Topic 9 – Pointers – P2

Creating linked lists

Chapter 12 in the online book

Pointers and Structs

You can have pointers in structs

Remember we use the dot operator to access the struct members

A struct without pointers

```
struct Person
{
      string name;
      int    age;
};

Person per;

pers.name = "Fred";
per.age     = 21;
```

A struct with pointers

```
struct PersonNode
{
      string       name;
      int          age;
      PersonNode *next;
};

PersonNode *perPtr;

perPtr          = new PersonNode;
(*perPtr).name = "Fred";
(*perPtr).age    = 21;
```

Note: You need the ()

-> operator

(*perPtr).name = “Fred”;

This notation is awkward

Instead, we can use the -> operator when using pointers to structs

These are equivalent

perPtr -> name = “Fred”;
perPtr -> age = 20;

NOTE:
Use the . Dot operator with a simple struct variable
Use the -> operator with a pointer to a struct

What is a Linked List?

- A linked list is a chain of nodes linked to each other
- The linked list starts with a pointer to the first node → this pointer is called the head
  - We need to keep track of the first node in our list
- The last node on the list points to NULL

What is a Node?

- A node can be a struct with at least 2 members
  - Data (can be any data type) → can be more than 2
  - A pointer to another node (of the same struct type)

EXAMPLE

```
struct PersonNode
{
   string          name;
   PersonNode *next;
};
```

5

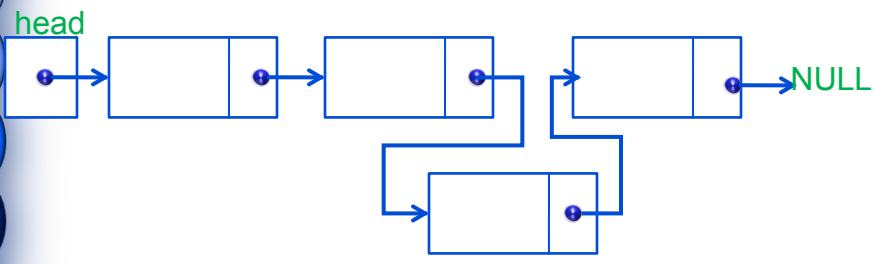
Why use Linked Lists?

- Arrays are great but have limitations
- They have a fixed size, so we have to guess a maximum
  - → Once full they cannot expand
  - → if not full, there is wasted memory space
- Link lists can have 0 or more nodes
  - → what you need as you need it!
- Have the ability to expand one item at a time
  - Have minimum size at all times

➔ optimal use of memory

Linked Lists vs. Arrays

- Linked lists can grow and shrink as needed, unlike arrays, which have a fixed size
- insertion or removal in the middle of the list is very efficient



How do we keep track of our list?

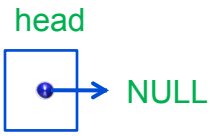
- The head node points to the first node in the list
- The first node points to the 2nd node (and so on)
- We know we've reached the end when we find NULL
  - The last node points to NULL

➔ We use head and the value NULL to access all the elements in the list

Empty List

A list with no nodes is called the empty list

In this case the list head is set to NULL



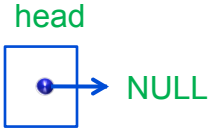
NOTE:
The head is just a pointer to the first struct (or object)

Creating an Empty List

- Define a pointer for the head of the list
- Then initialized it to NULL to indicate an empty list:

```
PersonNode *head;
head = NULL;
```

Remember NULL is a predefined constant



Basic Process for Linked Lists

Once we've created our empty list we can start adding to our linked list

1. Create a new node
2. Fill the data field(s)
3. Link the new node to the head of the list
4. Point the head to the new node

1. Creating a New Node
2. Filling in Data Fields

```
PersonNode *perPtr;
perPtr          = new PersonNode;
perPtr->name = "John Doe";
```

This creates our new node and then assigns "John Doe" to our name.

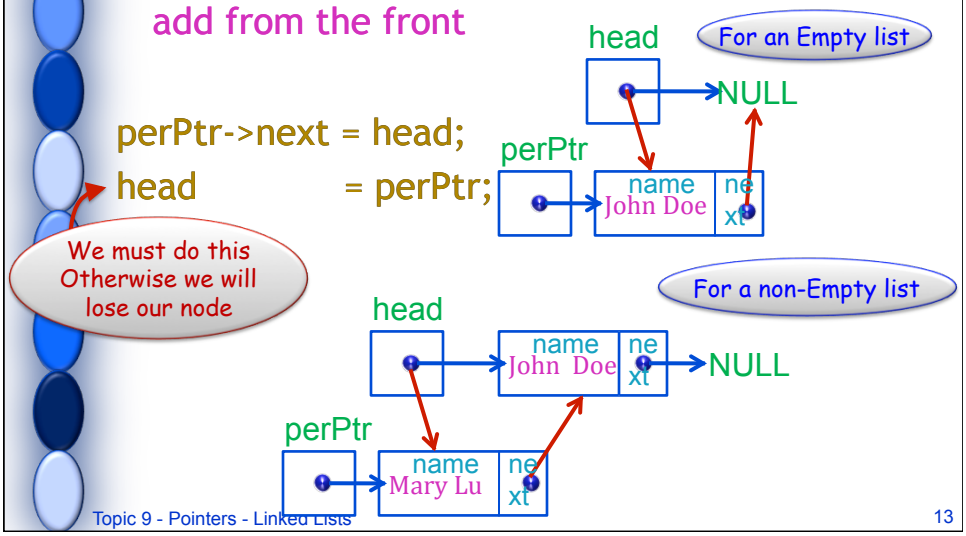
Now, we have to add it to our list

3. Link the New Node to the List
4. Point the Head to the New Node

- The easiest/fastest way to add to a list is to add from the front

perPtr->next = head;

