19 nov 07 17:12 **hello.c** Page 1/1

```c
#include <stdio.h>

int main() {

#pragma omp parallel
  printf("Hello Worlds\n");

}

/*
$ mpcc −qsmp=omp hello.c −o hello
$ export OMP_NUM_THREADS=2
$ ./hello
Hello Worlds
Hello Worlds
*/
```

19 nov 07 17:14 **hello_from.c** Page 1/1

```c
#include <stdio.h>
#include <omp.h>

int main() {

#pragma omp parallel
  printf("Hello Worlds from %d\n",
         omp_get_thread_num());
}


/*
$ mpcc −qsmp=omp hello_from.c −o hello_from
$ export OMP_NUM_THREADS=2
$ ./hello_from
Hello Worlds from 0
Hello Worlds from 1
*/
```

19 nov 07 17:15 **main.c** Page 1/1

```c
#include <stdio.h>
#include <omp.h>

int main() {
  int t = omp_get_num_procs();
  omp_set_num_threads(t);

#pragma omp parallel
#pragma omp master
  printf("Qui suis−je et combien sommes−nous ? %d %d\n",
         omp_get_thread_num(), omp_get_num_threads());
}

/*
IBM P690−Power4
$ ./soumission ./main
Qui suis−je et combien sommes−nous ? 0 32
*/

/*
Origin 3800
$ ./soumission ./main
Qui suis−je et combien sommes−nous ? 0 512
*/
```

15 nov 07 12:00 **main.c** Page 1/1

```c
#include <stdio.h>
#include <omp.h>
int main()
{
  float a;
  int   p;

  a = 92290. ;
  p = 0;
#pragma omp parallel
  {
#ifdef _OPENMP
    p=omp_in_parallel();
#endif
    printf("a vaut : %f ; p vaut : %d\n",a,p);
  }
  return 0;
}

/*
> xlc_r ... −qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out
a vaut : 92290. ; p vaut : 1
a vaut : 92290. ; p vaut : 1
a vaut : 92290. ; p vaut : 1
a vaut : 92290. ; p vaut : 1
*/
```

15 nov 07 12:01 **main.c** Page 1/1

```c
#include <stdio.h>
#include <omp.h>
int main()
{
  float a;

  a = 92000.;
#pragma omp parallel default(none) private(a)
  {
    a = a + 290.;
    printf("a vaut : %f\n",a);
  }
  return 0;
}

/*
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out


a vaut : 290.
a vaut : 290.
a vaut : 290.
a vaut : 290.
*/
```

15 nov 07 12:01 **main.c** Page 1/1

```c
#include <stdio.h>
int main()
{
  float a;

  a = 92000.;
#pragma omp parallel default(none) firstprivate(a)
  {
    a = a + 290.;
    printf("a vaut : %f\n",a);
  }
  printf("Hors region, a vaut : %f\n", a);
  return 0;
}

/*
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out


a vaut : 92290.
a vaut : 92290.
a vaut : 92290.
a vaut : 92290.
Hors region, a vaut : 92000.
*/
```

13 nov 07 17:31 **main.c** Page 1/1

```c
void sub(void);

int main()
{
#pragma omp parallel
  {
    sub();
  }
  return 0;
}

/*
> xlc_r ... -qsmp=omp main.c sub.c
> export OMP_NUM_THREADS=4
> a.out


Parallele ? : 1
Parallele ? : 1
Parallele ? : 1
Parallele ? : 1
*/
```

13 nov 07 17:30 **sub.c** Page 1/1

```c
#include <stdio.h>
#include <omp.h>

void sub(void)
{
  int p=0;

#ifdef _OPENMP
  p = omp_in_parallel();
#endif
  printf("Parallele ? : %d\n", p);
}
```

13 nov 07 17:34 **main.c** Page 1/1

```
void sub(void);

int main()
{
#pragma omp parallel default(shared)
  {
    sub();
  }
  return 0;
}

/*
> xlc_r ... -qsmp=omp main.c sub.c
> export OMP_NUM_THREADS=4
> a.out


a vaut : 92293
a vaut : 92291
a vaut : 92292
a vaut : 92290
*/
```

