbegin = time(NULL);
    sleep(x);
    tend = time(NULL);
    tdiff = tend - tbegin;
    printf("pland %ds, takes %ds\n",
        x, tdiff);  }
```

時間 *(cont.)*

実行結果 — 乱数を伴うので何度か実行する

```
% ./a.out
pland 7s, takes 7s
% ./a.out
pland 2s, takes 2s
% ./a.out
pland 4s, takes 4s
```

1秒から7秒の乱数分 sleep している
秒単位の処理を秒単位で計測するのは心もとない

時刻計測 gettimeofday

- gettimeofday で高い精度で時刻を計測可能

```
struct timeval t;
gettimeofday(&t, NULL);
```

- 表現能力はマイクロ秒 [us]

```
struct {
    long tv_sec;
    long tv_usec;
} timeval;
```

- しかし実質精度はミリ秒 [ms] 程度

時刻計測 gettimeofday *(cont.)*

- 精度の制約
  - ハードウェア能力
  - OS のコンテキストスイッチ
  - そのホストの負荷
  - 関数自体の処理時間
- timeval からミリ秒で差分を取り出す

```
#define SUB_MSEC(x,y) \
    ( (x.tv_sec  - y.tv_sec )*1000 \
    + (x.tv_usec - y.tv_usec)/1000 )
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <sys/time.h>
#define SUB_MSEC(x,y) \
        ( (x.tv_sec  - y.tv_sec )*1000 \
        + (x.tv_usec - y.tv_usec)/1000 )
int main() {
    struct timeval tbegin, tend;
    int tdiff, x;

    srand(time(NULL)%53);
    x = rand()%7+1;
```

```
    gettimeofday(&tbegin, NULL);
    sleep(x);
    gettimeofday(&tend, NULL);

    printf("tbegin %d.%06d\n",
        tbegin.tv_sec, tbegin.tv_usec);
    printf("tend   %d.%.6d\n",
        tend.tv_sec, tend.tv_usec);
    tdiff = SUB_MSEC(tend, tbegin);
    printf("planed %3ds %dms\n", x, x*1000);
    printf("takes       %dms\n", tdiff);

    exit(0);
}
```

時刻計測 **gettimeofday** ***(cont.)***

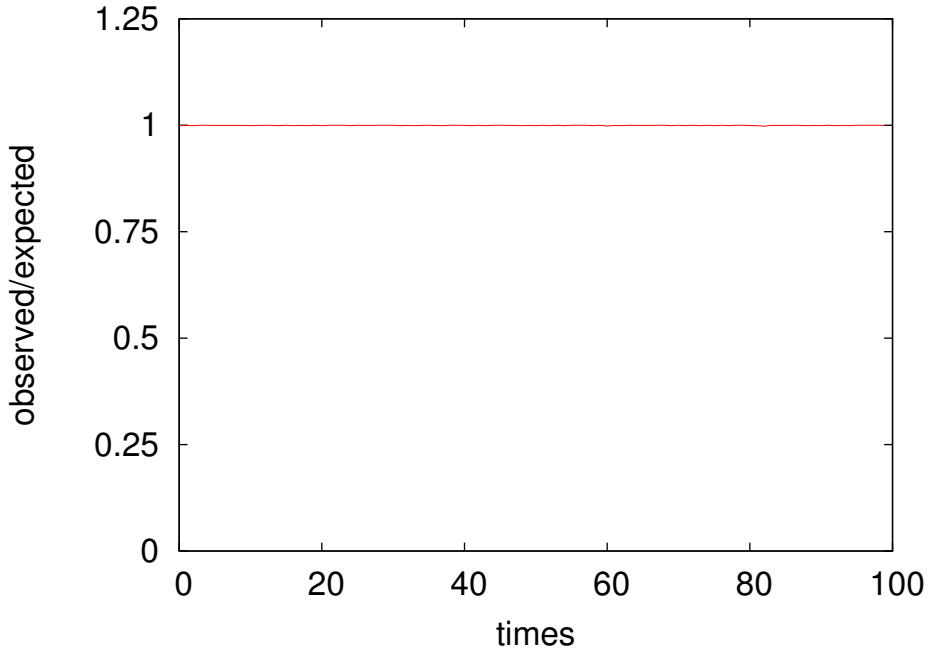
実行結果

