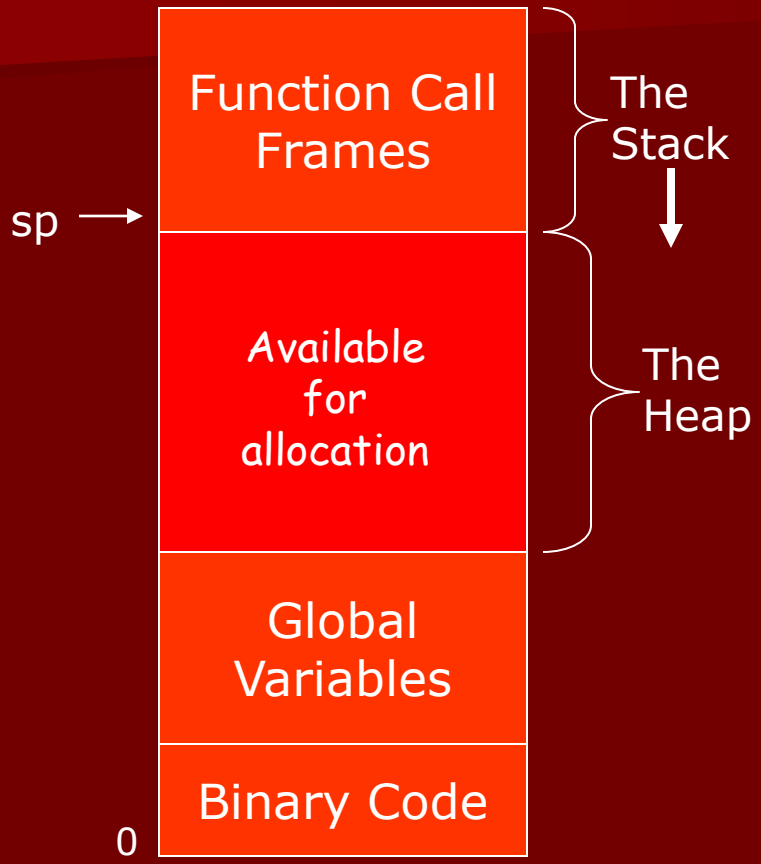
Memory Management in C

4010-350
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 ) ;
```

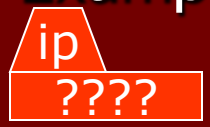
- Frees the memory assigned to ptr.
- The space must have been allocated by malloc.
- *No garbage collection in C (or C++).*
- Can slowly consume memory if not careful

How Much Space Is Needed? - 1

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

Allocation Examples

