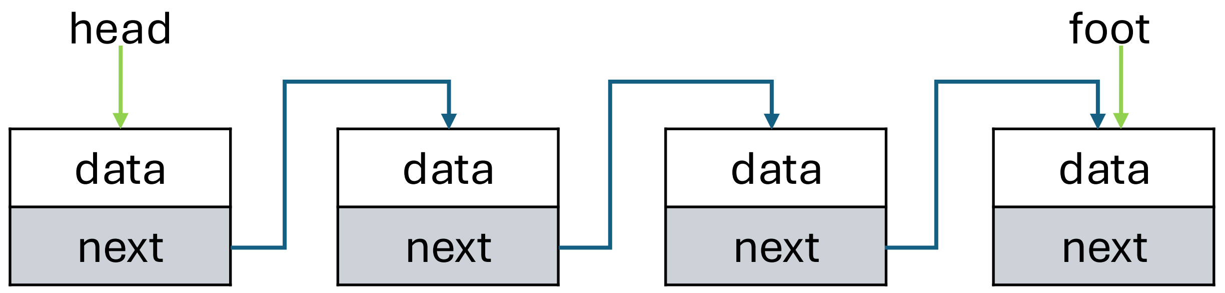
Linked List

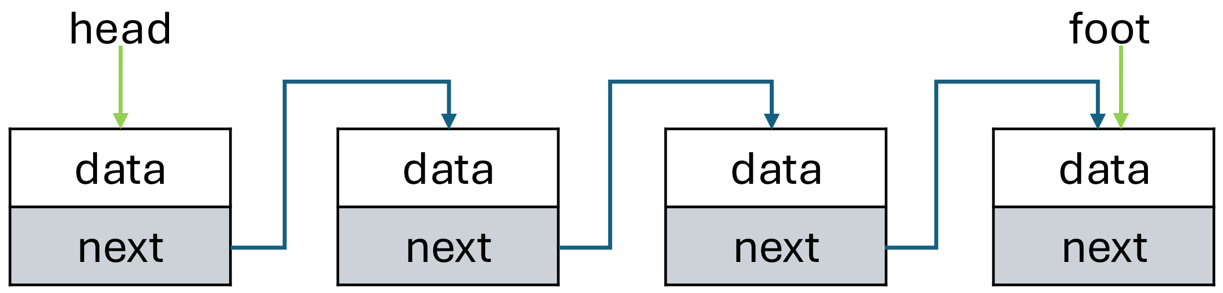
- A linked list is a one-dimensional data structure where each item is connected to the next one using a pointer in each node.

```
typedef struct node node_t
struct node{
    data_t data;
    node_t *next;
}
```



Linked List

- A linked list is a one-dimensional data structure where each item is connected to the next one using a pointer in each node.

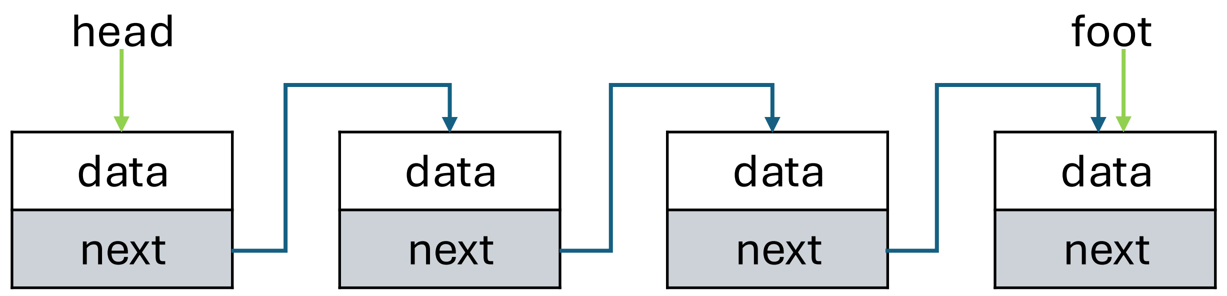


```
typedef struct node node_t
struct node{
    data_t data;
    node_t *next;
}

typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```

Linked List

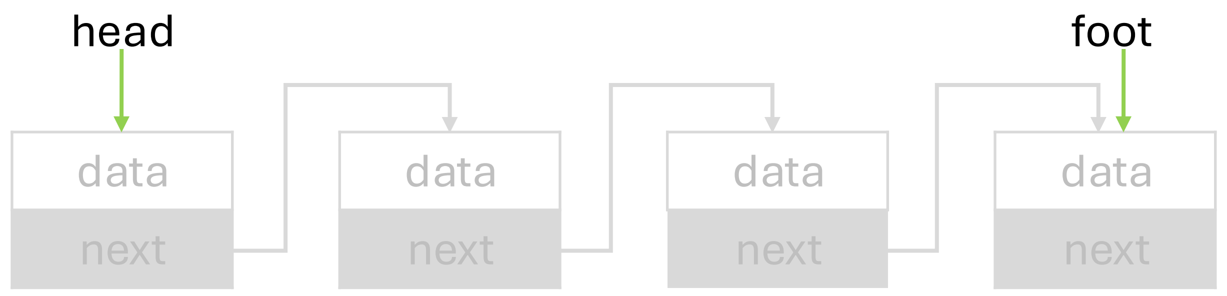
- Creating an empty list



Linked List

- Creating an empty list