```
% ./a.out
tbegin 1213320788.074777
tend   1213320791.065997
planed    3s 3000ms
takes        2992ms
```

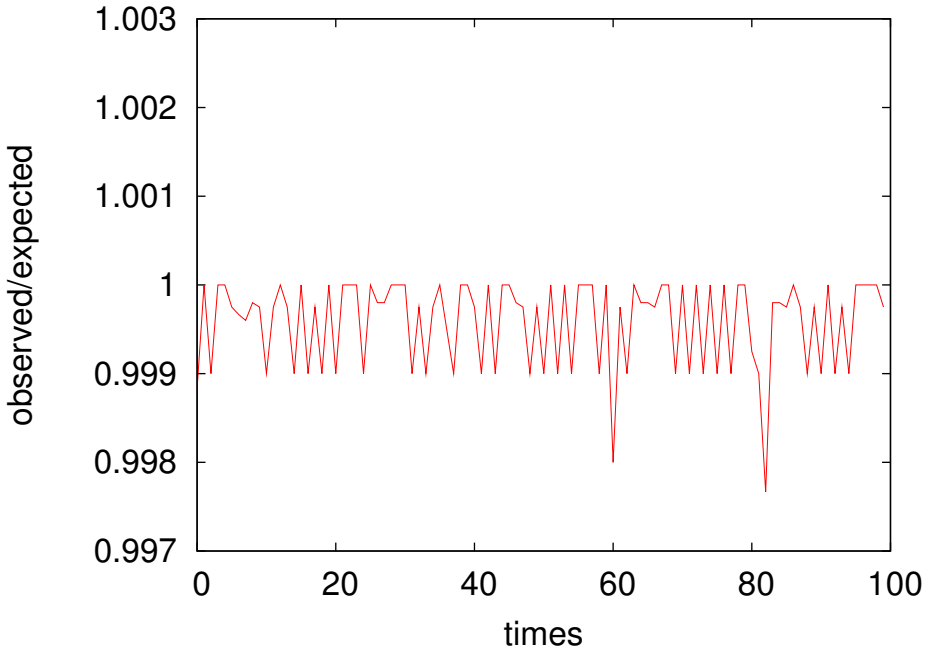
SunOS 5.9 の sleep は与えた秒数より短め

時刻計測 gettimeofday *(cont.)*

100回繰り返して、実測と計画との比率を算出



拡大



1を越えることはない、平均 0.9996

処理時間の短い対象を計測

短い処理時間を計測して平均すると...

- 計測のオーバヘッド
- 累積誤差

そこで、

- 何度も繰り返してその全体を計測
- 計測値を繰り返し数で割る（平均）

処理時間の短い対象を計測 *(cont.)*

たとえば、関数 func の処理時間を求める

