test BP2
RESPUESTAS
(Ver antes el PDF de preguntas)

1. V o F. Las directivas que aceptan cláusulas son parallel, for, sections, single y workshare

t

DIRECTIVA	ejecutable
con bloque estructurado	*parallel* *sections, worksharing, single* master critical ordered
bucle	*DO/for*
simple (una sentencia)	atomic
autónoma (sin código asocidado)	barrier, flush

2. V o F. Las directivas de control de ámbito de las variables son shared, private, lastprivate, firstprivate, default y copyprivate

f

copyprivate es de copia de valores

TIPO	Cláusula	Directivas					
		parallel	DO/for	sections	single	parallel DO/for	parallel sections
Control nº threads	If (1)	X				X	X
	num_threads (1)	X				X	X
Control ámbito de las variables	shared	X				X	X
	private	X	X	X	X	X	X
	lastprivate		X	X		X	X
	firstprivate	X	X	X	X	X	X
	default (1)	X				X	X
	reduction	X	X	X		X	X
Copia de valores	copyin	X				X	X
	copyprivate				X		
Planifica. iteraciones bucle	schedule (1)		X			X	
	ordered (1)		X			X	
No espera	nowait		X	X	X		

Se han destacado en color las cláusulas que se van a comentar en este seminario

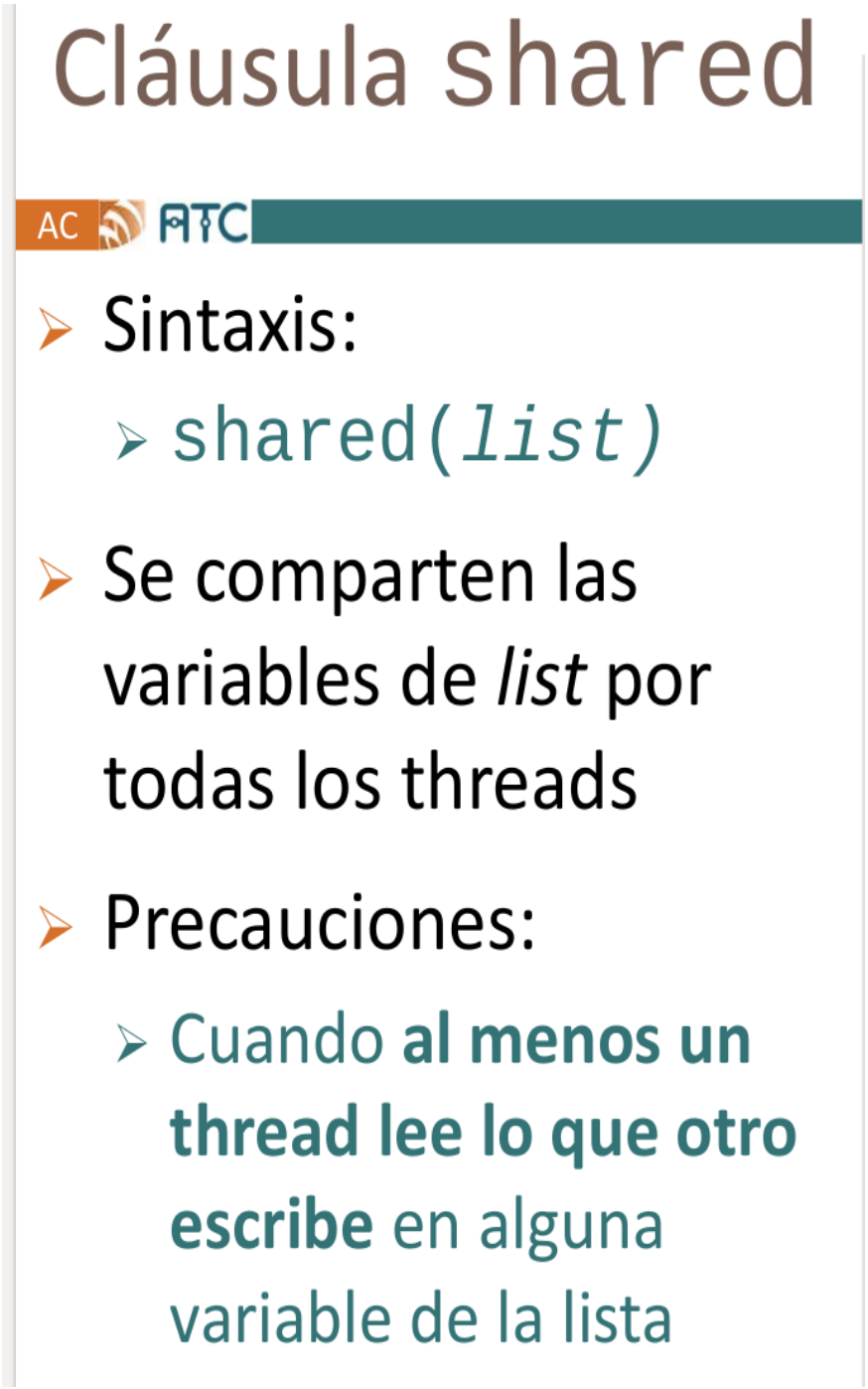
3. V o F. Por lo general, las variables declaradas dentro de una región son privadas

