

Sheet 05

PS Parallel Programming

Patrick Wintner

April 6, 2025

1 Missing Flush Directives

Effects of missing flush directives are observed.

1.1 Source Code

```
1  #include <omp.h>
2  #include <stdio.h>
3
4  int main() {
5
6      int data;
7      int flag = 0;
8
9      #pragma omp parallel num_threads(2)
10     {
11
12         if (omp_get_thread_num() == 0) {
13
14             data = 42;
15
16             flag = 1;
17
18         } else if (omp_get_thread_num() == 1) {
19
20             int flag_val = 0;
```

```

21
22     while (flag_val < 1) {
23
24         flag_val = flag;
25
26     }
27
28     printf("flag=%d data=%d\n", flag, data);
29
30 }
31
32 }
33
34 return 0;
35 }

```

The program spawns two threads. Thread 0 does some work (setting the value of the variable data) before setting a flag. Thread 1 should print the values of the flag and the variable after the other thread has finished his work.

1.2 Experiment Method

The experiment was done on the LCC3 cluster by calling

```
salloc -exclusive -tasks-per-node=1 -cpus-per-task=1 srun -pty bash .
```

Followed by calling

```
./main.sh
```

manually. The following scripts are involved in the experiment.

1.2.1 Main Script

```

1  #!/bin/bash
2  # Usage: ./main.sh
3  make
4  for i in {0..999}

```

```
5 do
6     ./ex1
7 done
8 make clean
```

1.3 Experiment Results

The program neither terminates nor prints any output. This is probable the case because thread 1 does not fetch the updated value of flag from shared memory. and is thus not able to leave the loop.

1.4 Discussion

The program does indeed require several flush directives, see the code below. Those are placed either after a variable (that is read in another thread) is written to or before a variable (that is written in another thread) is read. It does not require any atomic directives, because there is no variable to which is written in both threads.

```
1  #include <omp.h>
2  #include <stdio.h>
3
4  int main() {
5
6      int data;
7      int flag = 0;
8
9      #pragma omp parallel num_threads(2)
10     {
11
12         if (omp_get_thread_num() == 0) {
13
14             data = 42;
15             #pragma omp flush(data)
16
17             flag = 1;
18             #pragma omp flush(flag)
19
20         } else if(omp_get_thread_num() == 1) {
21
22             int flag_val = 0;
```

```
23
24     while (flag_val < 1) {
25
26         #pragma omp flush(flag)
27         flag_val = flag;
28
29     }
30
31     #pragma omp flush(data)
32     printf("flag=%d data=%d\n", flag, data);
33
34 }
35
36 }
37
38 return 0;
39 }
```

2 Parallelising Code Snippets