```
gettimeofday(&tbegin, NULL);
for(i=0;i<N;i++) {
    func()
}
gettimeofday(&tend, NULL);
ave = SUB_MSEC(tend, tbegin)/N;
```

補足

プラットホーム固有のより精度の高い計測方法も検討

- Intel CPU では TSC (Timestamp Counter)

NTP 等を利用すると計測に影響がでる

- 長い単位で見ると正確
- 短い単位で見ると時間が一定ではない
  - 時刻が遅いと時間を縮める（速くなる）
  - 時刻が速いと時間を伸ばす（遅くなる）

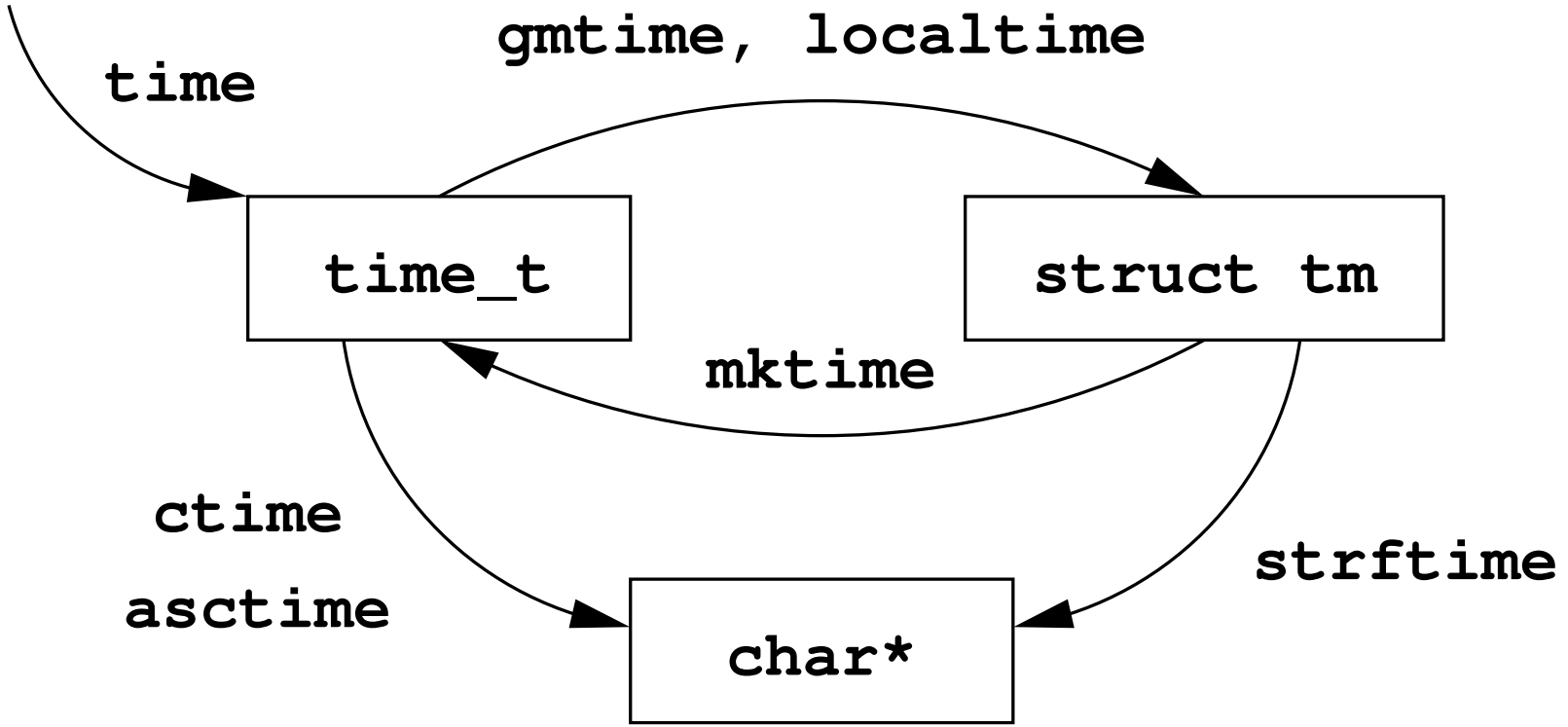
時刻処理関数群

struct tm — 時刻を格納する構造体

メンバ	内容
int tm_sec;	秒 [0,61] (閏秒を含む)
int tm_min;	分 [0,50]
int tm_hour;	時 [0,23]
int tm_mday;	日 [1,31]
int tm_mon;	月 [0,11]
int tm_year;	年 1900を起源とする
int tm_wday;	曜日 [0,6]
int tm_yday;	通し日 元旦を起源とする [0,365]
int tm_isdst;	夏時間の有無

時刻処理関数群 *(cont.)*

型変換



時刻処理関数群 *(cont.)*

time_t → struct tm

- locatime ローカル時刻に変換

```
struct tm *tmbuf;
tmbuf = localtime(&now);
printf("%d %d %d\n",
  1900+tmbuf->tm_year,
  1+tmbuf->tm_mon, tmbuf->tm_mday);
```

- gmtime グリニッジ時刻（多くはUTC）に変換

```
tmbuf = gmtime(&now);
```

現在時刻表示プログラム

```
#include <stdio.h>
#include <time.h>
int main(){
  time_t now; struct tm *tmbuf;
  now = time(NULL);
  tmbuf = localtime(&now);
  printf("%d %d %d %d %d %d\n",
    1900+tmbuf->tm_year,
    1+tmbuf->tm_mon, tmbuf->tm_mday,
    tmbuf->tm_hour, tmbuf->tm_min,
    tmbuf->tm_sec); }
```

時刻処理関数群 *(cont.)*

- 年は 1900、月は 1 を加える

実行例（date コマンドと一緒に）

