Thread

Praktikum Sistem Operasi

Ilmu Komputer IPB

2017

Thread

Thread

- *thread* adalah satuan dasar utilisasi CPU[1]
- tiap *thread* memiliki:
  - id, *program counter*, *register set*, dan *stack*
- dalam satu proses, *thread* berbagi:
  - segmen *code*, segmen *data*, dan sumberdaya lainnya, seperti *file*
- proses *multithreaded* memiliki beberapa *thread* yang dapat mengerjakan beberapa tugas secara bersamaan

[1] Silberschatz *et al.* (2013), *Operating System Concepts*, hlm 163.

POSIX Thread

- UNIX memakai standar POSIX[2] *thread* (`pthread`)
- saat kompilasi tambahkan *flag* `-pthread`

[2] The Portable Operating System Interface

Membuat Thread

```
pthread_create(&thread, attr, func, arg);
```

- membuat satu `thread` dengan atribut `attr` yang akan menjalankan fungsi `func` dengan argumen `arg`[3]
- deklarasi fungsi tersebut:
  - `void *func(void *arg);`[4]

[3] lihat `'man pthread_create'`

[4] `void*`: tipe data *generic pointer*

Menunggu Thread

```
pthread_join(thread, &retval);
```

- menunggu `thread` selesai dan menyimpan keluarannya ke variabel `retval`[5]

[5] lihat `'man pthread_join'`

Mengakhiri Thread

```
pthread_exit(retval);
```

- mengakhiri *thread* dengan nilai keluaran `retval`[6]

[6] lihat `'man pthread_exit'`

Contoh

Satu Thread Tanpa Argumen

```c
#include <pthread.h>
#include <stdio.h>

void *hello(void *arg) {
    printf("hello\n");
    pthread_exit(NULL);
}

int main() {
    pthread_t thread;
    pthread_create(&thread, NULL, hello, NULL);
    pthread_join(thread, NULL);
    return 0;
}
```

Dua Thread Tanpa Argumen

```
int main() {
    pthread_t thread1;
    pthread_t thread2;

    pthread_create(&thread1, NULL, hello, NULL);
    pthread_create(&thread2, NULL, hello, NULL);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    return 0;
}
```

Banyak Thread Tanpa Argumen

```
#define N 4

int main() {
    pthread_t thread[N];
    int i;

    for (i = 0; i < N; i++)
        pthread_create(&thread[i], NULL, hello, NULL);

    for (i = 0; i < N; i++)
        pthread_join(thread[i], NULL);

    return 0;
}
```

Satu Thread Dengan Argumen

```c
#include <pthread.h>
#include <stdio.h>

void* hello(void* arg) {
    printf("hello from thread %s\n", (char*)arg);
    pthread_exit(NULL);
}

int main() {
    pthread_t thread;
    pthread_create(&thread, NULL, hello, "0");
    pthread_join(thread, NULL);
    return 0;
}
```

Dua Thread Dengan Argumen

```
int main() {
    pthread_t thread1;
    pthread_t thread2;

    pthread_create(&thread1, NULL, hello, "0");
    pthread_create(&thread2, NULL, hello, "1");

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    return 0;
}
```

Banyak Thread Dengan Argumen

```
#define N 4

int main() {
    pthread_t thread[N];
    char *id[N] = {"0", "1", "2", "3"};
    int i;

    for (i = 0; i < N; i++)
        pthread_create(&thread[i], NULL, hello, id[i]);

    for (i = 0; i < N; i++)
        pthread_join(thread[i], NULL);

    return 0;
}
```

Latihan

Jumlah Array

- lengkapi program berikut untuk menjumlahkan nilai semua elemen *array* `A`
- gunakan variabel global `sum` untuk menyimpan hasilnya

```c
#include <stdio.h>
#define N 16

int sum = 0;

int main() {
    int A[N] = {68,34,64,95,35,78,65,93,
                51,67, 7,77, 4,73,52,91};
    // TODO: array sum
    printf("%d\n", sum);     // 954
    return 0;
}
```

Jumlah Array (Satu Thread)

- sekarang, buat satu buah *thread* untuk menjumlahkan nilai semua elemen *array* `A` dengan fungsi `array_sum()`
- *thread* utama hanya membuat dan menunggu *thread* ini selesai

Jumlah Array (Dua Thread)

- oke?
- sekarang gunakan 2 buah *thread* untuk menjumlahkan nilai semua elemen *array* A
- pastikan pembagian kerja antara kedua *thread* seimbang, yaitu tiap *thread* memproses $\frac{N}{2}$ elemen

Jumlah Array (Empat Thread)

- bisa?
- sekarang gunakan 4 buah *thread* untuk menjumlahkan nilai semua elemen *array* A
- pastikan pembagian kerja antara keempat *thread* seimbang, yaitu tiap *thread* memproses $\frac{N}{4}$ elemen
- kumpulkan di LMS