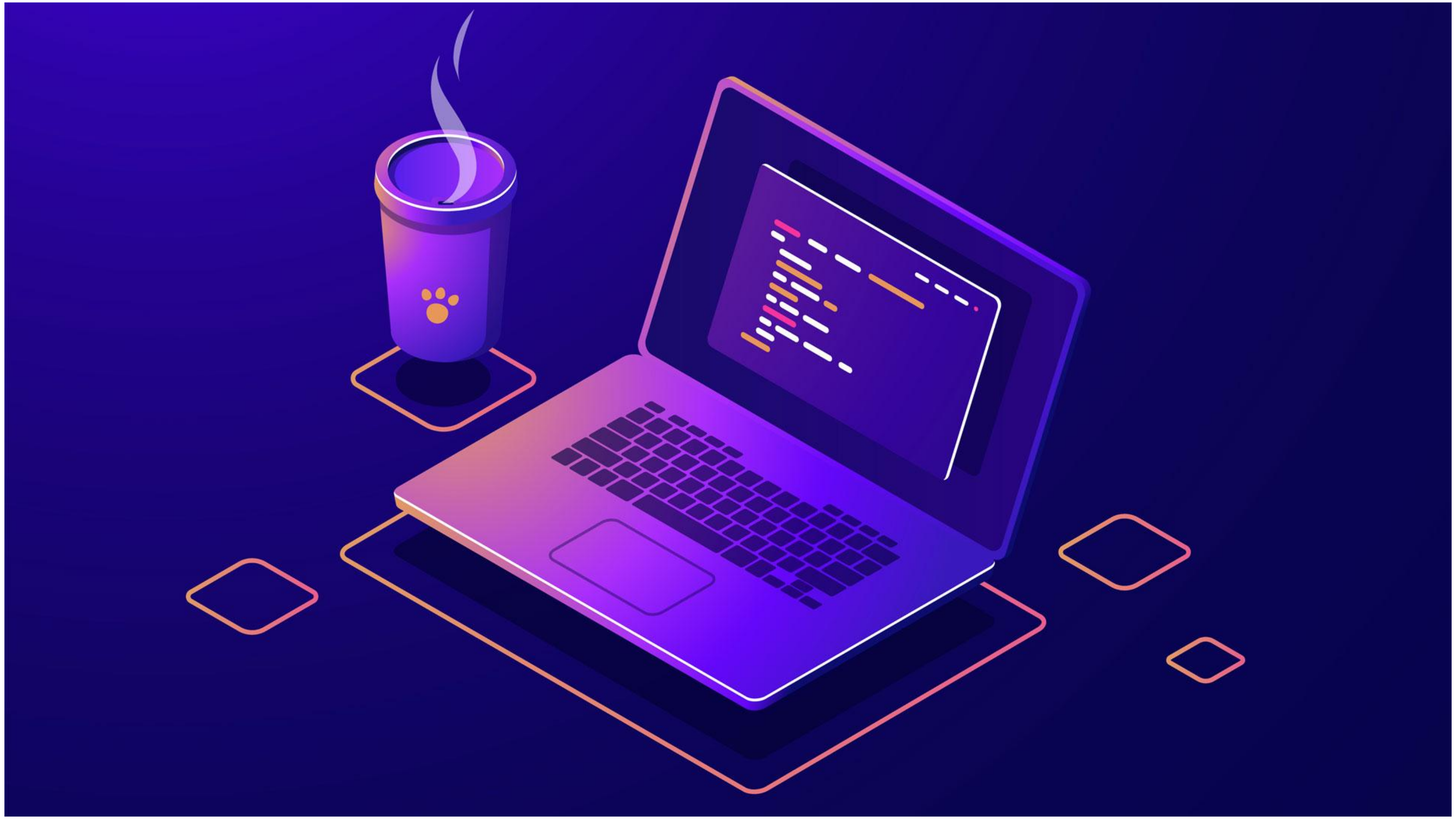
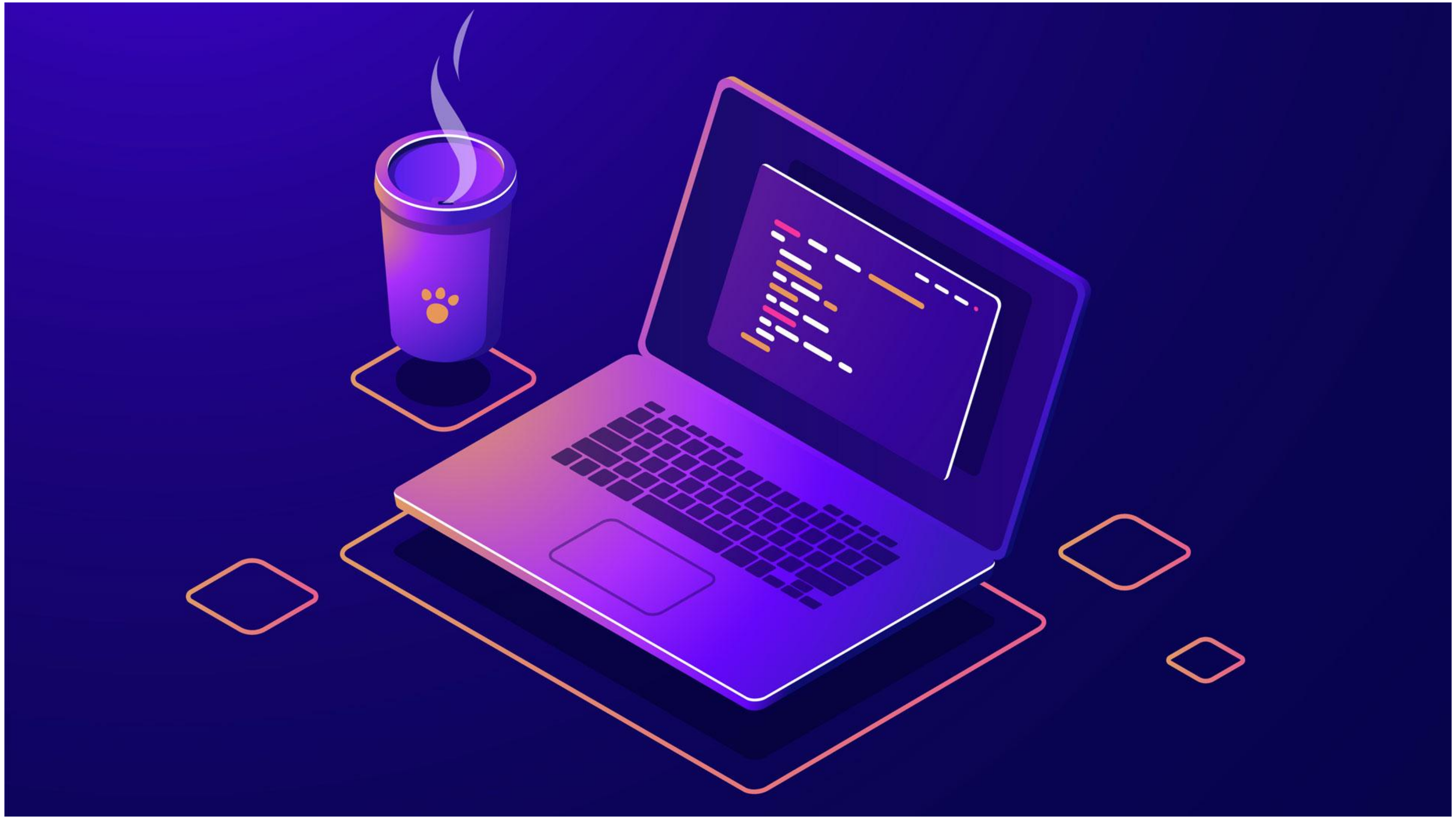
Reduce overhead and serialization



Critical Directive



Critical Directive

```
#pragma omp critical [(name)]
    structured block
```

- Provides a region of mutual exclusion where only one thread can be working at any given time
- By default all critical regions are the same
- Multiple mutual exclusion regions by providing them with a name
  - Only those with the same name synchronize

Example: Computation of Pi

```
void main ()
{
    int i, id;
    double x, pi, sum=0.0;

    step = 1.0/(double) num_steps;
    omp_set_num_threads(NUM_THREADS);
    #pragma omp parallel private(x, i, id)
    {
        id = omp_get_thread_num();
        for (i=id+1; i<=num_steps; i=i+NUM_THREADS) {
            x = (i-0.5)*step;
            #pragma omp critical
            sum = sum + 4.0/(1.0+x*x);
        }
    }
    pi = sum * step;
}
```

Critical Directive

```
int x=1,y=0;
#pragma omp parallel num_threads(4)
{
#pragma omp critical (x)
      x++;
#pragma omp critical (y)
      y++;
}
```

Different names: One thread can update x while another updates y

Instructor Social Media

Youtube: Lucas Science

Instagram: lucaasbazilio

Twitter: lucasebazilio