Pointer to Pointer

- Pointer to pointer stores address of some pointer variable.
- Level of indirection: Number of dereference operator to retrieve value.

```
int main() {
  double a = 1.2;
  double *p = &a;
  double **pp = &p;
  printf("%lf\n", a);
  printf("%lf\n", *p);
  printf("%lf\n", **pp);
  return 0;
}
```

Pointer Arithmetic

- Scale factor plays significant role in pointer arithmetic.
- n locations ahead from current location
  - ptr + n = ptr + n * scale factor of ptr
- n locations behind from current location
  - ptr - n = ptr - n * scale factor of ptr
- number of locations in between
  - ptr1 – ptr2 = (ptr1 – ptr2) / scale factor of ptr1
- When pointer is incremented or decremented by 1, it changes by the scale factor.
- When integer ‘n’ is added or subtracted from a pointer, it changes by n * scale factor.
- Multiplication or division of any integer with pointer is not allowed.
- Addition, multiplication and division of two pointers is not allowed.
- Subtraction of two pointers gives number of locations in between. It is useful in arrays.

Array

- Array is collection of similar data elements in contiguous memory locations.
- Elements of array share the same name i.e. name of the array.
- They are identified by unique index/subscript. Index range from 0 to n-1.
- Array indexing starts from 0.
- Checking array bounds is responsibility of programmer (not of compiler).
- Size of array is fixed (it cannot be grow/shrink at runtime).

```
int main() {
     int i, arr[5] = {11, 22, 33, 44, 55};
     for(i=0; i<5; i++)
           printf("%d\n", arr[i]);
     return 0;
}
```

- If array is initialized partially at its point of declaration rest of elements are initialized to zero.
- If array is initialized at its point of declaration, giving array size is optional. It will be inferred from number of elements in initializer list.
- The array name is treated as address of 0th element in any runtime expression.
- Pointer to array is pointer to 0th element of the array.

Makefile

- make utility uses makefile to update one or more target files
- names should be either GNUMakefile, makefile or Makefile
- makefile is set of rules which are used to compile your program
- every rule of makefile is made up of two lines
    - first line - Dependancy line - list of all dependent files
    - second line - Rule/Receipe line - command to create target file
- Rule syntax:

```
<Target>: [List of Dependancies]
    <command to create target>
```

- Target - final outcome of the rule
- second line of rule always starts with one tab
- in makefile comment starts with '#'
- rule to create final target should be the first rule in makefile
- if different name is given to makefile, use -f option of make to specify it

```
make -f <file name>
```

- Variables
    - in makefile variables are known as make variables
    - to create make variable

      ```
      <variable name>=<variable value>
      ```

    - to use make variable

      ```
      $(<variable name>)
      ```

    - generally make variables are written in uppercase