

# OpenMP Synchronization (5A)

---

Copyright (c) 2024 - 2016 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using LibreOffice.

# Based on

---

<https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf>

# Synchronization (1)

## Synchronization I

- Threads communicate through shared variables.

Uncoordinated access of these variables can lead to undesired effects.

– E.g. two threads update (write) a shared variable in the same step of execution, the result is dependent on the way this variable is accessed. This is called a race condition.

<https://www3.nd.edu/~zxu2/acms60212-40212-S12/Lec-11-02.pdf>

# Synchronization (2)

- To prevent race condition, the access to shared variables must be synchronized.
- Synchronization can be time consuming.
- The barrier directive is set to synchronize all threads.

All threads wait at the barrier until all of them have arrived.

<https://www3.nd.edu/~zxu2/acms60212-40212-S12/Lec-11-02.pdf>

# Synchronization (3)

## Synchronization II

- Synchronization imposes order constraints and is used to protect access to shared data
- High level synchronization:
  - critical
  - atomic
  - barrier
  - ordered
- Low level synchronization
  - flush
  - locks (both simple and nested)

<https://www3.nd.edu/~zxu2/acms60212-40212-S12/Lec-11-02.pdf>

# Critical (1)

Synchronization: critical

- Mutual exclusion: only one thread at a time can enter a critical region.

```
{  
double res;  
#pragma omp parallel  
{  
double B;  
int i, id, nthrds;  
id = omp_get_thread_num();  
nthrds = omp_get_num_threads();  
for(i=id; i<niters; i+=nthrds){  
B = some_work(i);  
#pragma omp critical  
consume(B,res);  
}  
}  
}  
} https://www3.nd.edu/~zxu2/acms60212-40212-S12/Lec-11-02.pdf
```

## References

- [1] <ftp://ftp.geoinfo.tuwien.ac.at/navratil/HaskellTutorial.pdf>
- [2] <https://www.umiacs.umd.edu/~hal/docs/daume02yaht.pdf>