LABORATORIUM JARINGAN KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

PRAKTIKUM SISTEM OPERASI

SEMESTER : GENAP **TAHUN : 2015/2016**

BAB II

JUDUL BAB : THREAD

DISUSUN OLEH : MOH ARIF ANDRIAN

NIM : 156150600111002

ASISTEN : SISKA PERMATASARI
ZAENAL KURNIAWAN

KOORDINATOR ASISTEN : DANY RAHMANA

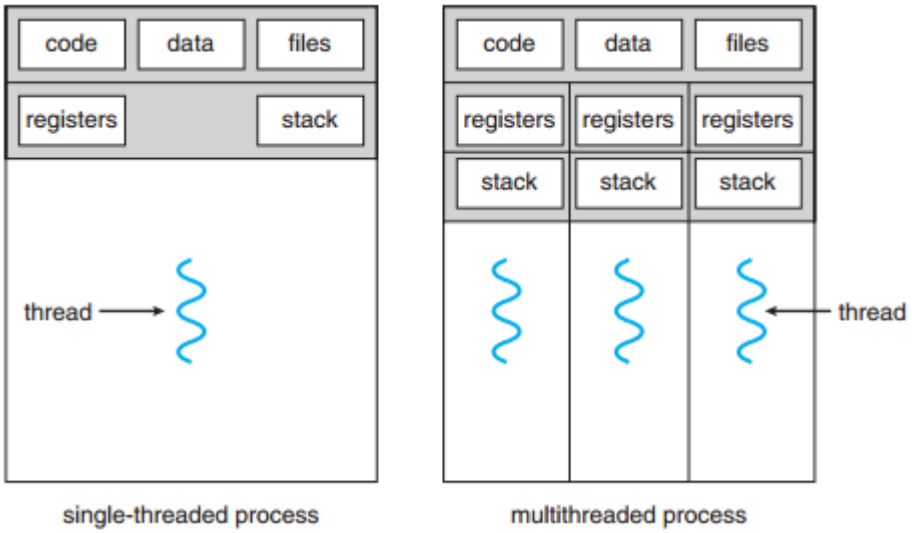
LAPORAN PRAKTIKUM SISTEM OPERASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

Nama : Moh. Arif Andrian
NIM : 156150600111002
Laporan : BAB II
Asisten : Siska Permatasari
Zaenal Kurniawan

BAB II
THREAD

DASAR TEORI

Thread adalah unit dasar dari *CPU utilization*. Thread terdiri dari *thread ID*, *program counter*, *register set*, dan *stack*. Beberapa Thread yang berada dalam proses yang sama saling berbagi *code section*, *data section*, dan *operating-system resources* (*files* dan *signals*).



Gambar 3.1 Perbedaan *single-threaded process* dengan *multithreaded process*
(Sumber: Operating System Concepts, 9th Edition)

2.1.1 Membuat Thread

Perintah pthread_create (thread,attr,start_routine,arg) berfungsi untuk membuat thread dan membuatnya dapat dieksekusi. Routine ini dapat dipanggil berkali-kali di dalam kode yang sudah dibuat. Argumen pada pthread_create :

a. thread : merupakan pengenal dari sebuah thread yang dibuat.
b. attr : merupakan atribut objek yang mungkin digunakan untuk mengeset atribut thread (bisa mengesetnya atau NULL untuk nilai default)
c. start_routine : merupakan fungsi pada C yang digunakan sebagai perintah untuk thread.
d. arg : merupakan nilai tunggal yang akan di passing-kan ke start_routine. passing yang ditentukan menggunakan passing by reference sebagai pointer pada tipe void. Gunakan NULL untuk nilai default.

2.1.2 Atrribut Thread

Secara default sebuah thread mempunyai attribut. Beberapa atribut dapat diubah oleh programmer menggunakan thread object atribut. pthread_attr_init(attr) dan pthread_attr_destroy(attr) digunakan untuk mengeset dan menghancurkan atribut pada thread.

2.1.3 Menghentikan Thread (pthread_exit())

Suatu thread dapat berhenti melalui beberapa cara diantaranya :

- Thread berhenti dengan normal saat pekerjaannya telah selesai.
- Thread dibatalkan dengan routine pthread_cancel.
- Semua proses diberhentikan dengan fungsi system exec atau exit ().
- Berakhirnya perintah pada main().
- Pemberhentian melalui pemanggilan pthread_exit(), baik pekerjaannya telah selesai atau belum.

Contoh penggunaan pthread_create dan pthread_exit :

teori-3.1.c

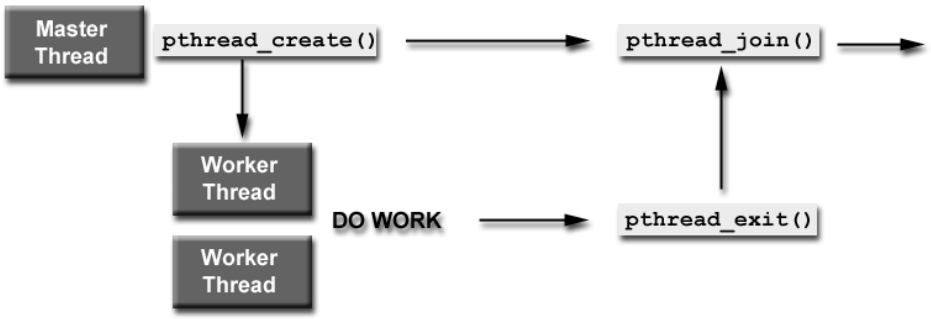
```
 1  #include <pthread.h>
 2  #include <stdio.h>
 3  #include <stdlib.h>
 4  #define NUM_THREADS   5
 5  void *PrintHello(void *threadid)
 6  {
 7     long tid;
 8     tid = (long)threadid;
 9     printf("Hello World! It's me, thread #%ld!\n",
    tid);
10     pthread_exit(NULL);
11  }
12  int main(int argc, char *argv[])
13  {
14     pthread_t threads[NUM_THREADS];
15     int rc;
16     long t;
17     for(t=0;t<NUM_THREADS;t++){
18       printf("In main: creating thread %ld\n", t);
19       rc = pthread_create(&threads[t], NULL,
    PrintHello, (void *)t);
20       if (rc){
21         printf("ERROR; return code from
    pthread_create() is %d\n", rc);
22         exit(-1);
23         }
24       }
    /* Last thing that main() should do */
25     pthread_exit(NULL);
26  }
```

```
  GNU nano 2.5.3                  File: teori-3.1.c

#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#define NUM_THREADS     5
void *PrintHello(void *threadid)
{
   long tid;
   tid = (long)threadid;
   printf("Hello World! It's me, thread #%ld!\n", tid);
   pthread_exit(NULL);
}
int main(int argc, char *argv[])
{
   pthread_t threads[NUM_THREADS];
   int rc;
   long t;
   for(t=0;t<NUM_THREADS;t++){
     printf("In main: creating thread %ld\n", t);
     rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
     if (rc){
       printf("ERROR; return code from pthread_create() is %d\n", rc);
       exit(-1);
       }
     }
/* Last thing that main() should do */
   pthread_exit(NULL);
}
```

```
root@156150600111002:/home/andrian/praktikum# nano teori-3.1.c
root@156150600111002:/home/andrian/praktikum# gcc -pthread -o teori-3.1 teori-3.1.c
root@156150600111002:/home/andrian/praktikum# ./teori-3.1
In main: creating thread 0
In main: creating thread 1
Hello World! It's me, thread #0!
In main: creating thread 2
Hello World! It's me, thread #1!
In main: creating thread 3
In main: creating thread 4
Hello World! It's me, thread #2!
Hello World! It's me, thread #3!
Hello World! It's me, thread #4!
root@156150600111002:/home/andrian/praktikum#
```