```
typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```



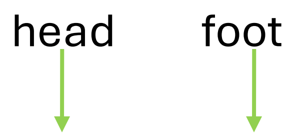
Linked List

- Creating an empty list
  (just 2 pointers – head and foot)

```
typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```

Linked List

- Creating an empty list
  (just 2 pointers – head and foot)



```
typedef struct node node_t
struct node{
    data_t data;
    node_t *next;
}

typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```

```
list_t
*make_empty_list(void) {
    list_t *list;
    list = (list_t*)malloc(sizeof(*list));
    assert(list!=NULL);
    list->head = list->foot = NULL;
    return list;
}
```

Linked List

- Creating an empty list
  (just 2 pointers – head and foot)

head foot

```
typedef struct node node_t
struct node{
    data_t data;
    node_t *next;
}

typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```

```
list_t
*make_empty_list(void) {
    list_t *list;
    list = (list_t*)malloc(sizeof(*list));
    assert(list!=NULL);
    list->head = list->foot = NULL;
    return list;
}
```

1. create an (empty) pointer to an object of the type list_t

Linked List

- Creating an empty list
  (just 2 pointers – head and foot)

head foot

```
typedef struct node node_t
struct node{
    data_t data;
    node_t *next;
}

typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```

```
list_t
*make_empty_list(void) {
    list_t *list;
    list = (list_t*)malloc(sizeof(*list));
    assert(list!=NULL);
    list->head = list->foot = NULL;
    return list;
}
```

1. create an (empty) pointer to an object of the type list_t
2. request enough memory to hold a list structure

Linked List

- Creating an empty list
  (just 2 pointers – head and foot)

head

foot

```c
typedef struct node node_t
struct node{
    data_t data;
    node_t *next;
}

typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```

```c
list_t
*make_empty_list(void) {
    list_t *list;
    list = (list_t*)malloc(sizeof(*list));
    assert(list!=NULL);
    list->head = list->foot = NULL;
    return list;
}
```

1. create an (empty) pointer to an object of the type list_t
2. request enough memory to hold a list structure (this returns a pointer to the address of the allocated memory of type void)

Linked List

- Creating an empty list
  (just 2 pointers – head and foot)

head foot

```
typedef struct node node_t
struct node{
    data_t data;
    node_t *next;
}

typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```

```
list_t
*make_empty_list(void) {
    list_t *list;
    list = (list_t*)malloc(sizeof(*list));
    assert(list!=NULL);
    list->head = list->foot = NULL;
    return list;
}
```

1. create an (empty) pointer to an object of the type list_t
2. request enough memory to hold a list structure (this returns a pointer to the address of the allocated memory of type void)
3. Make sure that this pointer is of type list type

Linked List

- Creating an empty list
  (just 2 pointers – head and foot)

head foot

```
typedef struct node node_t
struct node{
    data_t data;
    node_t *next;
}

typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```

```
list_t
*make_empty_list(void) {
    list_t *list;
    list = (list_t*)malloc(sizeof(*list));
    assert(list!=NULL);
    list->head = list->foot = NULL;
    return list;
}
```

1. create an (empty) pointer to an object of the type list_t
2. request enough memory to hold a list structure (this returns a pointer to the address of the allocated memory of type void)
3. Make sure that this pointer is of type list type
4. Assign the pointer list to the address of the newly allocated memory

Linked List

➢ Creating an empty list
(just 2 pointers – head and foot)

head foot

```
typedef struct node node_t
struct node{
    data_t data;
    node_t *next;
}

typedef struct {
        node_t *head;
        node_t *foot;
} list_t;
```

```
list_t
*make_empty_list(void) {
    list_t *list;
    list = (list_t*)malloc(sizeof(*list));
    assert(list!=NULL);
    list->head = list->foot = NULL;
    return list;
}
```

1. create an (empty) pointer to an object of the type list_t

2. request enough memory to hold a list structure (this returns a pointer to the address of the allocated memory of type void)

3. Make sure that this pointer is of type list type

4. Assign the pointer list to the address of the newly allocated memory

5. Double-check whether the memory allocation worked correctly