Nama: Daffa Harikhsan
NIM: 23/513044/PA/21918

Tugas 6

3.3.1 Activity 1

race.c > increment

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

void *increment();
void *decrement();
int shared = 1;

int main(){
    pthread_t thread1 , thread2 ;
    pthread_create (newthread: & thread1, attr: NULL, start_routine: increment, arg: NULL ) ;
    pthread_create (newthread: & thread2, attr: NULL, start_routine: decrement, arg: NULL ) ;
    pthread_join ( th: thread1, thread_return: NULL ) ;
    pthread_join ( th: thread2, thread_return: NULL ) ;
    printf (format: "Nilai akhir dari variabel bersama adalah %d\n", shared ) ;
}

void *increment(){
    int x ;
    x = shared;
    printf (format: "Utas 1 membaca nilai dari variabel bersama sebagai : %d\n", x ) ;
    x++;
    printf (format: "Pembaruan lokal oleh Utas 1: %d\n", x ) ;
    sleep (seconds: 1) ;
    shared = x;
    printf (format: "Nilai dari variabel bersama yang diperbarui oleh Utas 1 adalah : %d\n", shared ) ;
}

void *decrement(){
    int y;
    y = shared;
    printf (format: "Utas 2 membaca nilai dari variabel bersama sebagai : %d\n", y );
    y--;
    printf (format: "Pembaruan lokal oleh Utas 2: %d\n", y );
    sleep (seconds: 1);
    shared = y;
    printf (format: "Nilai dari variabel bersama yang diperbarui oleh Utas 2 adalah : %d\n", shared ) ;
}
```

```
dharikhsan@cloudshell:~$ gcc -o race race.c
dharikhsan@cloudshell:~$ ./race
Utas 1 membaca nilai dari variabel bersama sebagai : 1
Pembaruan lokal oleh Utas 1: 2
Utas 2 membaca nilai dari variabel bersama sebagai : 1
Pembaruan lokal oleh Utas 2: 0
Nilai dari variabel bersama yang diperbarui oleh Utas 1 adalah : 2
Nilai dari variabel bersama yang diperbarui oleh Utas 2 adalah : 0
Nilai akhir dari variabel bersama adalah 0
dharikhsan@cloudshell:~$
```

- Is the final result of the shared variable similiar to your friends?

Ya, hasilnya mirip dengan teman-temanku, variabel bersamanya = 0 di akhir

- Why does this happen?

Hal ini terjadi karena masalah race condition. Kedua thread, yaitu increment dan decrement, berjalan secara bersamaan tanpa sinkronisasi yang tepat, sehingga keduanya membaca dan menulis ke variabel shared secara bersamaan. Akibatnya, nilai shared bisa berubah dengan cara yang tidak diinginkan karena satu thread mungkin menimpa perubahan yang dilakukan oleh thread lainnya yang menyebabkan nilai shared = 0.

- Which part of the source code list above is the critical section?

Bagian yang menjadi critical section adalah blok kode yang mengakses dan memodifikasi variabel shared. Pada fungsi increment, bagian yang menjadi critical section adalah: (x = shared; setelah itu x++; dan yang terakhir shared = x), Thread ini membaca nilai dari shared, memodifikasi, dan menulis kembali nilainya. Sedangkan pada fungsi decrement, bagian yang menjadi critical section adalah: (y = shared; setelah itu y--; dan shared = y), thread ini membaca nilai shared, memodifikasi, dan menulis kembali.

3.4.1 Activity 2

Mutex.c > ...

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

void *increment();
void *decrement();
int shared = 1;
pthread_mutex_t lock ;

int main(){
    pthread_mutex_init (mutex: & lock , mutexattr: NULL );
    pthread_t thread1 , thread2 ;
    pthread_create (newthread: & thread1, attr: NULL, start_routine: increment, arg: NULL ) ;
    pthread_create (newthread: & thread2, attr: NULL, start_routine: decrement, arg: NULL ) ;
    pthread_join ( th: thread1, thread_return: NULL ) ;
    pthread_join ( th: thread2, thread_return: NULL ) ;
    printf (format: "Nilai akhir dari variabel bersama adalah %d\n", shared ) ;
}

void *increment(){
    int x;
    printf (format: "Utas 1 mencoba untuk mendapatkan kunci \n");
    pthread_mutex_lock (mutex: & lock );
    printf (format: "Utas 1 memperoleh kunci\n");
    x = shared;
    printf (format: "Utas 1 membaca nilai dari variabel bersama sebagai : %d\n", x );
    x++;
    printf (format: "Pembaruan lokal oleh Utas 1: %d\n", x );
    sleep(seconds: 1);
    shared = x;
    printf (format: "Nilai dari variabel bersama yang diperbarui oleh Utas 1 adalah : %d\n", shared );
    pthread_mutex_unlock (mutex: & lock );
    printf (format: "Utas 1 melepaskan kunci\n");
}

 void *decrement(){
    int y;
     printf (format: "Utas 2 mencoba untuk mendapatkan kunci\n");
     pthread_mutex_lock (mutex: & lock );
     printf (format: "Utas 2 memperoleh kunci\n");
     y = shared;
     printf (format: "Utas 2 membaca nilai dari variabel bersama sebagai : %d\n", y );
     y--;
     printf (format: "Pembaruan lokal oleh Utas 2: %d\n", y );
     sleep (seconds: 1);
     shared = y;
     printf (format: "Nilai dari variabel bersama yang diperbarui oleh Utas 2 adalah : %d\n", shared );
     pthread_mutex_unlock (mutex: & lock );
     printf (format: "Utas 2 melepaskan kunci\n");
 }
```

```
dharikhsan@cloudshell:~$ gcc -o Mutex Mutex.c
dharikhsan@cloudshell:~$ ./Mutex
Utas 1 mencoba untuk mendapatkan kunci
Utas 1 memperoleh kunci
Utas 1 membaca nilai dari variabel bersama sebagai : 1
Pembaruan lokal oleh Utas 1: 2
Utas 2 mencoba untuk mendapatkan kunci
Nilai dari variabel bersama yang diperbarui oleh Utas 1 adalah : 2
Utas 1 melepaskan kunci
Utas 2 memperoleh kunci
Utas 2 membaca nilai dari variabel bersama sebagai : 2
Pembaruan lokal oleh Utas 2: 1
Nilai dari variabel bersama yang diperbarui oleh Utas 2 adalah : 1
Utas 2 melepaskan kunci
Nilai akhir dari variabel bersama adalah 1
dharikhsan@cloudshell:~$
```

