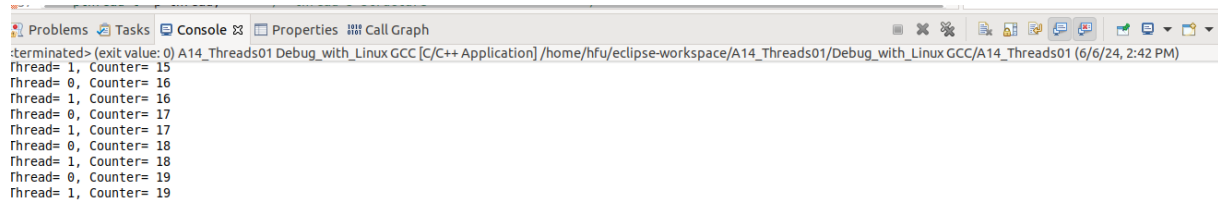


Aufgabenblatt 9

Julian Bertol

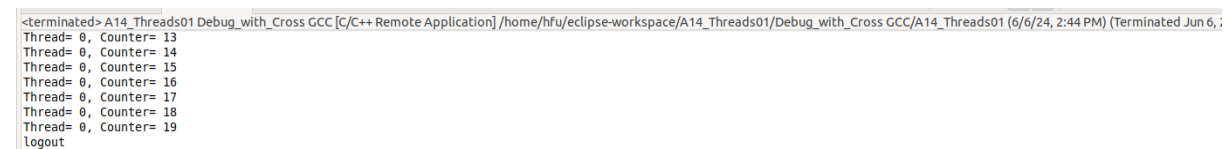
Aufgabe 1:

Ausführung des Codes auf dem Host:



```
Problems Tasks Console Properties Call Graph
:terminated> (exit value: 0) A14_Threads01 Debug_with_Linux GCC [C/C++ Application] /home/hfu/eclipse-workspace/A14_Threads01/Debug_with_Linux GCC/A14_Threads01 (6/6/24, 2:42 PM)
Thread= 1, Counter= 15
Thread= 0, Counter= 16
Thread= 1, Counter= 16
Thread= 0, Counter= 17
Thread= 1, Counter= 17
Thread= 0, Counter= 18
Thread= 1, Counter= 18
Thread= 0, Counter= 19
Thread= 1, Counter= 19
```

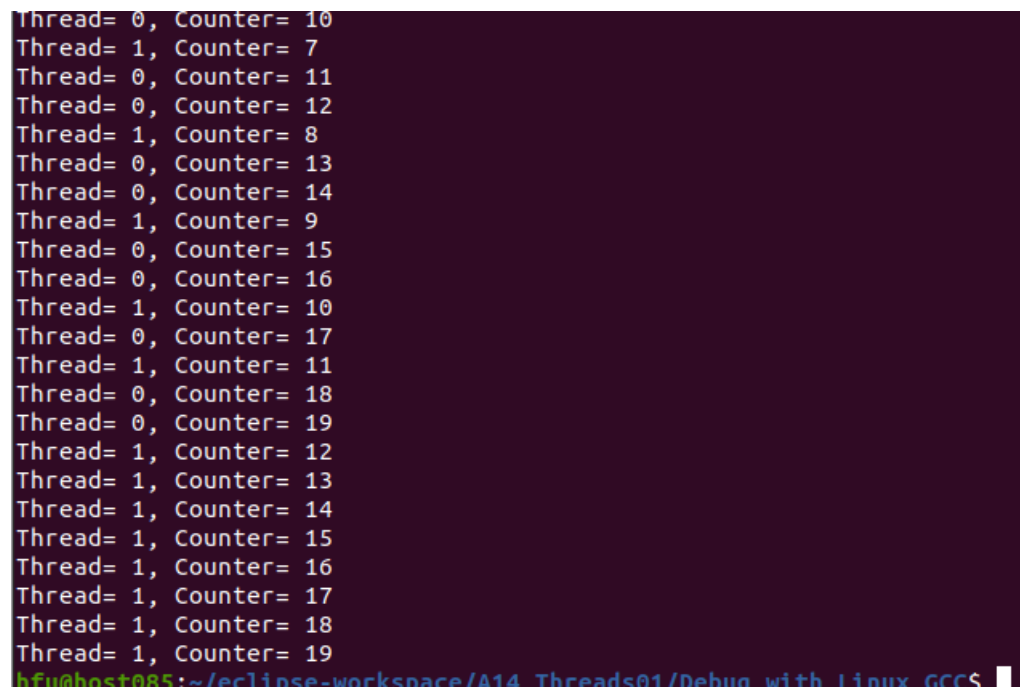
Ausführung des Codes auf dem Target:



```
<terminated> A14_Threads01 Debug_with_Cross GCC [C/C++ Remote Application] /home/hfu/eclipse-workspace/A14_Threads01/Debug_with_Cross GCC/A14_Threads01 (6/6/24, 2:44 PM) (Terminated Jun 6, 2024)
Thread= 0, Counter= 13
Thread= 0, Counter= 14
Thread= 0, Counter= 15
Thread= 0, Counter= 16
Thread= 0, Counter= 17
Thread= 0, Counter= 18
Thread= 0, Counter= 19
logout
```