```
% ./a.out ; date
2008 6 14 20 24 44
Sat Jun 14 20:24:44 JST 2008
```

gmtime を用いた場合の実行例

```
% ./a.out ; date -u
2008 6 14 11 31 29
Sat Jun 14 11:31:29 UTC 2008
```

時刻処理関数群 *(cont.)*

struct tm → time_t ; mktime
2008/02/29 00:00:00 に相当する time_t を算出

```
time_t now;
struct tm tmp, *src = &tmp, *dst;
src->tm_year = 2008-1900;
src->tm_mon  = 2 - 1;  src->tm_mday = 29;
src->tm_hour = src->tm_min = src->tm_sec = 0;
now = mktime(src);
```

無効な日時を与えると -1 を返す
ただし、読み変える（2/30 → 3/2 等）場合もある

時刻処理関数群 *(cont.)*

struct tm → char*
strftime 時刻を元にした文字列を生成する関数

- 様々な文字列を生成可能
  - ◇ 日時を数字や文字列で表現
- 用途例
  - ◇ 現在時刻の表示
  - ◇ 日付の付いたファイル名生成

```
/var/tmp/dummy-080614_220001.log
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){
    time_t now;
    struct tm *src;
    char dst[BUFSIZ];
    now = time(NULL);
    src = localtime(&now);
    strftime(dst, BUFSIZ,
        "%Y-%m-%d %H:%M:%S", src);
    printf("%s\n", dst); }
```

時刻処理関数群 *(cont.)*

実行例

```
% ./a.out ; date
2008-06-14 22:26:43
Sat Jun 14 22:26:43 JST 2008
```

先のファイル名は以下のように生成

```
strftime(dst, BUFSIZ,
  "/var/tmp/dummy-%y%m%d_%H%M%S.log",
  src);
```

補足

- GMT はグリニッジ標準時
  - 天文学的な定義が中心
- UTC は世界協定時
  - 原子時計中心

  ※ 今回はさほど気にしなくても良い
- 現行の時刻格納方法は 2038 年までしか表現できない

シェル上の計測

time コマンド
ユーザCPU時間、システムCPU時間、合計の 3つの時間を出力する
フォーマットや精度はプログラムによって微妙に違う
/bin/time の場合

```
% /bin/time  ./a.out
strop: 1000000 times total 3005ms, ave 3.00us 3005.00

real         3.0
user         3.0
sys          0.0
```

シェル上の計測 *(cont.)*

tcsh の内部コマンド time の場合

```
% time ./a.out
strop: 1000000 times total 3002ms, ave 3.00us 3002.00
3.00u 0.02s 0:03.07 98.3%
```

ユーザ 3.00秒、システム 0.02秒、計 3.07秒

date コマンド

さほど精度はないが、分かりやすい

```
% date ; find /var/tmp -name "foo" -print ; date
Sat Jun 14 11:45:50 JST 2008
Sat Jun 14 11:46:40 JST 2008
```

シェル上の計測 *(cont.)*

ヒストリ
さほど精度はないが、事後に処理時間がわかる
tcsh の場合

```
% history
...
  3564  11:45   date ; find /var/tmp -name "foo" -pri
  3565  11:46   date ; find /var/tmp -name "foo" -pri
  3566  11:47   history
```

演習

1) 標準ライブラリ関数 strcpy と同等の関数を作り、その関数と strcpy と処理時間を比較せよ★
   - 少なくとも計測時間が1秒を越える分だけ繰り返せ
2) 現在時刻をローカル時刻とグリニッジ標準時で出力するプログラムを作れ★
3) 2008/06/01 00:00:00 から 2008/6/30 23:59:59 までの秒数を算出するプログラムを作れ
   - 1日=86400 秒を元に四則演算で算出せよ★
   - mktime を用いて time_t の差で算出せよ★

演習 *(cont.)*

※ どちらも閏秒は考慮しなく良い

4) 2007年と 2008年の日曜日の数を算出するプログラムを作れ

- 1月1日から 1日ずつずらして（(ループして)曜日をしらべ、日曜日を数える★
- ループを用いずに日曜日を数える★★★★