2.1.4 Pthreads



Gambar 3.2 Pthread
(Sumber: https://computing.llnl.gov/tutorials/pthreads/#Joining)

Pthreads merujuk pada POSIX standard (IEEE 1003.1c) yang mendefinisikan API untuk pembuatan dan sinkronisasi thread. Pthreads merupakan spesifikasi perilaku thread, bukan implementasi. Pembuat sistem operasi boleh mengimplementasikan spesifikasi Pthreads ini dengan berbagai cara dengan bebas. Contoh sistem operasi mengimplementasikan spesifikasi Pthreads adalah *UNIX-type systems* (Linux, Mac OS X, dan Solaris). Program **teori-3.1.c** berikut ini adalah contoh penggunaan Pthread pada sistem operasi Linux (Sumber: Operating System Concepts, 9th Edition).

teori-3.2.c

```
 1  #include<pthread.h>
 2  #include<stdio.h>
 3
 4  int sum; /* this data is shared by the thread(s) */
 5  void *runner(void *param); /* threads call this
 6  function */
 7
 8  int main(int argc, char *argv[])
 9  {
10        pthread_t tid; /* the thread identifier */
11        pthread_attr_t attr; /* set of thread attributes
12  */
13
14        if (argc != 2) {
15              fprintf(stderr,"usage: latihan-3.1 <integer
    value>\n");
16              return -1;
17        }
18        if (atoi(argv[1]) < 0) {
19              fprintf(stderr,"%d must be >=
    0\n",atoi(argv[1]));
20              return -1;
21        }
22
23        /* get the default attributes */
24        pthread_attr_init(&attr);
25        /* create the thread */
26        pthread_create(&tid,&attr,runner,argv[1]);
27        /* wait for the thread to exit */
28        pthread_join(tid,NULL);
29
30        printf("sum = %d\n",sum);
31  }
32  /* The thread will begin control in this function */
33  void *runner(void *param)
34  {
35        int i, upper = atoi(param);
```

```
36          sum = 0;
37          for (i = 1; i <= upper; i++)
38                sum += i;
39
40          pthread_exit(0);
41    }
```

```
 GNU nano 2.5.3                File: teori-3.2.c

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

int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /* threads call this function */

int main(int argc, char *argv[])
{
        pthread_t tid; /* the thread identifier */
        pthread_attr_t attr; /* set of thread attributes */

        if (argc != 2) {
                fprintf(stderr,"usage: latihan-3.1 <integer value>\n");
                return -1;
        }
        if (atoi(argv[1]) < 0) {
                fprintf(stderr,"%d must be >= 0\n",atoi(argv[1]));
                return -1;
        }

        /* get the default attributes */
        pthread_attr_init(&attr);
        /* create the thread */
        pthread_create(&tid,&attr,runner,argv[1]);
        /* wait for the thread to exit */
        pthread_join(tid,NULL);

printf("sum = %d\n",sum);
}

/* The thread will begin control in this function */
void *runner(void *param)
```

Berikut adalah proses *compile* dan output dari program **teori-3.1.c** diatas

```
dataq@so:~/code$ gcc -pthread -o teori-3.1 teori-3.1.c
dataq@so:~/code$ ./teori-3.1 10
sum = 55
dataq@so:~/code$
```

```
root@156150600111002:/home/andrian/praktikum# gcc -pthread -o teori-3.2 teori-3.2.c
root@156150600111002:/home/andrian/praktikum# ./teori-3.2 10
sum = 55
root@156150600111002:/home/andrian/praktikum#
```

2.1.5 Java Threads

Thread adalah model fundamental dari eksekusi program dalam program Java. Bahasa pemrograman Java menyediakan banyak fasilitas untuk pembuatan dan manajemen thread. Semua program Java minimal terdiri dari satu thread. Java

thread tersedia di seluruh sistem operasi, seperti Windows, Linux dan Mac OS X, yang menyediakan Java Virtual Machine (JVM). Java Thread API juga tersedia untuk aplikasi Android. Program **DisplaySum.java** berikut ini adalah contoh penggunaan Java Thread pada sistem operasi Windows.