13 nov 07 17:34 **sub.c** Page 1/1

```
#include <stdio.h>
#include <omp.h>

void sub(void)
{
  int a;

  a=92290;
  a = a + omp_get_thread_num();
  printf("a vaut : %d\n", a);
}
```

15 nov 07 12:04 **main.c** Page 1/1

```
#include <stdio.h>

void sub(int x, int *y);

int main()
{
  int a, b;

  a = 92000;
#pragma omp parallel shared(a) private(b)
  {
    sub(a, &b);
    printf("b vaut : %d\n",b);
  }
  return 0;
}

/*
> xlc_r ... -qsmp=omp main.c sub.c
> export OMP_NUM_THREADS=4
> a.out


b vaut : 92003
b vaut : 92001
b vaut : 92002
b vaut : 92000
*/
```

13 nov 07 17:35 **sub.c** Page 1/1

```
#include <omp.h>

void sub(int x, int *y)
{
  *y = x + omp_get_thread_num();
}
```

13 nov 07 17:37 **main.c** Page 1/1

```
void sub(void);
float a;

int main()
{
  a = 92000;
#pragma omp parallel
  {
    sub();
  }
  return 0;
}

/*
> xlc_r ... -qsmp=omp main.c sub.c
> export OMP_NUM_THREADS=2
> a.out


B vaut : 92290
B vaut : 92290
*/
```

13 nov 07 17:37 **sub.c** Page 1/1

```
#include <stdio.h>

float a;

void sub(void)
{
  float b;

  b = a + 290.;
  printf("b vaut : %f\n",b);
}
```

15 nov 07 12:05 **main.c** Page 1/1

```
#include <stdio.h>
#include <omp.h>

void sub(void);
int a;
#pragma omp threadprivate(a)

int main()
{
  a = 92000;
#pragma omp parallel copyin(a)
  {
    a = a + omp_get_thread_num();
    sub();
  }
  printf("Hors region, A vaut: %d\n",a);
  return 0;
}

/*
> xlc_r ... -qsmp=omp main.c sub.c
> export OMP_NUM_THREADS=4
> a.out
B vaut : 92290
B vaut : 92291
B vaut : 92292
B vaut : 92293
Hors region, A vaut : 92000
*/
```

13 nov 07 17:38 **sub.c** Page 1/1

```
#include <stdio.h>

int a;
#pragma omp threadprivate(a)

void sub(void)
{
  int b;

  b = a + 290;
  printf("b vaut : %d\n",b);
}
```

13 nov 07 17:40 **main.c** Page 1/2

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
int main()
{
  int n, debut, fin, rang, nb_taches, i;
  float *a;

  n=1024, nb_taches=4;
  a=(float *) malloc(n*nb_taches*sizeof(float));
#pragma omp parallel default(none)                          \
  private(debut,fin,rang,i) shared(a,n) if(n > 512)
  {
    rang=omp_get_thread_num();
    debut=rang*n;
    fin=(rang+1)*n;
    for (i=debut; i<fin; i++)
      a[i] = 92291. + (float) i;
    printf("Rang : %d ; A[%.4d],...,A[%.4d] : %#.0f,...,%#.0f \n",
           rang,debut,fin−1,a[debut],a[fin−1]);
  }
  free(a);
  return 0;
}

/*
> xlc_r ... −qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out
```

13 nov 07 17:40 **main.c** Page 2/2

```
Rang : 3 ; A[3072],...,A[4095] : 95363.,...,96386.
Rang : 0 ; A[0000],...,A[1023] : 92291.,...,93314.
Rang : 1 ; A[1024],...,A[2047] : 93315.,...,94338.
Rang : 2 ; A[2048],...,A[3071] : 94339.,...,95362.
*/
```

13 nov 07 17:42 **main.c** Page 1/2

```
#include <stdio.h>
#include <omp.h>

int main()
{
  int rang;

#pragma omp parallel private(rang) num_threads(3)
  {
    rang=omp_get_thread_num();
    printf("Mon rang dans region 1 : %d \n",rang);
#pragma omp parallel private(rang) num_threads(2)
    {
      rang=omp_get_thread_num();
      printf("  Mon rang dans region 2 : %d \n",rang);
    }
  }
  return 0;
}

