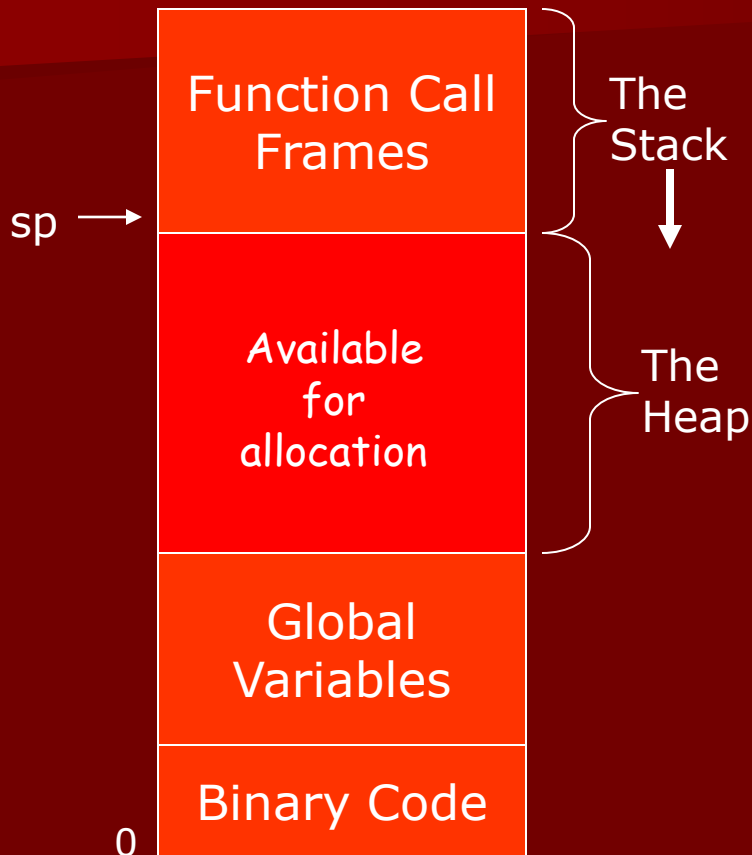


# Memory Management in C

Personal Software Engineering

# Memory Organization



- The call stack grows from the top of memory down
- Code is at the bottom of memory.
- Global data follows the code.
- What's left – the "heap" - is available for allocation.

# Allocating Memory

```
void *malloc( unsigned nbytes ) ;
```

- Allocates 'nbytes' of memory in the heap.
- Guaranteed not to overlap other allocated memory.
- Returns pointer to the first byte (or NULL if the heap is full).
- Similar to constructor in Java – allocates space.
- Space allocated uninitialized (random garbage).

```
void free( void *ptr ) ;
```

- Frees the memory assigned to ptr.
- The space must have been allocated by malloc.
- *No garbage collection in C (or C++).*
- Can slowly consume memory if not careful

# How Much Space Is Needed? - 1

`sizeof (type)` – gives the size of a type in bytes.

## Allocation Examples

```
int *ip ;
```



The diagram illustrates the memory layout for the code above. A red trapezoidal shape labeled 'ip' is positioned above a red rectangular box labeled '????'. A vertical line connects the bottom center of the 'ip' shape to the top center of the '????' box, representing a pointer relationship.

# How Much Space Is Needed? - 1

`sizeof (type)` – gives the size of a type in bytes.

## Allocation Examples

```
int *ip ;
```



```
ip = (int *) malloc( sizeof(int) ) ;
```



# How Much Space Is Needed? - 1

`sizeof (type)` – gives the size of a type in bytes.


## Allocation Examples

```
int *ip ;
```




A diagram showing a pointer variable 'ip' pointing to a memory location containing '????'.

```
ip = (int *) malloc( sizeof(int) ) ;
```



A diagram showing a pointer variable 'ip' pointing to a dynamically allocated memory block containing '????'.

```
*ip = 1234 ;
```



A diagram showing a pointer variable 'ip' pointing to a memory location containing the value '1234'.

# How Much Space Is Needed? - 1


`sizeof (type)` – gives the size of a type in bytes.

