PH502: Scientific Programming Concepts

Irish Centre for High End Computing (ICHEC)

September 23, 2020

Overview

- So far we have used fixed length arrays.
- However it is not very convenient to have several versions of a program for different problem sizes.
- You could declare an array to be bigger than any problem you might conceive. But as memory sizes increased this was always a losing battle.
- The answer is to be able to declare array sizes at run-time and not compile time.
- This process is called dynamic array allocation.

Allocating Memory

- Sometimes, you won't know beforehand how much space you need in an array.
- The memory for the array content needs to be allocated dynamically.
- Load `<stdlib.h>`, meet `malloc()`, and say hello to memory leaks.
- In FORTRAN memory can be allocated dynamically. Any variable must have the ALLOCATABLE attribute.

Allocating Memory

- malloc() takes an integer as argument, and returns a pointer to a block of memory that it reserved.
- The size of that memory block is as many bytes as the integer passed as argument.
- For convenience, the sizeof() function provides the size in bytes of a given type.

```
char *p, *q;
int  *r;

p = (char *) malloc(10);                    // 10 bytes
q = (char *) malloc(10 * sizeof(char)); // 1 char==1 byte==10 bytes
r = (int *) malloc(10 * sizeof(int));   // 1 int==4 bytes==40 bytes
r[8] == *(r+8);                  // true (both content of the
                                 // 9th cell of the r "array")
```

Allocating Memory

- Sometimes, `malloc()` is unable to allocate the requested memory.
- In such a case it returns `NULL`.

```c
int *p;

p = (int *) malloc(4000 * sizeof(int));
if (p == NULL){
  printf("Failed memory allocation!\n");
  exit(); // or recover in a nicer way
}
```

Allocating Memory

FORTRAN

- In FORTRAN ALLOCATE() does the job of malloc(). The amount of memory is determined by the array size and the variable type.
- The shape of the array must be determined at declaration.
- If there is a problem with allocation then the program will terminate unless "stat" is used.

```
integer (kind=4) :: ierr
integer (kind=4), allocatable :: arr1d(:), arr2d(:,:)

allocate(arr1d(10),arr2d(10,5),stat=ierr)
if (ierr .ne. 0) write(6,*) ' Allocation error'
```