/*
> xlc_r ... −qsmp=nested_par main.c
> export OMP_DYNAMIC=true
> export MP_NESTED=true
> a.out


 Mon rang dans region 1 : 0
   Mon rang dans region 2 : 1
```

13 nov 07 17:42 **main.c** Page 2/2

```
   Mon rang dans region 2 : 0
 Mon rang dans region 1 : 2
   Mon rang dans region 2 : 1
   Mon rang dans region 2 : 0
 Mon rang dans region 1 : 1
   Mon rang dans region 2 : 0
   Mon rang dans region 2 : 1

*/
```

15 nov 07 12:05 **main.c** Page 1/2

```
#include <stdio.h>
#include <omp.h>

#define N 4096
#define max(a,b) ((a) > (b) ? (a) : (b))
#define min(a,b) ((a) < (b) ? (a) : (b))

int main()
{
  float a[N];
  int i, i_min, i_max, rang, nb_taches;

#pragma omp parallel private(rang,i_min,i_max) shared(a,nb_taches)
  {
    rang=omp_get_thread_num();
    nb_taches=omp_get_num_threads();
    i_min=N, i_max=0;
#pragma omp for schedule(static,N/nb_taches) nowait
    for (i=0; i<N; i++) {
      a[i] = 92291. + (float) i;
      i_min=min(i_min,i);
      i_max=max(i_max,i);
    }
    printf("Rang : %d; i_min : %d; i_max : %d\n",rang,i_min,i_max);
  }
  return 0;
}

/*
```

15 nov 07 12:05 **main.c** Page 2/2

```
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out


Rang : 3 ; i_min : 3072 ; i_max : 4095
Rang : 0 ; i_min : 0    ; i_max : 1023
Rang : 1 ; i_min : 1024 ; i_max : 2047
Rang : 2 ; i_min : 2048 ; i_max : 3071

*/
```

13 nov 07 17:45 **main.c** Page 1/2

```
#include <stdio.h>
#include <omp.h>

#define N 4096
#define max(a,b) ((a) > (b) ? (a) : (b))
#define min(a,b) ((a) < (b) ? (a) : (b))

int main()
{
  float a[N];
  int i, i_min, i_max, rang;

#pragma omp parallel private(rang,i_min,i_max) shared(a,nb_taches)
  {
    rang=omp_get_thread_num();
    i_min=N, i_max=0;
#pragma omp for schedule(runtime)
    for (i=0; i<N; i++) {
      a[i] = 92291. + (float) i;
      i_min=min(i_min,i);
      i_max=max(i_max,i);
    }
    printf("Rang : %d; i_min : %d; i_max : %d\n",rang,i_min,i_max);
  }
  return 0;
}


/*
```

13 nov 07 17:45 **main.c** Page 2/2

```
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=2
> export OMP_SCHEDULE="STATIC,2048"
> a.out


Rang : 0 ; i_min : 0    ; i_max : 2047
Rang : 1 ; i_min : 2048 ; i_max : 4095
*/
```

13 nov 07 17:46 **main.c** Page 1/2

```c
#include <stdio.h>
#include <omp.h>

#define N 9

int main()
{
  int i, rang;

#pragma omp parallel default(none) private(rang,i)
  {
    rang=omp_get_thread_num();
#pragma omp for schedule(runtime) ordered nowait
    for (i=0; i<N; i++) {
#pragma omp ordered
      {
        printf("Rang : %d ; iteration : %d\n",rang,i);
      }
    }
  }
  return 0;
}

/*
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=4
> export OMP_SCHEDULE="STATIC,2"
> a.out
```

13 nov 07 17:46 **main.c** Page 2/2

```
Rang : 0 ; iteration : 0
Rang : 0 ; iteration : 1
Rang : 1 ; iteration : 2
Rang : 1 ; iteration : 3
Rang : 2 ; iteration : 4
Rang : 2 ; iteration : 5
Rang : 3 ; iteration : 6
Rang : 3 ; iteration : 7
Rang : 0 ; iteration : 8