```
int *ip ;
```



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 ;
```


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

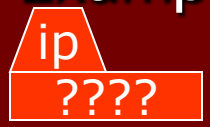

```
*ip = 1234 ;
```

How Much Space Is Needed? - 1

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

Allocation Examples

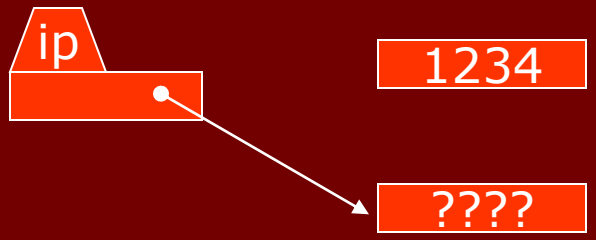
```
int *ip ;
```




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


```
*ip = 1234 ;
```


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

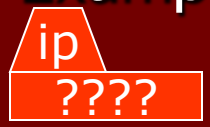


How Much Space Is Needed? - 1

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

Allocation Examples

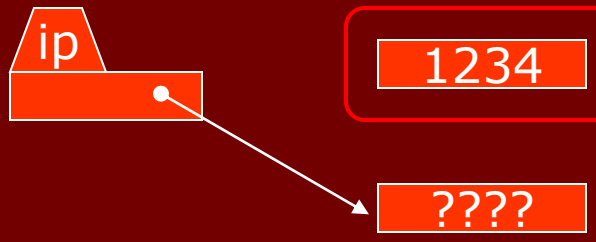
```
int *ip ;
```




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


```
*ip = 1234 ;
```


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



How Much Space Is Needed? - 1

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

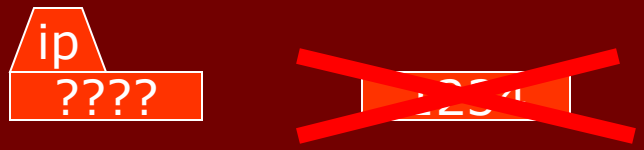
Allocation Examples

```
int *ip ;
```


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


```
*ip = 1234 ;
```

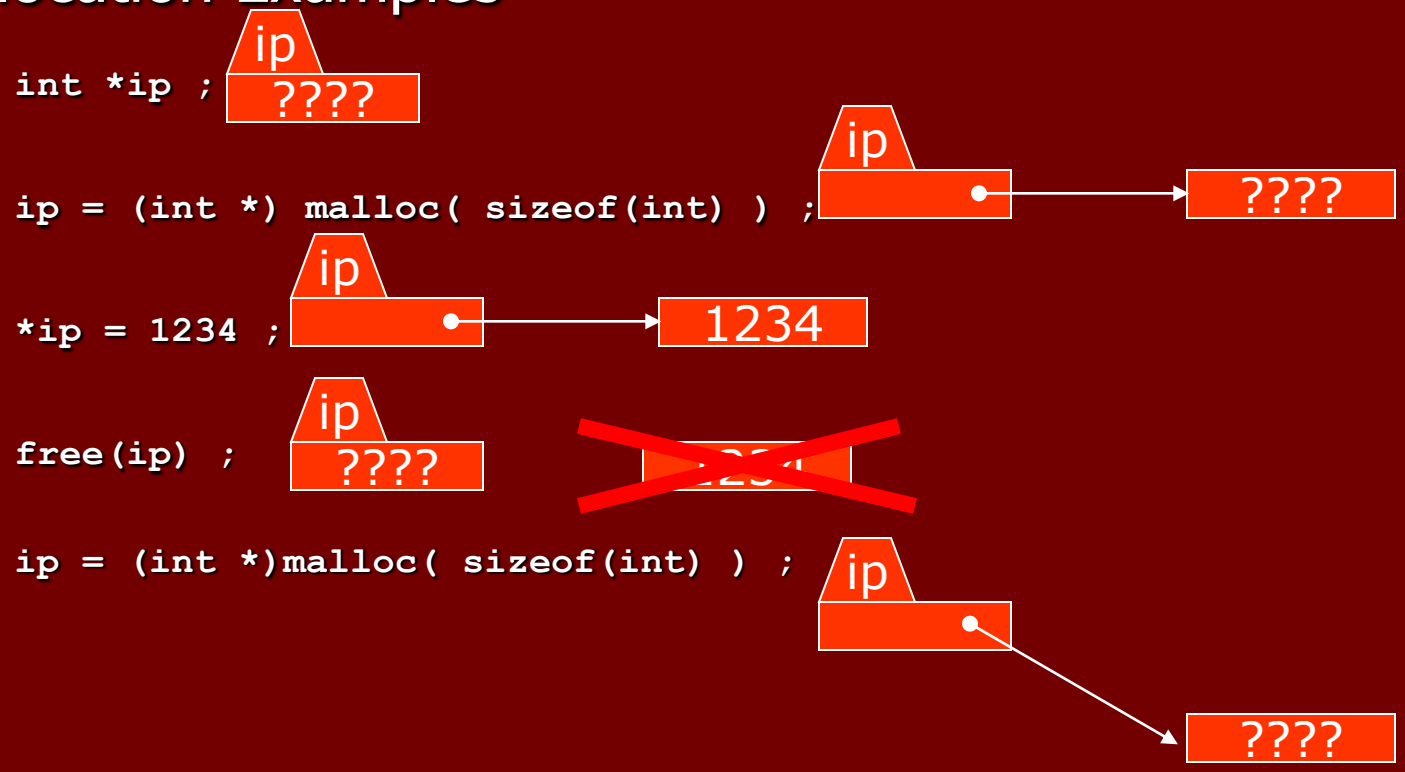

```
free(ip) ;
```



How Much Space Is Needed? - 1

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

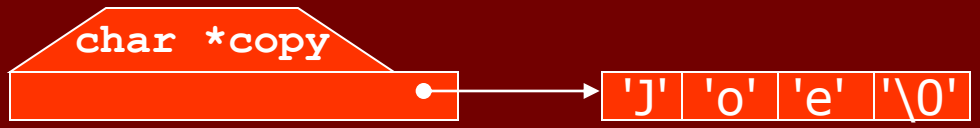
Allocation Examples



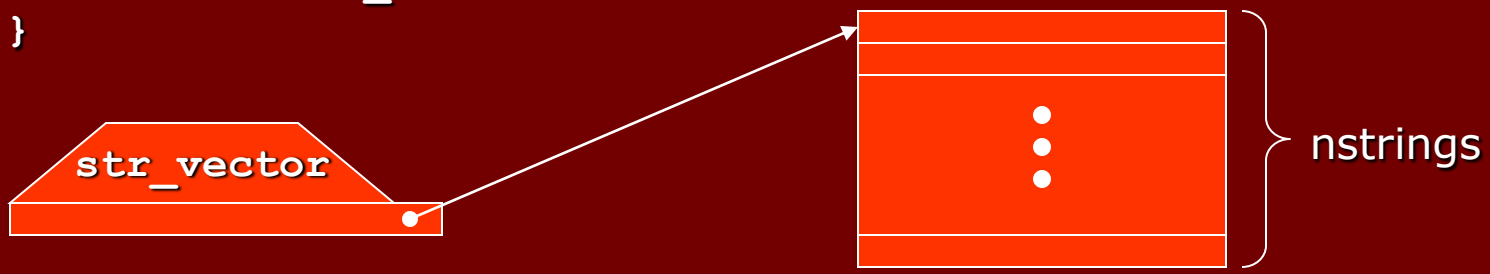
