codigobp0.pdf

PruebaAlien

Arquitectura de Computadores

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

EJERCICIO 2

(HelloOMP)

```
/* compilar con:
        gcc -O2 -fopenmp -o HelloOMP HelloOMP.c
*/
#include <stdio.h>
#include <omp.h>

int main(void){

        #pragma omp parallel
        printf("(%d):¡¡¡HOLA MUNDO!!!\n",omp_get_thread_num());
        return 0;
}
```

(HelloOMP2)

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

int main(void){

        #pragma omp parallel
        printf("(%d):¡¡¡HOLA MUNDO!!!\n",omp_get_thread_num());

        #pragma omp parallel
        printf("(%d):¡¡¡MUNDO!!!\n",omp_get_thread_num());
        return 0;
}
```

EJERCICIO 3

(SCRIPT SH)

```
#!/bin/bash
#Órdenes para el sistema de colas:
#1. Asigna al trabajo un nombre
#SBATCH --job-name=helloOMP
#2. Asignar el trabajo a una cola (partición)
#SBATCH --partition=ac
#2. Asignar el trabajo a un account
#SBATCH --account=ac

#Instrucciones del script para ejecutar código:

echo "---------------------------------65536"
./SumaVectoresC2 65536
echo " "
echo "---------------------------------131072"
./SumaVectoresC2 131072
echo " "
echo "---------------------------------262144"
./SumaVectoresC2 262144
echo " "
echo "---------------------------------524288"
./SumaVectoresC2 524288
echo " "
echo "---------------------------------1048576"
./SumaVectoresC2 1048576
echo " "
echo "---------------------------------2097152"
./SumaVectoresC2 2097152
echo " "
```

```
echo "---------------------------------4194304"

./SumaVectoresC2 4194304

echo " "

echo "---------------------------------8388608"

./SumaVectoresC2 8388608

echo " "

echo "---------------------------------16777216"

./SumaVectoresC2 16777216

echo " "

echo "---------------------------------33554432"

./SumaVectoresC2 33554432

echo " "

echo "---------------------------------67108864"

./SumaVectoresC2 67108864

echo " "
```

(SumaVectoresC2)

```
/* SumaVectoresC.c
 Suma de dos vectores: v3 = v1 + v2

 Para compilar usar (-lrt: real time library, es posible que no sea necesario usar -lrt):
        gcc  -O2  SumaVectoresC2.c -o SumaVectoresC2 -lrt
        gcc  -O2 -S SumaVectoresC2.c -lrt

 Para ejecutar use: SumaVectoresC longitud

*/

#include <stdlib.h>       // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h>        // biblioteca donde se encuentra la funci�n printf()
#include <time.h>         // biblioteca donde se encuentra la funci�n clock_gettime()

//S�lo puede estar definida una de las tres constantes VECTOR_ (s�lo uno de los ...
//tres defines siguientes puede estar descomentado):
//#define VECTOR_LOCAL        // descomentar para que los vectores sean variables ...
                              // locales (si se supera el tama�o de la pila se ...
                              // generar� el error "Violaci�n de Segmento")
#define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
                              // globales (su longitud no estar� limitada por el ...
                              // tama�o de la pila del programa)
//#define VECTOR_DYNAMIC    // descomentar para que los vectores sean variables ...
```

```
                        // din□micas (memoria reutilizable durante la ejecuci□n)

#ifdef VECTOR_GLOBAL
#define MAX 4294967295         //=2^32-1

double v1[MAX], v2[MAX], v3[MAX];
#endif
int main(int argc, char** argv){

 int i;

 struct timespec cgt1,cgt2; double ncgt; //para tiempo de ejecuci□n

 //Leer argumento de entrada (n□ de componentes del vector)
 if (argc<2){
  printf("Faltan n componentes del vector\n");
  exit(-1);
 }

 unsigned int N = atoi(argv[1]); // M□ximo N =2^32-1=4294967295 (sizeof(unsigned int) = 4 B)
 printf("Tama□o Vectores:%u (%lu B)\n",N, sizeof(unsigned int));
 #ifdef VECTOR_LOCAL
 double v1[N], v2[N], v3[N];   // Tama□o variable local en tiempo de ejecuci□n ...
                               // disponible en C a partir de C99
 #endif
 #ifdef VECTOR_GLOBAL
 if (N>MAX) N=MAX;
 #endif
 #ifdef VECTOR_DYNAMIC
 double *v1, *v2, *v3;
 v1 = (double*) malloc(N*sizeof(double));// malloc necesita el tama□o en bytes
 v2 = (double*) malloc(N*sizeof(double));
 v3 = (double*) malloc(N*sizeof(double));
 if ((v1 == NULL) || (v2 == NULL) || (v2 == NULL)) {
  printf("No hay suficiente espacio para los vectores \n");
  exit(-2);
 }
 #endif

