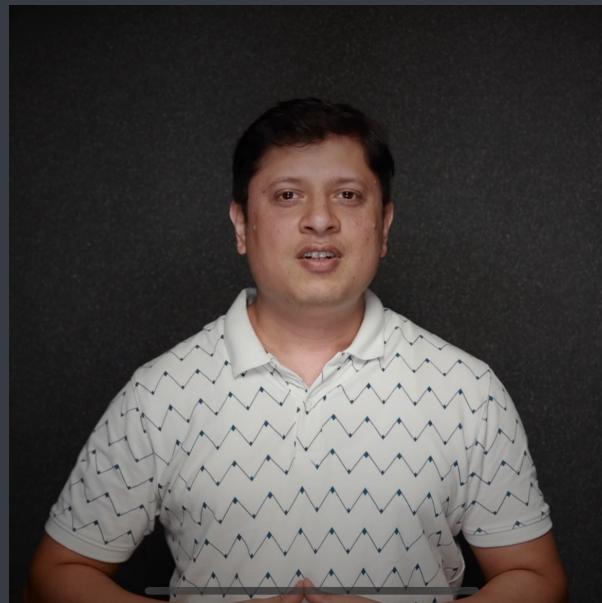


C Language

DMA



Saurabh Shukla (MySirG)

Agenda

- ① SMA vs DMA
- ② malloc()
- ③ Type Casting
- ④ calloc()
- ⑤ Memory Leak
- ⑥ free()
- ⑦ realloc()

SMA

Static Memory Allocation

```
int a;
float b;
char *p;
double d1;
struct Employee e1;
int A[5];
```

नाम छोटा है

Compile time

DMA

Dynamic Memory Allocation

```
malloc()
calloc()
free()
```

कोई नाम नहीं होता

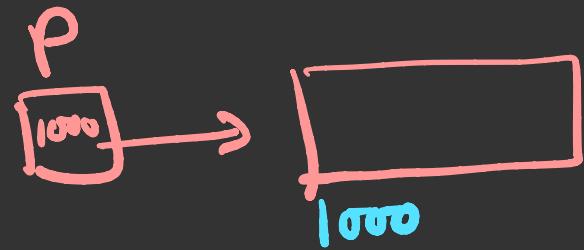
Runtime

malloc()

float *p;

p = (float*) malloc(4);

void* malloc(int s)
{



4 bytes

return address

}

int *p; Type casting

p = (int *)malloc(4);

Calloc()

calloc (no. of var , sizeof var)

```
int *P;
```

```
P = (int *)calloc ( 5, 4 );
```



$*(P + i)$

$P[i]$

Malloc vs Calloc

① one argument

② Garbage value

③ Single block

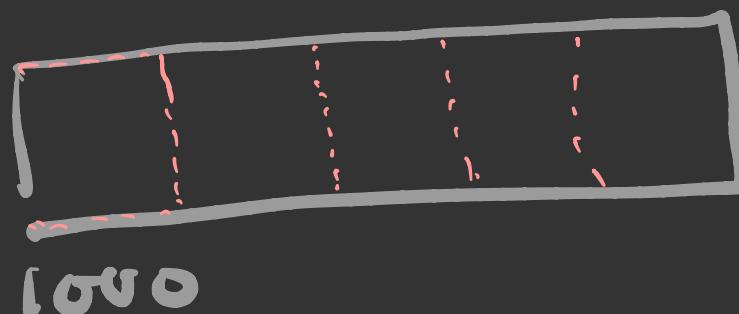
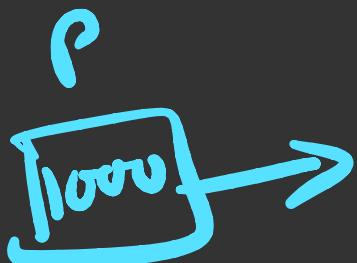
① two arguments

② 0 zero

③ Array of blocks

int *P;

P=(int *) malloc(20)



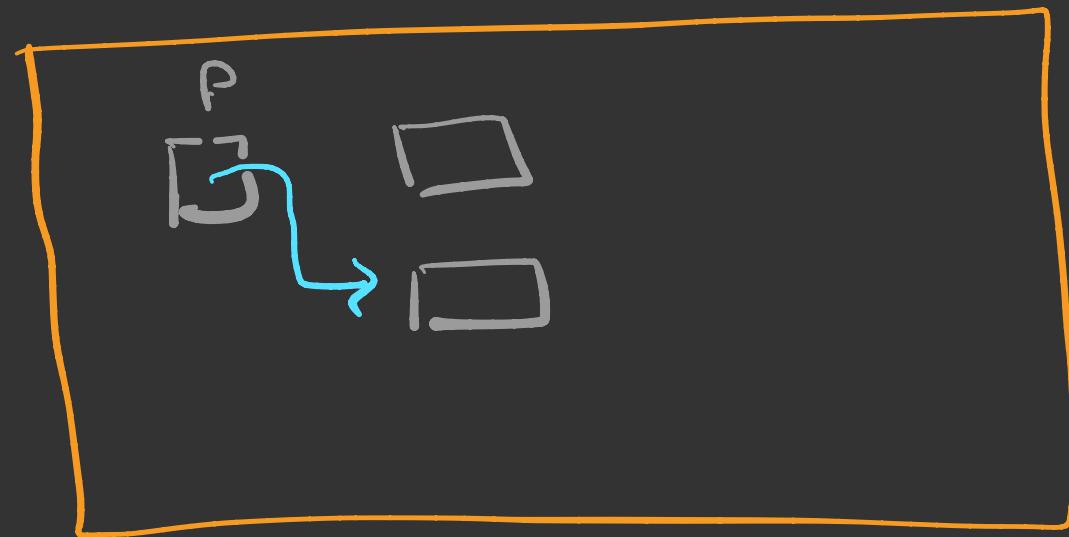
```
int *p;
```

Memory Leak

```
p = (int*)malloc(4);
```

==

```
p = (int*) malloc(4);
```



Total = Consumed + free

free()

free function is used only to
release memory of DMA variables.

free()

int x;
free(&x);
x

Two Options

- ① Return address
of DMA variable

```
int *q;
```

```
q = f1();
```

free()

```
int* f1() {  
    int *p;  
    p = (int*) malloc(4);  
    return p;  
}
```

The diagram illustrates a memory leak. The variable `p` is a pointer to dynamically allocated memory (a box). The variable `q` is another pointer that also points to the same heap-allocated memory (another box). Both pointers are underlined in red, showing they reference the same heap-allocated block.



Two Options

- ② free memory
of DMA variable

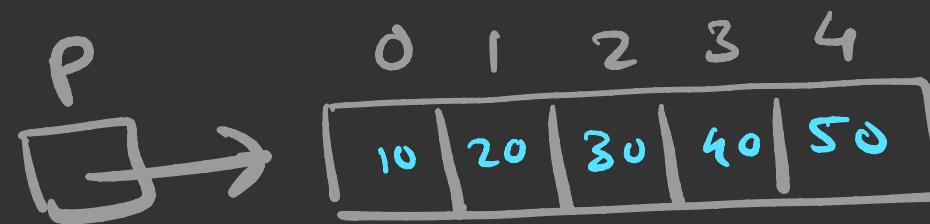
```
f1()  
{  
    int *p;  
    p=(int*)malloc(4);  
    ...  
    ...  
    ...  
    free(p);  
}
```

realloc()

realloc(pointer, newsize)

int *P;

P = (int *) malloc(20);



P = realloc(P, 40);