*/
```

15 nov 07 12:06 **main.c** Page 1/1

```c
#include <stdio.h>

#define N 5

int main()
{
  int i, s=0, p=1, r=1;

#pragma omp parallel
  {
#pragma omp for reduction(+:s) reduction(*:p,r)
    for (i=0; i<N; i++) {
      s = s + 1;
      p = p * 2;
      r = r * 3;
    }
  }
  printf("s = %d ; p = %d ; r = %d\n",s,p,r);
  return 0;
}


/*
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out
s = 5 ; p = 32 ; r = 243
*/
```

15 nov 07 12:07 **main.c** Page 1/2

```c
#include <stdio.h>
#include <omp.h>

#define N 10

int main()
{
  int i,rang;
  float temp;

#pragma omp parallel private(rang)
  {
#pragma omp for lastprivate(temp)
    for (i=0; i<N; i++)
      temp = (float) i;

    rang=omp_get_thread_num();
    printf("Rang : %d; temp = %f\n", rang, temp);
  }
  return 0;
}


/*
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out
```

```
Rang : 0; temp = 9.000000
Rang : 3; temp = 9.000000
Rang : 1; temp = 9.000000
Rang : 2; temp = 9.000000

*/

/* avec parallel for
#include <stdio.h>
#define N 9
int main()
{
  int i;
  float temp;

#pragma omp parallel for lastprivate(temp)
  for (i=0; i<N; i++)
    temp = (float) i;

  return 0;
}
*/
```



```
#include <stdio.h>
#include <omp.h>

int main()
{
  int rang;
  float a;

#pragma omp parallel private(a,rang)
  {
    a = 92290.;
#pragma omp single
    {
      a = −92290.;
    }
    rang=omp_get_thread_num();
    printf("Rang : %d ; A vaut : %f\n",rang,a);
  }
  return 0;
}
/*
> xlc_r ... −qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out
Rang : 1 ; A vaut :  92290.
Rang : 2 ; A vaut :  92290.
Rang : 0 ; A vaut :  92290.
Rang : 3 ; A vaut : −92290.
*/
```



```
#include <stdio.h>
#include <omp.h>

int main()
{
  int rang;
  float a;
#pragma omp parallel private(a,rang)
  {
    a = 92290.;
#pragma omp single copyprivate(a)
    {
      a = −92290.;
    }
    rang=omp_get_thread_num();
    printf("Rang : %d ; A vaut : %f\n",rang,a);
  }
  return 0;
}
/*
> xlc_r ... −qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out
Rang : 1 ; A vaut : −92290.
Rang : 2 ; A vaut : −92290.
Rang : 0 ; A vaut : −92290.
Rang : 3 ; A vaut : −92290.
*/
```



```
#include <stdio.h>
#include <omp.h>

int main()
{
  int rang;
  float a;
#pragma omp parallel private(a,rang)
  {
    a = 92290.;
#pragma omp master
    {
      a = −92290.;
    }
    rang=omp_get_thread_num();
    printf("Rang : %d ; A vaut : %f\n",rang,a);
  }
  return 0;
}
/*
> xlc_r ... −qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out
Rang : 1 ; A vaut :  92290.
Rang : 2 ; A vaut :  92290.
Rang : 0 ; A vaut : −92290.
Rang : 3 ; A vaut :  92290.
*/
```

13 nov 07 17:57 **main.c** Page 1/1

```c
#include <stdlib.h>

#define N 1025

void prod_mat_vect(float a[][N],float x[N],float y[N]);

int main()
{
  int i, j;
  float a[N][N], x[N], y[N];

  for(j=0; j<N; j++) {
    x[j]=(float) drand48();
    y[j]=0.;
    for(i=0; i<N; i++)
      a[i][j]=(float) drand48();
  }
#pragma omp parallel if(N > 256)
  {
    prod_mat_vect(a,x,y);
  }
  return 0;
}
```

13 nov 07 17:57 **sub.c** Page 1/1

```c
#define N 1025

void prod_mat_vect(float a[][N], float x[N], float y[N])
{
  int i, j;