 //Inicializar vectores
 for(i=0; i<N; i++){
  v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1;
 }

 clock_gettime(CLOCK_REALTIME,&cgt1);
 //Calcular suma de vectores
 for(i=0; i<N; i++)
  v3[i] = v1[i] + v2[i];

 clock_gettime(CLOCK_REALTIME,&cgt2);
 ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
```

```
      (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

  //Imprimir resultado de la suma y el tiempo de ejecuci�n
  if (N<10) {
  printf("Tiempo:%11.9f\t / Tama�o Vectores:%u\n",ncgt,N);
  for(i=0; i<N; i++)
    printf("/ V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
         i,i,i,v1[i],v2[i],v3[i]);
  }
  else
    printf("Tiempo:%11.9f\t / Tama�o Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f=%8.6f) / /
V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
         ncgt,N,v1[0],v2[0],v3[0],N-1,N-1,N-1,v1[N-1],v2[N-1],v3[N-1]);


  #ifdef VECTOR_DYNAMIC
  free(v1); // libera el espacio reservado para v1
  free(v2); // libera el espacio reservado para v2
  free(v3); // libera el espacio reservado para v3
  #endif
  return 0;
}
```

EJERCICIO 4

(script_helloomp.sh)

```
#!/bin/bash
#Órdenes para el sistema de colas:
#1. Asigna al trabajo un nombre
#SBATCH --job-name=helloOMP
#2. Asignar el trabajo a una cola (partición)
#SBATCH --partition=ac
#2. Asignar el trabajo a un account
#SBATCH --account=ac

#Obtener información de las variables del entorno del sistema de colas:
echo "Id. usuario del trabajo: $SLURM_JOB_USER"
echo "Id. del trabajo: $SLURM_JOBID"
echo "Nombre del trabajo especificado por usuario: $SLURM_JOB_NAME"
echo "Directorio de trabajo (en el que se ejecuta el script): $SLURM_SUBMIT_DIR"
echo "Cola: $SLURM_JOB_PARTITION"
echo "Nodo que ejecuta este trabajo:$SLURM_SUBMIT_HOST"
echo "Nº de nodos asignados al trabajo: $SLURM_JOB_NUM_NODES"
echo "Nodos asignados al trabajo: $SLURM_JOB_NODELIST"
echo "CPUs por nodo: $SLURM_JOB_CPUS_PER_NODE"
#Instrucciones del script para ejecutar código:
echo -e "\n 1. Ejecución helloOMP una vez sin cambiar nº de threads (valor por defecto):\n"
srun ./HelloOMP2
echo -e "\n 2. Ejecución helloOMP varias veces con distinto nº de threads:\n"
for ((P=12;P>0;P=P/2))
```

```
do
  export OMP_NUM_THREADS=$P
  echo -e "\n  - Para $P threads:"
  srun ./HelloOMP2
done
```

(HelloOMP2.C)

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

int main(void){

        #pragma omp parallel
        printf("(%d):¡¡¡HOLA MUNDO!!!\n",omp_get_thread_num());

        #pragma omp parallel
        printf("(%d):¡¡¡MUNDO!!!\n",omp_get_thread_num());
        return 0;
}
```

EJERCICIO 5

(script_suma_vectores.sh)

```
#!/bin/bash
#Órdenes para el sistema de colas:
#1. Asigna al trabajo un nombre
#SBATCH --job-name=helloOMP
#2. Asignar el trabajo a una cola (partición)
#SBATCH --partition=ac
#2. Asignar el trabajo a un account
#SBATCH --account=ac

#Instrucciones del script para ejecutar código:

#echo "---------------------------------65536"
#srun ./SumaVectoresC 65536
#echo " "
#echo "---------------------------------131072"
#srun ./SumaVectoresC 131072
#echo " "
#echo "---------------------------------262144"
#srun ./SumaVectoresC 262144
#echo " "
echo "---------------------------------524288"
srun ./SumaVectoresC 524288
echo " "
echo "---------------------------------1048576"
srun ./SumaVectoresC 1048576
echo " "
```

```
echo "--------------------------------2097152"
srun ./SumaVectoresC 2097152
echo " "
echo "--------------------------------4194304"
srun ./SumaVectoresC 4194304
echo " "
echo "--------------------------------8388608"
srun ./SumaVectoresC 8388608
echo " "
echo "--------------------------------16777216"
srun ./SumaVectoresC 16777216
echo " "
echo "--------------------------------33554432"
srun ./SumaVectoresC 33554432
echo " "
echo "--------------------------------67108864"
srun ./SumaVectoresC 67108864
echo " "
```