## Allocation Examples


```
int *ip ;
```

A diagram showing a pointer variable 'ip' represented by a blue trapezoid. Below it is a light blue rectangular box containing the text '????', representing the memory location for the pointer.

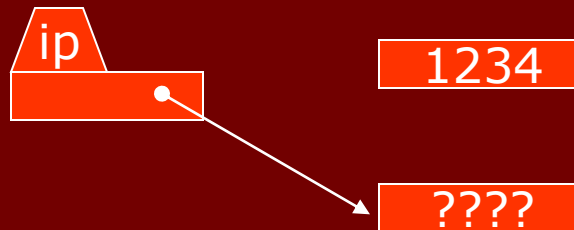
```
ip = (int *) malloc( sizeof(int) ) ;
```

A diagram showing the pointer variable 'ip' (blue trapezoid) pointing to a light blue rectangular box containing '????'. An arrow points from a dot inside the 'ip' box to the '????' box.

```
*ip = 1234 ;
```

A diagram showing the pointer variable 'ip' (blue trapezoid) pointing to a light blue rectangular box containing the value '1234'. An arrow points from a dot inside the 'ip' box to the '1234' box.

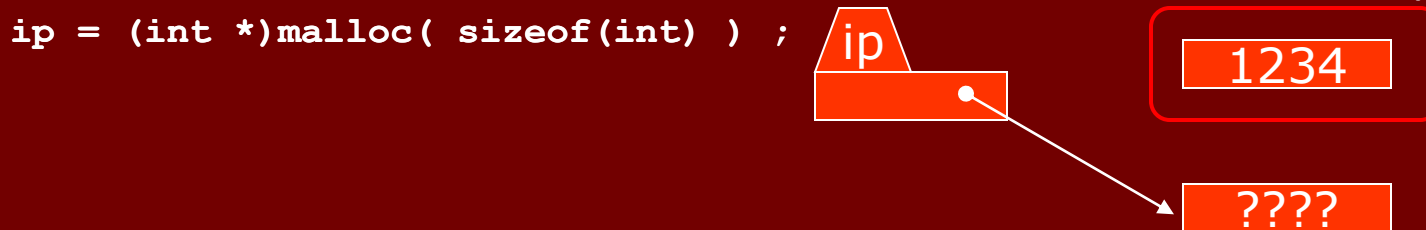
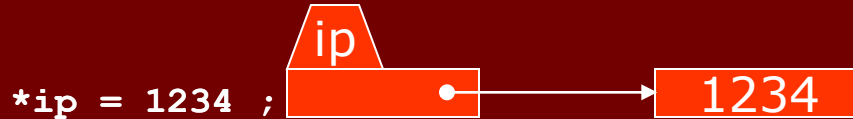
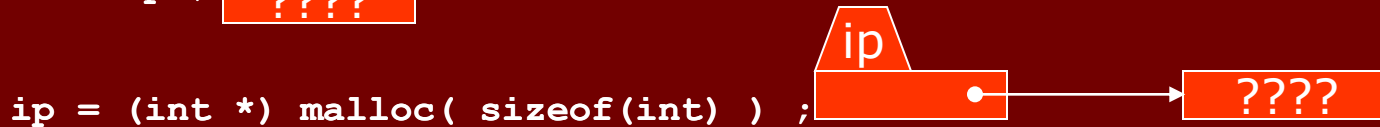
```
ip = (int *)malloc( sizeof(int) ) ;
```

A diagram showing the pointer variable 'ip' (blue trapezoid) pointing to a light blue rectangular box containing '????'. An arrow points from a dot inside the 'ip' box to the '????' box. To the right of the 'ip' box is another light blue rectangular box containing the value '1234'.

# How Much Space Is Needed? - 1

`sizeof (type)` – gives the size of a type in bytes.

## Allocation Examples





# How Much Space Is Needed? - 1

`sizeof (type)` – gives the size of a type in bytes.


## Allocation Examples

```
int *ip ;
```




A diagram showing a pointer variable 'ip' pointing to a memory location containing '????'.

```
ip = (int *) malloc( sizeof(int) ) ;
```



A diagram showing a pointer variable 'ip' pointing to a dynamically allocated memory block containing '????'.

```
*ip = 1234 ;
```



A diagram showing a pointer variable 'ip' pointing to a memory location containing the value 1234.

```
free(ip) ;
```



A diagram showing a pointer variable 'ip' pointing to a memory location containing '????'. A memory location containing the value 1234 is crossed out with a red 'X', indicating it has been freed.