DisplaySum.java

```
 1  public class DisplaySum implements Runnable {
 2      private int param;
 3      public DisplaySum(int param) {
 4          this.param = param;
 5      }
 6      public void run() {
 7          int sum = 0;
 8          for (int i = 1; i <= this.param; i++) {
 9              sum += i;
10          }
11          System.out.println("sum = " + sum);
12      }
13      public static void main(String[] args) {
14          try {
15              Runnable getSumRun = new
    DisplaySum(Integer.valueOf(args[0]));
16              Thread getSumThread = new
    Thread(getSumRun);
17              getSumThread.start();
18              getSumThread.join();
19          } catch (InterruptedException ex) {
20              System.err.println(ex.getMessage());
21          }
22      }
23  }
```

```
 GNU nano 2.5.3                File: DisplaySum.java

public class DisplaySum implements Runnable {
    private int param;
    public DisplaySum(int param) {
        this.param = param;
    }
    public void run() {
        int sum = 0;
        for (int i = 1; i <= this.param; i++) {
            sum += i;
        }
        System.out.println("sum = " + sum);
    }
    public static void main(String[] args) {
        try {
            Runnable getSumRun = new DisplaySum(Integer.valueOf(args[0]));
            Thread getSumThread = new Thread(getSumRun);
            getSumThread.start();
            getSumThread.join();
        } catch (InterruptedException ex) {
            System.err.println(ex.getMessage());
        }
    }
}
```

Berikut adalah proses *compile* dan output dari program **DisplaySum.java** diatas

```
C:\Users\dataq>javac DisplaySum.java

C:\Users\dataq>java DisplaySum 10
sum = 55
```

```
root@156150600111002:/home/andrian/praktikum# javac DisplaySum.java
root@156150600111002:/home/andrian/praktikum# java DisplaySum 10
sum = 55
root@156150600111002:/home/andrian/praktikum#
```

2.2 MATERI PRAKTIKUM

2.2.1 Thread ID

Lakukan percobaan sesuai dengan panduan berikut ini:

1. Login ke sistem GNU/Linux kemudian buka terminal.

```
andrian@156150600111002:~$ sudo su
[sudo] password for andrian:
root@156150600111002:/home/andrian#
```

2. Ketikkan program **materi-2.1.c** menggunakan editor favorit anda.

materi-2.1.c

```
 1  #include <stdio.h>
 2  #include <pthread.h>
 3  #include <stdlib.h>
 4  #include <time.h>
 5
 6  time_t current_time;
 7
 8  void *hello(void *param) {
 9      pthread_t self_id;
10      self_id = pthread_self();
```

```
11      int i = 0;
12      while(i < 3){
13          sleep(1);
14          current_time = time(NULL);
15          printf("Thread %lu say hello at %s",
16  self_id, ctime(&current_time));
17          i++;
18      }
19      printf("Thread %lu finished\n",self_id);
20      return &("joined");
21  }
22
23  main() {
24      pthread_t thread_id;
25      int ret;
26      void *res;
27
28  ret=pthread_create(&thread_id,NULL,&hello,NULL);
        printf("Created thread %lu \n", thread_id);
29      pthread_join(thread_id,&res);
30      printf("Thread %lu %s \n",thread_id,(char
    *)res);
31  }
```

```
 GNU nano 2.5.3                File: materi-2.1.c

#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <time.h>

time_t current_time;

void *hello(void *param) {
    pthread_t self_id;
    self_id = pthread_self();
    int i = 0;
    while(i < 3){
        sleep(1);
        current_time = time(NULL);
        printf("Thread %lu say hello at %s", self_id, ctime(&current_time));
        i++;
    }
    printf("Thread %lu finished\n",self_id);
    return &("joined");
}

main() {
    pthread_t thread_id;
    int ret;
    void *res;
    ret=pthread_create(&thread_id,NULL,&hello,NULL);
    printf("Created thread %lu \n", thread_id);

    pthread_join(thread_id,&res);
    printf("Thread %lu %s \n",thread_id,(char *)res);
}
```

3. Compile program dengan perintah: **gcc -pthread -o materi-2.1 materi-2.1.c**

```
root@156150600111002:/home/andrian/praktikum# gcc -pthread -o materi-2.1 materi-2.1.c
materi-2.1.c: In function 'hello':
materi-2.1.c:13:9: warning: implicit declaration of function 'sleep' [-Wimplicit-function-
declaration]
         sleep(1);
         ^
materi-2.1.c: At top level:
materi-2.1.c:22:1: warning: return type defaults to 'int' [-Wimplicit-int]
 main() {
 ^
```

4. Jalankan dengan perintah: **./materi-2.1** sebanyak 3 kali

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.1
```

5. Simpan tampilan output dan amati hasil ketiga output dari program **materi-2.1.c**

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.1
Created thread 139985745196800
Thread 139985745196800 say hello at Wed May  4 00:29:08 2016
Thread 139985745196800 say hello at Wed May  4 00:29:09 2016
Thread 139985745196800 say hello at Wed May  4 00:29:10 2016
Thread 139985745196800 finished
Thread 139985745196800 joined
root@156150600111002:/home/andrian/praktikum# ./materi-2.1
Created thread 140197796824832
Thread 140197796824832 say hello at Wed May  4 00:29:20 2016
Thread 140197796824832 say hello at Wed May  4 00:29:21 2016
Thread 140197796824832 say hello at Wed May  4 00:29:22 2016
Thread 140197796824832 finished
Thread 140197796824832 joined
root@156150600111002:/home/andrian/praktikum# ./materi-2.1
Created thread 139701685868288
Thread 139701685868288 say hello at Wed May  4 00:29:27 2016
Thread 139701685868288 say hello at Wed May  4 00:29:28 2016
Thread 139701685868288 say hello at Wed May  4 00:29:29 2016
Thread 139701685868288 finished
Thread 139701685868288 joined
```

6. Apakah ketiga output tersebut sama? Bila berbeda jelaskan letak perbedaannya!

   Jawaban:

   ketiga output tersebut memiliki perbedaan yaitu pada nomor created thread. Karena source code berdasarkan pada current time pada thread hanya menjalankan satu input dan mengeksekusi saja.

2.2.2 Multithread

Lakukan percobaan sesuai dengan panduan berikut ini:

1. Login ke sistem GNU/Linux kemudian buka terminal.

```
andrian@156150600111002:~$ sudo su
[sudo] password for andrian:
root@156150600111002:/home/andrian#
```

