Array cast

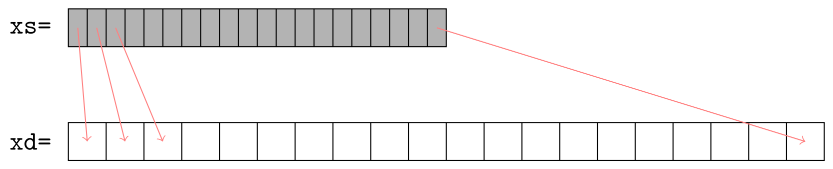
1 Description of the problem

Let's assume that we have an array `xs` of size `n` that contains single-precision (`float`, 32-bit) real values. We want to **cast** (i.e., convert) the coefficients of this array to double-precision (`double`, 64-bit) format. This can be done in two different ways.

Out-of-place: in this case we have a second array `xd` of size `n` of type `double`. The cast simply consists in copying `xs` into `xd` while converting the coefficients one by one as in the pseudo-code below:

```
for(i=0; i<n; i++)
   xd[i] = (double)xs[i];
```

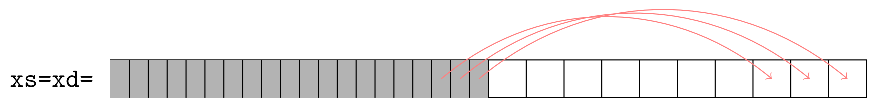
This is illustrated in the figure below



In-place: in this case, a second array of type `double` is not available. Rather, we assume that `xs` is twice as large (i.e., `n`×64-bit). In the beginning it contains `n` coefficients of type `float` which means that half of this array is empty. In the end, it will contain all the original coefficients cast to `double`. This is achieved by the pseudo-code below:

```
/* xd and xs point to the same memory */
double *xd = (double*)xs;
for(i=n-1; i>=0; i--)
   xd[i] = (double)xs[i];
```

Note that, in the code above, `xs` and `xd` are pointers of type `float` and `double`, respectively, but they both point to the same array. Note, also, that the loop proceeds backwards to avoid overwriting values that have not been cast yet. This is illustrated in the figure below:



This exercise is about parallelizing both the above versions of the array cast operation.

2 Package content

In the `inplace_cast` directory you will find the following files:

- `main.c`: this file contains the main program which first initializes the `float` data and, then, calls the `out_of_place_cast` and `in_place_cast` routines. For both of them, the correctness of the result is checked.**Only this file has to be modified for this exercise**.
- `aux.c, aux.h`: these two files contain auxiliary routines and **must not be modified**.

The code can be compiled with the `make` command: just type `make` inside the `in_place_cast` directory; this will generate a `main` program that can be run like this:

```
$ ./main n
```

where `n` is the size of the array to be cast.

3 Assignment

- At the beginning, the `out_of_place_cast` and `in_place_cast` routines contain sequential code (pretty-much the same as the pseudo-code in the previous section). Modify this routines in order to parallelize them.
- Report the execution times for the implemented parallel versions with 1, 2 and 4 threads. What speedup could you achieve? analyze and comment on your results. Report your answer in the form of comments at the bottom of the `main.c` file.

Advice

- For the in-place cast, you can make the assumption that the size `n` is a power of 2. This should make things much easier. To launch the code using a power of two you can use the following command:

  ```
  $ ./main $((2**10))
  ```

  In this case the size of the array will be 2^{10}.