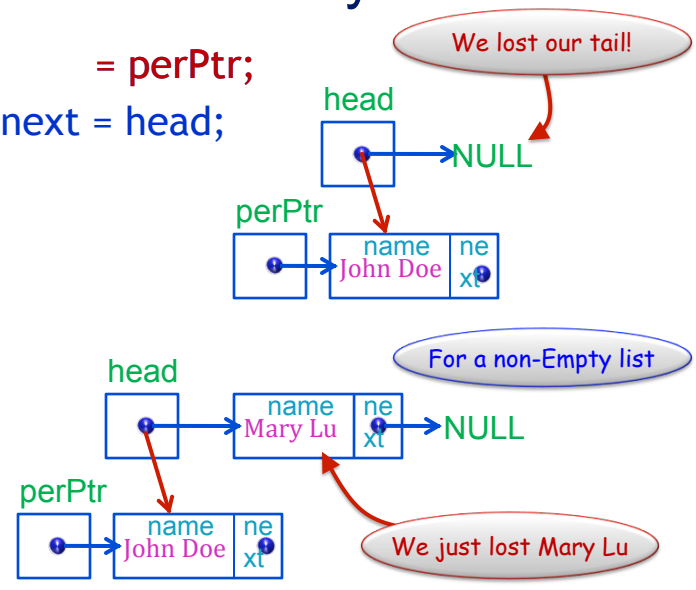
head = perPtr;



Careful not to lose your head!

head = perPtr;

perPtr->next = head;



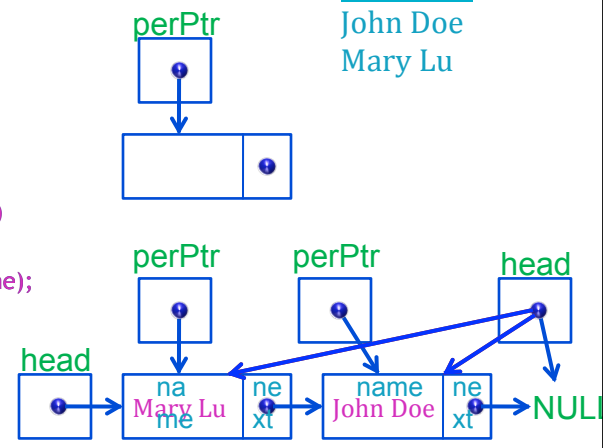
Example

What if we want to read a list of name in from a file and add it into our linked list?

```
structPersonNode
{
      string          name;
      PersonNode *next;
};
PersonNode *head = NULL;
PersonNode *perPtr;
perPtr = new PersonNode;

while(inFile && perPtr != NULL)
{
      getline(inFile, perPtr->name);
      perPtr->next = head;
      head   = perPtr;
      perPtr = new PersonNode;
}
delete perPtr;
```

Input File
John Doe
Mary Lu



15

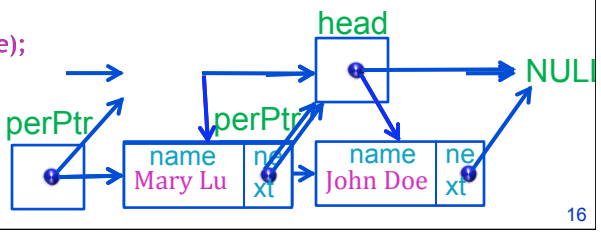
Example

What if we want to read a list of name in from a file and add it into our linked list?

```
struct PersonNode
{
      string name;
      PersonNode *next;
};
PersonNode *head;
PersonNode *perPtr;
head   = NULL;
perPtr = new PersonNode;

while(inFile && perPtr != NULL)
{
      getline(inFile, perPtr->name);
      perPtr->next = head;
      head   = perPtr;
      perPtr = new PersonNode;
}
delete perPtr;
```

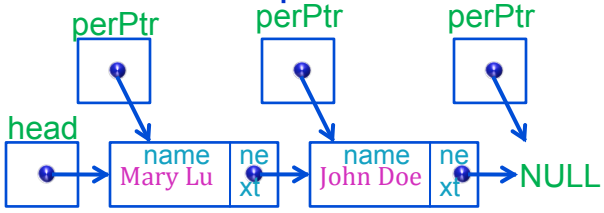
Input File
John Doe
Mary Lu



16

Output a list

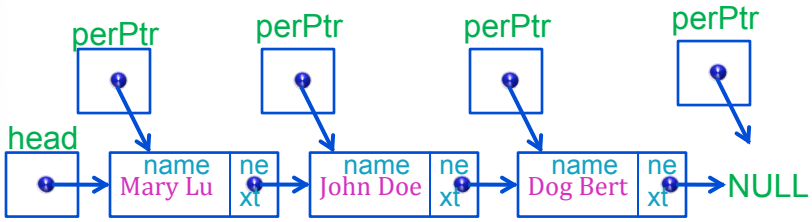
- How would we output a linked-list?



- Why does it output in reverse order?
  - Compared to the original input file.

Searching a linked-list

- How do we search a linked-list?




```
string key;
key = "Dog Bert";
```

What if the name is not in the list?