#pragma omp for
  for(i=0; i<N; i++)
    for(j=0; j<N; j++)
      y[i] = y[i] + a[i][j]*x[j];
}
```

13 nov 07 17:59 **main.c** Page 1/2

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define N 5

int main()
{
  int i;
  float *a, *b;
  FILE *f;

#pragma omp parallel
  {
#pragma omp single
    {
      a=(float *) malloc(N*sizeof(float));
      b=(float *) malloc(N*sizeof(float));
    }
#pragma omp master
    {
      if ( (f=fopen("fort.9", "r")) == NULL ) {
        perror( "fopen" );
        exit(1);
      }
      fread( a, N, sizeof(float), f );
    }
#pragma omp barrier
#pragma omp for
```

13 nov 07 17:59 **main.c** Page 2/2

```c
    for (i=0; i<N; i++)
      b[i] = 2.*a[i];
#pragma omp single nowait
    {
      free(a);
      fclose(f);
    }
  }
  for(i=0; i<N; i++)
    printf("b[%d] vaut : %f", i, b[i]);
  printf("\n");
  free(b);
  return 0;
}
```

15 nov 07 12:10 **main.c** Page 1/1

```c
#include <stdio.h>
#include <omp.h>

int main()
{
  int compteur, rang;
  compteur = 92290;
#pragma omp parallel private(rang)
  {
    rang=omp_get_thread_num();

#pragma omp atomic
    compteur++;

    printf("Rang : %d ; compteur vaut : %d\n",rang,compteur);
  }
  printf("Au total, compteur vaut : %d\n",compteur);
  return 0;
}
/*
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out
Rang : 3 ; compteur vaut : 92291
Rang : 2 ; compteur vaut : 92293
Rang : 1 ; compteur vaut : 92292
Rang : 0 ; compteur vaut : 92294
Au total, compteur vaut : 92294
*/
```

13 nov 07 18:03 **main.c** Page 1/1

```c
#include <stdio.h>

int main()
{
  int s, p;

  s = 0, p = 1;
#pragma omp parallel
  {
#pragma omp critical
    {
      s++;
      p*=2;
    }
  }
  printf("Somme et produit finaux : %d, %d\n",s,p);
  return 0;
}


/*
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out


 Somme et produit finaux : 4, 16
*/
```

13 nov 07 18:04 **main.c** Page 1/2

```c
#include <stdio.h>
#include <omp.h>

int main()
{
  int rang, nb_taches, synch=0;

#pragma omp parallel private(rang,nb_taches)
  {
    rang=omp_get_thread_num();
    nb_taches=omp_get_num_threads();

    if (rang == 0) {
      do {
#pragma omp flush(synch)
      }
      while(synch != nb_taches-1);
    }
    else {
      do {
#pragma omp flush(synch)
      }
      while(synch != rang-1);
    }
    printf("Rang : %d ; synch : %d\n",rang,synch);
    synch=rang;
#pragma omp flush(synch)
  }
  return 0;
```

13 nov 07 18:04 **main.c** Page 2/2

```c
}


/*
> xlc_r ... -qsmp=omp main.c
> export OMP_NUM_THREADS=4
> a.out


Rang : 1 ; synch : 0
Rang : 2 ; synch : 1
Rang : 3 ; synch : 2
Rang : 0 ; synch : 3
*/
```

13 nov 07 18:05 **main.c** Page 1/1

```
include <stdio.h>
#define N 9

int main()
{
  int i;
  float s, a[N];

  for(i=0; i<N; i++)
    a[i] = 92290.;

#pragma omp parallel private(s,i) shared(a)
  {
#pragma omp for lastprivate(s)
    for(i=0; i<N; i++)
      s = a[i];

    printf("s = %f ; a[8] = %f\n",s,a[N−1]);
  }
  return 0;
}
```

13 nov 07 18:06 **main.c** Page 1/1

```
#include <stdio.h>

int main()
{
  float s;

#pragma omp parallel default(none) shared(s)
  {
#pragma omp single
    {
      s=1.;
    }
    printf(s = %f\n",s);
  s=2.;
 }
 return 0;
}
```