Vor der Ausführung muss die Bibliothek Pthread dem Cross und dem Lokalen Linker hinzugefügt werden.

Ausführung in der Console auf dem Host:



```
Thread= 0, Counter= 10
Thread= 1, Counter= 7
Thread= 0, Counter= 11
Thread= 0, Counter= 12
Thread= 1, Counter= 8
Thread= 0, Counter= 13
Thread= 0, Counter= 14
Thread= 1, Counter= 9
Thread= 0, Counter= 15
Thread= 0, Counter= 16
Thread= 1, Counter= 10
Thread= 0, Counter= 17
Thread= 1, Counter= 11
Thread= 0, Counter= 18
Thread= 0, Counter= 19
Thread= 1, Counter= 12
Thread= 1, Counter= 13
Thread= 1, Counter= 14
Thread= 1, Counter= 15
Thread= 1, Counter= 16
Thread= 1, Counter= 17
Thread= 1, Counter= 18
Thread= 1, Counter= 19
hfu@host085:~/eclipse-workspace/A14_Threads01/Debug_with_Linux_GCC$
```

Man kann beobachten, dass die 1 und 0 regelmäßig in der Eclipse auftauchen und in der Console unregelmäßig.

Wenn man den Wert der inneren Schleife von 50000 auf 500000 ändert läuft jeder Thread öfter durch die Schleife. Dadurch kann es zu längeren Pausen zwischen den Ausgaben kommen.

Aufgrund der schwächeren CPU Leistung dauert es auf dem Target länger als auf dem Host.

Aufgabe 2:

```
copy_employee(struct employee* from, struct employee* to)
{
    to->number = from->number;
    to->id = from->id;
    strcpy(to->first_name, from->first_name);
    usleep (1000);
    strcpy(to->last_name, from->last_name);
    strcpy(to->department, from->department);
    to->room_number = from->room_number;
}
```

Wenn man hier das usleep erhöht, dann verzögert sich die komplette Laufzeit des Programmes, da es länger dauert irgendwelche Daten hin und her zu kopieren.

Dieses Usleep kann weggelassen werden, da es den Code fehleranfällig macht und die Laufzeit verlängert. Wenn es weggelassen wird, funktioniert das Programm ohne Fehler!

Aufgabe 3:

Wenn man nun ein usleep einbaut, dauert die Ausgabe zwar etwas länger, aber es kommt nicht zu einer Fehlermeldung.

Aufgabe 4:

Auf dem Host funktioniert es ohne Probleme, auf dem Target kommt es zu einem Deadlock, da nicht jeder eine Gabel bekommt.

Aufgabe 5:

Der Fehler liegt an den For-Schleifen, man muss in jeder For-Schleife das Argument von (i = 1; i<=5; i++) auf (i=0;i<5;i++) ändern.

Aufgabe 6:

```
void *philowork(void *num)
{
    int n = (int)(long)num;
    int eating_rounds = 0;

    while(eating_rounds < MAX_EATING_ROUNDS) {
        printf("\nPhilosopher %d is thinking ", n);

        if(n % 2 == 0) {
            // Even philosophers pick up left chopstick first
            pthread_mutex_lock(&chopstick[n]);
            printf(" -> %d got left chopstick ", n);
            pthread_mutex_lock(&chopstick[(n+1)%5]);
            printf(" -> %d got right chopstick ", n);
        } else {
            // Odd philosophers pick up right chopstick first
            pthread_mutex_lock(&chopstick[(n+1)%5]);
            printf(" -> %d got right chopstick ", n);
            pthread_mutex_lock(&chopstick[n]);
            printf(" -> %d got left chopstick ", n);
        }

        printf("\nPhilosopher %d is eating ", n);

        pthread_mutex_unlock(&chopstick[n]);
        pthread_mutex_unlock(&chopstick[(n+1)%5]);
        printf("\nPhilosopher %d finished eating ", n);

        eating_rounds++;
    }

    return NULL;
}
```