- What is the final value of the shared variable?

  Nilai akhir dari variable bersama yaitu = 1.

- Which thread obtains the lock first?

  Thread yang memperoleh kunci terlebih dahulu adalah thread 1, karena ada pada bagian output.

- If the other thread first obtains the lock, what will be the final value of the shared variable?

  Jika thread yang lain yang menerima/memperoleh kunci terlebih dahulu, maka urutan eksekusinya akan berbeda, dan nilai akhir dari variable bersama tetap 1 tidak berubah. Ini terjadi karena kedua thread melakukan modifikasi pada variable shared dengan menambahkan atau mengurangi nilai yang sama (satu increment, dan satu decrement).

3.5.1 Activity 3

semaphore.c > ...

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

void *increment();
void *decrement();
int shared = 1;
sem_t semaphore;

int main(){
    sem_init(sem: &semaphore, pshared: 0, value: 1);
    pthread_t thread1 , thread2 ;
    pthread_create (newthread: & thread1, attr: NULL, start_routine: increment, arg: NULL ) ;
    pthread_create (newthread: & thread2, attr: NULL, start_routine: decrement, arg: NULL ) ;
    pthread_join ( th: thread1, thread_return: NULL ) ;
    pthread_join ( th: thread2, thread_return: NULL ) ;
    printf (format: "Nilai akhir dari variabel bersama adalah %d\n", shared ) ;
}

void *increment(){
    int x;
    printf (format: "Utas 1 mencoba untuk mengurangi semaphore\n");
    sem_wait(sem: &semaphore);
    printf (format: "Utas 1 dapat masuk ke bagian kritisnya\n");
    x = shared;
    printf (format: "Utas 1 membaca nilai dari variabel bersama sebagai : %d\n", x );
    x++;
    printf (format: "Pembaruan lokal oleh Utas 1: %d\n", x );
    sleep(seconds: 1);
    shared = x;
    printf (format: "Nilai dari variabel bersama yang diperbarui oleh Utas 1 adalah : %d\n", shared );
    sem_post(sem: &semaphore);
    printf (format: "Utas 1 menambah semaphore\n");
}

void *decrement(){
    int y ;
    printf (format: "Utas 2 mencoba untuk mengurangi semaphore\n");
    sem_wait (sem: & semaphore );
    printf (format: "Utas 2 dapat masuk ke bagian kritisnya\n");
    y = shared;
    printf (format: "Utas 2 membaca nilai dari variabel bersama sebagai : %d\n", y );
    y--;
    printf (format: "Pembaruan lokal oleh Utas 2: %d\n", y );
    sleep (seconds: 1);
    shared = y;
    printf (format: "Nilai dari variabel bersama yang diperbarui oleh Utas 2 adalah : %d\n", shared );
    sem_post (sem: & semaphore );
    printf (format: "Utas 2 menambah semaphore\n");
}
```