2. Ketikkan program `materi-2.2.c` menggunakan editor favorit anda.

materi-2.2.c

```
1  #include <stdio.h>
2  #include <pthread.h>
3  #include <stdlib.h>
4  #include <time.h>
5
6  time_t current_time;
```

```
 7
 8  void *hello(void *param) {
 9      pthread_t self_id;
10      self_id = pthread_self();
11         int delay = atoi(param);
12      int i = 0;
13      while(i < 3){
14          sleep(delay);
15          current_time = time(NULL);
16          printf("Thread %lu say hello at %s", self_id,
17  ctime(&current_time));
18          i++;
19      }
20      printf("Thread %lu finished\n",self_id);
21      return &("joined");
22  }
23
24  main() {
25      pthread_t thread_id_1, thread_id_2, thread_id_3;
26      int ret_1, ret_2, ret_3;
27      void *res_1, *res_2, *res_3;
28
29      ret_1=pthread_create(&thread_id_1,NULL,&hello,"1");
30      printf("Created thread %lu \n", thread_id_1);
31      ret_2=pthread_create(&thread_id_2,NULL,&hello,"2");
32      printf("Created thread %lu \n", thread_id_2);
33      ret_3=pthread_create(&thread_id_3,NULL,&hello,"3");
34      printf("Created thread %lu \n", thread_id_3);
35
36      pthread_join(thread_id_1,&res_1);
37      printf("thread %lu %s \n",thread_id_1,(char *)res_1);
38      pthread_join(thread_id_2,&res_2);
39      printf("thread %lu %s \n",thread_id_2,(char *)res_2);
40      pthread_join(thread_id_3,&res_3);
41      printf("thread %lu %s \n",thread_id_3,(char *)res_3);
    }
```

```
  GNU nano 2.5.3                File: materi-2.2.c                          Modifi
```

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

time_t current_time;

void *hello(void *param) {
    pthread_t self_id;
    self_id = pthread_self();
        int delay = atoi(param);
    int i = 0;
    while(i < 3){
        sleep(delay);
        current_time = time(NULL);
        printf("Thread %lu say hello at %s", self_id, ctime(&current_time));
        i++;
    }
    printf("Thread %lu finished\n",self_id);
    return &("joined");
}

main() {
    pthread_t thread_id_1, thread_id_2, thread_id_3;
    int ret_1, ret_2, ret_3;
    void *res_1, *res_2, *res_3;
```

3. Compile program dengan perintah: **gcc -pthread -o materi-2.2 materi-2.2.c**

```
root@156150600111002:/home/andrian/praktikum# gcc -pthread -o materi-2.2 materi-2.2.c
materi-2.2.c: In function ‘hello’:
materi-2.2.c:14:9: warning: implicit declaration of function ‘sleep’ [-Wimplicit-functi
on-declaration]
         sleep(delay);
         ^
materi-2.2.c: At top level:
materi-2.2.c:23:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main() {
 ^
```

4. Jalankan dengan perintah: **./materi-2.2**

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.2
```

5. Simpan tampilan output dan amati hasil output dari program **materi-2.2.c**

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.2
Created thread 140432538310400
Created thread 140432529917696
Created thread 140432521524992
Thread 140432538310400 say hello at Wed May  4 01:06:13 2016
Thread 140432529917696 say hello at Wed May  4 01:06:14 2016
Thread 140432538310400 say hello at Wed May  4 01:06:14 2016
Thread 140432521524992 say hello at Wed May  4 01:06:15 2016
Thread 140432538310400 say hello at Wed May  4 01:06:15 2016
Thread 140432538310400 finished
thread 140432538310400 joined
Thread 140432529917696 say hello at Wed May  4 01:06:16 2016
Thread 140432521524992 say hello at Wed May  4 01:06:18 2016
Thread 140432529917696 say hello at Wed May  4 01:06:18 2016
Thread 140432529917696 finished
thread 140432529917696 joined
Thread 140432521524992 say hello at Wed May  4 01:06:21 2016
Thread 140432521524992 finished
thread 140432521524992 joined
root@156150600111002:/home/andrian/praktikum#
```

6. Apakah ketiga thread pada program tersebut selesai bersamaan? Bila tidak, thread manakah yang selesai terlebih dahulu? Jelaskan penyebabnya!
   Jawaban:
   Tidak, thread yang harus selesai dahulu adalah thread dengan input “1”,”2”, dan ”3” atau fungsi ret_1,ret_2 dan ret_3.

7. Ubah baris ke 28, 30, dan 32 menjadi seperti potongan program berikut, lalu simpan dengan nama **materi-2.2.c**

materi-2.2.c	
...	...
28	`ret_1=pthread_create(&thread_id_1,NULL,&hello,"3");`
...	...
30	`ret_2=pthread_create(&thread_id_2,NULL,&hello,"2");`
...	...
32	`ret_3=pthread_create(&thread_id_3,NULL,&hello,"1");`
...	...

```c
main() {
    pthread_t thread_id_1, thread_id_2, thread_id_3;
    int ret_1, ret_2, ret_3;
    void *res_1, *res_2, *res_3;

    ret_1=pthread_create(&thread_id_1,NULL,&hello,"3");
    printf("Created thread %lu \n", thread_id_1);
    ret_2=pthread_create(&thread_id_2,NULL,&hello,"2");
    printf("Created thread %lu \n", thread_id_2);
    ret_3=pthread_create(&thread_id_3,NULL,&hello,"1");
    printf("Created thread %lu \n", thread_id_3);
```

8. Compile program dengan perintah: `gcc -pthread -o materi-2.2 materi-2.2.c`

```
root@156150600111002:/home/andrian/praktikum# gcc -pthread -o materi-2.2 materi-2.2.c
materi-2.2.c: In function ‘hello’:
materi-2.2.c:14:9: warning: implicit declaration of function ‘sleep’ [-Wimplicit-functi
on-declaration]
         sleep(delay);
         ^
materi-2.2.c: At top level:
materi-2.2.c:23:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main() {
 ^
```

9. Jalankan dengan perintah: `./materi-2.2`

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.2
```

10. Simpan tampilan output dan amati hasil output dari program `materi-2.2.c`

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.2
Created thread 139635756492544
Created thread 139635748099840
Created thread 139635739707136
Thread 139635739707136 say hello at Wed May  4 01:14:23 2016
Thread 139635748099840 say hello at Wed May  4 01:14:24 2016
Thread 139635739707136 say hello at Wed May  4 01:14:24 2016
Thread 139635756492544 say hello at Wed May  4 01:14:25 2016
Thread 139635739707136 say hello at Wed May  4 01:14:25 2016
Thread 139635739707136 finished
Thread 139635748099840 say hello at Wed May  4 01:14:26 2016
Thread 139635756492544 say hello at Wed May  4 01:14:28 2016
Thread 139635748099840 say hello at Wed May  4 01:14:28 2016
Thread 139635748099840 finished
Thread 139635756492544 say hello at Wed May  4 01:14:31 2016
Thread 139635756492544 finished
thread 139635756492544 joined
thread 139635748099840 joined
thread 139635739707136 joined
root@156150600111002:/home/andrian/praktikum#
```

11. Apakah ketiga thread pada program tersebut selesai bersamaan? Bila tidak, thread manakah yang selesai terlebih dahulu? Jelaskan penyebabnya!

    Jawaban:

    perbedaan terletak pada joined, jika nomor created thread dimulai 1,2,3 maka joined keluar setelah finish. Jika nomor created thread dimulai 3,2,1 maka joined akan keluar diakhir.

2.2.3 Shared Variable

Lakukan percobaan sesuai dengan panduan berikut ini:

1. Login ke sistem GNU/Linux kemudian buka terminal.

