PH502: Scientific Programming Concepts

Irish Centre for High End Computing (ICHEC)

September 23, 2020

Overview

- This lecture we will explain how to pass information out of a C function via its arguments.

Passing by Reference

ICHEC
Irish Centre for High-End Computing

- Previously we have said that arguments in C were passed by value.
- That means that a the $r-value$ of the argument is copied to that of the dummy argument.
- Because the dummy argument is destroyed when the function completes, its value is not passed back to the calling function.
- This is a distinct difference to FORTRAN.
- There is a another way to pass arguments, by reference.

Passing by Reference

- Passing by reference means that the memory address of the argument is passed to the dummy argument.
- This of course means that the dummy argument needs to be a pointer.
- The dummy argument is still a different variable to the argument from the calling function.
- It is also destroyed when the function completes.
- However because it points to a valid memory address reserved for the calling function, any changes in the $r - value$ in that address can be accessed by the calling function.

Passing Arguments Out

C

- To pass arguments out of a function they must be passed by $l-value$.
- Below is an example where the $r-value$ of two variables are swapped. The $l-values$ of a and b are passed to "swap".

```c
#include <stdio.h>
void swap(int *px, int *py);
int main(void) {
  int a,b;
  a = 1; b = 2;
  swap(&a,&b);
  printf("a=%d and b=%d with l-values %p %p\n",a,b,&a,&b);
  return 0;
}
void swap(int *px,int *py) {
   int temp;

   temp = *px; *px = *py;
   *py = temp;
   printf(" In swap %p, %p\n",px,py);
/* return incorrect as void fn */
}
```

Passing Arguments

Fortran vs C

Fortran:

- rules for arguments are complicated and compiler dependent
- scope is controlled by: intent(in), intent(out), intent(inout)

C:

- non-array arguments
  - are passed by $r-value$ by default
  - use pointers to pass by $l-value$ and out of function
- arrays are passed by $l-value$

int scanf(const char* format, ...)

- Your basic formatted input function. Uses almost the same format as `printf()`.
- First argument describes how the input is going to be parsed. If static content is present, it is expected to be present in the input read.
- The next arguments are pointers to memory blocks of the correct type/size. Typically, to store some input into an already declared variable, one uses the & operator to get its address.

```
int i,j,k;
long l;
float f;
double g;
char z;
scanf("%d", &i); // reads an integer
scanf("(%d,%d)", &j, &k); // reads two integers between
     // brackets with a coma separating them
scanf("%c", &z); // reads a single character
scanf("%ld,%f,%lf", &l, &f, &g); // reads a long,
     // a float, a double, separated by comas
```

- Format codes are the same as shown earlier for `printf()` (almost).

Fortran Read

- Like writing the "read" statement is also associated with a logical unit. You can read and write to the same unit.
- Reading user input is done on unit 5.
- The "read" statement is similar to write, format follows the same notation.

```
read(UNIT=n,FMT=format,ADVANCE=s,IOSTAT=ivar,ERR=label1,&
END=label2) var1,var2,etc
```