```
dharikhsan@cloudshell:~$ gcc -o semaphore semaphore.c
dharikhsan@cloudshell:~$ ./semaphore
Utas 2 mencoba untuk mengurangi semaphore
Utas 2 dapat masuk ke bagian kritisnya
Utas 2 membaca nilai dari variabel bersama sebagai : 1
Pembaruan lokal oleh Utas 2: 0
Utas 1 mencoba untuk mengurangi semaphore
Nilai dari variabel bersama yang diperbarui oleh Utas 2 adalah : 0
Utas 2 menambah semaphore
Utas 1 dapat masuk ke bagian kritisnya
Utas 1 membaca nilai dari variabel bersama sebagai : 0
Pembaruan lokal oleh Utas 1: 1
Nilai dari variabel bersama yang diperbarui oleh Utas 1 adalah : 1
Utas 1 menambah semaphore
Nilai akhir dari variabel bersama adalah 1
```

- What is the final value of the shared variable?

  Nilai akhir dari variable bersama adalah 1.

- From the final value that you obtained, which thread first obtains access to the shared variable?

Thread 2 memperoleh pertama kali mengakses ke variabel bersama, setelah itu thread 2 melakukan decrement pada variable bersama menjadi 0, lalu thread 1 melakukan increment kembali sehingga nilai akhir variabel bersama adalah 1.

5.5.1 Activity 1

malloc.c > main

```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
int main () {
        int * ptr;
        ptr = (int*) malloc (size: 5 * sizeof (int) ) ;
        if(ptr != NULL) {
            printf (format: "Memory has been successfully allocated.\n");
            printf (format: "Starting address : %p\n",ptr);
            printf (format: "End address : %p\n",ptr+4);
            for(int i =0; i <5; i ++) {
                ptr [ i ] = i +1;
            }
            printf (format: "The elements of the array are :\n");
            for(int i =0; i <5; i ++) {
                printf (format: "%d\n", ptr [i]);
            }
        }
    sleep(seconds: 1000);
}
```

```
dharikhsan@cloudshell:~$ gcc -o malloc malloc.c
dharikhsan@cloudshell:~$ ./malloc &
[1] 3064
Memory has been successfully allocated.
Starting address : 0x56ab6b1352a0
End address : 0x56ab6b1352b0
The elements of the array are :
1
2
3
4
5
dharikhsan@cloudshell:~$ cat /proc/3064/maps
56ab69905000-56ab69906000 r--p 00000000 08:11 131385                     /home/dharikhsan/malloc
56ab69906000-56ab69907000 r-xp 00001000 08:11 131385                     /home/dharikhsan/malloc
56ab69907000-56ab69908000 r--p 00002000 08:11 131385                     /home/dharikhsan/malloc
56ab69908000-56ab69909000 r--p 00002000 08:11 131385                     /home/dharikhsan/malloc
56ab69909000-56ab6990a000 rw-p 00003000 08:11 131385                     /home/dharikhsan/malloc
56ab6b135000-56ab6b156000 rw-p 00000000 00:00 0                          [heap]
7ba166659000-7ba16665c000 rw-p 00000000 00:00 0
7ba16665c000-7ba166684000 r--p 00000000 08:01 3528850                    /usr/lib/x86_64-linux-gnu/libc.so.6
7ba166684000-7ba16680c000 r-xp 00028000 08:01 3528850                    /usr/lib/x86_64-linux-gnu/libc.so.6
7ba16680c000-7ba16685b000 r--p 001b0000 08:01 3528850                    /usr/lib/x86_64-linux-gnu/libc.so.6
7ba16685b000-7ba16685f000 r--p 001fe000 08:01 3528850                    /usr/lib/x86_64-linux-gnu/libc.so.6
7ba16685f000-7ba166861000 rw-p 00202000 08:01 3528850                    /usr/lib/x86_64-linux-gnu/libc.so.6
7ba166861000-7ba16686e000 rw-p 00000000 00:00 0
7ba166878000-7ba16687a000 rw-p 00000000 00:00 0
7ba16687a000-7ba16687b000 r--p 00000000 08:01 3528604                    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7ba16687b000-7ba1668a6000 r-xp 00001000 08:01 3528604                    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7ba1668a6000-7ba1668b0000 r--p 0002c000 08:01 3528604                    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7ba1668b0000-7ba1668b2000 r--p 00036000 08:01 3528604                    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7ba1668b2000-7ba1668b4000 rw-p 00038000 08:01 3528604                    /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
7fff53821000-7fff53843000 rw-p 00000000 00:00 0                          [stack]
7fff53891000-7fff53895000 r--p 00000000 00:00 0                          [vvar]
7fff53895000-7fff53897000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
```

- In what part of the memory is the pointer pointed to?

Pointer menunjuk ke alamat awal dari blok memori yang dialokasikan yakni 0x55aeb915a520. Alokasi memori terjadi pada alamat yang dimulai dari 0x55aeb915a520 dan berakhir di alamat 0x55aeb915a530. Pointer menunjuk ke blok memori yang dialokasikan secara dinamis menggunakan fungsi malloc (malloc()).

- What does it mean in terms of memory allocation?

Dalam program ini penggunaan fungsi malloc digunakan untuk mengalokasikan memori dinamis sebesar 5 elemen integer, serta menampilkan alamat awal dan akhir dari blok memori yang dialokasikan. Setelah dialokasikan, program mengisi array dengan nilai 1 hingga 5 dan menampilkannya

5.6.1 Activity 2

realloc.c > main