```
andrian@156150600111002:~$ sudo su
[sudo] password for andrian:
root@156150600111002:/home/andrian#
```

2. Ketikkan program `materi-2.4.c` menggunakan editor favorit anda.

materi-2.4.c

```
1  #include <stdio.h>
2  #include <pthread.h>
3  #include <stdlib.h>
4  #include <time.h>
5
6  time_t current_time;
7  int sharevar;
8
9  void *counter(void *param) {
```

```
10      int i = 0;
11      while(i < 10){
12          sleep(1);
13          sharevar += 1;
14          i++;
15      }
16      current_time = time(NULL);
17      printf("\nCounter thread finished at %s",
18  ctime(&current_time));
19      return NULL;
20  }
21
22  void *printer(void *param) {
23      int i = 0;
24      while(i < 10){
25          sleep(2);
26          current_time = time(NULL);
27          printf("\n%s", ctime(&current_time));
28          printf("sharevar = %i\n", sharevar);
29          i++;
30      }
31      current_time = time(NULL);
32      printf("\nPrinter thread finished at %s",
33  ctime(&current_time));
34      return NULL;
35  }
36
37  main() {
38      pthread_t tid_counter, tid_printer;
39      int ret_counter, ret_printer;
40      void *res_counter, *res_printer;
41
42
43  ret_counter=pthread_create(&tid_counter,NULL,&counter,NULL);
44      current_time = time(NULL);
45      printf("\nCounter thread created at %s",
46  ctime(&current_time));
47
48  ret_counter=pthread_create(&tid_printer,NULL,&printer,NULL);
49      current_time = time(NULL);
        printf("\nPrinter thread created at %s",
    ctime(&current_time));

        pthread_join(tid_counter,&res_counter);
        pthread_join(tid_printer,&res_printer);
    }
```

```
  GNU nano 2.5.3                File: materi-2.4.c

#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <time.h>

time_t current_time;
int sharevar;

void *counter(void *param) {
    int i = 0;
    while(i < 10){
        sleep(1);
        sharevar += 1;
        i++;
    }
    current_time = time(NULL);
    printf("\nCounter thread finished at %s", ctime(&current_time));
    return NULL;
}

void *printer(void *param) {
    int i = 0;
    while(i < 10){
        sleep(2);
        current_time = time(NULL);
        printf("\n%s", ctime(&current_time));
```

3. Compile program dengan perintah: **`gcc -pthread -o materi-2.4 materi-2.4.c`**

```
root@156150600111002:/home/andrian/praktikum# gcc -pthread -o materi-2.4 materi-2.4.c
materi-2.4.c: In function ‘counter’:
materi-2.4.c:12:9: warning: implicit declaration of function ‘sleep’ [-Wimplicit-functi
on-declaration]
         sleep(1);
         ^
materi-2.4.c: At top level:
materi-2.4.c:35:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main() {
 ^
```

4. Jalankan dengan perintah: **`./materi-2.4`**

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.4
```

5. Simpan tampilan output dan amati hasil output dari program **`materi-2.4.c`**

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.4

Counter thread created at Wed May  4 09:33:47 2016

Printer thread created at Wed May  4 09:33:47 2016

Wed May  4 09:33:49 2016
sharevar = 2

Wed May  4 09:33:51 2016
sharevar = 4

Wed May  4 09:33:53 2016
sharevar = 6

Wed May  4 09:33:55 2016
sharevar = 8

Counter thread finished at Wed May  4 09:33:57 2016

Wed May  4 09:33:57 2016
sharevar = 10

Wed May  4 09:33:59 2016
sharevar = 10

Wed May  4 09:34:01 2016
sharevar = 10

Wed May  4 09:34:03 2016
sharevar = 10

Wed May  4 09:34:05 2016
sharevar = 10

Wed May  4 09:34:07 2016
sharevar = 10

Printer thread finished at Wed May  4 09:34:07 2016
```

6. Apakah kedua thread pada program tersebut selesai bersamaan? Bila tidak, thread manakah yang selesai terlebih dahulu? Jelaskan dampaknya!

   Jawaban:

   current time pada thread yang selesai dahulu adalah thread dengan sleep (1), slepp (2) dan shareval juga berbeda jadi counter thread dikerjakan terlebih dahulu sehingga selesi terlebih dahulu, printer thread selesai paling akhir dikarenakan dikerjakan setelah counter thread.

7. Ubah baris ke 12 dan 24 menjadi seperti potongan program berikut, lalu simpan dengan nama **materi-2.5.c**

materi-2.5.c	
...	...
12	`sleep(2);`
...	...
24	`sleep(1);`
...	...

```
void *counter(void *param) {
    int i = 0;
    while(i < 10){
        sleep(2);
        sharevar += 1;
        i++;
    }
    current_time = time(NULL);
    printf("\nCounter thread finished at %s", ctime(&current_time));
    return NULL;
}