t

```
#pragma omp parallel
{  int sumalocal=0;
  #pragma omp for  schedule(static)
  for (i=0; i<n; i++)
  {   sumalocal += a[i];
      printf(" thread %d suma de a[%d]=%d sumalocal=%d \n“,
                  omp_get_thread_num(),i,a[i],sumalocal);
  }
  #pragma omp atomic
      suma += sumalocal;
}
```

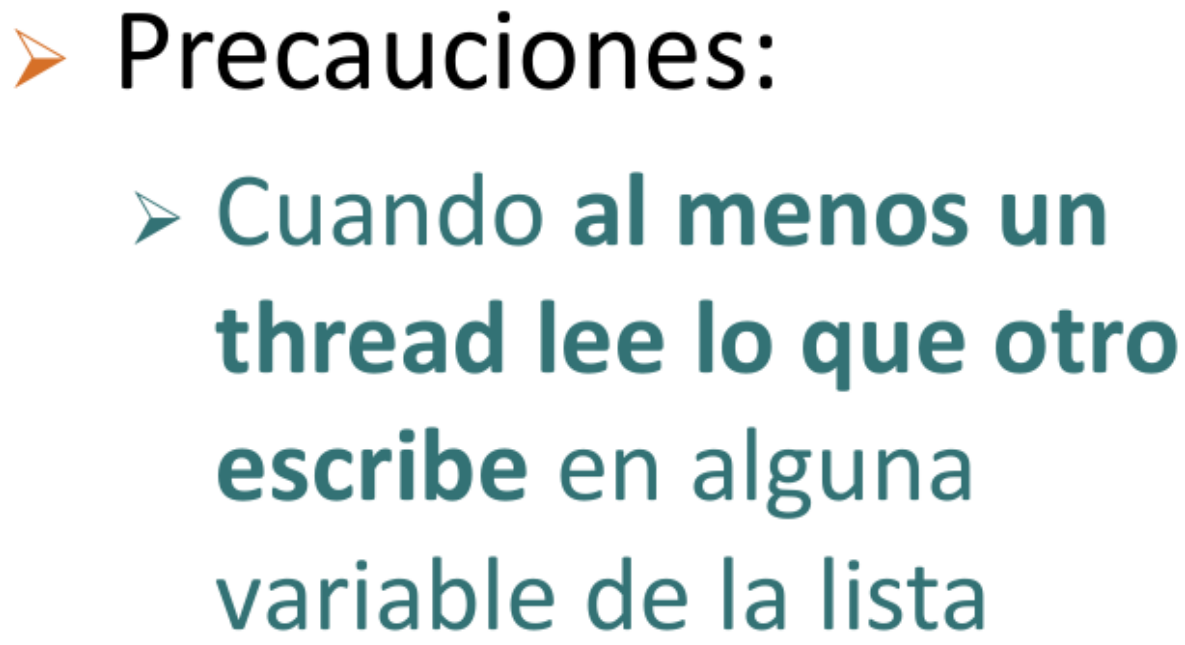
4. V o F. La cláusula shared solo sirve para arrays

f



5. V o F. OMP se ocupa de la exclusión mutua. Por lo tanto, podemos leer y escribir en una variable modificada por la cláusula shared sin problema.

f



6. V o F. Los dos siguientes fragmentos de código son equivalentes

1) #pragma omp parallel for
for (i=0;i<n;i++)
    a[i] += i;

2) #pragma omp parallel for shared(a)
for (i=0;i<n;i++)
    a[i] += i;

t

- Regla general para regiones paralelas (para variables que no se usan en cláusulas o directivas de ámbito):
  - Las variables declaradas fuera de una región y las **dinámicas** son compartidas por las threads de la región
  - Las variables declaradas dentro son privadas