```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
int main () {
    int *ptr;
    ptr = (int*) malloc (size: 5 * sizeof (int) );
    if(ptr != NULL) {
        printf (format: "Memory has been successfully allocated.\n") ;
        printf (format: "Starting address : %p\n", ptr );
        printf (format: "End address : %p\n", ptr +4);
        for(int i =0; i <5; i++) {
            ptr [i] = i+1;
        }
        /*ptr = realloc (ptr, 10 * sizeof (int));
        printf ("Successfully realocated the pointer\n");
        for(int i =5; i <10; i++) {
            ptr [i] = i+1;
        }*/
        printf (format: "The elements of the array are :\n");
        for(int i =0; i <10; i++) {
            printf (format: "%d\n", ptr [i]);
        }
    }
}
```

```
dharikhsan@cloudshell:~$ gcc -o realloc realloc.c
dharikhsan@cloudshell:~$ ./realloc
Memory has been successfully allocated.
Starting address : 0x56cf313472a0
End address : 0x56cf313472b0
The elements of the array are :
1
2
3
4
5
0
1041
0
825494064
909128761
dharikhsan@cloudshell:~$
```

- What is the result?

Hasil program tersebut setelah dijalankan menampilkan 5 elemen array pertama dengan lancar, tetapi untuk 5 elemen berikutnya menunjukkan angka yang acak. Ini terjadi karena Alokasi memori awal berhasil, namun kode untuk alokasi ulang menggunakan realloc dikomentari sehingga tidak dijalankan.

- Why does that happen?

Yang terjadi karena meskipun malloc mengalokasikan memori untuk 5 elemen, program mencoba mencetak 10 elemen. Karena kode untuk alokasi ulang memori (realloc) dikomentari, hanya 5 elemen yang benar-benar dialokasikan, dan elemen di luar rentang tersebut berisi nilai acak dari memori yang tidak terinisialisasi atau tidak dialokasikan.

5.7.1 Activity

free.c > main

```c
#include <stdlib.h>
#include <stdio.h>
int main () {
    int *ptr ;
    ptr = (int*) malloc (size: 5 * sizeof (int));
    if(ptr != NULL) {
        printf (format: "Memory has been successfully allocated.\n");
        printf (format: "Starting address : %p\n", ptr);
        printf (format: "End address : %p\n", ptr +4);
        free (ptr) ;
        for(int i =0; i <5; i ++) {
            ptr [i] = i +1;
        }
        printf (format: "The elements of the array are :\n");
        for(int i =0; i <5; i ++) {
            printf (format: "%d\n", ptr [i]);
        }
    }
}
```

```
dharikhsan@cloudshell:~$ gcc -o free free.c
dharikhsan@cloudshell:~$ ./free
Memory has been successfully allocated.
Starting address : 0x5a04b5fca2a0
End address : 0x5a04b5fca2b0
The elements of the array are :
1
2
3
4
5
dharikhsan@cloudshell:~$
```

- What is the result?

Program menunjukkan bahwa memori dialokasikan untuk 5 elemen integer, dan alokasi memori tersebut berhasil. Setelah itu, memori dibebaskan dengan fungsi free(). Namun, program tetap berusaha untuk mengisi dan mencetak nilai dari pointer yang sudah dibebaskan. Dan juga hasilnya tetap sesuai dengan nilai yang diharapkan

- Can the pointer still be used to store and print values? Why?

Pointer tersebut tidak boleh digunakan untuk menyimpan dan mencetak nilai setelah memori dibebaskan dengan free(). Secara teknis, setelah memori dibebaskan, akses ke pointer tersebut adalah undefined behavior, artinya program bisa menampilkan hasil yang tidak terduga, crash, atau (seperti dalam kasus ini) kebetulan masih berfungsi dengan baik karena memori tersebut belum ditimpa. Namun, ini bukanlah praktik yang aman atau benar.