# How Much Space Is Needed? - 1

`sizeof (type)` – gives the size of a type in bytes.


## Allocation Examples

```
int *ip ;
```




A diagram showing a pointer variable 'ip' represented by a blue trapezoid. Below it is a light blue box containing '????', representing the memory address stored in the pointer.

```
ip = (int *) malloc( sizeof(int) ) ;
```



A diagram showing the pointer variable 'ip' (blue trapezoid) pointing to a light blue box containing '????'. An arrow points from a dot inside the 'ip' box to the '????' box, representing the allocation of memory.

```
*ip = 1234 ;
```



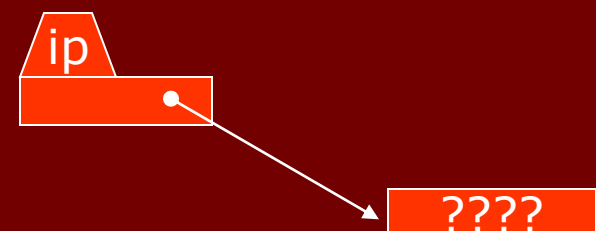
A diagram showing the pointer variable 'ip' (blue trapezoid) pointing to a light blue box containing the value '1234'. An arrow points from a dot inside the 'ip' box to the '1234' box.

```
free(ip) ;
```



A diagram showing the pointer variable 'ip' (blue trapezoid) pointing to a light blue box containing '1234'. A large red 'X' is drawn over the '1234' box, indicating that the memory has been freed.

```
ip = (int *) malloc( sizeof(int) ) ;
```



A diagram showing the pointer variable 'ip' (blue trapezoid) pointing to a light blue box containing '????'. An arrow points from a dot inside the 'ip' box to the '????' box, representing the allocation of memory.

# How Much Space Is Needed? - 2

```
char *make_copy( char *orig ) {  
    char *copy = (char *) malloc( sizeof(char) * strlen(orig) + 1 ) ;  
    (void) strcpy( copy, orig ) ;  
    return copy ;  
}
```

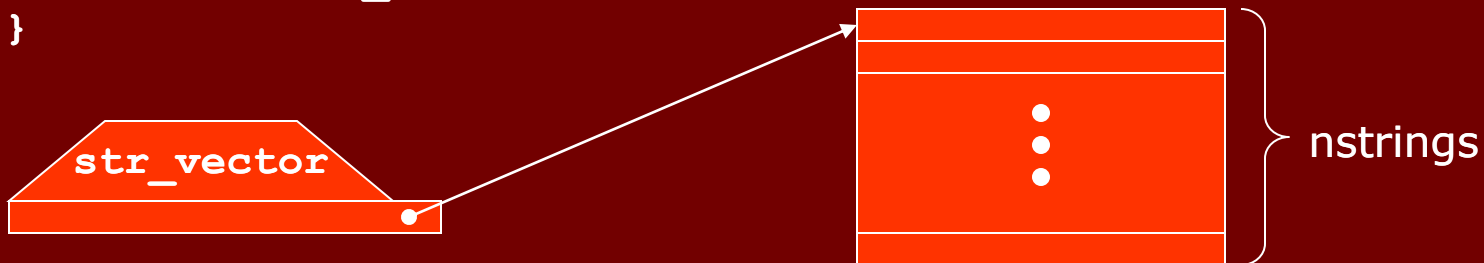
```
char orig[4] ;
```

'J'	'o'	'e'	'\0'
-----	-----	-----	------



'J'	'o'	'e'	'\0'
-----	-----	-----	------

```
char **create_string_vector( int nstrings ) {  
    char **str_vector ;  
    str_vector = (char **) malloc( nstrings * sizeof(char *) ) ;  
    return str_vector ;  
}
```



# Linked Lists

- Structures with values and link (pointer) to next – and possibly previous - structure.
- Example: List of strings:

```
typedef struct node {  
    char *string ;  
    struct node *next ;  
} node ;  
  
node *top ;
```

