■ pthread の実行時間計測

Linux で pthread の各スレッドの処理時間を
測定したいのだけど、どうしたらいいのかと尋ねられた。
普通に時間測定するなら getrusage だけど、
man pthreads
とかしてみると、
Posix.1 の pthreads で共有する属性の一つとして、CPU 時間とリソース消費量が挙げられている。
NPTL では、

```
- The information returned by times(2) and getrusage(2)
  is per-thread rather than process-wide (fixed in kernel 2.6.9).
```

とあるので、やはり、getrusage ではダメみたい。
実際やってみると、経過時間は、ちゃんと総和が得られているよう。
で、いろいろ調べてみると、pthread_getcpuclockid を使って

```
pthread_getcpuclockid(id, &c);
clock_gettime(c, &tp);
```

とすれば、各スレッドでの処理時間がとれるみたい。

ところで、FreeBSD の pthread には、
pthread_getcpuclockid はないみたいだけど、
どうすればいいのかな？

ちなみに、Linux では、

```
man pthreads
```

FreeBSD では、

```
man pthread
```

で、ちょっとびっくりした。

ちなみにテストしてみたソースコードは、次のようなもの。
コンパイルは、

```
gcc test.c -lpthread -lrt
```

とライブラリを指定する必要がある。

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <sys/resource.h>
#include <pthread.h>
```

```
#define N 16

void print_rusage_usec(pthread_t id){
  clockid_t c;
  struct timespec tp;
  struct timeval utime;
  struct timeval stime;
  pthread_getcpuclockid(id, &c);
  clock_gettime(c, &tp);

  printf("print_rusage_usec [%u] sec:%ld nsec:%ld\n",
                 id,
                 tp.tv_sec,
                 tp.tv_nsec
                 );
}

void *counter(void *arg)
{
  volatile int i,j,k;
  pid_t   pid;
  pthread_t thread_id;

  pid = getpid();
  thread_id = pthread_self();
  print_rusage_usec(thread_id);

  for(i = 0; i < 1000; i++){
    for(j = 0; j < 1000; j++){
      for(k = 0; k < 1000; k++){
      }
    }
  }
  print_rusage_usec(thread_id);

  return(arg);
}

int main()
{
  int status;
  void *result;
  pthread_t thread_id[N];
  int i;
  struct timespec tp;

  clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &tp);
  printf("process sec:%ld nsec:%ld\n", tp.tv_sec, tp.tv_nsec);

  for(i = 0; i < N; i++){
        status = pthread_create(&thread_id[i], NULL, counter, (void *)NULL);
        if(status != 0){
          fprintf(stderr, "pthread_create : %s", strerror(status));
        }
        else{
          //printf("thread %d is createed\n", thread_id[i]);
        }
  }

  for(i = 0; i < N; i++){
        pthread_join(thread_id[i], &result);
        //printf("thread %d is ended\n", thread_id[i]);
  }

  clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &tp);
  printf("process sec:%ld nsec:%ld\n", tp.tv_sec, tp.tv_nsec);

}
```