void *printer(void *param) {
    int i = 0;
    while(i < 10){
        sleep(1);
        current_time = time(NULL);
        printf("\n%s", ctime(&current_time));
        printf("sharevar = %i\n", sharevar);
        i++;
    }
    current_time = time(NULL);
    printf("\nPrinter thread finished at %s", ctime(&current_time));
    return NULL;
}
```

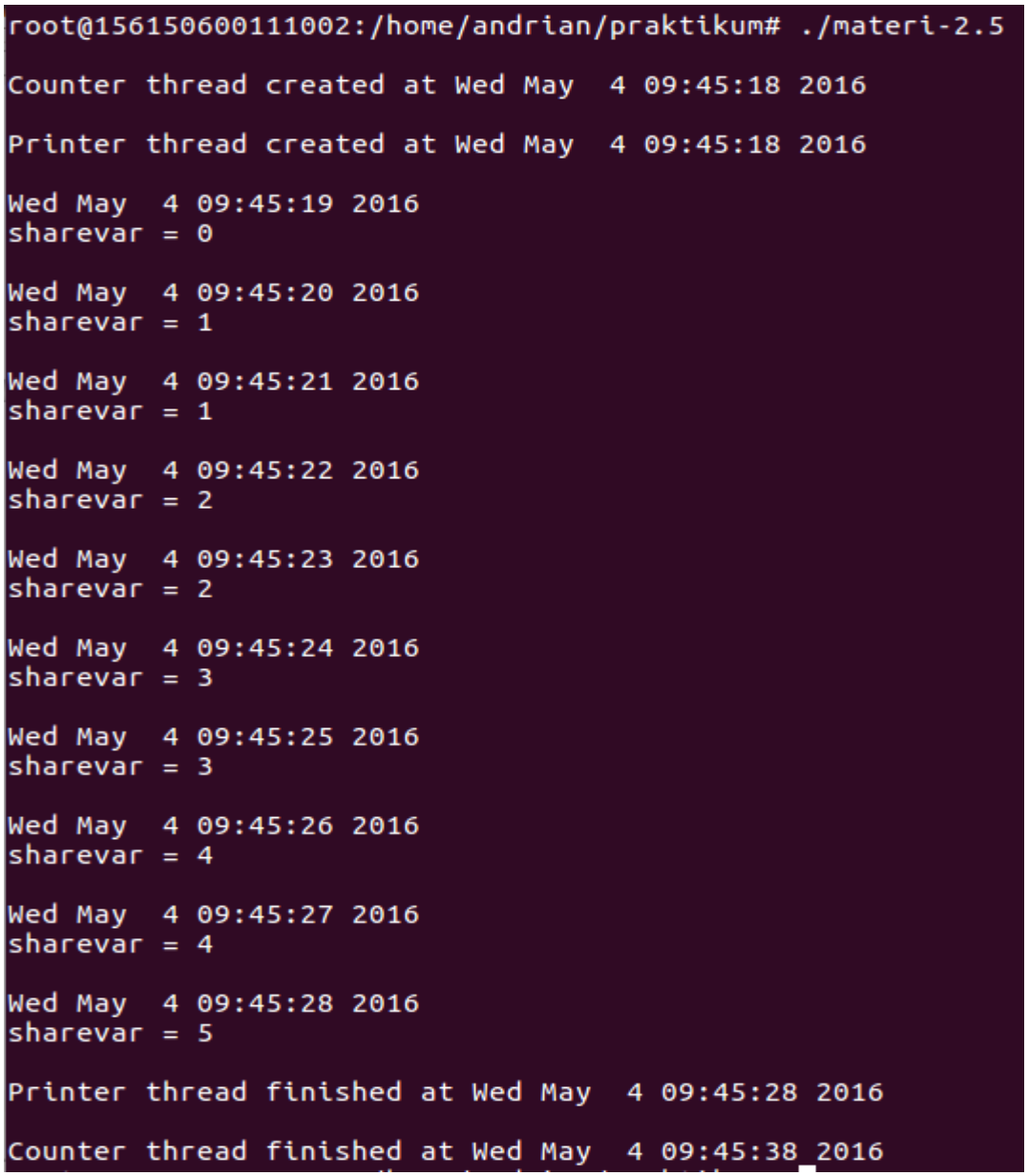
8. Compile program dengan perintah: **`gcc -pthread -o materi-2.5 materi-2.5.c`**

```
root@156150600111002:/home/andrian/praktikum# gcc -pthread -o materi-2.5 materi-2.5.c
materi-2.5.c: In function ‘counter’:
materi-2.5.c:12:9: warning: implicit declaration of function ‘sleep’ [-Wimplicit-functi
on-declaration]
         sleep(2);
         ^
materi-2.5.c: At top level:
materi-2.5.c:35:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main() {
 ^
```

9. Jalankan dengan perintah: **`./materi-2.5`**

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.5
```

10. Simpan tampilan output dan amati hasil output dari program **`materi-2.5.c`**



11. Apakah kedua thread pada program tersebut selesai bersamaan? Bila tidak, thread manakah yang selesai terlebih dahulu? Jelaskan dampaknya!
    Jawaban:
    bersamaan perbedaan setelah source code dirubah dari sleep(2) da sleep(1) adalah keluaran tergantung pada current time dan sharevarnya juga beda dan selisih waktunya lebih sedikit.

2.2.4 Joining and Detaching Threads

Joining (Bergabung) merupakan salah satu cara untuk mensinkronisasikan antar thread. The pthread_join Function digunakan untuk menunda thread yang sekarang sampai thread yang dispesifikkan dihentikan atau terhenti. Return value thread lain akan disimpan pada alamat memory yang ditunjuk oleh thread_return

jika nilainya tidak NULL. Sumber daya memory yang digunakan oleh sebuah thread tidak akan dialokasikan sampai pthread_join selesai menggunakan. Jadi routin pthread_join harus dipanggil sekali pada setiap thread yang terhubung. Thread harus dapat digabungkan(joinable) bukan pada posisi detach (detached state) dan tidak ada thread lain yang mencoba menggunakan pthread_join pada thread yang sama. Thread bisa pada kondisi detach state dengan menggunakan pendekatan argumen attr saat pthread_create atau dengan memanggil routine pthread_detach. Kemudian juga pthread_detach juga digunakan untuk memisahkan thread yang berada pada kondisi joinable / terhubung

Lakukan percobaan sesuai dengan panduan berikut ini:

1. Login ke sistem GNU/Linux kemudian buka terminal.

```
andrian@156150600111002:~$ sudo su
[sudo] password for andrian:
root@156150600111002:/home/andrian#
```