7. V o F. Para el siguiente código, daría error si olvidamos compilar con -fopenmp

```
#include <stdio.h>
#ifdef _OPENMP
 #include <omp.h>
#endif

main()
{
  int i, n = 7;
  int a[n];

  for (i=0; i<n; i++)
    a[i] = i+1;

  #pragma omp parallel for shared(a)
  for (i=0; i<n; i++)    a[i] += i;

  printf("Después de parallel for:\n");

  for (i=0; i<n; i++)
    printf("a[%d] = %d\n",i,a[i]);
}
```

f

La compilación condicional lo evita

8. V o F. La ejecución del siguiente código es:
x
x
x
x

OMP_NUM_THREADS=2
a=’x’;
puts(a);
#pragma omp prarallel private(a)
    {puts(a)}
puts(a);

f

- `private(list)`
- Precauciones:
  - El valor de **entrada** y de **salida** está **indefinido** aunque la variable esté declarada fuera de la construcción

9. V o F. A partir del siguiente código

```
OMP_NUM_THREADS=2
a=’x’;
puts(a);
#pragma omp prarallel private(a)
    {puts(a)}
puts(a);
```

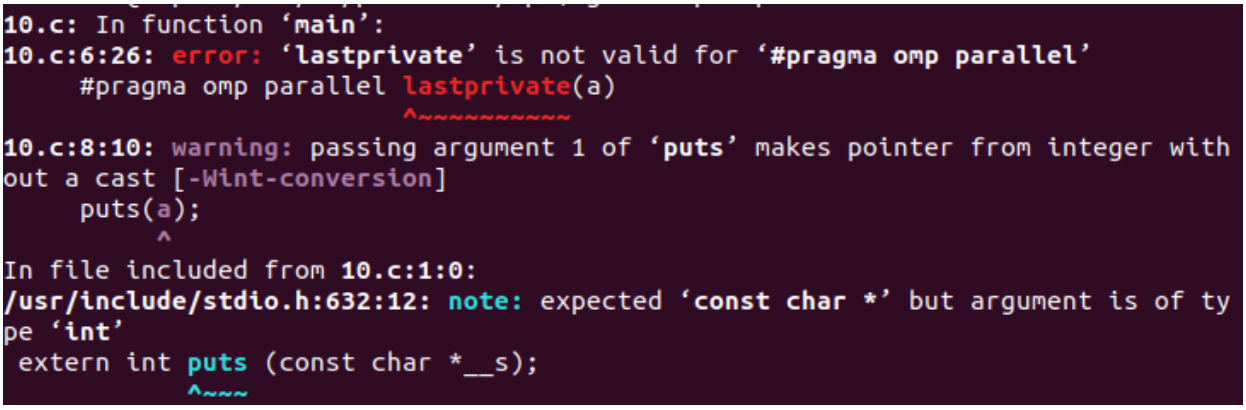
Podemos estar seguros de que al menos la segunda letra en salir es x

f

Solo la primera

10. V o F. Se puede hacer lastprivate con una directiva parallel

f



11. Cual es la salida por pantalla

code

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

int main () {
    int a = 3;
    #pragma omp parallel for lastprivate(a)
    { a = 4; }
    printf("%d",a);
    return 0;
}
```

4

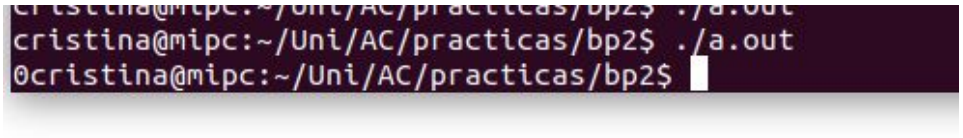
```
cristina@mipc:~/Uni/AC/practicas/bp2$ gcc -fopenmp 10.c
cristina@mipc:~/Uni/AC/practicas/bp2$ ./a.out
4cristina@mipc:~/Uni/AC/practicas/bp2$
cristina@mipc:~/Uni/AC/practicas/bp2$
```

12. Escribir el resultado por pantalla de lo siguiente

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

int main () {
    int a = 3;
    #pragma omp parallel for lastprivate(a)
    for (int i=0; i<1; i++)
    { printf("%d",a);  }
    return 0;
}
```