How Much Space Is Needed? - 2

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

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




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

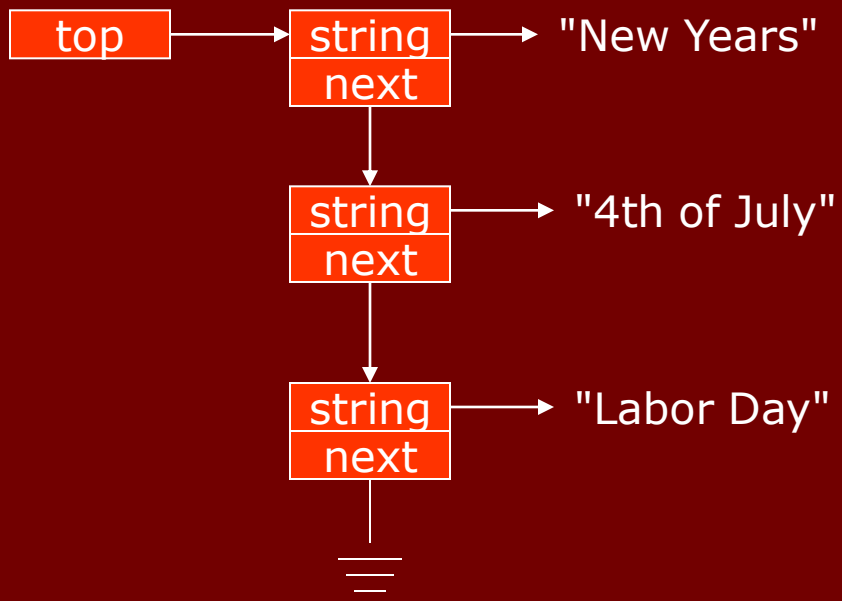


Linked Lists

- 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 ;
```



Implementation (String Stack) – I

```
static node *top = NULL ;

void push(char *s) {
    node *np = (node *) malloc( sizeof(node) ) ;
    np->string = (char *) malloc( strlen(s) + 1 ) ;
    (void) strcpy( np->string, s ) ;
    np->next = top ;
    top = np ;
}

void pop(char *s) {
    node *np = top ;
    if ( np == NULL )
        return ;
    top = np->next ;
    (void) strcpy( s, np->string ) ;
    free(np->string) ;
    free(np);
}
```

```
bool is_emtpy() {
    return top == NULL ;
}
```

Implementation (String Stack) – II

```
static node *top = NULL ;

void push(char *s) {
    node *np = (node *) malloc( sizeof(node) ) ;
    np->string = (char *) malloc( strlen(s) + 1 ) ;
    (void) strcpy( np->string, s ) ;
    np->next = top ;
    top = np ;
}

char *pop() {
    node *np = top ;
    char *top_string = NULL ;
    if ( np != NULL ) {
        top = np->next ;
        top_string = np->string ;
        free(np);
    }
    return top_string ;
}
```

```
bool is_emtpy() {
    return top == NULL ;
}
```