2. Ketikkan program **`materi-2.6.c`** menggunakan editor favorit anda.

materi-2.6.c

```
 1  #include <pthread.h>
 2  #include <stdio.h>
 3  #include <stdlib.h>
 4  #define NUM_THREADS 4
 5  void *BusyWork(void *t)
 6  {
 7     int i;
 8     long tid;
 9     double result=0.0;
10     tid = (long)t;
11     printf("Thread %ld starting...\n",tid);
12     for (i=0; i<1000000; i++)
13     {
14        result = result + sin(i) * tan(i);
15     }
16     printf("Thread %ld done. Result = %e\n",tid, result);
17     pthread_exit((void*) t);
18  }
19  int main (int argc, char *argv[])
20  {
21     pthread_t thread[NUM_THREADS];
22     pthread_attr_t attr;
23     int rc;
24     long t;
25     void *status;
26  /* Initialize and set thread detached attribute */
27     pthread_attr_init(&attr);
28     pthread_attr_setdetachstate(&attr,
29  PTHREAD_CREATE_JOINABLE);
30  for(t=0; t<NUM_THREADS; t++) {
31        printf("Main: creating thread %ld\n", t);
32        rc = pthread_create(&thread[t], &attr, BusyWork, (void
33  *)t);
34        if (rc) {
35
```

```
36          printf("ERROR; return code from pthread_create()
37  is %d\n", rc);
38          exit(-1);
39          }
40       }
41  /* Free attribute and wait for the other threads */
42     pthread_attr_destroy(&attr);
43     for(t=0; t<NUM_THREADS; t++) {
44        rc = pthread_join(thread[t], &status);
45        if (rc) {
46          printf("ERROR; return code from pthread_join()
47  is %d\n", rc);
48          exit(-1);
49          }
50        printf("Main: completed join with thread %ld having a
    status of %ld\n",t,(long)status);
        }
    printf("Main: program completed. Exiting.\n");
    pthread_exit(NULL);
    }
```

```
  GNU nano 2.5.3              File: materi-2.6.c

#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#define NUM_THREADS     4
void *BusyWork(void *t)
{
   int i;
   long tid;
   double result=0.0;
   tid = (long)t;
   printf("Thread %ld starting...\n",tid);
   for (i=0; i<1000000; i++)
   {
      result = result + sin(i) * tan(i);
   }
   printf("Thread %ld done. Result = %e\n",tid, result);
   pthread_exit((void*) t);
}
int main (int argc, char *argv[])
{
   pthread_t thread[NUM_THREADS];
   pthread_attr_t attr;
   int rc;
   long t;
   void *status;
/* Initialize and set thread detached attribute */
   pthread_attr_init(&attr);
   pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
for(t=0; t<NUM_THREADS; t++) {
```

3. Compile program dengan perintah: **`gcc -pthread -o materi-2.6 materi-2.6.c`**

```
root@156150600111002:/home/andrian/praktikum# gcc -pthread -o materi-2.6 materi-
2.6.c
materi-2.6.c: In function 'BusyWork':
materi-2.6.c:14:25: warning: implicit declaration of function 'sin' [-Wimplicit-
function-declaration]
       result = result + sin(i) * tan(i);
                         ^
materi-2.6.c:14:25: warning: incompatible implicit declaration of built-in funct
ion 'sin'
materi-2.6.c:14:25: note: include '<math.h>' or provide a declaration of 'sin'
materi-2.6.c:14:34: warning: implicit declaration of function 'tan' [-Wimplicit-
function-declaration]
       result = result + sin(i) * tan(i);
                                  ^
materi-2.6.c:14:34: warning: incompatible implicit declaration of built-in funct
ion 'tan'
materi-2.6.c:14:34: note: include '<math.h>' or provide a declaration of 'tan'
/tmp/cc3ioNcU.o: In function `BusyWork':
materi-2.6.c:(.text+0x46): undefined reference to `sin'
materi-2.6.c:(.text+0x59): undefined reference to `tan'
collect2: error: ld returned 1 exit status
```

4. Jalankan dengan perintah: **`./materi-2.6`**

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.6
```

5. Simpan tampilan output dan amati hasil output dari program **`materi-2.6.c`**

```
root@156150600111002:/home/andrian/praktikum# ./materi-2.6
bash: ./materi-2.6: No such file or directory
root@156150600111002:/home/andrian/praktikum#
```

2.2.5 Pembatalan Thread

Fungsi pthread_cancel membolehkan thread sekarang membatalkan thread lain.

materi-2.7.c