(SumaVectoresC.c)

```
/* SumaVectoresC.c
 Suma de dos vectores: v3 = v1 + v2

 Para compilar usar (-lrt: real time library, es posible que no sea necesario usar -lrt):
        gcc  -O2  SumaVectores.c -o SumaVectores -lrt
        gcc  -O2 -S SumaVectores.c -lrt

 Para ejecutar use: SumaVectoresC longitud

*/

#include <stdlib.h>       // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h>        // biblioteca donde se encuentra la funci�n printf()
#include <time.h>         // biblioteca donde se encuentra la funci�n clock_gettime()

//S�lo puede estar definida una de las tres constantes VECTOR_ (s�lo uno de los ...
//tres defines siguientes puede estar descomentado):
//#define VECTOR_LOCAL        // descomentar para que los vectores sean variables ...
                              // locales (si se supera el tama�o de la pila se ...
                              // generar� el error "Violaci�n de Segmento")
//#define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
                              // globales (su longitud no estar� limitada por el ...
                              // tama�o de la pila del programa)
#define VECTOR_DYNAMIC      // descomentar para que los vectores sean variables ...
                              // din�micas (memoria reutilizable durante la ejecuci�n)

#ifdef VECTOR_GLOBAL
#define MAX 33554432 //=2^25

double v1[MAX], v2[MAX], v3[MAX];
```

```
#endif
int main(int argc, char** argv){

 int i;

 struct timespec cgt1,cgt2; double ncgt; //para tiempo de ejecuci�n

 //Leer argumento de entrada (n� de componentes del vector)
 if (argc<2){
  printf("Faltan n componentes del vector\n");
  exit(-1);
 }

 unsigned int N = atoi(argv[1]); // M�ximo N =2^32-1=4294967295 (sizeof(unsigned int) = 4 B)
 printf("Tama�o Vectores:%u (%lu B)\n",N, sizeof(unsigned int));
 #ifdef VECTOR_LOCAL
 double v1[N], v2[N], v3[N];   // Tama�o variable local en tiempo de ejecuci�n ...
                               // disponible en C a partir de C99
 #endif
 #ifdef VECTOR_GLOBAL
 if (N>MAX) N=MAX;
 #endif
 #ifdef VECTOR_DYNAMIC
 double *v1, *v2, *v3;
 v1 = (double*) malloc(N*sizeof(double));// malloc necesita el tama�o en bytes
 v2 = (double*) malloc(N*sizeof(double));
 v3 = (double*) malloc(N*sizeof(double));
 if ((v1 == NULL) || (v2 == NULL) || (v2 == NULL)) {
  printf("No hay suficiente espacio para los vectores \n");
  exit(-2);
 }
 #endif

 //Inicializar vectores
 for(i=0; i<N; i++){
  v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1;
 }

 clock_gettime(CLOCK_REALTIME,&cgt1);
 //Calcular suma de vectores
 for(i=0; i<N; i++)
  v3[i] = v1[i] + v2[i];

 clock_gettime(CLOCK_REALTIME,&cgt2);
 ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
    (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

 //Imprimir resultado de la suma y el tiempo de ejecuci�n
 if (N<10) {
 printf("Tiempo:%11.9f\t / Tama�o Vectores:%u\n",ncgt,N);
 for(i=0; i<N; i++)
```

```
    printf("/ V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
          i,i,i,v1[i],v2[i],v3[i]);
  }
  else
    printf("Tiempo:%11.9f\t / Tama�o Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f=%8.6f) / /
V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
          ncgt,N,v1[0],v2[0],v3[0],N-1,N-1,N-1,v1[N-1],v2[N-1],v3[N-1]);


  #ifdef VECTOR_DYNAMIC
  free(v1); // libera el espacio reservado para v1
  free(v2); // libera el espacio reservado para v2
  free(v3); // libera el espacio reservado para v3
  #endif
  return 0;
}
```