0



13. V o F. Siempre que se use default, el programador tienen que especificar el alcance de todas las variables que se vayan a utilizar

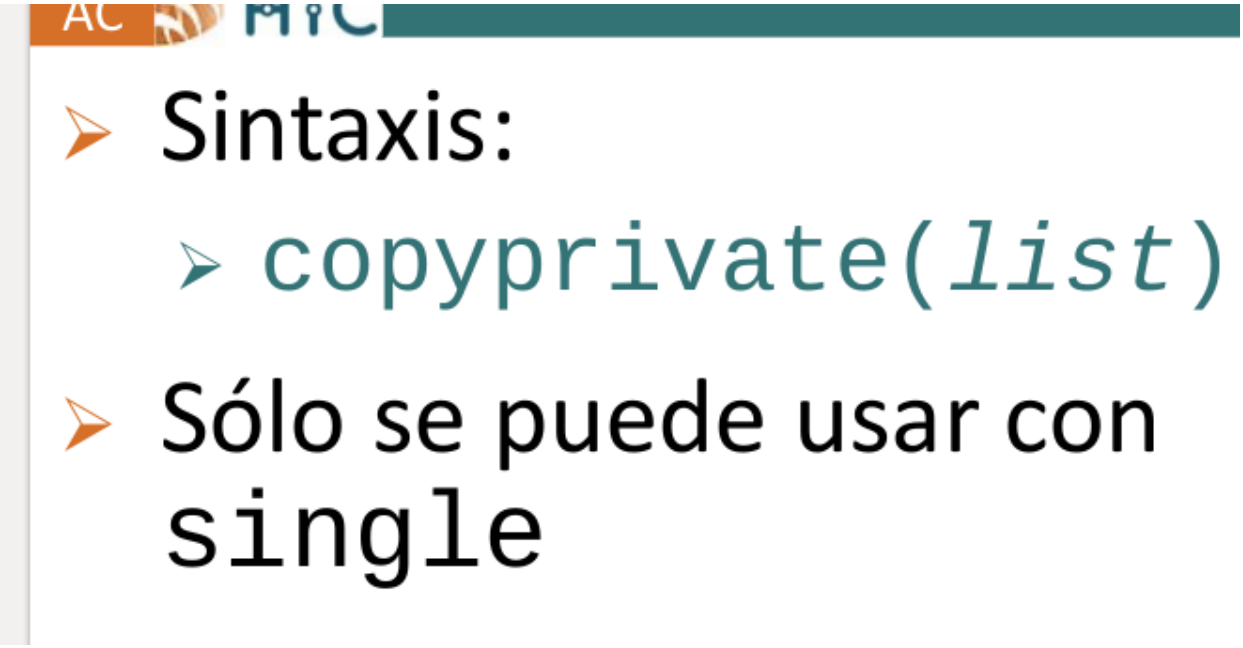
f

```
default(none|shared)
```

- Con none el programador debe especificar el alcance de todas las variables usadas en la construcción
  - excepción variables `threadprivate` y los índices en Directivas con `for`

14. V o F. La directiva copyprivate solo se puede usar con single

v



15. V o F. Utilizar copyprivate es útil cuando se quiere hacer una lectura de un valor por teclado y llenar un vector con ella

v

```
#pragma omp parallel
{  int a;
   #pragma omp single copyprivate(a)
   {
     printf("\nIntroduce valor de inicialización a: ");
     scanf("%d", &a );
     printf("\nSingle ejecutada por el thread %d\n",
          omp_get_thread_num());
   }
   #pragma omp for
   for (i=0; i<n; i++)  b[i] = a;
}
```

16. V o F. Se puede hacer dos bucles for aninados con parallel, de la siguiente forma

f

```
[c1estudiante24@atcgrid ejer8]$ gcc -fopenmp pmv-secuencial.c -O2
pmv-secuencial.c: En la función ‘main’:
pmv-secuencial.c:25:11: error: la región de trabajo compartido puede no estar bien anidada dentro de la r
egión de trabajo compartido, ‘critical’, ‘ordered’, ‘master’, ‘task’ explícita o región ‘taskloop’
   25 |    #pragma omp for
      |           ^~~
```

17. Función de OMP que calcula el tiempo de un trozo de código

omp_get_wtime()

```
double start;
double end;
start = omp_get_wtime();
... work to be timed ...
end = omp_get_wtime();
printf("Work took %f seconds\n", end - start);
```