Null Pointer and Void Pointer

Dr. Rahul Das Gupta

Null Pointer

A null pointer is a pointer which points nothing.

Some uses of the null pointer are:

a) To initialize a pointer variable when that pointer variable isn't assigned any valid memory address yet.

b) To pass a null pointer to a function argument when we don't want to pass any valid memory address.

c) To check for null pointer before accessing any pointer variable. So that, we can perform error handling in pointer related code e.g. dereference pointer variable only if it's not NULL.

Void Pointer

A void pointer is a pointer that has no associated data type with it. A void pointer can hold address of any type and can be typcasted to any type.

Advantages of void pointers:

1) malloc() and **calloc()** return void * type and this allows these functions to be used to allocate memory of any data type (just because of void *)

```
int main(void)
{
   /* Note that malloc() returns void * which can be
   typecasted to any type like int *, char *, … */
   int * x = (int *) malloc(sizeof(int) * n);
   …
   …
   …
}
```

2) **void pointers in C are used to implement generic functions in C.**

Some Interesting Facts:

1) void pointers cannot be dereferenced. For example the following program doesn't compile.

```
#include<stdio.h>
int main()
{
   int a = 10;
   void *ptr = &a;
   printf("%d", *ptr);
   return 0;
}
```

Output:

```
Compiler Error: 'void*' is not a pointer-to-object type
```

The following program compiles and runs fine.

```
#include<stdio.h>
int main()
{
   int a = 10;
   void *ptr = &a;
   printf("%d", *(int *)ptr);
   return 0;
}
```

Output:

```
10
```

2) The C standard doesn't allow pointer arithmetic with void pointers. However, in GNU C it is allowed by considering the size of void is 1. For example the following program compiles and runs fine in gcc.

```
#include<stdio.h>
int main()
{
   int a[2] = {1, 2};
   void *ptr = &a;
   ptr = ptr + sizeof(int);
   printf("%d", *(int *)ptr);
   return 0;
}
```

Output:

```
2
```