```
1  int pthread_cancel (pthread_t thread);
2  int pthread_setcancelstate ( int state, int *oldstate);
3  int pthread_setcanceltype ( int type, int *oldtype);
4  void pthread_testcancel(void);
```

Sebuah thread mengeset state pembatalan menggunakan pthread_statecancelstate yang mana mempunyai dua argumen. Argumen state merupakan state yang baru, dan argumen oldstate adalah pointer ke variable yang menyimpan state thread yang lama (oldstate) jika oldstate tidak null. Jika state adalah PTHREAD_CANCEL_ENABLE, maka permintaan pembatalan akan dilakukan. Sedangkan jika state pada kondisi PTHREAD_CANCEL_DISABLE maka permintaan pembatalan akan diabaikan.
Fungsi pthread_setcanceltype merubah bagaimana sebuah respon thread menjadi permintaan pembatalan. Jika type adalah PTHREAD_CANCEL_ASYNCHRONOUS thread akan dibatalkan dengan segera. Sedangkan jika type adalah PTHREAD_CANCEL_DEFERRED maka akan menunda pembatalan sampai tercapai titik pembatalan (cancelation point). Titik pembatalan dapat dicapai dengan memanggil fungsi pthread_testcancel(void) yang mana akan membatalkan thread sekarang jika ada penangguhan permintaan pembatalan yang tertunda. Fungsi cancel diatas mengembalikan nilai 0 jika sukses dan error untuk nilai lainnya kecuali fungsi pthread_cancel tidak mengembalikan value.

LAPORAN PRAKTIKUM SISTEM OPERASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

Nama : Moh. Arif Andrian
NIM : 156150600111002
Tugas : BAB II
Asisten : Siska Permatasari
Zaenal Kurniawan

TUGAS PRAKTIKUM

Kerjakan soal-soal berikut ini!

1. Jelaskan perbedaan dari pemrograman sekuensial dan pemrograman multithread!

   Jawaban:

   Pemrograman sekuensial menjalankan proses sesuai dengan urutan pengerjaan yang terdapat pada program, tetapi pemrograman multithread mengerjakan program dengan cara membagi program menjadi beberapa bagian kemudian menyelesaikan bagian tersebut, pada akhirnya proses akan dijalankan secara lengkap.

2. Jelaskan keungggulan dan kelemahan dari:

   a. pemrograman sekuensial

   Keunggulan: permrograman sekuensial tidak memiliki kemungkinan untuk menimbulkan deadlock karena hanya 1 proses, proses tersebut bebas menggunakan resource manapun sehingga tidak terdaji perebutan resource yang bias mengakibatkan deadlock.

   Kelemahan: Pemrograman sekuensial membutuhkan waktu yang lama untuk menjalankan proses karena sistem harus mengerjakan keseluruhan proses hanya dengan melalui 1 cara (dengan menyelesaikan proses dari awal sampai akhir).

   b. pemrograman multithread

   Keunggulan: Pemrograman multithread membutuhkan waktu yang cepat untuk menyelesaikan semua proses, karena proses - proses yang terdapat didalam pemrograman dapat dibagi kedalam thread yang kemudian akan seluruh proses yang telah dibagi tersebut akan dieksekusi secara bersamaan.

   Kelemahan: Pemrograman multithread memiliki kemungkinan untuk menimbulkan deadlock pada komputer karena proses - proses yang terbagi pada pemrograman multithread bisa saja membutuhkan resource yang sama pada satu waktu yang sama.

3. Apakah penggunaan thread dapat mempercepat waktu komputasi program dalam menyelesaikan sebuah kasus / permasalahan? Bila iya, kasus / permasalahan seperti apakah yang pengerjaannya dapat lebih optimal bila menggunakan thread? Buktikan dengan membuat program sederhana dalam bahasa C atau JAVA!

Jawaban:

Iya, penggunaan thread (multi) membuat komputer dapat menangani lebih dari 1 proses dalam satu waktu, caranya yaitu dengan mengerjakan proses secara bergantian namun dalam interval waktu yang sangat tipis sehingga terasa hanya seperti mengerjakan 1 proses.

Berikut ini merupakan contoh program dengan multithread menggunakan bahasa c dengan fokus thread untuk menjalankan fungsi loop.

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <errno.h>
#ifndef FALSE
#define FALSE 0
#define TRUE 1
#endif
volatile int is_stop;
pthread_t print_thread;
void *print_loop(void *unused)
{
  while (!is_stop) {
    printf("This comming from the print_loop.\n");
    sleep(3);
  }
  pthread_exit(NULL);
}
int main(int argc, char *argv[])
{
  int i = 0;
  is_stop = FALSE;
  int res = pthread_create(&print_thread, NULL,
print_loop, NULL);
  errno = 0;
  if (res != 0) {
    fprintf(stderr, "Error creating thread: %s\n",
strerror(errno));
    exit(EXIT_FAILURE);
  }
  for (i=0; i<20; i++) {
    printf("This comming from main program. %d.\n", i);
    sleep(1);
  }
  printf("Stopping print_loop...\n");
  is_stop = TRUE;
  errno = 0;
  res = pthread_join(print_thread, NULL);
```

```
  if (res != 0) {
    fprintf(stderr, "Error joining thread: %s\n",
strerror(errno));
  }
  return 0;
}
```

LAPORAN PRAKTIKUM SISTEM OPERASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

Nama : Moh. Arif Andrian
NIM : 156150600111002
Kesimpulan : BAB II
Asisten : Siska Permatasari
Zaenal Kurniawan

KESIMPULAN

Thread adalah sebuah alur kontrol dari sebuah proses. Suatu proses yang multithreading mengandung beberapa perbedaan alur kontrol dengan ruang alamat yang sama. Keuntungan dari multithreading meliputi peningkatan respon dari pengguna (responsif), pembagian sumber daya proses, ekonomis, dan kemampuan untuk mengambil keuntungan dari arsitektur multiprosesor.

Thread juga merupakan unit dasar dari penggunaan CPU, yang terdiri dari Thread_ID, program counter, register set, dan stack. Sebuah thread berbagi code section, data section, dan sumber daya sistem operasi dengan Thread lain yang dimiliki oleh proses yang sama. Thread juga sering disebut lightweight process. Sebuah proses tradisional atau heavyweight process mempunyai thread tunggal yang berfungsi sebagai pengendali. Perbedaan antara proses dengan thread tunggal dengan proses dengan thread yang banyak adalah proses dengan thread yang banyak dapat mengerjakan lebih dari satu tugas pada satu satuan waktu.

Macam-macam Thread

- Single threading
  Proses hanya mengeksekusi satu thread dalam satu waktu.
- Multi-threading
  Proses dapat mengeksekusi sejumlah thread dalam satu waktu.

Model MultiThreading:

- One to one
  Memetakan setiap user thread ke dalam 1 kernel thread.
  Kelebihan: Model one-to-one lebih sinkron daripada model many-to-one karena mengizinkan thread lain untuk berjalan ketika suatu thread membuat pemblokingan terhadap sistem pemanggilan, hal ini juga membuat multiple thread bisa berjalan secara parallel dalam multiprosesor .
  Kekurangan: Dalam pembuatan user thread diperlukan pembuatan korespondensi thread pengguna. Karena dalam proses pembuatan kernel thread dapat mempengaruhi kinerja dari aplikasi, maka kebanyakan dari implementasi model ini membatasi jumlah thread yang didukung oleh sistem.
  Model ini ada pada Windows NT dan OS/2.
- One to Many
  Memetakan beberapa tingkatan thread user hanya ke satu buah kernel thread.

Kelebihan: Managemen proses thread dilakukan oleh (di ruang) pengguna, sehingga menjadi lebih efisien.

Kekurangan: multi thread tidak dapat berjalan atau bekerja secara paralel di dalam multiprosesor karena hanya satu thread saja yang bisa mengakses kernel dalam suatu waktu.

Keterangan :

Model ini ada pada Solaris Green dan GNU Portable.

- Many to Many

  Membolehkan setiap tingkatan user thread dipetakan ke banyak kernel thread.

  Kelebihan: Developer dapat membuat user thread sebanyak yang diperlukan dan kernel thread yang bersangkutan dapat berjalan secara parallel pada multiprocessor. Dan ketika suatu thread menjalankan blocking system call maka kernel dapat menjadwalkan thread lain untuk melakukan eksekusi.

  Kekurangan: Developer dapat membuat user thread sebanyak mungkin, tetapi konkurensi tidak dapat diperoleh karena hanya satu thread yang dapat dijadwalkan oleh kernel pada suatu waktu.

  Model ini ada pada Solaris, IRIX, dan Digital UNIX.